arXiv:2304.00955v1 [cs.CR] 3 Apr 2023

关于论文《随机化缓存真的随机吗？》的简短说明[4]

Anirban Chakraborty[1],Sarani Bhattacharya[2],Sayandeep Saha[3],和 Debdeep Mukhopadhyay[1]

[1] 印度理工学院克勒格布尔分校 anirban.chakraborty@iitkgp.ac.in,
debdeep@cse.iitkgp.ac.in[2] Imec，比利时 sarani.bhattacharya@imec.be[3]
南洋理工大学，新加坡 sayandeep.iitkgp@gmail.com

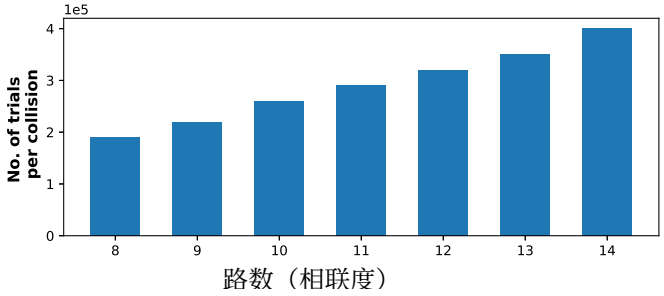
摘要。 本文分析了发表于 2023 年 HPCA 会议的论文 《随机化分区缓存真的随机吗？随机化分区缓存的形式化分析》中提出的结果与论断。此外，我们还分析了 MIRAGE [6] 中提出的“Bucket and Ball”解析模型在安全性评估中的适用性。我们指出了原始桶球模型中的谬误，并讨论其影响。最后，我们展示了仅使用总缓存容量的 10% 即可对 MIRAGE 发起缓存占用攻击，并扩展该框架以实现隐蔽信道和基于模板的指纹攻击。

关键词： 随机缓存 · MIRAGE · 缓存占用攻击。

1 引言

缓存攻击是针对微体系结构最有力的侧信道攻击之一。攻击者通过观察数据在缓存中的存储和访问模式，从系统中提取敏感信息。文献中有各种各样的攻击，可泄露加密密钥、云端用户数据等关键信息。尽管已知缓存攻击会针对缓存层次结构的所有层级，但针对末级缓存（LLC）的攻击更值得关注，因为末级缓存通常由多个核心共享，因此多个进程会同时共享末级缓存。缓存攻击中一种广为人知的技术，是利用攻击者的数据驱逐特定缓存组中的数据，以推断有关目标受害进程的安全关键信息。自然而然，多年来人们提出了多种对策，包括缓存分区和随机化地址到缓存组映射。缓存随机化和分区通过混淆地址到缓存组的映射，并在逻辑上将缓存进一步划分为多个分区，从而减缓驱逐集的构建过程。然而，文献中已有攻击表明，可以设想更新、更先进的算法来发现安全随机化缓存中的驱逐集。还需注意，尽管存在其他同样恶意且威力强大的缓存攻击变体，但缓存保护方案通常更侧重于阻止基于驱逐的攻击。

在 [4], 发表于 HPCA’23 的论文中，我们提供了一个系统框架，用于分析文献中提出的不同类型的安全缓存设计。论文首先根据在这些缓存中分配地址时非确定性和随机性的程度，将不同类型的缓存设计划分为四大类。这种分类有助于根据缓存设计的随机化特性和设计原理，将多种缓存设计归类。随后，我们建立了一个数学框架，从碰撞概率、自碰撞概率以及成功实施基于驱逐的攻击所需的驱逐集大小等方面，形式化地分析随机化分区缓存设计的安全影响。利用这一数学框架，我们展示了攻击者驱逐集各元素之间的自碰撞概率，这构成了整体过程中的噪声 。具体而言，我们展示了对手如何利用预先计算的自碰撞概率值和驱逐集中的元素数量，为所有已知类别的缓存设计构造精心筛选的驱逐集。



Fi图 1：MIRAGE 中冲突的频率，相联度从 8 变化至 $8+6$ [4]

2 关于 MIRAGE 的相关主张 [6]

除了上述贡献外，论文 [4]还首次展示了在一种名为 MIRAGE [6] 的新型随机化设计中出现的组相联驱逐（SAE）现象。论文还利用 SAE 事件作为驱逐信号，提出了一种生成驱逐集的算法。在论文中，我们首先指出，MIRAGE [6] 原论文提出的桶球模型并未考虑所有可能导致桶溢出 现象的事件。MIRAGE 原论文假设了一种桶球场景：每个缓存组（连同 MIRAGE 中配置的额外标签）代表一个桶，而球则代表要装入缓存的传入地址行数量。

为了估算桶溢出的概率（即一个桶中包含超过 N 个球），MIRAGE 的作者将桶状态建模为生灭链，以计算包含 N 个球的桶转变为包含 $(N+1)$ 个球的桶的概率。我们分别将这些桶状态称为 N^{th} 状态和 $(N+1)^{th}$ 状态。假设从 N^{th} 状态到 $(N+1)^{th}$ 状态以及反向转变的速率相等，作者得出了以下桶溢出方程

$$Pr(n=N+1)=\frac{B}{\mathcal{B}\times(N+1)}\times Pr(n=N)^2 \tag{1}$$

其中，$\mathcal{B}$ 是桶的数量，B 是投出的球的总数。

在 [4], 中，我们解释说，虽然 N^{th} 状态下某个特定桶转变为（$N+1$）th 状态的事件发生概率极低，但这对攻击者而言是一个限制性假设。作为攻击者，它可以自由地填满任意一个桶，从而使缓存偏向更脆弱的状态。因此，利用优惠券收集问题，恰好包含$N+1$ 个球的任意一个桶出现的概率可表示为

$$\begin{aligned} Pr[n=N+1] &= \binom{B}{N+1}\left(\frac{1}{\mathcal{B}}\right)^{N+1}\left(1-\frac{1}{\mathcal{B}}\right)^{B-(N+1)}\times\mathcal{B} \\ &\approx \frac{\lambda^{N+1}e^{-\lambda}}{(N+1)!}\times\mathcal{B} \qquad where \quad \lambda=\frac{\mathcal{B}}{B} \end{aligned} \tag{2}$$

在上述分析中，攻击者从空缓存开始，随机尝试在 $\mathcal{B}$ 个桶中的任意一个造成桶溢出。不过，我们也在 [4] 中指出，即使不将缓存的初始状态视为完全空，也仍有可能耗尽某个同级组中的所有路（即发生桶溢出）。使用公式 2，我们在自研的 MIRAGE 末级缓存模拟器上展示了实测结果：当缓存参数设为 MIRAGE 推荐的版本时，约经过 10^5 – 10^6 次投球即可观察到桶溢出（图 1、2）。

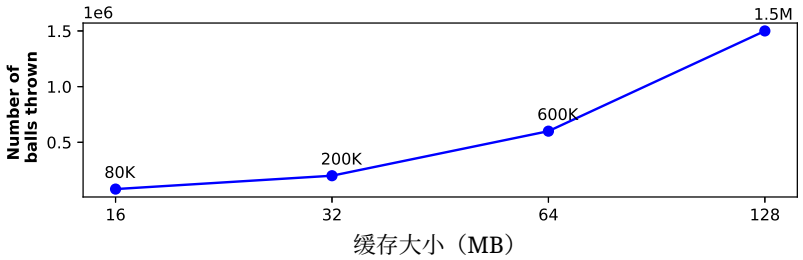


图2：不同缓存大小下，至少发生一次桶溢出所需投掷的球数 [4]

3 该主张中的技术问题

尽管第 2 节讨论的数学模型适用于桶与球分析，但我们发现，由于对 MIRAGE 的表示有误，出现了一个关键问题。根据 MIRAGE 的原始论文， [6]，作者声称，桶与球模型恰当地表示了 MIRAGE 的工作原理。他们关于 MIRAGE 安全保障的主张，建立在桶与球模型有效这一前提之上。本文指出了原始桶与球模型表示中的一个重大缺陷，并解释了为什么这样的模型无法恰当地证明实际缓存设计中的抽象是合理的。在继续讨论之前，为求简洁，我们先简要介绍 MIRAGE 的设计背景。

3.1 Mirage 背景

MIRAGE [6] 是一种随机化缓存设计，于 USENIX Security 2021 上提出。MIRAGE 的设计理念不同于其他类别的随机缓存和分区缓存架构。传统缓存设计以及大多数知名的随机化缓存设计，都会将任意特定地址映射到少量缓存行，因为随机化地址仍按照组相联模式进行映射。即使引入分区，地址可映射到的缓存行总数仍然固定。MIRAGE 试图引入一种“伪”全相联随机化缓存设计，从而使构造驱逐集变得困难。MIRAGE 将标签存储与数据存储分离（如图3所示），并在标签存储中采用组相联查找，同时保持数据存储的全相联特性。

核心思想是在缓存未命中时强制执行全局替换，以避免组冲突。为支持全局驱逐，MIRAGE 在标签存储区的每个组中配置了额外的无效标签。因此，当一个新地址到达并映射到缓存时，可以使用标签存储区中的一个无效标签来映射该地址，同时从数据存储区中随机选取一个条目，从而实现全局驱逐。除了配置全局驱逐并通过指针间接寻址解耦标签存储区与数据存储区之外，MIRAGE 还结合了倾斜缓存和输入地址随机化的概念。标签存储区被划分为两个倾斜组，同时配有一个全相联数据存储区。

3.2 通过桶球模型进行安全性分析

为了对其方案进行安全性分析，MIRAGE 论文的作者 [6] 将标签存储中的每个组抽象为一个桶，并将每次地址安装抽象为一次投球。然而，他们提出的解析模型（如公式 1 所示）未能正确反映该设计的工作原理。作者将桶球系统的行为建模为一条生灭链 [6]。因此，MIRAGE 中的分析本质上描述了这样一种情形：一个桶在其 N^{th}

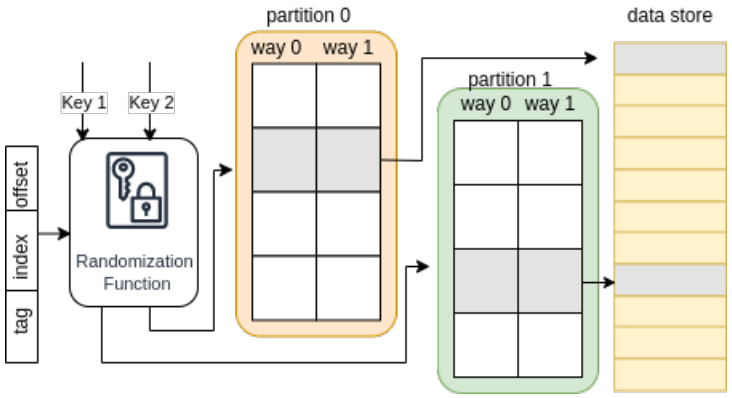


图 3：MIRAGE [6]。组相联标签存储区和全相联数据存储区

状态转移到 $(N+1)^{th}$ 状态。在 [4], 中，我们表明，这种概率并不像 MIRAGE 所声称的那样低，因为在这种情况下，攻击者可以设法找到任何一个从 N^{th} 转移到 $(N+1)^{th}$ 状态的桶。然而，*MIRAGE* 的这种建模并不恰当。请注意，MIRAGE 会在每个组中额外配置无效标签。此外，它还采用负载均衡的概念，任何传入地址都可以映射到两个同级组。因此，考虑到每个偏斜中额外的 m 路，攻击者必须填满两个组中的所有 m 路，才能造成冲突。

我们以 MIRAGE 的一个参考设计为例来理解这一场景。考虑一个 16MB 的 MIRAGE 缓存，其标签存储区有 $8+6$ 路 [1]。因此，标签存储区采用组相联索引方式，每个偏斜中有 $16K$ 个组（总计 $16K$ 个组）。同样，假设每个组中有 8 个有效路，则数据存储区中的条目数为 $16K \times 8$。遵循原始 MIRAGE 论文 [6], 中桶球模型的假设，我们用一些有效标签和一些无效标签初始化缓存。给定每个组中有 x 个有效标签和 y 个无效标签的初始状态，攻击者访问任意地址，将其映射到兄弟集合中。对于 64 位寻址系统，输入地址首先使用两个固定密钥加密，生成两个相互独立的 64 位密文 $\mathcal{C}_1$ 和 $\mathcal{C}_2$。根据 MIRAGE 的负载均衡原理，$\mathcal{C}_1$ 和 $\mathcal{C}_2$ 的索引位都用于索引到两个同级组。由于共有 $16K$ 个组，密文中的 14 位用于映射到每个偏斜中对应的组。换句话说，一个输入地址经过加密后会生成一个唯一的元组（i_1，i_2），其中 i_1 表示偏斜 1 中的组编号，i_2 表示偏斜 2 中的组编号。因此，要产生冲突，对手必须选择一个地址，使其对应密文中的 14 位（i_1，i_2），共 28 位，与这些同级组中已有元素相匹配。现在我们将说明，找到这种“冲突”的概率并非微不足道（这与 MIRAGE [6] 中的说法相反）。

3.3 使用生日悖论的冲突

生日悖论 [9] 是一种统计现象，表明当样本数量足够大时，任意两个事件发生冲突的概率会增至接近 50%。对于 MIRAGE，要在选定的标签组中找到冲突，就需要匹配来自随机分布的 28 位（假设加密函数生成的密文服从均匀分布）。然而，根据生日悖论，对手可以找到一对索引（j_1，j_2），使它们在$2^{\frac{28}{2}} = 2^{14}$ 次访问中的任意一个组中发生冲突。将这一关系应用于桶球问题，对手需要2^{14} 次访问，才能将处于 N^{th} 状态的任意一个桶转变为（$N+1$）th 状态。然而，MIRAGE

[1] MIRAGE 的默认版本要求每个组中有 8 个基本标签和 6 个额外标签。

会在每个标签存储区中额外配置无效标签。对于我们的示例缓存，每个组都会有 6 个额外的无效标签，因此一对兄弟集合中无效标签的总数为 12。因此，要强制发生组相联逐出，对手需要在其中一个同级组中造成 12 路冲突。桶球模型的解析模型描述的是桶从 N^{th} 状态转变为（$N+1$）th状态的事件，而正确的模型应考虑任意一个处于 N^{th} 状态的桶转变为（$N+m$）th 状态的事件，其中 m 是每个标签组中的额外标签数。我们得出结论， [6] 和 [4] 中给出的解析结果都未能恰当地表示 MIRAGE 的缓存行分配操作。两篇论文都试图将“桶溢出”事件建模为从 N^{th} 状态转变为（$N+1$）th 状态，而正确的状态转变应是从 N^{th} 转变为（$N+m$）th。这种状态转变需要 m路冲突（而非单次冲突），这在统计上极不可能发生，因此能够为其提供充分保护，抵御基于冲突的攻击。

4 关于 [4]中 MIRAGE 结果的说明

在上一节中，我们讨论了论文 [6] 和 [4]中 MIRAGE 的桶球模型存在的谬误。然而， [4], 中关于 MIRAGE 的结果，具体来说是图 7 和图 10（来自论文 [4]），是由专门为 MIRAGE 编写的模拟器生成的，并未采用桶球模型抽象。这两幅图都表明，在 Mirage 中可以构造组相联驱逐，最终可用于创建驱逐集。遗憾的是，我们发现这些结果有误，原因是随机化分组密码的实现不正确。该专用模拟器使用 PRESENT 分组密码 [2] ，通过两个不同的密钥对输入地址进行随机化。PRESENT 是一种轻量级分组密码，已被证明能够抵御所有已知攻击，具有较强的密码分析安全性。MIRAGE 的设计假设所选密码必须确保地址均匀映射到缓存组。该专用模拟器使用了一个以 Python2 [5] 编写的 GitHub PRESENT 实现。由于模拟器其余部分是用 Python3 编写的，因此也必须移植密码实现。移植后的密码实现以地址的二进制表示作为输入，并将其转换为整数，以便进行后续密码运算。问题就出在这一步转换：二进制表示在无意中被当作十六进制格式转换成了整数。结果，64 位输入被转换成 128 位，而其余密码运算只使用转换后数值的最后 64 位。这导致两个不同的输入地址 a_1 和 a_2 的最后 16 位相同，进而产生相同的密文，并映射到相同的兄弟集合。修正后，该密码实现使组索引呈均匀分布；对于每个标签组含*16 MB MIRAGE* 且无冲突$8+6$路 的情况，未观察到冲突。这一观察结果也符合我们在第 3.3 节中的论点：此类事件需要发生 12 路冲突，而其概率极低。

不过，我们也注意到，论文 [4] 提出了一个框架，可为各种安全缓存设计生成高效的驱逐集。尽管由于密文分布不均匀，关于 MIRAGE 的结果并不正确，但论文中的其他结果不受这一问题影响。在 [4], 中，我们针对随机化分区缓存设计构建了一个数学框架，并推导出自驱逐概率以及驱逐集中所需最少元素数的公式。我们利用这些参数开发了一种通用算法，可适用于除 MIRAGE 之外的所有缓存设计变体。MIRAGE 属于伪全相联缓存，其行为不同于文献中其他现有的缓存设计。

5 MIRAGE 真的安全吗?

要在 MIRAGE 中进行组相联驱逐（这是创建驱逐集的关键），对手需要在任意标签集合中生成 m 路冲突。因此，基于驱逐的 Mirage 攻击

由于提供了额外的无效标签，这类攻击看似并不切实际。然而，基于驱逐的攻击只是文献中诸多缓存攻击变体之一。要判断一种设计是否足够安全且不会泄漏信息，必须结合所有已知且切实可行的攻击来分析。

5.1 MIRAGE 中的全局驱逐

MIRAGE 的一个主要特点是，它将数据存储区实现为全相联缓存。不过，两者之间存在显著差异。在全相联缓存中，任何传入地址都可以放入任意槽位，前提是该槽位包含一个无效条目或为空闲状态。因此，对于全相联缓存，CPU 会搜索整个缓存，查找包含无效条目的槽位，以便安装新传入的地址。如果找不到这样的无效条目，就会根据当前采用的替换策略驱逐一个有效条目。然而，全相联缓存并不适用于末级缓存，因为穷举搜索会带来显著的性能开销。为缓解这一问题，MIRAGE 对所有条目采用随机驱逐策略，这称为全局驱逐。对于任何传入地址，都会从数据存储区随机驱逐一个条目，并使标签存储区中与之对应的条目失效。需要注意的是，在 MIRAGE 中，全局驱逐是一个确定性事件。换言之，每次安装新地址都会发生全局驱逐，无论数据存储区中是否包含任何无效条目。这与全相联缓存及其他随机化设计形成鲜明对比：在这些设计中，只有满足特定条件时才会发生驱逐。本节将讨论这一看似无害的事件如何泄露受害进程的信息，实际上还会引入新的攻击途径。

5.2 缓存占用攻击

缓存攻击可大致分为三类：访问攻击、定时攻击和跟踪攻击。基于驱逐的攻击（如 Prime+Probe、Evict+Time 等）会尝试填满缓存中的特定组，以便与受害进程产生冲突。这类攻击的一个重要标准是，攻击者必须填满目标缓存组中的所有路，才能泄露受害者的信息。然而，近期出现了另一类缓存攻击，这类攻击不依赖于缓存组信息的粒度。这些攻击通常称为缓存占用攻击，关注的是受害者的缓存足迹（即缓存占用），而不是依赖特定缓存组中的活动。这类攻击功能强大，已有研究利用它们执行网站指纹识别 [8,7]。由于 MIRAGE 的策略规定每安装一个新地址，就必须从数据存储区中强制驱逐数据，因此它尤其容易受到此类缓存占用攻击的影响。下一节将讨论缓存占用攻击如何在 MIRAGE 中形成隐蔽信道并实现基于模板的指纹识别。

6 MIRAGE 中的隐蔽信道

隐蔽信道是一种非法通信信道，旨在躲避或隐藏于系统的检测机制之外。隐蔽信道是严重的安全隐患，因为它们可用于跨越进程边界传输敏感信息。由于末级缓存由所有处理器共享，因此它为运行在不同物理核心上的进程之间建立隐蔽信道提供了合适的载体。本节将展示一种在 MIRAGE 上实现的低错误率隐蔽信道，该信道只需占用整个数据存储区不到 10% 的空间即可建立。

6.1 信道设置

为展示隐蔽信道，我们使用内部开发的 MIRAGE 末级缓存模拟器，其容量为 16MB，每个标签组有 $8+6$ 路，并采用 2 个倾斜组。每个传入地址都使用 PRESENT

分组密码进行随机化，所用密钥固定且彼此不同。值得注意的是，PRESENT 分组密码的实现通过了原论文 [2] 中给出的测试向量。因此，数据缓存中的位置数量为 131，072，因为共有 16K 组和 8 个有效路。

隐蔽信道通常是单向信道，两个参与方——发送方和接收方——通过利用共享资源和预先约定的模式来串通传输信息。为了使隐蔽信道具备实用性和隐蔽性，从而避免引起怀疑，双方不应拥有任何显式通信信道，例如共享内存。缓存存储器此前已被文献用于构建高带宽隐蔽信道。然而，要有意义地传输信息，发送方和接收方需要占用缓存的大部分空间。在本节中，我们展示了一种针对 MIRAGE 使用缓存占用攻击实现的缓存隐蔽信道，其占用的缓存存储器不到整个缓存存储器的 10%。此外，由于每个传入地址都会被随机映射（这是由分组密码实现的随机化所导致的），双方的内存占用实际上会在缓存中形成随机访问模式，因此不会引起对隐蔽操作的怀疑。为详细说明该攻击的工作原理，下面我们将讨论双方执行的操作。

接收方： 发送方分配一段内存空间，并随机访问 10,000 个地址。为了确保每个地址占据整条缓存行，地址按顺序访问，且相邻地址之间的间隔为 1,000。例如，如果首次访问的地址是p，那么接下来要访问的地址将是 $1000 + p$。这样可以确保没有任何地址发生缓存命中，因此每个地址都会占据一整条缓存行。需要注意的是，由于每有一个新地址到来都会发生全局驱逐（无论缓存的占用情况如何），这 10,000 个地址中的多个地址会被随机驱逐。在实验中，我们发现，整个操作结束时，10,000 个地址中约有 400 个被驱逐。需要注意的是，全部 10,000 个地址都映射到不同的缓存行，而且初始缓存状态完全为空。然而，MIRAGE 的全局替换策略会从数据缓存中随机选择并驱逐约 400 个已载入的地址。因此，发送方完成对其地址的访问后，数据存储区中约有 9600 个条目填入了有效项，占整个数据存储区容量的 7.32%。

发送方： 我们假设接收方在一个独立的物理核心上执行，完全不了解发送方访问的地址。此外，MIRAGE 的随机化方案本就会打乱任意两个地址之间的关联。发送方插入自己的地址后，接收方开始操作。与接收方类似，发送方会分配一段内存空间，并根据它想要传输的比特值（0 或 1），执行 1000 次或 4000 次内存访问。此外，所访问的地址间隔为 1000，以确保没有两次访问映射到同一缓存行。发送方每插入一个地址，都会根据全局驱逐策略驱逐一个数据存储区条目。有趣的是，我们观察到，由于数据存储区会随机选择驱逐候选项，发送方先前已安装的部分条目也会被驱逐。具体而言，发送方执行 1000 次访问后，平均有 90 个地址在操作结束时被驱逐；执行 4000 次访问时，平均有 350 个发送方地址被驱逐。

接收方： 发送方完成操作后，接收方会重新访问其安装在缓存中的全部 10,000 个地址。值得一提的是，此时接收方并不知道自己的地址有多少被缓存驱逐，也不知道发送方的地址目前有多少安装在缓存中。接收方重新访问这些地址，并分别测量每个地址的访问时间。仍在缓存中的地址会命中缓存，而因被驱逐而不在缓存中的地址则会

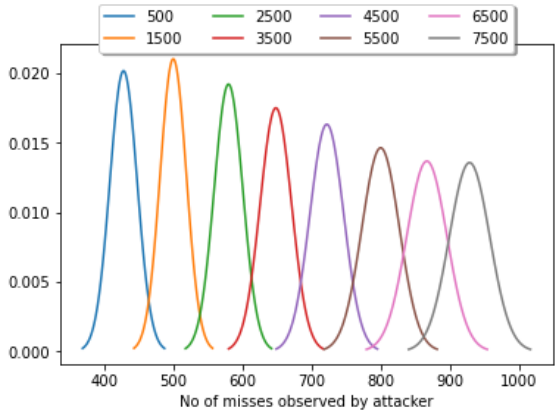


图 4：受害者访问模板，访问总数范围为 500 至 7500 次，间隔为 1000 次

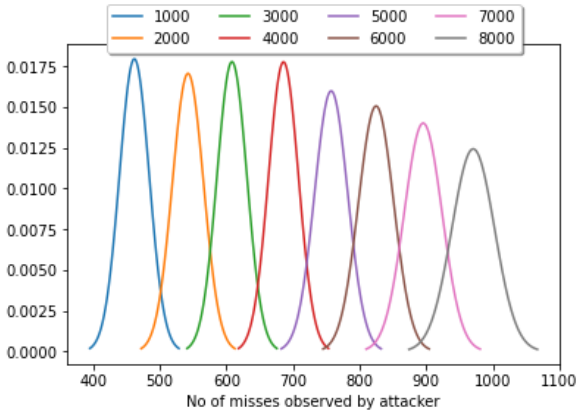


图 5：受害者访问模板，访问总数范围为 1000 至 8000 次，间隔为 1000 次

发生缓存未命中，这是随机全局驱逐造成的。在我们的实验中，我们观察到，当发送方访问 1000 个地址时，接收方发现自己预先安装的地址中约有 450 个发生缓存未命中。同样地，当发送方访问 4000 个地址时，接收方观察到约有 750 次缓存未命中。因此，通过测量缓存未命中的总数，接收方能够确定发送方传输的秘密比特。

7 基于模板的指纹识别

MIRAGE 中用于全局驱逐的数据缓存条目采用随机选择，这使得攻击者只需占用不到 10% 的缓存，就能在不同核心之间建立隐蔽性极高的隐蔽信道。本节中，我们利用缓存占用漏洞扩展攻击，实施威力更大的侧信道攻击。具体而言，我们展示了如何根据系统中未知进程的访问次数，对其进行潜在的基于模板的指纹识别。 [8,7] 已展示利用缓存占用攻击进行网站指纹识别的方法。缓存占用攻击被认为比基于争用的攻击更强大，因为它的要求较低，也不需要像其他缓存攻击变体那样获取细粒度信息。然而，在传统缓存以及大多数随机化方案中，缓存占用攻击都要求攻击者占用缓存的很大一部分，才能迫使受害进程驱逐大量攻击者的地址。MIRAGE 的全局驱逐带来了新的漏洞，可利用它根据缓存未命中的总数创建模板，从而对进程进行指纹识别。沿用隐蔽信道实验设置，攻击者分配一段内存空间，并访问 10，000 个地址，将它们分别放入不同的缓存行。攻击者的目标是针对假设的／虚拟受害者进程执行不同数量的访问，创建相应模板。虚拟受害者进程执行的缓存访问次数从 1000 到 8000 不等，间隔为 500。因此，我们创建了 16 个总访问次数模板，每个模板之间相差 500 次访问。图 4 和图 5 展示了不同总访问次数对应的模板，其中 x 轴表示受害进程完成访问后，攻击者观察到的未命中总数。折线图实际上表示攻击者在受害进程执行不同访问次数时观察到的未命中总数分布。从图中可以明显看出，如果两个（或更多）进程的缓存足迹至少相差 1000 次不同的内存访问，就可以根据这些模板将它们

清晰地区分开来。这使得只需测量对手自身内存地址的缓存未命中次数，就能对进程进行基于模板的指纹识别。

7.1 密码选择的影响

MIRAGE 假设组索引推导函数产生均匀分布的输出。图 4 和图 5 所示的模板是在一个模拟器上创建的，该模拟器采用具有两个固定且不同密钥的 PRESENT 作为随机化函数。为确定密钥选择及整体密码不会对缓存占用攻击产生影响，我们使用另一种分组密码进行了模板构建实验。由于 Mirage 的作者在其分析中使用了 PRINCE [3] 分组密码，我们使用 PRINCE 密码进行了相同的模板创建实验，并在每次模板构建时随机选择密钥。这表明模板构建不依赖于密码或密钥的选择。这一观察结果相当直观，因为全局驱逐并不基于密码或密钥，而是从数据存储区中随机选择受害者条目。只要随机化函数产生均匀分布的输出，使用任意选定密钥创建的模板便可用于对地址经完全不同密钥加密的进程进行指纹识别。这一结果还表明，攻击无法通过重新生成密钥来阻止。模板一旦创建，攻击者便可使用它，而不受重新生成密钥周期的影响。

7.2 其他随机化变体上的缓存占用

此时一个重要的问题是，与其他缓存变体（包括经典的组相联缓存）相比，MIRAGE 是否更容易受到缓存占用攻击的影响。我们的实验对此给出了肯定的回答。在这些缓存设计中，攻击者需要占用缓存的很大一部分才能发动缓存占用攻击。要观察到缓存未命中，攻击者需要填满多个组；而在 MIRAGE 中，全局驱逐有助于驱逐攻击者的地址。

8 未来方向与结论

本文展示的攻击是在基于 Python3 构建的内部末级缓存模拟器上进行的概念验证，该模拟器遵循 MIRAGE 的设计原则。未来的工作将是在标准体系结构模拟器（如 gem5 [1]）上验证这些观察结果，并对不同进程和工作负载进行指纹识别。

缓存占用攻击是一种强大且可能造成严重危害的攻击技术，因为它没有严格的要求，并且能够泄露有关受害者操作的信息。大多数安全缓存设计都侧重于增强对基于争用的攻击的抵御能力，却忽视了其他攻击。针对 MIRAGE 的攻击鲜明地提醒我们：针对某一类攻击设计对策，可能会使设计容易受到其他攻击的影响。

致谢

我们感谢 Mirage [6] 的作者指出 PRESENT 密码输出并非均匀分布，这促使我们发现了实现中的问题，以及对 Bucket andBall 模型的不正确表述。

参考文献

1. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., 等：gem5 模拟器。《ACM SIGARCH 计算机体系结构新闻》**39**(2), 1–7（2011）2. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe：Present：一种超轻量级分组密码。载于：国际密码硬件与嵌入式系统研讨会。第 450–466 页。施普林格（2007）3. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., 等：PRINCE——一种面向普适计算应用的低延迟分组密码。载于：密码学进展——ASIACRYPT 2012：第 18 届密码学与信息安全理论及应用国际会议，2012 年 12 月 2 日至 6 日，中国北京。论文集 18。第 208–225 页。施普林格（2012）4. Chakraborty, A., Bhattacharya, S., Saha, S., Mukhopadhyay, D.：《随机缓存真的随机吗？随机分区缓存的形式化分析》。载于：第 29 届高性能计算机体系结构国际研讨会（HPCA）论文集。第 – 页。IEEE（2023）5. GitHub：Present Python 实现。https://github.com/xSAVIKx/PRESENT-cipher/blob/master/ present/pyPresent.py（2015）6. Saileshwar, G., Qureshi, M： {MIRAGE}：通过实用的全相联设计缓解基于冲突的缓存攻击。载于：第 30 届 {USENIX} 安全研讨会（{USENIX} Security 21）（2021）7. Shusterman, A., Avraham, Z., Croitoru, E., Haskal, Y., Kang, L., Levi, D., Meltser, Y., Mittal, P., Oren, Y., Yarom, Y.：通过缓存占用通道进行网站指纹识别及其现实世界中的实用性。IEEE《可靠与安全计算汇刊》**18**(5), 2042–2060（2020）8. Shusterman, A., Kang, L., Haskal, Y., Meltser, Y., Mittal, P., Oren, Y., Yarom, Y.：通过缓存占用通道实现稳健的网站指纹识别。载于：第 28 届 USENIX 安全研讨会（USENIX Security 19）。第 639–656 页（2019）9. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.：多重碰撞的生日悖论。载于：信息安全与密码学——ICISC 2006：第 9 届国际信息安全与密码学会议，2006 年 11 月 30 日至 12 月 1 日，韩国釜山。论文集 9。第 29–40 页。施普林格（2006）