

南京邮电大学

毕业设计（论文）

题 目 定向覆盖模糊测试工具的设计与实现

专 业 计算机科学与技术

学生姓名 雷尚远

班级学号 B190303 B19030334

指导老师 王子元

指导单位 计算机学院、软件学院、网络空间安全学院

日期： 2023 年 3 月 x 日至 2023 年 6 月 x 日

毕业设计（论文）原创性声明

本人郑重声明：所提交的毕业设计（论文），是本人在导师指导下，独立进行研究工作所取得的成果。除文中已注明引用的内容外，本毕业设计（论文）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本研究做出过重要贡献的个人和集体，均已在文中以明确方式标明并表示了谢意。

论文作者签名：

日期： 年 月 日

摘 要

模糊测试（Fuzzing）是一种通过向目标系统提供非预期的输入并监视异常结果来发现软件安全漏洞的方法，是软件安全领域常用的方法之一。由于代码覆盖率与漏洞覆盖率密切相关，大多数模糊测试工具都是以代码覆盖率为导向。然而，由于大多数被覆盖测试的代码可能并不包含漏洞，这使得盲目地扩展代码覆盖率的方式在实际测试时效率较低。极端情况尤为如此。与盲目增加代码覆盖率的模糊测试不同，定向覆盖的灰盒模糊测试（DGF）将大部分时间用于检测特定目标区域（例如，易出错代码段）而不会浪费资源于不相关的部分。因此，DGF 特别适用于补丁测试、漏洞复现以及特殊漏洞检测等场景。目前，DGF 已成为一个快速发展的研究方向。基于一些先进的定向覆盖模糊测试工具的研究和相关调查，本文主要做了以下点工作：

- (1) 基于现有的模糊测试工具框架 AFL（American Fuzzy Lop）以及 AFLGo 做了定向覆盖策略的设计和集成；
- (2) 实现了简单的定向覆盖的模糊测试命令行工具；
- (3) 针对相应的公开通用漏洞集（CVE）做了复现及定向实验对比测试。

此外本文亦通过分析工具设计以及实现过程中的局限性与不足，对于未来该方向的研究发展做出了一些展望。

关键词： 模糊测试；定向覆盖模糊测试；灰盒测试；软件安全

ABSTRACT

Fuzzing is a method of discovering software security vulnerabilities by providing unexpected inputs to a target system and monitoring for abnormal results. It is one of the commonly used methods in the field of software security. As code coverage is closely related to vulnerability coverage, most fuzz testing tools are guided by code coverage. However, blindly extending code coverage may be inefficient in practical testing since most of the covered code may not contain vulnerabilities, especially for corner cases. In contrast to blind code coverage-based fuzz testing, directed grey-box fuzzing (DGF) spends most of its time detecting specific target regions (such as error-prone code segments) rather than wasting resources on irrelevant parts. Thus, DGF is particularly suitable for scenarios such as patch testing, bug reproduction, and special bug detection. For now, DGF has become a fast-growing research area. Based on some advanced directed coverage fuzz testing tools and relevant investigations, this article mainly focuses on the following points of work:

- (1) Designed and integrated a directed coverage strategy based on the existing fuzzy testing tool framework AFL (American Fuzzy Lop) and AFLGo;
- (2) Implemented a simple command-line tool for directed fuzz testing;
- (3) conducted reproductions and directed experiments on corresponding public vulnerability databases (CVE) for comparative testing.

In addition, this article also provides some prospects for the future research and development of this direction by analyzing the limitations and deficiencies in the design and implementation process of the tool.

Keywords: Fuzzing; Directed Greybox Fuzzing; Greybox test; Software Security

目 录

第一章 绪论.....	1
1.1 背景分析.....	1
1.2 国内外研究现状.....	1
1.3 研究内容.....	3
1.4 论文结构.....	3
第二章 相关技术研究.....	4
2.1 模糊测试技术.....	4
2.1.1 黑盒模糊测试技术.....	4
2.1.2 白盒模糊测试技术.....	4
2.1.3 灰盒模糊测试技术.....	4
2.2 定向模糊测试技术.....	4
2.2.1 白盒定向模糊测试技术.....	4
2.2.2 灰盒定向模糊测试技术.....	4
2.3 研究动机.....	4
2.4 本章小结.....	4
第三章 定向模糊测试策略设计.....	5
3.1 AFLGo 架构研究.....	5
3.1.1 距离计算机制.....	5
3.1.2 能量调度机制.....	5
3.2 定向适应度指标.....	5
3.2.1 距离定义.....	5
3.2.2 目标函数集覆盖率.....	5
3.2.3 能量调度机制.....	5
3.3 本章小结.....	5
第四章 基于 AFLGo 的定向模糊测试系统的实现.....	6
4.1 需求分析.....	6
4.2 架构设计.....	6
4.3 静态分析器的改进.....	6
4.4 定向模糊测试工具.....	6
4.5 本章小结.....	6
第五章 系统测试.....	7
5.1 系统测试概述.....	7
5.1.1 系统测试目标.....	7
5.1.2 系统测试环境.....	7
5.2 功能测试.....	7

5.3 实验评估	7
5.3.1 定向模糊测试工具	7
5.4 本章小结	7
第六章 总结与展望	8
6.1 总结	8
6.2 展望	8
结束语	9
致谢	10
参考文献	11

第一章 绪论

1.1 背景分析

“常用系统中可能会潜伏着严重的漏洞^[1]。”这一论述源自于模糊测试首次面世的论文。其揭示了一个事实，即随着软件技术的不断发展，软件安全问题就将日益成为愈发重视的议题。在当前的信息化时代，软件已经成为了人们生活、工作和娱乐的重要组成部分，这也意味着我们将面临着越来越多的安全威胁。因此，确保软件安全已经成为了一项非常重要的任务。

软件安全（Software Security）就是使软件在受到恶意攻击的情形下依然能够继续正确运行及确保软件被在授权范围内合法使用的思想。在当今社会，软件越来越普及，并被广泛应用于各个领域，包括电商、金融、医疗等。但是，由于软件的复杂性和开发过程中的缺陷，软件本身也存在着各种安全问题。这些问题可能导致信息泄露、数据损坏、远程攻击等，对个人、企业甚至整个社会造成巨大的损失。因此，保障软件安全显得尤为重要。

近年来，因为软件漏洞造成的损失案例屡见不鲜。2017 年，全球范围内爆发了 WannaCry 勒索病毒攻击事件，该攻击利用了微软 Windows 操作系统中的漏洞，并导致了数十亿美元的经济损失；2019 年，美国资讯技术服务公司 SolarWinds 遭受了一次大规模的网络攻击，该攻击利用了 SolarWinds Orion 平台软件中的漏洞，影响了包括美国联邦政府在内的许多组织和机构；2021 年 2 月，法国 LCL 银行的客户登录自己的银行应用程序时，看到的是别人的银行账户信息。原因是由于备份超级计算机系统（日本惠普公司制造）的程序存在缺陷，超级计算机系统出现了意外，其中存储（/LARGE0）中的某些数据被误删除；2021 年 12 月，知名日志框架 Log4j2 被爆出远程代码执行漏洞，影响了大量使用该框架的中间件和应用，给企业和用户带来了巨大的安全风险。

以上诸多例子可以说明，大多数的安全事件都是攻击者利用软件系统中的漏洞从而进行攻击引发的。因而可以帮助发现和修复安全漏洞的软件测试技术（Software testing）一直以来都是软件安全领域的一个重要议题。

软件测试可以通过模拟攻击者的行为来发现这些安全漏洞，并提供关于如何修复这些漏洞的信息。例如，黑盒测试可以探测应用程序中的安全问题，白盒测试可以评估应用程序的源代码中是否有漏洞，静态分析可以扫描源代码以发现潜在的安全问题，动态分析可以模拟攻击场景并检查应用程序的反应。

此外，软件测试还可以帮助确保应用程序在面对各种攻击时具有足够的鲁棒性和可靠性。它可以测试应用程序的身份验证和授权机制、加密技术、网络协议、输入输出数据验证等方面的功能，以确保应用程序满足安全需求。

1.2 国内外研究现状

软件安全信息系统和软件安全代码的有效安全项目往往依靠两种自动的安全测试：静态安全扫描测试和动态安全扫描测试。

在软件的开发期间，为了保证软件的安全性，通常会进行软件安全静态扫描。

这个过程是通过威胁建模和分析来完成的，其目的是对静态代码进行全面地扫描，以便及早地发现任何可能存在的安全漏洞。其是在不运行程序的情况下对软件进行测试和评估。静态分析可以检查代码、设计和文档等，以发现潜在的问题和错误，并确保软件符合某些标准或规范。由于本文主要探讨针对代码的漏洞审查，关于软件工程部分的文档、标准以及接口设计的测试技术在此不再赘述。利用数据流分析，符号执行以及污点分析等静态软件分析技术可以检查源代码中的错误和缺陷，包括语法错误、类型错误、内存泄漏、空指针引用等。与传统的动态测试相比，静态扫描可以更早地发现安全问题，因为它可以在代码尚未被编译或执行之前就进行检测。此外，静态扫描还可以减少测试成本，提高测试效率，并帮助开发团队更好地理解代码中的潜在安全风险。

软件安全动态扫描是一种对工作环境中实际运行的代码进行扫描的技术，它能够在代码运行时检测和分析可能存在的漏洞、缺陷和错误。与静态代码分析不同，动态扫描具有更强的准确性和实时性，因为它是在真实的环境中对代码进行测试和评估。通过使用动态扫描技术，开发人员和安全专家可以有效地识别并修复潜在的安全漏洞，从而保护软件系统免受攻击和破坏。此外，动态扫描还可以帮助企业遵循各种合规性标准和法规要求，确保其软件应用程序的安全性和稳定性。

模糊测试技术（Fuzzing Test）是动态安全扫描测试中重要的一种方式。而自从 1988 年模糊测试这一概念被提出后，这一方法一直在软件安全测试领域保持着较高的活跃度和关注度。在提出伊始，其主要用于测试操作系统。之后，随着软件技术的发展，模糊测试技术不断得到改进和推广，并应用于网络、移动设备等领域。目前，模糊测试技术已成为一种成熟的自动化测试技术，可以有效地检测软件中存在的漏洞和安全隐患。并且迄今为止其社区依然十分活跃，在 GitHub 上有超过 1000 个与模糊测试相关的仓库^[2]。为了防止被恶意攻击，许多商业软件公司，例如 Adobe, Cisco, Google, 和 Microsoft 都将模糊测试作为其雇员软件开发安全测试的必要环节。可以说，模糊测试是软件安全领域中一个经久不衰的热门议题。

而定向模糊测试（Directed Fuzzing）作为模糊测试的一个研究方向，主要关注重点区域（例如，易出错区域）并且将大部分的时间用于到达测试这些位置而不浪费资源在无关部分^[3]。从本源上讲，定向模糊测试工具早期的解决思路主要是基于利用程序分析和约束求解来生成检测不同程序执行路径输入符号的执行技术^[4-9]。然而，由于定向符号执行技术（DSE）依赖于大量的程序分析和约束求解，其受限于兼容性和可扩展性的限制。

在 2017 年，Böhme 等人引入了定向灰盒模糊测试（DGF）的概念^[10]。这是定向模糊测试的又一重要工作。其开创式地将位置的可达性问题转化为生成种子和其目标集之间距离的最小值问题。通过给更靠近目标集的种子更多的变异机会，它可以逐渐引导灰盒测试接近程序目标位置。与定向符号执行技术相比，DGF 有更好的可拓展性，并且在测试效率上有几个数量级的提升。例如，Böhme 等人可以在 20 分钟内重现 Heartbleed^[11]（CVE-2014-0160）漏洞，而定向符号执行工具 KATCH^[12]需要 24 小时以上^[10]。

此后，许多针对于

1.3 研究内容

1.4 论文结构

第二章 相关技术研究

2.1 模糊测试技术

2.1.1 黑盒模糊测试技术

2.1.2 白盒模糊测试技术

2.1.3 灰盒模糊测试技术

2.2 定向模糊测试技术

2.2.1 白盒定向模糊测试技术

2.2.2 灰盒定向模糊测试技术

2.3 研究动机

2.4 本章小结

第三章 定向模糊测试策略设计

3.1 AFLGo 架构研究

3.1.1 距离计算机制

3.1.2 能量调度机制

3.2 定向适应度指标

3.2.1 距离定义

3.2.2 目标函数集覆盖率

3.2.3 能量调度机制

3.3 本章小结

第四章 基于 AFLGo 的定向模糊测试系统的实现

4.1 需求分析

4.2 架构设计

4.3 静态分析器的改进

4.4 定向模糊测试工具

4.5 本章小结

第五章 系统测试

5.1 系统测试概述

5.1.1 系统测试目标

5.1.2 系统测试环境

5.2 功能测试

5.3 实验评估

5.3.1 定向模糊测试工具

5.4 本章小结



第六章 总结与展望

6.1 总结

6.2 展望

结束语

致 谢

本论文采用 \LaTeX 模版编写的，是基于南京邮电大学 2021 年理工艺教类的 Word 模板进行严格迁移编写的。本模板地址<https://github.com/dhiyu/NJUPT-Bachelor>感谢imguozr(<https://github.com/imguozr/NJUPThesis-Bachelor>)和lemoxiao(<https://github.com/lemoxiao/NJUPThesis-Scholar>)的工作，为本模板的形成奠定了大量的基础。

参考文献

- [1] Miller B P, Fredriksen L, So B. An empirical study of the reliability of unix utilities[J]. Communications of the ACM, 1990, 33(12): 32-44.
- [2] Manès V J, Han H, Han C, et al. The art, science, and engineering of fuzzing: A survey[J]. IEEE Transactions on Software Engineering, 2019, 47(11): 2312-2331.
- [3] Wang P, Zhou X, Lu K, et al. The progress, challenges, and perspectives of directed greybox fuzzing: arXiv:2005.11907[EB/OL]. arXiv, 2022.
- [4] Ganesh V, Leek T, Rinard M. Taint-based directed whitebox fuzzing[C]//2009 IEEE 31st International Conference on Software Engineering. IEEE, 2009: 474-484.
- [5] Ma K K, Yit Phang K, Foster J S, et al. Directed symbolic execution[C]//Static Analysis: 18th International Symposium, SAS 2011, Venice, Italy, September 14-16, 2011. Proceedings 18. Springer, 2011: 95-111.
- [6] Person S, Yang G, Rungta N, et al. Directed incremental symbolic execution[J]. Acm Sigplan Notices, 2011, 46(6): 504-515.
- [7] Do T, Fong A C M, Pears R. Dynamic symbolic execution guided by data dependency analysis for high structural coverage[C]//Evaluation of Novel Approaches to Software Engineering: 7th International Conference, ENASE 2012, Warsaw, Poland, June 29-30, 2012, Revised Selected Papers 7. Springer, 2013: 3-15.
- [8] Ge X, Taneja K, Xie T, et al. Dyta: dynamic symbolic execution guided with static verification results[C]//Proceedings of the 33rd International Conference on Software Engineering. 2011: 992-994.
- [9] Li H, Kim T, Bat-Erdene M, et al. Software vulnerability detection using backward trace analysis and symbolic execution[C]//2013 International Conference on Availability, Reliability and Security. IEEE, 2013: 446-454.
- [10] Böhme M, Pham V T, Nguyen M D, et al. Directed greybox fuzzing[C/OL]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas Texas USA: ACM, 2017: 2329-2344. DOI: 10.1145/3133956.3134020.
- [11] Heartbleed - a vulnerability in openssl[EB/OL]. [2023-05-13]. <https://heartbleed.com/>.
- [12] Marinescu P D, Cadar C. Katch: High-coverage testing of software patches[C]//Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. 2013: 235-245.