软件安全静态检测技术与工具*

杨洪路[1,2],宫云战[2],高文龄[1],白哥乐[2]
(1.北京装甲兵工程学院,北京 100072;
2.北京邮电大学 网络与交换技术国家重点实验室,北京 100876)

摘 要:提出了一种软件安全漏洞的检测方法,重点介绍了静态测试。对当前基于源码分析的软件安全测试工具进行了分类并加以分析。

关键词:软件安全;漏洞;安全测试;静态分析;测试工具
中图分类号:TP311.5 **文献标识码**:A

Static analysis of software security techniques and tools

YANG Hong Lu[1,2], GONG Yun Zhan[2], GAO Wen Ling[1], BAI Ge Le[2]
(1.Armored Force Engineering Institute, Beijing 100072, China;
2.State Key Laboratory of Networking and Switch Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: Put forward the detection method of security vulnerabilities, putting emphasis on static testing. Lastly, some software security testing tools based on source-code analysis are introduced and classified into several categories.

Key words: software security; vulnerability; security testing; static analysis; testing tools

随着Internet的迅速普及和社会信息化程度的不断加深,信息系统面临着越来越多的安全威胁,信息安全问题日益突出。研究表明,相当数量的安全问题都是由于软件自身的安全漏洞引起的[1]。软件漏洞的产生,既有内因也有外因。内因方面:设计人员在架构阶段没有明确用户对安全性方面的需求,在设计上也没有考虑安全性,因此,使得软件本身存在着缺陷和Bug。外因方面,软件的运行环境不安全,容易被植入恶意代码,遭到黑客攻击,利用软件漏洞传播已成为时下恶意代码最为常用的手段之一。

1 软件安全漏洞检测方法

1.1 手工分析

手工分析是目前大部分安全研究人员仍在采用的方法。针对开源软件,手工分析人员一般是通过诸如SourceInsight之类的源码阅读工具来加速源码检索和查询的速度。比如,对C或C++程序最简单的分析一般都是先对系统中gets、strcpy之类不安全的函数调用进行审查,进一步的审核安全库函数和循环的使用。闭源软件的审查与开源软件不同,闭源软件难以获得源代码,因此反汇编引擎和调试器扮演了最重要的角色,通常在反汇编得到的汇编代码基础上进行分析,其难度要远远高于源代码阅读。由此造成对程序的理解和对程序的逆向工程等很多困难。

因此,不论采用什么方法进行手工分析,都要求安全分析人员既对软件安全漏洞的原理有深入的了解,同时还要熟悉软件本身的结构和功能。即便软件开发人员懂得软件安全漏洞检测技术,手工进行漏洞检测仍然是一件费时耗力的事情。但由于完全自动化的软件安全漏洞检测还没有实现,所以,人工的参与是必不可少的。比如,对静态程序分析结果的确认、动态程序分析数据的构造等。

1.2 动态测试

软件动态测试是通过运行软件来检验软件的动态行为和运行结果的正确性。目前,动态测试也是软件测

* 基金项目:国家863计划(2007AA010302)

试工作的主要方式。

动态测试需要通过程序的执行来完成，有别于静态测试得到程序每次执行都不变的特性。动态分析得到程序一次或多次执行的信息，根据这些信息对特定的漏洞模式进行检测，从而完成软件的安全分析。程序测试和剖析是最常见的标准动态测试，因为动态测试没有对程序进行抽象处理，所以其分析结果是十分准确的，如：程序的哪条路径被执行，程序计算的数据值是什么，程序使用了多少内存或是程序执行了多长时间等等。

但动态测试的结果是不完整的，一次程序的执行情况并不能代表程序以后的可能执行情况。也就是说，一个测试输入数据集不可能保证程序的所有可能执行路径，仅仅依靠一次或多次的程序执行情况有可能无法发现软件的安全漏洞，而这些安全漏洞却是真实存在的，关键是如何设计有良好分支覆盖和状态覆盖的测试。

1.3 静态测试

静态测试作为一种高效的程序分析方法，受到越来越多的重视。当用户给出语言的抽象语义后，该类方法能够自动发现满足所有可能执行状态的软件属性。静态测试方法具有自动化程度高、分析速度快的优点。在实际使用中，静态测试比动态测试更有效率，并能快速找到缺陷（发现30%~70%的逻辑设计和编码缺陷）。尽管静态分析方法可能产生一定的漏报(false negatives)或误报(false positives)，但仍然是当今最实用、最有效的安全漏洞检测方法之一[2]。

静态测试使用静态分析技术，直接分析程序的源代码，通过词法分析、语法分析和静态语义分析，检测程序中潜在的安全漏洞。目前，静态分析主要有类型推断、数据流分析和约束分析3种方法。

(1)类型推断

程序语言的类型系统包括一种定义类型的机制，有关类型等价、类型相容和类型推理的规则。在将运算符作用于运算对象、执行赋值，或者把实际参数传递给子程序时，都存在着类型是否合适的问题。类型推断是一个处理过程，其目的是保证每个操作都是针对一组数目正确、类型合适的对象进行，以保证操作的有效性。

类型推断可以检查类型错误，选择合适的操作，根据情况确定必要的类型转换。类型推断方法具有简单、高效的特点，非常适合软件安全漏洞的快速检测。采用类型推断方法检测的安全漏洞主要有C程序中的格式化字符串漏洞、操作系统内核中的权限检查[3]，以及操作系统内核中不安全的指针使用[4]等。

(2)数据流分析

数据流分析是一项编译时使用的技术，它能从程序代码中收集程序的语义信息并通过代数的方法在编译时确定变量的定义和使用。数据流分析被用于解决编译优化、程序验证、调试、测试、并行、向量化和串行编程环境等问题。数据流分析是通过对变量构造定义实现的。

数据流分析在安全检测中有着广泛的用途。应用数据流分析技术，可以检测C程序中的数组越界漏洞[5]等多种程序中的安全漏洞。

(3)约束分析

约束分析方法将程序分析过程分为约束产生和约束求解2个阶段，前者利用约束产生规则建立变量类型或分析状态之间的约束系统，后者对这些约束系统进行求解。

约束系统可以分为等式约束、集合约束和混合约束3种形式。等式约束的约束项之间只存在相等关系。集合约束把每个程序变量看成一个值集，变量赋值被解释为集合表达式之间的包含关系。混合约束系统由部分等式约束和部分集合约束组成。

约束分析在安全检测中的应用也很广泛。比如，可以利用集合约束方法检测C程序中的缓冲区溢出漏洞。

(4)3种主要静态分析方法的比较

以上介绍的3种主要静态分析方法都是通过解释程序的抽象语义，建立程序属性的数学模型，再通过求解这个数学模型确定程序的属性。相比较而言，约束分析具有最强的检测能力和最慢的检测速度，适合进行软件的安全检测；数据流分析具有较强和较快的检测速度，适合需要考虑控制流信息，变量属性之间的操作简单的静态分析问题；类型推断则具有最弱的检测能力和最快的检测速度，适合检查属性域有限、与控制流无关的安全属性。

2 软件安全静态测试工具

静态分析技术通过发现源代码中的安全漏洞防止入侵攻击。静态分析程序时不需要执行所测试的程序，它扫描所测试程序的正文，并对程序的数据流和控制流进行分析，然后产生非常人性化的错误报告，告诉用户发生错误的类型、位置并提出改正的建议，帮助用户改进软件质量。

软件安全测试的工具种类很多，根据保护程序或者保护软件系统安全的方式和途径，对基于源码分析的软件安全测试工具进行了分类，主要分为：词汇语法分析类工具、约束分析类工具、扩展编译类工具、基于模型测试工具等。

2.1 词汇语法分析类工具

词汇语法分析类工具是安全测试工具中最简单的工具，通过程序的词法和语法分析，对源代码进行静态分析，通过模式匹配找出特定的函数中可能导致安全问题的缺陷。

词汇语法分析类工具使用简单、快速，易于实现，通常嵌入到程序编译器中，在软件开发的过程中发挥重要作用。但其局限性比较大，很难检测出比较复杂的安全问题，同时此类工具的误报率比较高。

词汇语法分析类工具主要有：ITS4、Flawfinder、RATS 和 PScan 等。

2.2 约束分析类工具

约束分析类工具通常利用解析树产生的约束(也可以利用代码中添加注释)，对源代码进行分析，检查出可能导致安全问题的缺陷。

相比于词汇语法类的分析工具，此类工具能够更好地分析函数的安全性，可以更全面地检查特定的一些安全漏洞，通过对解析树的分析，确定缺陷的位置。但因其对安全问题检查是根据约束完成的，所以检查缺陷范围比较有限。

约束分析类工具主要有：BOON、CQUAL、Splint、UNO 以及 ESC/JAVA。

2.3 扩展编译类工具

扩展编译类工具是通过程序员根据对安全规则的理解，把规则和注释写入到源程序中传播。同时对编译器进行扩展，并对编译后的程序进行分析。

扩展编译类的工具把特殊域合并到编译过程中，使应用的编写者不需要懂得内部的编译方式，减少了程序员对于编译的了解，可以进行系统级的规则检查；隐藏执行过程中不容易理解的规则；降低了误报率和测试的开销。但是这类工具不能检测多线程类的错误。

扩展编译类工具主要有：xg++、MC。

2.4 基于模型测试的工具

基于模型测试类工具一般是对程序构造出有限状态模型，再检查其是否存在违反特定安全规则的问题。

不同于前面工具的是，模型检查会为程序建模，检查的对象变成抽象化的模型。首先构建程序的有限状态模型，使用形式化方法表示需要检查的安全属性，设计分析算法检测模型中存在的缺陷。但是现存的模型检测工具一般针对有限状态模型进行分析，对于模型构建复杂的系统，测试增加了难度。模型的统一形式化表示也是这方面研究的重点。

模型检测类的工具有：SLAM、BLAST、Banera、MOPS、The Boop Toolkit、ESP 以及 FindingBugs 等工具。

对软件测试工具进行的对比如表 1 所示。

表 1 软件测试工具的对比

工具类型	优点	缺点或困难
词汇语法类工具	简单且易于实现，通常嵌入编译器中，在软件开发过程中使用	检测局限性比较大，局限在词法与语法级别，且误报率比较高
约束分析类工具	能够依据不同的规则对不同的系统进行分析，发现潜在错误	受到规则描述机制的局限，只能分析特定类型的错误
扩展编译类工具	减少了程序员对编译的了解，可以隐藏在执行过程中不易理解的规则，降低了误报率和测试开销	复杂系统的编译扩展需要一些 metal 的经验，需要考虑完全性的折衷方案，MC 原型的实现还有些笨拙
模型检测类工具	可以分析各类属性，更善于发现系统的复杂行为	分析成本昂贵，很难应用到大规模的实际应用中

随着由软件安全漏洞引起的信息安全问题日益突出，人们越来越重视软件安全测试的重要性。对源代码进行分析的静态测试是保证软件安全的一个重要方法，它与动态测试的检测方法互为补充，但不可互相替换。软件安全静态测试工具也已经在测试工具中占据一席之地，随着软件安全测试技术和工具的不断发展，已经不再只注重于测试安全方面的漏洞，而是综合了更多方面的测试[6]，在实际应用中将更加可靠实用。

参考文献

[1] 刘海燕，杨洪路，王崛.C 源代码静态安全检查技术[J].计算机工程，2004，30(2)：28-30.

[2] 夏一民，罗军，张民选.基于静态分析的安全漏洞检测技术研究[J].计算机科学，2006，33(10)：279-282.

[3] ZHANG Xiao Lan，EDWARDS A.Using CQUAL for static analysis of authorization hook[C].Usenix security symposium，USA，2002.

[4] JOHNSON R，WAGNER D.Finding user/kernel pointer bugs with type inference[C].Usenix security symposium，2004.

[5] XIE Yi Chen，CHOU A，ARCHER E D.Using symbolic pathsensitive analysis to detect memory access errors[C].ESEC/FSE'03，helsinki，finland，Sep 2003.

[6] 白哥乐，宫云战，杨朝红.基于源码分析的软件安全测试工具综述[C].第五届中国测试学术会议，2008.

(收稿日期：2009-02-13)