目录

[1 课题概述 1](#_Toc385145426)

[1.1 课题来源 1](#_Toc385145427)

[1.2 课题目的 1](#_Toc385145428)

[1.3 课题意义 1](#_Toc385145429)

[2 国内外研究概况 2](#_Toc385145430)

[3 总体设计 3](#_Toc385145431)

[4 课题研究进展计划 4](#_Toc385145432)

[5 参考文献 5](#_Toc385145433)

# 1 课题概述

## 1.1 课题来源

程序中一个漏洞可能带来无法估量损失，特别是在广泛使用的协议或者软件中。因此，软件的安全性越来越受到人们的重视，如何检测软件中存在的安全问题也逐渐成为软件工程领域的研究热点。而对源代码进行分析与检测是一种有效的检测软件安全问题的方法。

源代码分析方法可分为静态分析和动态分析两种。静态分析是指在不运行代码的方式下，通过词法分析、语法分析、控制流分析等技术对程序代码进行扫描，验证代码是否满足规范性、安全性、可靠性、可维护性等指标的一种代码分析技术。符号执行技术是其中最常用的技术之一。由于静态符号执行技术存在路径空间爆炸等问题，将符号执行结合动态分析，以弥补符号执行的不足。

本课题采用动态符号执行技术进行源代码分析，主要分析程序中全局变量的类型，为进一步分析做准备。

## 1.2 课题目的

由于并发程序的不确定性，相对于单进程、单线程程序，对并发程序的分析是更为困难的。而在并发程序中，多个线程或多个进程经常需要访问同一个变量，需要进行同步与互斥。对于这些共享变量，读与写操作的意义是不同的。两个线程读取同一个共享变量，它们的执行顺序不会影响结果，而存在写操作时就不同了。因此，在进行进一步分析之前，先获取这些变量的类型是很有必要的。

本课题研究旨在深入了解熟悉动态符号执行技术和程序的编译过程和机制，并通过动态符号执行技术分析程序源代码，获取代码中所有全局变量的位置和类型。最后设计并实现一个源代码分析工具，来获取源代码中全局变量的读写类型，为下一步进行多线程的时序分析做准备。

## 1.3 课题意义

本课题采用动态符号执行技术，对源代码进行分析，最后将设计并实现一个获取程序中全局变量类型的工具，作为研究成果输出。该工具可以用在并发程序的分析中，以简化并发程序分析的过程。

# 2 国内外研究概况

目前，源代码分析根据是否需要执行程序分为静态分析和动态分析两种。

静态分析在不运行代码的方式下，通过词法分析、语法分析、控制流分析等技术对程序代码进行扫描，验证代码是否满足规范性、安全性、可靠性、可维护性等指标的一种代码分析。静态分析方法涉及的理论包括指称语义、公理化语义、操作语义、抽象解释理论等,常见的实现技术有模型检验、数据流分析、抽象解释、谓词转换、定理证明、类型推导、符号执行等。由于静态分析技术对代码的分析可能不准确，会产生一定的误报和漏报，静态分析技术的局限性也是很明显的。

而动态分析技术通过实际的执行程序，来检测与分析程序。动态分析技术在源代码分析，与漏洞检测上也有广泛的应用。陈华[8]通过动态分析技术对系统进行测试，收集执行信息，利用测试结果进行系统修复。

符号执行技术是广泛使用的一种静态分析技术。符号执行技术使用符号值表示变量，来模拟程序执行，得到的变量的值是由输入变 量的符号值和常量组成的表达式。符号执行分为过程内分析和过程间分析（全局分析）。过程内分析首先对待分析的代码构建控制流图，在控制流图入口节点上开始模拟执行，遇到分支节点时，使用约束求解器判断分支是否可行，然后对所有可行路径遍历分析，最后输出每条可行路径的分析结果。而过程间分析，则先对整个程序生成函数调用图，对每个节点进行过程间分析，最终给出每种可行的调度序列的结果。王晓亮[7]使用符号执行技术，在对路径进行遍历时，做数据相关性测试，以实现较精确的数据相关性测试。

动态符号执行是以具体数值作为输入来模拟执行程序代码，与传统静态符号执行相比，其输入值的表示形式不同。动态符号执行使用具体值作为输入，同时启动模拟执行器。然后，并从当前路径的分支语句的谓词中搜集所有符号约束。然后修改该符号约束内容构造出一条新的可行的路径约束，并用约束求解器求解出一个可行的新的具体输入，接着符号执行引擎对新输入值进行一轮新的分析。通过使用这种输入迭代产生变种输入的方法，理论上所有可行的路径都可以被计算并分析一遍。动态符号执行相对于静态符号执行的优点是每次都是具体输入的执行，在模拟执行这个过程中，符号化的模拟执行比具体化的模拟执行的花销大很多；并且模拟执行过程中所有的变量都为具体值，而不必使用复杂的数据结构来表达符号值，使得模拟执行的花销进一步减少。但是，动态执行的结果是对程序所有路径的一个下逼近，最后产生的路径可能会有所遗漏。

目前，已经有很多相关的分析工具，如：EXE[9]、CUTE[10]、DART[11]、SAGE[12]、KLEE[13]等。

# 3 总体设计

先使用llvm，将待分析的源代码文件转换成中间文件。在用动态符号执行的方法分析中间文件。

由于完全从头实现动态符号执行工程量太大，我们选择在klee的基础上实现分析源代码的工具。在klee模拟执行时，对全局变量的进行类型判断，并记录。获取每处全局变量在源代码中的位置，并将其对应的类型和位置保存。

输入：源代码文件

输出：记录源代码中各个全局变量每处调用的类型及位置的文件

# 4 课题研究进展计划

2014.1.1~2014.1.20 了解课题背景，收集资料，阅读相关论文与材料，安装环境

2014.2.10~2014.2.25 完成论文翻译和开题报告

2014.2.26~2014.3.15 完成需求分析和系统总体设计

2014.3.16~2014.5.10 系统实现

2014.5.11~2014.5.30 撰写论文

2014.5.31~2014.6.5 制作答辩ppt，准备答辩

# 5 参考文献

[1] Zamfir C, Candea G. Execution synthesis: a technique for automated software debugging[C]//Proceedings of the 5th European conference on Computer systems. ACM, 2010: 321-334.

[2] King J C. Symbolic execution and program testing[J]. Communications of the ACM, 1976, 19(7): 385-394.

[3] 林锦滨, 张晓菲, 刘晖. 符号执行技术研究[J]. 全国计算机安全学术交流会论文集, 2009.

[4] 蒋思远. 基于动态符号执行的测试工具设计与实现[D]. 华东师范大学, 2012.

[5] 丁保贞. 基于动态符号执行的代码分析工具的研究[D]. 上海交通大学, 2012.

[6] 崔宝江, 国鹏飞, 王建新. 基于符号执行与实际执行的二进制代码执行路径分析[J]. 清华大学学报 (自然科学版), 2009, 2.

[7] 王晓亮. 基于路径符号执行的数据相关性分析[D]. 中国科学院研究生院 (软件研究所), 2004.

[8] 陈华, 钱剑飞, 俞瑞钊. 基于动态执行特性分析的 JAVA 软件系统架构恢复[J]. 计算机应用与软件, 2005, 22(8): 30-31.

[9] Cadar C, Ganesh V, Pawlowski P M, et al. EXE: automatically generating inputs of [J]. ACM Transactions on Information and System Security (TISSEC), 2008, 12(2): 10.

[10] Sen K, Marinov D, Agha G. CUTE: a concolic unit testing engine for C[M]. ACM, 2005.

[11] Godefroid P, Klarlund N, Sen K. DART: directed automated random testing[C]//ACM Sigplan Notices. ACM, 2005, 40(6): 213-223.

[12] Godefroid P, Levin M Y, Molnar D A. Automated Whitebox Fuzz Testing[C]//NDSS. 2008, 8: 151-166.

[13] Cadar C, Dunbar D, Engler D R. KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs[C]//OSDI. 2008, 8: 209-224.