CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 44 卷第 12 期 2022 年 12 月
Vol. 44, No. 12, Dec. 2022

文章编号:1007-130X(2022)12-2173-14

模糊测试技术研究综述*

牛胜杰[1],李　鹏[2],张玉杰[1,2]

(1.南京邮电大学计算机学院,江苏 南京 210023;2.江苏省无线传感网高技术研究重点实验室,江苏 南京 210023)

摘　要:随着人们对软件系统安全问题关注度的不断提升,模糊测试作为一种用于安全漏洞检测的安全测试技术,具有自动化程度高、误报率低等优点,其应用越来越广泛,地位也越来越重要。经过近些年的不断改进,模糊测试无论在技术发展上还是在应用创新上,都取得了诸多成就。首先,对模糊测试的相关概念和基本理论进行简要说明,总结了模糊测试在各领域的应用情况,针对不同领域的漏洞挖掘需求,分析得出相应的模糊测试解决方案。其次,重点总结了近几年来模糊测试的重要发展成果,包括测试工具、框架、系统及方法的改进与创新,并分析总结了各发展成果所采用的创新方法,提出的理论以及各工具、系统的优点与不足。最后,分别从协议逆向工程应用、云平台建设、新兴技术结合、模糊测试对抗技术研究及模糊测试工具集成的角度,为模糊测试下一步的研究提供了方向参考。

关键词:模糊测试;漏洞挖掘;软件测试;协议测试

中图分类号:TP391.41　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2022.12.011

Survey on fuzzy testing technologies

NIU Sheng-jie[1], LI Peng[2], ZHANG Yu-jie[1,2]

(1. School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023;
2. Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing 210023, China)

Abstract: As people pay more and more attention to software system security issues, fuzzy testing, as a security testing technology for security vulnerability detection, has become more and more widely used and more and more important due to its high degree of automation and low false alarm rate. After continuous improvement in recent years, fuzzy testing has achieved many achievements in both technical development and application innovation. Firstly, we briefly explain the related concepts and basic theories of fuzzing, summarize the application of fuzzy testing in various fields, and analyze the corresponding fuzzy testing solutions according to the needs of vulnerability mining in different fields. Then, we focus on the important development results of fuzzy testing in recent years, including the improvement and innovation of testing tools, frameworks, systems, and methods. We also analyze and summarize the innovative methods and theories adopted by each development results, as well as the advantages and disadvantages of each tools and systems. Finally, from the perspectives of protocol reverse engineering application, cloud platform construction, emerging technology integration, fuzzy testing countermeasure technology research, and fuzzing tool integration, we provide direction reference for the further research of fuzzy testing.

Key words: fuzzy testing; vulnerability mining; software test; protocol test

* **收稿日期**:2021-08-17;**修回日期**:2022-03-18
基金项目:国家自然科学基金(61872196, 61872194, 61902196, 62102194, 62102196);江苏省科技支撑计划(BE2019740, BK20200753, 20KJB520001);江苏省高等学校自然科学研究重大项目(18KJA520008);江苏省六大人才高峰高层次人才项目(RJFW-111)
通信作者:李鹏(lipeng@njupt.edu.cn)
通信地址:210023 江苏省南京市南京邮电大学计算机学院
Address: School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, Jiangsu, P. R. China

1 引言

如今,恶意攻击者利用系统漏洞对软件系统进行攻击的现象屡见不鲜,已对企业和用户造成严重的影响。传统的漏洞检测方法(如静态分析方法和动态分析方法)具有操作流程繁琐、测试覆盖范围小的缺点,已不符合现阶段的需要,这就促使学术界必须提出漏洞挖掘的新技术、新工具和新系统。模糊测试是漏洞检测方法的一种,通过使用针对目标程序生成的随机字符流,对目标程序进行多次测试,以检测可能存在的漏洞。模糊测试技术具有不依赖目标程序源码、不受限于被测系统内部结构和可复用性强等优点,已成为漏洞挖掘领域的重点研究内容。近年来,研究人员通过对模糊测试技术的不断创新发展,例如采用漏洞挖掘新方法,提出漏洞挖掘新框架,创造漏洞挖掘新工具等方式,不断提高软件系统的漏洞挖掘技术水平。本文结合10年来在国际著名安全会议上发表的有关模糊测试文献,重点对模糊测试技术的发展及应用进行总结。

本文第2节简要介绍模糊测试基本概念及工作流程,关注对不同测试目标采用的模糊测试技术,对其进行技术分类并总结其优缺点。提供基于时间轴的模糊技术发展史图,选取重要的技术提出以及代表型模糊测试工具的创新问世为时间点,帮助读者直观地了解模糊测试技术的诞生与发展。第3节介绍模糊测试技术在各领域的应用。针对不同应用场景,模糊测试分别从不同维度提供技术支持,提高漏洞挖掘能力。第4节重点介绍近年来模糊测试技术的创新发展成果。针对不同领域产生的工具、框架及系统原型,分别对其采用的创新理论和具体流程进行梳理总结,并分析其优缺点。第5节结合协议逆向工程、云平台建设、新兴技术(如机器学习和大数据分析)应用、模糊测试对抗技术及模糊测试工具集成,提出模糊测试技术的挑战和发展方向。

2 模糊测试

2.1 模糊测试的定义与流程

模糊测试(Fuzzing)[1]是一种针对不同测试目标采用不同的技术手段,结合用例生成策略产生测试用例,将其输入到目标系统后通过监视异常结果来发现漏洞的安全检测技术。具体来说,模糊测试是利用插桩、符号执行及污点分析等技术收集目标信息,根据对目标内部信息的依赖程度采用不同的测试方法(如黑盒、白盒和灰盒),基于种子生成策略对正常数据进行自动或半自动地变异生成大量预期或非预期的测试用例,通过分析目标的输出结果进行安全漏洞检测的方法。

通过对大量文献的总结梳理,模糊测试的一般流程可分为以下环节:

(1)确定模糊测试对象。测试对象选择首先要考虑对象本身的因素,如目标程序或系统的性质、功能、运行环境和实现语言等。测试对象包括二进制代码或者软件系统,除了自身开发的程序或系统外,通常软件源代码不容易得到,因此模糊测试对象大多数为二进制代码。测试对象的宏观审视,是整个模糊测试的基础,它将直接影响模糊测试的技术选择(如黑盒、白盒和灰盒)。

(2)选择输入向量。测试对象的因素组成即输入向量包括文件数据、网络数据、注册表键、环境变量及其他信息等。恶意破坏者之所以能够利用系统安全漏洞,究其根本是由于系统没有对输入进行校验或非法输入处理。输入向量和测试用例生成策略可概括为测试用例参考要素,考虑各输入向量的影响权重,结合测试用例生成策略,找到具有高覆盖率的测试用例是模糊测试成功的关键。

(3)生成测试用例。测试用例的生成基于选定的输入向量,可以基于变异或生成方法生成大量测试用例。基于变异的测试用例生成方法是通过对已知数据样本进行定向(如bitflip按位翻转、interest替代、splice绞接等)或随机变异的方式产生新的测试用例,该方法利用正常输入中的信息绕过错误检查代码,达到可执行的目的,具有较高的扩展性。基于生成的测试用例生成方法是通过人工编写测试用例生成规则,使得测试用例可以按照目标规则进行生成,保证生成的测试用例可以绕过目标程序的错误检查代码。

(4)执行测试用例。将测试用例发送到目标软件或系统,以确保测试对象能够成功处理测试用例。

(5)监视器。测试用例执行完成后,需要对目标对象的产生结果加以监视。当目标对象崩溃或报告错误时,监视器模块将会收集并分析相关信息,记录产生异常的测试用例和产生的异常的详细信息,确定漏洞的真实性。

(6)有效性评估。分析异常产生的原因,跟踪

异常产生前后的处理流程,判断发现的漏洞是否可被利用。

模糊测试流程图如图 1 所示。

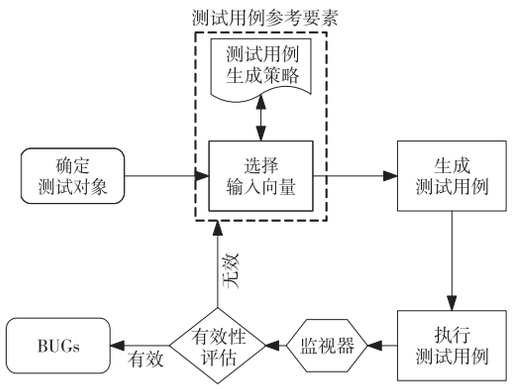

Figure 1 Flow chart of fuzzing

图 1 模糊测试流程图

2.2 模糊测试技术的分类

根据在模糊测试过程中所需的输入信息或对程序内部信息分析的程度,模糊测试技术可分为 3 大类:黑盒模糊测试、白盒模糊测试和灰盒模糊测试。

(1)黑盒模糊测试。黑盒模糊测试也称为输入输出驱动模糊测试或者功能模糊测试,其原理是把目标当做一个看不到内部逻辑结构的黑盒,在完全不考虑内部结构和性能的情况下,使用一些预定义的种子文件创建表单输出的模糊测试技术。在测试过程中,由于黑盒模糊测试无法跟踪目标内部的执行状态,只能通过检测目标的输出数据来判断目标的状态。测试过程中会生成大量冗余的测试用例,是导致该类技术的代码覆盖率低、测试效果差的主要原因。

(2)白盒模糊测试。白盒模糊测试也称为逻辑驱动模糊测试,与黑盒模糊测试截然不同,白盒模糊测试是将目标看作一个内部结构高度可视化的透明盒,在全面了解目标内部逻辑的基础上进行的模糊测试技术。白盒模糊测试是将目标内部结构单元化,并将单元测试的范围扩展到整个目标的安全测试。白盒模糊测试具有测试全覆盖的优势,但正是由于其高度可视化,在实际应用中,目标对象内部的复杂程度严重制约了它的发展。

(3)灰盒模糊测试。灰盒模糊测试是白盒模糊测试的一种变体,继承了黑盒模糊测试与白盒模糊测试的优点,同时对两者的缺点进行了改进。它是在对目标对象有部分了解的情况下进行的漏洞检测方式。与白盒模糊测试相比,两者都是利用目标程序的信息来减轻黑盒模糊测试的盲目性,但对目标信息的依赖程度不同。

各类模糊测试技术特点如表 1 所示。

Table 1 Classification of fuzzing technologies

表 1 模糊测试技术分类

类别	性质	优点	缺点
黑盒模糊测试	输入输出驱动	比较简单,不需要了解目标内部实现	代码覆盖率低、测试复用性低
白盒模糊测试	逻辑驱动、单元测试	大大提高了代码覆盖率	测试基于代码,可能会遗漏功能需求;当系统复杂时,开销巨大
灰盒模糊测试	白盒模糊测试的变体	代码覆盖率较高,测试开销适中	不适用于简单系统测试

2.3 模糊测试技术的发展简史

模糊测试技术发展史如图 2 所示。

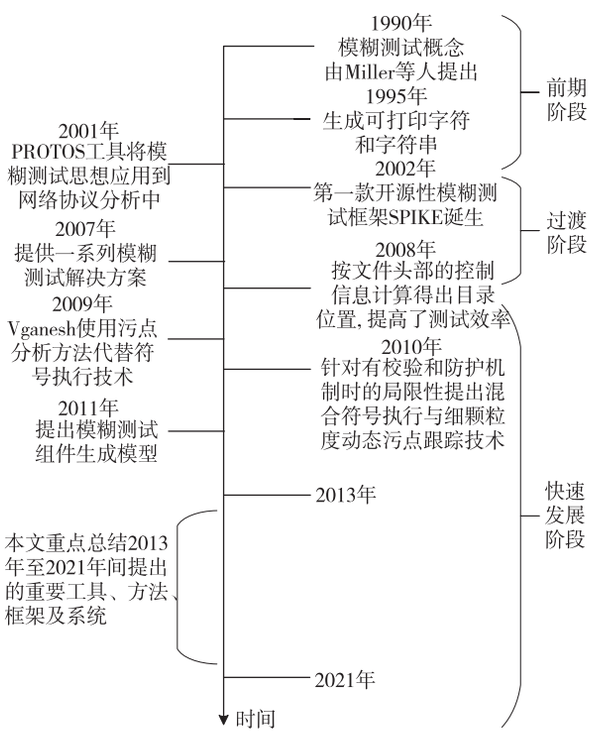

Figure 2 History of the development of fuzzing technologies

图 2 模糊测试技术发展史

1990 年,模糊测试由 Miller 等人[1]率先提出。1995 年第一款针对 UNIX 系统的模糊测试工具 Fuzz 诞生,Miller 设计的模糊测试工具包括一个随机字符串产生器[2],通过该产生器对 setuid 应用程序进行随机模糊测试,可提高 setuid 应用程序的可靠性。2001 年,Protos 项目[3]诞生,标志着模糊测试应用于网络协议分析的新起点,也是模糊测试技术成为实用性工具的开始。2002 年,Aitel[4]详细讲解并演示了基于生成的模糊测试工具 SPIKE。2008 年,针对 Microsoft Office 等复合文档安全,Gao 等人[5]提出了一种可以有效提高测试效率的测试用例构造方法。2009 年,基于污点分

析的 Vganesh 技术[6]被广泛应用。2010 年,Wang 等人[7]结合混合符号执行与细颗粒度动态污点跟踪技术,提出一种绕过系统校验和防护机制的方法,提高了漏洞挖掘能力。2011 年,Zhu 等人[8]提出了模糊测试组件生成模型,可以从庞大的数据集样本中自动生成具有较高覆盖率的测试用例。

2013 年至今的模糊测试技术发展,是本文重点总结的内容,将在第 3、4 节中详述,在图 2 中不再一一说明。

3 模糊测试的应用领域

模糊测试已成为挖掘系统漏洞的中坚力量,本节对模糊测试在各领域的重要应用进行归纳整理。通过对大量文献调研与归类,本文将模糊测试应用领域分为软件测试、协议测试及其他应用领域。

3.1 软件测试

3.1.1 Android 测试

随着技术水平和社会经济的迅速发展,智能手机在人们的日常生活中越来越普及,从衣食住行等不同方面影响着人们的生产和生活方式。智能手机开发平台多种多样,其中 Android 是应用最为广泛的平台之一。为应对快速增长的系统安全威胁,研究人员提出了针对智能手机系统安全的解决方法,例如静态分析技术[9]和自动动态分析技术[10,11]。

有效触发恶意行为是实现智能手机恶意软件自动化分析的前提。Wang 等人[12,13]专注于对抗 Android 恶意软件中的各种躲避技术,在自动动态分析中触发更多隐藏的恶意行为,从而帮助生成更全面的恶意软件报告。针对 Android 组件通信过程中由于组件暴露产生的安全问题,王凯等人[13]结合模糊测试和逆向分析技术,设计并实现了 Android 漏洞挖掘工具 KMDroid。张密等人[14]提出了一种基于模糊测试方法的组件通信测试方案,以 Android SDK 提供的数据为基础,引导高覆盖率的测试用例生成,设计并实现了工具 FuzzerAPP。关于 Android 应用程序的健壮性测试,赵赛等人[15]提出了一种基于模糊测试的检测方法,使用 Android 组件相关信息来构造测试用例,设计实现了全自动测试工具 ICCDroidFuzzer,实验中结合测试结果对源码进行分析,得出了抛出异常的根本原因。针对 Android 驱动安全,何远等人[16]提出了基于黑盒模糊测试的遗传算法,通过测试执行结果的实时反馈来引导遗传算法,实现了 Android 驱动的模糊测试系统 Add-fuzz(Android device driver fuzz)。

3.1.2 Windows 测试

图形用户界面 GUI(Graphical User Interface)可视化计算机程序,其目的是促进用户和设备之间的交互。GUI 测试是软件测试的关键部分,验证界面和用户之间的交互行为,而不考虑任何编码细节。Din 等人[17]提出了一种基于模糊自适应教与学优化 ATLBO(Adaptive Teaching Learning-Based Optimization)算法的 GUI 功能测试策略,该算法是基于教与学优化 TLBO(Teaching Learning-Based Optimization)算法的变体[17],利用事件交互图 EIG(Event-Interaction Graph)来生成高质量测试用例。为提高 GUI 测试时空转时机判定的准确率,张兴等人[18]提出了基于 Bi-Gram 模型以及统计分析的空转状态识别方法。

3.1.3 浏览器测试

针对浏览器安全,Zalewski[19]提出了模糊测试工具 AFL(AmericanFuzzy Lop),根据测试结果计算覆盖率,进而指导数据变异生成测试用例。霍玮等人[20]提出了一种基于模式生成的浏览器模糊测试方法,通过对测试模式分析、定义与自动化提取,产生模糊测试器完成测试数据构造,设计并实现了一个浏览器模糊测试工具 autofuzzy。不足之处在于对获取的测试模式没有一个较好的管理和反馈机制,还需要更多的基础数据来满足样本的生成。

3.2 协议测试

3.2.1 互联网协议

网络协议的安全问题近期受到人们的广泛关注。随着网络协议逆向工程的提出,针对手工模糊测试操作繁琐且代码覆盖率低的问题,李伟明等人[21]采用了一种可以对网络协议实现自动化识别并且能对其进行有效模糊测试的漏洞挖掘方法,显著提升了测试效率。针对现有网络不能较好地支持状态转换的问题,Ma 等人[22]根据有状态网络协议的特点,提出了有状态网络协议半有效模糊测试 SFSNP(Semi-valid Fuzzing for the Stateful Network Protocol),通过分析协议交互,构建出带有路径标记的扩展有限状态机,并根据扩展有限状态机、协议状态规则树及半有效算法获得模糊测试序列,采用状态转换标记算法,以减少冗余测试用例

生成,缩短测试时间。针对传输层安全协议 TLS(Transport Layer Security)安全性测试,Walz 等人[23]设计了一种通用的动态数据结构——通用消息树 GMTs(Generic Message Trees)。基于 GMT 概念,提出了一种随机算法来生成高度多样化和大部分有效的 TLS 握手消息;使用消息生成算法来差异测试现有较为流行的 TLS 服务器,经过实验验证其算法的高效性;开发的工具 tls-diff-testing 作为开源项目已经向外界公布。

支持实时流传输协议 RTSP(Real Time Streaming Protocol)的视频监控设备在市场中应用广泛,针对 RTSP 协议模糊测试用例改进,Ma 等人[24]先后提出了基于分类树、启发式算子、规则状态机和状态树指导模糊测试用例生成的方法,可有效删除无用项,大大提高了测试数据生成质量。

3.2.2 工控协议

工业控制系统是用于管理关键基础设施服务的操作技术系统的运行和功能的重要组成部分。Tacliad 等人[25]开发了一款模糊测试工具 ENIP Fuzz,用于检测可编程逻辑控制器中使用的以太网/IP 软件漏洞。针对传统模糊测试应用于数据采集与监控系统 SCADA(Supervisory Control And Data Acquisition)的不足,张亚丰等人[26]提出了一种基于状态的工控协议模糊测试方法,设计基于协议状态机的测试序列生成算法 PSTSGM(Protocol State based Test Sequences Generating Method),提出了基于心跳的异常监测与定位方法 HFDLM(Heartbeat based Faults Detection and Location Method),设计并实现了原型系统 SCADA-Fuzz,在此基础上对实际电力 SCADA 系统进行了测试,得到了良好的实验效果。为提高模糊测试用例的覆盖率和模糊测试效率,针对工控协议制造报文规范 MMS(Manufacturing Message Specification)的模糊测试,Kim 等人[27]采用对数据相关信息进行分类的方法引导测试用例生成。Li 等人[28]引入遗传算法并设计了其适应度函数来生成测试用例。Wang 等人[29]采用静态分析、动态监测和符号执行等策略,选择出安全性更强的种子对测试对象进行定向模糊测试,设计并实现了 Seeded-Fuzz 模型。李佳莉等人[30]提出了根据状态关系构造协议状态图,并基于协议状态图进行深度遍历选择覆盖率更高的测试用例的方法。

3.2.3 未知协议

随着互联网技术的不断进步,大量未知的、私有的通信协议广泛应用在不同领域。针对未知协议的复杂多样且安全性较差的问题,张蔚瑶等人[31]提出基于协议特征库的未知协议的逆向分析方法,自动学习协议结构和语义特征,提出多维变异的测试数据自动化生成方法生成测试数据,设计并实现了自动化模糊测试工具 UPAFuzz。

3.3 其他测试

除了上述应用场景外,模糊测试还广泛应用在内核安全测试[32]、机器人操作系统 ROS(Robot Operating System)[33]、云计算[34]、嵌入式固件[35]、软件定义网络 SDN(Software Defined Network)[36]和大数据[37]等方面。

针对图形处理单元 GPU(Graphics Processing Unit)内核高度复杂,导致模糊测试编程实现困难的问题,Peng 等人[32]提出了一种基于突变的模糊和选择性约束求解相结合的开放运算语言 OpenCL(Open Computing Language)内核测试生成技术,以实现快速、有效和可扩展。该方法随机改变输入内核参数值,以增加分支覆盖率。

针对 ROS 安全问题,王颖等人[33]提出了一种差分模糊测试方法,对输入文件进行加工处理并基于生成策略生成测试用例,对测试结果中的不一致输出进行差异计算并评估,分别对 ROS 不同版本的功能包进行测试。

针对云平台基础设施及服务 IaaS(Infrastructure as a Service)层虚拟化机制安全问题,结合已知虚拟化平台相关漏洞的先验知识,沙乐天[34]等人通过抽取并推演目标数据集合,提出了一种基于灰度马尔可夫模型的自动化预测方法,并实现了原型系统 VirtualFuzz。

针对嵌入式固件安全问题,Gao 等人[35]提出了一种将模糊测试与实时内存检查紧密结合的固件漏洞检测工具 EM-Fuzz。基于内存插件,固件模糊化不仅可以通过传统的分支覆盖引导生成高质量的种子来探索难以到达的区域,还可以通过记录的内存敏感操作来持续锻炼易受攻击的敏感区域。该工具还集成了现有模糊测试工具未实现的实时内存检查器来暴露内存漏洞。

针对 SDN 安全漏洞检测,Shukla 等人[36]提出了一种新的网络验证方法 PAZZ,通过定期检测测试数据包报头空间,将检测结果与实际数据平面状态进行比较,以自动检测和定位数据平面中表现不一致的故障。

数据密集型可扩展计算 DISC(Data Intensive Scalable Computing)系统有助于解决大数据可扩

展性问题,但由于数据具有不完整性和随时间不断发展的特点,实现自动化测试具有挑战性。Zhang等人[37]提出了一种基于覆盖导向的模糊测试工具BigFuzz。基于对DISC应用程序错误类型的深入研究,设计了模式感知的数据变异操作符。

4 模糊测试技术的发展现状

模糊测试工具是以指定文件格式或网络协议为测试目标,对支持目标格式或协议的应用程序进行压力测试,实现某一部分的功能;框架可描述为若干功能的整合体,是针对某个具体领域的通用解决方案,它允许使用可复用实用程序使其达到扩展最大化;系统的概念是更高的抽象,包括各组成元素、各元素的业务逻辑及交互关系等。通过大范围的调研,本节整理和总结近10年来模糊测试的发展近况,具体分为模糊测试工具、框架、系统及方法的汇总,并为方便读者查阅,在此节给出直观的表格。

4.1 模糊测试工具的发展

漏洞检测方法可分为静态分析方法和动态分析方法。利用静态分析方法对测试目标进行信息收集,可以为模糊测试提供可行的变异方案。Intent是Android通信中的一种运行绑定机制,用于连接程序运行过程的2个组件,是组件间信息交流的常用载体。为解决Android应用程序的安全问题,FuzzerAPP[14]、ICCDroidFuzzer[15]都是通过提取静态注册的组件信息,生成具有高覆盖率的Intent测试用例,同时监测Android设备日志文件,实现应用程序健壮性的自动化测试,不同点在于前者采用空域、交叉值、随机值的方法完成完全有效匹配、半有效匹配和完全不匹配Intent,而后者在测试用例生成策略上采用显式Intent和隐式Intent构造相结合的方法。

动态分析方法是一种重要的动态检测方法,在测试过程中通过检测测试目标状态,从而检测出是否存在漏洞的技术。针对浏览器特定测试模式漏洞挖掘测试成本高、开发周期长以及生存时间短的问题,Autofuzzy[20]结合静态识别和动态执行的方法获取浏览器模块间的依赖关系,根据依赖关系形成测试模式。李伟明等人[38]结合静态分析和动态跟踪技术,提出了基于内存模糊测试工具PyFuzzer。该工具首先通过对输入数据中的危险函数进行优先级划分,然后基于动态跟踪技术对其进行自动化识别,最后采用基于快照恢复的方法对测试对象进行内存模糊测试。基于动态符号执行和动态污点分析技术,EWFT(Execution-based Whitebox Fuzzing Tool)[39]在动态执行目标程序时进行动态指令插桩,得到准确的动态程序执行信息,通过构建程序执行路径之间的动态结构和各数据元素变量之间的依赖关系,对相关敏感操作进行脆弱性分析。基于遗传变异算法构造测试用例,通过执行测试用例时监控执行状态和计算程序路径权重,完成对目标程序的模糊测试。EM-Fuzz[35]和Mem-Lock[40]皆利用覆盖率和内存消耗信息来动态指导模糊化过程。

种子生成策略是模糊测试整个过程的核心组成部分。针对以太网虚拟机EVM(Ethereum Virtual Machine)安全问题,Fu等人[41]提出了第一个使用差分模糊技术检测EVM漏洞的工具EVM-Fuzzer。该工具的核心思想是通过不断生成种子合约,并将其输入到目标EVM和基准EVM,从而尽可能多地发现执行结果之间的不一致,通过输出交叉引用发现漏洞。为有效解决种子输入选择时代码覆盖率低的问题,MutaGen[42]通过动态地改变生成代码的程序片段,利用输入格式信息来引导生成具有高覆盖率的测试用例,能够达到被测程序的深层状态。

逆向技术是一种技术重现技术,它通过对目标进行逆向分析及研究,可推演出功能相近或者结构相似的结果。近些年来,逆向技术为模糊测试提供了可靠且有效的技术支持。例如针对Android私有组件保护问题,KMDroid[13]结合模糊测试技术和逆向分析方法,可以有效挖掘出应用通信过程中的安全漏洞。多文本格式RTF(Rich Text Format)是一种跨平台文档格式,RTF数组溢出漏洞挖掘工具RAVD(RTF Array Vulnerability Detector)[43]通过逆向分析文件结构,明确文件结构中特定数据块与代码块的联系,可提高漏洞的挖掘效率,实现定向模糊测试,可挖掘RTF数组溢出漏洞。RAVD的缺点是采用手工逆向分析,导致整体效率较低,且存在一定的错检率。BigFuzz[37]与RAVD设计逻辑较为相似,通过抽象框架代码和分析应用程序逻辑,执行源到源的自动化转换,以构建一个适用于快速测试的等效数据密集型可扩展计算DISC应用程序,使DISC应用程序的模糊测试变得易于处理。与随机模糊测试相比,BigFuzz在测试时间、应用程序代码覆盖率、检测应用程序错误上均有较大提高。MutaGen[42]考虑静态反汇编的不确定性,使用动态切片的方式选择变异

的指令,以此提高代码的覆盖率。

此外,逆向技术和模糊测试技术被广泛应用于协议的逆向分析与漏洞检测中。针对未知协议安全测试,张蔚瑶等人[31]提出了 UPAFuzz 工具,对输入协议数据进行分析和学习,可以在无先验知识的情况下实现未知协议的分析和格式提取,基于协议特征形成描述协议特征的脚本程序,结合常见漏洞攻击模式生成多维变异的测试数据。对比现有模糊测试工具,UPAFuzz 具有效率更高、性能更优的优势。SFSNP[22]根据协议交互建立带有路径标记的扩展有限状态机模型,基于该模型获得测试序列,对测试序列中的每个状态转换执行半有效变异操作,得到协议模糊测试用例,并采用状态转移标记算法来解决在生成协议模糊化序列时生成多余测试用例的问题,SFSNP 不仅能保证发现漏洞,还能减少测试用例数量,缩短测试执行时间。

灰盒模糊测试中影响较为广泛的研究成果是 AFL[19],其工作原理是在程序编译时进行插桩,同时引入了进化算法,模糊测试的效果得到了有效改善,但 AFL 测试覆盖范围较小,这严重限制了它发现漏洞的能力。Fairfuzz[44]是由 Lemieux 等人提出的基于 AFL 的改进工具。该工具通过自动识别由少数 AFL 生成的输入(稀有分支)执行的分支,结合针对稀有分支的变异掩码创建算法,来增加 AFL 的覆盖范围。通过实验对比,FairFuzz 可以更快的速度实现高分支覆盖率。

上述各工具汇总及特点分析如表 2 所示。

4.2 模糊测试框架的发展

近几年,模糊测试框架不断创新,为软件测试提供更为高效的漏洞检测技术。例如,目前 APP 测试任务大部分交给了工作人员,而人工提交的测试报告存在冗余度高、质量参差不齐的问题,测试报告模糊聚类框架 TERFUR[45]通过将冗余和错

Table 2 Development of fuzzing tools

表 2 模糊测试工具发展

名称	诞生时间	测试对象	工具特点	实验环境
PyFuzzer[38]	2013	WarFTPD 1.65	结合静态分析和动态跟踪技术,基于快照恢复的方法对测试对象进行内存模糊测试	Windows7SP1
KMDroid[13]	2014	Android 暴露组件	结合模糊测试技术和逆向分析方法对私有组件进行保护性测试	Android
EWFT[39]	2014	bzip2;gzip;Xpdf;Transmission;Kaffeine;Dia	基于动态符号执行和动态污点分析技术,采用遗传变异算法构造测试用例	X86/Linux
MutaGen[42]	2015	Linux 程序	在生成程序的机器代码上使用动态切片的方式进行变异,提高代码的覆盖率	Linux Mint 17.1
RAVD[43]	2017	RTF 文件	通过逆向分析文件结构,明确文件结构中特定数据块与代码块的联系,实现定向模糊测试	Windows 7 SP1
FuzzerAPP[14]	2017	Android 组件	通过分析系统文件和应用程序的反编译,获取待测 APP 的组件信息,利用构建策略生成测试用例	Android
ENIP Fuzz[25]	2017	EtherNet/IP service requests	基于 Scapy 开源框架实现 EtherNet/IP 支持库,通过远程故障检测策略实现漏洞检测	Windows 7 Mac OS X
SFSNP[22]	2017	SMTP 协议	基于有限状态机模型对各状态进行半有效变异,采用状态标记算法解决测试用例冗余问题	Windows 7
Autofuzzy[20]	2018	浏览器	基于静态分析和动态分析方法获取浏览器组件关系,而今生成测试模式,完成浏览器漏洞挖掘	Windows 7
FairFuzz[44]	2018	Xmllint;Tcpdump;c++filt;readelf;readpng;ob jdump	通过自动识别由少数 AFL 生成的执行分支,结合变异掩码创建算法,增加 AFL 的覆盖范围	--
EVMFuzzer[41]	2019	EVM	通过一组预定义的生成种子合约,采用动态优先级调度算法来指导种子合约选择,利用基准 EVM 作为交叉引用预言机来避免手动检查	Ubuntu 16.04.4
BigFuzz[37]	2020	DISC 系统	通过抽象框架代码和分析应用程序逻辑,构建可适用模糊测试的 DISC 应用程序	Ubuntu 16.04
UPAFuzz[31]	2020	未知协议	基于逆向分析方法对输入协议数据进行分析和学习,获取协议特征,结合常见漏洞攻击模式生成多维变异的测试数据	Ubuntu 16.04.5 LTS; Windows10 professional
ICCDroidFuzzer[15]	2020	Android 暴露组件	通过静态分析获取组件相关信息来构造测试用例	Android 7.1.2 MiA1
EM-Fuzz[35]	2020	嵌入式固件	基于内存检测,通过代码覆盖率和内存信息动态指导模糊测试过程	Ubuntu 16.04.6 LTS
MemLock[40]	2020	Nm;nasm;openjpeg;Bento4;libsass	使用覆盖和内存消耗信息来指导模糊化过程	Ubuntu LTS 16.04

误的测试报告聚合到集群中来减少检查的测试报告的数量。该框架包含过滤器、预处理器以及两阶段合并组件分别解决无效屏障、不均匀屏障和多漏洞屏障的问题。针对高级恶意软件通常采用各种规避技术来隐藏恶意行为问题,DirectDroid[12]通过实现静态分析器和动态执行器,分别完成待检测的 Android 应用程序包 APK(Android application PacKage)信息收取和为静态分析器反馈动态信息,并处理了强制执行导致的潜在程序崩溃漏洞。黄桦烽等人[46]建立的程序运行时漏洞模型 Weak-Tainted 结合脆弱性和污点属性对漏洞进行定义,通过漏洞挖掘、分析及利用流程形成一体化的描述规范,采用基于页面标签的动态污点传播分析方法和基于输出特征反馈的输入求解方法,有效解决了符号执行无法求解的问题。

为满足通用型模糊测试器的低成本定制和高可扩展性需求,杨梅芳等人[47]提出了一种可编程模糊测试框架 Puzzer。首先提出一种基于模糊测试原语的模糊测试器编写语言规范,根据该规范编写制导程序,通过解析器解析,生成模糊测试抽象语法树。通过结合遍历语法树和模糊测试原语库,模糊测试引擎可生成一个由 Python 语言编写的模糊测试器。该框架可有效降低模糊测试器的开发成本与门槛,且生成的模糊器具有高扩展性、高针对性的优点。

覆盖率是模糊测试评估中的一大指标,具体来说覆盖率指的是在执行测试用例的过程中,测试对象被覆盖到的分支占所有分支的比重。具有较高覆盖率的测试用例往往能够发现更多的漏洞。基于覆盖率的模糊测试工具的典型代表是 AFL[19],其核心思想是增加边覆盖率。除此之外,各种增加覆盖率的模糊测试框架被相继提出。DeepHunter[48]是一种覆盖引导的模糊测试框架,采用一种结合基于多样性的种子选择策略,可提高代码覆盖率,解决基于深度神经网络的软件安全问题。Zhou 等人[49]提出了一个覆盖敏感的跟踪和调度框架 Zeror,利用自修改跟踪机制和实时调度机制,为更有效地覆盖收集提供零开销检测。

大多数基于覆盖的模糊器平等地考虑程序的所有部分,并且过多地关注如何提高代码覆盖,这种方式是低效的。Li 等人[50]设计并实现了一个名为 V-Fuzz 的进化模糊化框架。对于给定的一个二进制程序,漏洞预测模型将给出一个关于易受攻击程序片段的先验估计,模糊器利用进化算法结合易受攻击预测结果引导生成更有可能到达易受攻击位置的输入用例,能够在有限的时间内高效快速地发现二进制程序的错误。

在协议测试方面,tls-diff-testing[23]实现了通用消息树 GMT 概念的软件框架,允许在不依赖协议本身的情况下完成 TLS 协议消息语法解析。

上述各模糊测试框架汇总如表 3 所示。

4.3 模糊测试系统的发展

根据测试前对目标对象内部结构的掌握情况,模糊测试方法可分为黑盒模糊测试、白盒模糊测试和灰盒模糊测试。很多模糊测试系统的提出都是基于这 3 种模糊测试方式之一进行的相关改进,并应用在不同领域中。例如,为提高 Android 驱动漏洞挖掘能力,何远等人[16]提出了基于黑盒测试的遗传算法,设计并实现了模糊测试系统 Add-Fuzz,利用测试结果递归指导和改进遗传算法,生成高效的测试用例,根据已知漏洞特征优化可变参数输入,以提高模糊测试的命中率。ArtFuzz[51]采用基于白盒模糊测试的方法提高暴露通过缓冲区溢出进行恶意输入漏洞的概率,通过模糊测试结果和内存布局信息指导生成测试用例。

Table 3 Development of fuzzing framework

表 3 模糊测试框架发展

名称	诞生时间	测试对象	改进对象	实验环境
TERFUR[45]	2018	APPs	测试报告冗余	Windows 8.1 JavaJDK1.8.0_60
Puzzer[47]	2018	--	开发成本与门槛	Windows 7×86; Linux Ubuntu ×86-64
DeepHunter[48]	2019	DNNs	种子选择	搭载 Ubuntu 16.04 系统的服务器
DirectDroid[12]	2019	恶意软件	恶意规避检查行为	Windows 10 Android 4.4.2 模拟器
Weak-Tainted[46]	2019	Linux 二进制程序	有限资源下漏洞利用	Ubuntu 16.04.5 64b
V-Fuzz[50]	2020	二进制程序	测试效率	IDA Pro PyTorch
Zeror[49]	2020	fuzzer-test-suite	测试速度、代码覆盖率	Linux 5.5.13

模糊测试技术的改进不仅仅体现在模糊测试用例的生成上,其他步骤的改进也同样影响测试的整体指标,例如在评估阶段如何对目标系统的异常行为进行及时检测与精确定位。为提高测试引擎对程序进入空转状态时机判断的能力,张兴等人[18]提出了基于Bi-Gram模型以及统计分析的空转状态识别方法。Bi-Gram算法大量收集正常状态下目标程序的执行迹,并采用特征提取方法将执行迹转变为特征序列。通过统计分析实现空转特征序列的提取,将得到的空转特征输入实时检测算法中,实现空转时机实时检测。PAZZ[36]采用比较控制平面和数据平面的转发规则、拓扑及路径的方法进行异常检测和定位。利用当前控制信息主动计算所有源到目标的可达数据包,生成预期报告,并对路径和规则进行编码;模糊器计算控制器和流量未覆盖的数据包报头空间,同时为模糊测试网络生成活动流量;数据平面组件对转发规则的路径和序列进行编码,以生成采样的实际报告;通过比较预期报告和实际报告实现故障的检测和定位,解决软件定义网络中发生实际数据平面状态与预期控制平面状态不一致时定位困难的问题。

除此之外,还有一些针对特定领域所提出的应用型原型系统,例如SCADA-Fuzz[26]和Virtual-Fuzz[34]。

Peach[52]是以黑盒模糊测试方法为技术背景所诞生的测试工具,主要应用于网络、文件、物联网模糊测试中。SCADA-Fuzz[26]系统是在Peach基础上实现的工控协议模糊测试原型系统,目的是对电力SCADA系统进行模糊测试。以工控协议会话流量报文作为输入,通过状态机推断模块得到协议状态机,由脚本生成模块编写系统可识别的描述脚本,结合迁移信息和用例生成策略生成包含正常交互的模糊测试用例。SCADA-Fuzz虽然耗时较长,但可发现Peach未成功检测出的拒绝服务漏洞。

VirtualFuzz[34]是面向IaaS层虚拟化平台实现的原型系统,该系统基于源码静态审计的方法得出虚拟化各模块的函数关系,通过与静态源码对比,进行动态调试分析动态函数调用路径和执行结果,以此确定函数执行的具体信息。基于标准化的随机种子生成方法生成测试用例,将其输入到目标系统后与基于灰度马尔科夫链的预测模型进行拟合,从而指导当前虚拟化模块下的模糊测试方向,可实现虚拟化漏洞高效挖掘与检测。

上述各模糊测试系统汇总如表4所示。

4.4 模糊测试方法的创新

应用较为广泛的程序分析技术包括符号执行和污点分析。符号执行通过推导一个表示程序执行的逻辑等式,基于此等式可推断出该程序在不同输入上的行为。污点分析的技术思想在于观测程序数据受到污染源的污染范围,以此来跟踪污染源和汇聚点之间的信息流。模糊测试技术结合程序分析技术的优点,在测试方法上有着重要创新。

结合符号执行与模糊测试的优点,谢肖飞等人[53]提出了基于符号执行与模糊测试的混合测试方法Afleer。该方法首先将给定程序进行编译,生成经过插桩的可执行文件,利用模糊测试和符号执行技术,不断地生成具有高覆盖率的测试用例并更

Table 4 Development of fuzzing system

表4 模糊测试系统发展

名称	诞生时间	测试对象	实验环境
SeededFuzz[29]	2016	mpeg3dump、png2swf、gif2swf、cjpeg、speexenc 1.2	--
ArtFuzz[51]	2016	Elecard MPEG Player 5; RealPlayer SP 1.1.4; Adobe Reader 11.0; Notepad++ 6.6.9; LeapFTP 3.1.0.50;	Windows XP SP4
Add-fuzz[16]	2017	Android驱动程序	三星GT-I9300 Android4.1; 华为mtl-u06 Android4.1; 小米4Android4.4; 三星GT-N7100 Android4.1; 三星G9280 Android5.1
Bi-Gram[18]	2017	Windows GUI软件	Windows 7或更高
SCADA-Fuzz[26]	2017	工控协议	Windows XP SP3
VirtualFuzz[34]	2018	IaaS层虚拟化平台	Windows7旗舰版 Service Pack1; SUSE Linux Enterprise Server 12
PAZZ[36]	2020	网络验证	--

新覆盖信息,符号执行负责遍历程序执行树搜索未覆盖的分支,经过反复迭代,该方法可覆盖更多的分支,从而发现更多的漏洞。为检测二进制文件中的漏洞,Chen 等人[54]提出了彩色污点分析方法CTAM(Colorful Taint Analysis Method)来计算警戒条件,使用不同的污点颜色确定目标程序中的输入数据与变量之间的关系,与传统污点分析方法相比,该方法可减少时间和空间开销。为解决大型或路径条件复杂的目标程序模糊测试效率低下的问题,李明磊等人[55]提出了一种针对复杂路径条件下的漏洞检测技术 SymFuzz,结合导向式模糊测试技术与选择符号执行技术,首先通过静态分析定位漏洞函数,利用导向式模糊测试技术探索程序路径,以生成可达漏洞函数的测试用例。不足之处在于 SymFuzz 对静态分析的依赖性较高。

符号执行和污点分析技术都有一个缺点,由于程序的复杂性,其存在的多条分支可引起路径爆炸的问题,且路径爆炸问题会随着程序复杂程度的增加变得更加严峻。对此,部分研究人员致力于解决该问题。例如 PWA(Path Weight Analysis)算法[39]是动态检测二进制程序中针对程序执行路径空间所提出的覆盖测试算法,通过分析目标程序相关信息并结合程序执行的路径空间测试方法构造测试用例集合,解决程序执行路径空间覆盖率低、路径状态空间爆炸等问题。

PSTSGM[26]通过遍历协议状态机有向图,建立状态引导报文序列库,根据状态引导报文序列实现对目标系统的状态引导,有助于被测设备触发异常。HDLMF[26]则是基于心跳的异常监测与定位方法,利用心跳探测和回溯定位来精确发现出现异常的目标序列,可实现工控嵌入式设备利用传统异常检测方法进行漏洞安全检测。值得注意的是,与 HDLMF 方法核心类似,Autofuzzy[20]采用活动性检查、意外响应和性能测量,在测试期间监控远程设备,实现远程故障检测策略;UPAFuzz[31]基于逆向分析技术可与被测设备实时交互,可主动探测被测设备运行状态。

机器学习技术已经成熟,部分研究人员已实现把机器学习更好地应用在模糊测试上。例如 Lean&fuzz[56]是 Godefroid 等人提出的使用样本输入和基于神经网络的统计机器学习技术自动生成适合输入语法的测试用例的方法。将输入的 PDF 文件对象看作字符序列,基于循环神经网络的字符集语言模型学习 PDF 序列的生成模型,结合对象生成策略构建新的 PDF 对象实例。该算法基于输入概率分布智能地指导模糊输入的位置,不仅可生成格式良好的对象实例,而且能够增加 PDF 解析器的覆盖率。

上述各模糊测试方法的特点汇总如表 5 所示。

4.5 模糊测试用例生成算法改进

传统模糊测试用例存在覆盖率低的问题,相关科研工作者提出结合测试结果优化生成策略和利用机器学习的方法改进测试用例生成过程。除了前面提到的 Lean&fuzz[56]外,还有一些其他重要的改进算法被相继提出。例如在并行模糊测试中,邹燕燕等人[58]提出了一种基于变异策略感知的并行模糊测试方法,通过动态调整变异策略和进行多实例间有效样本同步,减少测试实例的冗余度。为解决高结构样本识别问题,张羿辰等人[59]提出了

Table 5 Innovation of fuzzing testing methods

表 5 模糊测试方法的创新

名称	诞生时间	针对目标	方法特点	实验环境
CTAM[54]	2012	Grep、7za、Fsum、Image2PDF	基于黑盒和白盒测试	Windows XP SP3
PWA[39]	2014	bzip2、gzip、Xpdf、Transmission、Kaffeine、Dia	对目标程序执行空间进行的覆盖式测试	X86/Linux(kernel 2.6)
PSTSGM[26]	2017	工控协议	结合协议状态机,实现状态报文序列库的建立	Windows XP SP3
HFDLM[26]	2017	工控协议	利用心跳探测和回溯定位精确从而发现异常序列	Windows XP SP3
Lean&fuzz[56]	2017	PDF 文件	利用样本输入和基于神经网络的统计机器学习技术生成输入语法	Windows 10 VMs
Afleer[53]	2019	LAVA-M[57]、oSIP	基于符号执行与模糊测试的混合测试方法	Ubuntu 16.04LTS
SymFuzz[55]	2021	libxml2、libtiff、libarchive、OpenJPEG、ImageMagick	结合导向式模糊测试技术和符号选择执行技术的复杂路径漏洞检测技术	Ubuntu16.04

一种基于神经网络的敏感区域预测的模糊测试方法,利用变异算法生成与敏感区域相关联的种子,通过循环增量学习,不断完善预测算法模型,可高效提升模糊测试效率和检测深度。针对 Web 安全模糊测试,涂玲等人[60]提出了将测试模板规则与网站过滤协议变形相结合的测试用例生成方法,实现在不同过滤机制下生成最有效的测试用例,并且采用基于污染传播策略的漏洞响应数据分析方法解决漏洞响应的有效验证问题。刘渊等人[61]提出了一种基于遗传算法的漏洞挖掘测试用例生成方法,解决多输入测试用例难以生成和非线性问题求解困难问题。为解决 Web 应用跨站脚本 XSS (Cross-Site Scripting)安全性问题,程诚等人[62]提出了一种基于模糊测试和遗传算法的 XSS 攻击样本优化生成方法,以有效挖掘漏洞。

5 模糊测试的机遇与挑战

模糊测试技术从诞生发展至今,已逐步成为一种占据重要地位的安全测试技术。随着网络规模的扩大和应用种类的增多,协议逆向的时效性要求越来越高,自动化协议逆向分析技术成为人们追求的目标,将模糊测试应用在协议逆向分析上是一重要方向。云计算技术因其可虚拟化和大规模化的优势不断满足人们的需要,这也对云计算环境安全提出新的挑战。结合机器学习和云计算等新兴技术的发展,模糊测试技术与之相融是大势所趋。模糊测试拥有强大的漏洞检查能力,同样也为恶意攻击者提供了便利,因此这也为模糊测试对抗技术带来了巨大的挑战。根据不同需求,大量的模糊测试工具应运而生,如若将各种工具进行集成化处理,则可减少针对不同测试目标的开发周期和实现成本。综上所述,模糊测试技术未来的研究热点可能包括以下几个方面:

(1)应用于协议逆向工程。

一个完整的协议逆向分析系统包括输入预处理、协议格式提取和协议状态机推断 3 个阶段。模糊测试技术作为软件或系统漏洞检测的先进技术之一,可以为协议逆向系统提供技术支持。例如在状态机推断阶段,模糊测试技术可以为协议状态机生成测试序列,对其进行漏洞检测。在模糊测试技术应用在协议逆向工程方面,李伟明等人[21]和张蔚瑶等人[31]进行了相关研究。

(2)应用于云平台。

云计算具有规模大、可虚拟化、可靠性高、通用性强、可扩展性高和价格低等优点,近几年的发展和应用格外迅速。云计算在为人们日常生活提供便利的同时,也带来了较大的安全威胁。利用模糊测试技术对云平台进行安全防护,可以说是今后模糊测试技术的一个重要研究方向。关于模糊测试技术在云计算方面的研究,读者可阅读林闯等人[63]和沙乐天等人[34]发表的相关文章。

(3)与新兴技术的结合。

近几年来,机器学习、云计算技术等新兴技术迅速崛起,模糊测试技术也同样可以针对不同的需要进行合理且高效的利用。例如模糊测试技术想要利用机器学习,就必须解决数据源的处理、样本数据不平衡问题以及模糊测试的执行速度等一系列问题。

(4)模糊测试对抗技术的研究。

随着软件和网络安全意识的增强,模糊测试作为一种高效的漏洞检测技术,深受研究人员的热爱。事物都有其两面性,模糊测试技术也不例外。模糊测试可以作为安全漏洞检测技术存在,目的是安全防护,正是由于其在该方面的突出表现,恶意破坏者也同样可以利用模糊测试进行攻击破坏。因此,模糊测试技术在发展的同时要统筹兼顾反模糊测试技术。无论是模糊测试对抗技术自身的研究,还是其带给模糊测试技术的挑战,都是研究的重要方向。

(5)模糊测试工具的集成。

为针对不同需要,如今模糊测试工具的种类繁多,可以说是各有利弊。例如著名的基于生成的模糊测试工具 SPIKE[4]使用基于块的方法定义输入格式,虽然 SPIKE 在测试方面表现得非常成功,但对于不容易转换为块模式的输入数据来说,利用 SPIKE 则不能取得良好的测试效果。在低成本与高扩展性的基础上实现模糊测试工具的集成,应该是模糊测试今后的重要发展方向。

6 结束语

模糊测试是一种安全测试技术,主要应用于安全漏洞检测。本文详细介绍了模糊测试的基本概念、测试流程、技术方法以及应用。特别地,本文总结了近 10 年来的优质论文,并对模糊测试工具、框架、系统及方法的创新发展进行了分类和概要说明,这有助于相关科研人员进行查阅和总结。最后,本文讨论了模糊测试技术的挑战和发展方向,为了模糊测试技术的发展,鼓励进一步的研究和实

践以解决相关问题。

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] Miller B P. Fuzz Revisited:A re-examination of the reliability of UNIX utilities and services[J/OL]. [2001-12-09]. ftp://grilled. cs. wisc. edu/technical_papers/fuzz-revisited. ps. Z.

[3] Kaksonen R,Laakso M,Takanen A. System security assessment through specification mutations and fault injection[C]// Proc of International Conference on Communications and Multimedia Security Issues of the New Century, 2001:27.

[4] Aitel D. The advantages of block-based protocol analysis for security testing[Z]. USA:Immunity Inc,2002.

[5] Gao Jun, Xu Zhi-da, Li Jian. Research on fuzzing test for compound cocument[J]. Computer & Digital Engineering, 2008,36(12):116-119. (in Chinese)

[6] Ganesh V,Leek T,Rinard M. Taint-based directed whitebox fuzzing[C]//Proc of the 31st IEEE International Conference on Software Engineering,2009:474-484.

[7] Wang T,Tao W,Gu G,et al. TaintScope:A checksum-aware directed fuzzing tool for automatic software vulnerability detection[C]//Proc of the IEEE Symposium on Security and Privacy,2010:497-512.

[8] Zhu X Y,Wu Z Y,Atwood J W. A new fuzzing method using multi data samples combination[J]. Journal of Computers, 2011,6(5):881-888.

[9] Fratantonio Y,Bianchi A,Robertson W,et al. TriggerScope: Towards detecting logic bombs in Android applications[C] // Proc of the IEEE Symposium on Security and Privacy,2016: 377-396.

[10] Zhang Y,Yang M,Yang Z,et al. Permission use analysis for vetting undesirable behaviors in Android apps[J]. IEEE Transactions on Information Forensics and Security,2014,9 (11):1828-1842.

[11] Fu H,Hu P,Zheng Z,et al. Towards automatic detection of nonfunctional sensitive transmissions in mobile applications [J]. IEEE Transactions on Mobile Computing, 2021, 20 (10):3066-3080.

[12] Wang X,Yang Y,Zhu S. Automated hybrid analysis of android malware through augmenting fuzzing with forced execution[J]. IEEE Transactions on Mobile Computing,2019, 18(12):2768-2782.

[13] Wang Kai, Liu Qi-xu, Zhang Yu-qing. Android inter-application communication vulnerability mining technique based on fuzzing[J]. Journal of University of Chinese Academy of Sciences,2014,31(6):827-835. (in Chinese)

[14] Zhang Mi,Yang Li,Zhang Jun-wei. FuzzerAPP:The robustness test of application component communication in Android[J]. Journal of Computer Research and Development, 2017,54(2):338-347. (in Chinese)

[15] Zhao Sai,Liu Hao,Wang Yu-feng,et al. Fuzz testing of Android inter-component communication[J]. Computer Science,2020,47(S2):303-309. (in Chinese)

[16] He Yuan,Zhang Yu-qing,Zhang Guang-hua. Android driver vulnerability discovery based on black-box genetic algorithm [J]. Chinese Journal of Computers,2017,40(5):1031-1043. (in Chinese)

[17] Din F,Zamli K. Fuzzy adaptive teaching learning-based optimization strategy for GUI functional test cases generation [C]//Proc of the 7th International Conference on Software and Computer Applications,2018:92-96.

[18] Zhang Xing,Feng Chao,Lei Jing,et al. Real time idle state decection method in fuzzing test in GUI program[J]. Journal of Software,2018,29(5):1288-1302. (in Chinese)

[19] Zalewski M. American fuzzy lop [CP/OL]. [2021-08-14]. https://lcamtuf. coredump. cx/afl/.

[20] Huo Wei,Dai Ge,Shi Ji,et al. Browser fuzzing technology based on pattern-generation[J]. Journal of Software,2018, 29(5):1275-1287. (in Chinese)

[21] Li Wei-ming, Zhang Ai-fang,Liu Jian-cai,et al. An automatic network protocol fuzz testing and vulnerability discovering method[J]. Chinese Journal of Computers,2011,34(2): 242-255. (in Chinese)

[22] Ma R,Ren S M,Ma K,et al. Semi-valid fuzz testing case generation for stateful network protocol[J]. Tsinghua Science and Technology,2017,22(5):458-468.

[23] Walz A, Sikora A. Exploiting dissent: Towards fuzzing-based differential black-box testing of TLS implementations [J]. IEEE Transactions on Dependable and Secure Computing,2020,17(2):278-291.

[24] Ma R,Wang D,Hu C,et al. Test data generation for stateful network protocol fuzzing using a rule-based state machine [J]. Tsinghua Science and Technology, 2016, 21(3): 352-360.

[25] Tacliad F,Nguyen T,Gondree M. DoS exploitation of allen-bradley's legacy protocol through fuzz testing[C]//Proc of the 3rd Annual Industrial Control System Security Workshop,2017:24-31.

[26] Zhang Ya-feng,Hong Zheng,Wu Li-fa,et al. Protocol state based fuzzing method for industrial control protocol[J]. Computer Science,2017,44(5):132-140. (in Chinese)

[27] Kim S, Jo W, Shon T. A novel vulnerability analysis approach to generate fuzzing test case in industrial control systems[C] //Proc of the IEEE Information Technology,Networking,Electronic and Automation Control Conference, 2016:566-570.

[28] Li H F,Wang S L,Zhang B, et al. Network protocol security testing based on fuzz[C]//Proc of the 4th International Conference on Computer Science and Network Technology, 2015:955-958.

[29] Wang W,Sun H,Zeng Q. SeededFuzz:Selecting and generating seeds for directed fuzzing[C]//Proc of the 10th International Symposium on Theoretical Aspects of Software Engineering,2016:49-56.

[30] Li Jia-li, Chen Yong-le, Li Zhi, et al. Mining RTSP protocol vulnerabilities based on traversal of protocol state graph [J]. Computer Science, 2018, 45(9): 171-176. (in Chinese)

[31] Zhang Wei-yao, Zhang Lei, Mao Jian-ling, et al. An automated method of unknown protocol fuzzing test[J]. Chinese Journal of Computers, 2020, 43(4): 653-667. (in Chinese)

[32] Peng C, Rajan A. Automated test generation for OpenCL kernels using fuzzing and constraint solving[C]//Proc of the 13th Annual Workshop on General Purpose Processing using Graphics, 2020: 61-70.

[33] Wang Ying, Wang Bing-qing, Guan Yong, et al. Differential fuzz testing of robot operating system[J]. Journal of Software, 2021, 32(6): 1867-1881. (in Chinese)

[34] Sha Le-tian, Xiao Fu, Yang Hong-ke, et al. Vulnerability discovery method for virtualization in IaaS based on self-adapting fuzzing test[J]. Journal of Software, 2018, 29(5): 1303-1317. (in Chinese)

[35] Gao J, Xu Y, Jiang Y, et al. EM-Fuzz: Augmented firmware fuzzing via memory checking [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(11): 3420-3432.

[36] Shukla A, Saidi S J, Schmid S, et al. Toward consistent SDNs: A case for network state fuzzing[J]. IEEE Transactions on Network and Service Management, 2020, 17(2): 668-681.

[37] Zhang Q, Wang J, Gulzar M A, et al. BigFuzz: Efficient fuzz testing for data analytics using framework abstraction[C]// Proc of the 35th IEEE/ACM International Conference on Automated Software Engineering, 2020: 722-733.

[38] Li Wei-ming, Yu Jun-qing, Ai Shao-bo. PyFuzzer: Automatic in-memory fuzz testing method[J]. Journal on Communications, 2013, 34(S2): 64-68. (in Chinese)

[39] Wang Ying, Gu Li-ze, Yang Yi-xian, et al. EWFT: Execution whitebox fuzzing for executables[J]. Chinese Journal of Electronics, 2014, 42(10): 2016-2023. (in Chinese)

[40] Wen C, Wang H, Li Y, et al. MEMLOCK: Memory usage guided fuzzing[C]//Proc of the 42nd IEEE/ACM International Conference on Software Engineering, 2020: 765-777.

[41] Fu Y, Ren M, Ma F, et al. EVMFuzzer: Detect EVM vulnerabilities via fuzz testing[C]//Proc of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019: 1110-1114.

[42] Kargén U, Shahmehri N. Turning programs against each other: High coverage fuzz-testing using binary-code mutation and dynamic slicing[C]//Proc of the 10th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2015: 782-792.

[43] Le De-guang, Gong Sheng-rong, Wu Shao-gang, et al. Research on RTF array overflow vulnerability detection [J]. Journal on Communications, 2017, 38(5): 96-107. (in Chinese)

[44] Lemieux C, Sen K. FairFuzz: A targeted mutation strategy for increasing greybox fuzz testing coverage[C]//Proc of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018: 475-485.

[45] Jiang H, Chen X, He T, et al. Fuzzy clustering of crowdsourced test reports for apps[J]. ACM Transactions on Internet Technology, 2018, 18(2): 1-28.

[46] Huang Hua-feng, Wang Jia-jie, Yang Yi, et al. Automatic software vulnerability discovery and exploit under the limited resource conditions[J]. Journal of Computer Research and Development, 2019, 56(11): 2299-2314. (in Chinese)

[47] Yang Mei-fang, Huo Wei, Zou Yan-yan, et al. Programmable fuzzing technology[J]. Journal of Software, 2018, 29(5): 1258-1274. (in Chinese)

[48] Xie X F, Ma L, Juefei-Xu F, et al. DeepHunter: A coverage-guided fuzz testing framework for deep neural networks[C]//Proc of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2019: 146-157.

[49] Zhou C, Wang M, Liang J, et al. Zeror: Speed up fuzzing with coverage-sensitive tracing and scheduling[C]//Proc of the 35th IEEE/ACM International Conference on Automated Software Engineering, 2020: 858-870.

[50] Li Y, Ji S, Lyu C, et al. V-Fuzz: Vulnerability prediction-assisted evolutionary fuzzing for binary programs[J]. IEEE Transactions on Cybernetics, 2020, 52(5): 3745-3765.

[51] Chen K, Zhang Y, Liu P. Dynamically discovering likely memory layout to perform accurate fuzzing [J]. IEEE Transactions on Reliability, 2016, 65(3): 1180-1194.

[52] Eddington M. Peach fuzzing platform [CP/OL]. [2021-08-14]. https://community.peachfuzzer.com/WhatIsPeach.html.

[53] Xie Xiao-fei, Li Xiao-hong, Chen Xiang, et al. Hybrid testing based on symbolic execution and fuzzing[J]. Journal of Software, 2019, 30(10): 3071-3089. (in Chinese)

[54] Chen K, Feng D, Su P, et al. Black-box testing based on colorful taint analysis[J]. Science China Information Sciences, 2012, 55(1): 171-183.

[55] Li Ming-lei, Huang Hui, Lu Yu-liang, et al. SymFuzz: Vulnerability detection technology under complex path conditions [J]. Computer Science, 2021, 48(5): 25-31. (in Chinese)

[56] Godefroid P, Peleg H, Singh R. Lean&-Fuzz: Machine learning for input fuzzing[C]//Proc of the 32nd IEEE/ACM International Conference on Automated Software Engineering, 2017: 50-59.

[57] Dolan-Gavitt B, Hulin P, Kirda E, et al. LAVA: Large-scale automated vulnerability addition [C]// Proc of the IEEE Symposium on Security and Privacy, 2016: 110-121.

[58] Zou Yan-yan, Zou Wei, Yin Jia-wei, et al. Research on mutator strategy-aware parallel fuzzing[J]. Journal of Information Security, 2020, 5(5): 1-16. (in Chinese)

[59] Zhang Yi-chen, Zhao Lei, Jin Yin-shan. Sensitive region prediction based on neural network in fuzzy test algorithm re-

search[J]. Journal of Information Security,2020,5(1):10-19. (in Chinese)

[60] Tu Ling,Ma Yue,Cheng Cheng,et al. Hybrid protocol deformation based Web security fuzzy testing and utility evaluation approach[J]. Computer Science,2017,44(5):141-145. (in Chinese)

[61] Liu Yuan,Yang Yong-hui,Zhang Chun-rui,et al. A novel method for fuzzing test cases generating based on genetic algorithm[J]. Acta Electronics Sinica,2017,45(3):552-556. (in Chinese)

[62] Cheng Cheng, Zhou Yan-hui. Findding XSS vulnerability based on fuzzing test and genetic algorithm[J]. Computer Science,2016,43(Z6):328-331. (in Chinese)

[63] Lin Chuang, Su Wen-bo,Meng Kun,et al. Cloud computing security:Architecture,mechanism and modeling[J]. Chinese Journal of Computers,2013,36(9):1765-1784. (in Chinese)

附中文参考文献:

[5] 高峻,徐志大,李健. 针对复合文档的 Fuzzing 测试技术[J]. 计算机与数字工程,2008,36(12):116-119.

[13] 王凯,刘奇旭,张玉清. 基于 Fuzzing 的 Android 应用通信过程漏洞挖掘技术[J]. 中国科学院大学学报,2014,31(6):827-835.

[14] 张密,杨力,张俊伟. FuzzerAPP:Android 应用程序组件通信鲁棒性测试[J]. 计算机研究与发展,2017,54(2):338-347.

[15] 赵赛,刘昊,王雨峰,等. Android 组件间通信的模糊测试方法[J]. 计算机科学,2020,47(S2):303-309.

[16] 何远,张玉清,张光华. 基于黑盒遗传算法的 Android 驱动漏洞挖掘[J]. 计算机学报,2017,40(5):1031-1043.

[18] 张兴,冯超,雷菁,等. 一种面向模糊测试的 GUI 程序空转状态实时检测方法[J]. 软件学报,2018,29(5):1288-1302.

[20] 霍玮,戴戈,史记,等. 基于模式生成的浏览器模糊测试技术[J]. 软件学报,2018,29(5):1275-1287.

[21] 李伟明,张爱芳,刘建财,等. 网络协议的自动化模糊测试漏洞挖掘方法[J]. 计算机学报,2011,34(2):242-255.

[26] 张亚丰,洪征,吴礼发,等. 基于状态的工控协议 Fuzzing 测试技术[J]. 计算机科学,2017,44(5):132-140.

[30] 李佳莉,陈永乐,李志,等. 基于协议状态图遍历的 RTSP 协议漏洞挖掘[J]. 计算机科学,2018,45(9):171-176.

[31] 张蔚瑶,张磊,毛建瓴,等. 未知协议的逆向分析与自动化测试[J]. 计算机学报,2020,43(4):653-667.

[33] 王颖,王冰青,关永,等. 面向 ROS 的差分模糊测试方法[J]. 软件学报,2021,32(6):1867-1881.

[34] 沙乐天,肖甫,杨红柯,等. 基于自适应模糊测试的 IaaS 层漏洞挖掘方法[J]. 软件学报,2018,29(5):1303-1317.

[38] 李伟明,于俊清,艾少波. PyFuzzer:自动化高效内存模糊测试方法[J]. 通信学报,2013,34(S2):64-68.

[39] 王颖,谷利泽,杨义先,等. EWFT:基于程序执行过程的白盒测试工具[J]. 电子学报,2014,42(10):2016-2023.

[43] 乐德广,龚声蓉,吴少刚,等. RTF 数组溢出漏洞挖掘技术研究[J]. 通信学报,2017,38(5):96-107.

[46] 黄桦烽,王嘉捷,杨轶,等. 有限资源条件下的软件漏洞自动挖掘与利用[J]. 计算机研究与发展,2019,56(11):2299-2314.

[47] 杨梅芳,霍玮,邹燕燕,等. 可编程模糊测试技术[J]. 软件学报,2018,29(5):1258-1274.

[53] 谢肖飞,李晓红,陈翔,等. 基于符号执行与模糊测试的混合测试方法[J]. 软件学报,2019,30(10):3071-3089.

[55] 李明磊,黄晖,陆余良,等. SymFuzz:一种复杂路径条件下的漏洞检测技术[J]. 计算机科学,2021,48(5):25-31.

[58] 邹燕燕,邹维,尹嘉伟,等. 变异策略感知的并行模糊测试研究[J]. 信息安全学报,2020,5(5):1-16.

[59] 张羿辰,赵磊,金银山. 模糊测试中基于神经网络的敏感区域预测算法研究[J]. 信息安全学报,2020,5(1):10-19.

[60] 涂玲,马跃,程诚,等. 基于协议混合变形的 Web 安全模糊测试与效用评估方法[J]. 计算机科学,2017,44(5):141-145.

[61] 刘渊,杨永辉,张春瑞,等. 一种基于遗传算法的 Fuzzing 测试用例生成新方法[J]. 电子学报,2017,45(3):552-556.

[62] 程诚,周彦晖. 基于模糊测试和遗传算法的 XSS 漏洞挖掘[J]. 计算机科学,2016,43(Z6):328-331.

[63] 林闯, 苏文博, 孟坤, 等. 云计算安全:架构、机制与模型评价[J]. 计算机学报, 2013, 36(9): 1765-1784.

作者简介:

牛胜杰(1997 -),男,山东菏泽人,硕士生,研究方向为物联网安全和模糊测试技术。**E-mail**:1220045122@njupt.edu.cn

NIU Sheng-jie, born in 1997, MS candidate, his research interests include Internet of Things security and fuzzing technique.

李鹏(1979 -),男,福建长汀人,博士,教授,CCF 会员(48573M),研究方向为计算机通信网络、无线传感器网络和信息安全。**E-mail**:lipeng@njupt.edu.cn

LI Peng, born in 1979, PhD, professor, CCF member(48573M), his research interests include computer communication network, wireless sensor network, and information security.

张玉杰(1989 -),男,河南周口人,博士生,讲师,CCF 会员(E3619G),研究方向为计算机通信网络、无线传感器网络和信息安全。**E-mail**:zhangyujie@njupt.edu.cn

ZHANG Yu-jie, born in 1989, PhD candidate, lecturer, CCF member(E3619G), his research interests include computer communication network, wireless sensor network, and information security.