# 摘 要

# Abstract

目录

[摘 要 I](#_Toc389485113)

[Abstract II](#_Toc389485114)

[1 绪言 1](#_Toc389485115)

[1.1 课题背景 1](#_Toc389485116)

[1.2 课题的主要研究工作 1](#_Toc389485117)

[1.3 课题研究的目的和意义 1](#_Toc389485118)

[1.4 国内外概况 2](#_Toc389485119)

[2 4](#_Toc389485120)

[参考文献 5](#_Toc389485121)

# 1 绪言

## 1.1 课题背景

随着计算机科学技术的飞速发展，特别是Internet出现以来，计算机迅速普及，进入各行各业，千家万户。软件中一个漏洞可能带来无法估量的损失，特别是在广泛使用的协议或者软件中。比如，OpenSSL“Heartbleed”安全漏洞[1]至少影响几十万的服务器的安全，很多国内外知名的网站的安全都受到威胁，影响到广大网民的隐私和财产安全。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。

近年来，多线程程序设计技术已经广泛应用到软件开发中。同时，多线程技术的发展也带来了数据竞争的问题。线程之间的同步和互斥，是编写多线程程序时必须要考虑的。对于多线程程序的分析测试工作也比单线程程序要困难得多。针对源代码的静态分析技术，是一种有效的程序分析方法。

## 1.2 课题的主要研究工作

静态分析方法中，有一种常用的分析方法，符号执行。符号执行是以符号量代替程序中的变量值，并模拟执行程序，覆盖程序中的所有路径并根据路径上约束，求出变量取值的分析技术。而符号执行技术由于存在路径爆炸等问题[2]，实际使用时效率较低。为了解决这个问题，提高符号执行技术的可行性，P.Godefroid等人[3]将符号执行技术与动态分析技术想结合，提出了动态符号执行技术。本课题的主要研究工作就是使用动态符号执行技术，分析源代码文件，并给出源代码文件中全局变量每处调用的位置和读写类型。

## 1.3 课题研究的目的和意义

课题研究的意义：

（1）研究动态符号执行技术及相关的基础理论，并应用到程序分析中；

（2）给出全局变量的调用信息，为多线程时序分析和数据竞争检测提供便利。

## 1.4 国内外概况

程序分析技术有静态分析技术和动态分析技术两类。

静态分析技术是指在不执行程序的情况下，对程序源代码进行分析，验证代码是否符合规范性、安全性、可靠性、可维护性等要求的一种程序分析技术。静态分析工具一般包括预处理器、数据库、错误分析器、报告生成器四部分[4]。林阿龙[5]利用程序静态分析技术，设计了一个C源程序分析工具CAAT。常用的静态分析技术的实现方法有模型检验、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。

动态分析技术是通过实际地执行程序，检测程序执行过程中的状态、结果等。既有传统的测试人员手工测试，也有使用自动化测试工具进行测试的。如，IBM的Rational PurifyPlus[6]就是一个使用动态分析技术的测试工具，并且投入商业化使用。动态分析技术与静态分析技术相比，它关注的是程序的功能和表现，只能检测出实际存在的缺陷，但它更符合生产环境的需要。静态分析技术能够对程序中所有可能执行到的路径进行检测，能够发现潜在的缺陷，但是误报率高，容易将一些正确代码定位为缺陷。可以说，静态分析技术与动态分析技术各有所长，两者结合使用可以取长补短，更好的排除程序中的错误。

静态分析技术中非常重要的一种方法是符号执行。符号执行于1976年由James C．King[7]首先提出，经过三十多年的发展，至今仍然被广泛研究。符号执行技术在软件测试和程序验证中发挥着重要作用。符号执行是指在不执行实际程序的前提下，把源程序翻译为一种中间语言，用符号值表示程序变量的值，然后基于中间语言模拟程序执行来进行相关分析的技术。目前，符号执行技术被广泛应用在程序分析检测[8]和测试数据生成[9]中。

Cristian Zamfir和George Candea使用符号执行技术实现了一个用于软件自动Debug的工具[10]。只需要输入程序和错误报告，就能自动找到导致这个错误的执行路径。这为调试一些难以重现的bug提供了极大的便利。

由于符号执行存在路径空间爆炸等问题，结合动态分析的特点，P.Godefroid等人[3]提出了动态符号执行技术。动态符号执行与传统静态符号执行的区别在于输入值的表现形式不同，动态符号执行以具体值作为输入来模拟程序执行。由于动态符号执行使用具体值，模拟执行过程中的开销比使用符号值的符号执行要小很多。但是，也正是因为这个原因，动态符号执行产生的路径集合比符号执行要小，是一个近似的路径集合。

目前，已经有许多动态符号执行测试工具，大致可以分为三个系列。

P．Godefroid等人开发的DART[3]是最早的动态符号执行测试工具之一。它的测试目标包括了程序崩溃、断言失败等标准错误，只支持整数类型约束，用随机测试来处理遇到含有指针类型约束的情况。SMART[11]是基于DART开发的测试工具，在DART上增加了一个效率更高的搜索算法，目标是使组合测试输入生成效率提高。基于DART，P．Godefroid等人还开发了SAGE[12]，用于测试机器码的动态符号执行测试工具，该工具目前不支持指针解引用的符号化。SAGE在针对函数指针的处理上，增加了高阶测试输入数据生成的想法，有初步试验证实其有效性。

K．Sen等人开发的CUTE[13]，CREST[14]都是用于C程序测试的动态符号执行测试工具。CUTE不能处理含有符号偏移(输入变量作为偏移)的约束。CREST是基于CUTE之上做的扩展。CREST不支持函数指针调用，但支持位移运算(不支持位掩码)。实证研究[14]表明了CREST在1000-2000行代码的单元测试中的有效性。jCUTE[15]是K．Sen等人开发的用于测试Java程序的测试工具。

C．Caciar等人开发了EXE[16]，KLEE[17]和Rwset[18]，它们是另外一系列的动态符号执行测试工具。这三个测试工具也都是用于C程序测试的。KLEE基于EXE，为了应对路径爆炸的问题，对程序与外部环境的交互做了额外处理。和CREST—样，EXE也不支持函数指针。不同于DART和CUTE，KLEE可以处理含有指针符号、偏移符号的约束。Rwset也是对路径爆炸的问题做了处理，主要增加了对不必要的路径进行剪枝的算法。

现有的动态符号执行测试工具中，除了上述三个系列之外，还有许多针对面向对象编程语言的测试工具。比如，jFuzz[19]，它是K．Jayaraman等人基于JPF[20]开发的。jFuzz用于java程序测试，它能支持位运算。PEX[21]则是N．Tillman等人基于.Net平台开发的，用于测试C#程序。由于java和C#都没有指针类型，因此无需考虑指针约束。

# 2

# 参考文献

[1] Ding Y．SSL implementieren–aber sicher![J]．2014．

[2] 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，2009：404-408．

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

[4] 文昌辞，王昭顺．软件测试自动化静态分析研究[J]．计算机工程与设计，2005，26(4)：987-989．

[5] 林阿龙．C 源程序分析及理解的辅助工具—CAAT[J]．小型微型计算机系统，1992，1：001：9-15．

[6] URL：<http://www.ibm.com/developerworks/cn/rational/products/purifyplus>

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

[8] 杨宇，张健．程序静态分析技术与工具[J]．计算机科学，2004，31(2): 171-174．

[9] 赵跃华，阚俊杰．基于符号执行的测试数据生成方法的研究与设计[J]．计算机应用与软件，2014，31(2)：303-306．

[10] 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．

[11] Godefroid P．Compositional dynamic test generation[C]．Acm Sigplan Notices．ACM，2007，42(1)：47-54．

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

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

[14] Burnim J，Sen K．Heuristics for scalable dynamic test generation[C]．Proceedings of the 2008 23rd IEEE/ACM international conference on automated software engineering．IEEE Computer Society， 2008：443-446．

[15] Sen K，Agha G．CUTE and jCUTE：Concolic unit testing and explicit path model-checking tools[C]．Computer Aided Verification．Springer Berlin Heidelberg， 2006：419-423．

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

[17] 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．

[18] Boonstoppel P，Cadar C，Engler D．RWset：Attacking path explosion in constraint-based test generation[M]．Tools and Algorithms for the Construction and Analysis of Systems．Springer Berlin Heidelberg，2008：351-366．