# Effective Program Debloating via Reinforcement Learning

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/57721832-cf97-4b33-ab8a-30024cb29007/ccs182d4.pdf

## 当前论文引出的问题

由于当前环境下，软件规模急剧增长，降低了程序的性能，以及增加了攻击面和安全隐患，并且由于当前的软件工程实践中将大量代码一刀切的进行代码服用，并将其打包到可以复用的源码中，攻击着可以不通过插入源码就可以调用原有的一些代码来执行，造成了安全风险。所以很多研究针对软件规模的增长进行研究，用于简化程序。

## 现有的研究

1. 粗粒度简化程序：基于动态分析的方式获得程序执行的信息，从而简化运行在DOCKER上的容器，并根据用户给定的约束将复杂的容器分解为多个简单容器。
2. 细一点的粒度简化程序：通过减去不用的方法和类或者通过基于函数级别依赖关系的静态分析来简化应用程序，只留下加载或者运行时使用的函数

3. 有的研究着眼点在程序运行时的内存膨胀。
4. 目前移动appios中可以根据用户设备类型去自动下载相关内容（即不下载程序的全部内容），但需要开发人员对代码进行标记，所以使用较少

### 现有研究的限制与对比

1. 现有的方法较为保守，只删除不可达代码，而本论文中提出的CHISEI方法同时也删除冗余代码。
2. 其他的冗余检测方法，只是检测哪些代码可能是冗余的，但删除该代码需要用户进行干预。而本论文提出的方法将自动删除冗余代码。而且目前的方法都
3. 目前都有些方法是对现有程序的轻量版本进行补充，但限制是需要获得该程序但源码，并要花费大量的人力。

---

## 提出的方法

针对上述问题提出了CHISEI系统或方法。

### 目的

将要被改造的程序以及被改造后所需功能的高级规范作为输入，改造或者简化待简化的程序，阿最后输出一个与高级规范需求文档所对应的程序。

### 方法建立的基础

使用强化学习为工具，并建立，并完善了一个统计模型，来确定每个候选的最小化程序通过测试脚本的可能性，并且抓取程序元素间的依赖性来让该系统进行最小化搜索。

### 系统的特点

1. 对待检测程序的语言和规范说明，不可知，因为模型从程序的矢量表示以及性能测试结果进行学习，所以又被称为是一个黑盒测试系统
2. 能够有效收敛到最小化程序，性能优于现有工具，并在后面提到的10个测试软件中消除来6个CVE，并平均减少来66.2%的小组件（每个软件的函数）。

---

## 评判标准

该论文针对此问题及系统提出来五个评判标准来评判CHISEL

1. 最小化：依据需求规范将程序尽可能最小化
2. 有效性：最小化程序等效未简化的程序（根据需求文档最小化程序，所以保证CHISEL的可靠性）
3. 健壮性：不产生新的错误和漏洞
4. 自然性：生成的简化代码是否可拓展和可维护（不将程序变得面目全非保证可拓展性和可维护性）
5. 通用性：能够处理各种程序和规范（将程序和标准需求文档当作黑盒，保证通用性）

### 与C-Reduce和PERSES的对比

**是否满足标准**

| Name | CHISEL | C-REDUCE | PERSES |
| --- | --- | --- | --- |
| 最小化 | ☑ | ☑ | ☑ |
| 有效性 | ☑ | ☐ | ☐ |
| 健壮性 | ☑ | ☑ | ☑ |
| 自然性 | ☑ | ☐ | ☑ |
| 通用性 | ☑ | ☐ | ☐ |

性能是c的7.1倍是p的3.7倍

---

# 测试工作

1. 用10个unix运行软件进行简化代码，消除了6个cve，并平均减少了66.2%的小组件函数
2. 运用AFL模糊测试软件测试了3天验证了程序的健壮性

---

# 结论

1. 提出的系统确实能够有效的简化程序并且减小攻击面
2. 一个通用的用于程序简化领域的强化学习框架，与程序的语言和规范无关
3. 在10个UNIX系统的程序上执行，实验验证了它能够减少已知漏洞和攻击面

---