1. 基于硬件分支事件的ROP攻击防御方法

本文提出了一种基于硬件分支处理的ROP攻击防御方法（简称BranchChecker）。该方法通过硬件性能监控单元（Performance Monitor Unit，简称PMU）的采样事件触发性能监控中断（Performance Monitoring Interrupt，简称PMI）进行检测，基于最近分支记录（Last Branch Record，简称LBR）获取精确的分支信息，结合内存指令信息和系统调用参数信息，根据相应的检测算法进行配件检测，并识别出ROP攻击可能利用的配件链。该方法面临如下挑战：

1. 检测出发点选择策略：选择哪一种或多种的PMU事件类型作为检测触发点，既能保证对所有可能用于ROP攻击的指令片段都进行了检测，同时保证检测次数尽量减少以降低性能开销？
2. 攻击检测策略：采用何种检测策略进行配件和配件链检测，能有效检测ROP攻击，避免历史覆盖攻击、短配件链攻击等问题？

本节将先描述本文针对的威胁模型，然后详细描述BranchChecker的检测触发点选择以及攻击检测策略。

2.1威胁模型

本文主要面向应用程序进行保护，假定硬件，操作系统，动态链接器是可信的，目标操作系统支持数据不可执行技术（DATA Execution Prevention，简称DEP）。攻击者可以通过远程或本地正常终端访问应用，可以利用应用程序中的内存漏洞任意读取或修改应用数据内存页内容，也可以通过这些漏洞获取所有内存空间排布信息，但DEP技术限定攻击者不能通过数据插入恶意代码或直接修改应用代码，攻击者仅能操控内存数据，利用应用本身的代码进行攻击，即进行代码重用攻击。

2.2 检测触发点选择

本文基于PMU采样事件触发的PMI中断作为检测点，需要针对所有可能用于ROP攻击的指令片段进行检查。ROP攻击的指令片段的一个重要特征是：需要通过分支指令来实现指令片段间的衔接，因此，本文需要在分支相关事件中进行筛选，降低采样事件频率，同时不遗漏ROP攻击相关分支。

本文结合ROP攻击特征及硬件相关特性进行了待检测分支的选择：

1. 用户态vs内核态：本文针对应用程序进行保护，假定操作系统是可信得，因此所需检查的分支均为用户态分支；
2. 执行vs提交：现代CPU为了提高效率基本都支持乱序执行技术，部分分支会被投机执行，但最终可能会被抛弃，最终不会提交，所以分支可分为执行的分支和提交的分支。ROP攻击过程中的分支是程序真正运行时用到的分支，都属于会提交的分支（只是攻击者破坏了控制流完整性，使其偏离了程序开发人员的本意），因此仅需对提交的分支进行检测；
3. 跳转vs不跳转：ROP攻击链所需要的分支应该均为跳转的分支，如果不跳转，那么直接执行下一条指令，攻击者不能利用该分支跳转到他预期的目标，因此仅需对跳转的分支进行检测；
4. 预测正确vs预测失败：现代CPU的分支预测机制主要基于历史信息来对分支进行预测，由于ROP攻击会让分支偏离原有的跳转目标，导致其与正常历史信息不一致，让分支预测失败，因此仅需对预测失败的分支进行检测；
5. 直接vs间接：gadget是以间接分支结尾的指令片段，因此仅需对间接分支进行检测（在不能直接修改程序代码的前提下，攻击者无法修改直接分支的目标）。

综上所述，我们所需检查的最理想分支为：用户态的、提交的、跳转的、预测失败的间接分支。但实际中各类CPU支持的PMU分支事件粒度略有不同，并不能保证能够包含该最理想分支相应的采样事件，系统实现一节中会对具体的采样事件进行描述。

2.3 ROP攻击检测策略

BranchChecker对ROP攻击的检测包含三个部分：配件检测、配件链检测以及ROP攻击判定。

* **gadget检测**

配件是以间接分支指令结尾的短指令片段，本文针对配件“短”以及call-ret特性，提出了常规的配件检测方法：当满足下列条件之一时，认为当前指令片段为配件：

1. 非调用型返回指令：对于所有的返回指令，如果目标地址的上一条指令不是调用指令，那么将该指令片段识别为配件；
2. 短指令序列：如果从上一个检测分支到当前分支的指令长度低于一定阈值（maxGadgetLength），那么将该指令片段识别为配件。

在该检测条件下，超长配件会被识别成正常指令片段。为了缓解该问题，BranchChecker引入了对长无效片段进行识别和过滤。长无效片段是指副作用较小的长指令片段。补充检测条件如下：

如果在上一个疑似配件末尾到下一个疑似配件开始，系统状态（本文用系统架构寄存器的值来表征系统状态）的修改粒度低于一定阈值，将该指令片段标为长无效（long-nop）配件。

* **配件链检测**

配件链检测实际上是对连续的配件数目进行统计，BranchChecker采用以下策略对配件链长度进行统计：

1. 初始配件链长度为0；
2. 如果是当前指令片段被识别为配件，那么将配件链长度加1；
3. 如果当前指令片段为长无效配件，那么配件链长度不变；
4. 非2）和3）的情形则将配件链长度清0。

* **ROP攻击判定**

ROP攻击需要一串配件链（也就是一系列的配件）来完成，该链长度与常规执行情况下可能出现的正常配件链长度有较大的区别，因此本文沿用了常规的ROP攻击判定方法：

当配件链长数超过一定阈值（minChainLength）时， BranchChecker认为受保护的应用遭受了ROP攻击。

该判定方法适用于最常见的ROP攻击情形，但不适用于短配件链ROP攻击。

因此，针对短ROP攻击的重要特征：攻击者通过直接利用配件来进行敏感系统调用（如execve “/bin/sh”）来达到攻击目的。BranchChecker引入系统调用参数检测方法：

敏感系统调用中的参数和上一个配件时系统状态（本文用系统架构寄存器的值来表征系统状态）进行比较，如果核心参数来源于上一配件时的状态，那么认定当前遭受了ROP攻击。

BranchChecker综合使用以上两种判定方法来进行ROP攻击的自动检测