计算机科学与技术学院

毕业设计（论文）

文献综述

设计（论文）题目： 基于动态符号执行的源程序分析

专 业 班 级： 计算机科学与技术1002班

学 生 姓 名： 徐华福 学号： U201014264

指 导 教 师： 李国徽

2014年 3月 1日

# 一、前言

## 1.1 写作目的

本课题研究的是“基于动态符号执行的源程序分析”。撰写本文的目的是整理动态符号执行相关的文献资料，归纳国内外前人在动态符号执行和源程序分析与检测上的研究工作，以参考和借鉴。

## 1.2 相关概念

本课题最终的目的是对源程序分析，因此必然会涉及到程序分析技术。程序分析是指对计算机程序行为进行自动分析的过程，主要可以分为静态分析技术和动态分析技术两类。

程序静态分析（Program Static Analysis）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。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运行程序，在输入不同的情况下跟踪程序执行，以检测分析程序的程序分析技术。与静态分析技术相比，动态分析技术侧重于对程序功能的测试。静态分析技术则是对程序代码的进行分析，检测出程序中存在的缺陷和不合理之处。

符号执行是静态分析技术的一种。符号执行是指在不执行程序的前提下，用符号值表示程序变量的值，然后模拟程序执行来进行相关分析的技术。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。

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，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。

## 1.3 综述范围

综述主要涉及程序分析、软件测试、多线程程序等方面。

引用的文献时间范围为：1976-2010年。

# 二、主体部分

程序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”安全漏洞危及到成千上万的服务器，很多知名的网站的安全都受到威胁，影响到广大网民的信息和财产安全。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。

一般说来，程序验证要求通过推理或者穷举的手段来判定程序的行为是否符合规约。由于要涵盖所有可能情况，而程序设计语言的复杂性使得程序的复杂性随着程序尺寸的增大呈指数级增长，同时证明任一程序正确与否本身是一个不可判定问题，因此程序验证目前只用于证明一些关键的核心模块的正确性而没有得到更广泛的应用。就目前而言，程序验证方法虽然可以保证软件质量，但是往往需要有一定经验的用户花费相当多的时间，因而并不一定能提高软件的生产率[1]。

另一方面，在现实的软件开发中大量的时间被用于发现和消除软件中的错误，也就是软件测试。除了一些大公司在大型软件系统开发中使用了一些自动测试手段[2]，在很多情况下，软件测试仍然停留在手工测试阶段。手工测试不仅效率很低，而且容易出错。测试任务往往很繁重，在资源有限、时间紧迫的情况下测试任务常常不能充分完成。测试和调试手段的匾乏己经成为制约软件生产率和软件质量的一个瓶颈。

程序检测按是否需要执行程序可以分为：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。

# 三、总结

静态分析技术与动态分析技术相比，它的优势是能够分析程序中所有可能的路径信息，覆盖率较高，有助于发现程序中潜在的缺陷，提高代码质量。同时，它也有一定缺点，由于是对代码进行分析，容易把一些正确的代码定位为缺陷，误报率高，排除误报需要花费额外的成本。

符号执行技术是静态分析技术中最广泛使用的技术之一，已经普遍应用在软件测试、测试数据生成甚至bug重现等中。而动态符号执行的出现，更是提高了符号执行技术的可行性。

在目前的众多的动态符号执行测试工具中，KLEE是功能较为完善的一个，它能够处理含有指针符号和偏移符号的约束。因此，在本课题研究中，选择基于KLEE项目进行源程序分析工具的开发。

# 四、参考文献

[1] Brooks Jr F P．The Mythical Man-Month，Anniversary Edition：Essays on Software Engineering[M]．Pearson Education，1995．

[2] Fewster M，Graham D．Software test automation：effective use of test execution tools[M]．ACM Press/Addison-Wesley Publishing Co，1999．

[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](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．

[19] Visser W，Havelund K，Brat G，et al. Model checking programs[J]. Automated Software Engineering，2003，10(2)：203-232．

[20] Jayaraman K，Harvison D，Ganesh V，et al．jFuzz：A Concolic Whitebox Fuzzer for Java[C]．NASA Formal Methods．2009：121-125．

[21] Tillmann N，De Halleux J．Pex–white box test generation for．net[M]．Tests and Proofs．Springer Berlin Heidelberg，2008： 134-153．