# 评估

本节将对PerfCFI原型系统的性能及安全性进行评估。

## 性能评估

我们的原型系统运行实验环境为：Intel i5-3320M CPU 2.6GHz 、4G RAM；操作系统为Ubuntu 15.04 X86\_64（内核版本3.13.19）。我们采用常用的SPEC2006作为Benchmark对PerfCFI的性能开销进行了评估。

首先，我们针对SPEC2006中分支的相关事件数进行了采样，如表3所示：

Table.3 Branch event count of SPEC2006 benchmark

表3 SPEC2006上分支相关事件计数

|  |  |  |  |
| --- | --- | --- | --- |
|  | 执行的间接分支数 | 预测失败的间接分支执行事件数 | 所需检查的分支减少比例 |
| 429.mcf | 1254075 | 8556327 | 85.34% |
| 445.gobmk | 33486205 | 808133481 | 95.85% |
| 456.hmmer | 1263206 | 8658138 | 85.41% |
| 462.libquantum | 695361 | 5009294 | 86.11% |
| 464.h264ref | 3860320 | 26420321 | 85.38% |
| 471.omnetpp | 3884031 | 28671677 | 86.45% |
| 473.astar | 1278374 | 8642528 | 85.20% |
| 433.milc | 653106 | 4457463 | 85.34% |
| 444.namd | 1185799 | 8169297 | 85.48% |
| 447.dealII | 607929978 | 4557668892 | 86.66% |
| 450.soplex | 2529271 | 17239736 | 85.32% |
| 470.lbm | 665000 | 4557532 | 85.40% |
| 482.sphinx3 | 10352577 | 75785775 | 86.33% |

由上表可以看出本文利用硬件BPU的特性，能够将所需检查的间接分支平均减少86.49%（最多的减少了95%，最少减少了85.21%）。与CFIGuard [19]每16个分支做一次检查相比，PerfCFI所需进行的检查数约为其2.17倍，考虑到系统调用进行的检查，两者实际进行检查数的比率应该低于该值，因此本文检测方法引入的性能开销应该在6.3%以内（16\*2.17=6.3）。

为了进一步确认PerfCFI真实引入的性能开销，本文对开启PerfCFI和未开启PerfCFI下的SPEC2006程序分别进行了性能采样（每个程序运行10遍取平均值），性能评估结果对比如图3所示。

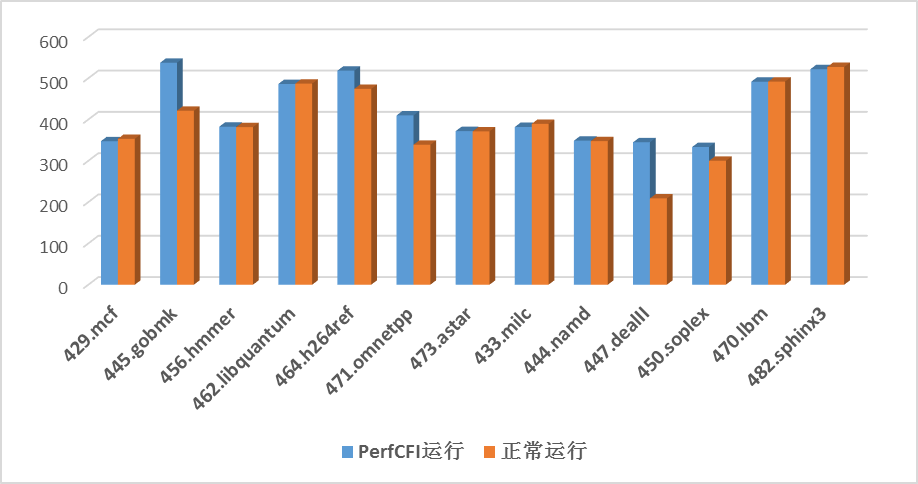


Fig.3 performance cost of PerfCFI on SPEC2006 benchmark

图3 PerfCFI在SPEC2006上的性能开销

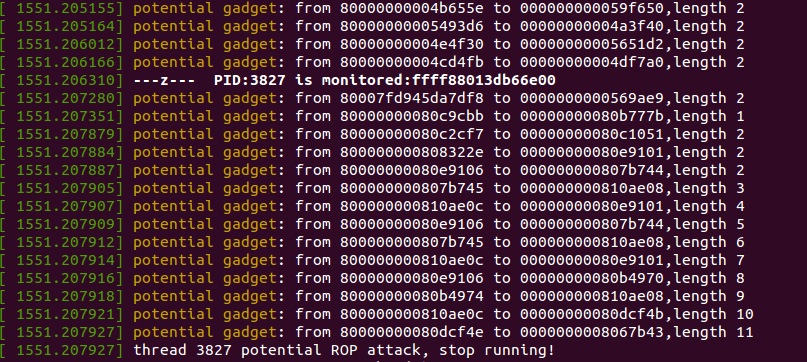
实验结果表明：PerfCFI平均引入5.4%左右的性能开销，对于绝大部分SPEC程序开销均在3%以内，少部分程序开销超过了10%，总体来说，性能开销在可接受的范围以内。

## 安全性评估

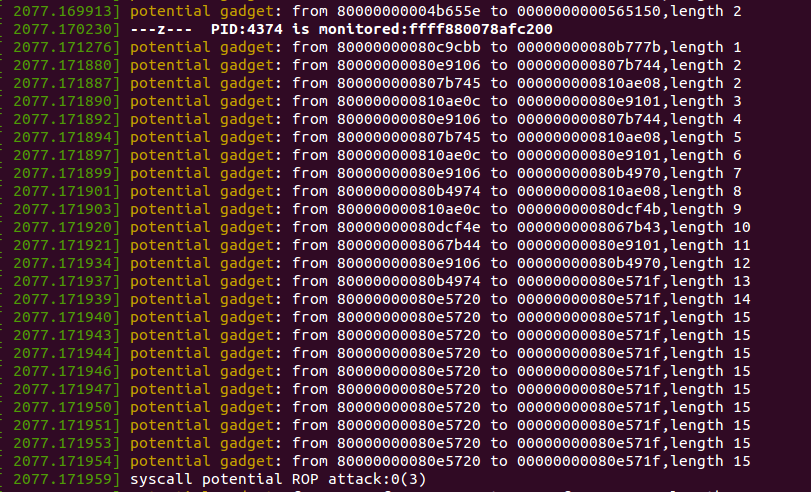
本文提出了PerfCFI检测方法，不同于kbouncer[10]和ROPecker[1]，是针对所有可能用于攻击的间接分支进行ROP检测，能够有效避开两者由于LBR有限引起的绕过问题。针对通过kbouncer和ROPecker会被长gadget打破检测链的问题，本文引入了两种新的检测方式：

1. 本文提出了长无效gadget概念及检测方法，能够有效检测gadget链中基本不修改架构状态的长gadget，规避此类长gadget引入历史刷新及破坏攻击问题；
2. 本文引入的系统调用参数检测方法，通过判断其与最近检测出gadget时保存的系统架构值之间相似度来检测ROP攻击。对于ROP攻击链中引起大量架构状态修改的gadget，虽然会打破本文的常规ROP检测链。但在进行最终高危行为（通常为系统调用）之前，攻击者依旧需要短的gadget链来恢复系统调用所需的状态，会被PerfCFI检测到并保存当时系统状态，如果，在攻击者进行系统调用时被PerfCFI拦截，。

根据分析可知，本文提出的PerfCFI方法能够有效的避开kbouncer和ROPecker的刷新历史以及破坏攻击问题。为了进一步验证PerfCFI的有效性，本文重现了几个现实的攻击实例[20,21]，以jad overflow[20]为例，实验结果如图4所示，红框为检测到的ROP攻击链以及ROP攻击结果。实验结果表明，PerfCFI能够有效的检测出这些攻击，复现完整的攻击gadget链，并及时阻止程序的继续运行。



a) maxChainLength=10



b)maxChainLength=20

Fig.4 PerfCFI running result on jad attack program: maxlength =10 and maxLength =20

图4 PerfCFI在jad攻击程序上的运行结果：maxLength=10和maxLength=20两种情形

为了进一步验证PerfCFI能否检测较短gadget链构成的ROP攻击。本文构造了一个通过较短的gadget链（未超过阈值)调用系统调用来完成ROP攻击的攻击实例，如图5所示，该攻击利用缓冲区溢出漏洞进行ROP攻击，攻击链由3个gadget构成：1) pop rdi;ret ; 2) pop rax;pop rdx;pop rbx;ret ; 3) syscall。

#include <stdio.h>

int main() {

char name[64];

printf("%p\n", name); // Print address of buffer.

puts("What's your name?");

gets(name); // buffer overflow bug

printf("Hello, %s!\n", name);

}

return 0;

}

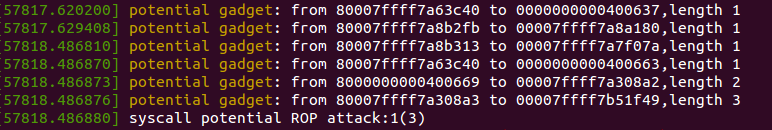


Fig.5 Victim program and ROP gadget chain check result

图5. victim程序以及ROP gadget链检测结果

实验表明，PerfCFI能够有效的检测出通过较短的gadget链进行系统调用的ROP攻击。