

Cache 侧信道攻击防御量化研究

王占鹏^{1,2}, 朱子元^{1,2}, 王立敏^{1,2}

¹中国科学院信息工程研究所 北京 中国 100093

²中国科学院大学(网络安全学院) 北京 中国 100049

摘要 芯片安全防护技术关系到国家、企业和个人的信息安全, 相关的研究一直是计算机安全领域的热点。片上高速缓存对芯片性能起着重要作用, 可以有效提升芯片内核访问效率。传统的缓存设计并没有充分考虑安全性, 侧信道攻击会对 Cache 造成巨大威胁, 可以窃取加密密钥等内存存储敏感信息。攻击者利用侧信道的技术窃取用户的隐私数据或加密算法密钥时不会改变片上系统芯片的运行状态, 从而使计算机系统很难检测是否受到了攻击。与基于电磁信号和基于能量检测的侧信道攻击相比, 基于存储共享的侧信道攻击只需要利用软件测量就可以实现, 对芯片安全的威胁更大。目前存在多种侧信道攻击和防御手段, 但缺乏一套完善的关于系统架构的安全度量方法, 对 Cache 的安全性进行有效评估。本文对 Cache 侧信道攻击和防御手段进行模型化分析, 提出一套 Cache 安全性量化研究方法。首先, 我们采用 CVSS 漏洞评分模型对 Cache 侧信道攻击进行量化评分。然后, 利用贝叶斯公式, 构建侧信道攻击和防御的关系模型。最后, 通过图模型对 Cache 侧信道攻击机理进行建模, 计算在防御架构基础上不同威胁的攻击成功率, 并结合 CVSS 防御得分求得不同防御方法的得分。本文针对 Cache 侧信道攻击进行机理建模, 对攻击和防御进行评估和探索, 为硬件安全人员提供理论支持。

关键词 Cache 侧信道; CVSS; 贝叶斯模型; 安全量化; 安全架构

中图法分类号 TP391 DOI号 10.19363/J.cnki.cn10-1380/tn.2022.12.10

Evaluations of Cache Side Channels Attacks and Defends

WANG Zhanpeng^{1,2}, ZHU Ziyuan^{1,2}, WANG Limin^{1,2}

¹Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

²School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract The technology of SoC (System on a Chip) security protection has a significant impact on the information security of countries, enterprises and individuals, which has always been a hot spot in the field of computer security. On-chip cache plays an important role in chip performance, which can effectively improve the access efficiency of chip core. The traditional cache design does not fully consider the security. Unfortunately, side channel attacks will pose a huge threat to the Cache, they can steal encryption keys and other sensitive information stored in the memory. Attackers use side-channel technology to steal users' private data or even encryption algorithm keys without changing the executing state of the SOC, making it difficult for computer systems to detect attacks. Compared with side-channel attacks based on electromagnetic signals and energy detection, side-channel attacks based on shared storage can be implemented only by software measurement, which pose a greater threat to chip security. At present, there are multiple side channel attacks and defense methods, but there is a lack of security measurement methods on the system architecture to effectively evaluate the security of the cache. In this paper, a model analysis about Cache side channel attacks and defenses is carried out, and a set of security quantitative research methods for Cache security is proposed. First, we use the CVSS vulnerability scoring model to quantitatively score cache side channel attacks. Secondly, using Bayes model to build a relationship model of side channel attacks and defenses. Finally, the graph model is introduced to represent the cache side channel attack mechanism, with this model, we can calculate the attack success rate of different threats based on the defense architecture, and combine the CVSS defense score to obtain the score of different defense methods. This paper models the mechanism of cache side channel attacks, evaluates and explores attacks and defenses, and provides theoretical support for hardware security personnel.

Key words Cache side channel; CVSS; Bayes model; security; security architecture

通讯作者: 朱子元, 博士, 研究员, Email: zhuziyuan@iie.ac.cn。

本课题得到国家科技重大专项(No. 2018ZX01028101)和国家重点研发计划(No. 2018YFB2202104)资助。

收稿日期: 2020-08-24; 修改日期: 2020-11-20; 定稿日期: 2022-12-07

1 引言

由于摩尔定律的存在, 中央处理器 (Central processing unit, CPU) 的发展速度远远大于内存, 高速缓存 Cache 的设计, 是为了解决现有差距, 利用 Cache 的局部性原理, 可以最大程度的逼近 CPU 的速度, 提高系统的性能^[1]。片上高速缓存对芯片性能起着重要作用, 可以有效提升芯片内核访问效率。但传统的缓存设计并没有充分考虑安全性, 数据访问 Cache 和内存的时间差异可以泄露安全信息, 对芯片系统造成重大安全隐患^[2-5]。

侧信道攻击是攻击者利用受害者在运行时所泄露的侧信息, 包括能耗、电磁、时序等信息, 根据敏感信息与侧信息之间的相关性, 来利用侧信息推断敏感信息, 对重要信息进行窃取^[2]。Cache 侧信道通常是时序攻击, 利用 Cache 命中和失效之间读取数据的时间差异, 进行推断攻击。攻击者一般不能直接获得获取受害者数据, 通过各种攻击方法获得受害者内存访问地址, 分析与加密算法的关联性^[3]。

现有研究^[6-11]表明, 不同的 Cache 侧信道攻击已经对系统造成非常大的影响。Gullasch 等^[6]对 L1 Cache 进行攻击, 成功的恢复了 AES 的密钥; 陈财森等人^[7]利用 Cache 访问踪迹攻击, 破解了 RSA 加密算法; Liu fangfei 等人^[8]针对多核系统, 成功对 LLC 进行攻击; Xu yunjing 等^[9]成功对云环境场景下的加密进行攻击, 提取了密钥。传统的侧信道攻击更多的是结合加密算法, 利用 Cache 访问的时间差与加密算法运行时的关联信息进行推断, 获得密钥信息。而 Spectre^[10]和 Meltdown^[11]等硬件漏洞的出现, 对 Cache 内存带来更大的危害性, 这些漏洞通过推测执行的方式可以直接攻击密钥本身, 因此展开 Cache 侧信道相关的研究对计算机安全具有非常重要的意义。

针对现有侧信道攻击, 不同的防御手段已经被提出, 包括软件的防御方案和硬件的防御方案。软件的防御方法一般是通过随机使用查找表^[12]、提前将查找表载入 Cache^[13]等方法进行保护, 但这种方法存在很大的局限性: (1) 针对特定算法的安全防护无法及时、准确性应对攻击的变化; (2) 算法提升安全性的同时会显著增加加密算法的运行时间。而基于硬件的防御手段可以有针对的对侧信道攻击进行防御, 设计安全的 Cache 系统, 在减少性能损耗的同时提升 Cache 的安全。目前已经提出的安全 Cache 防护方法包括: Cache 组隔离^[14]、行锁定^[15-16]、路隔离^[17-18]、随机填充^[19]、随机驱逐^[20]、随机地址映

射^[21]、组置换^[22]、内存隔离^[23]、flush 失效^[23]等。

关于 Cache 侧信道攻击手段和安全 Cache 设计等现有研究已经取得较大进展, 但由于攻击和防御的方法种类繁多并且螺旋上升的, 我们无法判断现有的防御手段能否有效应对不断变化和衍生的新型攻击^[24]。因此需要我们对攻击和防御进行准确度量, 分析不同防御手段对各种攻击的防御效果, 总结防御手段的优势, 进而设计更加安全的 Cache 结构^[25]。信息安全量化方法包含系统仿真和模型仿真: 系统仿真是建立真实的攻防场景, 模拟不同攻击手段在不同防御架构上的攻击效果。但这种方法存在无法准确描述各种攻击变种、平台搭建困难、仿真时间过长等缺陷^[26-27]。针对目前侧信道攻击和防御量化研究所存在的问题, 本文提出了基于 CVSS 和图模型结合的 Cache 侧信道机理建模方法, 对现有 Cache 侧信道攻击和防御方法进行有效评估。本文的主要创新点如下所示:

- 1) 整合新型硬件漏洞和传统侧信道攻击方法, 采用 CVSS 漏洞评估模型对所有攻击进行量化评估;
- 2) 基于侧信道攻击机理, 利用图模型进行建模, 引入攻击成功概率来描述 Cache 侧信道攻击和防御;
- 3) 采用贝叶斯模型构建侧信道攻防关系, 计算所有攻击在不同防御下的攻击成功概率, 并结合 CVSS 评分获得不同防御的有效性得分;
- 4) 通过对 Cache 侧信道现有安全体系量化, 整合防御手段, 探索更有效的防护方法。

2 背景介绍

2.1 Cache 侧信道攻击

利用访问 Cache 和内存时间的差异性, 学者们提出了多种攻击方法。我们对现有攻击算法进行总结, 按照攻击场景, 将 Cache 侧信道攻击划分为碰撞计时攻击、访问驱动攻击、推测执行攻击。

2.1.1 碰撞计时攻击

Cache 碰撞攻击是基于统计学分析, 利用加密算法不同明文加密过程查找同一个表时, 统计 Cache 访问命中和失效。这种攻击仅仅统计加密整体执行时间, 采集加密时从发送明文到接收密文的时间差。

以 AES 加密算法为例^[28], 碰撞攻击的攻击原理如下所示: 假设存在两个 $k_i \oplus x_i$ 和 $k_j \oplus x_j$, 其中当第一轮加密时 k_i 和 k_j 是 AES 的加密密钥, x_i 和 x_j 代表的是明文; 最后一轮加密, k_i 和 k_j 是最后一轮扩展密钥, x_i 和 x_j 代表的是上一轮的加密密文。当两次加密发生碰撞时, 表明 $k_i \oplus x_i$ 和 $k_j \oplus x_j$ 访问相同

的查找表, 因此:

$$k_i \oplus x_i = k_j \oplus x_j \Leftrightarrow k_i \oplus k_j = x_i \oplus x_j, \quad (1)$$

通过一系列采样, 攻击者可以获得所有 256 种 $x_i \oplus x_j$ 的可能值。通过对所有可能值的访问时间进行累加, 总时间最小的 $x_i \oplus x_j$ 即与密钥值 $k_i \oplus k_j$ 有直接关联。更详细的碰撞攻击实验分析可参考文献[29]。

2.1.2 访问驱动攻击

攻击者和受害者共用 Cache 存储空间, 其中单核系统不同程序共用所有 Cache 存储, 多核系统共用末级缓存存储[32]。访问驱动 Cache 攻击则是利用间谍进程对同密码进程查表对应 Cache 组区域相同的私有数据进行访问, 根据访问时间得到命中和失

效信息, 并获取密码进程查表访问 Cache 组地址, 然后转化为查表索引进行密钥分析[12]。

利用密码进程数据访问内部冲突导致的命中和失效进行密码分析不同, 访问驱动 Cache 攻击利用的是间谍进程和密码进程之间的数据访问外部冲突导致的一次 Cache 访问命中和失效时间差异, 对计时精度要求比较高, 要求攻击者能够精确采集一次 Cache 访问时间, 并区分出是访问命中还是失效[31-32]。现有的基于查找表的常见的 Cache 侧信道攻击包括 prime-probe[8]、evict-time[33]、flush-reload[6]、flush-flush[34]等。我们通过简化的 Cache-memory 模型来表示几种攻击, 描述访问驱动攻击原理。其中左侧为 4 路 12 组 Cache, 右侧为简化的内存模型, 包括查找表和要填充的大数组。

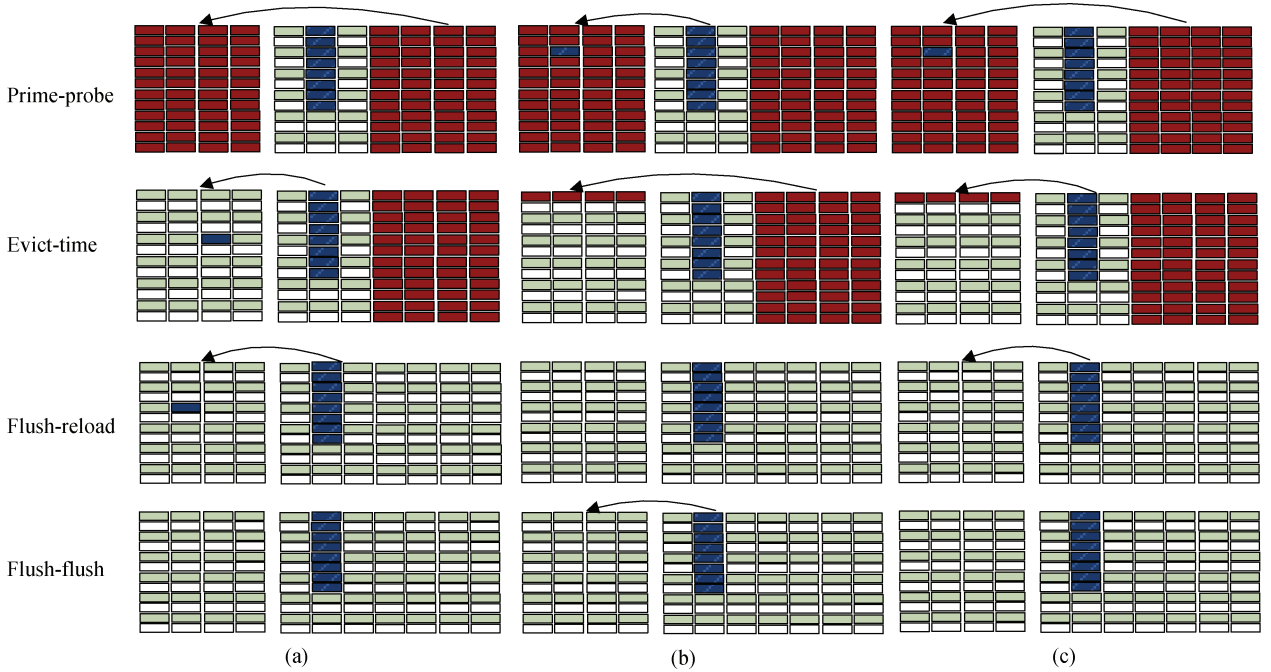


图 1 访问驱动攻击的攻击流程

Figure 1 The process of access-driven attack

(1) prime-probe 攻击

攻击者是观察自己加载的大数组是否发生 Cache 失效, 访问相同地址时是否花费了更多的时间, 探索自己的数据是否被驱逐。该攻击的步骤是: (a)在 Cache 中先加载一个大数组, 使其能完全覆盖整个 Cache; (b)当受害者执行加密查表操作时, 查找表的数据被加载到 Cache 中时, 攻击者的大数组就会被驱逐; (c)攻击者 probe 自己的数据, 观察自己的数据访问时长来判断是否发生 Cache 失效。

(2) evict-time 攻击

攻击者想判断受害者是否使用某部分数据, 采

用驱逐包含这部分数据的某些行或组, 观察受害者是否会发生 Cache 失效, 被迫访问内存而消耗更多的访问时间。驱逐计时攻击的步骤主要包括: (a)受害者正常访问 Cache; (b)攻击者驱逐指定的 Cache 行; (c)观察受害者是否使用了更长时间。

(3) flush-reload 攻击

受害者想观察在清空 Cache 行的时候, 如果受害者将这部分数据载入了 Cache, 观察自己是否会发生 Cache 命中。清空重载攻击的步骤包括: (a)攻击者清空和受害者共享的 Cache 区域, 等待受害者执行程序; (b)受害者执行安全操作的过程中, 将数据加载到共享区域; (c)攻击者再次重载自己访问空间, 观察

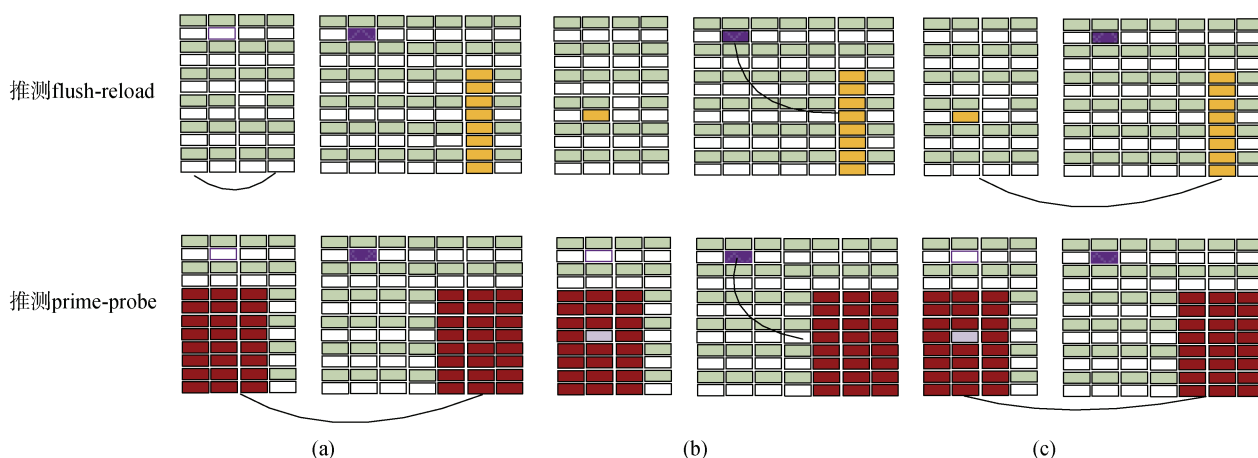


图 2 推测执行攻击的攻击流程
Figure 2 The process of speculative execution attack

是否发生 Cache 命中。如果发生 Cache 命中, 说明受害者刚刚使用到了这部分数据。

(4) flush-flush 攻击

受害者观察在清空 Cache 行的时候, 如果受害者将这部分数据载入了 Cache, 再次清空后会花费更长的时间。清空再清空攻击的步骤包括: (a)攻击者清空和受害者共享的 Cache 区域, 等待受害者执行程序; (b)受害者执行安全操作的过程中, 将数据加载到共享区域; (c)攻击者再次刷新自己访问空间, 观察清空较长时间的区域, 说明受害者刚刚使用到了这部分数据。

2.1.3 推测执行攻击

传统的 Cache 侧信道通常采用时序攻击, 利用 Cache 命中和非命中之间读取数据的时间差异, 来进行推断攻击。攻击者一般不能直接获得获取受害者数据, 通过各种攻击方法获得受害者内存访问地址, 分析与加密算法的关联性。

Spectre、Meltdown 等硬件漏洞^[35-37]突破了原有的攻击局限性, 可以通过预测执行等, 诱导密钥数据本身加载到 Cache 中, 再结合传统的 Cache 侧信道攻击方式, 对数据进行直接获取。我们将 Spectre 和 Meltdown 漏洞统称为推测执行攻击, 因为这两种攻击在 Cache 部分作用相同。利用这些硬件漏洞, 对 Cache 进行的攻击主要有两种方式, X86 系统上结合 flush-reload 的推测执行攻击^[10-11], 和结合 prime-probe 的推测执行攻击^[30]。

(1) 推测 flush-reload 攻击

攻击者清空 Cache 后, 利用系统漏洞越界执行, 将隐私信息转化成大数组的访问地址, 再根据访问情况获得密钥。推测执行清空重载攻击的步骤包括: (a)攻击者清空和受害者共享的 Cache 区域, 等待受

害者执行程序; (b)受害者通过预测执行的方式, 将密钥数据转化成大数组的访问地址; (c)攻击者再次重载定义的大数组, 观察是否发生 Cache 命中。如果发生 Cache 命中, 说明受害者刚刚使用到了这部分数据。

(2) 推测 prime-probe 攻击

攻击者是观察自己加载的大数组是否发生 Cache 失效, 访问相同地址时是否花费了更多的时间, 探索自己的数据是否被驱逐。该攻击的步骤是: (a)在 Cache 中先加载一个大数组, 使其能完成覆盖整个 Cache; (b)受害者通过预测执行的方式, 将密钥数据转化成大数组的访问地址, 该地址对应的数据被加载到 Cache 中时, 攻击者的大数组就会被驱逐; (c)攻击者 probe 自己的数据, 观察自己的数据的访问时长来判断是否发生 Cache 失效, 进而判断哪个组的查找表被受害者所使用。

2.2 防御方案介绍

当前片上系统芯片缓存的防护手段主要分成两种: 一种是基于隔离的方法, 将安全进程与其他进程所在的 Cache 空间隔离, 执行安全操作; 另一种方法是基于随机化, 在执行安全任务时, 安全防护 Cache 会随机存储或映射, 用来干扰攻击者的攻击行为。

2.2.1 隔离防护方案

缓存的隔离设计是减少受害者和攻击者之间在缓存中的交互^[38], 使安全进程隔离在自己的执行空间。基于隔离的安全 Cache 设计按隔离方式主要分为组隔离^[14]、行隔离^[15]、路隔离^[17]、内存隔离^[23]。

(1) 组隔离: 组隔离 Cache 将 Cache 存储动态地分割成受保护的区域, 彻底的减少了 Cache 之间的交互。在组隔离中, Cache 为某一应用进行专门配置,

而不是对所有应用进行优化配置。隔离 Cache 与内存之间是直接映射,它通过某些特殊的 Cache 管理指令动态的分配受保护区域。

(2) 行隔离: 对于 Cache 是以行为基本单位进行防护的,对 Cache 行增加保护位,通过行锁定的方式来进行隔离。当受害者执行安全操作时,通过指令对 Cache 行进行锁定,当攻击者使用 Cache 时,将无法映射到该 Cache 行,因此无法产生攻击。

(3) 路隔离: 动态的保持一些 Cache 行,防止被恶意驱逐。进程在使用 Cache 时,很可能会分配到多个组中,路隔离在每个组中保留一些行,保留的行数取值范围 Y 为 $[0, W]$, W 为路数。当 $Y = 0$ 时,和传统 Cache 一样没有隔离,当 $Y = W$ 时,代表着 Cache 对于现有进程是完全隔离的。随着 Y 值的波动,代表着路隔离 Cache 的性能和安全性不断折中。如果受害者使用 Cache 行的数量小于 Y ,路隔离仍然会锁定多余的 Cache 行,防止其他进程侵犯。当使用数量超过 Y 时,仍然会泄露少量信息。

(4) 内存隔离: 共享内存地址空间也给侧信道攻击埋下隐患,很多攻击通过利用这一性质对私密信息进行窃取^[31-32]。而通过隔离的方式,可以有效避免不同进程访问相同的内存地址,对系统进行保护。

2.2.2 随机化方法

侧信道攻击的特点是利用系统运行产生的侧信息,探索侧信息与安全信息之间的关联性,来进行推断攻击,Cache 侧信道的原理是利用访问 Cache 和内存时间的差异性。基于随机化的方法是目标是对访问序列随机化,打乱访问踪迹和安全操作之间的关联性,提升系统防御效果^[19-22]。

(1) 随机变化组合: 这种防护方法是允许 Cache 之间的共享的,但是进行随机化交互。它使用查找表的方式,对不同进程的采用不同的 Cache 映射方式,改变安全操作的映射方式。当受害者所要映射的组属于其他的进程,这样就会随机生成一个组,接下来的操作就会映射到这个组中,并将表格中的数据互换。

(2) 随机驱逐: 采用随机化驱逐 Cache 中的数据也是对攻击的一种干扰方式。通过每隔一段时间周期,来驱逐几个 Cache 行中的数据。

(3) 随机填充: 随机填充是在发生 Cache 失效后,一种调整预期 Cache 行的策略。当从内存载入所需 Cache 行的同时,将临近的 Cache 行也加载进来,其中前后共 m 行是以同样概率载入的,攻击者无法直接判断到哪一个 Cache 行。

3 基于 CVSS 的 Cache 侧信道攻击量化

通用漏洞评估方法(Common vulnerability scoring system, CVSS),为信息安全产业从业人员交流网络中所存在的系统漏洞的特点与影响提供了一个开放式的评价方法^[39]。CVSS 主要包括三个度量组:基础、时间和环境。基础得分组表示一个漏洞的内在特征,该得分随时间和跨用户环境保持不变,它由可执行度和影响度组成。时间得分组反应漏洞随着时间推移的影响而不受环境影响,例如,随着一个漏洞软件的补丁不断增加,该漏洞的 CVSS 分数会随之减少。环境得分组代表特定环境下执行漏洞的分数,允许根据相应业务需求提高或降低该分值^[40]。

3.1 不同攻击的评分量化

本文采用 CVSS 评分标准对 Cache 侧信道攻击进行评分量化,使用官网^[39]最新的 3.1 版本中的计算器进行计算。根据之前介绍的 Cache 攻击步骤,将攻击分成三步进行考虑(Cache 碰撞攻击只考虑攻击部分),计算每步的 CVSS 得分,最后通过加权得到每种攻击的最终得分,其中得分越高,代表该攻击的危害性更大。所有攻击的取值和得分结果如表 1 所示,之后我们依次介绍不同攻击作用下各量化指标的取值。

攻击向量(AV)反应了恶意攻击所处的条件,分数越大,代表攻击者和脆弱部件所处的安全距离越远,包括逻辑和物理距离。攻击向量的取值包括远程网络(N)、相邻网络(A)、本地(L)、物理(P)。其中 prime-probe、evict-time、flush-reload、flush-flush 攻击需要攻击者与受害者共享相同的 Cache,因此取值为(L);Cache 碰撞攻击只监测受害者加密信息,但是不能跨路由器攻击,因此选择(A);推测执行的两种攻击中第一步和第三步仍然需要共享 Cache,但 Spectre 和 Meltdown 能摆脱 Cache 的限制,通过分支预测和乱序执行来突破限制,第二步取值(A)。

攻击复杂度(AC)表示为攻击者无法控制的条件,需要攻击者提前收集受害者系统配置或计算异常等信息,包括两项取值:低复杂度(L)代表攻击者可以随意攻击,不存在惩罚;高复杂度(H)代表攻击者在攻击之前需要对脆弱组件进行大量的前期准备。推测执行攻击中,无论是 Spectre 和 Meltdown 都要进行大量的训练准备才能完成攻击,取值为(H),而其他攻击取值为(L)。

权限要求(PR)描述的是攻击者所进行攻击之前所要获得的权限层次,取值范围包括三种级别:

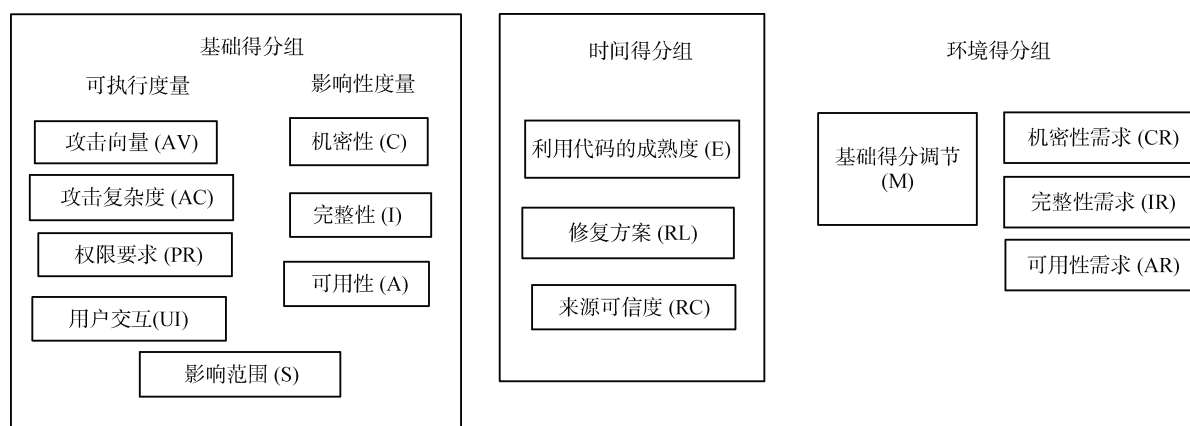


图 3 通用漏洞评分系统组成

Figure 3 The composition of a general vulnerability scoring system

无要求(N)、低权限要求(L)、高权限要求(H)。碰撞攻击需要全程监视系统加密过程, 获取加密信息, 需要的权限较高(RP:H)。其他攻击只需要触发受害者加密即可, 取值定为(L)。

用户交互(UI)描述对除攻击者之外的用户参与的需求, 即确定实施攻击是仅攻击者本身就可以随意利用, 还是需要其他用户以某种方式参与进来。包括两项取值范围: 无需求(N)和有需求(R)。Cache 侧信道攻击的过程中, 需要触发受害者进行加密, 包括碰撞攻击, 和 prime-probe、evict-time、flush-reload、flush-flush 攻击的第二步执行加密中的取值(PR:R)。

影响范围(S)反应漏洞是否会影响其以外的资源, 或者获得其以外的权限。取值范围包括固定(U)和变化(C)两种。传统的侧信道攻击主要是通过触发受害者加密来进行攻击, 而推测执行攻击是采用分支预测器和乱序执行单元来窃取密钥, 超出原本攻击范围, 因此推迟执行攻击部分取值(C), 其他攻击均取值(U)。

机密性影响(C)是指成功利用该漏洞后对系统的机密性造成的影响程度, 这里的机密性是对用户访问和披露信息的权限。取值范围包括: 毫无影响(N)、低程度影响(L)、高度影响(H)。Prime、evict、flush 等攻击步骤不会对系统机密性造成影响(C: N)。

完整性影响(I)是指成功利用该漏洞后对系统的完整性造成的影响程度, 完整是指系统信息的可靠性和准确性。取值范围包括: 毫无影响(N)、低程度影响(L)、高度影响(H)。侧信道攻击是利用系统运行的侧信息来进行关键信息推测, 并不影响系统正常运行, 所有攻击的取值为(N)。

可用性影响(A)是指成功利用该漏洞后对系统的可用性造成的影响程度, 包括数据丢失、消耗带宽和存储空间等。取值范围包括: 毫无影响(N)、低程度

影响(L)、高度影响(H)。Prime、evict 这两种攻击步骤以及 flush-flush 的最后一步攻击会将受害者部分数据驱逐回内存, 取值定为(L); 其余攻击的 flush 步骤, 会将整个 Cache 的数据清空, 对可用性影响较大(A:H); 其余的攻击步骤取值为(N)。

利用代码成熟度(E)反应漏洞被利用的程度, 即当前时刻被利用的状态, 漏洞越容易被利用得分就越高。取值范围包括: 未定义(X)、理论性存在(U)、功能性代码可以(P)、高度可用(F)。Cache 碰撞攻击主要是利用采集到的信息进行分析, 还没有可直接使用的代码进行攻击, 因此取值(U); 各个攻击的触发加密步骤和推测执行攻击的训练阶段, 需要额外的代码进行触发, 还不能直接利用, 取值为(P); 其余步骤包括 flush、prime、probe、reload、evict 等步骤, 均可采用指令直接进行, 取值定为(F)。

补丁修复水平(RL)是指现有补丁的修复水平, 随着漏洞的不断修复, 评分会因此降低。它的取值范围包括: 未定义(X)、正式修复补丁(O)、临时修复补丁(T)、非官方修复方式(W)、无可修复方案(U)。在之后我们会量化 Cache 侧信道的硬件防御效果, 此处只考虑软件补丁对攻击的影响。文献[8]中介绍 Cache 碰撞攻击通过软件的方法可以进行限制, 但还没有官方的修复方案, 因此取值定为(W); 推测执行 Spectre 和 Meltdown 官方均推出了正式补丁, 但还没法完全修复, 取值为(T); 其他攻击步骤均无法通过补丁进行限制(RL:U);

报告可信度(RC)是指该指标衡量对漏洞存在及已知技术细节的可信度, 是否被研究机构所证实。它的取值范围包括: 未定义(X)、未知(U)、可信(R)、已确认(C)。目前, 本文所列举的攻击均由正规的研究机构所发布, 已经被广泛证实, 因此该选项所有攻击均选择已确认(RC:C)。

表 1 侧信道攻击评估得分

Table 1 The evaluation scores of side channel attacks

攻击	步骤	CVSS 取值	基础得分	总得分	加权
碰撞	(b)	CVSS:3.1/AV:A/AC:L/PR:H/UI:N/S:U/C:L/I:N/A:N/E:U/RL:W/RC:C	2.4	2.2	2.20
	(a)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:L/E:F/RL:U/RC:C	3.3	3.3	
Prime-probe	(b)	CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N/E:P/RL:U/RC:C	2.8	2.7	3.09
	(c)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N/E:F/RL:U/RC:C	3.3	3.3	
Evict-time	(a)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:L/E:F/RL:U/RC:C	3.3	3.3	
	(b)	CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N/E:P/RL:U/RC:C	2.8	2.7	3.10
	(c)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N/E:F/RL:U/RC:C	3.3	3.3	
Flush-reload	(a)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H/E:P/RL:U/RC:C	5.5	5.2	
	(b)	CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N/E:P/RL:U/RC:C	2.8	2.7	3.66
	(c)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N/E:F/RL:U/RC:C	3.3	3.3	
Flush-flush	(a)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H/E:P/RL:U/RC:C	5.5	5.2	
	(b)	CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N/E:P/RL:U/RC:C	2.8	2.7	3.60
	(c)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:L/E:P/RL:U/RC:C	3.3	3.2	
Spec-prime	(a)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:L/E:F/RL:U/RC:C	3.3	3.3	
	(b)	CVSS:3.1/AV:A/AC:H/PR:L/UI:N/S:C/C:H/I:N/A:L/E:P/RL:T/RC:C	6.5	5.9	4.97
	(c)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N/E:F/RL:U/RC:C	5.5	5.4	
Spec-flush	(a)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H/E:P/RL:U/RC:C	5.5	5.2	
	(b)	CVSS:3.1/AV:A/AC:H/PR:L/UI:N/S:C/C:H/I:N/A:L/E:P/RL:T/RC:C	6.5	5.9	5.54
	(c)	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N/E:F/RL:U/RC:C	5.5	5.4	

环境评价是指用户在特定条件下对评分标准进行设定,包括机密性要求(CR)、完整性要求(IR)、可用性要求(AR),以及其他调整攻击向量(M)。本论文中对 Cache 侧信道攻击的量化环境属于通用环境,不对环境进行额外要求,因此环境评分组均不进行额外选取。

3.2 攻击结果对比

我们采用了 CVSS 评分标准对不同 Cache 侧信道攻击的每个攻击步骤进行量化评估,接下来我们将对攻击整体得分进行加权获得。除了 Cache 碰撞攻击之外的其他攻击均由三步组成,需要通过三步攻击得分来获得每种侧信道攻击的总得分。我们将采用加权平均的方式,赋予每个攻击步骤不同的权重 ω 。因此攻击的 CVSS 得分可以由下面公式求取:

$$Score(attack) = \sum_{i=1}^3 \omega_i \cdot score(attackstep_i) \quad (2)$$

权重规则:在对 Cache 侧信道攻击原理分析基础上,根据攻击步骤的危害程度,赋予其相应的权重。在三步攻击中,如果仅有一步对攻击影响偏大,该步骤分配权重 $\omega = 0.4$,其他两步权重 $\omega = 0.3$;如果攻击中某两步对攻击成功影响较大,该两步得分权重为 $\omega = 0.35$,其他步权重为 $\omega = 0.3$;而对于每个步骤所占权重相当,采用加和平均的方式求得。

根据上述规则我们赋予攻击权重如下: prime-

probe 设为 $[0.3, 0.35, 0.35]$, evict-time 为 $[1/3, 1/3, 1/3]$, 而 flush-reload 为 $[0.3, 0.35, 0.35]$, flush-flush 为 $[0.3, 0.4, 0.3]$, 推测 prime 攻击为 $[0.3, 0.4, 0.3]$, 推测 flush 攻击为 $[0.3, 0.4, 0.3]$ 。因此,结合公式 2,我们求出每种 Cache 侧信道攻击的加权得分如表 1 最右列所示。

排名最高的依然是推测执行的两种攻击,说明相关硬件漏洞对 Cache 造成的风险非常大,根据 CVSS 的得分评级,这两种攻击均为中级。而推测 flush-reload 攻击比推测 prime-probe 攻击具有更高得分,危害性更大。其他五种攻击得分均为 CVSS 低级,其中 flush-reload 和 flush-flush 得分相对更高,主要因为 flush 步骤是系统自身功能,被攻击者利用之后造成的危害性更大。此外,其余的攻击得分排名分别为 evict-time 攻击约等于 prime-probe 攻击,Cache 碰撞攻击得分最低。

4 Cache 侧信道防御评估建模

在之前章节,我们已经通过 CVSS 漏洞评估方法得到了各个攻击的得分,接下来我们将对防御方案进行建模,获得各个防御方案的得分情况。

本文所提出的整体评估流程如图 4 所示,目的是为了评估不同防御方法在不同攻击下的得分情况。如图左侧,不同攻击被拆解为按攻击步骤进行量

化, 利用 CVSS 漏洞评估获得步骤得分, 并对得分进行加权求和, 即为每种攻击得分情况, 我们在第三节内容对攻击得分进行了计算。之后, 我们对攻击和防御进行建模, 利用 Cache 机理模型来描述攻击和防御过程, 并引入攻击成功概率, 来量化攻击和防御。最后, 结合攻击得分以及防御对攻击成功概率的改变, 来得到防御在攻击下的得分情况。

表 2 符号定义

Table 2 Symbol definition

符号	含义
Pr	某种攻击的攻击成功概率
A_j	事件采用攻击方法 j
D_i	事件防御方法 i
$D_i A_j$	事件攻击 j 能够成功前提下采用防御 i
$D_i \cap A_j$	事件攻击 j 和防御 i 共同作用

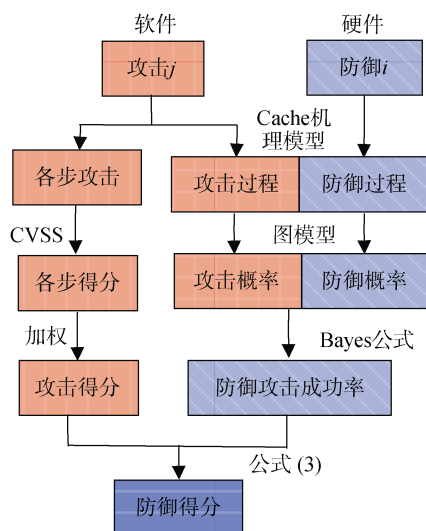


图 4 评估整体流程

Figure 4 The process of evaluation

4.1 攻击防御建模推导

在本节, 我们将对攻击和防御评估方法进行原理推导。首先引入攻击概率得分, 建立评分与攻击效果的关系; 由于攻击效果是由攻击防御共同作用, 引入了贝叶斯公式来解决攻击和防御的相互博弈性。最后通过贝叶斯公式中的后验概率部分, 结合攻击能力同等性, 来对 Cache 侧信道防御方法进行评估。

4.1.1 攻击成功概率得分

我们在之前章节使用 CVSS 计算了 Cache 侧信道攻击的得分 $Score(attack)$ 。我们求解这个攻击得分时, 考虑的是这个攻击已经攻击成功, 并且未采取相关硬件防御措施的情况下, 通过 CVSS 评分标

准进行评价得到每个攻击的得分, 代表该种攻击的危害程度。

然而我们在实际攻击过程中并不能保证每次攻击能百分百完成, 产生原因包括: (1)攻击者自身攻击能力不足, 只能完成部分攻击窃取部分私密信息, 或者攻击成功只能概率性完成; (2)采用有效安全防御方法后, 原本有效的攻击会改变攻击成功概率, 或者攻击失效。

因此, 在本文模型中我们定义了一个利用攻击概率来评判攻击防御得分参数: 攻击成功概率得分 SPAS, 利用该参数对 Cache 侧信道攻击和防御进行整体评估。不同攻击成功概率 Pr 下的得分:

$$SPAS = Score(attack) \cdot Pr \quad (3)$$

Pr 可以表示不同程度的攻击效果, 取值范围为 $[0,1]$ 。例如在不考虑硬件防御方法时, 并且该种攻击都在正常情况下都能攻击成功时, 这时成功概率 $Pr=1$, $Score(Pr)=Score(attack)$; 当攻击完全失败或防御完全成功, $Pr=0$, 此时得分为 0; 而当攻击成功概率 $Pr=0.5$ 时, $Score(Pr)=0.5 \cdot Score(attack)$, 不同的攻击成功率会获得不同的得分。

4.1.2 攻击和防御对成功概率影响

由于攻击和防御是一个博弈过程, 攻击成功概率同时受攻击能力和防御能力影响, 单单考虑攻击或防御都无法准确评估防御方法的有效性, 我们将通过简单攻击模型对此进行阐述。

以骰子游戏为基础, 骰子包含六面, 分别为 1~6 的数字, 攻击者和防御者每个人都展示一种点数, 若点数相同即为攻击成功。攻击者采取随机猜的形式对骰子点数进行猜测攻击, 写下猜测点数。防御者在知道攻击者所写点数的情况下, 采用随机化的防御方法, 通过掷骰子方式展示骰子点数, 从防御者的角度这种防御方式能使攻击者 $1/6$ 的概率攻击成功。然而, 攻击者角度来看, 因为是随机点数攻击, 攻击成功概率即为 $1/6$, 防御者的防御方法并没有起到效果。

所以, 在对防御方法的有效性评估时, 当攻击有概率发生时, 并不能准确的评判防御的有效性。因此我们在对 Cache 侧信道防御方法评估中, 需要考虑攻击概率对防御结果的影响。

4.1.3 贝叶斯概率模型

为了考虑攻击防御的博弈性, 我们使用了贝叶斯模型。虽然攻击和防御的发展是螺旋上升, 但防御方法更多是为了应对侧信道攻击所带来安全缺陷, 即攻击先于防御产生。因此本文采用了贝叶斯公式中的后验概率部分, 来计算 Cache 侧信道攻击和防

御的有效性关系。

$$Pr = P(D_i \cap A_j) = P(A_j) \cdot P(D_i | A_j) \quad (4)$$

$P(A_j)$ 代表这种攻击在没有防御情况下的攻击成功概率, 为攻击能力对总体攻击成功概率的影响; $P(D_i | A_j)$ 代表在攻击成功条件下采取防御手段后的攻击成功率, 为防御能力对总体攻击成功概率的影响。通过公式 4, 我们可以得到防御对攻击成功率的影响。

在上节骰子游戏中, 攻击自身概率 $P(A_j)=1/6$, 增加防御方法后攻击者的攻击成功概率 $P(D_i \cap A_j)$, 所以 $P(D_i | A_j)=1$, 采取防御手段并没有改变攻击成功概率。当 $P(D_i | A_j) < 1$, $P(D_i \cap A_j) \neq P(D_i | A_j)$, 因此在不考虑自身攻击概率情况下, 防御效果不能根据攻击成功概率直接进行评估。

4.1.4 防御评估方法

为了对 Cache 侧信道攻击防御进行有效评估, 我们采用 CVSS 漏洞评估和攻击成功概率相结合的方式来进行量化分析。由公式(3)和公式(4), 当 $Pr = P(D_i \cap A_j)$ 代入公式(3), 即可求出攻击在防御作用下的评分。然而, 正如我们在前一节的介绍, 模型中所得到的评分不能直接代表防御效果评分, 因为参数中还有攻击自身的成功概率 $P(A_j)$ 。

相关工作中, 对于 Cache 侧信道防御的评估模型中, 并没有考虑各种攻击自身攻击能力的差距。在本文中, 我们赋予每种攻击相同程度的攻击能力, 令每种的自身攻击成功概率 $P(A)=1$, 防御能力的得分就变为在攻击能百分之百成功的基础上, 增加防御能力对攻击成功概率的改变。

此时, 对于防御能力的评估, 只由采用防御方法后的攻击概率 $P(D_i | A_j)$ 决定。由于 $P(D_i | A_j)$ 是定义攻击成功前提下, 初始值为 1, 当增加防御方法后, 根据防御方法的有效性, $Pr = [1 \rightarrow 0]$, 攻击成功概率得分也会相应减少。因此防御得分即为增加防御后, 攻击成功概率得分的差值, 即:

$$\begin{aligned} Score(defend) &= Score(attack) - Score(attack) * P(D_i | A_j) \\ &= Score(attack) \cdot [1 - P(D_i | A_j)] \end{aligned} \quad (5)$$

我们通过本节阐述了 Cache 侧信道防御评估方法。将通过 CVSS 和攻击成功前提下添加防御方法

后攻击成功概率, 来获得防御评估得分。在将针对具体的 Cache 侧信道攻击和防御方法展开建模, 去求解不同攻击在不同防御方法下的攻击成功概率。

4.2 基于 Cache 机理的攻防建模

我们根据 Cache 侧信道的攻击原理^[41-42], 分析侧信道的攻击流程, 对侧信道各攻击和防御进行模型表达, 来获得不同攻击在不同防御方法下的攻击成功概率, 进而对不同防御方法有效性进行评估。

4.2.1 基于图模型的 Cache 运行机理建模

Cache 的设计目的、地址映射、局部性原理等基础知识我们已经在第二节结束, 为了能更好的表达 Cache 运行机理, 我们使用图模型对其进行建模。

Cache 运行过程中, 包含四个过程: 访存地址进行组查找、Cache 行索引选择、Cache 行读写操作、内存读取数据等。通过图模型可以有效表达 Cache 运行的这一系列操作。

节点集合 $V = \{V_1, V_2, V_3, V_4, V_5\}$, 节点代表执行操作时相应的物理模块, V_1 为访存地址, V_2 为 Cache 地址映射机构, V_3 为 Cache 行查找, V_4 为 Cache 数据操作, V_5 为内存, 节点在模型中并不独立使用, 主要是通过路径和路径所通过的概率来进行建模和计算。

边界集合 $E = \{E_1, E_2, E_3, E_4\}$, 边界代表 Cache 执行的操作流程, E_1 是 Cache 访问过程中由内存地址映射到相应组, E_2 是 Cache 索引映射到 Cache 行的路径, E_3 是将 Cache 行与内存数据进行驱逐或载入操作, E_4 是数据在 Cache 和内存之间传输路径。

该模型用于表示 Cache 运行的基本流程, 为了面向于 Cache 侧信道攻击和防御建模, 我们将在之后进行模型扩展。

4.2.2 图模型的 Cache 侧信道流程建模

接下来我们对侧信道攻击和防御进行建模描述, 在 Cache 基本模型的基础上, 我们新增一个节点 V_0 以及边界 E_0 , V_0 表示外力能触发受害者执行的操作, E_0 表示加密、分支预测、乱序执行等操作。增加新节点的目的是在 Cache、内存之后, 还有其他操作影响着 Cache 侧信道攻击的实施, 但并不是所有的侧信道攻击包含这部分结构。

对于边 E_3 、 E_4 为数据通路边界, 由于数据流向是双向的, 因此每个边界包含两种数据方向的边界。后标 1 代表数据流出 Cache 流入内存方向, 后标 2 代表数据流出内存流入 Cache 方向, $E_3 = \{E_{31}/E_{32}\}$,

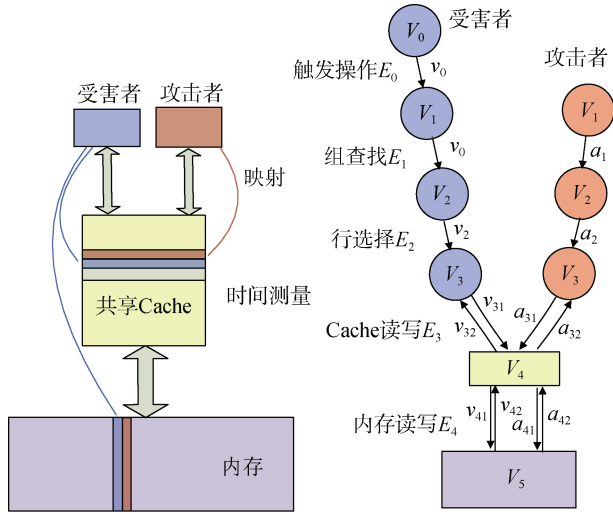


图 5 基于图模型的 Cache 建模

Figure 5 Cache modeling based on graph model

$E_4 = \{E_{41}/E_{42}\}$ 。因此, 针对 Cache 操作流程的通用图模型可表示为:

$$E = \{E_0/\text{null}, E_1, E_2, E_{31}/E_{32}, E_{41}/E_{42}\} \quad (6)$$

需要指出, 图模型的使用要满足下面原则: E_0 并不存在所有攻击步骤中, 当不存在时只有四个步骤; E_3 和 E_4 要满足数据流向一致性, 即 E_{31} 和 E_{41} 、 E_{32} 和 E_{42} 均需同时出现; 虽然数据具有方向性, 但图模型是代表操作流程, 模型中所有边均具有同向性。

针对 Cache 侧信道, 需要攻击者和触发受害者轮流进行 Cache 访问操作, 我们使用图模型对攻击操作进行表达, 就可以通过模型表达 Cache 侧信道攻击流程。记受害者为 v , 攻击者为 a , 攻击者和受害者的访问流程表示为:

$$E_v = \{v_0/\text{null}, v_1, v_2, v_{31}/v_{32}, v_{41}/v_{42}\} \quad (7)$$

$$E_a = \{a_0/\text{null}, a_1, a_2, a_{31}/a_{32}, a_{41}/a_{42}\} \quad (8)$$

通过 E_v 和 E_a , 我们可以表达所有侧信道攻击步骤的攻击流程, 对于完整 Cache 信道攻击可以表达为由多个 E_v 或 E_a 串联而成。

4.2.3 攻击成功概率建模

首先, 我们探索公式 6 中通用图模型的攻击成功概率。在 Cache 访问操作过程中, 每个步骤之间满足局部马尔科夫特性, 并且每个操作步骤相互独立^[13]。因此, 在图模型中, 图的整体边界概率等于每个边界概率的乘积, 即:

$$P(E) = P(E_0) \cdot P(E_1) \cdot P(E_2) \cdot P(E_3) \cdot P(E_4) \quad (9)$$

式中模型参数 E 在表示侧信道攻防建模时, 对应公式 7 中 E_v 或 E_a 以及相应的子集。对于每个攻击

步骤, 攻击成功概率就等于每步操作成功概率的乘积。

而在对侧信道攻击整体描述时, 除 Cache 碰撞之外的攻击都包括三个步骤。因为每个步骤攻击仍然相互独立, 所以整体的攻击成功概率等于所有攻击步骤的攻击成功概率乘积。与第三节对攻击评分需要对每个步骤加权处理不同, 在考虑攻击成功率时每个步骤都需要攻击成功, 某一步骤攻击失败导致整体攻击失败, 因此在计算整体攻击成功概率时不需对每步加权处理。结合公式(4)中 $P(D_i|A_j)$, 得到每种攻击的攻击成功概率是构成该攻击图模型的所有边界攻击成功概率之积。

$$P(D_i|A_j) = \prod P(E) \quad (10)$$

本文中的模型是通过图模型来构造 Cache 侧信道攻击流程, 以及根据边界概率来构造攻击的攻击成功概率。而侧信道防御方法的作用就是采取手段限制攻击者进行攻击, 因此本模型中可以通过防御对其对应边界攻击成功概率的改变来表达防御效果, 对增加防御之后求解的 $P(D_i|A_j)$ 值代入公式 5 中, 就可得到防御在攻击下的得分。

4.3 侧信道防御方法建模与求解

在本节, 我们将对 2 节中列举的所有侧信道攻击进行详细建模, 将攻击步骤转化成图模型; 然后使图模型中所有边界初始攻击成功率均为 1, 在假设初始攻击成功的基础下增加防御方案并对模型求解, 得到增加防御后的攻击成功概率; 最后结合 CVSS 评分对防御方法进行评分, 得到防御有效性评估结果。

表 3 符号定义与赋值

符号	含义	取值
N	总 Cache 行数	512
S	组数	64
W	每组路数	8
m	随机填充的对话框	128
X	每轮随机驱逐的行数	8
Y	路隔离保留的行数	2

添加防御措施后, 不同的防御方法会影响不同的边界, 改变边界的攻击成功概率。在探讨防御方法对边界作用时, 我们在模型中引入“&”算子, $a_1 \& v_1$ 表示攻击者和受害者能同时映射到相同的组, $a_{32} \& v_{31}$ 则表示攻击者能准确驱逐受害者的数据。因

为攻击者在写入或驱逐数据时,需要同时伴随受害者的数据驱逐和载入。一些攻击步骤只有交互成功才能发生,同样的某些防御只对交互的攻击步骤起到效果。

组隔离和组置换作用于边界 E_1 , 组隔离是阻止攻击者映射到受害者的组, 组隔离影响模型中边 $a_1 \& v_1$; 组置换是受害者对自身组映射的随机化, 影响攻击者结果推测部分。路隔离和 newcache 影响边界 E_2 , 路隔离每组保存的行数 $Y=2$, 可以覆盖加密时每次访问时关键数据, 使攻击者无法映射到保留的行, 影响边 $a_2 \& v_2$; Newcache 使行选择随机化, 概率为 $1/N$ 。行锁定影响边界 E_3 的交互操作, 防御生效后, 攻击者无法读取或驱逐受害者的数据。随机填充和随机读取分别为边界 E_3 中 a_{32} 、 a_{31} 的保护, 干扰攻击行为。内存隔离影响边界 E_4 , 防止攻击者访问受害者内存空间的数据。边界 E_0 是 Cache 外建模, 目前还未有相关防御方法。由于利用系统 flush 操作产生多种攻击, 所以增加 flush 功能失效方法。

4.3.1 碰撞攻击

与其他攻击不同, 碰撞攻击相当于受害者自己访问, 利用两次访问的关系推断, 因此只有执行阶段, 攻击成功概率只计算执行阶段加密成功概率。对碰撞攻击的建模如图 6 所示, 利用触发受害者访问进行攻击, 攻击模型为 $E_v = \{v_0, v_1, v_2, v_{32}, v_{42}\}$ 。

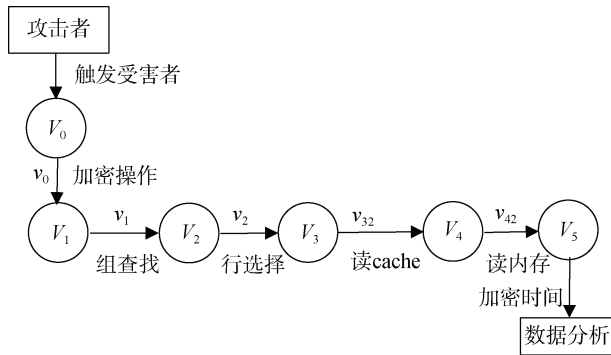


图 6 Cache 碰撞攻击建模

Figure 6 Model Cache collision attack

根据贝叶斯公式, 由表 1 中我们可以得到 Cache 碰撞攻击的攻击得分为 2.2, 在计算这种攻击在各种防御方案下的成功概率 Pr 后, 此时的防御得分可以由 $2.2 \cdot (1 - Pr)$ 所求。

我们计算防御方法对 Cache 碰撞攻击的概率改变, 结果如表 4 所示。由于 Cache 碰撞攻击在攻击过程中并不干扰加密的正常运行, 因此除了随机驱逐

表 4 Cache 碰撞攻击结果

Table 4 Results of Cache collision attack

方法	执行	Pr	防御得分
组隔离	1	1	0
行锁定	1	1	0
路隔离	1	1	0
内存隔离	1	1	0
组置换	1	1	0
随机填充	$1/(m+1)$	$1/(m+1)$	2.18
随机驱逐	$1-X/N$	$1-X/N$	0.03
Newcache	1	1	0
无 flush	1	1	0

和随机填充之外的其他防御方案的防御效果均无效, 因此攻击成功概率 $Pr=1$ 。随机填充提前预取了 $m+1$ 个数据块, 影响边界 v_{32} , 对于 Cache 碰撞攻击的攻击成功概率 $P(v_{32})$ 改变为 $1/(m+1)$, 而其他边界概率依然为 1, 因此总的概率也为 $1/(m+1)$ 。随机驱逐可以将之前预取的数据随机的驱逐出去, 攻击成功概率改为 $1-X/N$ 。

4.3.2 prime-probe

我们来计算 prime-probe 攻击的在不同防御下的攻击成功概率和防御得分。包括 prime、执行、probe 三个步骤进行计算。在之前的计算中, prime-probe 的攻击得分为 3.09。Prime-probe 攻击模型如下图所示:

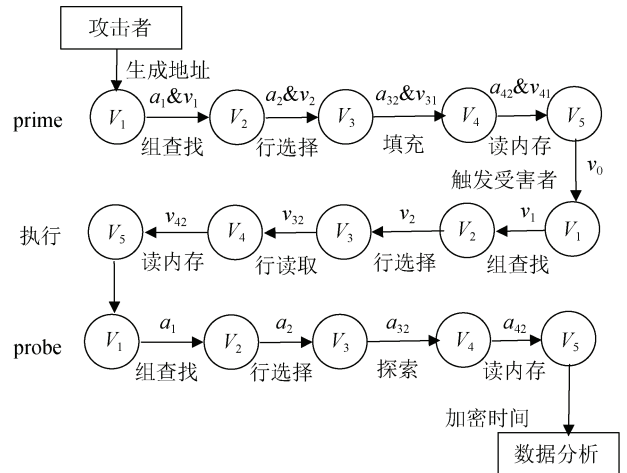


图 7 Prime-probe 攻击建模

Figure 7 Model prime probe attack

接下来我们对模型进行求解。在 prime 阶段时, 攻击者需要采用数据填充受害者的 Cache 访问空间, 防护的关键在于阻止受害者数据被攻击者数据交互, 组隔离、行锁定、路隔离能分别时路径 $a_1 \& v_1$ 、 $a_2 \& v_2$ 、 $a_{32} \& v_{31}$ 为概率 0, 因此防御概率为 0。与文献[27]相比, 由于本文设定所有攻击成功概率为 1,

需要填充所有的组, 所有组置换不能改变攻击成功概率。

执行加密过程, 除了随机填充之外没有防御方法保护触发受害者的加密操作, 攻击者可以正常完成此步骤的攻击, 不改变先验概率。而随机填充的攻击概率为 $1/(m+1)$ 。Probe 阶段, 攻击者再次访问数据看是否发生 Cache 失效, 组置换和 newcache 使攻击者探索到的数值是随机的, 所以他们两个的攻击成功概率分布为 $1/S$ 和 $1/N$ 。代入公式后, 防御得分结果如下:

表 5 Prime probe 攻击结果

Table 5 Results of prime probe attack

方法	填充	执行	探索	Pr	防御得分
组隔离	0	1	1	0	3.09
行锁定	0	1	1	0	3.09
路隔离	0	1	1	0	3.09
内存隔离	1	1	1	1	0
组置换	1	1	$1/S$	$1/S$	3.04
随机填充	1	$1/(m+1)$	1	$1/(m+1)$	3.07
随机驱逐	1	1	1	1	0
Newcache	1	1	$1/N$	$1/N$	3.08
无 flush	1	1	1	1	0

4.3.3 flush-reload

攻击者利用 X86 架构存在的 flush 功能清空系统数据, 再触发受害者加密, 通过访问 Cache 判断攻击者访问了哪个数据, 进而推断受害者信息, 对 flush-reload 攻击建模如图 8。

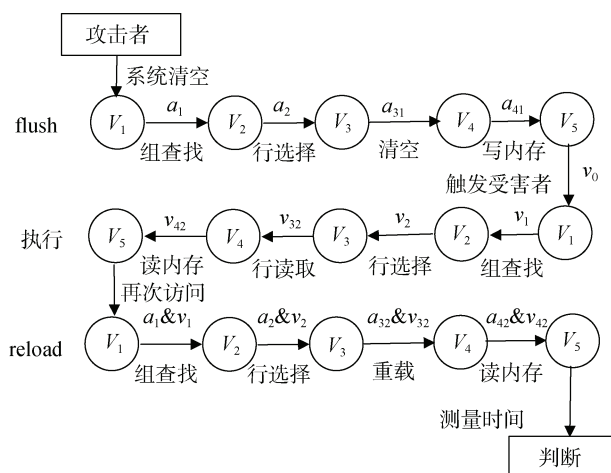


图 8 Flush-reload 攻击建模

Figure 8 Model flush-reload attack

Flush 阶段, 是因为 X86 系统存在的功能, 采用清空 Cache 的手段进行攻击。传统的隔离方法在

Flush 清空之后, 就没有隔离了, 受害者访问一次, 只隔离了一个组, 即使攻击者不能读, 也能判断在哪个组。除了禁用 flush 功能之外, 没有有效手段来解决这一攻击步骤。

表 6 Flush-reload 攻击结果

Table 6 Results of flush-reload attack

方法	清空	执行	重载	Pr	防御得分
组隔离	1	1	1	1	0
行锁定	1	1	1	1	0
路隔离	1	1	1	1	0
内存隔离	1	1	0	0	3.66
组置换	1	1	$1/S$	$1/S$	3.60
随机填充	1	$1/(m+1)$	1	$1/(m+1)$	3.63
随机驱逐	1	$1-X/N$	1	$1-X/N$	0.06
Newcache	1	1	$1/N$	$1/N$	3.65
无 flush	0	1	1	0	3.66

执行加密过程, 除了随机填充之外没有防御方法保护触发受害者的加密操作, 攻击者可以正常完成此步骤的攻击, 不改变先验概率。而随机填充的攻击概率为 $1/(m+1)$, 随机驱逐攻击成功概率改为 $1-X/N$ 。Reload 阶段和 probe 类似, 攻击者再次访问数据看是否发生 Cache 命中, 组置换和 newcache 使攻击者探索到的数值是随机的, 攻击成功概率分布为 $1/S$ 和 $1/N$ 。根据 flush-reload 的攻击得分为 3.66, 防御得分计算结果见表 6。

4.3.4 evict-time

驱逐计时攻击通过驱逐受害者加密的数据, 并再次出发受害者加密, 观察是否发生 Cache 失效, 驱逐计时攻击的攻击模型如图 9。

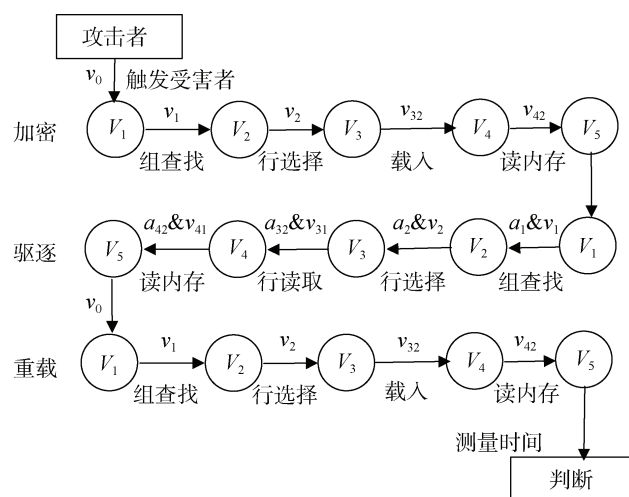


图 9 Evict-time 攻击建模

Figure 9 Model evict-time attack

驱逐计时的攻击得分为 3.10, 计算过程如下: 触发受害者加密时, 防御方法都无法对攻击行为进行干扰, 攻击成功概率并没有受到改变, 依然为 1。随机填充方法与在其他攻击防御能力不同, 由于预取的数据被受害者进行清空驱逐, 所以并不能防御。

表 7 Evict-time 攻击建模
Table 7 Results of evict-time attack

方法	驱逐	执行	加载	Pr	防御得分
组隔离	1	0	1	0	3.10
行锁定	1	0	1	0	3.10
路隔离	1	0	1	0	3.10
内存隔离	1	1	1	1	0
组置换	1	1/S	1	1/S	3.05
随机填充	1	1	1	1	0
随机驱逐	1	1	1	1	0
Newcache	1	1/N	1	1/N	3.09
无 flush	1	1	1	1	0

在驱逐数据的过程中, Cache 隔离方案都能保护受害者数据, 防止攻击者驱逐, 攻击成功概率为 0; 组置换和 newcache 使攻击者驱逐受害者的数值是随机的, 并不能准确判断驱逐的是受害者的哪部分数据, 所以他们攻击成功概率分布为 $1/S$ 和 $1/N$ 。

4.3.5 flush-flush

通过采用系统刷新功能进行攻击, 先清空 Cache 内的数据, 受害者执行加密操作, 再清空 Cache, 根据清除的数据的时间来推断地址信息。准备阶段的 flush 操作, 也是由于 X86 系统存在的功能, 只有禁用其功能才能对攻击进行防护。加密过程中随机填充的攻击概率为 $1/(m+1)$, 随机驱逐攻击成功概率改为 $1-X/N$ 。

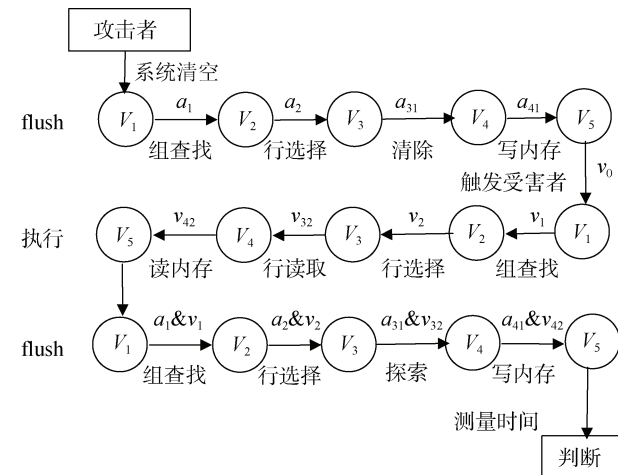


图 10 Flush-flush 攻击建模
Figure 10 Model flush-flush attack

与第一个 flush 操作不同, 此操作需要逐行地址进行刷新操作, 根据刷新时间快慢来判断此行是否被使用。组置换和 newcache 使受害者读取数值所在位置是随机的, 并不能准确判断驱逐的是受害者的哪部分数据, 所以他们的攻击成功概率分布为 $1/S$ 和 $1/N$ 。而 flush 失效操作同样能使此步骤攻击成功概率为 0。根据攻击得分为 3.60, 计算结果如表 8。

表 8 Flush-flush 攻击建模
Table 8 Results of flush-flush attack

方法	清空	执行	清空	Pr	防御得分
组隔离	1	1	1	1	0
行锁定	1	1	1	1	0
路隔离	1	1	1	1	0
内存隔离	1	1	1	1	0
组置换	1	1	1/S	1/S	3.54
随机填充	1	$1/(m+1)$	1	$1/(m+1)$	3.57
随机驱逐	1	$1-X/N$	1	$1-X/N$	0.06
Newcache	1	1	1/N	1/N	3.59
无 flush	0	1	0	0	3.60

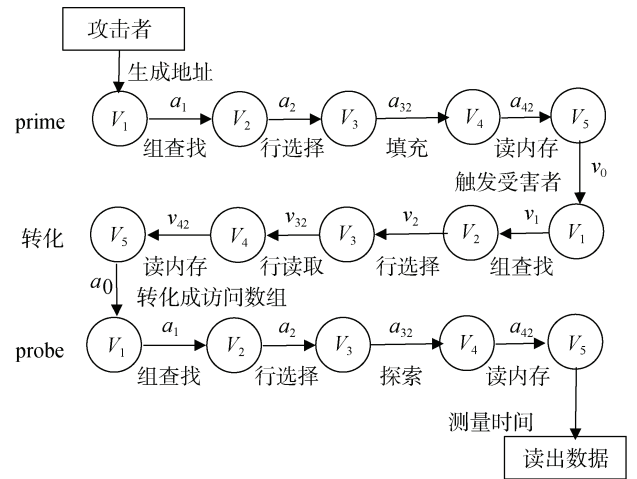


图 11 推测 prime-probe 攻击建模
Figure 11 Model speculative prime-probe attack

4.3.6 推测 prime-probe

推测执行攻击是通过分支预测、乱序执行等手段来对受害者数据进行转移, 因此防御难度较大。推测 prime 阶段, 与传统 prime 攻击不同, 推测执行因为能诱导受害者执行数据转入自己的地址空间, 因此受害者只需在自己的 Cache 空间填充数据即可。对 prime-probe 攻击有效的隔离防御方法失效了, 这一阶段目前没有好的防御方法。

推测执行阶段, 由于 Spectre 和 Meltdown 是系统漏洞, Cache 相关的防御方法还不能解决, 内存隔离

是个好的办法, 来限制数据访问。

Probe 阶段, 由于是攻击者在自己的访问空间进行查找, 因此同样没有合适的办法进行防御。系统本身的随机驱逐, 可以限制一下攻击者访问。推测 prime-probe 的攻击得分 4.97, 计算结果如表 9。

表 9 推测 prime-probe 攻击建模

Table 9 Results of speculative prime-probe attack					
方法	填充	转化	探索	Pr	防御得分
组隔离	1	1	1	1	0
行锁定	1	1	1	1	0
路隔离	1	1	1	1	0
内存隔离	1	0	1	0	4.97
组置换	1	1	1	1	0
随机填充	1	1	1	1	0
随机驱逐	1	1	1-X/N	1-X/N	0.08
Newcache	1	1	1	1	0
无 flush	1	1	1	1	0

4.3.7 推测 flush-reload

推测执行攻击是通过系统 flush 功能, 以及分支预测、乱序执行等, 防御难度同样较大。

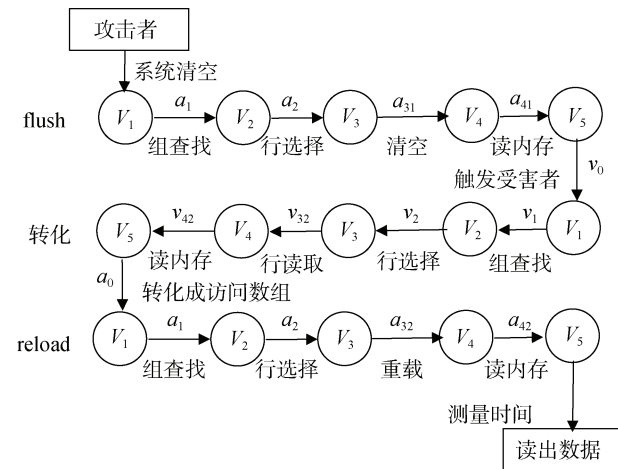


图 12 推测 flush-reload 攻击建模

Figure 12 Model speculative flush-reload attack

和 flush-reload 攻击一下, 使 flush 功能失效, 是解决攻击者清空 Cache 的最好方法。推测执行阶段, 现有防御方法中, 只有内存隔离可以限制数据越界访问等问题。推测 flush-reload 的攻击得分是 5.54, 防御结果如表 10 所示。

5 实验结果分析

5.1 防御方案评分对比

我们采用 CVSS 漏洞评分标准和贝叶斯模型,

表 10 推测 flush-reload 攻击建模

Table 10 Results of speculative flush-reload attack					
方法	清空	转化	加载	Pr	防御得分
组隔离	1	1	1	1	0
行锁定	1	1	1	1	0
路隔离	1	1	1	1	0
内存隔离	1	0	1	0	5.54
组置换	1	1	1	1	0
随机填充	1	1	1	1	0
随机驱逐	1	1	1	1	0
Newcache	1	1	1	1	0
无 flush	0	1	1	0	5.54

计算了 Cache 侧信道防御方法在不同攻击方案下的得分情况。

5.1.1 有效性对比

根据之前求得的不同防御得分, 我们来分析防御对攻击的有效性。通过得分情况, 我们将防御效果分成三个等级, 其中大于 1 认为很有效用“√”表示, 得分等于 0 时完全没有效果表示为“×”, 得分介于 0 和 1 之间认为只是起到限制作用采用“—”表示。

表 11 防御的有效性对比

Table 11 Comparison of defense effectiveness							
	碰撞	pp	flu	驱逐	ff	spr	sfr
组隔离	×	√	×	√	×	×	×
行锁定	×	√	×	√	×	×	×
路隔离	×	√	×	√	×	×	×
内存隔离	×	×	√	×	×	√	√
组置换	×	√	√	√	√	×	×
随机填充	√	√	√	×	√	×	×
随机驱逐	—	×	—	×	—	—	×
Newcache	×	√	√	√	√	×	×
Flush 失效	×	×	√	×	√	×	√

几种防御方法中, 组置换、随机填充、newcache 是有效性最高的防御方法, 可以免疫现有常见 Cache 侧信道攻击中的 4 种。内存隔离和 flush 失效能解决 3 种现有侧信道攻击。随机驱逐可以对 4 种攻击具有防御效果, 但只是有限制作用。组隔离、行锁定、路隔离只能解决两种攻击, 易受其他攻击攻破。

5.1.2 总体对比

我们来探索防御总体评分情况, 将各防御方法在面对不同攻击下的得分情况进行累加, 对比结果如图 12 所示:

从总得分角度对几种防御方法进行对比, 内存隔离具有最高的得分, 因为它可以有效的减少数据

表 12 联合防御效果
Table 12 Effects of combined defenses

	碰撞	pp	flu	驱逐	ff	spr	sfr	总得分
组置换+ Flush 失效	×	✓	✓	✓	✓	×	✓	18.89
组隔离/行锁定+ Flush 失效	×	✓	✓	✓	✓	×	✓	18.99
随机填充+ Flush 失效	✓	✓	✓	×	✓	×	✓	18.05
随机填充+内存隔离	✓	✓	✓	×	✓	✓	✓	22.99
Newcache+内存隔离	×	✓	✓	✓	✓	✓	✓	23.93
随机填充+行锁定+内存隔离	✓	✓	✓	✓	✓	✓	✓	26.11

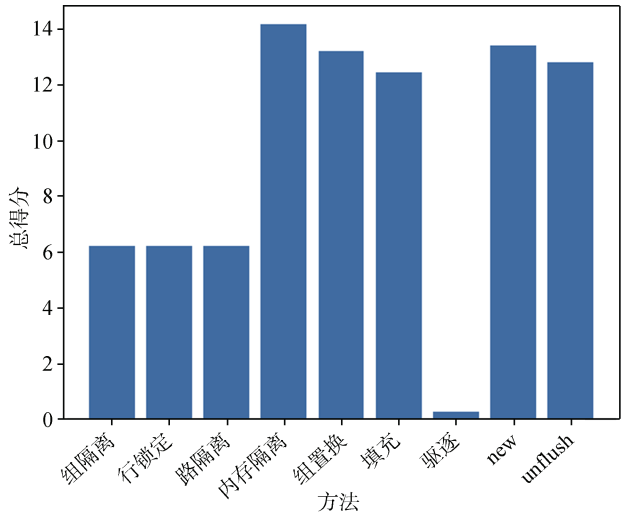


图 13 每种防御方法总得分
Figure 13 Total score of each defense method

在内存中的交互。Newcache 以及组置换的方法都是随机化地址信息的映射, 干扰攻击行为, 也获得很高分数。Flush 失效方法可以针对性的消除几种 flush 相关攻击, 随机填充可以有效抵御访问和碰撞攻击, 得到的分数同样很高。组隔离、行锁定、路隔离只能解决特定攻击, 分数中等水平。随机驱逐只能对一些攻击进行限制, 因此得到的分数最低。

5.2 组合防御探讨

前面我们分析了某种攻击在某种防御方案下的攻击成功概率, 属于一对一防御, 但目前还没有完美的防御方法来抵御所有的 Cache 侧信道攻击, 接下来我们继续采用贝叶斯模型, 来探索联合防御来抵挡现有侧信道攻击。

组置换+flush 失效: 组置换和 flush 失效都是实现较为简单的防御方法, 两组防御方法组合起来可以解决五种 Cache 侧信道攻击, 但仍然无法防御碰撞攻击和推迟 prime-probe 攻击。

组隔离/行锁定+flush 失效:和组置换类似, 组隔离和行锁定也能解决 flush 失效所不能防护的 prime-probe 攻击和驱逐计时攻击, 因此组合起来也

能解决 5 种攻击。

随机填充+flush 失效: 随机填充是能防护攻击种类比较多的方法, 加上易于实现的 flush 失效方案, 可以有效进行 5 种攻击的防护。

随机填充+内存隔离:这两种防御方案的结合, 能有效解决 6 种侧信道攻击, 但不能解决随机填充无法防护的驱逐计时攻击。

Newcache+内存隔离: 这两种方法的结合, 能解决除 Cache 碰撞之外的全部 6 种攻击, 在组合方案的设计上, 需要 Cache 和内存物理结构分开。

随机填充+行锁定+内存隔离: 通过三种防御方法的结合, 可以解决目前所有 Cache 侧信道攻击, 但相应的硬件设计会变得复杂, 需要研究人员进一步探索。

6 相关工作对比

Cache 侧信道攻击的量化评估研究, 可以用来度量侧信息泄露程度, 以及对侧信道攻击和防御效果进行排名。将我们研究方法与之前的工作^[20,27,43-45]进行对比, 总结现有评估方法局限性, 指出本模型的优缺点, 指标和对比结果如表 13 所示。

表 13 相关工作对比
Table 13 Compared with the related work

	SVF	CSV	Zhang	Deng	He	本文
精确性	高	高	中	中	中	中
仿真时间	高	高	中	中	低	低
样本数量	多	多	无	无	无	无
评估范围	低	低	中	高	高	高
硬件漏洞	否	否	否	否	否	是
攻击防御	是	是	否	否	否	是
建模统一						
攻击先验	无	无	无	无	无	有
同等性						

本文所提出的方法以及其他基于模型的评估方法^[20,43,45], 所具有最大的缺点是相对于真实攻击的

代码仿真^[20,44,46-47], 并没有以密钥位作为评判标准, 精度有所降低^[48]。但是搭建真实攻击程序最大的缺陷是仿真时间过长, SVF^[20]和 CSV^[44]中在仿真时需要运行几天时间, Zhang 等人^[27]提出的形式化模型^[27]需要进行迭代, Deng 等人^[45]的计算树模型需要形式化仿真, 这两个模型所需时间中等, 而基于模型的评估方法只需对系统进行建模, 通过计算得出评估结果。除此之外, 基于代码仿真的方法还需要大量的数据样本, 需要采集仿真平台数据, 基于模型评估的方法不需要大量搜集数据。

评估范围是指该方法所具有的通用性, SVF 和 CSV 只能评估具有代码的测试程序, 使用功能具有局限性; Zhang 的模型只能针对于一类侧信道攻击进行建模; He、Deng 及本文方法可以同时针对所有侧信道攻击进行评估。Spectre、Meltdown 等硬件漏洞对系统造成的影响, 也在本文中模型加以对比。

攻击防御建模统一性是指评估方法同时对 Cache 侧信道攻击和防御进行联合评估。SVF 和 CSV 是使用系统泄漏参数评价系统的安全性, 能间接表明程序攻击能力, 防御后的泄露参数能评估防御能力。Zhang、Deng、He 的模型只对防御进行评估, 而我们模型引入了 CVSS 同时对攻击和防御进行评估。

攻击平等性是指不同侧信道攻击在评价过程中赋予相同的攻击能力, 之前的研究工作均未对此进行设定, 因此并不能公平的对侧信道攻击和防御进行评估。我们的方法采用贝叶斯公式对攻击进行建模, 使防御效果更加具有公平性。

7 总结与展望

本文基于侧信道攻击机理, 提出一种图模型, 用来描述侧信道攻击和防御。建立 Cache 侧信道攻击安全性量化标准, 包括攻击和防御概率, 对攻击防御有效性进行评估。通过对 Cache 侧信道现有安全体系量化, 整合防御手段, 提出更有效的防护方法。但本文研究工作仍然存在不足, 评估场景与实际芯片设计具有很大差距, 需要对模型进一步扩展。

之后的工作, 我们将展开以下的研究: (1)在安全性评估基础上, 建立增加功耗、性能、面积等评估指标的更全面的评估模型, 使防御方案的评估更加接近真实场景^[38]; (2)利用文中模型对防御方案的评估结果, 提出更加安全有效的安全 Cache 架构^[42-43], 完成硬件实现, 同时对架构进行探索和优化。

致谢 在此向对本文工作提供帮助的课题组各位老师同学以及提出建议的评审专家表示感谢。

参考文献

- [1] John L Hennessy, David A Patterson. Computer Architecture: A Quantitative Approach[M]. Morgan Kaufmann Publishers Inc. 2003.
- [2] Lyu Y, Mishra P. A survey of side-channel attacks on caches and countermeasures[J]. *Journal of Hardware and Systems Security*, 2018, 2(1): 33-50.
- [3] Ge Q, Yarom Y, Cock D, et al. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware[J]. *Journal of Cryptographic Engineering*, 2018, 8(1): 1-27.
- [4] Tromer E, Osvik D A, Shamir A. Efficient cache attacks on AES, and countermeasures[J]. *Journal of Cryptology*, 2010, 23(1): 37-71.
- [5] Tan Y, Wei J, Guo W. The Micro-architectural Support Countermeasures against the Branch Prediction Analysis Attack[C]. *IEEE International Conference on Trust. IEEE Computer Society*, 2014.
- [6] David Gullasch, Endre Bangerter, and Stephan Krenn. 2011. Cache Games—Bringing Access-Based Cache Attacks on AES to Practice[C]. *IEEE Symposium on Security and Privacy*, 2011: 490-505.
- [7] Chen C, Tao W, Guo S. Research on Trace Driven Data Cache Timing Attack Against RSA[J]. *Chinese Journal of Computers*, 2014, 37(5): 1039-1051.
(陈财森, 王韬, 郭世泽, 等. 针对 RSA 算法的踪迹驱动数据 Cache 计时攻击研究[J]. *计算机学报*, 2014, 37(5): 1039-1051.)
- [8] Fangfei Liu, Yuval Yarom, Qian Ge, et al. Last level cache side-channel attacks are practical[C]. *In IEEE Symposium on Security and privacy*. 605-622
- [9] Gulmezoglu B, Inci M S, Irazoqui G, et al. Cross-VM Cache Attacks on AES[J]. *IEEE Transactions on Multi-Scale Computing Systems*, 2016, 2(3): 211-222.
- [10] Kocher P, Genkin D, Gruss D, et al. Spectre Attacks: Exploiting Speculative Execution[J]. *Communications of the ACM*, 2018, 63(7).
- [11] M. Lipp, M. Schwarz, D. Gruss, et al. Meltdown: Reading kernel memory from user space[C]. *USENIX Security Symposium*, 2018, 973-990.
- [12] Refazul I R, Islam E, Khan M M. An Efficient Indexing Technique for AES Lookup Table to Prevent Side-Channel Cache Timing Attack[J]. *International Journal of Computer Network and Information Security*, 2018, 9(9): 25.
- [13] He Z, Lee R B. How secure is your cache against side-channel attacks?[C]. *IEEE/ACM International Symposium*, 2017.
- [14] Page D. Partitioned Cache Architecture as a side-Channel Defence Mechanism[J]. *IACR Cryptology ePrint Archive*, 2005.
- [15] Wang Z, Lee R B. New cache designs for thwarting software cache-based side channel attacks[J]. *ACM SIGARCH Computer Architecture News*, 2007, 35(2): 494-505.
- [16] Wang Z, Lee R B. A novel cache architecture with enhanced performance and security[C]. *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*. 2008: 83-93.
- [17] Domnitsier L, Jaleel A, Loew J, et al. Non-monopolizable caches:

- Low-complexity mitigation of cache side channel attacks[J]. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2012, 8(4): 35.
- [18] Fangfei Liu, Qian Ge, Yuval Yarom, et al. Catalyst: Defeating last-level cache side channel attacks in cloud computing[C]. In *2016 IEEE International Symposium on High Performance Computer Architecture*, 406-418.
- [19] Fangfei Liu and Ruby B Lee. Random fill cache architecture[C]. *IEEE/ACM International Symposium on Micro architecture*. 2014: 203-215.
- [20] John Demme, Robert Martin, Adam Waksman, et al. Side-channel vulnerability factor: a metric for measuring information leakage[C]. *ACM SIGARCH Computer Architecture News*. 2012: 106-117.
- [21] Fangfei Liu, Hao Wu, Kenneth Mai, et al. Newcache: Secure Cache Architecture Thwarting Cache Side-Channel Attacks[C]. *IEEE/ACM International Symposium on Micro architecture*. 2016: 8-16.
- [22] Kong J, Aciicmez O, Seifert J P, et al. Deconstructing new cache designs for thwarting software cache-based side channel attacks[C]. *Proceedings of the 2nd ACM workshop on Computer security architectures*. ACM, 2008: 25-34.
- [23] Wang L, Zhu Z, Wang Z, et al. Analyzing The Security of The Cache Side Channel Defences With Attack Graphs[C]. *Asia and South Pacific Design Automation Conference*, 2020.
- [24] Wang L, Zhu Z, Wang Z, et al. Colored Petri Net Based Cache Side Channel Vulnerability Evaluation [J]. *IEEE Access*, 2019.
- [25] Fangfei Liu and Ruby B Lee. Security testing of a secure cache design[C]. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM, 3.
- [26] Szefer, Jakub. Principles of Secure Processor Architecture Design[J]. *Synthesis Lectures on Computer Architecture*. 2018: 1-173.
- [27] Zhang T, Lee R B. Secure cache modeling for measuring side-channel leakage[D]. Princeton University, 2014.
- [28] Daemen J, Rijmen V. AES proposal: Rijndael[J]. 1999.
- [29] Joseph Bonneau and Ilya Mironov. Cache-collision timing attacks against AES[C]. In *International Workshop on Cryptographic Hardware and Embedded Systems*, 2006: 201-215.
- [30] Zhao Xinjie, Wang Tao, Guo Shize, and Liu Huiying. Cache Attacks on Block Ciphers[J]. *Journal of Computer Research and Development*, 2012, 49(3): 453-468.
(赵新杰, 王韬, 郭世泽, 等. 分组密码 Cache 攻击技术研究[J]. *计算机研究与发展*, 2012, 49(3): 453-468.)
- [31] Daniel J Bernstein. 2005. Cache-timing attacks on AES. Online, November 2004. <http://cr.yp.to/cachetiming.papers.html#>.
- [32] Colin Percival. 2005. Cache missing for fun and profit[C]. *BSDCan 2005*, 2005.
- [33] Osvik D A, Shamir A, Tromer E, et al. Cache attacks and counter-measures: the case of AES[C]. *The cryptographers track at the rsa conference*, 2006: 1-20.
- [34] Gruss D, Maurice C, Wagner K, et al. Flush+ Flush: a fast and stealthy cache attack[C]. *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2016: 279-299.
- [35] Deng S, Xiong W, Szefer J. Secure tlbs[C]. *Proceedings of the 46th International Symposium on Computer Architecture*, 2019: 346-259.
- [36] Li P, Zhao L, Hou R, et al. Conditional Speculation: An Effective Approach to Safeguard Out-of-Order Execution Against Spectre Attacks[C]. *2019 IEEE International Symposium on High Performance Computer Architecture*. 2019: 264-276.
- [37] Trippel C, Lustig D, Martonosi M. Meltdown-Prime and SpectrePrime: Automatically-Synthesized Attacks Exploiting Invalidation-Based Coherence Protocols[J]. 2018.
- [38] Zhang Suiyu, Han Jun, Lu Shiting, and Zeng Xiaoyang. Cache Based AES Attack Implementation and Its Theoretical Analysis[J]. *Journal of Computer Research and Development*, 2011, 48(6): 955-963
(张随欲, 韩军, 卢仕昕等. 针对 SoC 系统的 Cache 攻击方法及建模分析[J]. *计算机研究与发展*, 2011, 48(6): 955-963.)
- [39] <https://www.first.org/cvss/>.
- [40] Song Z, Chensi W U, Weiqiang X, et al. Research on network security measurement based on attack graph[J]. *Journal of Cyber Security*, 2019, 4(1): 52-66.
(赵松, 吴晨思, 谢卫强, 等. 基于攻击图的网络安全度量研究[J]. *信息安全学报*, 2019, 4(1): 52-66.)
- [41] Page D. Theoretical use of cache memory as a cryptanalytic side-channel[J]. *IACR Cryptology ePrint Archive*, 2002, 2002(169).
- [42] Lee S, Shih M W, Gera P, et al. Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing[J]. 2016.
- [43] Meng D, Hou R, Shi G, et al. Security-first architecture: deploying physically isolated active security processors for safeguarding the future of computing[J]. *Cybersecurity*, 2018, 1(1):2.
- [44] Tianwei Zhang, Fangfei Liu, Si Chen, and Ruby B Lee. 2013. Side channel vulnerability metrics: the promise and the pitfalls[C]. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. 2013.
- [45] Deng S, Xiong W, Szefer J. Cache timing side-channel vulnerability checking with computation tree logic[C]. *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, 2018: 2.
- [46] Deng S, Xiong W, Szefer J. Analysis of Secure Caches and Timing-Based Side-Channel Attacks[J]. *IACR Cryptology ePrint Archive*, 2019: 167.
- [47] Deng S, Xiong W, Szefer J. A Benchmark Suite for Evaluating Caches' Vulnerability to Timing Attacks[EB/OL]. 2019: ArXiv Preprint ArXiv: 1911.08619.
- [48] Doychev, Goran, et al. Cacheaudit: A tool for the static analysis of cache side channels[J]. *ACM Transactions on Information and System Security (TISSEC)*, 18.1 (2015): 4.



王占鹏 于 2012 年在沈阳理工大学控制工程专业获得硕士学位。现在中国科学院信息工程研究所计算机体系结构专业攻读博士学位。研究领域为计算机系统安全。研究兴趣包括: 系统防御、侧信道分析、安全量化。Email: wangzhanpeng@iie.ac.cn



朱子元 于 2010 年在同济大学控制理论与控制工程专业获得博士学位。现任中国科学院信息工程研究所研究员。研究领域为网络空间安全、计算机系统结构。研究兴趣包括: 安全芯片技术、处理器安全技术、系统安全理论与技术。Email: zhuziyuan@iie.ac.cn



王立敏 于 2017 年在杭州电子科技大学计算机专业获得学士学位。现在中国科学院信息工程研究所计算机体系结构专业攻读硕士学位。研究领域为系统安全。研究兴趣包括: 侧信道攻击、形式化方法。Email: wanglimin@iie.ac.cn