不稳定代码静态分析的设计与实现

陈广庆

指导老师：王立斌

原代码

```
1 char *buf = ...;
2 char *buf_end = ...;
3 unsigned int len = ...;
4 if (buf + len >= buf_end)
5     return; /* len too large */
6 if (buf + len < buf)
7     return; /* overflow, buf+len wrapped around */
8 /* write to buf[0..len-1] */
```

原代码

```
1 char *buf = ...;
2 char *buf_end = ...;
3 unsigned int len = ...;
4 if (buf + len >= buf_end)
5     return; /* len too large */
6 if (buf + len < buf)
7     return; /* overflow, buf+len wrapped around */
8 /* write to buf[0..len-1] */
```

原代码

```
1 char *buf = ...;
2 char *buf_end = ...;
3 unsigned int len = ...;
4 if (buf + len >= buf_end)
5     return; /* len too large */
6 if (buf + len < buf)
7     return; /* overflow, buf+len wrapped around */
8 /* write to buf[0..len-1] */
```

原代码

```
1 char *buf = ...;
2 char *buf_end = ...;
3 unsigned int len = ...;
4 if (buf + len >= buf_end)
5     return; /* len too large */
6 if (buf + len < buf)
7     return; /* overflow, buf+len wrapped around */
8 /* write to buf[0..len-1] */
```

优化后

```
1 char *buf = ...;
2 char *buf_end = ...;
3 unsigned int len = ...;
4 if (buf + len >= buf_end)
5     return; /* len too large */
6 /* write to buf[0..len-1] */
```

优化后

```
1 char *buf = ...;
2 char *buf_end = ...;
3 unsigned int len = ...;
4 if (buf + len >= buf_end)
5     return; /* len too large */
6 /* write to buf[0..len-1] */
```

不稳定代码

涉及**未定义行为**，
且在被编译器优化之后，
目标代码跟程序员原意不一致，
甚至可能导致**安全漏洞**的代码。

前人工作

SOSP13wang

PLDI15hathhorn

https://github.com/kframework/c-semantics

https://github.com/xiw/stack

前人工作

SOSP13wang

PLDI15hathhorn

https://github.com/kframework/c-semantics

https://github.com/xiw/stack

多年未维护，所用框架LLVM已从3.4升到9.0

无法分析使用新编译器的系统

我的工作

- 自学程序静态分析
- 阅读相关论文
- 研究大量LLVM源码
- **重写了原项目大部分代码**
  （近4K行C++代码）

我的工作

- 自学程序静态分析
- 阅读相关论文
- 研究大量LLVM源码
- **重写了原项目大部分代码（近4K行C++代码）**

成果

基于最新的编译框架LLVM

中间代码扫描管理器

可满足性模理论求解器等

进行重新设计并优化了缓存机制等实现细节

重焕活力、提高效率

我的工作

- 自学程序静态分析
- 阅读相关论文
- 研究大量LLVM源码
- **重写了原项目大部分代码（近4K行C++代码）**

成果

基于最新的编译框架LLVM

中间代码扫描管理器

可满足性模理论求解器等

进行重新设计并优化了缓存机制等实现细节

重焕活力、提高效率

代码：https://github.com/gou4shi1/stack

论文：https://cloud.goushi.me/bachelor.pdf

演示：https://asciinema.org/a/wZ0SL4J7o7eCAsvFvMcaDbjBK

不足

- 未对核心算法作出改进。
- 未支持所有不稳定代码。
- 报告为纯文本不够可读。

不足

- 未对核心算法作出改进。
- 未支持所有不稳定代码。
- 报告为纯文本不够可读。

展望

- 结合更精密的形式语义改进算法。
- 探索编译器的优化机制，支持更多不稳定代码。
- 开发工具把纯文本的报告转换成现代化的网页。
- 逐步应用到各大系统中，发现可被外部利用的安全漏洞，提交补丁。

Thanks