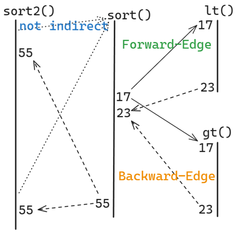
基于控制流签名的上下文敏感控制流完整性机制研究

CFS-CFI: Enforcing Context-Sensitive Control-Flow Integrity Based on Control-flow Signatures

张军，吕林航，郑晨，张玉清
联系方式：郑晨 zhengchen@iscas.ac.cn

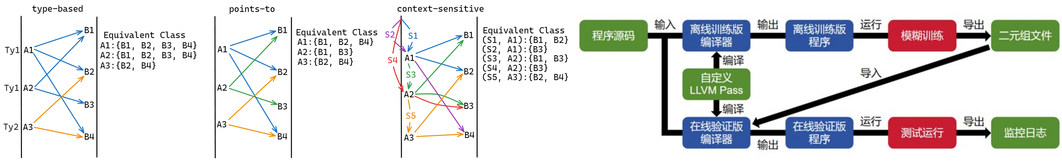
背景动机

① 控制流完整性通过保障间接控制权转移（如间接跳转指令等）的正确性，防御控制流攻击；
② 现有的控制流完整性方案仍然存在改进空间：通过引入更多上下文信息以更好地刻画控制流图，或者改进上下文信息的获取方式以优化性能开销。



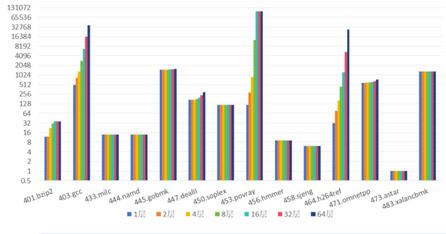
系统设计

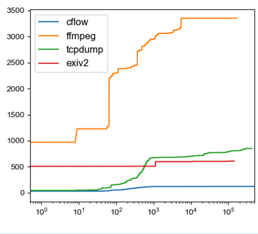
① 基于控制流签名方法，用增量累积方式记录程序执行时路径，将程序基本块的信息体现为控制流签名，与间接控制权转移组成二元组；
② 提出CFS-CFI 的控制流完整性方案，使用控制流签名作为间接控制权转移发生时的上下文信息，使用 LLVM Pass 进行代码插桩，于离线训练阶段保存控制流签名、导出二元组集合，于在线验证阶段检查生成的二元组是否与训练阶段相同；
③ 在离线训练阶段借助 AFL 框架对程序进行遍历，收集程序能生成的所有二元组；
④ 利用 ARM PA 硬件特性，使用 PACGA 指令减少控制流签名计算的性能开销。



实验结果

① 离线训练得到的二元组集合的规模与程序本身的结构、控制流签名记录的执行路径深度有关，程序本身的结构越复杂、记录的路径深度越深，训练阶段所收集到的二元组的规模越大；
② 在数十万次的执行后，AFL 框架收集的二元组数量趋于稳定，收敛趋势较为显著；
③ CFS-CFI 在线验证阶段的性能与现有的控制流完整性方案相仿，相较于其他基于 ARM PA 实现的方案，控制流签名的计算不依赖指针空闲位的设置，并节省CPU运行资源；
④ CFS-CFI 能够与后向控制流完整性方案如 PAC-RET 相结合，进一步增强对控制流的保护；





测试程序	AArch64		X86	
	CFS-CFI	PACTight-CPI	OS-CFI	μCFI
401.bzip2	0.05%	0.8%	0.4%	1.82%
444.namd	0.84%	0.5%	3%	-0.01%
445.gobmk	-0.15%	1.0%	11.2%	18.63%
447.dealII	5.08%	0.6%	9.6%	-
456.hmmer	-0.52%	0.4%	2.3%	0.78%
450.soplex	0.86%	0.02%	2%	3.37%
453.povray	36.37%	43.8%	9.9%	-
458.sjeng	4.83%	1.2%	2%	18.36%
471.omnetpp	4.85%	36.4%	13%	-
473.astar	6.59%	1.0%	2.2%	13.97%
483.xalancbmk	26.31%	23.5%	12.8%	-
Average	6.99%	5.8%	7.6%	12.74%

典例分析

① 提取 456.hmmer 中文件读取部分的代码，构造出具有不同执行路径的程序结构，尝试进行栈溢出攻击并被 CFS-CFI 识别。验证了 CFS-CFI 能够将程序的执行路径提取为上下文信息，是上下文敏感的控制流完整性方案；
② 提取 exiv2 中导致了 CVE-2025-26623 的代码片段，利用 Use After Free 漏洞构造了 fastbin 攻击，通过覆盖虚函数表指针试图修改控制流信息。而 CFS-CFI 发现调用析构函数时的控制流签名与训练阶段不符，成功识别到了此次攻击，说明 CFS-CFI 能够防御修改虚函数表的控制流攻击，具有实用价值。

```
hmmer *SampleOpen(int a) { // P1
  if (a > 20) {
    hm->fp = &CallA; // P2
    hm->fp(); // I1
  } else if (a > 10) {
    hm->fp = &CallB; // P3
    hm->fp(); // I2
  } else {
    hm->fp = &CallC; // P4
    hm->fp(); // I3
  }
  cin >> (hm->s); // buffer overflow, P5, skipped
  return hm;
}
```

```
class TiffDirectory : public TiffComponent { // P6, skipped
public:
    ~TiffDirectory() { // unique_ptr.h, I3, D6, P5, skipped
        for (auto &&component : components_) {
            delete component; // I6
        }
    }
    TiffComponent *doAddPath(UniquePtr tiffComponent) override { // D10
        return addChild(std::move(tiffComponent));
    }
    Components components_;
};
```