Rowhammer复现实验

内存模板化

实验目的

内存模板化（Memory Templating）是Rowhammer攻击中的一个重要步骤，其主要作用是**预先识别和定位那些容易发生位翻转（bit-flips）的内存地址**，这些位置通常被称为“易受影响的行”（vulnerable rows）。这一过程对于攻击的成功至关重要，因为它帮助攻击者找到那些在Rowhammer攻击下最有可能被利用的内存位置。

此外，许多防御机制（如TRR，Target Row Refresh）会尝试检测和缓解Rowhammer攻击。通过模板化，攻击者可以找到那些即使在这些防御机制下仍然容易被攻击的内存位置，从而绕过这些防御。

实验条件

1. 目标处理器需要能逆向出物理地址到Dram地址
2. 目标系统要能实现虚拟地址到物理地址的转换，如果只能转换部分位也可以(比如使用THP可以控制虚拟地址的低21位，有些平台控制Row地址的位数都在这低21位当中)

实验效果

攻击者对受害者内存进行分析，最终得到的结果是检测出来的**易受影响的行**。

实验步骤

开源实现

可以通过**zenhammer, blacksmith, trrespass**等开源项目完成这一步骤。

手动复现

1. **确定虚拟地址到物理地址的映射**: 这一步骤一般并没有完全破解虚拟地址到物理地址的映射， 因为这一行为往往权限要求较高，**在Rowhammer攻击中我们只需要获得一些物理空间连续的内存即可**。一般通过**透明大页**(***推荐***)实现,或者使用**伙伴分配器**的特性。
   - **透明大页(THP)** :在开启了透明大页支持的操作系统上申请页时仍然是申请4 kb大小的页面，但**操作系统会自动将这些请求合并成2MB大小的透明大页**。也就是说攻击者可以通过这一机制申请到**低21位(2MB)连续的内存**，其低21位地址与物理地址相同，部分处理器架构中物理地址到Dram地址的映射与Row相关的地址就在低21位，因此控制这一部分就足够了。
   - **伙伴分配器**[1]: 攻击者通过利用伙伴分配器的确定性行为，迫使内核分配物理地址连续的内存。具体来说，攻击者**持续使用带有** `MAP_POPULATE` **标志的** `mmap` **系统调用，请求**

分配较小的空闲内存块，直到小于10阶的块中剩余的空闲空间少于2 MB。当小于10阶的块中空闲空间不足2 MB时，攻击者发送两个2 MB的分配请求。为了满足第一个请求，内核会拆分一个10阶块（大小为4 MB）。此时，第二个请求将由连续的物理内存空间满足。由于第二个请求分配的内存的虚拟地址和物理地址的最低21位相同，因此该块在虚拟地址空间和物理地址空间中可能具有相同的偏移量。

2. **逆向物理地址到Dram地址的映射**:由于Rowhammer攻击依赖于对特定DRAM行的重复访问，以诱导相邻行中的位翻转。通过逆向映射，攻击者可以确定哪些物理地址映射到DRAM中的相邻行，从而选择合适的攻击行和受害者行。这一步骤参考**ZenHammer**[2]。
3. **内存初始化：** 分配大量的内存，并按照 Row地址 聚类。将目标内存区域初始化为特定的值，以便更容易检测位翻转(**比如 将攻击行（aggressor rows）初始化为全1（`0xFFFFFFFF`）将受害者行（victim rows）初始化为全0（`0x00000000`）**)
4. **锤击：** 进行 double-sided, half-double等pattern的锤击，需要快速、重复地访问攻击行
5. **检测位翻转：** 如果检测到位翻转，记录下该内存地址。

受害者内存放置

实验目的

将**受害者变量或数据放置到特定的物理内存行**中，这些行是通过**内存模板化（Memory Templating）** 步骤预先确定的易受影响的行。只有当受害者数据**位于特定的DRAM行**时，Rowhammer攻击才能成功诱导位翻转。如果受害者数据不在这些行中，攻击者的行为将无法影响到受害者行，攻击无法成功。

该步骤一般被称为**内存定位(Memory Massaging)**，根据攻击目标的不同(是否为内核数据，目标位于堆还是栈空间)会有不同的技术方案，此处介绍被广泛使用的一种方案以及SpecHammer中用于定位的方案。

实验条件

Memory Waylaying

1. Rowhammer攻击目标为可执行文件，通过修改它实现越权等攻击
2. 攻击者可以多次调用该可执行文件
3. 攻击者可分配大量内存

SpecHammer

实验效果

这一过程旨在使Rowhammer攻击产生实际效果，例如篡改受害者的数据。在此过程中，存在一个非特权级别的攻击者进程和一个受害者进程。攻击者**需要将受害者的内存定位到易受攻击的行**，并通过Rowhammer技术实现越权写入，最终成功修改受害者的数据。

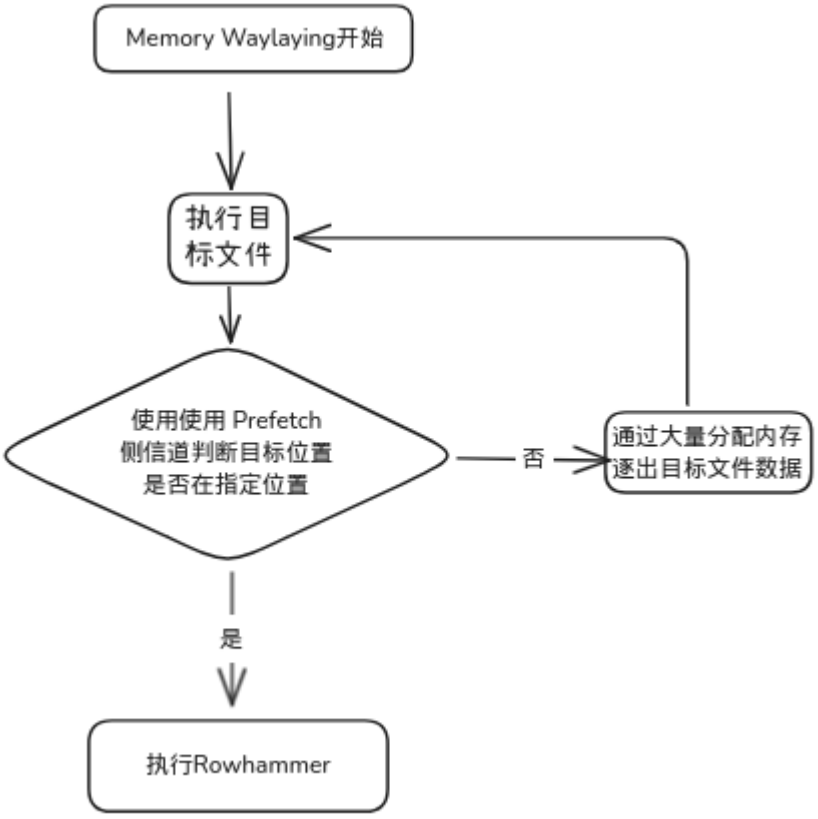
实验步骤

Memory Waylaying

在记忆伏击攻击中，攻击者通过监控页面放置来检测二进制文件和共享库中的某个偏移量是否映射到目标内存位置之一。在该技术中使用预取地址转换预言机(**Address-Translation Oracle**)[3]来执行此监控。。预取地址转换预言机为攻击者提供了两个虚拟地址是否映射到同一物理地址的信息。

1. **页面缓存驱逐（Page Cache Eviction）**：在Windows和Linux系统中，文件在首次访问时**以页为单位被缓存到文件页缓存**（**page cache**）中。Linux的页面缓存替换算法通常会优先驱逐非可执行页面，而不是可执行页面。然而，当页面缓存中需要填充只读可执行页面时，它也会驱逐可执行页面。攻击者可以利用操作系统的页面缓存机制，通过特定的内存访问模式，将目标页面从页面缓存中驱逐出去。**当目标页面再次被访问时，操作系统会重新分配一个新的物理页面来加载该内容**。通过这种方式，攻击者可以**反复触发目标页面的重新分配**，直到该页面被放置在攻击者预期的物理位置。
2. **预测目标页面位置（Prediction Oracle）**：攻击者使用 **Prefetch 侧信道**攻击来检测目标页面是否已经被放置在正确的物理位置。Prefetch 侧信道攻击利用了操作系统的直接物理映射（direct-physical mapping），**可以检测虚拟地址是否映射到特定的物理地址**，即使在地址空间布局随机化（ASLR）的保护下也能实现。
3. **等待目标页面就位（Waylaying）**：攻击者**周期性地驱逐目标页面**并重新访问它，同时使用 Prefetch 侧信道攻击检测其物理位置。
4. **实施Rowhammer攻击并验证翻转结果**

该步骤整体流程图如下:



SpecHammer

Linux系统的内存布局如下图所示:

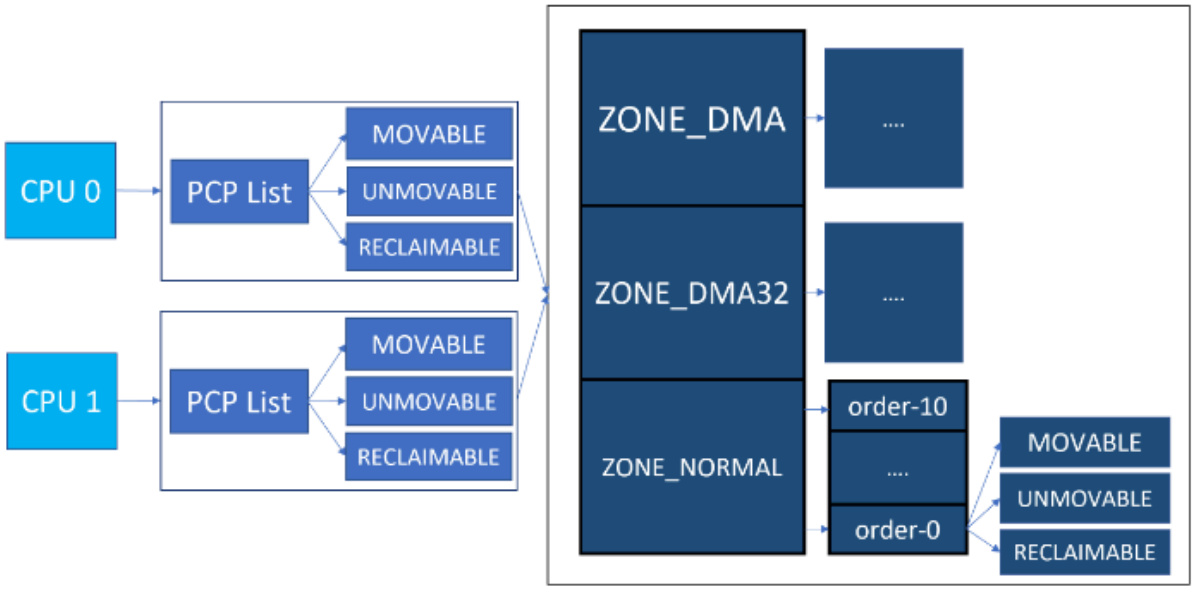


Fig. 3: **Linux memory organization**

其中**PCP列表**（又称页帧缓存）是一个**存储最近释放的内存页**（特别是order-0，即单个页）的缓存机制，它**与系统中的每个CPU相关联**。这些列表按照内存区域和页面迁移属性进行组织，并且遵循**先进后出**的原则。当系统需要分配一个order-0大小的内存页时，内存分配器会**优先从对应CPU的PCP列表中获取页**；如果PCP列表中没有可用页，它才会从内存分配器的空闲列表中获取。**释放页面时，这些页面会被放回它们所属CPU的PCP列表中**。即使释放的是大内存块的一部分，**每个单独的页面也会被单独放入PCP列表**，直到它们最终被合并回内存分配器的空闲列表。这个机制的目的是快速地重新利用那些最近在同一CPU上释放的内存页，从而减少对内存分配器的直接访问需求。

1. **填充页分配(Fodder Allocations)**: 目标变量可能不位于受害者堆栈的第一页。因此，我们必须首先计算在受害者分配包含目标的堆栈页之前，并分配相应数量的填充页。
2. **释放页(Unmapping Pages)：** 释放易受影响的页，将其收回至PCP 列表中，随后释放掉填充页，将其放置在同一个List中较上的位置。
3. **启动受害者程序：** 创建受害者进程，并强制其执行预测的内存分配操作，同时针对堆栈分配进行干预。在目标分配发生之前，任何先前的分配都会将填充页从PCP 列表中移除，从而确保堆栈分配被迫使用目标页。

---

1. Yes, {One-Bit-Flip} Matters! Universal {DNN} Model Inference Depletion with Runtime Code Fault Injection ↩
2. {ZenHammer}: Rowhammer Attacks on {AMD} Zen-based Platforms ↩
3. Prefetch Side-Channel Attacks: Bypassing SMAP and Kernel ASLR ↩