arXiv:2302.00732v2 [cs.CR] 5 Jun 2023

防御推测执行攻击和侧信道攻击的缓存状态保护

Guangyuan Hu 和 Ruby B. Lee

普林斯顿大学

{gh9,rblee}@princeton.edu

摘要

硬件缓存是现代处理器中至关重要的性能优化功能，可缩短有效内存访问时间。遗憾的是，它们也是针对计算机处理器发起攻击的主要目标，因为它们是高带宽且可靠的侧信道或隐蔽信道，可用于泄露秘密。传统的缓存时序攻击通常会泄露加密密钥，而近期的推测执行攻击通常会通过缓存时序通道泄露任意非法获取的秘密。尽管针对每类攻击，人们都提出了许多硬件防御措施，但我们指出，针对传统（非推测执行）缓存时序通道的防御机制无法抵御所有推测执行攻击，反之亦然。我们认为，除非缓存能够防御这两种主要攻击类别，否则就不安全。

我们提出了一种新的方法论和框架，用于覆盖这类相对广泛的攻击面，从而构建具有推测执行与定时攻击韧性（STAR）的缓存子系统。我们以此设计了两种全面的安全缓存架构：STAR-FARR 和 STAR-NEWS，其性能开销分别低至 5.6% 和 6.8%。据我们所知，这是首批同时防范非推测性缓存侧信道和基于缓存的推测执行攻击的安全缓存设计。

我们的方法论可用于组合并检查其他安全缓存设计，也可扩展到其他攻击类别和硬件系统。此外，我们还强调了随机化缓存（例如真正的随机替换全相联缓存（FARR））以及低延迟、可感知推测执行的版本（NEWS）所固有的安全性与性能收益。

1 引言

基于定时的侧信道攻击通过观察访问与受害者共享资源时的时序差异来恢复秘密。缓存命中与缓存未命中之间存在较大的时序差异，因此缓存常被利用来泄露秘密信息。在近期的推测执行攻击中，也会使用基于缓存的隐蔽信道，泄露在错误预测的执行路径上进行瞬态执行时获取的秘密。

本文聚焦于阻止利用微体系结构缓存状态泄露秘密信息的攻击。尽管还有许多其他类型的微体系结构状态可被用作定时信道，但缓存状态是最常被利用的定时信道之一，因为缓存在现代计算机中无处不在，具有明显的时序差异和高带宽，而且其状态会持续存在，无法快速清除。

由于非推测执行缓存侧信道和基于推测执行的缓存攻击都针对硬件微体系结构，因此硬件微体系结构解决方案更为理想。尽管已有许多硬件防御机制被提出，以防止缓存状态中的信息泄漏，但它们只针对其中一种攻击类别，而不能同时防御两者。我们提出首个全面的安全缓存架构——推测与定时攻击弹性（STAR）缓存，可抵御这两种攻击类别，且性能开销较低。

我们提出了一种用于分析各种防御机制安全性的方法论。我们的关键洞见是，在推测执行攻击中，发送方和接收方可能处于同一安全域，因此基于安全域的防御机制对此类攻击无效。另一方面，传统侧信道攻击假设发送方和接收方处于不同的安全域。我们的分析表明，侧信道防御机制和推测执行防御机制都无法全面防御另一类攻击。

随后，我们提出硬件安全特性，以构建全面的防御机制。我们证明，全相联缓存采用随机替换策略（FARR）可以用于防止基于竞争（基于未命中）的攻击。通过标记缓存行所属安全域，可以缓解跨域基于命中的攻击。我们还发现了同域、基于命中的推测执行攻击，并证明可以通过一种新的“推测性地填充缓存，但在冲刷流水线时失效（***SFill-Inv***）”防御措施来抵御这类攻击。与其他覆盖

推测性缓存通道。

我们进一步表明，可以用一种新的推测感知缓存替换全相联缓存，在维持相同安全特性的同时降低访问延迟或功耗。NewCache[16,68] 是一种随机化缓存，每次缓存未命中时都会进行动态重映射。该缓存旨在抵御基于争用的侧信道攻击，且 15 年来未被攻破 [8,51,61]。我们展示了如何借鉴 NewCache，设计一种采用推测感知、低延迟随机化缓存的全面防御措施，从而同时抵御推测攻击。

我们的主要贡献如下：

• 展示涵盖缓存侧信道攻击和基于缓存的推测执行攻击的攻击空间。识别发送方和隐蔽信道接收方位于同一安全域内的推测攻击。指出现有防御措施均未能全面应对这两类攻击。

• 提出一种方法论，用于分析硬件安全特性对攻击空间的覆盖情况。此外，识别出性能开销显著的防御特性（表1）。

• 提出一种新的防御功能，可在不常见的冲刷流水线路径上，以低成本失效机制支持推测性缓存填充。

• 设计一种能够感知推测执行、延迟低、支持动态重映射的随机化缓存，其灵感源自先前的 NewCache 设计 [42,68]。

• 设计两种推测与定时攻击弹性（STAR）缓存架构——STAR-FARR 和 STAR-NEWS，分别以 5.6% 和 6.8% 的低性能开销抵御所有纳入考量的缓存时序攻击。

2 背景

2.1 缓存时序侧信道攻击

缓存侧信道攻击会观察缓存访问时间，并推断受害者使用的地址。

基于命中的缓存侧信道允许攻击者通过观察缓存命中来推断受害者的信息。基于命中的侧信道的一个常见示例是使用冲刷-重载技术，例如， [75], 其步骤如下：

1.（冲刷）攻击者首先将共享内存区域的所有缓存行从缓存中冲刷出去。2. 受害者使用依赖于秘密的地址访问其中一个缓存行。3.（重载）攻击者重新加载所有缓存行，并在受害者刚刚访问的地址处发现缓存命中。由此可以从该地址恢复秘密。

flush-reload缓存侧信道是一种有效的技术，只要依赖于秘密的地址位于缓存中，就能恢复秘密。尽管第一步在 x86 架构中，刷新通过 clflush 指令实现；在其他架构中，也可以通过引发冲突以驱逐相关缓存行来实现，从而导致 Evict+Reload 侧信道的一种类似变体。

发送方代码：

if (x < array_size) // 授权
y = array[x] // 访问
z = 共享**[y*4096]** // 使用并发送

接收方代码：
对于 i，从 0 到 255 // 接收 **t[i]** = 重新加载耗时(shared**[i*4096]**) 找出最小的 **t[i]**

图 1：Spectre-v1 攻击的代码。分支预测器的误训练以及对共享 所有缓存块的冲刷，发生在发送方代码执行之前，因此未在图中显示。

基于缺失的缓存侧信道 [5,43,53], 会使受害者依赖秘密的访问驱逐攻击者的缓存行。随后，攻击者可以通过观察访问先前缓存行时哪些位置发生缓存未命中，来恢复秘密值。利用缓存争用的常见 Prime+Probe 攻击 [50]分为三个步骤：

1.（填充）攻击者用自己的缓存行填满缓存。2. 受害者访问依赖秘密的地址，从而驱逐攻击者的一条缓存行。3.（探测）攻击者再次访问自己的缓存行，并在受害者访问过的缓存组中发现缓存未命中。被驱逐地址的信息由此泄露。

基于缺失的缓存侧信道攻击无需受害者与攻击者之间存在共享内存即可成功，因此比某些基于命中的侧信道攻击更具实用性。不过，如果存在其他内存访问并引入无关的缓存冲突，基于缺失的缓存侧信道攻击的有效性就会降低。

2.2 推测执行攻击

推测执行攻击或瞬态执行攻击是微妙的微架构攻击，会泄露软件本不应访问的秘密。研究工作 [10,12,21,23,72] 系统地描述了这些推测执行攻击。推测执行攻击主要包含两个操作：非法访问秘密，以及通过隐蔽信道将秘密传出。 [21,23] 进一步将攻击分解为六个关键步骤：设置微架构状态，授权以绕过软件授权，访问以进行推测性秘密访问，使用秘密，发送和接收以进行隐蔽发送与接收，其中可使用缓存状态。

如果推测执行有误，即访问未获授权，硬件微体系结构会撤销推测执行的指令，从而使访问、使用和发送操作失效。然而，由于微架构变化（例如缓存状态变化）理应对软件不可见，因而不会被撤销。这些变化是不可见的——

——因此，在错误推测被冲刷流水线后，发送操作导致的缓存状态变化仍可能可见，进而通过微架构隐蔽通道泄露秘密。

推测执行攻击的变体 [4, 11, 22, 25–31, 35–37, 40, 46, 54, 59, 60, 63–66, 70] 利用不同的漏洞，在访问得到验证或授权之前，短暂地非法访问秘密数据。根据所利用的漏洞，秘密数据可能来自特殊寄存器、缓存、内存或其他微架构缓冲区。之后，所有这些漏洞都可以利用缓存通道传送秘密数据。

图1展示了一个使用冲刷-重载缓存通道的 Spectre-v1 攻击示例。发送方代码通过推测执行将越界秘密数据读入 y，并执行隐蔽发送，即使 x 大于数组大小也是如此。设置步骤（针对非法秘密访问进行误训练，并通过冲刷来设置隐蔽信道）在发送方代码执行之前完成，图中未显示。

3 动机与威胁模型

缓存是关键的泄漏通道。还有其他硬件单元可用作非推测攻击和推测攻击的微架构通道，例如执行端口 [6], 未命中状态保持寄存器（MSHR）的使用情况 [4] 以及分支预测器状态 [17, 69]。然而，缓存状态具有以下优势，使其成为可靠且常用的通道。

可区分的时序差异。时序上存在明显差异，例如，加载指令在 L1 缓存命中时需要几个周期，而在缓存未命中、必须访问主内存时则需要数百个周期。这使缓存定时信道既可靠又快速。

持久性。程序结束或终止后，缓存中仍会保留缓存行。端口竞争和未命中状态保持寄存器的使用情况等状态，可以通过不调度并发进程来加以保护。然而，攻击程序可以在受害者程序结束后测量缓存行。

清除成本高。分支预测器等核内硬件状态可以被擦除或屏蔽，例如在程序结束或发生上下文切换后，执行间接分支预测屏障 [2]。然而，缓存是一种具有多个缓存层级和大量缓存行的硬件单元。因此，在硬件中选择并清除受害者的所有缓存行是一项困难且成本高昂的操作。

泄漏带宽高。与只能编码 1 位信息的硬件单元用法相比，用于访问缓存的地址可以编码更多位。图 1 展示了泄漏 8 位信息的示例。通过使用更大的共享数组，还可以泄漏更多位。

威胁模型。我们的威胁模型考虑这样一种攻击者：攻击者可以通过引发依赖秘密的缓存状态变化来泄露秘密。适用范围包括受害者以非推测方式执行时发生的缓存侧信道攻击，以及基于缓存的推测执行攻击。具体而言，我们保护缓存行的地址和缓存替换状态。

在推测执行攻击的范围内，如果加载执行时，并非所有先前指令都已完成执行且未发生故障，我们就认为该加载是推测性的。这涵盖了使用不同技术进行的所有形式的恶意推测执行，例如预测器误训练、恶意跳转目标注入、推测性存储转发至加载，或延迟异常或故障。我们假设攻击者是强攻击者，即使受害者自身的行为也能触发推测执行 [60]。

我们保护秘密，使其不受内存、缓存、寄存器、微架构缓冲区及其他单元的影响；也就是说，内存访问地址的计算方式不会影响这种保护。我们假设攻击者能够进行计时测量，以检测其缓存行是否存在于不同缓存层级中。我们的解决方案专注于保护靠近处理器核心、且容易遭受缓存时序通道攻击的一级（L1）数据缓存。使用加密组索引等技术来保护共享末级缓存的工作 [52, 53, 71] 与我们的工作并行进行。

我们不防范因软件漏洞或恶意代码注入而导致的安全问题，因为这类问题会赋予攻击者合法访问秘密的权限。我们假设系统能够自动为软件程序分配安全域，或根据程序的请求进行分配。对安全敏感的程序应分配一个唯一的域标识符。安全域 ID 并不等同于进程标识符。一个进程可以拥有多个安全域，一个安全域也可以包含多个进程。

我们允许并保护不同安全域之间共享只读内存区域，以用于共享软件库等用途 [19]。在安全域之间共享可写页面是危险的，因为这会导致信息直接泄露。

我们不处理物理攻击或电路级故障，例如 Rowhammer 类攻击 [33, 47]。

4 方法论

我们提出了一种方法论，用于研究需要防御的攻击范围（第 4.1 节）、哪些机制能够抵御哪些攻击（第 4.2 节），以及会产生多少性能开销（第 4.3 节）。安全分析揭示的关键一点是：侧信道防御机制无法缓解在同一安全域内进行隐蔽发送与接收的推测执行攻击。另一方面，针对推测执行攻击的防御机制无法保护受害者在非推测性缓存侧信道攻击中，经授权执行所造成的缓存状态变化。我们还展示了对导致性能开销的硬件事件的观察结果。

硬件特性	安全性：缓存攻击的覆盖范围						对性能的影响					
	NS-CD-FR	NS-CD-PP	S-CD-FR	S-CD-PP	S-SD-FR	S-SD-PP	暂停推测执行	禁止正确路径缓存填充	周期性移动缓存行	额外访问所需	由于停顿额外访问	额外缓存延迟
1. 缓存分区		√		√			No	No	No	No	No	No
2. 域间无命中（由软件定义）	√	√	√	√			No	No	No	No	No	命中筛选
3. 具有域专属 基于密钥的重新映射，例如 CEASER		√*		√*		√*	No	No	Yes	No	No	加密
4. 缓存未命中时动态重映射， 例如 NewCache		√		√			No	No	No	No	No	小型映射表
5. 防止推测执行的特殊栅栏 缓存填充			√	√	√	√	No	Yes	No	No	No	No
6. 延迟推测性数据访问			√src	√src	√src	√src	Yes	延迟至已授权	No	No	No	No
7. 延迟推测性数据转发			√src	√src	√src	√src	Yes	延迟至已授权	No	No	No	No
8. 延迟推测性缓存填充			√	√	√	√	Yes	延迟至已授权	No	No	No	No
9. 缓冲区并重做推测性缓存填充			√	√	√	√	No	延迟至已授权	No	对于正确路径（往返）	停顿直至已返回	No
10. 撤销推测性缓存驱逐 并填充			√	√	√	√	No	No	No	对于错误路径（往返）	停顿，直到已返回	No
新型防御机制												
用于防止跨域 命中的域标记缓存（NoHit）	√		√				No	No	No	No	No	No
随机全相联缓存 替换的全相联缓存（FARR）		√		√		√	No	No	No	No	No	大型映射表
具备推测执行韧性的 NewCache （NEWS）		√		√		√	No	在标签时 miss	No	No	No	小型映射表
推测性缓存填充 + 失效 （SFill-Inv）					√		No	No	No	对于错误路径（一种方法）	停顿，直到发出	No
综合防御												
STAR-FARR (SFill-Inv + 否NoHit + FARR)	√	√	√	√	√	√	No	No	No	对于错误路径(一种方式)	停顿直到发出	大型映射表
STAR-NEWS (SFill-Inv + 否NoHit + NEWS)	√	√	√	√	√	√	No	遇到标签时 miss	No	对于错误路径（一种方式）	停顿，直到发出	小型映射表

表1：针对不同缓存时序攻击的防御机制。NS 表示非推测侧信道攻击，S 表示推测执行攻击。CD 表示跨域攻击，SD 表示同域攻击。FR 表示 flush-reload 缓存通道，PP 表示预取-探测缓存通道。 ✓* 表示安全性依赖于设计参数。$✓_{src}$ 表示该防御措施保护某些秘密来源，通常是缓存和内存。

4.1 攻击的相似之处与差异

缓存时序攻击包括两类：非推测（NS）执行攻击和推测执行（S）攻击。攻击还可从两个维度进一步分类：基于命中（以冲刷-重载攻击（FR）为例）与基于未命中（以探测-探测攻击（PP）为例），以及跨域（CD）与同域（SD）。由此形成表格各列所示的 6 类攻击1。

非推测执行和推测执行这两类攻击都使用基于命中或基于未命中的通道。缓存状态的准备和测量方式相同（参见第2节）。状态变化由受害者的加载操作引起，该操作可以是非推测执行，也可以是推测执行。

非推测侧信道攻击与推测执行攻击之间的一个关键区别在于，发送方和接收方可以处于同一域，也可以处于不同域。对于非推测的传统缓存侧信道，攻击者和受害者处于不同的安全域中。拥有合法权限的同域攻击者可以直接访问并泄露秘密，而不必利用侧信道。在推测执行攻击中，发送方和接收方可以处于同一域，也可以处于不同域。同域攻击更加危险，而且已被证明切实可行 [24, 46]。

4.2 以往的硬件防御机制分析

表1 展示了各项硬件特性对哪些基于缓存的攻击具有缓解作用的安全分析。第 1 至 10 行列出的是以往工作中由硬件防御机制实现的特性。各列对应六种缓存时序攻击（参见第4.1节）。表中的勾号表示相应的硬件特性能够防止对应的攻击。

侧信道防御。 针对缓存侧信道攻击的安全缓存主要有两个方向。第一个方向是在安全域之间划分缓存资源 [9, 15, 41, 49, 67]（第 1 行）。如果系统能够区分受害者和攻击者，这种方法可以防止通过依赖秘密的缓存驱逐造成的泄漏。先进设计，例如

DAWG [34] 和 Hybcache [14], 实现了过滤缓存命中的机制（第 2 行），从而也防止基于跨域命中的攻击。

第二种思路是随机化缓存变化，使攻击者无法获得有用信息。第 3 行中的基于密钥的重映射，例如 CEASER [52] 和 ScatterCache [71],使用域专用密钥来随机化缓存行放置。为了防止基于驱逐集的攻击 [43,53], , 需要定期更换密钥，这要求移动现有缓存行以适应新的映射。重新加密周期可能会影响其抵御同域探测式推测执行攻击的安全性（✓* 见表 1） [8,51,61]。

第 4 行中的动态重映射缓存，例如 RPCache [67]和 NewCache [42, 68], , 会在基于争用的驱逐时随机替换缓存行，因此攻击者无法获得有关受害者所用缓存行的任何有用信息。它们可以缓解所有基于未命中的缓存侧信道，包括通过测量整个操作所需时间（例如加密整个数据块）而非单次内存访问所需时间来实施的攻击。由于这些攻击（例如驱逐时间攻击 [5], ）要慢得多，我们在此不予考虑。NewCache [42] 使用“域 ID”和“P 位”这两个术语来定义不同缓存行所属的安全域。动态重映射实际上会在每次缓存未命中时进行重映射，并且近期关于随机化缓存的研究尚未攻破这种机制 [8,51,61]。我们研究它能否作为一种全面解决方案，同时应对缓存侧信道攻击和基于缓存的推测执行攻击。

过去，动态重映射主要应用于 L1 缓存，而基于密钥的重映射则应用于末级缓存（LLC）。我们设想，随着伪全相联末级缓存 [56] 和高性能硬件查找表 [45]设计的出现，未来这两种技术都可用于任意缓存级别。

我们注意到，侧信道防御无法防御同域推测执行攻击。

针对推测执行攻击的防御措施。 针对推测执行攻击的防御机制可分为：阻止微体系结构状态的设置 、阻止对秘密的访问、阻止使用 秘密，以及阻止发送 操作改变状态 [21]。

防止恶意设置，例如通过加密或清除预测器状态 [2, 20, 62], , 并不能阻止非推测执行缓存侧信道攻击，以及能够从同一域触发推测执行的强攻击者。我们未将其列入表1。

一种防御措施可以延迟推测执行，直到获得授权。表 1 的第 6 行列出了延迟推测性秘密访问的机制，例如上下文敏感栅栏 [62]和安全边界检查 [48]。第 7 行分析了延迟秘密转发（使用）的防御措施，例如 NDA [69], SpectreGuard [18], ConTExT [58], SpecShield [3] 和 STT [76]。这些防御机制可以缓解多种隐蔽信道，但只能保护特定访问，例如内存加载或

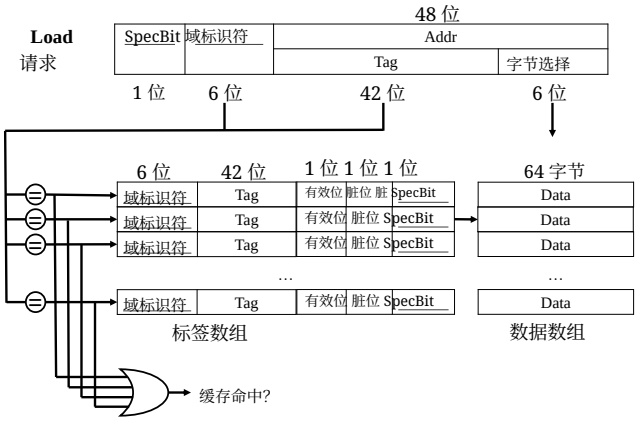

图2：STAR-FARR 的架构。域标识符和 SpecBit 附加在内存请求和缓存行上，用于指示其所有者和推测执行状态。

特殊寄存器读取。

对于缓存状态，可以插入特殊栅栏（第 5 行），使推测加载无法缓存，以避免获取新的缓存行 [62]。一种防御措施可以仅延迟发生缓存未命中的推测加载（第 8 行），例如 Conditional Speculation [39],高效推测 [57] 和 DOLMA [44]。也可以将推测执行产生的缓存行放入特殊缓冲区（第 9 行），并在其获得授权后使其可见，例如 InvisiSpec [73], SafeSpec [32] 和 MuonTrap [1]。

为了提升性能，利用良性程序中的大多数指令都不会被冲刷流水线这一观察，防御措施也可以允许推测性缓存填充，并在执行被冲刷流水线时恢复缓存状态（第 10 行）。对于每个被冲刷流水线的加载操作，CleanupSpec [55],会移除推测性获取的缓存行，并重新获取被驱逐的缓存行。然而，CleanupSpec 必须等待恢复完成。研究发现，等待时间取决于秘密，并且可以通过 unXpec 攻击 [38]进行测量。

这些防御机制可以缓解来自同一域或不同域的基于缓存的推测执行攻击。然而，这些针对推测执行攻击的防御机制无法抵御非推测性侧信道攻击，因为受害者的执行不会被冲刷流水线。

要点。 我们表明，现有的缓存防御措施都无法同时应对同域推测执行攻击和非推测侧信道攻击。因此，我们的目标是设计一种安全缓存，以尽可能小的性能影响和硬件复杂度应对所有这些攻击。

4.3 性能开销的影响因素

我们确定了六种会导致表中防御措施造成性能下降的硬件事件1。良好的防御措施应在维持攻击覆盖范围的同时，尽量降低所有这些事件发生的可能性。

延迟正确路径执行。第 6 至第 8 行的防御机制会延迟推测指令的执行及相关缓存填充。由于正确路径执行更为频繁，这可能会造成严重的性能开销。

禁止正确路径缓存填充。缓存填充对性能至关重要。在允许推测性数据访问的同时，如果采用严格的威胁模型，第 5 行的填充阻止栅栏可能会禁止很大一部分缓存填充。

周期性移动缓存行。由于密钥化重映射防御（第 2 行）会改变地址到缓存行的映射，因此需要周期性移动现有缓存行。这是一项开销很大的硬件操作，需要检查现有缓存行的归属并进行操作。

需要额外访问。使用推测缓冲区的防御机制（第 9 行）需要进行第二次访问，以便在更频繁地执行正确路径时重做访问。这是一项缓存往返操作，需要访问该地址并将其重新载入缓存。填充并撤销防御（第 10 行）则需要同时使推测访问的地址失效，并在推测执行被冲刷流水线（错误路径）时重新填充被驱逐的地址。

基于撤销的防御机制在性能开销方面本质上可能更优，因为它们会在出现频率较低的错误推测路径上执行额外工作。不过，CleanupSpec 必须将被驱逐的缓存行恢复到 L1 缓存中，这一过程复杂且耗时。

因额外访问而停顿。使用推测缓冲区的防御机制（第 9 行）需要暂停提交推测指令，因为必须验证推测访问的数据是否有效。填充并恢复防御机制（第 10 行）也需要在恢复完成前暂停后续内存操作的执行。

额外缓存延迟。防御机制所需的硬件单元可能会增加缓存访问延迟，例如，用于随机化的加密操作以及大型全相联缓存。

5 STAR 缓存架构

5.1 概述

推测与定时攻击弹性（STAR）缓存是一种综合性防御机制，可抵御推测执行攻击和侧信道攻击。为应对表 1 中的非推测性缓存侧信道攻击，我们建议对基于竞争的 PP 攻击采用某种形式的随机化缓存，并通过安全域字段禁止 FR 攻击中的跨域缓存命中。这些措施涵盖了表 1中针对非推测执行和推测执行攻击的所有攻击类别

除了同域的推测性刷新重载攻击外。我们提出了一种低成本、高性能的解决方案来应对这一问题（*SFill-Inv*）。

我们提出以下四种新的防御特性和两种保护 L1 缓存的 STAR Cache 变体。这些

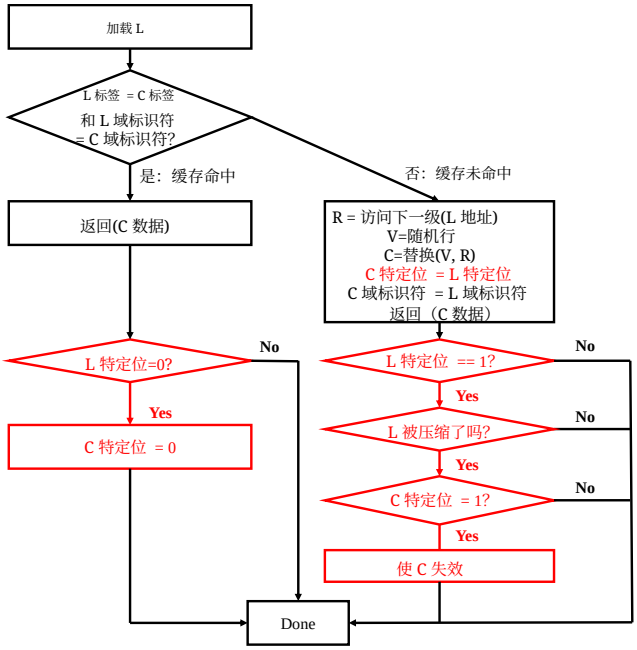

图 3：STAR-FARR 中的加载处理。用于处理推测执行攻击的修改以红色显示。

基于采用随机替换的全相联缓存（第 5.2 节中的 STAR-FARR）以及一种新型推测感知随机缓存（第 5.3 节中的 STAR-NEWS）。结果表明，它们能够抵御所有列出的攻击类型，并在表 1

用于防止跨域命中的域标记缓存（否-命中）。我们为缓存行以及未命中状态保持寄存器和写回缓冲区等内部缓冲区附加域标识符。来自不同域的内存访问无法在域标识符不同的缓存行中命中，从而挫败不同域之间的刷新重载攻击。尽管现有的软件定义命中过滤器 [34] 要求系统软件了解硬件配置，并手动为每个域分配资源，但域标记缓存只要求系统软件分配域标识符，缓存分配则由硬件自动完成。

采用随机替换策略的全相联缓存（*FARR*）。我们表明，作为一种基本缓存架构，*FARR* 能够防御基于未命中的缓存攻击。这是因为无论访问哪个地址，攻击者的缓存行被替换的概率都相同。*FARR* 还能防止通过缓存替换状态泄漏信息。

具备推测执行感知能力的 NewCache（*NEWS*）。NewCache [42,68]具有与*FARR*相同的安全性配置文件。FARR 在概念上更简单，但可能会增加访问延迟。研究表明，原始 NewCache 的访问时间与同等大小的组相联缓存相同 [16]。受 NewCache 启发，我们提出了一种具备推测执行感知能力、可动态

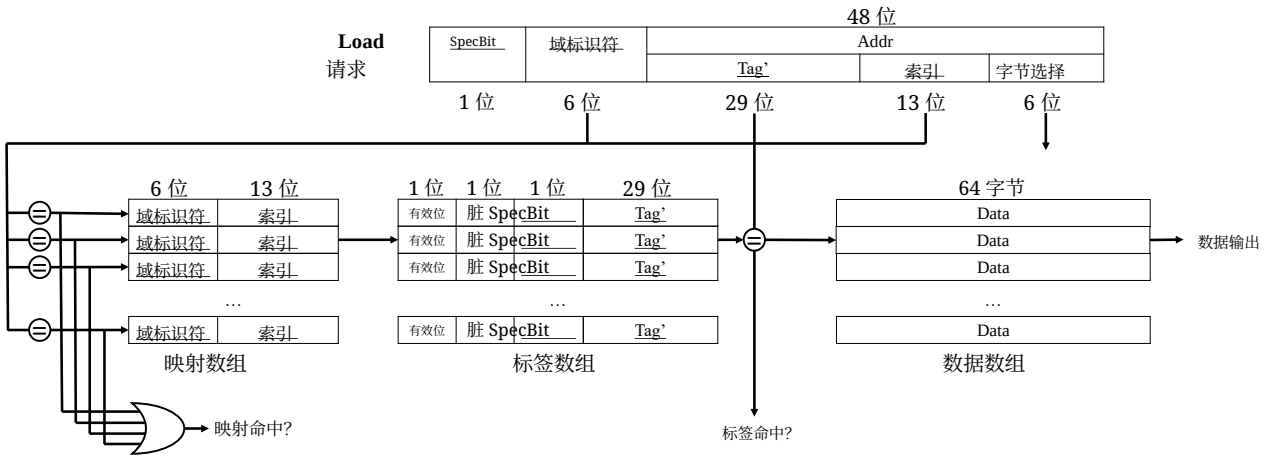

图 4：受 NewCache 启发的 STAR 缓存（STAR-NEWS）架构。加载请求和缓存行中新增及修改的字段以带下划线标示。对于一个 32kB、缓存行大小为 64 字节的 L1 缓存，其缓存行数为 $2^9 = 512$，因此索引宽度需要 9 位。通过增加 k 位（本例中 $k = 4$），加宽索引字段，相当于映射到一个容量大 2^k 倍的逻辑缓存。如果使用 48 位地址，修改后的 *索引* 字段有 $9 + 4 = 13$ 位，而 *标签'* 字段有 $42 - 13 = 29$ 位。

重新映射的缓存。我们在第 5.3 节介绍针对安全性的调整。**推测性缓存填充与冲刷流水线时失效（*SFill-Inv*）**。我们提出了 *SFill-Inv*，这是一种在冲刷流水线时使推测性获取的缓存行失效的新机制。这项新特性可以提升安全性，与 CleanupSpec 等先前的撤销型防御相比，还能降低性能开销 [55]。在安全性方面，*SFill-Inv* 无需处理器等待执行完成，从而消除了 CleanupSpec 中 unXpec [38] 攻击的根本原因，即可测量的恢复时间。在性能方面，它只在冲刷流水线时使缓存行失效，无需恢复被驱逐的缓存行。第 *SFill-Inv* 节介绍了该机制的详细内容，参见第 5.4 节。

5.2 STAR-FARR

STAR-FARR 实现了推测性缓存填充 + 失效（*SFill-Inv*），防止跨域命中（*NoHit*），并采用随机替换的全相联缓存（*FARR*）。STAR-FARR 可击败表1中的所有攻击列。

STAR-FARR 的硬件修改如图 2所示。加载请求会在 *SpecBit* 中记录其是否以推测方式执行，并将其视为不安全。存储操作只有在无法再被冲刷流水线时才会发出，因此存储操作的 SpecBit 始终为 0。请求中的 *DomainID* 字段和缓存行中的 DomainID 字段分别表示内存操作和缓存行所属的安全域。

图3 展示了 STAR-FARR 中的加载处理。前缀 *L.* 表示加载请求，而前缀 *C.* 表示缓存行。STAR-FARR 会检查标签和域标识符，以判断是否发生缓存命中。非推测性请求可以清除发生命中的缓存行的*SpecBit* 。即使未命中请求是推测性的，也可以随机替换旧缓存行。如果之后推测加载被冲刷流水线，则会发送一个 *SFill-Inv* 请求到缓存，以使该地址失效。

5.3 STAR-NEWS

在 STAR-FARR 中使用全相联缓存会增加访问延迟、电路规模和功耗。STAR-NEWS 通过采用具有推测执行韧性的增强型 NewCache（*NEWS*）替代全相联缓存，从而优化架构，降低访问延迟和功耗。

STAR-NEWS 架构如图所示4。加载请求和缓存行中新增或修改的字段以带下划线的项目表示。在类似 NewCache 的架构中，标签字段在 *FARR*（图2）中被划分为较短的*索引*字段和*标签'*字段。由于并行比较只针对较短的标签，*FARR* 中与每个较长标签进行比较所需的功耗得以降低。缓存访问时，如果域标识符和*索引*字段均匹配，则会与该缓存行的*标签'*进行比较。

STAR-NEWS 允许将地址中的更多位分配为索引位，以表示"更大的逻辑缓存（大于实际物理缓存大小）"，这在后文中会展示出进一步的性能收益。图 4展示了一个示例，其中额外分配了 4 位给*索引*字段。缓存行的映射条目由域标识符和*索引*组成，从而实现该域内动态的地址到缓存重新映射。

图 5 展示了 STAR-NEWS 中的加载处理。对于内存访问，STAR-NEWS 首先检查映射条目数组，查看是否存在具有匹配的域标识符和*索引*位的缓存行。

如果不存在与请求具有相同映射条目位的缓存行，就会发生映射未命中。只有当映射条目数组发生**映射**

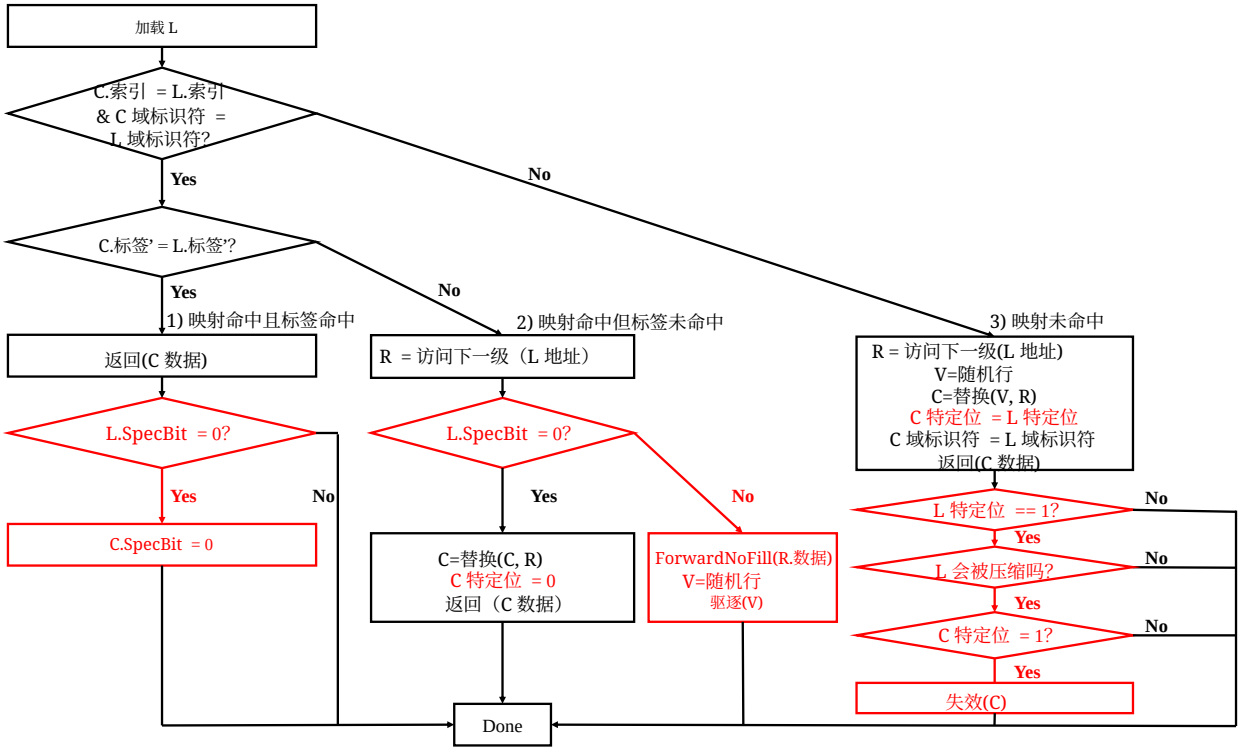

图5：加载处理程序 STAR-NEWS 的.....。处理推测执行攻击的修改以红色显示。

命中。缓存命中是指缓存同时发生映射命中和标签命中。如果映射命中，但标签' 字段不匹配，则会发生标签未命中。我们说明，不同路径中的加载请求处理方式不会泄露信息，也无法用于测量现有缓存状态。

映射命中和标签命中。对于缓存命中，STAR-NEWS 会像传统缓存中的普通命中一样返回数据。作为副作用，如果该缓存行是由先前的推测内存操作获取的（其 *SpecBit* 已置位），并且被非推测操作访问，那么它的*SpecBit* 就会被清除。

在查找映射命中时引入域标识符 比较，意味着共享内存区域中的缓存行不再共享。换言之，即使地址位匹配，来自其他安全域的程序也无法命中当前安全域的缓存行。此*NoHit* 特性可防止基于命中的侧信道攻击，因为攻击者无法观察到对受害者所用地址的命中。

映射命中但标签未命中。对于缓存行 C 的映射命中但标签'未命中，需要从下一级内存获取该缓存行。如果该加载是非推测加载，且其*SpecBit* 已清零，那么当请求的缓存行 R 返回时，允许其替换 C。同时，STAR-NEWS 会清除 R 的*SpecBit* 并返回数据。

然而，对于推测加载，如果缓存行 C 是为同域推测执行攻击而放入缓存的，那么替换缓存行 C 可能会导致地址泄露。驱逐 C 使攻击者能够推断加载地址的索引位。

STAR-NEWS 不允许将 R 填入缓存，而是转发数据，因此不会造成泄露。此外，STAR-NEWS 需要驱逐一条随机缓存行，因为保留所有旧缓存行可能导致观察结果不同于处理映射未命中时的情况；映射未命中会驱逐其中一条缓存行。强攻击者可以尝试利用这种差异构造驱逐集 [43, 53] 。区别处理推测加载和非推测加载，使 STAR-NEWS 能够抵御同域未命中攻击（表1中的 S-SD-PP），而原始 NewCache 无法防止这种攻击。这也表明，设计全面的方案并非只是组合不同的安全特性，还应进行调整以避免引入新的攻击。

尽管 STAR-NEWS 能够保障安全性，但驱逐缓存行而不填充缓存可能会产生性能开销。STAR-NEWS 可以通过增加索引中的位数来降低不填充的频率，这样 STAR-NEWS 就不太可能出现映射命中和标签未命中的情况。此外，在维持安全性和物理缓存大小的同时，它还支持灵活地调整性能（参见第 6.3 节）。

映射未命中。发生映射未命中时，也会访问下一级存储器。当请求的缓存行 R 返回后，将随机缓存行 V 替换为 R。随后，将 R 的 *SpecBit* 和*DomainID* 设置为与加载请求相同。如果加载是推测执行的，之后又被冲刷流水线，处理器将向缓存发送 *SFill-Inv* 信号，使推测获取的缓存行 R 失效。如果 R 已被

某些非推测内存操作访问过，并且在冲刷流水线之前已清除其 *SpecBit*，则可以保留 R。

在基于未命中的侧信道攻击中，由于双方的域标识符不匹配，受害者总会遇到映射未命中，并替换攻击者的缓存行。STAR-NEWS 中的随机化替换可防止信息泄露，因为受害者的缓存行以相同的概率驱逐攻击者的任意缓存行。

该机制也能对推测执行攻击提供类似的保护。基于未命中的不同域攻击者只能观察到随机驱逐，无法获知推测加载 L 的地址。对于同域攻击，如果发送方的推测加载发生映射未命中，它仍会随机替换一条缓存行，不会向攻击者泄露任何信息。

要点。 STAR-NEWS 实现了低延迟、推测感知的随机化缓存。尽管其设计受到 New- Cache 的启发，但它进一步解决了利用映射命中情况实施的微妙却关键的同域 Prime+Probe 攻击。

5.4 *SFill-Inv*的实现

我们优化了新功能“推测性缓存填充 + 失效（*SFill-Inv*）”的性能，以减少流水线停顿和往返内存流量。我们介绍对内存请求、内存响应、加载的冲刷过程，以及缓存中处理*SFill-Inv*的方式所做的修改。

内存请求与响应。每个内存响应都增加了一个额外的 *SourceLevel* 字段。*SourceLevel* 记录内存访问找到请求地址时所在的缓存级别。在双级缓存系统中，L1 缓存命中时，*SourceLevel* 设为 1。同样，L2 缓存命中时，*SourceLevel* 设为 2；如果数据来自内存，则设为 3。

加载冲刷过程流水线冲刷时，被清除的加载必须发送一个 *SFill-Inv* 请求，以使其获取的缓存行失效。如果推测加载命中 L1 缓存且已返回的*SFill-Inv* 请求将被跳过，此时返回的*SourceLevel* 为 1。*SourceLevel* 附加到*SFill-Inv* 请求中并用于处理。与需要停顿处理器直到第二次访问的响应返回的防御机制（第4.3节）相比，*SFill-Inv* 会在发送完所有 *SFill-Inv* 请求后立即恢复执行。试图观察恢复时间的微妙 unXpec [38] 攻击将无法奏效，因为处理器只会发送 *SFill-Inv* 请求，并不知道请求何时会完成。

处理*SFill-Inv*。我们实现了一种低成本的*SFill-Inv* 操作，不会获取新的缓存行，也不会生成
响应。这使得*SFill-Inv* 操作成为单向请求，与往返内存请求相比，将缓存系统中的流量减少了一半。我们的实现基于包含式写回缓存系统。当缓存接收到*SFill-Inv* 请求时，它会查找*SFill-Inv*的地址，该操作有三种可能的结果。

如果找到地址且缓存行的*SpecBit* 为 0，这意味着该缓存行已被非推测性内存访问访问过。该缓存行被视为安全，因此可以安全地丢弃*SFill-Inv*请求。

如果找到该地址，且缓存行的 *SpecBit* 为 1，则该缓存行会失效。随后，缓存会检查 *SFill-Inv* 的*SourceLevel*，如果 *SourceLevel* 大于当前缓存级别，则将请求发送到下一级。

如果未找到该地址，可能是因为推测获取的缓存行在冲刷流水线之前被替换或驱逐。如果*SourceLevel* 大于当前缓存级别，请求将传递到下一级缓存。

Name	SA-LRU	STAR-FARR	STAR-NEWS
处理器	1 个乱序 X86 核心，2GHz #加载队列项：32 #存储队列项：32 #重排序缓冲区项：192 #分支目标缓冲区项：4096 #返回地址栈项：16		
一级指令缓存	32KB，缓存行 64B 4 路，128 个组		
32 KB 一级数据缓存，64 B 缓存行			
	8 路，64 个组 最近最少使用替换 1周期延迟	512 路，1 个组 随机替换 1周期或2周期延迟	512条缓存行 修改后的 NewCache 1周期延迟
二级缓存	2MB，64B缓存行，16路，2048组 12周期延迟		
内存	50ns延迟（100周期）		

表2：GEM5模拟器的硬件配置。

5.5 性能影响

考虑第 4.3 节中的性能因素，STAR-FARR 和 STAR-NEWS 既不会影响频繁的正确路径执行，也不需要定期移动缓存行。它们只会在发生冲刷流水线时（错误路径较少发生）额外执行操作，发送轻量级的 *SFill-Inv*请求，而无需取回被驱逐的缓存行。由于无需等待*SFill-Inv*完成，流水线停顿时间也会缩短。

6 评估

我们评估了硬件架构的安全性和性能，其设计参数如图 2 所示。为获得可比结果，我们采用与 InvisiSpec[73] 和 CleanupSpec [55] 类似的容量和延迟参数。这些架构在周期精确GEM5模拟器 [7]中实现。我们将采用最近最少使用策略（SA-LRU）的组相联缓存作为基线。

STAR-FARR 和 STAR-NEWS 是我们提出的防御机制，可击败所有攻击。由于全相联缓存的访问延迟并不总能像组相联缓存那样短，我们针对 STAR-FARR 分别采用 1周期延迟的快速设计和 2周期延迟的慢速设计进行实验。STAR-NEWS 中 NewCache型L1缓存的访问延迟在实际电路中可以低至组相联缓存的水平 [16]。L2 缓存采用组相联设计，代表末级缓存（LLC）。最新的安全

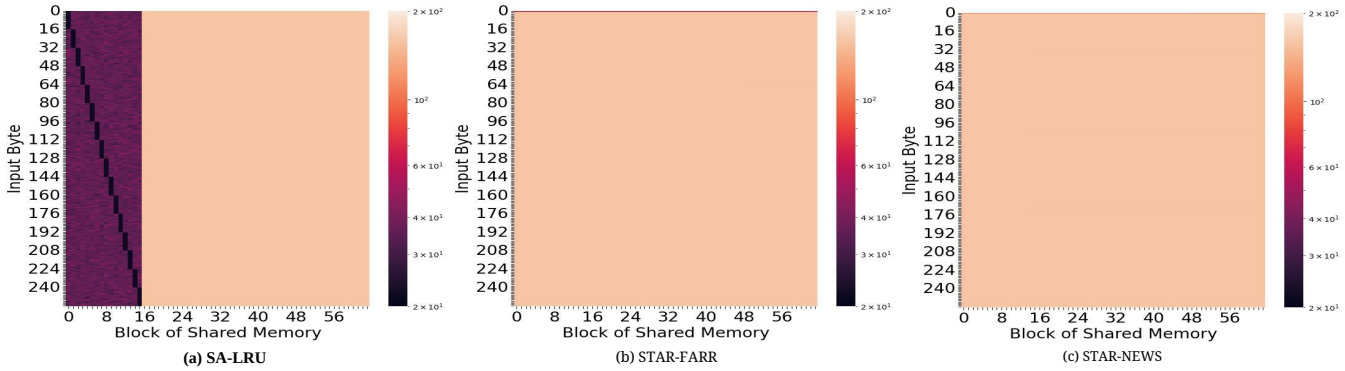

图6：针对 AES 的刷新-重载侧信道攻击。颜色越浅，缓存访问时间越长。密钥字节为 0，深色对角线表示它与输入字节异或后的结果，该结果会使执行时间缩短。

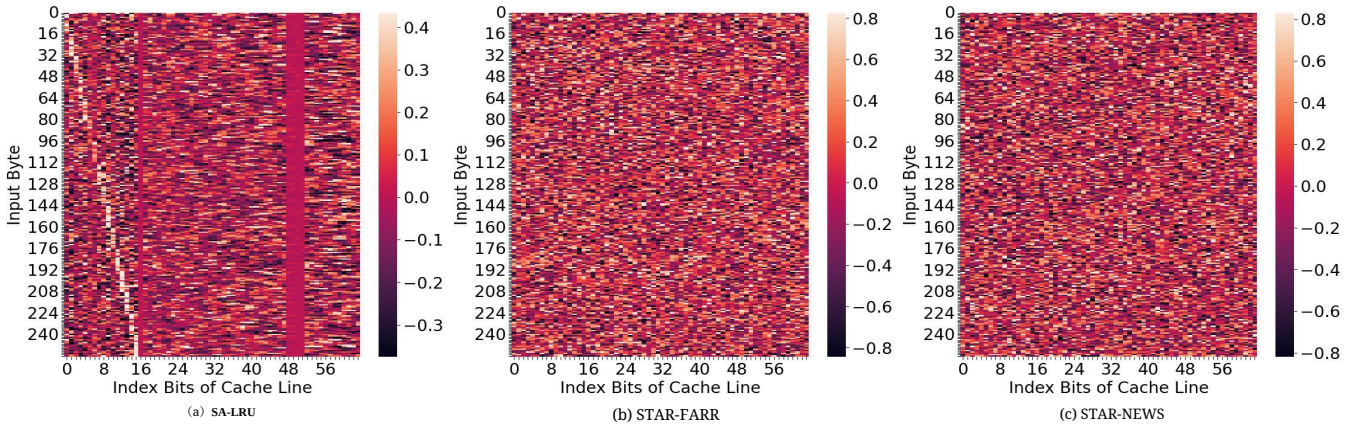

图7：针对 AES 的探测-驱逐侧信道攻击。颜色越浅，缓存访问时间越长。密钥字节为 0，浅色对角线表示它与输入字节异或后的结果，该结果会使执行时间延长。

采用随机化的末级缓存 [52, 71] 也采用组相联设计，因此该配置还可以模拟在重新映射缓存组之前启用末级缓存防护时的时序。

安全性评估采用以下 4 种代表性攻击，涵盖侧信道攻击和推测执行攻击：(1) 针对高级加密标准 (AES) 算法的刷新-重载侧信道攻击。(2) 针对 AES 的探测-驱逐侧信道攻击。(3) 利用刷新-重载缓存隐蔽信道的 Spectre-v1 推测执行攻击。(4) 利用探测-驱逐缓存隐蔽信道的 Spectre-v1 推测执行攻击。对于这两种 Spectre-v1 攻击，我们采用同域攻击者，这种攻击者比跨域攻击更危险。

6.1 安全性：侧信道攻击

我们评估了针对 AES-128 加密过程第一轮的侧信道攻击。优化后的 AES 实现 [13] 使用预先计算的查找表来执行其操作，字节代换、行移位 和列混淆。由于查找表的读取地址由加密密钥决定，侧信道攻击者可以通过分析缓存的内存访问模式，恢复密钥中的位。

在第一轮中，加密算法会读取 4 个查找表，每个表有 256 个 4 字节条目。我们将 16 字节的输入数据表示为 D_i，将 16 个密钥字节表示为 K_i，其中 i 从 1 到 16。对查找表 T_1 到 T_4 的访问如下：

$$T_1[D_1 \oplus K_1], T_1[D_5 \oplus K_5], T_1[D_9 \oplus K_9], T_1[D_{13} \oplus K_{13}]$$
$$T_2[D_2 \oplus K_2], T_2[D_6 \oplus K_6], T_2[D_{10} \oplus K_{10}], T_2[D_{14} \oplus K_{14}]$$
$$T_3[D_3 \oplus K_3], T_3[D_7 \oplus K_7], T_3[D_{11} \oplus K_{11}], T_3[D_{15} \oplus K_{15}]$$
$$T_4[D_4 \oplus K_4], T_4[D_8 \oplus K_8], T_4[D_{12} \oplus K_{12}], T_4[D_{16} \oplus K_{16}]$$

每次访问查找表时，都会将包含该地址的缓存块载入缓存。我们通过一名强攻击者来验证安全性：该攻击者了解或控制输入 AES 算法的数据，并尝试通过执行内存访问、观察时序差异来恢复密钥。对 AES 的侧信道攻击会使用随机输入数据重复 2^{15} 次，并将接收方的时序测量取平均值作为最终结果。

刷新-重载侧信道攻击。 当 AES 查找表位于攻击者与受害者共享的内存区域时，就可能发生刷新-重载攻击

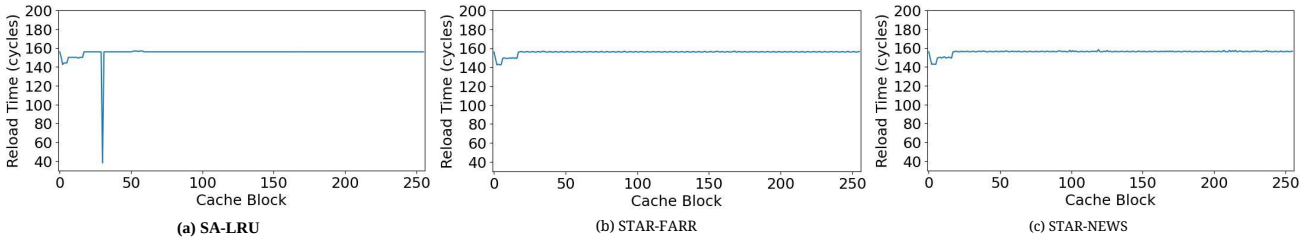

图8：刷新-重载型 Spectre v1 攻击。秘密为 30。

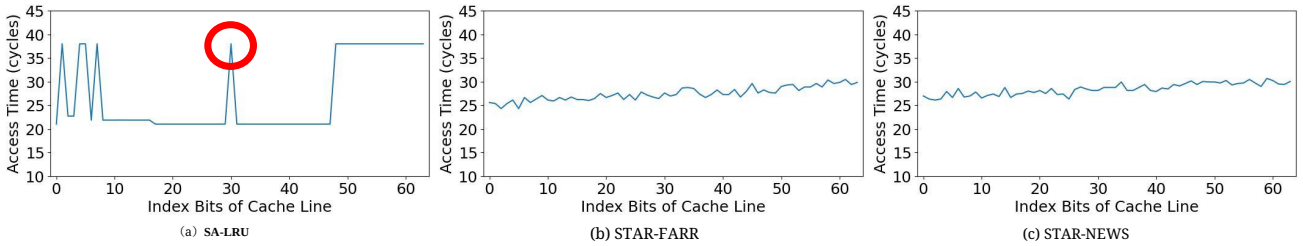

图9：探测-驱逐型 Spectre v1 攻击。秘密为 30。

。为恢复一个密钥字节，攻击者会刷新查找表条目，然后等待受害者执行。由于第一轮中会使用密钥字节 K_1,K_5,K_9 和 K_{13}来访问T_1，攻击者可以通过重新加载T_1 的所有条目并观察缓存命中情况，推断这些字节的值。同样，攻击者可以刷新并重新加载T_2,T_3和T_4 中的条目，从而恢复其他密钥字节。

图 6 (a) 展示了针对 AES 的一次成功的刷新-重载侧信道攻击。该实验表示一个共享的 4kB 内存区域（以 64 个缓存块作为 x 轴），其前半部分存放着一个 1kB 的 AES 查找表。y 轴表示输入字节的值 D_1。在 AES 加密过程中从未访问过浅色区域，因此重新加载这些缓存块所需的时间最长。前 16 个缓存块包含 AES 表项，平均访问延迟较短。图中有一条深色对角线，表示当输入字节为某些特定值时，缓存块的重载时间会更短。当输入字节的值介于 0 和 15（0x00 到 0x0f）之间时，会访问 AES 表的第一个缓存块 T_1 （偏移量：0x00）。由于第一轮中对 T_1 的访问为 $T_1[D_1 \oplus K_1]$, K_1 被部分泄露为 0x0u（u 是未知的 4 位，因为同一缓存行中的 16 个查找表项无法区分）。

在 STAR-FARR 和 STAR-NEWS 中，受害者的 AES 加密与 flush-reload 攻击者具有不同的域标识符，因此受害者使用的任何缓存行都不会在攻击者重新加载时造成缓存命中（见图6(b) 和 (c)）。

探测-驱逐侧信道攻击。 Prime+Probe 攻击不需要共享 AES 查找表。攻击者用其数组的缓存行填满缓存。查找表访问（例如 $T_1[D_1 \oplus K_1]$,）会替换由 D_1 和 K_1, 决定的某个缓存组中的缓存行，因此攻击者稍后探测该缓存组时会产生更长的延迟。

图 7(a) 展示了针对值为 0 的 AES 密钥字节成功实施的探测-驱逐侧信道攻击。对于每个缓存组（横轴），热力图显示某些输入值是否会导致访问时间长于平均值。浅色线表示攻击者在前 16 个缓存组中的缓存行与 AES 表项发生冲突，从而导致更长的访问时间。与之前的冲刷-重载攻击类似，可以将该密钥字节恢复为 0x0u（u 为未知位）。

STAR-FARR 中采用随机替换的全相联缓存能够挫败针对 AES 的 Prime+Probe 攻击（见图 7 (b)）。STAR-NEWS 中的动态重映射缓存也能确保随机替换，因为 DomainID 不匹配总会导致图 *5* 中的映射未命中路径，从而挫败图7 (c) 中的攻击。

6.2 安全性：推测执行攻击

Spectre v1 攻击 [36] 利用条件分支的错误预测。发送方会暂时绕过阻止非法越界访问的边界检查。基于缓存的 Spectre v1 发送方会暂时读取越界秘密，并利用该秘密生成一次内存访问，以获取或驱逐接收方可观察到的缓存行。

Flush-reload推测执行攻击。 图 1 所示的小工具通过 flush-reload 隐蔽信道泄露一个 8位秘密。条件分支中的发送方代码因分支预测错误而执行。数组[x]中的秘密通过非法偏移量 x 访问，并用于访问 共享 数组中的缓存行。接收方可以通过查找*i* 来推断秘密值，其中该值对应最短的重载时间。

图 8 展示了秘密为 30 时运行 flush-reload Spectre v1 攻击的结果。对于 SA-LRU 基线

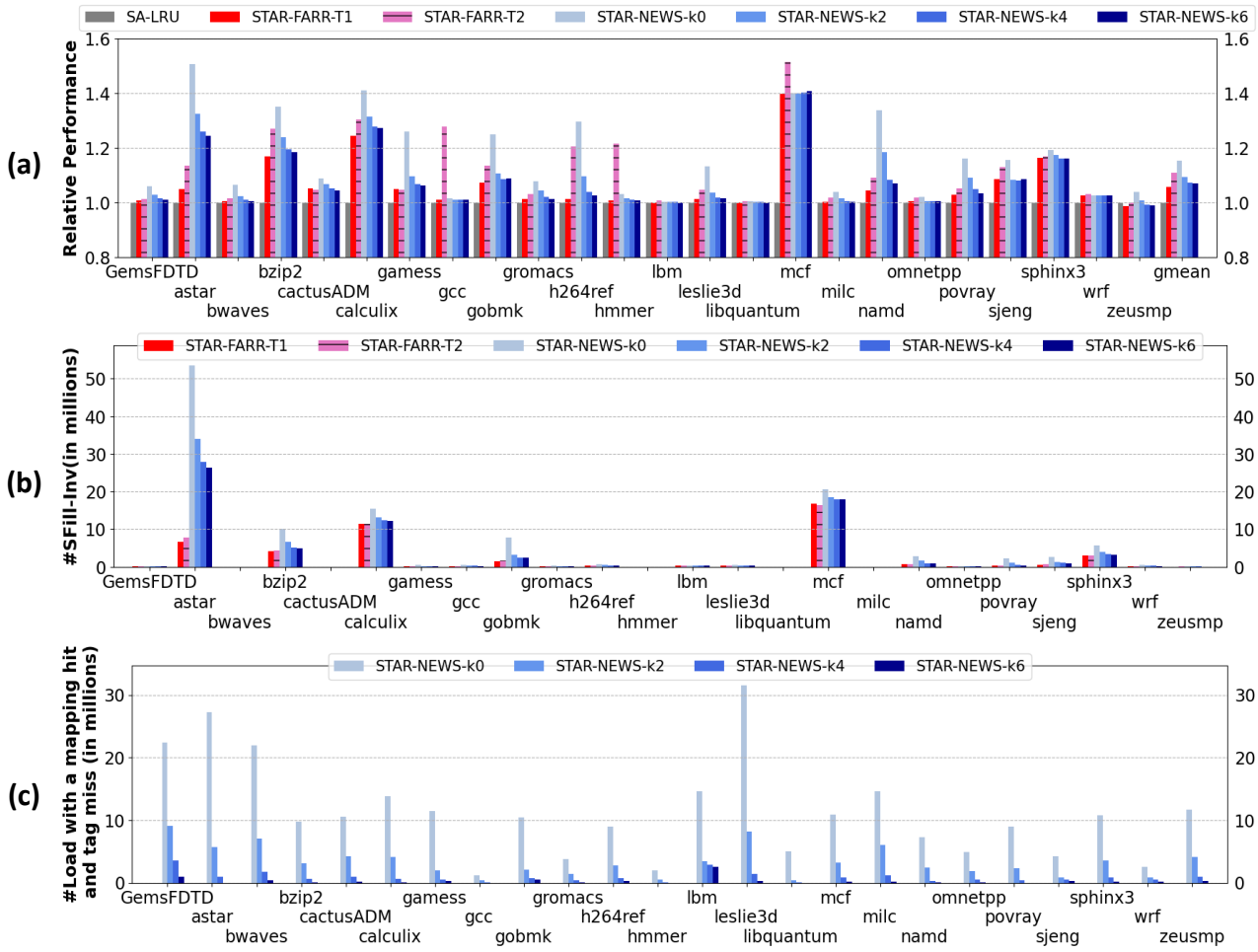

图 10：性能评估统计数据：(a) 总体性能：不同架构运行不同基准测试时的相对执行时间。最后一组为相对性能的几何平均值。(b) 发送到缓存系统的 *SFill-Inv* 请求数。(c) STAR-NEWS 系统中映射命中但标签未命中的推测加载数。

（蓝色），第 30 个块的访问时间较短，泄露了值为 30 的秘密。STAR-FARR（橙色）和 STAR-NEWS（绿色）没有可观察到的低重载延迟，因为推测获取的缓存行已被*SFill-Inv* 功能清除。

Prime-probe 推测执行攻击。 在 Prime-probe Spectre v1 攻击中，攻击者使用自己的数组填充缓存。Spectre v1 发送方小工具会访问秘密，并在将秘密用作地址时驱逐一条缓存行。随后，接收方会累计对不同缓存组的访问延迟。推测执行期间发生驱逐的缓存组会表现出更长的访问时间。

图 9 展示了 prime-probe Spectre v1 攻击（访问值为 30 的秘密）如何在 SA-LRU 系统（蓝色）中逐出接收方用圆圈标出的缓存行，其索引为 30。由于发送方代码中还有其他内存访问，因此会出现多个高访问延迟峰值。这些无关的访问会在固定的缓存组中引发冲突，而随着秘密值变化，用圆圈标出的峰值也会移动。结果表明，尽管 prime-probe 通道不需要共享内存区域，但其结果中的噪声比 flush-reload 推测执行攻击更多。

如图 9所示，STAR-FARR（橙色）的全相联缓存能够挫败该攻击。运行于 STAR-NEWS（绿色）中的发送方代码在遇到 标签未命中 或 映射未命中时，都会随机驱逐缓存行，因此不会产生依赖于秘密的时序差异。

虽然推测攻击还有许多其他变体，但它们主要区别在于非法获取秘密的方式。由于我们只关注防止秘密通过缓存时序通道泄露，因此使用 flush-reload 和探测-驱逐型 Spectre v1 攻击进行的测试具有代表性。我们还测试了推测存储绕过攻击 [22]，分别使用 flush-reload 和 prime-probe 隐蔽信道，结果相似（未展示）。

6.3 性能评估

通过在 SPEC CPU2006 基准测试套件中使用参考数据集运行 24 个基准测试，测量所测试架构的性能。我们跳过基准测试开头的 100 亿条指令，然后运行接下来的 5 亿条指令。STAR-FARR 有两种测试配置：STAR-FARR-T1 使用 1 周期全相联 L1D 缓存，STAR-FARR-T2 使用 2 周期 L1D 缓存。STAR-NEWS 有四种测试配置，分别增加 0、2、4 和 6 个索引位，记作 STAR-NEWS-k0、k2、k4 和 k6。图中展示并说明了 STAR-NEWS-k4。

总体性能。 图10 (a) 展示了 STAR Cache 相对于 SA-LRU 基线在执行时间方面的性能，SA-LRU 的性能归一化为 1。图10 (a) 的最后一列为开销的几何平均值。

STAR-FARR-T1 的平均执行时间比 SA-LRU 长 5.6%。在最佳情况下，它使 zeusmp 的性能提升了 1.3%，表现出更高的安全性和更好的性能。最坏情况下，mcf 的开销为 39.6%。在 STAR-FARR-T2 中采用速度较慢的全相联 L1 缓存，会使平均开销增至 10.8%。

对于 STAR-NEWS（蓝色柱），增加索引位 k 可提升性能。STAR-NEWS-k0 的平均开销较高，为 15.3%。STAR-NEWS-k2 将开销降至 9.4%（比 STAR-NEWS-k0 低 5.9%）。STAR-NEWS-k4 的开销为 7.2%（比 STAR-NEWS-k2 低 2.2%），而 STAR-NEWS-k6 的开销为 6.8%（比 STAR-NEWS-k4 低 0.4%）。增加索引位可以提升性能，但收益会逐渐递减。进一步增大 k 可能会增加缓存访问延迟，从而降低整体系统性能。

由于我们采用相同的硬件配置，因此可以将 STAR-FARR-T1（开销为 5.6%）和 STAR-NEWS-k6（开销为 6.8%）与 InvisiSpec 等领先的防御机制进行比较 [74] 其开销为 16.8%；CleanupSpec [74]的开销为 5.1%。然而，InvisiSpec 和 CleanupSpec 无法防御所有非推测性侧信道攻击，而 STAR-FARR 和 STAR-NEWS 可以。

我们还发现，如果能够实现一种快速的全相联缓存设计，且其访问延迟与 SA-LRU 相同，那么 STAR-FARR 会更出色。尽管 STAR-FARR-T1 的开销为 5.6%，STAR-FARR-T2 的开销则会升至 10.8%。在这种情况下，开销仅为 6.8% 的 STAR-NEWS-k6 会是更好的选择。

导致性能开销的因素。造成 STAR Cache 性能下降的主要因素有两个。第一个因素同时影响 STAR-FARR 和 STAR- NEWS，即被清除的加载需要发送 *SFill-Inv* ，以便在发生错误推测时使缓存行失效。第二个因素仅影响 STAR-NEWS，即映射命中但标签未命中时的 *ForwardNoFill* 路径（见图 5），该路径会阻止缓存填充并驱逐缓存行。这两者都可能导致缓存未命中率升高，进而降低性能。

图 10 (b) 展示了每个基准测试中发送的 *SFill-Inv* 请求数量。比较图 10 (a) 与图 10 (b) 可以发现二者之间存在很强的相关性。在 STAR-FARR 和 STAR-NEWS 系统中性能开销最高的基准测试，例如 astar、bzip2、calculix 和 mcf，也是需要更多 *SFill-Inv* 操作的基准测试。这意味着推测访问的缓存行很可能会被后续指令使用。尽管使这些缓存行失效会降低性能，但这一功能对于确保防御推测执行攻击是必要的。

图 10 (c) 显示了映射命中但标签未命中（TagMiss 加载）的推测加载数量。对于 TagMiss 加载，不允许缓存填充，而是直接将数据转发给处理器；同时还会触发随机缓存行驱逐（见图 5 中间的红色路径）。STAR-NEWS 拥有更多索引位的优势显而易见。没有额外位时，STAR-NEWS-k0 在 leslie3d 等基准测试中会产生大量 TagMiss 加载。增加少量额外位，就能显著减少 TagMiss 加载的数量。

我们展示了，即使使用较小的 k 值（例如 2、4 和 6），也能大幅减少这些缓存填充阻止和缓存行驱逐。在图 10 (c) 中，对于所有基准测试，STAR-NEWS-k2、STAR-NEWS-k4 和 STAR-NEWS-k6 通过采用 *ForwardNoFill* 处理 TagMiss 加载的数量平均分别减少了 70.4%、91.5% 和 97.3%。这是因为索引字段中增加位数可以降低发生映射命中的概率（图 $==5$ 中的 C.Index $\leqslant$ L.Index），从而实现更多有效替换。与整体性能方面的观察结果类似，我们建议将 $k=4$ 设为 2 或 6，作为最佳选择。

错误路径加载比例。我们统计了所有基准测试中被冲刷流水线的推测加载比例。该比例介于 0.0% 至 52.7% 之间，平均为 11.1%。这意味着大多数推测加载都属于正确路径指令，这也是 CleanupSpec 和我们提出的 *SFill-Inv*性能开销较低的原因。

7 结论

缓存状态是侧信道攻击和推测执行攻击窃取信息的重要目标。此前的硬件防御机制只能应对侧信道攻击或推测攻击，无法兼顾二者。据我们所知，本文首次清晰地展示了这两类攻击的不同维度，强调了同域推测攻击的危险性，并针对这两类攻击提出了全面的解决方案。我们提出的 STAR-FARR 和 STAR-NEWS 缓存架构，性能开销分别仅为 5.6% 和 6.8%。我们提出的设计方法还可用于设计其他安全缓存，以应对其他攻击类别并保护不同的安全硬件子系统。

参考文献

[1] AINSWORTH, S. 和 JONES, T. M. Muontrap：通过捕获推测状态来防止跨域 Spectre 类攻击。载于 *ACM/IEEE* 第 *47* 届计算机体系结构国际研讨会（*ISCA*）（2020年）。

[2] AMD。Amd64 技术间接分支控制扩展。https://developer.amd.com/wp-content/resources/Architecture_指南_更新_间接_分支_控制.pdf，2018年。

[3] BARBER, K., BACHA, A., ZHOU, L., ZHANG, Y.,AND TEODORESCU, R. Specshield：保护推测数据免受微架构隐蔽通道攻击。发表于第*28*届并行体系结构与编译技术国际会议 *(PACT)*（2019年）。

[4] BEHNIA, M., SAHU, P., PACCAGNELLA, R., Y U, J., ZHAO, Z. N., ZOU, X., UNTERLUGGAUER, T., TOR- RELLAS, J., ROZAS, C., MORRISON, A., MCKEEN, F., LIU, F., GABOR, R., FLETCHER, C. W., BASAK, A., AND ALAMELDEEN, A. 推测干扰攻击：破解隐形推测方案。发表于第*26*届 *ACM* 国际编程语言与操作系统体系结构支持会议论文集（美国纽约州纽约市，2021年），ASPLOS '21，计算机协会，第1046–1060页。

[5] BERNSTEIN, D. J. 针对高级加密标准的缓存时序攻击。在线（2005年）。

[6]BHATTACHARYYA, A., SANDULESCU, A., NEUGSCHWANDTNER, M., SORNIOTTI, A., FALSAFI, B., PAYER, M., AND KURMUS, A. Smotherspectre：通过端口争用利用推测执行。载于 *2019*年 *ACM SIGSAC* 计算机与通信安全会议（2019年）。

[7] BINKERT, N., BECKMANN, B., BLACK, G., REIN- HARDT, S. K., SAIDI, A., BASU, A., HESTNESS, J., HOWER, D. R., KRISHNA, T., SARDASHTI, S., SEN, R., SEWELL, K., SHOAIB, M., VAISH, N., HILL, M. D., AND WOOD, D. A. GEM5模拟器。*SIGARCH* 计算机体系结构新闻第*39*卷，第2期（2011年8月），1–7。

[8] BOURGEAT, T., DREAN, J., YANG, Y., TSAI, L., EMER, J., AND YAN, M. Casa：随机映射缓存的端到端定量安全分析。发表于*2020*年第*53*届 *IEEE/ACM* 微体系结构国际年会（*MICRO*）（2020年），第 1110–1123 页。

[9] BOURGEAT, T., LEBEDEV, I., WRIGHT, A., ZHANG, S., ARVIND 和 DEVADAS, S. Mi6：安全 enclave 在推测式乱序处理器中。第*52*届 *IEEE/ACM* 国际微体系结构研讨会 (2019年)。

[10] CANELLA, C., BULCK, J. V., SCHWARZ, M., LIPP, M., VON BERG, B., ORTNER, P., PIESSENS, F., EV- TYUSHKIN, D., AND GRUSS, D. 对瞬态执行攻击和防御机制的系统评估。发表于第*28*届*USENIX*安全研讨会（2019年）。

[11] CANELLA, C., GENKIN, D., GINER, L., GRUSS, D., LIPP, M., MINKIN, M., MOGHIMI, D., PIESSENS, F., SCHWARZ, M., SUNAR, B., VAN BULCK, J., AND YAROM, Y. Fallout：在抗 Meltdown CPU 上泄露数据。发表于*2019*年 *ACM SIGSAC* 计算机与通信安全会议论文集（美国纽约州纽约，2019年），CCS '19，计算机协会，第 769–784 页。

[12] CENTER, M. S. R. 缓解推测执行侧信道硬件漏洞。https://msrc-blog.microsoft.com/2018/03/15/mitigating-speculative-execution-side-channel-hardware-vulnerabilities。

[13] DAEMEN, J. 和 RIJMEN, V. *Rijndael* 的设计，第 2 卷。Springer，2002 年。

[14] DESSOUKY, G.、FRASSETTO, T. 和 SADEGHI, A.- R. Hybcache：面向可信执行环境的混合型抗侧信道缓存。载于第 *29* 届 *USENIX* 安全研讨会论文集（2020年），第 451–468 页。

[15]DOMNITSER, L.、JALEEL, A.、LOEW, J.、ABU-GHAZALEH, N. 和 PONOMAREV, D. 不可被独占的缓存：以低复杂度缓解缓存侧信道攻击。*ACM* 架构、代码优化汇刊 *8*，4（2012 年）。

[16] ERBAGCI, B.、LIU, F.、CAKIR, C.、AKKAYA, N. E. C.、LEE, R. 和 MAI, K. 一种采用动态替换映射的 65 纳米体硅 CMOS 32 kb 安全缓存存储器。发表于*2015 I EEE* 亚洲固态电路会议 *(A-SSCC)*（2015年），第 1–4 页。

[17] EVTYUSHKIN, D.、RILEY, R.、ABU-GHAZALEH, N. 和 PONOMAREV, D. Branchscope：一种针对方向分支预测器的新型侧信道攻击。发表于第二十三届国际编程语言与操作系统体系结构支持会议 （2018年）。

[18] FUSTOS, J.、FARSHCHI, F. 和 YUN, H. Spectre-guard：一种针对幽灵攻击的高效以数据为中心的防御机制。载于第 *56* 届设计自动化大会（*DAC*）（2019年）。

[19] GINGELL, R. A.、LEE, M.、DANG, X. T. 和 WEEKS, M. S. SunOS 中的共享库。AUUGN 8，5（1987），112。[20] GRAYSON, B.、RUPLEY, J.、ZURASKI, G. Z.、QUINNELL, E.、JIMÉNEZ, D. A.、NAKRA, T.、KITCHIN, P.、HENSLEY, R.、BREKELBAUM, E.、SINHA, V. 和 GHIYA, A. 三星 Exynos CPU 微架构的演进。载于 ACM/IEEE 第 47 届计算机体系结构国际研讨会（ISCA）（2020年）。[21] HE, Z.、HU, G. 和 LEE, R. 关于推测执行攻击的理解与推理新模型。载于 2021 年 IEEE 高性能计算机体系结构国际研讨会（HPCA）（2021年），第 40–53 页。[22]HORN, J. 推测执行，变体 4：推测性存储绕过，2018年。网址：https://bugs.chromium.org/p/project-zero/issues/detail?id=*1528*（2018年）。[23] HU, G.、HE, Z. 和 LEE, R. B. SoK：针对推测执行攻击的硬件防御措施。载于 2021 年安全与隐私执行环境设计国际研讨会（SEED）（2021年），IEEE，第 108–120 页。[24] IAIK. 熔毁概念验证。https://github.com/IAIK/meltdown，2019年。[25] 英特尔。惰性浮点状态保存。https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00145.html，2018年。[26] 英特尔。幽灵漏洞 V3a（RSRE）。https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00115.html，2018年。[27] 英特尔。微架构数据采样。
https://software.intel.com/content/www/us/en/develop/articles/software-security-guidance/technical-documentation/intel-analysis-microarchitectural-data-sampling.html，2019年。[28] 英特尔。事务异步中止。
https://software.intel.com/content/www/us/en/develop/articles/software-security-guidance/advisory-guidance/intel-tsx-asynchronous-abort.html，2019年。[29] 英特尔。L1D 驱逐采样。
https://software.intel.com/content/www/us/en/develop/articles/software-security-guidance/advisory-guidance/l1d-eviction-sampling.html，2020年。[30] 英特尔。特殊寄存器缓冲区数据采样。https://software.intel.com/content/www/us/en/develop/articles/software-security-guidance/technical-documentation/special-register-buffer-data-sampling.html，2020年。[31] 英特尔。VRS。
https://software.intel.com/content/www/us/en/develop/articles/software-security-guidance/advisory-guidance/vector-register-sampling.html，2020年。[32]KHASAWNEH, K. N., KORUYEH, E. M., SONG, C., EVTYUSHKIN, D., PONOMAREV, D., AND ABU- GHAZALEH, N. SafeSpec：以无泄漏推测执行驱散熔毁的阴影。载于第*56*届设计自动化会议（*DAC*）（2019年）。[33] KIM, Y., DALY, R., KIM, J., FALLIN, C., LEE, J. H., LEE, D., WILKERSON, C., LAI, K., AND MUTLU, O. 无需访问内存即可翻转其中的位：对 DRAM 扰动错误的实验研究。载于第*41*届年度国际计算机体系结构研讨会论文集（2014年），ISCA '14，IEEE 出版社，第361–372页。[34] KIRIANSKY, V., LEBEDEV, I., AMARASINGHE, S., DEVADAS, S., AND EMER, J. DAWG：一种防御措施，可防止推测执行处理器遭受缓存时序攻击。载于第*51*届*IEEE/ACM*国际微体系结构研讨会（*MICRO*）（2018年）。[35] KIRIANSKY, V., AND WALDSPURGER, C. 推测缓冲区溢出：攻击与防御机制。*arXiv*预印本 *arXiv:1807.03757*（2018年）。[36]KOCHER, P., HORN, J., FOGH, A., GENKIN, D., GRUSS, D., HAAS, W., HAMBURG, M., LIPP, M., MANGARD, S., PRESCHER, T., SCHWARZ, M., AND YAROM, Y. 幽灵攻击：利用推测执行。载于*2019* 年 *IEEE* 安全与隐私研讨会（*SP*）（2019年），第1–19页。[37] KORUYEH, E. M., KHASAWNEH, K. N., SONG, C., AND ABU-GHAZALEH, N. 幽灵回归！利用返回栈缓冲区的推测执行攻击。载于第*12*届 *USENIX* 攻击性技术研讨会（*WOOT 18*）（2018年）。[38] LI, M., MIAO, C., YANG, Y., AND BU, K. unXpec：攻破基于撤销的安全推测执行。载于*2022*年 *IEEE* 国际高性能计算机体系结构研讨会（*HPCA*）（2022年），第98–112页。[39] LI, P., ZHAO, L., HOU, R., ZHANG, L., AND MENG, D. 条件推测：一种有效的方法，可保护乱序执行免受幽灵攻击。载于*2019*年 *IEEE* 国际高性能计算机

*p*器架构 *(HPCA)* (2019 .

[40] LIPP, M.、SCHWARZ, M.、GRUSS, D.、PRESCHER, T.、HAAS, W.、FOGH, A.、HORN, J.、MANGARD, S.、KOCHER, P.、GENKIN, D.、YAROM, Y. 和 HAM- BURG, M. 熔毁：从用户空间读取内核内存。载于 第*27*届 *USENIX* 安全研讨会（2018年）。

[41] LIU, F., GE, Q., YAROM, Y., MCKEEN, F., ROZAS, C., HEISER, G., 和 LEE, R. B. Catalyst：在云计算中挫败针对末级缓存的侧信道攻击。载于*2016*年 *IEEE* 高性能计算机体系结构国际研讨会 *(HPCA)*（2016），第 406–418 页。

[42] LIU, F., WU, H., MAI, K., 和 LEE, R. B. Newcache：抵御缓存侧信道攻击的安全缓存体系结构。*IEEE Micro36*，5（2016），第 8–16 页。

[43] LIU, F., YAROM, Y., GE, Q., HEISER, G., 和 LEE, R. B. 针对末级缓存的侧信道攻击在实践中可行。载于*2015* 年 *IEEE* 安全与隐私研讨会（2015），第 605–622 页。

[44] LOUGHLIN, K.、NEAL, I.、MA, J.、TSAI, E.、WEISSE, O.、NARAYANASAMY, S. 和 KASIKCI, B. DOLMA：基于瞬态不可观测性原则保障推测执行安全。载于第*30*届 *USENIX* 安全研讨会（*USENIX Security 21*）（2021年）。

[45] MA, X.、ZHONG, H.、XIU, N.、CHEN, Y.、YIN, G.、NARAYANAN, V.、LIU, Y.、NI, K.、YANG, H. 和 LI, X. Capcam：一种用于高精度、高可扩展性搜索与计算应用的多级电容式内容寻址存储器。*IEEE* 超大规模集成电路（*VLSI*）系统汇刊 *30*，第11期（2022年），1770–1782页。

[46] MAISURADZE, G. 和 ROSSOW, C. Ret2spec：利用返回栈缓冲区进行推测执行。载于*2018*年 *ACM SIGSAC* 计算机与通信安全会议（2018年）。

[47] MUTLU, O., 和 KIM, J. S. Rowhammer：回顾。*IEEE* 集成电路与系统计算机辅助设计汇刊 *39*, 8（2020年），1555–1571。[48] OJOGBO, E. J., THOTTETHODI, M., 和 VIJAYKU-MAR, T. N. 安全自动边界检查：预防胜于治疗。载于第 *18* 届 *ACM/IEEE* 国际代码生成与优化研讨会（2020年）。[49] OMAR, H., 和 KHAN, O. Ironhide：一种安全的多核系统，可高效缓解交互式应用中的微架构状态攻击。载于*2020*年 *IEEE* 高性能计算机体系结构国际研讨会（*HPCA*）（2020年）。

[50] OSVIK, D. A., SHAMIR, A., 和 TROMER, E. 缓存攻击与对策：AES 的案例。载于密码学专题 – *CT-RSA 2006*（2006年），D. Pointcheval 编。

[51] PURNAL, A., GINER, L., GRUSS, D., 和 VER-BAUWHEDE, I. 基于随机化的受保护缓存架构的系统性分析。发表于*2021 IEEE* 安全与隐私研讨会（*SP*）（2021年），第 987–1002 页。

[52] QURESHI, M. K. Ceaser：通过加密地址和重新映射缓解基于冲突的缓存攻击。发表于第*51*届*IEEE/ACM*国际微体系结构研讨会（*MICRO*）（2018年）。

[53] QURESHI, M. K. 加密地址缓存的新型攻击与防御措施。发表于*2019 ACM/IEEE* 第*46*届年度国际计算机体系结构研讨会（*ISCA*）（2019年），第 360–371 页。

[54] RAGAB, H., MILBURN, A., RAZAVI, K., BOS, H., 和 GIUFFRIDA, C. Crosstalk：跨核心的推测数据泄漏确实存在。发表于*2021 IEEE* 安全与隐私研讨会（*SP*）（2021年）。

[55] SAILESHWAR, G., AND QURESHI, M. K. Cleanup-spec：一种实现安全推测的“撤销”方法。发表于第*52*届*IEEE/ACM*国际微体系结构研讨会（*MICRO*）（2019年）。

[56] SAILESHWAR, G., AND QURESHI, M. K. Mirage：采用实用的全相联设计缓解基于冲突的缓存攻击。发表于 *USENIX* 安全研讨会（2021年），第1379–1396页。

[57] SAKALIS,C.,KAXIRAS,S.,ROS,A., JIMBOREAN,A., AND SJÄLANDER, M. 通过选择性延迟和值预测实现高效的不可见推测执行。发表于第*46*届国际计算机体系结构研讨会（*ISCA*）（2019年）。

[58] SCHWARZ, M., LIPP, M., CANELLA, C., SCHILLING, R., KARGL, F., AND GRUSS, D. ConTExT：一种缓解 Spectre 的通用方法。发表于第*27*届网络与分布式系统安全年度研讨会（*NDSS'20*），美国加利福尼亚州圣迭戈（2020年）。

[59] SCHWARZ, M.、LIPP, M.、MOGHIMI, D.、VAN BULCK, J.、STECKLINA, J.、PRESCHER, T. 和 GRUSS, D. Zombieload：跨特权边界数据采样。发表于*2019*年 *ACM SIGSAC* 计算机与通信安全会议（2019年）。

[60] SCHWARZ, M.、SCHWARZL, M.、LIPP, M.、MASTERS, J. 和 GRUSS, D. Netspectre：通过网络读取任意内存。载于欧洲计算机安全研究研讨会（2019年）。

[61] SONG, W.、LI, B.、XUE, Z.、LI, Z.、WANG, W. 和 LIU, P. 随机化末级缓存仍易受缓存侧信道攻击！但我们可以修复它。载于*2021 IEEE* 安全与隐私研讨会（*SP*）（2021年），第 955–969 页。

[62] TARAM, M.、VENKAT, A. 和 TULLSEN, D. 上下文敏感的栅栏：通过微码定制保障推测执行的安全。在国际编程语言与操作系统架构支持大会（2019年）。

[63] VAN BULCK, J.、MINKIN, M.、WEISSE, O.、GENKIN, D.、KASIKCI, B.、PIESSENS, F.、SILBERSTEIN, M.、WENISCH, T. F.、YAROM, Y. 和 STRACKX, R. Foreshadow：利用瞬态乱序执行提取英特尔 SGX 王国的密钥。在第*27*届 *USENIX* 安全研讨会（2018年）。

[64] VAN BULCK, J.、MOGHIMI, D.、SCHWARZ, M.、LIPPI, M.、MINKIN, M.、GENKIN, D.、YAROM, Y.、SUNAR, B.、GRUSS, D. 和 PIESSENS, F. Lvi：通过微架构负载值注入劫持瞬态执行。在*2020*年 *IEEE* 安全与隐私研讨会（*SP*）（2020年）。

[65] VAN SCHAIK, S., MILBURN, A., ÖSTERLUND, S., FRIGO, P., MAISURADZE, G., RAZAVI, K., BOS, H., 和 GIUFFRIDA, C. Ridl：恶意在途数据加载。载于*2019*年 *IEEE* 安全与隐私研讨会（*SP*）（2019年）。

[66] VAN SCHAIK, S., MINKIN, M., KWONG, A., GENKIN, D., 和 YAROM, Y. Cacheout：通过缓存驱逐从英特尔 CPU 中泄露数据。载于*2021*年 *IEEE* 安全与隐私研讨会（*SP*）（2021年）。

[67] WANG, Z., 和 LEE, R. B. 用于抵御基于软件缓存的侧信道攻击的新型缓存设计。载于第*34*届年度国际计算机体系结构研讨会论文集（美国纽约州纽约，2007年），ISCA '07，计算机协会，第494–505页。

[68] WANG, Z. 和 LEE, R. B. 一种兼具高性能与安全性的新型缓存体系结构。载于*2008*年第*41*届 *IEEE/ACM* 国际微体系结构研讨会（2008年），第83–93页。

[69] WEISSE, O.、NEAL, I.、LOUGHLIN, K.、WENISCH, T. F. 和 KASIKCI, B. NDA：从源头防止推测执行攻击。载于第*52*届 *IEEE/ACM* 国际微体系结构研讨会（*MICRO*）（2019年）。

[70] WEISSE,O., VAN BULCK, J., MINKIN, M., GENKIN, D., KASIKCI, B., PIESSENS, F., SILBERSTEIN, M., STRACKX, R., WENISCH, T. F., AND YAROM, Y. Foreshadow-NG：利用瞬态乱序执行打破虚拟内存抽象。技术报告（2018年）。

[71] WERNER, M.、UNTERLUGGAUER, T.、GINER, L.、SCHWARZ, M.、GRUSS, D. 和 MANGARD, S. ScatterCache：通过随机化缓存组挫败缓存攻击。载于第*28*届*USENIX*安全研讨会（*USENIX Security 19*）（加利福尼亚州圣克拉拉，2019年8月），USENIX协会，第675–692页。

[72] XIONG, W., 和 SZEFER, J. 瞬态执行攻击及其缓解措施综述。*ACM*计算调查 (2021年)。

[73] YAN, M., CHOI, J., SKARLATOS, D., MORRISON, A., FLETCHER, C., 和 TORRELLAS, J. Invisispec：让推测执行在缓存层次结构中不可见。载于第*51*届*IEEE/ACM*国际微体系结构研讨会(2018年)。

[74] YAN, M., CHOI, J., SKARLATOS, D., MORRISON, A., FLETCHER, C. W., 和 TORRELLAS, J. 更正：Invisispec：让推测执行在缓存层次结构中不可见。在线(2019年)。

[75] YAROM, Y., 和 FALKNER, K. Flush+reload：一种高分辨率、低噪声的L3缓存侧信道攻击。载于第*23*届*USENIX*安全研讨会(2014年)。

[76] YU, J., YAN, M., KHYZHA, A., MORRISON, A., TOR- RELLAS, J., AND FLETCHER, C. W. 推测污点跟踪（stt）：针对推测访问数据的全面保护。载于第*52*届年度*IEEE/ACM*国际微体系结构研讨会（*MICRO*）（2019年）。