OSmart: Whitebox Program Option Fuzzing

基于白盒分析的程序选项模糊测试

王柯林 陈孟达 和亮✉ 苏璞睿✉ 蔡彦 陈炯峄 张斌 冯超 唐朝京

In CCS'24 [2024.10.15]

mengda2020@iscas.ac.cn heliang@iscas.ac.cn

研 究 概 述

为了融合多种功能，现实中的程序提供数量众多的选项。现有的漏洞挖掘方法通常针对程序输入进行变异，忽略了选项带来的影响。此外，多个选项也会同时作为一个组合来使用，而有效的选项组合可以帮助发现未知漏洞。现有选项相关研究主要依赖官方文档来获取选项列表，并通过自然语言处理或随机策略等方法构造选项组合。它们均无法提取官方文档中未记录的选项及组合，从而遗漏大量有效的选项组合。

OSmart首次从代码中自动提取合法选项及组合，并借助两种模糊测试策略挖掘出程序安全漏洞。

Existing Works	Option Extract	Option Grouping	Documented Option	Undocumented Option	Option Type	Option Impacts	Option Group	Bugs
AFL++-argv* [22]	generation-based	generation-based	✓	✓	✗	✗	✓	✓
CrFuzz [45]	document-based	enumeration	✓	✗	Manual	✗	✓	✗
POWER [28]	document-based	random	✓	✗	Manual	✗	✓	✗
ConfigFuzz [52, 53]	document-based	grammar-based	✓	✗	Manual	✗	✓	✗
CarpetFuzz [49]	document-based	NLP-based	✓	✗	Manual	✗	✓	✗
OSmart	code-based	impact-based	✓	✓	✓	✓	✓	✓

*✓=unpractical or incomplete.

图1. OSmart与现有工作之间的比较

设 计 实 现

1. 程序选项提取

程序在处理选项时采用一种较为通用的模式：

① **循环(Loop)**：选项解析过程循环比较argv数组中的元素

② **选择(Select)**：采用switch-case或strcmp函数将argv的元素与某些常量字符串进行比较

③ **目标(Target)**：比较操作针对的变量来源于argv

遍历选项影响图(Option Impact Graph)来确定选项的控制域，从而确定直接影响变量(DIV)以及其类型

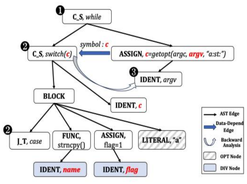

图3. LST命令解析模式

2. 选项影响分析

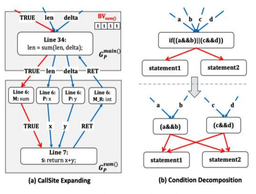

图4. 选项影响分析

将直接影响变量(DIV)视为污点源，通过以依赖分析识别其影响范围。传播过程遵循以下规则：

- **DIV定位**：结合LST解析模式匹配结果完成初始定位
- **过程间分析**：将实参与形参间的数据依赖关系及调用点到函数定义间的控制依赖关系进行连接
- **条件语句**：将判断条件分解为最小判断条件
- **全局变量**：维护全局映射表进行记录
- **结构体变量**：采用字段敏感分析的传播规则

3. 选项组合生成

- **选项影响图生成**：以选项影响变量的控制流和数据流依赖分析结果构建选项影响图
- **选项组合提取**：使用笛卡尔积自动生成潜在的选项组合

图5. 利用笛卡尔积生成数据依赖选项组示例

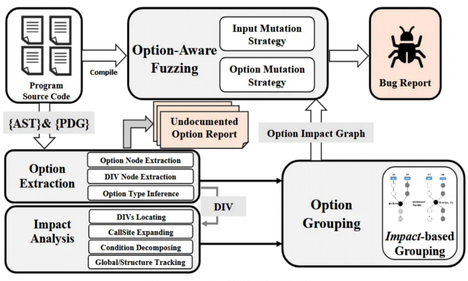

图2. OSmart整体设计框架

4. 选项感知模糊测试

- **选项变异策略**：固定输入内容，根据选项类型对选项值进行变异
- **输入变异策略**：固定选项组合，分配初始选项值，测试后对输入变异

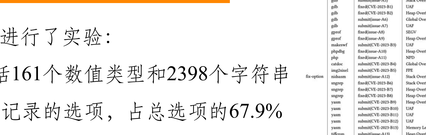

图6. 选项变异&输入变异

实 验 效 果

OSmart在56个广泛使用的开源程序中进行了实验：

- 提取了**4924**个直接影响变量，包括161个数值类型和2398个字符串类型。发现**193**个在官方文档中未记录的选项，占总选项的67.9%
- 生成了**12560**个有意义的选项组，显著高于官方文档中记录的61个
- 相比AFL++额外发现**40.3%**的代码覆盖，未记录选项发现了12833行额外的代码覆盖，占总比例的4.45%
- 发现了**51**个新的安全漏洞，申请了18个CVE。包括栈溢出、堆溢出、空指针解引用、UAF等类型漏洞

图7. OSmart在56个程序中的实验效果

图8. OSmart发现的漏洞列表