基于符号执行的测试数据生成方法的研究与设计

赵跃华 阚俊杰
(江苏大学计算机科学与通信工程学院 江苏 镇江 212013)

摘 要 软件漏洞的研究是信息安全的一个重要分支。漏洞挖掘的主要方法是通过精心构造测试数据输入程序来触发漏洞,由此可见如何生成测试数据是该技术的关键,也是成功挖掘漏洞的关键。在分析漏洞存在原理和触发条件的基础上,提出一种更为高效的测试数据生成方法。该方法通过不安全函数来定位漏洞的触发点,深度与宽度混合遍历来确定触发的路径,利用符号执行技术来确立漏洞触发的条件,最后再根据条件生成测试数据,使生成的数据不仅有更高针对性,并且还提高了代码的覆盖率,从而能提高漏洞挖掘的效率和准确性。实验结果表明该方法具有良好的效率和准确性。

关键词 测试数据生成 不安全函数 混合遍历 符号执行技术 漏洞挖掘

中图分类号 TP306 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2014.02.081

RESEARCH AND DESIGN OF SYMBOL EXECUTION-BASED TEST DATA GENERATION METHOD

Zhao Yuehua Kan Junjie
(*School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang 212013, Jiangsu, China*)

Abstract The research on software vulnerabilities is one of the most important branches in information security. The main vulnerability discovery method is to input the elaborately structured test data to the program to trigger the vulnerability. So one can see that how to generate the test data is the key of the technology and the key for successful vulnerability discovery. Based on the analysis of existence principle of the vulnerability and triggering condition, we present a kind of more efficient test data generation method. In this method, the trigger points of the vulnerabilities are located by unsafe functions, the trigger paths are determined by mixed traversal in depth and width, and the trigger conditions of the vulnerabilities are determined by symbols execution technology, at last the test data are generated on the basis of these conditions. The test data formed in this way become more targeted, and the coverage rate of the code is raised as well, consequently the efficiency and accuracy of software vulnerability discovery can be increased. Experimental results show that the method has good efficiency and accuracy.

Keywords Test data generation Unsafe functions Mixed traversal Symbols execution technology Vulnerability discover

0 引 言

近年来,计算机病毒和各种恶意程序日趋泛滥,给计算机用户和网络带来很大的损失,也给信息安全带来巨大的挑战。作为计算机病毒的主要攻击手段和传播方式,软件安全漏洞成了计算机系统及网络信息安全中一个极大的安全隐患,也是网络信息安全中研究的核心问题之一。软件安全漏洞的存在,使得非法用户可以轻易绕过杀毒软件和防火墙的检测,对计算机系统实施攻击和入侵,获取某些系统权限,进而执行非法操作和进行恶意攻击。所以漏洞挖掘技术的研究日益受到重视。

1 研究现状

当前比较流行的漏洞挖掘的方法是 Fuzzing 测试,它是一种基于错误注入的黑盒随机测试技术,该技术通过向目标程序输入精心构造的测试集,分析程序的执行结果及检测程序状态,发现目标程序中隐藏的各种安全漏洞[1]。由此可见是否能高效地生成测试用例将直接影响到 Fuzzing 测试的结果。

现阶段已经有人关注到测试数据的生成的重要性,并将很多方法应用到测试数据的生成上。例如文献[2]根据文件格式结构化存储的特征,给出一种启发式的畸形数据的构造方法。文献[3]提出了一种基于遗传算法的测试数据生成方法,每个个体代表一个输入测试数据,对当前空间中的个体通过适应度函数计算出其适应度,依据“优胜劣汰”的思想选择适应度较高的个体进行交叉、变异,生成下一代个体,即输入测试集,依此类推进行繁殖,直到程序执行到目标路径。文献[4]和文献[5]是基于库函数的测试数据生成方法,根据漏洞产生的原因和重现的方法对漏洞库进行整理分类,然后动态跟踪目标程序对不安全库函数的调用,并在输入数据中搜索匹配函数调用参数,以此来准确定位错误注入点。但是文献[2]的方法,当文件格式相当复杂时,详尽掌握文件格式及语法的构造是比较困难的。其他方法考虑了如何约束生成测试数据,但是没有考虑到所生成

收稿日期:2012-12-15。**赵跃华**,教授,主研领域:信息安全。**阚俊杰**,硕士生。

的测试数据是否能够到达错误注入点,即生成的测试数据的针对性不足,存在较多的无用数据,从而使挖掘漏洞的效率较低,同时代码覆盖率也比较低。

由此可见欲准确高效地进行漏洞挖掘,就需要增强测试数据的针对性。本文将在分析漏洞产生原理的基础上,综合运用不安全函数来定位漏洞的触发点,深度与宽度混合遍历来确定触发的路径,利用符号执行技术来避免二进制代码的细节纠缠,提出一种高效的测试数据的生成,从而可以有效地解决测试数据盲目性的问题,提高代码的覆盖率,使漏洞挖掘更加具有针对性。

2 相关技术分析

2.1 漏洞原理的研究

常见的漏洞主要有缓冲区溢出、格式化字符串、整数溢出三种漏洞。缓冲区溢出是最常见的漏洞之一,它是在对缓冲区进行读写时没有去检查缓冲区的长度。黑客可以通过精心构造的输入数据使程序能按其思路发生溢出,即能通过溢出来改变程序的执行流程,使程序转去执行其所构造的恶意代码。格式化字符串漏洞存在于 printf 类的格式化输出函数中,其主要原因是没有指定格式字符串参数而直接输出字符串内容时,会导致格式化字符串漏洞的发生,例 printf(buffer),如果 buffer 中的数据是"%p%p%p%p",就可能会输出内存信息;也可以通过"%n"向内存中写入指定的数据,若一些敏感数据被篡改,就有可能改变程序执行的方向。对整数溢出漏洞,如果运算的结果超出相应类型的整数所能表示的范围,便产生整数溢出,而这些运算的潜在溢出结果值都能被用于关键的操作中(如内存的分配、复制等)。

综合上述三种常见漏洞的原因是在于所引用的函数中存在着不确定性。

可以归纳出常见的不安全函数有 malloc()、calloc()、realloc()、memcpy()、strcnpy()、strncat()、printf()、fprintf()、vprintf()、vfprintf()、vsprintf()、strcat()、strcpy()、scanf()、fscanf()、sprintf()、vscanf()等等。

由此可以发现软件漏洞的存在是由于一些不安全函数的存在,或者是在使用这些函数时没有进行严格的检查,所以可认为不安全函数是漏洞的触发点。准确定位出不安全函数是生成有针对性的测试数据的前提,也是漏洞挖掘的前提。

2.2 符号技术的研究

符号化执行技术主要思想是屏蔽程序细节,使用符号值代替具体的数据作为程序输入,在执行程序的过程中以符号计算代替普通执行中的数值计算,并用符号表达式对变量进行表示,程序执行的最终输出结果为由输入符号值组成的表达式。

符号执行技术已经被应用在源码检测上,而且效果很好,其中具有代表性的漏洞检测工具有 EXE[7] 和 CUTE[8]。本文要将其应用到二进制代码上,需要对其进行抽象。输入数据被符号化成输入变量,通过动态二进制分析监控、跟踪目标程序的执行,将程序执行路径上的数据流信息和控制流信息通过路径约束进行表示,然后对当前执行路径上的约束条件进行约束求解,动态构造生成能够引导程序执行新路径的输入测试数据,这样通过约束条件生成的数据是针对每条对应的路径,这种针对危险路径的测试数据可以有效地减少 Fuzzing 测试的范围,而且避免生成无用的测试数据,从而可提高漏洞挖掘的针对性。

系统输出了某一执行路径下的所有对输入变量的约束,每条约束都代表了一个节点处的条件跳转满足的条件。这一系列约束是一个逐渐细化的过程。由于每个条件都代表了一个节点处的条件,那么给任意一个条件取反,在对应的节点处就会执行另一个分支。通过这种方法,只要设计合适的遍历算法,就可以提高路径的覆盖率。

3 系统框架

由前面对漏洞原理和符号执行技术的研究,可以得出本文生成测试数据的方法,本文将应用于源码检测的符号执行技术应用到二进制代码中。首先通过不安全函数来定位漏洞的出发点,然后再利用路径覆盖技术来确定从程序输入点到漏洞触发点的路径,之后针对得到的路径通过符号执行技术来获取约束条件。具体方法是将可执行程序反汇编后,确立基本块的调用流程,在基本块的基础上结合不安全函数库来定位不安全函数,然后通过路径覆盖的方法找出程序输入点到不安全函数的路径,本文采用的路径算法是宽度优先结合深度优先的算法,之后针对这些路径进行符号执行,同时记录执行过程中的实时信息,将符号执行的结果求解,得到约束条件,并通过该约束条件生成测试数据。总框架见图 1 所示。

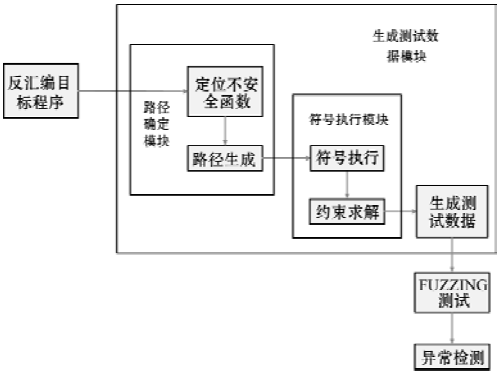

图 1 原型系统的总体框架

4 关键技术及实现

4.1 不安全函数的定位

不安全函数在二进制代码中的表现形式有 2 种,标准库函数中不安全函数的普通形式和展开形式。普通形式的不安全函数有独立的代码,对它们的引用通过 Call 指令完成的;而不安全函数展开形式则是编译器为了避免函数调用带来的开销和加快程序运行速度,会在程序中每一调用该函数的代码处插入实现该函数功能指令序列。

不安全函数定位就可根据它们的表现形式来实现的。本文是在 IDA pro 对二进制代码进行反汇编的基础上进行的。对于普通形式,在 IDA 中识别的方法很简单。因为 IDA pro 对库函数有很好的支持,能识别出二进制代码中对库函数的引用,并能够在反汇编后通过 call 指令调用其库函数名。因此只要维护一张不安全函数列表,然后检索 IDA pro 的反汇编结果,就能够确定二进制代码中调用了哪些不安全函数,同时还能确认对应不安全函数的调用信息,如调用的地址、调用的基本块,然后通过地址就能定位该不安全函数。具体处理流程如图 2 所示。

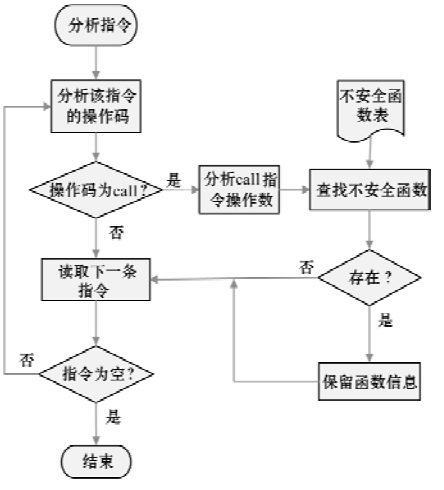

图 2　不安全函数普通形式的查找流程图

对于展开形式,通过分析 strcpy 函数的反汇编代码,可发现不安全函数的展开形式虽然具有不固定的形式,但仍具有显著的共同特征。该特征是实现某功能必然出现的指令及其顺序。所以只要在不安全函数展开形式表中添加不安全函数的固定顺序的指令,然后通过算法在二进制代码中来识别之。具体算法是顺序读取代码中每条指令,当找到第一条指令后,在两条关键指令可能间隔的最大范围内依次读取指令,进行第二条关键指令的匹配,若匹配成功则依此继续匹配下一条关键指令,如此进行下去,直到所有指定的关键指令都匹配成功为止。如果在此过程中有一条关键指令无法匹配,则此次识别失败。

4.2　路径算法的分析

路径生成模块先通过对基本块覆盖的方法查找从不安全函数到程序输入点的所有路径,所得的路径为后续符号执行模块而服务。路径生成算法直接影响到本文测试方法的效率和开销。传统的路径搜索算法是基于深度优先的算法,该算法每次选择的路径都是与当前路径有最长前缀的路径,如图 3 遍历得到的第 1 条路径为 1 -2 -4 -7 -9,第 2 条 1 -2 -4 -7 -10,这两条路径就存在着最长路径前缀 1 -2 -4 -7,由此可见该两条路径覆盖的结点只有 1、2、4、7、9、10,这样就会影响路径的覆盖率。而且采用这种方法,如果控制流图的左子树存在路径爆炸的问题,那么它的右子树就不会被访问到。

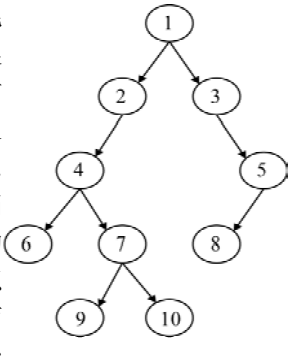

图 3　深度优先示例图

本文将采用一种深度和宽度混合优先遍历的算法。该方法可以很好地与符号执行相呼应,每个父节点右子树的约束可以直接通过取反其左子树的约束而得到,这样可以减少时间的开销。该算法总是试图使访问的路径与当前路径有最短的路径前缀,这样访问的路径的相似性将会减少,扩散性将会增加,从而提高路径的覆盖率。并且在具体处理中保存路径前缀,在测试其它不安全函数时,当确定该函数所在的基本块后,便可以通过直接的路径前缀匹配来获得从程序输入点到该基本块父结点的路径,然后进入当前基本块并找到不安全函数,生成路径。符号执行模块中将会保存每个父亲结点的约束条件,从而为其他不安全函数的漏洞挖掘降低工作量。

具体算法操作:

(1) 首先对反汇编后的基本块流程图进行宽度优先遍历,将结点按遍历次序入队列,这里的每个结点都是一个基本块。

(2) 如果队列为空,结束;否则,每次考察队列头部的结点:

① 如果其后继结点都已经被访问过,则将其出队。

② 否则,将其作为路径前缀的最后一个结点,从图起始结点选取一条到该结点的可执行路径作为当前正在生成的路径的前缀。

(3) 从选取的队列头部结点开始搜索后继结点,优先选取没有被访问过的结点,直到到达结束结点,生成一条路径。

(4) 重复步骤(2)。

该算法与基于深度优先搜索的路径调度算法相比,在生成路径之前增加了一个将该程序的控制流图(CFG)进行宽度优先搜索,并将结点按遍历次序入对的操作,复杂性为 $O(n)$,n 为图 CFG 的结点数。每选取一条路径的时间复杂度为 $O(d)$,d 为图的半径,即最大路径长度。每次产生的路径,或者前缀不同,当前缀相同时,其后继的第一个结点肯定不同,保证了每次返回的路径都是各不相同的。

4.3　符号执行模块

符号执行模块是通过对路径生成模块得到的路径进行符号执行,将约束条件交给约束求解器求解。符号执行技术能够引导程序执行不同的路径,有利于减少输入测试数据生成的盲目性,提高测试过程中的代码覆盖率,同时能够在分析过程中获取程序运行时的信息,使其分析过程更加准确,所收集到的路径约束既是遵循程序本身语义的也是精确的。

符号执行技术原先用于源码分析,现在将之运用到二进制代码上,需要首先定义一些变量,这些变量是程序输入的数据,然后建立映射关系,同时记录每个跳转点对变量的约束条件,并为原始变量和中间变量建立一方程组,最后求解该方程组得到约束条件。本模块首先对机器指令的解析,然后再进行符号执行的分析与路径约束条件的提取工作,同时进行程序的实际执行,具体流程见图 4 所示。

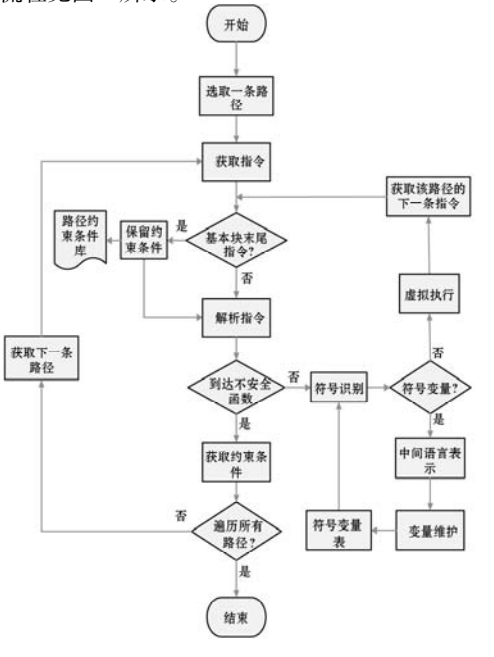

图 4　符号执行模块的流程图

在符号执行的过程中,保留从程序输入点到各个基本块的路径约束条件,以利在进行其他不安全函数的漏洞挖掘时,可以优先查找该路径约束库,如果查找成功便会节省后续的符号执行过程,从而提高执行效率;如果没有查到当前基本块的约束,则可以查找出该基本块的父基本块,然后再进行符号执行,同样可以提高执行效率的。

5 实验与分析

本文采用纯随机的、基于库函数的及本文的方法进行对比实验,利用它们所生成的测试数据进行 Fuzzing 测试。测试环境为 LenovoPC 机 - Intel Core2 Duo CPU E7500,2.93GHz,2GB;Windows XP sp3,安装了 IDA、Windbg 等辅助软件。测试对象是 FoxPlayer 播放器,该软件存在 12 个漏洞[10]。

纯随机的 Fuzzing 测试是给输入数据进行逐字节 00 - ff 的替换。基于库函数的 Fuzzing 测试方法是利用开源 Fuzzer 工具 SPIKE[9],采用常见易引发漏洞的特殊字段对输入数据逐字节进行替换。测试中记录生成的测试用例数、覆盖的基本块数、测试的不安全函数数目、测试的时间、发现的漏洞数,得到的结果如表 1 所示。

表 1 测试结果对比表

测试方法	生成测试用例数	覆盖基本块数	测试函数数	测试时间(h)	发现漏洞数
纯随机的 Fuzzing	13 264	3 862	137	3.5	3
基于库函数的 Fuzzing	4 213	6 625	513	3.5	7
本测试方法	782	9 351	770	0.5	12

由表 1 可以看出纯随机的 Fuzzing 效率较差,生成了较多的测试用例,而发现的漏洞数仅有 3 个;基于库函数的 Fuzzing 在同样的时间内生成的测试用例是前者的 31.8%,但覆盖的基本块数却比前者增加了 71.2%,发现了 7 个漏洞,比前者提高了 133.3%。本文方法产生的测试用例数仅是基于库函数方法的 18.6%,但覆盖的基本块数却比它增加了 41.1%,并且该方法测试的函数数也增加了 50.1%,并发现了全部的 12 个已知漏洞。实验结果表明,本方法生成的测试用例数更少并增强了漏洞挖掘的针对性,同时在测试时间上也大幅度减少,测试的效率明显提高了,而且准确率也可以保证。

6 结 语

如何生成测试数据将直接影响到漏洞挖掘的准确性和效率。本文将应用在源码检测的符号执行技术应用到二进制代码上,结合不安全函数定位技术、路径覆盖技术来辅助测试用例的生成。该方法解决测试数据生成盲目性的问题,有效提高漏洞挖掘的针对性,并且在每次触发漏洞后就能生成一个对应的测试用例,使漏洞挖掘的效率进一步提高。实验表明该方法是有效的。下一步的工作是简化约束求解的复杂性;加强符号执行时对数据、变量的实时监控。

参 考 文 献

[1] Oehlert P. Violating Assumptions withFuzzing[J]. IEEE Security &Privacy,2005,3(2):58 - 62.

[2] 唐彰国,钟明全. 基于 Fuzzing 的文件格式漏洞挖掘技术[J]. 计算机工程,2010,36(16):151 - 153,160.

[3] Sparks S, Embleton S, Cunningham R, et al. Automated Vulnerability Analysis: Leveraging Control Flow for Evolutionary Input Crafting [C]//Computer Security Applications Conference, 2007:477 - 486.

[4] 黄奕,曾凡平,曹青. 基于库函数动态跟踪的 Fuzzing 测试方法[J]. 计算机工程,2010,36(16):39 - 41.

[5] 张美超,曾凡平. 基于漏洞库的 fuzzing 测试技术[J]. 小型微型计算机系统,2011,32(4):651 - 655.

[6] King J C. Symbolic Execution and Program Testing[J]. Journal of the ACM,1976,19(7):385 - 394.

[7] Cadar C, Ganesh V, Pawlowski P M, et al. EXE:Automatically Generating Inputs of Death[C]//CCS'06:Proceedings of the 13th ACM conference on Computer and communications security, New York, USA,2006:322 - 335.

[8] Sen K, Marinov D, Agha G. Cute:a concolic unit testing engine for c [C]//ESEC/FSE-13: Proceedings of the 10th European software engineering conference held jointly with 13th ACMSIGSOFT international symposium on Foundations of software engineeringNew York, USA, 2005:263 - 272.

[9] Spike[EB/OL]. 2006 - 07 - 03. http://lists. immunitysec. com/mailman/listinfo/spike.

[10] 杨俊. 基于函数摘要的二进制漏洞挖掘技术的研究[D]. 安徽:中国科学技术大学,2011.

(上接第 265 页)

大的商业机会。本文通过向量自回归条件异方差模型,设计实现了社会网络的动态演化模型。此方法为企业了解市场、制定正确的营销决策提供信息支持,不仅能更准确地挖掘出用户的观点及购买行为,而且能更高效地指导生产商和服务商改进产品、改善服务、提高竞争力。

参 考 文 献

[1] 李实,叶强,李一军,等. 挖掘中文网络客户评论的产品特征及情感倾向[J]. 计算机应用研究,2010,27(8):3016 - 3019.

[2] Cohen J, Dolan B, Dunlap M, et al. MAD Skills:New Analysis Practices for Big Data[J]. PVLDB,2009,2(2):1481 - 1492.

[3] 叶强,张紫琼,罗振雄. 面向互联网评论情感分析的中文主观性自动判别方法研究[J]. 信息系统学报,2007,1(1):79 - 91.

[4] Backstrom L, Leskovec J. Supervised Random Walks:Predicting and Recommending Links in Social Networks[C]//WSDM 2011:Proceedings of the 4th ACM International Conference on Web Search and Data Mining, Hong Kong, February 9 - 12,2011. USA:CRC Press,2011.

[5] Robert Engle. The Use of ARCH/GARCH Models in Applied Econometrics[J]. Economic Perspectives,2001,15(4):157 - 168.

[6] Bollerslev. Modeling the Coherence in Short-run Nominal Exchange Rates:A Multivariate Generalized ARCH Model[R]. The Review of Economics and Statistics,1990,72(3):498 - 505.

[7] 朱亮,韩定定. 动态复杂网络的同步拓扑演化[J]. 计算机应用,2012,32(2):330 - 334,339.

[8] 王甲生,吴晓平,廖巍,等. 改进的加权复杂网络节点重要度评估方法[J]. 计算机工程,2012,38(10):74 - 76.