软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(2):872−898 [doi: 10.13328/j.cnki.jos.006942]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

网络协议软件漏洞挖掘技术综述*

喻 波[1], 苏金树[1], 杨 强[1], 黄见欣[1], 盛周石[1], 刘润昊[1], 卢建君[1], 梁 晨[1], 陈 晨[1], 赵 磊[2]

[1](国防科技大学 计算机学院, 湖南 长沙 410073)

[2](武汉大学 国家网络安全学院, 湖北 武汉 430072)

通信作者: 苏金树, E-mail: birchsu@139.com; 黄见欣, E-mail: jxin8585@nudt.edu.cn

摘 要: 网络协议软件部署和应用非常广泛, 在网络空间提供了诸如通信、传输、控制、管理等多样化的功能. 近年来, 其安全性逐渐受到学术界和工业界的重视, 及时发现和修补网络协议软件漏洞, 成为一项重要的课题. 网络协议软件由于部署形态多样、协议交互过程复杂、相同协议规范的多个协议实现存在功能差异等特点, 使得其漏洞挖掘技术面临诸多挑战. 首先对网络协议软件漏洞挖掘技术进行分类, 对已有关键技术的内涵进行界定. 其次, 进一步综述网络协议软件漏洞挖掘 4 个方面的技术进展, 包括网络协议描述方法、挖掘对象适配技术、模糊测试技术和基于程序分析的漏洞挖掘方法, 通过对比分析归纳不同方法的技术优势及评价维度. 最后, 总结网络协议软件漏洞挖掘的技术现状和挑战, 并提炼 5 个潜在研究方向.

关键词: 网络协议软件; 漏洞挖掘; 模糊测试; 程序分析; 符号执行

中图法分类号: TP393

中文引用格式: 喻波, 苏金树, 杨强, 黄见欣, 盛周石, 刘润昊, 卢建君, 梁晨, 陈晨, 赵磊. 网络协议软件漏洞挖掘技术综述. 软件学报, 2024, 35(2): 872–898. http://www.jos.org.cn/1000-9825/6942.htm

英文引用格式: Yu B, Su JS, Yang Q, Huang JX, Sheng ZS, Liu RH, Lu JJ, Liang C, Chen C, Zhao L. Survey on Vulnerability Mining Techniques of Network Protocol Software. Ruan Jian Xue Bao/Journal of Software, 2024, 35(2): 872–898 (in Chinese). http://www.jos.org.cn/1000-9825/6942.htm

Survey on Vulnerability Mining Techniques of Network Protocol Software

YU Bo[1], SU Jin-Shu[1], YANG Qiang[1], HUANG Jian-Xin[1], SHENG Zhou-Shi[1], LIU Run-Hao[1], LU Jian-Jun[1], LIANG Chen[1], CHEN Chen[1], ZHAO Lei[2]

[1](College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

[2](School of Cyber Science and Engineering, University of Wuhan, Wuhan 430072, China)

Abstract: The network protocol software is widely deployed and applied, and it provides diversified functions such as communication, transmission, control, and management in cyberspace. In recent years, its security has gradually attracted the attention of academia and industry. Timely finding and repairing network protocol software vulnerabilities has become an important topic. The features, such as diversified deployment methods, complex protocol interaction processes, and functional differences in multiple protocol implementations of the same protocol specification, make the vulnerability mining technique of network protocol software face many challenges. This study first classifies the vulnerability mining technologies of network protocol software and defines the connotation of existing key technologies. Secondly, this study summarizes the technical progress in four aspects of network protocol software vulnerability mining, including network protocol description method, mining object adaptation technology, fuzz testing technology, and vulnerability mining method based on program analysis. In addition, through comparative analysis, the technical advantages and evaluation dimensions of different methods are summarized. Finally, this study summarizes the technical status and challenges of network protocol software vulnerability mining and proposes five potential research directions.

Key words: network protocol software; vulnerability mining; fuzz testing; program analysis; symbolic execution

* 基金项目: 国家自然科学基金 (61902416); 湖南省自然科学基金 (2019JJ50729)
收稿时间: 2021-08-22; 修改时间: 2021-10-08, 2022-06-06, 2022-11-03, 2023-02-04; 采用时间: 2023-03-29; jos 在线出版时间: 2023-08-30
CNKI 网络首发时间: 2023-08-31

网络协议软件, 通常是指在网络层和应用层提供网络服务功能, 部署于通用操作系统并具有独立运行形态的网络服务程序; 但随着分布式系统和物联网技术的发展, 网络协议软件越来越多地以专用软件组件的形式出现, 通过嵌入到其他应用软件、嵌入式系统和物联网固件当中, 构成网络服务驱动的应用系统的一部分. 网络协议的类型多样, 相应软件实现的种类繁多, 相关的漏洞也频繁出现. 由于网络协议软件运行时对外暴露接口、协议交互过程复杂、在网络上部署量大、漏洞成因隐蔽等方面的特点, 漏洞类型及带来的危害也更广泛. 网络协议软件漏洞的产生主要源于对网络协议规范的理解偏差、开发人员的不正确编码、不同厂商的个性化功能开发等方面. 网络协议软件漏洞会对部署的网络应用及其运维企业产生重大影响, 及时检测修复网络协议软件漏洞, 降低其造成的损失, 提升网络协议软件的安全性和健壮性, 是网络空间安全的重要研究课题.

近年来, 网络协议软件的漏洞挖掘技术得到了国内外学者的广泛关注. 从挖掘对象的角度, 有不少工作研究通用操作系统环境下的网络协议软件漏洞挖掘, 也有针对专用网络协议组件的漏洞挖掘工作, 如 FIRM-AFL[1]针对物联网固件网络协议软件进行模糊测试. 从漏洞类型的角度, 由于网络协议软件的漏洞类型多种多样, 除了程序崩溃类漏洞, 也有不少工作致力于解决非程序崩溃类的漏洞挖掘, 如隐蔽性更高的信息泄露类[2]、认证绕过类[3]、状态混淆类[4]等. 从漏洞挖掘技术与方法的角度, 黑盒测试和灰盒测试等模糊测试是常用的网络协议软件漏洞挖掘方法, 也有不少学者尝试引入静态分析、符号执行、污点分析等程序分析技术, 辅助模糊测试生成更有针对性的测试用例, 提高测试效率和漏洞发现能力. 以外, 随着机器学习技术的发展, 部分研究工作也通过引入强化学习等智能化方法来进行网络协议软件的漏洞挖掘. 综合来说, 网络协议软件漏洞挖掘的研究目标是: 基于模糊测试技术, 以及符号执行和污点分析等软件分析方法, 机器学习等智能化技术, 构建针对通用网络服务软件、物联网固件网络协议组件、嵌入式系统网络协议组件等不同类型目标的漏洞挖掘框架, 实现对程序崩溃、命令执行、信息泄露、认证绕过等多类型漏洞的挖掘.

现有网络协议软件漏洞挖掘技术相关的综述工作主要有两个, 文献 [5] 针对工业控制系统的安全问题和来源进行分析总结, 对工业控制领域现有漏洞挖掘技术进行归纳和梳理. 但该工作主要聚焦于工控协议模糊测试技术. 不涉及本文研究的网络协议描述方法、挖掘对象适配技术、基于程序分析的网络协议软件漏洞方法等内容. 文献 [2] 主要分析网络协议的脆弱性类型, 从明文通信、协议消息解析错误、协议状态处理错误、抗重放攻击、协议认证等方面对网络协议软件的脆弱性进行分析. 该文献更多地关注软件缺陷和脆弱性的分类, 与本文的研究对象和关键技术不重叠. 此外, 文献 [6] 综述了网络协议逆向工具的研究现状和技术进展, 与本文研究的网络协议软件漏洞挖掘有较大差异.

为了对网络协议软件漏洞挖掘技术的相关进展和潜在问题进行研究, 本文对现有的漏洞挖掘技术进行系统地分析、总结和比较, 结合作者在网络协议智能模糊测试技术、混合模糊测试、符号化分析技术方面的成果积累[7−10], 通过对国内外相关研究工作进行综述, 剖析了现有技术的能力现状和不足, 进一步展望网络协议软件漏洞挖掘的技术发展和潜在方向. 具体来说, 首先介绍了网络协议软件漏洞挖掘的技术分类; 其次, 从网络协议描述方法、挖掘对象适配技术、基于模糊测试的网络协议软件漏洞挖掘方法、基于程序分析的网络协议软件漏洞挖掘方法等方面, 对现有的研究工作进行系统地分析、总结和比较. 我们在 IEEE, ACM, Springer, Elsevier 和 CNKI 等论文数据库中进行检索, 检索时采用的主要英文关键词包括“network protocol”“network protocol fuzzing”“firmware vulnerability mining”“network protocol vulnerability mining”等; 然后, 对检索出的论文, 聚焦于与研究问题相关的论文, 并通过查阅相关的参考文献和相关研究人员发表的论文列表来进一步识别出遗漏的论文; 最终, 我们选择出与该研究问题直接相关的高质量论文共 104 篇. 从选择出的论文所发表的会议和期刊来看, 绝大部分论文发表在系统安全和软件工程领域的权威会议或期刊上, 例如网络与系统安全、软件工程、人工智能领域国际会议、ACM/IEEE 国际期刊、国内学报等重要期刊.

本文第 1 节介绍网络协议软件漏洞挖掘的技术分类. 第 2 节介绍面向漏洞挖掘的网络协议描述方法, 包括网络协议语法抽取、网络协议描述模型的构建. 第 3 节综述网络协议软件漏洞挖掘的挖掘对象适配技术, 包括异常发现技术、面向专用系统的支撑技术和面向专有协议的支撑技术. 第 4 节介绍基于模糊测试的网络协议软件漏洞挖掘技术, 主要包括模糊测试的用例生成和模糊测试的过程调度. 第 5 节对基于程序分析的网络协议软件漏洞挖

掘方法进行分析. 第 6 节讨论了网络协议软件漏洞挖掘的技术不足, 提出了 4 个方面的技术挑战, 并对未来值得关注的研究方向进行展望. 最后第 7 节总结全文.

1 网络协议软件漏洞挖掘的技术分类

网络协议软件漏洞挖掘的技术发展经历了多个阶段, 从经典的黑盒模糊测试技术到基于执行路径反馈机制的灰盒模糊测试技术阶段, 目前正朝着基于模糊测试和程序分析技术相结合的混合测试阶段在发展. 黑盒模糊测试技术是最早引入用于网络协议软件的漏洞挖掘, 流行的模糊测试工具有 Boofuzz[11], 以及工业界的 Peach[12]和 beSTORM[13]. 黑盒模糊测试需要采用网络协议消息格式描述功能, 设计测试用例生成策略, 通过将测试用例注入给目标网络协议软件来检测异常. 随着智能有反馈模糊测试技术的发展, 网络协议软件的模糊测试也引入了有反馈机制, 这阶段的漏洞挖掘技术通过构建通用网络协议软件和专用网络协议软件组件的插桩分析环境, 设计基于信息反馈的测试用例生成机制. 与黑盒模糊测试不同, 有反馈的模糊测试通过捕获网络协议软件的执行轨迹等信息, 作为测试用例有效性的反馈, 从而激励生成更加有用的测试用例, 以提升模糊测试的效率. 目前主要技术路线有: 一是修改 AFL[14]以支持网络协议交互特性, 二是研究通用网络协议软件和物联网固件的插桩技术, 以适配 AFL[14]这一经典的有反馈模糊测试引擎. 这一阶段典型的代表有 AFLNet[15]、FIRM-AFL[1]等.

近年来, 也有不少学者致力于结合灰盒模糊测试与程序分析的各自优势, 甚至引入人工智能技术, 实现网络协议软件的智能漏洞挖掘, 网络协议软件漏洞挖掘技术进入第 3 阶段. 通过引入动态污点分析和符号执行等技术, 提供智能化测试用例生成、自动化异常发现等漏洞挖掘功能. 第 3 阶段的网络协议软件漏洞挖掘技术, 其主要特点是基于测试用例生成的模糊测试技术与面向软件程序内部路径追踪的符号执行技术相结合, 形成多种技术相结合的混合模糊测试框架.

针对网络协议软件开展漏洞挖掘, 网络协议描述方法、模糊测试技术、符号执行技术和污点分析等都发挥了各自的作用. 这些技术分别从不同的角度提供网络协议软件的漏洞挖掘能力. 对于网络协议描述方法, 网络协议软件的漏洞挖掘首先是针对网络协议进行协议描述, 获得网络协议消息构成、会话序列、会话依赖等协议要素. 网络协议描述的作用, 一是形成理解协议交互过程的协议知识模型和机器可读的网络协议格式表示, 直接用于模糊测试过程; 二是提供一个生成测试用例的模板. 网络协议描述方法通过构建与挖掘对象相适配的网络协议描述模型, 提供与挖掘对象交互的数据表示模板.

此外, 污点分析技术能够通过跟踪程序内部的运行状态, 尤其是程序输入与程序运行状态的污点关系, 来提升测试用例生成的针对性. 而符号执行技术通过跟踪和收集程序分支的约束, 并进行求解计算, 直接作用于测试用例的生成. 这些程序分析技术不但在终端命令行输入软件漏洞挖掘方面发挥重要的作用, 也是网络协议软件漏洞挖掘的主要技术. 如 Polar[16]提出一种代码感知框架, 在不获取协议数据包格式规范的情况下, 通过动态污点分析记录污染源在数据包中的偏移位置识别关键字段, 指导模糊测试程序进行有效的测试用例变异. 如 Wen 等人[17]采用基于模型的符号执行方法, 收集协议报文的格式结构和通信交互约束的逻辑状态, 进而利用这些知识来进行混合模糊测试. 通过对协议状态的建模, 并根据报文格式和符号化约束来构造新的输入, 利用协议模型指导符号执行, 提高程序路径的覆盖率.

然而, 网络协议软件漏洞挖掘过程是与挖掘对象相关的, 需要与挖掘对象进行适配, 具体表现在: 一是网络协议软件的漏洞挖掘, 需要构建网络协议的描述模型. 这是因为网络协议软件的正常运行, 需要通信双方对网络协议消息格式和状态机保持一致. 模糊测试过程中, 如果缺乏网络协议规范相关的知识, 会生成大量无效的测试用例, 如产生不正确的协议字段校验和, 其测试过程就容易中断. 而且, 由于网络协议软件存在交互特性, 需要漏洞挖掘程序同时充当网络协议主体的角色, 实现协议交互过程的逻辑代码. 第二个方面是网络协议软件的漏洞挖掘, 要有配套的支撑技术, 需要具备通用网络协议软件和专用协议组件的支撑能力.

因此, 综合上述分析, 本文将相关工作归纳为以下 4 类: 协议描述方法, 模糊测试技术, 程序分析技术和挖掘对象适配技术. 表 1 介绍了这 4 个方面的代表工作.

表 1 网络协议软件漏洞挖掘技术的代表工作

漏洞挖掘代表工作	协议描述方法	模糊测试技术		程序分析技术			挖掘对象适配技术	
		黑盒模糊测试	灰盒模糊测试	动态符号执行	动态污点分析	程序静态分析	适配通用协议软件	适配专用协议组件
Boofuzz[11]	√	√	—	—	—	—	√	—
IxFIZZ[18]	√	√	—	—	—	—	—	√
AFLNet[15]	√	—	√	—	—	—	√	—
FIRM-AFL[1]	—	—	√	—	—	—	—	√
Wen等人[17]	—	—	—	√	—	—	—	—
BitFuzz[19]	—	—	√	√	—	—	—	√
Polar[16]	√	√	—	—	√	—	—	√
SwordDTA_NT[20]	√	—	√	—	√	—	√	—
Autogram[21]	√	—	—	—	√	—	√	—
DRLgencert[22]	—	—	√	—	—	—	√	—
SPFuzz[8]	√	—	√	—	—	—	√	—
IoTHunter[9]	—	—	√	—	—	—	—	√
Shastry等人[23]	√	√	—	—	—	√	√	—
Tang等人[24]	—	—	—	—	—	√	—	√
Dacosta等人[25]	√	√	—	—	—	√	—	√

在表 1 所述的代表工作中, 黑盒模糊测试技术一般需要与网络协议描述方法配合, 程序分析方法可辅助模糊测试进行测试用例生成. 其中, 网络协议描述部分解决的是网络协议语法知识的来源和模型表示问题; 挖掘对象适配部分主要解决挖掘对象的多种适配问题, 包含: 网络协议软件的异常发现技术、针对专用系统的支撑技术、针对专有协议的支撑技术. 针对专用网络软件组件的漏洞挖掘, 需要支持嵌入式系统、物联网固件等专用系统的仿真和插桩分析环境; 漏洞挖掘的核心部分包括以模糊测试为主的测试挖掘方法和以程序分析为主的分析挖掘技术. 模糊测试以测试用例生成和测试过程调度为主, 分别解决如何生成测试用例和使用何种策略生成测试用例这两个基本问题. 分析挖掘技术是以被挖掘对象为目标, 采用静态分析、污点分析、符号执行等技术手段, 通过分析和获取挖掘对象的动静态信息, 来实现辅助测试用例生成、漏洞识别、漏洞定位等目标.

图 1 表示了网络协议描述、挖掘对象适配和分析测试等环节的多项关键技术及其关联关系, 图中用不同的图形来表示关键技术, 用箭头来表示关键技术之间的相互支撑关系, 其中单箭头表示支撑组件对模糊测试和动态分析挖掘的支撑关系, 双箭头表示模糊测试挖掘和动态分析挖掘之间的相互支撑关系. 模糊测试技术和符号执行技术都可以独立完成网络协议软件的漏洞挖掘, 也可以相互支撑形成混合测试框架, 不同在于模糊测试技术是通过基于策略生成测试用例来发现新路径, 并发现软件崩溃等漏洞; 而符号执行技术是跟踪程序内部并发现新的路径来实现生成新的测试用例. 此外, 基于符号执行提取的程序路径语义, 以及异常发现技术都能够辅助发现程序崩溃之外更多的漏洞类型. 而针对网络协议的描述方法是通过收集协议软件语法格式及相互关系, 来支撑网络协议软件模糊测试, 而污点分析技术主要是通过构造程序执行路径上数据流之间的约束关系, 辅助进行测试用例有效性的筛选, 减少无效的测试用例生成. 该图中也用数字标识了相关技术的综述在本文中所在的章节.

与文献 [26] 中所阐述的命令行输入软件漏洞挖掘框架不同, 网络协议软件漏洞的协议描述方法、挖掘对象适配技术等部分都是网络协议软件漏洞挖掘所独有的. 其中网络协议描述部分主要是以协议理解为目标. 挖掘对象适配技术部分是以功能支撑为目标, 解决的是挖掘对象适配和漏洞类型多样且成因复杂这一挑战; 分析测试部分以漏洞挖掘分析为目标, 主要解决的漏洞类型多样且成因复杂, 以及测试智能化自动化程度低这一目标.

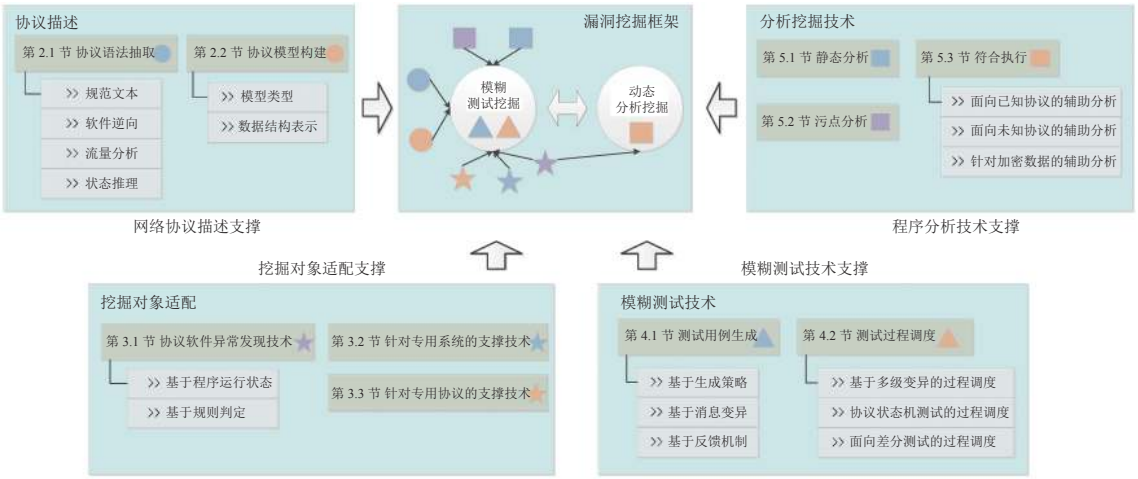


图 1 网络协议软件漏洞挖掘的技术分类及关系

2 面向漏洞挖掘的网络协议描述方法

网络协议描述是通过对网络协议知识的抽取, 构建网络协议描述模型, 设计网络协议描述的数据结构表示, 形成机器可读的协议描述信息. 网络协议描述模型的构建, 是对网络协议消息格式、消息序列、会话过程的抽象表示, 并形成机器可读的描述文本.

2.1 网络协议语法抽取

语法知识抽取主要包括基于规范文本的协议语法抽取、基于软件逆向的协议语法抽取、基于流量分析的协议语法抽取和基于响应报文推理的状态机结构抽取这 4 类. 在抽取过程中, 需要设计抽取方法或算法实现准确地抽取协议语法信息, 来源包括协议规范、目标二进制软件、网络流量和测试过程中的响应报文等方面, 抽取内容包括协议字段、字段类型、状态构成等信息. 如表 2 所示描述了这 4 类相关工作的比较.

表 2 网络协议语法抽取相关工作比较

方法	文献	语法来源	抽取方法/算法	抽取内容	语法描述模型	针对协议目标	关键结论
基于规范文本进行抽取	[27]	RFC规范文档	基于NLP的零次学习算法	协议语法格式	领域知识模型	GRE, IPv6, IP, TCP, DCCP, SCTP	实现了人工抽取同等的有效性, 且提高了基于语法的模糊测试的效率实现了对人工抽取语法规则的替代
	[18]	RFC规范文档	人工描述及SPIN校验	状态机和消息格式	Promela描述语言	SIP及其实现Linphone	基于模型描述的协议语法可支持模糊测试检测到更多的错误
	[28]	RFC规范文档	人工描述	状态机和消息格式	增强的BNF	IPv4, SIP	基于规范语法描述的模糊测试可有效发现网络协议漏洞
基于软件逆向进行抽取	[21]	Java程序	动态污点	协议语法	上下文无关语法	HTTP URL, JSON, CSV, INI	可有效根据程序输入提取语法格式
	[29]	软件二进制程序	基于中间语言的符号执行TEMU和BAP	字段、语法和约束等协议信息	语法结构树	DNS, eDonkey, FTP, HTTP, McAfee ePO	针对5种协议的测试结果表明这个协议语法方法可达到95.3%的字段识别率, 识别了所有静态字段, 比现有方法准确度高
	[30]	软件二进制程序	构建内存传播树	字段边界、类型等语法格式	自定义的数据模型	Modbus/TCP, IEC, Siemens S7等工控协议	可有效抽取网络协议数据模型, 并在4款开源协议模糊测试工具上验证了其有效性

表 2 网络协议语法抽取相关工作比较 (续)

方法	文献	语法来源	抽取方法/算法	抽取内容	语法描述模型	针对协议目标	关键结论
基于流量分析进行抽取	[31]	Pcap文件	自定义提取代码	字段及其取值范围等信息	PRDF协议和关系描述	FTP, PN-DCP两种协议的4个目标软件	可有效支持多字段关联识别, 支撑对真实协议软件进行漏洞检测, 发现了4个漏洞
	[32]	Pcap文件	多序列比对方法	字节簇、可变域	SPIKE脚本描述	FTP, TNS, EM, ISQLPIus等	漏洞挖掘效果超过手工分析, 提升了测试效率
	[33]	流量包	自定义提取转换	关键字、字段边界等信息	JSON模板	物联网MQTT协议	可支持模糊测试发现多个安全漏洞
	[34]	流量包	流量类型识别和分词器	字段和字段边界	LZ表	Mt-daapd和Net-SNMP共两种SCADA协议	适合于测试未知格式的专有协议, 而且比现有的随机变异器高效
	[35]	Pymodbus流量生成器	基于GAN算法训练	字段及报文结构	基于GAN训练的语法模型	工业网络协议Modus-TCP 的4种协议软件	可支撑提升代码覆盖率和测试深度, 可适用于已知协议和未知协议
基于响应报文推理状态机	[36]	测试响应报文	基于LearnLib的状态学习算法	输入、迁移和节点构成的状态机	米利型有限状态机	OpenVPN和OpenVPN-NL实现	可有效检测状态迁移过程中的逻辑错误
	[37]	测试响应报文	基于模型的协议状态机重构	迁移和节点构成的状态机	状态迁移图	EMV协议	基于测试的方法从协议交互中提取状态机, 可有效发现状态错误
	[38]	协议规范及测试响应报文	专用的消息生成组件	消息字段及关联规则	XML结构化描述	SIP协议	可有效支持协议扩展, 支持有状态协议测试
	[39]	测试响应报文	协议状态测试	协议状态机	状态迁移图	TLS协议及其实现	可支持9种协议实现的状态机提取与分析

方法的优势分析: 基于规范文本的协议语法抽取过程, 针对性强, 消息格式规范清晰, 产生有效报文的比率高, 但只适用于可以获取协议规范或者有源代码的场景, 而且需要使用者对目标协议有着较深入的理解, 使用门槛比较高. 基于软件逆向的协议语法抽取工作较好地解决了基于二进制软件获取协议语法的问题, 能够实现语法信息提取、格式识别、字段边界识别、字段类型识别的等, 形成完善的协议语法模型. 网络流量为网络协议格式分析和知识抽取提供了一种独特的途径. 通过收集和分析网络流量, 能够学习到网络协议消息格式和状态机, 如字段边界、字段类型、会话序列及状态机等. 如文献 [35] 等方法采用遗传算法、序列比对、聚类方法等对网络流量进行分析, 可有效获取网络协议语法知识. 协议状态机推理是网络协议软件漏洞挖掘中一种独有测试技术, 目标是获取网络协议软件的状态机构成, 为后续协议状态的测试提供知识.

评价维度及分析: 主要从抽取方法和算法的准确性以及语法描述方法的能力两个方面来评价. 语法描述方法, 如文献 [18,28] 采用的 Promela 和 BNF 模型, 具有较强的协议语法和状态迁移表示能力. 抽取算法, 如引入自然语言处理方法[27]学习协议规范、多序列比对算法[32]学习网络流量, 可自动抽取网络协议描述所需的字段、边界、关联等协议要素, 提升协议语法抽取的准确度, 降低人工参与的工作量和成本.

2.2 网络协议描述模型的构建

网络协议描述模型是对网络协议消息和状态机知识的抽象表示, 需要支持协议消息类型、字段、消息序列以及状态机等要素. 网络协议描述模型一方面能够用于设计测试用例生成策略, 另外一方面又能够指导模糊测试的调度过程设计. 协议描述模型包括设计的描述模型和数据结构, 以及描述要素的范围, 如表 3 所示描述了这两类相关工作[8,28,40–46]的比较.

方法的优势分析: 网络协议描述模型提供了消息格式的抽象表示, 其主要作用是指导模糊测试的测试过程设计. 常用的网络协议描述模型类型, 有马尔可夫模型、BNF、ASN.1 等模型. 也有部分工作采用自定义的元模型, 如基于规则的状态机模型等. 文献 [41] 建立了描述网络协议的通用模型, 根据网络协议规范定义了消息规范、状

态模型和模型拓展, 可使用该模型导出任意格式的网络协议描述. 对于网络协议状态机的描述, 如文献 [42] 提出一种基于规则的状态机表示, 通过构造一个基于规则的状态机模型, 来形式化描述网络协议状态. 这些描述模型为网络协议的模糊测试技术等提供了重要的支撑, 即使在 AFLNet 这类智能有反馈的模糊测试系统当中, 也需要定义协议的字段构成和状态迁移知识. 网络协议描述的数据结构表示, 是为了能够存储和提供机器可读的协议表示. 常用的网络协议数据结构表示方法, 包含 XML 结构、JSON 结构等. 文献 [46] 设计并实现了通用消息树的概念, 是一种通用的动态数据结构, 用于高效地操作高度结构化的协议消息. 该方法基于模糊化的 TLS 消息生成和通用消息树的方法, 提供了一种生成高度多样化且大部分有效的 TLS 握手消息的随机算法. 网络协议描述的数据结构, 既有适合于计算过程中的内存表示, 也有适合于持久化存储的文件格式表示.

表 3　网络协议描述模型相关工作比较

方法	文献	语法来源	描述模型/结构	描述要素	针对协议	关键结论
描述模型设计	[40]	采集的网络流量	隐马尔可夫模型	关键字和数据字段	HTTP, SSDP, BitTorrent, QQ, DNS, NetBIOS	基于HMM模型的协议描述可有效支持入侵检测和异常检测, 消息类型推理可达到95%以上的准确率
	[28]	RFC规范文档	增强的BNF	状态机和消息格式	IPv4, SIP	基于规范语法描述的模糊测试可有效发现网络协议漏洞
	[8]	RPC规范文档	Boofuzz描述脚本	消息字段及依赖, 状态迁移	FTP, SSL	基于描述和AFL有反馈引擎相结合可有效提升覆盖率和触发漏洞
	[41]	RFC规范文档	自定义的通用元模型	消息规范、状态模型和模型拓展	DCP, ENP	提出的通用元模型可用于描述网络协议, 支持对真实IACS设备进行有效的安全测试
	[42]	RFC规范文档	基于规则的状态机模型	协议的状态规则树、消息格式	SIP	可以生成覆盖状态和状态转换轨迹的测试数据
	[43]	RFC规范文档	SFTCG模型	协议类型、端口号、属性和约束关系	ICMP	提出的模型可有效支持路由器的漏洞挖掘
数据结构设计	[44]	RFC规范文档	SPIKE数据结构	消息长度和块定义	HTTP	支持多种网络协议的结构化描述, 允许快速创建网络协议测试
	[45]	RFC规范文档	XML结构化描述	消息序列	6LowPAN	提出了基于结构化消息描述的测试套件, 支持6LoWPAN协议
	[46]	RFC规范文档	通用消息树	字段及其类型等属性信息	SSL	设计并实现了通用消息树GMTs, 可高效地操作结构化的协议消息

评价维度及分析: 从模型的描述能力和准确度两个方面进行评价, 模型描述能力涉及可描述的字段、属性及字段关系等. 准确度涉及模型表示的精确度, 如隐马尔可夫模型针对网络流量的关键字和数据字段进行抽取, 可达到 95.3% 的准确率[29]. 而 XML 等结构化表示、通用消息树等关联关系表示可实现网络协议字段结构及关联关系的有效表示.

3　漏洞挖掘的挖掘对象适配技术

网络协议由于存在协议通信交互过程, 而部署于嵌入式系统和物联网固件中的专用网络协议组件, 由于其独特的运行形态和部署环境, 都需要相应的适配技术来支撑, 以适配黑盒模糊测试、灰盒模糊测试、符号执行和污点分析等. 这是网络协议软件漏洞挖掘的特殊性所在. 如在图 2 所示的典型网络协议灰盒模糊测试场景中, 需要为挖掘对象这一被测目标提供插桩、通信交互和异常发现 3 大功能. 异常发现能力直接影响模糊测试的效果和能力, 对专用系统和专有协议的支撑能力又影响模糊测试的适用性.

具体来说, 挖掘对象适配技术的类型包括几个方面: (1) 异常发现技术. 网络协议软件, 尤其是嵌入到其他系统中的专用网络协议组件, 其运行状态和异常难以捕获. 如嵌入式系统和物联网固件中的程序发生崩溃, 需要基于全系统模拟环境并通过插桩技术才能发现; 对于非内存破坏类和逻辑类漏洞, 需要采用定制的漏洞规则才能发现.

(2) 专用协议支撑技术. 部分加密协议、专用通信协议, 由于存在会话过程加密、依赖底层通信硬件等特征, 导致现有的漏洞挖掘技术无法应用于被测协议, 需要引入相关的支撑技术在挖掘对象与测试工具之间进行适配. (3) 专用系统的支撑技术. 针对嵌入式系统、物联网设备固件等复杂系统上的网络协议服务程序和协议栈进行测试, 存在运行状态难以获取、测试效率低等问题, 需要真实设备或仿真系统进行适配.

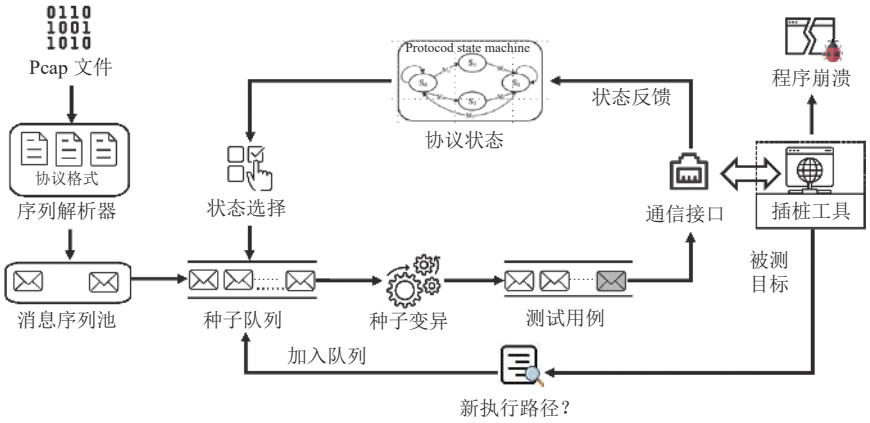


图 2 网络协议软件灰盒模糊测试的典型过程

3.1 典型网络协议灰盒模糊测试流程网络协议软件的异常发现技术

对网络协议软件状态异常进行检测发现, 是漏洞挖掘的重要环节之一, 解决的是怎么知道挖掘对象发生了异常这个问题. 主要有两种技术: 基于程序运行状态的异常发现和基于规则判定的异常发现. 基于程序状态的异常发现主要利用操作系统机制或进程状态检测工具, 对程序运行状态进行实时监测, 记录发现的异常状态. 基于规则判定的异常发现, 则主要通过信息收集和分析等方法, 利用知识和规则去检测协议软件的状态. 前者解决了在操作系统层面进行状态判定的情况下发现异常, 后者解决了需要通过制定规则来进行间接判定的情况, 异常发现能力取决于所采用的判断模型和支持的异常类型. 表 4 对这两类异常发现技术的相关工作对比分析.

表 4 网络协议软件异常发现相关工作比较

方法	文献	针对协议	判定对象	所用模型	针对异常类型	关键结论
基于程序运行状态	[47]	针对部署IKE软件的硬件设备和应用软件	基于ping和telnet连接, 以及GDB调试器	自定义的监控和记录模型	目标崩溃	可有效支持硬件设备和Windows/Linux平台IKE协议软件
	[17]	Windows和Linux平台协议软件	基于程序异常和CPU状态	基于内核插桩的异常模型	未处理的崩溃	基于FTP漏洞数据集验证表明检测率提高到75%
	[48]	Parrot Bebop嵌入式平台的WiFi无人机	基于CPU和内存使用率	基于CPU和内存数据的异常行为模型	缓冲区溢出、DoS、ARP缓存投毒	可有效支持发现无人机通信协议的3类漏洞
	[49]	DICOM 协议	基于捕获网络流量	Wireshark监控配合人工分析	目标崩溃	可有效获取目标崩溃异常及数据包内容
	[43]	路由器 ICMP协议	基于SNMP监控和CPU利用率	自定义的状态检测模型	DoS漏洞、异常关机	可有效发现大量ICMP导致的DoS漏洞
	[50]	防火墙 IKEv2协议	基于ping连接	基于响应状态的异常模型	DoS、缓冲区溢出等异常	可有效发现IKE新协议3类交换类型种的漏洞
	[51]	有状态网络协议	基于网络协议程序状态和能够到达目标状态的网络消息链	基于网络协议程序状态的检测模型	目标崩溃、堆栈缓冲区溢出等漏洞	可有效获取目标程序崩溃异常, 挖掘网络协议程序漏洞, 提高模糊测试的效率

表 4 网络协议软件异常发现相关工作比较 (续)

方法	文献	针对协议	判定对象	所用模型	针对异常类型	关键结论
基于规则判定	[52]	DVMA等4种Web应用	JavaScript执行过程中动态事件	基于污点分析构建的调用图异常检测规则	基于DOM的XSS漏洞	可有效检测基于JavaScript的多类型Web应用XSS漏洞
	[53]	2G、3G和4G协议及基带固件	收集设备运行状态	基于固件运行状态的异常判定规则	崩溃、重启等异常	可有效检测基带固件测试过程中的异常
	[54]	安卓设备固件的升级协议	目标处理过程中的记录的API数据	基于API数据自定义分析器和分析规则	程序异常	可有效发现程序异常, 但需要目标程序的API知识
	[18]	SIP及其实现Linphone	记录协议状态及其迁移	针对协议规范的分析比对	协议状态偏离等异常	可动态生成会话状态机及发现状态异常, 但需要详细的协议规范支持
	[55]	专业无线协议	调制解调信号并将信号转成文本	分析协议的通信交互逻辑	协议程序异常	可有效支持无线协议异常发现
	[56]	私有ICS协议ILC 171和ILC150	输出端信号	对比输出信号同预期的偏差	感知程序状态异常	能够监测输出并发现异常行为
	[4]	MSN协议	收集协议状态信息并构建状态机模型FSM	基于FSM模型进行转状态机比对	程序崩溃类, 状态混淆类	测试模型可达到较好的消息类型覆盖, 可发现非预期的安全缺陷
	[57]	RTSP协议	响应报文	响应报文与RFC规范对比	非程序崩溃类漏洞	可针对专有网络协议组件的漏洞挖掘

方法的优势分析: 模糊测试过程中的异常发现, 主要是利用操作系统的进程和线程异常处理机制, 来判定目标进程是否发生了程序崩溃. 基于操作系统的机制能够快速地判定目标软件的运行状态. 其他常见的机制包括基于挂载调试器的异常判定机制, 基于 ping 响应的异常判定机制等. 除此之外, 也有部分工作采用自定义的机制来捕获程序运行异常. 但这类机制的适用性限定于操作系统层面观察到的事件和数据. 基于规则判定的异常发现机制, 主要是在不能够直接监控到目标运行状态的条件下, 采用间接的方式进行异常发现. 该方法需要采集挖掘对象的运行数据并基于规则来进行异常判定. 该方法虽然需要依靠人工定义知识规则, 但是在网络协议的多类型漏洞挖掘当中, 却有很好的必要性和适用性.

评价维度及分析: 主要是从判定对象和方法的适用性、判定模型的准确性和支持的异常类型 3 个方面进行评价. 基于操作系统信号及目标状态数据[17,43]进行判定是准确性较高的技术手段, 但该方法仅依赖表层现象而导致精度和漏洞类型支持不够, 而基于运行状态规则、响应报文对比规则、程序路径属性对比规则等方法, 虽然技术难度较高, 能支持更多的异常类型, 是具有更好的适用性.

3.2 面向专用系统的支撑技术

为了解决嵌入式系统、物联网设备固件等复杂目标系统上的漏洞挖掘, 需要部署真实设备或采用模拟器的方式, 来适配模糊测试等漏洞挖掘分析技术. 近年来, 国内外许多学者开展这方面的研究. 其主要方法有: 系统模拟与仿真技术、硬件设备的适配技术、嵌入式设备专用支撑组件等. 如表 5 所示描述了这 3 类相关工作的比较.

方法的优势分析: 基于系统模拟和固件仿真等机制, 已经实现了对嵌入式系统和物联网固件的漏洞挖掘. 针对系统模拟与仿真, 自动化工具 Firmadyne[58]存在固件文件系统启动成功率较低的问题, 而 FirmAE[59]采用一种被称为“仲裁模拟”的新技术, 对固件初始化过程进行干预, 通过多次自动模拟和启动过程分析, 创建能够正常启动固件的动态分析环境. FirmAE 可以模拟许多 Firmadyne 未能模拟的固件, 并有效地辅助发现漏洞. 此外, FIRM-AFL[1]是一个针对物联网固件的高吞吐量有反馈模糊测试系统, 采用用户态和系统态混合的“增强过程模拟”技术, 旨在通过结合系统态模拟的真实性和用户态模拟的高效性, 被测程序大部分时间会在用户态中运行以提升效率, 只有遇到内存缺页和系统调用时, 才会迁移到系统态来执行. 在“增强过程模拟”技术的支持下, FIRM-AFL 可以实现对物联网固件进行智能模糊测试. 但是, 这些工作依然存在不足, 仿真成功率不高, 且有些仿真方法只针对特定系统, 没有普适性. 此外, FIRM-AFL[1]虽然引入了 AFL 引擎生成网络协议消息的测试用例, 但是该工作实验验证使用的是 HTTP

文本协议, 由于是文本协议, 那么 AFL 的 bit 相关的变异策略使用 AFL 原生的变异策略而不进行优化, 显然会产生大量无效的测试用例. 因此, 面向专用系统的支撑技术, 亟待从固件仿真和真实设备适配方面进行深入研究.

表 5　专用系统支撑技术相关工作比较

方法	文献	针对对象	所用方法	支撑类型	关键结论
系统模拟与仿真	[58]	物联网固件	专用的Linux内核, 及Qemu模拟器	固件提取与仿真	支持物联网固件的漏洞挖掘, 但存在固件启动成功率较低的问题
	[59]	物联网固件	采用仲裁模拟对固件初始化过程进行学习和挑战	固件仿真	可以模拟许多Firmadyne未能模拟的固件映像, 并有效地辅助发现漏洞
	[1]	物联网固件	采用用户态和系统态混合的“增强过程模拟”技术	固件仿真, 测试过程加速	可用于提升模糊测试的速度
	[49]	DICOM开源库	模拟PACS系统	环境模拟	可支持DICOM开源库的自动测试、异常监控
	[60]	嵌入式固件, 多种通用操作系统	提供X86、ARM等多种架构的模拟执行	模拟执行与监控	提供跨平台的调试能力和多种层次的监控方法, 有效支持目标测试
嵌入式系统硬件设备适配	[61]	嵌入式系统	为嵌入式系统开发了特定类型的监控	设备监控	可以评估智能电表和其他网络化嵌入式系统的安全性
	[53]	蜂窝基站和目标设备	可扩展的基带测试框架	基带协议的反馈监控	可以自动测试基带处理器运行过程中的缺陷
	[62,63]	嵌入式设备	协调仿真器和真实硬件的执行	设备动态分析	可以对嵌入式设备进行分析, 实现不同分析框架、调试器、仿真器和真实物理设备之间的互操作性
	[64]	嵌入式设备	基于GDB断点机制来实现动态插桩	动态插桩	能够支持对嵌入式固件环境中的网络协议软件进行模糊测试
专用组件支撑	[65]	移动应用程序	通过识别和重用移动应用程序特定的逻辑	测试上下文和专用通信	支撑物联网固件协议的模糊测试
	[66]	4G/LTE的移动设备	定制适配移动4G设备的专用组件	专用通信	支持从进程故障中恢复系统, 实现测试自动化执行
	[67]	LTE协议	设计数据变异组件	测试用例生成	支持对LTE协议进行测试

评价维度及分析: 主要从适配对象的范围、支撑方法的有效性和支持的漏洞类型方面进行评估. 目前现有研究多以针对特性设备类型, 研究仿真、异常监控等针对性方法, 针对设备固件和真实设备等缺乏通用的适配支撑方法. 此外, 与传统模糊测试目标相比, 嵌入式系统模糊测试通常面临在物理上与被挖掘对象分离的挑战. 需要从执行反馈、异常发现等角度, 提高支撑方法的有效性和扩展支持的异常类型.

3.3　面向专有协议的支撑技术

一些专有协议存在有状态、长时间执行、会话过程加密等特征, 导致现有的漏洞挖掘技术无法运用于被测协议. 需要引入相关的支撑技术, 以适配被测协议与现有测试分析技术. 现有的面向专有协议的支撑技术, 包括构建支持被测协议的支持框架、采用代理专用组件等进行适配等. 表 6 描述了这两类相关工作[33,45,47,50,56,68–78]的比较.

方法的优势分析: 为了支持专用协议的漏洞挖掘, 解决专有协议的语法提取和自动模糊测试等问题. 部分工作提出了适配专有协议的支持框架. 如针对 SSL 的 ESPIKE 框架[69]和 Chandrasekar[70]、针对 IKE 协议的 IKEPro-Fuzzer[47]等, 其中 ESPIKE 框架是对 SPIKE 模糊测试工具的扩展, 以支持对 SSL 加密协议的模糊测试. ESPIKE 则可以在 OpenSSL 服务器与 SPIKE 之间进行 SSL 加密通信, 从而达到对加密协议进行模糊测试的目的, 较好地解决了 SSL 这类安全协议难以进行有效测试的问题. 还有一类针对工业控制系统协议的专用测试框架, 分别针对工业控制 ICS 协议[56]、SCADA 系统私有协议[71–73]、智能电网 IEC61850 协议[74]等专用协议进行漏洞挖掘. ICS 协议中的大多数都是专有的, 没有公开可用的文档. 在工控系统 IDE 和 PLC 之间传输数据, 需要进行 TCP/IP 握手, 且通常会有额外的专有握手, 这些是对专有协议模糊测试的挑战. Tu 等人[74]提出了一种有效的漏洞挖掘系统 IECFuzzer, 其可以发现 IEC61850 协议潜在的拒绝服务漏洞, 还可以验证采用 IEC61850 协议的 PLC 设备的健壮性.

评价维度及分析: 由于相关工作都是针对专用协议, 因此主要从支撑技术的能力和有效性方面来评价. 目前, 中间人通信机制对于复杂网络协议客户端和服务端之间的通信协议具有较高的适用性. 现有的专用协议支撑技术都有针对性, 一方面能够让黑盒模糊测试和灰盒模糊测试适用于专用协议软件, 另外一方面也提供了定制策略、专用调度等功能, 较好地解决了部分专用协议的测试问题.

表 6 专有协议支撑技术相关工作比较

方法	文献	针对协议	所用方法	支撑类型	关键结论
专有协议的支持框架	[68]	Compact FTP等4款FTP实现	中间人通信机制	模糊测试框架	可以用来理解网络协议实现的消息序列结构和协议语法
	[69]	SSL加密协议	扩展通信机制支持SSL加密协议	模糊测试框架	支持对加密协议进行模糊测试
	[70]	数字证书	基于OpenSSL创建具有特定字段的数字证书	模糊测试框架	提供一种针对SSL协议有效的黑盒测试方法
	[47]	IKE协议	两阶段测试用例生成机制	模糊测试框架	具有可扩展性和自动监控能力
	[50]	IKEv2协议	基于scapy的测试用例生成	模糊测试框架	在不具备硬件设备的条件下, 采用模拟固件环境进行测试
	[56]	工业控制ICS协议	扩展测试用例生成支持ICS协议	模糊测试框架	能够测试专有的工业控制系统协议和监控控制器的行为
	[71]	SCADA系统的私有协议	程序执行过程中提取的动态信息,	模糊测试框架	可以在没有协议语义、目标插桩的情况下进行测试
	[72]	SCADA系统的私有协议	基于语法模板生成测试用例	模糊测试框架	可提高代码覆盖率, 验证了该方法的有效性
	[73]	Modbus-TCP	自适应的测试用例生成算法	模糊测试框架	能够很好地满足Modbus-TCP漏洞检测的要求
	[74]	智能电网IEC61850协议	融合变异和数据重装的生成算法	模糊测试框架	挖掘潜在的拒绝服务漏洞, 并可以验证PLC设备的健壮性
	[75]	蓝牙BR/EDR L2CAP协议	覆盖L2CAP状态且适配目标组件的测试用例生成	模糊测试框架	可以提高协议状态覆盖率, 有效地生成适配目标蓝牙设备的测试用例
适配专有协议的支撑组件	[33]	物联网MQTT协议	基于中间人通信	代理通信组件	实现代理通信和数据包拦截, 从而实现对目标协议的测试
	[45]	无线网6LoWPan协议	基于嵌入式驱动程序注入6LoWPAN数据包	专用定制组件	通过定制组件, 实现对目标协议的测试
	[76]	ZigBee协议	拦截NWK层帧并且向生成模块发送测试用例	专用置换模块	使得模糊测试可用于资源受限的ZigBee协议
	[77]	ZigBee协议	综合模糊测试算法	测试组件	实现在过滤规则下的高通过率和漏洞触发, 提高了测试效率
	[78]	汽车总线协议	适配CAN总线测试用例生成	专用测试组件	可实现自动对汽车的CAN总线进行模糊测试

4 基于模糊测试的网络协议软件漏洞挖掘方法

模糊测试是网络协议软件漏洞挖掘的主要方法之一, 最早引入来挖掘网络协议软件漏洞. 通过向目标软件注入大量的非预期输入找出潜在漏洞. 其中黑盒模糊测试采用盲注的方式, 基于网络协议语法和策略生成机制来生成大量测试用例. 随着模糊测试技术的发展, 也引入了基于信息反馈的灰盒模糊技术. 灰盒模糊测试由于能够通过信息反馈实现对测试策略及测试用例的有效性进行评估, 能够自主学习测试用例变异的方式和位置, 因此其测试调度过程更有针对性. 然而, 由于网络协议软件的有状态特性, 模糊测试过程中既需要完成测试用例生成的调度, 也需要维持网络协议软件的会话状态. 这是与命令行软件模糊测试的主要差别之一.

在基于反馈的网络协议软件模糊测试技术中, 由于网络协议软件规范性较强的特点, 同一协议规范存在多种实现. 差分测试技术可有效辨别不同网络协议软件实现的细粒度差异, 以及发现网络协议软件的漏洞. 针对多个目

标的差分测试与单个目标的有反馈模糊测试技术不同, 其反馈是采集多个不同网络协议软件的动态信息进行比对, 其调度过程是以生成能够有效区分不同网络协议软件实现为目标. 这一技术也吸引了不少学者展开研究. 这也是与命令行输入软件模糊测试的主要差别之一.

4.1 模糊测试的用例生成

常见的模糊测试策略有基于消息生成、基于消息变异等方面. 基于生成的模糊测试策略, 是指基于协议语法描述, 利用策略生成大量报文; 基于变异的模糊测试策略, 主要通过对已有测试用例进行突变来构造畸形数据. 除此之外, 还有一类采用反馈机制的测试用例生成方法, 采用程序分析等收集测试用例执行的反馈, 如图 2 所示. 该机制不依赖于现有协议知识, 而是通过得到的反馈信息对策略进行有效性评估, 其方法与变异方法类似, 都是对种子报文进行修改和突变, 区别在于基于反馈机制的生成机制, 变异时机和位置是由程序自己“学”到的, 带有智能性, 而基于生成的测试用例生成过程是预先确定的. 如表 7 所示描述了这 4 类相关工作的比较.

表 7　测试用例生成相关工作比较

方法	文献	针对协议/协议软件	种子数据内容/反馈数据	生成机制/算法	关键结论
基于生成策略	[28]	SIP	N/A	基于增强的BNF模型	模型可有效生成消息元素关联的测试用例
	[79]	MMS	N/A	基于字段特征和字段分类模型	能够生成符合字段类型要求的测试用例
	[80]	MSN	N/A	基于FSM模型的字段变异和消息类型	能够生成满足协议规范的测试用例
	[81]	SGW, SPGW	N/A	基于消息结构树模型	能够生成满足消息结构的测试用例
	[30]	Modbus/TCP, IEC, Siemens S7等工控ICS协议	N/A	基于内存传播树模型	提高了生成测试用例的性能, 以较少的测试用例实现更高的分支覆盖率
	[82]	Open-FTPD	N/A	基于协议分类树模型	可有效通过协议字段替换生成测试用例
	[83]	SNMP	N/A	对SNMP协议字段集进行多种生成操作	可有效生成SNMP测试用例
基于消息变异	[70]	OpenSSL	创建的数字证书	基于字段变异	能够有效地对数字证书进行模糊测试
	[31]	FTP, PN-DCP两种协议的4个目标软件	网络流量	基于字段和消息关系进行关系突变	可辅助进行模糊测试并支持动态的反馈和调整
	[77]	ZigBee	网络流量	基于边界算法和结构节点克隆算法来生成数据	能够产生通过率高、效率高、体积小、漏洞触发率高的数据
	[84]	智能交通系统ITS	网络流量	协议结合静态模糊向量、随机数据进行变异	可变异产生新的测试用例, 粒度到字段级
	[85]	HTTP	网络流量	基于消息矩阵摄动模式进行数据突变	可生成所有语法级别的测试数据, 在复杂的突变组合中保持灵活性
	[86]	工控网络协议	网络流量	多字段关联的测试用例生成	可进行私有协议结构分析和状态机识别, 并进一步突变
	[87]	专有网络协议	网络流量	模型引导的变异机制	能够支持既无协议规范也无源码的专有网络协议
	[75]	蓝牙BR/EDR L2CAP协议	网络流量	基于消息格式的字段变异, 通过协议状态的覆盖情况来引导测试	可有效生成针对蓝牙BR/EDR L2CAP的测试用例
基于反馈机制	[15]	通用协议	网络流量/执行力路径	执行路径引导的测试用例变异	能够有效支持网络协议的有反馈模糊测试
	[81]	通用协议	基本块统计信息	基于统计信息和崩溃信息反馈	能够容易通过早期检验、得到高覆盖率的测试用例
	[88]	SSL X.509证书校验程序	路径信息	基于路径约束求解生成	可以通过交叉验证来检测不同库中的代码错误
	[20]	通用协议	路径及敏感函数	基于动态污点信息引导生成	能够克服传统模糊测试方法的盲目性, 提高了测试效率

表 7 测试用例生成相关工作比较 (续)

方法	文献	针对协议/协议软件	种子数据内容/反馈数据	生成机制/算法	关键结论
基于机器学习模型	[89]	专有网络协议	网络流量	基于神经网络学习数据输入模型	基于模型生成新的测试用例, 实现与该未知协议软件进行通信
	[90]	Server-U 和 Filezilla两个FTP软件	协议规范	基于Seq2Seq-attention模型和LSTM模型学习变异位置和字符序列	生成的测试用例与基于语法的方法相比, 可以覆盖到更多地基本块结果表明3层LSTM模型正确性最高
	[35]	Pymodbus流量生成器, pcap文件	N/A	生成对抗网络和SeqGan算法对协议语法学习	可引导具有高覆盖率的输入生成
	[64]	嵌入式设备	固件程序运行状态	采用遗传算法模型来生成	可引导生成有高覆盖率的输入
	[91]	SSL协议	证书模板库	使用进化的方法来生成	不同的TLS服务器实现中诱导出比以前的算法更明显的响应差异

注: N/A表示不存在此项

关于测试用例这一术语, 在网络协议软件漏洞挖掘的领域有不同的表述. 从模糊测试的角度, 一般被称为测试用例, 但从挖掘对象的角度, 又可以被称为数据包、网络报文、网络流量等不同的表述. 从网络协议的角度, 又可以被称为网络消息. 在本文中, 一般用测试用例这一词语来表述, 但在涉及挖掘对象和网络协议的相关描述时, 又被称为数据包或网络流量.

方法的优势分析: 基于生成策略和基于消息变异机制的测试用例生成方法, 其特征是根据已有的网络协议知识来产生测试用例. 而基于反馈机制的测试用例生成方法则不同, 该方法通过测试过程中收集到的信息进行反馈, 感知挖掘对象的运行状态, 来决定下一步如何生成测试用例. 主要包括基于反馈机制、基于机器学习的测试用例生成优化等技术. 基于消息变异的测试用例生成, 旨在从网络协议测试用例的局部进行消息变异来生成新的测试用例, 如文献 [88] 提出一种针对 X.509 证书校验实现的异常检测方法 SymCerts, 该方法采用符号执行技术, 来收集挖掘对象运行过程中的路径信息. 由于采用符号执行分析库函数的证书校验部分, 获得了用于描述接受证书和拒绝证书的路径约束, 从而可以通过交叉验证来检测不同库中的代码错误. 基于消息变异的测试用例生成方法既可以在缺乏网络协议描述知识的情况下自动产生测试用例, 也能够在具备网络协议描述知识的情况下产生新的测试用例. 针对缺乏网络协议模型及知识的情况, 现有的技术采用从网络流量中提取种子, 并进行突变的方式来实现生成测试用例. 在针对专有协议的模糊测试方面, 基于流量变异的方法通过观测专有协议的网络数据流量, 设计生成模型来推测消息格式和协议状态, 能够支持既无协议规范也无源码的专有网络协议.

评价维度及分析: 主要从生成测试用例方法的效率和效果、场景适用性两个方面来评价, 在相关工作的实验验证中, 多以路径数量、独立崩溃数量和已知漏洞验证数量等方面进行评价效果. 测试用例生成方法所采用的种子数据和所依赖的反馈机制和学习数据决定了其适用的场景, 而测试用例生成方法所采用的机制和算法, 决定了测试用例生成在字段类型和关联等协议语法的支持能力以及复杂测试用例的生成能力. 在这类方法中, 文献 [90] 的实验结果表明 3 层 LSTM 所生产的测试用例获得的正确性最高.

4.2 模糊测试的过程调度

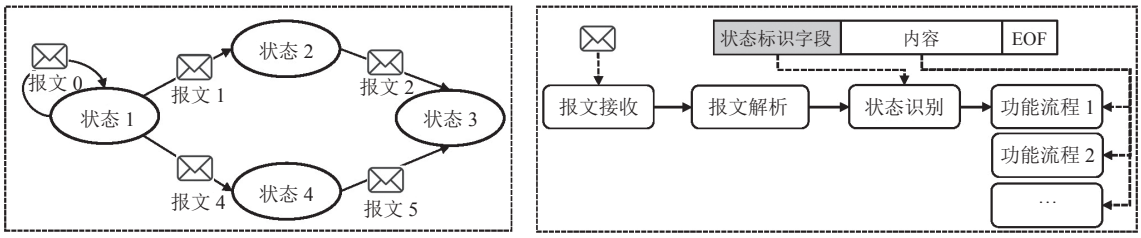
网络协议软件模糊测试过程调度, 是对下一步使用哪个测试用例、使用哪个变异和生成策略进行有序和高效的选择, 其主要方式有: 基于多级变异的过程调度、协议状态机测试的过程调度、基于差分测试反馈的过程调度. 基于多级变异的调度方法通过设计层次化的调度方案, 分别处理网络报文头和报文内容等; 协议状态机测试的过程调度, 通过在协议特定状态下, 探索下一步可接受的协议消息; 基于差分测试反馈的过程调度, 则是利用差分测试技术, 根据不同协议实现的执行结果来决定如何生成新的测试用例. 如表 8 所示描述了这 3 类相关工作的比较.

状态测试是网络协议模糊测试的一个特殊过程, 如图 3 所示. 图 3(a) 则进一步展示了在报文序列下引导的协议状态转换, 网络协议软件具有巨大的状态空间, 在通信过程中只有使用符合协议规范的报文序列才能正确地触

发其内部的状态转换. 图 3(b) 则展示了协议软件的报文与状态处理过程. 协议报文通常具有高度格式化的数据结构, 图中以文本类协议为例, 其结构由多个字段组成, 包含状态标识字段和传输数据内容的字段组成. 在该流程中, 格式中的状态标识字段将决定数据内容部分由何种功能流程来处理. 报文状态标识是报文有效性和状态测试的基础, 解析失败的报文无法驱动协议软件的状态转换.

表 8　测试过程调度相关工作比较

方法类型	文献	针对协议	所用算法/方法	调度目标	关键结论
基于多级变异的	[8]	FTP等通用协议	头部、内容和序列三级变异策略	覆盖有状态协议会话和信息空间	对同一报文进行多级变异后, 会产生与原来差别很大的新报文
	[79]	MMS等通用协议	长度、内容和规则分级变异	覆盖状态协议信息空间	能够对每个域进行类型标识
面向协议状态测试的	[92]	IPv6的邻居发现协议	基于状态行为模型的测试策略	覆盖状态协议状态空间	模糊测试的过程进行反馈学习进而不断选择优化测试策略
	[66]	4G/LTE网络协议	通过强化学习的方法来引导	发现目标协议的异常行为	可引导模糊测试对LTE网络模拟器的测试行为, 从而实现自动化的测试4G/LTE移动设备
	[51]	有状态网络协议	通过网络协议程序状态快照和消息链分析算法来引导	覆盖状态协议状态空间	可引导模糊测试探索网络协议程序中更多更深的状态, 提高有状态网络协议漏洞挖掘的效率
	[75]	蓝牙BR/EDR L2CAP协议	通过基于字段变异和状态反馈的算法来引导	覆盖蓝牙协议状态空间	可生成符合协议规范且覆盖状态空间大的测试用例
	[93]	有状态网络协议	蒙特卡洛树搜索算法	覆盖状态协议状态空间	所提出的算法比现有的状态选择算法更高效, 但依然存在改进的空间
	[94]	有状态网络协议	基于程序插桩分析的状态识别和引导变异算法	覆盖状态协议状态空间	所提出的算法改进了协议状态感知的准确性, 提升了模糊测试的吞吐率和效率
面向差分测试	[95]	SSL协议	自动机学习算法	生成有差异的证书模型	可检查模型中是否存在错误
	[91]	TLS协议	使用进化的方法	诱导出更明显的响应差异	可诱导生成测试用例, 并支持衡量TLS协议的异常行为
	[46]	TLS实现	通用消息树	产生响应差别很大的测试消息	可实现对不同的TLS实现进行差分测试
	[22]	SSL网络协议	利用强化学习	引导生成有区分度的测试用例	可生成能够最大化区分多个SSL软件的测试用例



(a) 状态机示意图　　(b) 报文解析与状态处理流程

图 3　网络协议状态转换及报文处理流程

方法的优势分析: 网络协议的过程调度, 目前的基于多级变异、协议状态机测试和面向差分测试的过程调度这 3 种方式, 都是针对网络协议的特性提出的模糊测试过程, 针对性比较强. 基于多级变异的测试过程调度, 以对网络协议的消息空间和状态空间进行充分测试为目标. 而面向状态协议测试的, 则以状态空间覆盖并发现状态异常为目标. 面向差分测试的测试过程的调度, 适用于同一协议规范的不同协议实现开展差异性测试, 也可以发现漏

洞, 同时引导生产有区分度的测试用例. 如 DRLgencert[22]针对多款 SSL 网络协议软件进行差分测试, 采用数字证书作为种子, 通过采集协议软件的响应并引导生成测试用例, 利用强化学习的学习能力来优化策略的使用和变异的位置, 从而能够支持最大化区分多个 SSL 协议软件.

评价维度及分析: 主要从所采用方法的能力和适用性两个维度进行评价, 能力维度涉及覆盖哪些网络协议内容, 如头部、内容和状态序列等; 适用性是指所能够支持的场景和针对的协议. 其中, DRLgencert[22]能够发现针对多款 SSL 协议实现具有区分度的大量测试用例, 也能够发现 SSL 软件实现的多种缺陷, 虽然其实验测试仅仅针对了 SSL 协议, 但该方法具有很好的普遍性, 可应用于其他协议的差分测试.

5 基于程序分析的网络协议软件漏洞挖掘方法

目前有一些相关工作通过引入程序静态分析、动态污点分析和动态符号执行, 实现对网络协议软件漏洞挖掘的辅助分析. 程序静态分析技术, 主要是对挖掘对象通过静态分析来定位危险函数、特定漏洞点等信息, 以辅助进行漏洞挖掘.

污点分析技术通常用于跟踪程序中变量的定义和引用关系, 是一种重要的程序分析手段. 污点分析分为静态污点分析和动态污点分析[96]. 静态污点分析是在不运行程序的基础上, 分析代码获取语法和语义, 检测程序中的数据流和控制流关系. 动态污点分析监控程序的执行流程, 对不受信任的输入相关的内存、寄存器等进行标记, 跟踪程序运行时对数据的处理, 记录数据传播过程, 以此找到受污染源影响的有漏洞代码. 在目前的相关工作中, 动态污点分析技术通常与模糊测试相结合, 记录测试用例在程序执行过程中的数据流和控制流关系, 指导模糊测试有针对性地生成测试用例, 提高模糊测试的代码覆盖率.

符号执行技术出现在 20 世纪 70 年代中期, 由 Boyer 等人[97]、King[98]、Howden[99]等相继提出. 符号执行技术是一种形式化的分析方法, 通过将程序中变量的值表示为符号值, 并与已知的常量组成计算表达式, 将程序计算得到的输出刻画为输入符号值的函数. 因此, 对于程序的每条执行路径, 都会生成若干个包含了输入符号值的条件表达式, 称为符号执行的路径约束. 对于程序分析而言, 判断各条执行路径上条件表达式的可满足性, 就可建模为某些符号化变量的取值是否可满足相应约束的问题.

动态符号执行技术, 适合在真实的测试环境中, 精确地跟踪程序的执行路径, 而不用担心状态爆炸问题. DART[100]首次将实际执行与符号执行结合起来, 即混合执行测试 (concolic testing). 混合执行需要维护程序执行时的整个实际值状态, 因此它需要一个具体的初始值, 一次只分析一个输入, 当执行到某条路径的分支时, 通过翻转条件约束, 达到覆盖新路径的效果. 另一方面, EXE[101]和 KLEE[102]提出并扩展了执行生成测试 (execution-generated testing, EGT). 该方法的工作原理是在程序执行的过程中, 区分实际值状态和符号值状态. 如果在执行某个操作之前, 所有相关的操作数都是精确值, 则直接执行原始程序; 若是其中存在符号化值, 那么该操作将会符号化执行.

5.1 程序静态分析技术

网络协议软件漏洞挖掘中目前采用的程序静态分析技术主要有控制流分析、函数调用图分析、函数过程间分析等. 引入程序静态分析技术分析网络协议软件漏洞, 可提取某些特定的字段值, 识别出危险函数, 定位危险代码的数据污染源等, 尤其是与网络协议软件相关的 system 函数、exec 函数等, 而后再将这些数据辅助之后的漏洞挖掘过程. 如表 9 所示描述了程序静态分析相关工作的比较, 主要从分析方法和分析目的方面来比较.

方法的优势分析: 基于程序静态分析的相关工作, 主要针对网络协议的二进制软件开展控制流分析、数据量分析和敏感函数识别等工作, 鲜有针对源代码形态的静态分析. 程序静态分析由于能够预先获取程序目标的信息, 并辅助模糊测试的测试用例生成, 可提升检测漏洞效率和效果.

评价维度及分析: 静态分析技术的适用性强, 所提取的信息类型也比较丰富, 因此其可有效应用于网络协议漏洞挖掘, 辅助生成测试用例和模糊测试的过程调度.

表 9 程序静态分析相关工作比较

文献	针对协议	静态分析方法	静态分析目的	关键结论
[23]	通用协议	控制流分析	优化模糊测试的测试用例生成	可优化模糊测试的测试用例生成过程
[24]	苹果移动应用中网络协议	动静态相结合的局部代码分析	检测漏洞	可进行细粒度的潜在漏洞检测
[25]	IP Phone设备	结合控制流和数据流, 敏感函数	检测漏洞	该自动化方法可有效提升检测效率
[16]	工业控制协议	函数代码感	筛选候选函数代码参数并且识别危险操作	可优化后续测试阶段
[94]	有状态网络协议	动静态相结合的方法分析程序执行轨迹	识别网络事件相关的代码分支点和提取状态变量	可辅助提升模糊测试的状态覆盖率
[103]	有状态网络协议	动静态相结合的方法分析程序执行轨迹	识别程序状态变量取值变化和提取状态迁移模型	可辅助提升模糊测试的状态覆盖率

5.2 动态污点分析技术

动态污点分析通过记录污染源在数据包中的偏移位置, 识别关键字段, 指导模糊测试程序进行有效的测试用例变异. 在网络协议软件漏洞挖掘过程中能够发挥敏感函数识别、输入格式逆向、用户输入关键字段定位等功能. 如表 10 所示描述了动态污点分析相关工作的比较, 主要从污点源和污点分析目标两个方面进行对比.

表 10 动态污点分析相关工作比较

文献	针对协议	污点源	污点分析目标	关键结论
[20]	通用协议	网络数据包	识别受污点源影响的敏感函数	可支持测试用例变异策略对数据包进行变异
[104]	工业互联网协议	通信数据	得到相关的动态多模态传感器通信数据	可实现高效的测试用生成
[21]	Java程序	网络数据包	输入格式逆向	可将程序输入结构逆向为上下文无关的语法
[16]	工业互联网协议	网络数据包	识别关键字段	可指导模糊测试的测试用例变异, 在实验中发现了工业互联网协议中的多个未知漏洞

方法的优势分析: 污点信息引导的网络协议软件模糊测试, 通过将网络数据包作为污染源, 识别出受网络数据包影响的目标程序敏感函数, 增强模糊测试测试用例变异的针对性. 如文献 [16] 提出了一种代码感知框架, 在不获取协议数据包格式规范的情况下, 通过动态污点分析记录污染源在数据包中的偏移位置, 识别关键字段, 指导模糊测试程序进行有效的测试用例变异, 并在实验中发现了工业互联网协议中的多个未知漏洞. 这些工作表明, 污点分析技术在网络协议软件漏洞挖掘中能够发挥敏感函数识别、输入格式逆向、用户输入关键字段定位等功能.

评价维度及分析: 由于污点分析技术的复杂度高, 对应用场景的要求高, 又面临着网络协议软件的程序逻辑复杂等问题, 目前在实际网络协议软件漏洞挖掘过程中应用污点分析技术的工作较少. 现有的动态污点分析工具如 Dytan[105]、Minemu[106]等主要用于分析终端命令行输入软件. 网络协议软件以网络数据包作为输入并且逻辑复杂, 而跟踪数据包的传播过程比跟踪终端命令行输入软件更困难.

5.3 动态符号执行技术

网络协议软件的漏洞挖掘需要探索尽可能多的、更深入的程序路径. 针对每一条程序路径, 测试工具还应当具备生成一个具体输入值的集合, 以及检查除了程序崩溃之外, 是否还存在断言违规、约束违背异常、内存损坏、认证绕过等各种缺陷的能力. 这推动了动态符号执行应用于网络协议软件的漏洞挖掘. 以 SAGE[107]为代表的混合模糊测试工具不仅可以在较短的时间内, 快速发现大量易于通过变异覆盖的程序路径; 还能利用混合执行测试, 以

可容忍的开销, 求解出难以通过变异得到的复杂路径条件, 让模糊测试可以触发分支约束, 探索到更深的路径, 提高了程序脆弱性检测的能力. 根据网络协议软件不同的特征和分析侧重点, 本部分主要介绍面向已知协议的辅助分析技术、面向未知协议的辅助分析技术和针对加密数据的辅助分析方法. 如表 11 所示描述了动态符号执行相关工作的比较, 从所采用的模型、方法和分析目标 3 个方面来进行对比.

表 11 动态符号执行相关工作比较

方法类型	文献	针对协议	所用模型	所用方法	辅助分析目标	关键结论
面向已知协议	[17]	通用协议	基于协议状态模型的符号执行	利用协议模型指导符号执行生成报文	提高程序路径和协议状态的覆盖率	可有效提高覆盖率, 难以应对现实中有状态网络协议复杂的交互和状态转换
	[88]	SSL协议	对库函数中证书校验过程的代码做混合执行测试	路径约束集上的异常分析	提高程序路径发现的数量	可能够发现证书实现上的错误, 是对灰盒模糊测试技术的有效补充
面向未知协议	[108]	木马通信协议	基于符号执行提取通信协议的语法	推测服务端解析消息的谓词	比对差异, 寻找协议漏洞	可有效发现通信协议缺陷
	[29]	DNS, eDonkey, FTP, HTTP, McAfee ePO	基于中间语言的符号执行	基于符号执行推理程序行为	自动化的应用层协议逆向	为输入的数据如何使用提供了额外的信息, 可从执行指令的语义中提取出协议格式
针对加密数据	[109]	Winamp, tcpdumpd	基于符号执行构造畸形的测试用例	对比执行路径来定位检查点	求解复杂约束的测试用例	受限于约束求解器的计算能力和效率, 无法应对复杂的加密机制
	[19]	Zbot等	代码局部符号执行	采用分割求解的方法	降低加密机制符号执行中的复杂度	方法需要根据解密函数的特征来定位加密函数, 在非对称加密机制中难以获得加密函数的逆函数
	[110]	Apache http, PCshare等	分解-重构的方法	定位加密数据的解码位置	适用于非对称加密机制	无需获取加解密算法的具体实现, 可在非对称加密机制中使用

方法的优势分析: 符号执行辅助模糊测试是根据报文格式来符号化标记通信报文作为符号执行的输入, 利用协议模型指导符号执行提高程序路径和协议状态的覆盖率. 其主要优势是: 在查找给定程序路径上的错误时, 符号执行比传统的模糊测试技术更适用, 这得益于其生成触发错误的具体输入的有效性. 而且, 与其他程序分析技术不同, 符号执行不仅限于查找诸如缓冲区溢出之类的错误, 而且可以发现更高级别的程序缺陷, 例如复杂的程序状态混淆、断言违规、内存泄漏等. 当然, 其也受制于符号执行引擎约束求解器的能力, 而且面临着路径爆炸的问题.

评价维度及分析: 主要是目标的适用性和符号执行的求解能力. 目标的适用性主要体现在所采用的符号执行方法, 能够针对哪些协议目标和哪些形态目标开展符号执行. 而符号执行的求解能力, 主要是评价所针对目标的复杂性和所采用的符号执行引擎的求解能力, 如 SymCerts[88]是一款针对 X.509 证书实现的规约违背错误检测工具, 它通过对 X.509 证书链采用符号值与确定值混合的方法, 通过对库函数中证书校验过程的代码做混合执行测试, 获得了用于描述接受证书和拒绝证书的路径约束, 能够发现证书实现上的错误, 是对灰盒模糊测试技术的有效补充. 考虑到该方法是直接对 SSL/TLS 库加载符号执行引擎, 面临着一定程度的路径爆炸问题.

6 讨论与展望

6.1 网络协议软件漏洞挖掘技术的现状分析

现有的网络协议漏洞挖掘技术如网络协议描述方法、模糊测试技术、程序分析技术等, 从不同的角度提供了不同的功能. 但是目前针对网络协议软件的漏洞挖掘技术依然存在一些不足, 由于网络协议软件漏洞挖掘对象适配的这一特点, 针对部署于嵌入式系统和物联网固件中专用网络协议组件, 现有的适配支撑技术依然存在不足. 由于网络协议软件的漏洞类型多样及成因复杂的特点, 目前的网络协议软件漏洞挖掘方法, 其检测漏洞的能力依然比较弱.

综合网络协议漏洞挖掘的技术分类及第 2–5 节相关工作的综述分析, 针对实现高效、自动和可扩展性好的网络协议软件漏洞挖掘这一目标, 本部分剖析了漏洞挖掘技术的不足, 主要包括: 多类型漏洞的判定能力不足、挖掘

对象适配可扩展性差、模糊测试的效率低和可扩展性差、漏洞挖掘的智能化程度不高. 具体阐述如下.

(1) 多类型漏洞的判定能力不足

由于网络协议软件存在不同的漏洞类型, 包括常见的程序崩溃类, 也包括其特有的类型, 如信息泄露类、认证绕过类、协议状态混淆类等, 如表 4 所示. 这些不同的漏洞类型, 是由网络协议软件的功能和特征决定的. 程序崩溃类漏洞, 主要是由于大量网络协议软件是由底层 C 和 C++语言编写而成, 导致其存在指针处理不当、数据访问越界等问题. 信息泄露类漏洞, 主要是由于数据访问越界等原因导致内部不应该被访问的数据发生了泄露, 以心脏滴血漏洞 CVE-2014-0160 为例, 该漏洞是典型的信息泄露类漏洞, 可泄露多达 64 KB 的服务端内存数据. 认证绕过类漏洞, 主要是在语义层面, 外部请求导致程序逻辑在未通过认证函数校验的情况下, 执行了系统命令. 协议状态混淆类, 主要是由于网络协议软件在处理协议状态机时, 导致程序状态发生了异常偏离, 使得程序进入了不正常的状态. 由于网络协议的有状态特性, 使得网络协议软件缺陷的形成, 不仅仅发生于与外界的一次交互过程中, 也出现在与多次交互过程中. 此类漏洞成因更加复杂, 漏洞出现也更加隐蔽, 检测困难. 以 OpenSSL 协议软件漏洞 CVE-2021-3449 为例, 该漏洞存在于 ClientHello 报文第 1 次交互与重协商过程中的第 2 次 ClientHello 交互过程的核心逻辑代码中, 导致内部代码错误出现指针被两次释放.

从网络协议软件漏洞成因的角度, 网络协议软件存在有状态特性, 网络协议软件程序的运行状态, 不仅仅受当前用户输入的影响, 也跟协议软件的会话状态、当前配置有关. 而且由于网络协议软件承担着控制和管理的功能, 更加容易产生信息泄露、认证绕过、命令执行等类型漏洞. 传统模糊测试技术采用的测试用例生成方法, 一方面会受到网络协议会话状态的限制, 另外一个方面也难以发现信息泄露和认证绕过等非程序崩溃类漏洞. 文献 [47] 对 IKE 协议漏洞数据集的统计分析表明: 有 28% 的漏洞无法通过模糊测试技术来发现.

总之, 网络协议漏洞的漏洞类型多样, 其成因是与网络协议软件的有状态特性及其特殊的功能相关. 现有的智能有反馈模糊测试技术和符号执行技术, 难以发现于网络协议软件多类型漏洞, 不支持网络协议软件中状态相关的漏洞挖掘, 存在多个方面的技术局限性.

(2) 挖掘对象及插装工具适配的可扩展性较差

网络协议软件形态多样的特点, 决定了漏洞挖掘框架需要与待挖掘的网络协议软件对象进行适配, 目前的漏洞挖掘方法缺乏对二进制协议软件和嵌入式固件中网络协议软件的支持. 如表 5 所示, 对于不同形态的挖掘对象, 如源代码态、二进制形态、嵌入式固件形态, 以及相应的插桩工具, 如 Intel Pin、LLVM、Qemu 等, 也缺乏统一的插桩监控机制和规范流程, 导致适配工作难度大, 适配过程复杂. 面临着如何为多形态的网络协议软件提供运行时插桩监控的困难.

对嵌入式系统和物联网环境中的专用网络协议组件进行智能有反馈模糊测试, 需要构建相应的仿真环境和插桩方法, 来适配智能反馈机制所需要的信息反馈和状态监控. 以 SSL 协议为例, 部署于通用操作系统平台的 SSL 服务软件与部署于物联网固件的 SSL 服务组件, 其模糊测试过程就存在较大差异. 为了适配专用网络协议组件, 漏洞挖掘工作需要研究可扩展性好的适配支撑技术, 才能够支持专用协议组件和专用系统.

(3) 模糊测试的效率低和可扩展性差

网络协议模糊测试过程中协议交互过程复杂导致测试速度慢, 而且其测试空间不仅包括网络协议输入数据这个维度, 还包括网络协议多次交互构成的状态空间这个维度. 现有的网络协议模糊测试, 其测试效率远低于终端命令行输入软件的模糊测试. 此外, 终端命令行输入软件的模糊测试, 能够通过动态符号执行辅助模糊测试进行路径探索, 以及通过污点分析等技术辅助进行约束收集等方式, 以增强目标软件路径的探索效率. 况且, 由于大量厂商自定义的扩展协议和私有协议的存在, 这种规范化描述的来源也不仅来自网络协议的标准规范, 也需要借助程序分析的手段, 自动提取网络协议目标软件的语法描述. 传统路径分析优化和协议语法抽取技术应用的一个前提是测试目标和分析感知工具在统一操作系统的进程空间, 这使得这些技术难以应用于嵌入式系统及物联网固件等环境. 网络协议软件模糊测试, 面临着测试过程测试效率低、辅助分析工具缺乏等问题.

此外, 由于网络协议的类型多和测试目标多种多样, 不同的网络协议软件目标不能简单地与 AFL 类工具进行适配. 现有的网络协议模糊测试工具, 如表 7 中的 AFLNet[15]、STATEAFL[111]等, 都需要修改模糊测试引擎的代

码, 对测试协议逐个进行适配. 这种测试引擎与测试目标紧耦合的框架, 导致模糊测试工具难以适配不同协议目标和不同的测试引擎开展测试, 存在可扩展性差的问题.

(4) 漏洞挖掘的智能化手段缺乏

经典的灰盒模糊测试技术, 虽然测试过程简单, 但是受限于人为描述网络协议消息格式, 导致测试自动化程度不高、可扩展性不好. 基于全系统模拟和插桩技术的智能有反馈模糊测试技术, 虽然能够基于反馈信息生成测试用例, 但模糊测试技术覆盖的漏洞类型有限, 缺乏对其他漏洞类型的支持. 此外, 现有的符号执行技术多以终端命令行输入软件为目标设计和开发, 其基本假设是被测命令行输入软件与符号执行工具运行在同一操作系统. 然而, 对于专用网络协议组件, 缺乏符号执行技术需要的运行条件.

表 8 中文献 [22] 以多款 SSL 协议软件证书校验程序的自动差分测试为目标, 提出一种基于强化学习的智能化模糊测试方法, 引入了深度强化学习来学习报文变异的位置和有效的策略, 能够提升 SSL 证书校验程序差分测试过程中测试用例生成的智能化程度. 该方法虽然引入了机器学习等方法, 提升测试过程中的自动化和智能化程度, 但是仅仅适用于差分测试这个场景. 现有的技术虽然实现了一定的自动化, 但智能化程度依然较低. 为了提升网络协议软件漏洞挖掘的自动化和智能化程度, 需要在协议描述格式的自动抽取、测试用例生成、程序执行路径信息反馈计算、固件仿真等过程提出新的智能化技术, 才能够降低人工参与程度, 提高网络协议软件漏洞挖掘的自动化和智能化水平.

6.2 网络协议软件漏洞挖掘面临的技术挑战

现有网络协议软件漏洞挖掘的效率、可扩展性等方面依然存在不足, 使得网络协议软件漏洞挖掘面临一些技术挑战. 以下阐述 4 个方面的技术挑战.

(1) 如何为多形态网络协议软件提供运行时插桩监控

运行时插桩监控是进行网络协议软件灰盒模糊测试的基本要求, 运行时插桩监控需要完成网络协议软件的进程状态提取、程序执行轨迹记录和状态重置等功能. 针对物联网固件和嵌入式系统的进程进行运行时监控, 需要采用固件仿真和虚拟机内省等技术, 实现固件的仿真运行和运行时状态获取和执行轨迹提取, 如表 5 和表 6 所示, 目前尚没有通用性强的物联网固件和嵌入式系统运行时插桩工具.

此外, 对于目前常用的源代码插桩、用户态插桩和系统态插桩这 3 种插桩方式, 目前尚没有统一的插桩规范, 其插桩控制机制和输出数据结构无统一格式, 使得模糊测试无法适用于不同插桩方式的相同网络协议. 而且由于网络协议软件功能逻辑复杂, 一般采用多线程或多进程方式为网络请求提供服务, 其线程和进程调度的随机性, 给智能有反馈模糊测试带来了测试过程的不稳定性, 课题组的前期研究成果表明, 网络协议软件模糊测试的稳定性是一个需要改进的问题[112].

(2) 如何定义网络协议的测试用例空间及如何进行状态反馈

与终端命令行输入软件不同, 网络协议不但是有状态的, 而且不同状态对应的消息之间存在约束或依赖关系. 这给模糊测试技术带来了新的挑战, 忽略消息之间的依赖, 会造成大量无效的测试用例; 而考虑消息之间的约束, 则需要测试用例生成过程包含额外测试策略和调度机制. 现有的网络协议软件模糊测试工具, 如表 7 中的 AFLNet[15]和表 5 中的 FIRM-AFL[1], 只是完成了 AFL 引擎在网络协议软件及嵌入式固件上的移植, 并未考虑网络协议有状态等特殊性. 另外一个方面是运行时配置的问题, 大多数网络协议软件是由配置驱动的, 配置的不同会使得软件的执行路径不同, 实际上配置文件也是网络协议软件的输入源, 也是网络协议软件存在漏洞之处.

此外, 现有基于 AFL 的网络协议软件模糊测试工具是以程序执行轨迹为反馈的, 并没有考虑网络协议状态测试与消息测试融合这个特点, 也没有考虑网络协议状态如何影响网络协议的模糊测试. 这带来的问题是, 一是忽略测试用例导致的会话状态会给模糊测试带来何种影响, 会降低测试用例的有效性, 二是尚不清楚如何利用这种状态反馈来辅助进行测试用例的有效性和状态相关漏洞发生的判定.

(3) 能否基于漏洞发生根源进行漏洞发现的辅助判定

现有基于模糊测试的网络协议软件漏洞挖掘技术主要是依靠程序崩溃这一操作系统信号来判定漏洞是否发生. 这一判定机制主要存在两个方面的不足, 一是无法应用于逻辑类漏洞发生的判定, 逻辑类漏洞如信息泄露漏

洞, 往往不会造成系统或进程的异常; 二是由于即使测试用例触发了漏洞点所在的程序执行路径, 也需要满足程序运行时的内存状态才能触发内存破坏内漏洞, 从而导致难以发现漏洞, 表 4 表明, 现有的相关工作缺乏从程序语义层面基于漏洞发生根源进行漏洞判定. 网络协议软件漏洞挖掘, 能否基于漏洞的根源, 探索漏洞发现的新机制, 提升漏洞发现的能力和效果. 程序崩溃、认证绕过及信息泄露、状态混淆及异常未崩溃等不同层级的漏洞类型, 这些漏洞类型, 有着不同的异常发现机制, 并且需要结合模糊测试和污点分析与符号执行等程序分析技术进行混合测试, 并从程序语义属性的角度来实施漏洞检测.

对于网络协议软件中常见的内存类漏洞以及逻辑类漏洞, 漏洞挖掘模型如何支持对不同漏洞类型的发生进行判定. 对于复杂成因的网络协议软件漏洞的发现, 还需要设计软件漏洞的知识规则, 并通过状态和信息收集, 进行漏洞发生的判定.

(4) 如何提升智能模糊测试的性能和可扩展性

由于网络协议软件形态多样的特点, 导致其存在测试性能低和可扩展性差等问题. 挖掘对象的不同形态对模糊测试的框架的要求不同. 如表 5 所示, 用户态网络协议软件虽然可以通过在通用平台上部署模糊测试工具, 但是网络连接及握手交互等过程带来的时间开销会极大降低测试速度; 正如文献 [113] 所述, 嵌入式固件的模糊测试效率低是一个主要的问题. 这是因为针对物联网及嵌入式系统中的网络协议专用组件进行测试, 频繁地启动物联网设备和嵌入式系统将使得测试过程无法正常进行, 也无法降低网络连接和握手带来的时间开销.

可扩展性方面的挑战, 主要在两个方面, 第 1 个是模糊测试引擎适配方面的可扩展性不好. 现有的模糊测试引擎, 如 AFL 引擎, 采用共享内存和进程间通信等机制, 来实现快速的进程状态和反馈数据的交互. AFL 工具的改进形成了一系列具有不同特点的 AFL 引擎, 将 AFL 类引擎用于网络协议软件的智能模糊测试, 即使是同一运行态的网络协议软件, 也需要在测试引擎和测试对象之间进行一对一的适配. 这极大地降低了现有漏洞挖掘框架的通用性.

6.3 发展方向展望

网络协议软件的漏洞挖掘, 与其他挖掘对象如脚本引擎[114,115]等的软件漏洞挖掘类似, 是属于对象结合的漏洞挖掘, 其挖掘技术往往具有一定的对象结合特性. 近年来, 对象结合的漏洞挖掘技术已经成为了一个热点研究方向, 对象结合的漏洞挖掘技术以具有一定输入规范的挖掘对象为目标, 在漏洞挖掘过程中结合挖掘对象的特性, 产生更有针对性的测试用例. 网络协议的漏洞挖掘, 其对象结合的特征也有多个方面的体现: 一是网络协议软件挖掘对象是二进制格式协议还是文本格式协议是有区别的, 而且在二进制格式协议中, 是位级格式协议还是字节级格式协议也有区别; 二是加密协议还是非加密协议的区别; 三是有状态网络协议还是无状态网络协议的区别. 网络协议软件的漏洞挖掘, 需要结合挖掘对象研究模糊测试技术、程序分析技术、挖掘对象适配技术等, 才能够有效提升漏洞挖掘的效率和智能化程度.

本文结合第 2–5 节网络协议软件漏洞挖掘技术综述分析和第 6.1 节和第 6.2 的分析, 提炼了如下 5 个方面的潜在研究方向, 其中 (1)–(3) 是从漏洞挖掘技术的角度, 而 (4) 和 (5) 是从挖掘对象的角度.

(1) 面向漏洞判定的网络协议软件属性分析技术

模糊测试技术是以测试用例生成和程序路径探索为目标, 模糊测试的检测能力主要依靠基于操作系统提供的进程状态这个单一的异常发现机制. 程序路径可达但不触发崩溃、非崩溃类漏洞在网络协议软件中广泛存在. 因此, 针对网络协议软件的漏洞类型多样和漏洞成因复杂的特点, 迫切需要一种面向漏洞判定的网络协议软件属性分析技术, 结合模糊测试和符号执行技术的优势, 构建包含网络协议软件的指令语义、函数语义、路径语义、数据流关系、路径时序关系等要素在内的软件属性分析技术, 支撑对网络协议软件多类型漏洞的判定.

(2) 网络协议目标的 Fuzz 外壳自动生成技术

网络协议描述方法提供了对网络协议语法和状态机等的建模和表示, 包含了从网络协议规范或网络流量中提取和学习到的知识. 而网络协议模糊测试的测试用例生成机制, 缺乏与网络协议描述知识进行关联, 存在一些研究空白. 如针对相同规范下存在多个网络协议实现, 基于生成策略和消息变异的测试用例生成, 是否能够在测试过程中产生对已有知识造成“破坏”的测试用例? 又如针对不同的挖掘对象, 网络协议知识如何快速辅助模糊测试技术的测试用例生成? 对这两个方面, 需要结合其各自的特点和优势, 形成有机统一的测试用例生成机制. 网络协议的

Fuzz 外壳, 是从网络协议软件模糊测试的角度, 生成大致正确和能够规避检查与校验的测试用例, 以及配套的协议规范描述, 使得模糊测试技术能够自动适用不同的测试协议, 提升模糊测试工具的可扩展性. 因此, 需要一种结合网络协议知识与测试用例生成机制的 Fuzz 外壳自动生成技术.

(3) 网络协议软件的模糊测试策略设计技术

黑盒模糊测试的策略设计较为完善, 但主要依赖人工描述网络协议语法, 存在固有的缺陷. 目前的灰盒模糊测试技术, 聚焦于解决挖掘对象适配的固件仿真与系统模拟、模糊测试框架以及程序分析辅助的漏洞挖掘框架等方面. 虽然引入 AFL 这一灰盒引擎提供了测试用例的自动生成功能, 组合形成了针对网络协议的有反馈模糊测试, 但目前已有的灰盒模糊测试工具, 如 AFLNet、FIRM-AFL、Fw-Fuzz 等, 都缺乏挖掘对象适配的策略设计. AFL 的位翻转策略、字节翻转策略, 没有考虑网络协议存在文本协议格式和二进制协议格式、位级协议格式和字节级协议格式等方面的区别, 也不能够理解协议报文头部和内容中包含的长度字段、关键字字段、消息类型域等类型. 因此, 需要设计用于有反馈模糊测试的网络协议策略, 以支持状态测试和高效测试用例生成.

(4) 面向专用系统和专有协议的支撑技术

面向专用系统和专有协议漏洞挖掘的场景类型多样, 虽然现有面向专用系统和专有协议的支撑技术取得了不少进展, 但依然存在固件仿真成功率低、全系统仿真下的程序分析能力不足、真实设备适配手段单一等问题, 难以满足自动化大批量固件仿真、全系统态下精准程序分析、真实设备网络协议软件测试分析等方面需求. 面向专用系统和专有协议的支撑技术, 如固件自动化仿真技术、面向全系统仿真的局部符号执行分析技术等亟待研究突破.

(5) 多线程多进程运行态网络协议软件的漏洞挖掘技术

网络协议软件由于其功能复杂以及对处理能力有要求, 普遍以多线程和多进程的形态对外提供服务. 对多线程运行态的网络协议软件进行漏洞挖掘, 面临着动态线程创建造成的程序执行轨迹不稳定等问题, 导致灰盒模糊测试的反馈机制失效. 针对多线程运行态网络协议软件的漏洞挖掘, 需要研究更加细粒度的程序插桩技术, 提出新型的模糊测试过程调度方法, 以实现网络协议软件自动化智能化的漏洞挖掘.

7 总 结

本文对网络协议软件漏洞挖掘技术展开研究. 首先阐述了网络协议软件漏洞挖掘的技术分类. 其次, 本文综述了网络协议描述方法、挖掘对象适配技术、基于模糊测试的网络协议软件漏洞挖掘方法和基于程序分析的网络协议软件漏洞挖掘方法这 4 个方面的技术进展, 对每类技术开展了技术优势分析和评价分析. 进一步, 本文系统地分析了现有网络协议软件漏洞挖掘技术的技术不足和挑战, 总结和提炼 5 个潜在研究方向. 本文的研究对提高和保障网络协议软件的质量具有重要的研究意义.

References:

[1] Zheng YW, Davanian A, Yin H, Song CY, Zhu HS, Sun LM. FIRM-AFL: High-throughput greybox fuzzing of IoT firmware via augmented process emulation. In: Proc. of the 28th USENIX Conf. on Security Symp. Santa Clara: USENIX Association Press, 2019. 1099–1114.

[2] Pothamsetty V, Akyol BA. A vulnerability taxonomy for network protocols: Corresponding engineering best practice countermeasures. In: Proc. of the 2004 IASTED Int'l Conf. on Communications, Internet, and Information Technology. St. Thomas: IASTED/ACTA Press, 2004. 168–175.

[3] Shoshitaishvili Y, Wang RY, Hauser C, Kruegel C, Vigna G. Firmalice—Automatic detection of authentication bypass vulnerabilities in binary firmware. In: Proc. of the 22nd Annual Network and Distributed System Security Symp. San Diego: Internet Society Press, 2015. 8–11.

[4] Hsu Y, Shu GQ, Lee D. A model-based approach to security flaw detection of network protocol implementations. In: Proc. of the 2008 IEEE Int'l Conf. on Network Protocols. Orlando: IEEE Press, 2008. 114–123. [doi: 10.1109/ICNP.2008.4697030]

[5] Lai YX, Liu J, Liu ZH, Zhang JW. Review on vulnerability analysis and vulnerability mining technology of industrial control system. Journal of Beijing University of Technology, 2020, 46(6): 571–582 (in Chinese with English abstract). [doi: 10.11936/bjutxb2019120008]

[6] Duchêne J, Le Guernic C, Alata E, Nicomette V, Kaâniche M. State of the art of network protocol reverse engineering tools. Journal of Computer Virology and Hacking Techniques, 2018, 14(1): 53–68. [doi: 10.1007/s11416-016-0289-8]

[7] Yue T, Wang PF, Tang Y, Wang EZ, Yu B, Lu K, Zhou X. EcoFuzz: Adaptive energy-saving greybox fuzzing as a variant of the adversarial multi-armed bandit. In: Proc. of the 29th USENIX Conf. on Security Symp. USENIX Association Press, 2020. 130.

[8] Song CX, Yu B, Zhou X, Yang Q. SPFuzz: A hierarchical scheduling framework for stateful network protocol fuzzing. IEEE Access, 2019, 7: 18490–18499. [doi: 10.1109/ACCESS.2019.2895025]

[9] Yu B, Wang PF, Yue T, Tang Y. Poster: Fuzzing IoT firmware via multi-stage message generation. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: ACM Press, 2019. 2525–2527. [doi: 10.1145/3319535.3363247]

[10] Mi XY, Wang BS, Tang Y, Wang PF, Yu B. SHFuzz: Selective hybrid fuzzing with branch scheduling based on binary instrumentation. Applied Sciences, 2020, 10(16): 5449. [doi: 10.3390/app10165449]

[11] Pereyda J. Boofuzz: Network protocol fuzzing for humans. 2023. http://boofuzz.readthedocs.io/en/latest/.

[12] Yi SW, Zhang CB, Xie F, Xiong Q, Xiang C, Liang LL. Security analysis of industrial control network protocols based on Peach. Journal of Tsinghua University (Science and Technology), 2017, 57(1): 50–54 (in Chinese with English abstract). [doi: 10.16511/j.cnki.qhdxxb.2017.21.010]

[13] Nishimura R, Kurachi R, Ito K, Miyasaka T, Yamamoto M, Mishima M. Implementation of the CAN-FD protocol in the fuzzing tool beSTORM. In: Proc. of the 2016 IEEE Int'l Conf. on Vehicular Electronics and Safety. Beijing: IEEE Press, 2016. 1–6. [doi: 10.1109/ICVES.2016.7548161]

[14] American fuzzy lop. 2023. https://lcamtuf.coredump.cx/afl/

[15] Pham VT, Böhme M, Roychoudhury A. AFLNet: A greybox fuzzer for network protocols. In: Proc. of the 13th Int'l Conf. on Software Testing, Validation and Verification. Porto: IEEE Press, 2020. 460–465. [doi: 10.1109/ICST46399.2020.00062]

[16] Luo ZX, Zuo FL, Jiang Y, Gao J, Jiao X, Sun JG. Polar: Function code aware fuzz testing of ICS protocol. ACM Trans. on Embedded Computing Systems, 2019, 18(5s): 93. [doi: 10.1145/3358227]

[17] Wen SM, Meng QK, Chao F, Tang CJ. A model-guided symbolic execution approach for network protocol implementations and vulnerability detection. PLoS One, 2017, 12(11): e0188229. [doi: 10.1371/journal.pone.0188229]

[18] Petrică L, Vasilescu L, Ion A, Radu O. IxFIZZ: Integrated functional and fuzz testing framework based on sulley and SPIN. Romanian Journal of Information Science and Technology, 2015, 18(1): 54–68.

[19] Caballero J, Poosankam P, McCamant S, Babi ĆD, Song D. Input generation via decomposition and re-stitching: Finding bugs in malware. In: Proc. of the 17th ACM Conf. on Computer and Communications Security. Chicago: ACM Press, 2010. 413–425. [doi: 10.1145/1866307.1866354]

[20] Cai J, Zou P, Xiong DP, He J. A guided fuzzing approach for security testing of network protocol software. In: Proc. of the 6th IEEE Int'l Conf. on Software Engineering and Service Science. Beijing: IEEE Press, 2015. 726–729. [doi: 10.1109/ICSESS.2015.7339160]

[21] Höschele M, Zeller A. Mining input grammars from dynamic taints. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Automated Software Engineering. Singapore: IEEE Press, 2016. 720–725. [doi: 10.1145/2970276.2970321]

[22] Chen C, Diao WR, Zeng YP, Guo SQ, Hu CY. DRLgencert: Deep learning-based automated testing of certificate verification in SSL/TLS implementations. In: Proc. of the 2018 IEEE Int'l Conf. on Software Maintenance and Evolution. Madrid: IEEE Press, 2018. 48–58. [doi: 10.1109/ICSME.2018.00014]

[23] Shastry B, Leutner M, Fiebig T, Thimmaraju K, Yamaguchi F, Rieck K, Schmid S, Seifert JP, Feldmann A. Static program analysis as a fuzzing aid. In: Proc. of the 20th Int'l Symp. on Research in Attacks, Intrusions, and Defenses. Atlanta: Springer, 2017. 26–47. [doi: 10.1007/978-3-319-66332-6_2]

[24] Tang ZS, Tang K, Xue MH, Tian Y, Chen S, Ikram M, Wang TL, Zhu HJ. iOS, your OS, everybody's OS: Vetting and analyzing network services of iOS applications. In: Proc. of the 29th USENIX Security Symp. Boston: USENIX Association, 2020. 2415–2432.

[25] Dacosta I, Mehta N, Metrock E, Giffin J. Security analysis of an IP phone: Cisco 7960G. In: Proc. of the 2nd Int'l Conf. on Principles, Systems and Applications of IP Telecommunications. Berlin Heidelberg: Springer, 2008. 236–255. [doi: 10.1007/978-3-540-89054-6_12]

[26] Li YK, Chen BH, Chandramohan M, Lin SW, Liu Y, Tiu A. Steelix: Program-state based binary fuzzing. In: Proc. of the 11th Joint Meeting on Foundations of Software Engineering. Paderborn: ACM Press, 2017. 627–637. [doi: 10.1145/3106237.3106295]

[27] Jero S, Pacheco ML, Goldwasser D, Nita-Rotaru C. Leveraging textual specifications for grammar-based fuzzing of network protocols. In: Proc. of the 33rd AAAI Conf. on Artificial Intelligence. Honolulu: AAAI Press, 2019. 9478–9483. [doi: 10.1609/aaai.v33i01.33019478]

[28] Rontti T, Juuso AM, Takanen A. Preventing DoS attacks in NGN networks with proactive specification-based fuzzing. IEEE

Communications Magazine, 2012, 50(9): 164–170. [doi: 10.1109/MCOM.2012.6295728]

[29] Pan F, Wu LF, Hong Z, Li HB, Lai HG, Zheng CH. Icefex: Protocol format extraction from IL-based concolic execution. KSII Trans. on Internet and Information Systems, 2013, 7(3): 576–599. [doi: 10.3837/tiis.2013.03.010]

[30] Chen K, Song C, Wang LM, Xu Z. Using memory propagation tree to improve performance of protocol fuzzer when testing ICS. Computers & Security, 2019, 87: 101582. [doi: 10.1016/j.cose.2019.101582]

[31] Han X, Wen QY, Zhang Z. A mutation-based fuzz testing approach for network protocol vulnerability detection. In: Proc. of the 2nd Int'l Conf. on Computer Science and Network Technology. Changchun: IEEE, 2012. 1018–1022. [doi: 10.1109/ICCSNT.2012.6526099]

[32] Li WM, Zhang AF, Liu JC, Li ZT. An automatic network protocol fuzz testing and vulnerability discovering method. Chinese Journal of Computers, 2011, 34(2): 242–255 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.00242]

[33] Ramos SH, Villalba MT, Lacuesta R. MQTT security: A novel fuzzing approach. Wireless Communications and Mobile Computing, 2018, 2018: 8261746. [doi: 10.1155/2018/8261746]

[34] Shapiro R, Bratus S, Rogers E, Smith S. Identifying vulnerabilities in SCADA systems via fuzz-testing. In: Proc. of the 5th Int'l Conf. on Critical Infrastructure Protection. Hanover: Springer Press, 2011. 57–72. [doi: 10.1007/978-3-642-24864-1_5]

[35] Hu ZC, Shi JQ, Huang YH, Xiong JW, Bu XX. GANFuzz: A GAN-based industrial network protocol fuzzing framework. In: Proc. of the 15th ACM Int'l Conf. on Computing Frontiers. Ischia: ACM Press, 2018. 138–145. [doi: 10.1145/3203217.3203241]

[36] Daniel LA, Poll E, De Ruiter J. Inferring OpenVPN state machines using protocol state fuzzing. In: Proc. of the 2018 IEEE European Symp. on Security and Privacy Workshops. London: IEEE, 2018. 11–19. [doi: 10.1109/EuroSPW.2018.00009]

[37] Poll E, de Ruiter J, Schubert A. Protocol state machines and session languages: Specification, implementation, and security flaws. In: Proc. of the 2015 IEEE Security and Privacy Workshops. San Jose: IEEE, 2015. 125–133. [doi: 10.1109/SPW.2015.32]

[38] Banks G, Cova M, Felmetsger V, Almeroth K, Kemmerer R, Vigna G. SNOOZE: Toward a stateful network protocol fuzzer. In: Proc. of the 9th Int'l Conf. on Information Security. Samos Island: Springer, 2006. 343–358. [doi: 10.1007/11836810_25]

[39] De Ruiter J, Poll E. Protocol state fuzzing of TLS implementations. In: Proc. of the 24th USENIX Security Symp. Washington: USENIX Association, 2015. 193–206.

[40] Cai J, Luo JZ, Lei FY. Analyzing network protocols of application layer using hidden semi-Markov model. Mathematical Problems in Engineering, 2016, 2016: 9161723. [doi: 10.1155/2016/9161723]

[41] Pfrang S, Meier D, Fleig A, Beyerer J. A meta model for a comprehensive description of network protocols improving security tests. In: Proc. of the 6th Int'l Conf. on Information Systems Security and Privacy. Valletta: SciTePress, 2020. 671–682. [doi: 10.5220/0009150206710682]

[42] Ma R, Wang DG, Hu CZ, Ji WD, Xue JF. Test data generation for stateful network protocol fuzzing using a rule-based state machine. Tsinghua Science and Technology, 2016, 21(3): 352–360. [doi: 10.1109/TST.2016.7488746]

[43] Li FJ, Zhang LY, Chen DJ. Vulnerability mining of Cisco router based on fuzzing. In: Proc. of the 2nd Int'l Conf. on Systems and Informatics. Shanghai: IEEE, 2014. 649–653. [doi: 10.1109/ICSAI.2014.7009366]

[44] Black Hat USA 2002 Topics and Speakers. 2023. https://www.blackhat.com/html/bh-usa-02/bh-usa-02-speakers.html

[45] Lahmadi A, Brandin C, Festor O. A testing framework for discovering vulnerabilities in 6LoWPAN networks. In: Proc. of the 8th IEEE Int'l Conf. on Distributed Computing in Sensor Systems. Hangzhou: IEEE, 2012. 335–340. [doi: 10.1109/DCOSS.2012.48]

[46] Walz A, Sikora A. Exploiting dissent: Towards fuzzing-based differential black-box testing of TLS implementations. IEEE Trans. on Dependable and Secure Computing, 2020, 17(2): 278–291. [doi: 10.1109/TDSC.2017.2763947]

[47] Yang H, Zhang YQ, Hu YP, Liu QX. IKE vulnerability discovery based on fuzzing. Security and Communication Networks, 2013, 6(7): 889–901. [doi: 10.1002/sec.628]

[48] Hooper M, Tian YF, Zhou RX, Cao B, Lauf AP, Watkins L, Robinson WH, Alexis W. Securing commercial WiFi-based UAVs from common security attacks. In: Proc. of the 2016 IEEE Military Communications Conf. Baltimore: IEEE Press, 2016. 1213–1218. [doi: 10.1109/MILCOM.2016.7795496]

[49] Wang ZQ, Li QQ, Wang YZ, Liu B, Zhang JY, Liu QX. Medical protocol security: DICOM vulnerability mining based on fuzzing technology. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: ACM Press, 2019. 2549–2551. [doi: 10.1145/3319535.3363253]

[50] Cui YP, Yu T, Hu JW. IKEv2 protocol fuzzing test on simulated ASA. In: Proc. of the 2018 IEEE Int'l Conf. on Smart Internet of Things. Xi'an: IEEE Press, 2018. 111–116. [doi: 10.1109/SmartIoT.2018.00-16]

[51] Li JQ, Li SY, Sun G, Chen T, Yu HF. SNPSFuzzer: A fast greybox fuzzer for stateful network protocols using snapshots. IEEE Trans. on Information Forensics and Security, 2022, 17: 2673–2687. [doi: 10.1109/TIFS.2022.3192991]

[52] Hassanshahi B, Lee H, Krishnan P. Gelato: Feedback-driven and guided security analysis of client-side Web applications. In: Proc. of

the 2022 IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering. Honolulu: IEEE Press, 2022. 618–629. [doi: 10.1109/SANER53432.2022.00079]

[53] Hernandez G, Butler KRB. Basebads: Automated security analysis of baseband firmware: Poster. In: Proc. of the 12th Conf. on Security and Privacy in Wireless and Mobile Networks. Miami: ACM Press, 2019. 318–319. [doi: 10.1145/3317549.3326310]

[54] Park J, Jang YH, Park Y. New flash memory acquisition methods based on firmware update protocols for LG Android smartphones. Digital Investigation, 2018, 25: 42–54. [doi: 10.1016/j.diin.2018.04.002]

[55] Pohl J, Noack A. Universal radio hacker: A suite for wireless protocol analysis. In: Proc. of the 2017 Workshop on Internet of Things Security and Privacy. Dallas: ACM Press, 2017. 59–60. [doi: 10.1145/3139937.3139951]

[56] Niedermaier M, Fischer F, von Bodisco A. PropFuzz—An IT-security fuzzing framework for proprietary ICS protocols. In: Proc. of the 2017 Int'l Conf. on Applied Electronics. Pilsen: IEEE Press, 2017. 1–4. [doi: 10.23919/AE.2017.8053600]

[57] Zhang BF, Zhang CB, Xu Y. Network protocol vulnerability discovery based on fuzzy testing. Journal of Tsinghua University (Science and Technology), 2009, 49(S2): 2113–2118 (in Chinese with English abstract). [doi: 10.16511/j.cnki.qhdxxb.2009.s2.017]

[58] Chen DD, Egele M, Woo M, Brumley D. Towards automated dynamic analysis for Linux-based embedded firmware. In: Proc. of the 23rd Annual Network and Distributed System Security Symp. San Diego: Internet Society Press, 2016. 21–24. [doi: 10.14722/ndss.2016.23415]

[59] Kim M, Kim D, Kim E, Kim S, Jang Y, Kim Y. FirmAE: Towards large-scale emulation of IoT firmware for dynamic analysis. In: Proc. of the 2020 Annual Computer Security Applications Conf. Austin: ACM Press, 2020. 733–745. [doi: 10.1145/3427228.3427294]

[60] Qiling Framework. 2023. https://qiling.io

[61] Dantas H, Erkin Z, Doerr C, Hallie R, van der Bij G. eFuzz: A fuzzer for DLMS/COSEM electricity meters. In: Proc. of the 2nd Workshop on Smart Energy Grid Security. Scottsdale: ACM Press, 2014. 31–38. [doi: 10.1145/2667190.2667194]

[62] Zaddach J, Bruno L, Francillon A, Balzarotti D. Avatar: A framework to support dynamic security analysis of embedded systems' firmwares. In: Proc. of the 21st Annual Network and Distributed System Security Symp. San Diego: Internet Society, 2014. 23–26. [doi: 10.14722/ndss.2014.23229]

[63] Muench M, Nisi D, Francillon A, Balzarotti D. Avatar^2: A multi-target orchestration platform. In: Proc. of the 18th Workshop on Binary Analysis Research. San Diego: ISOC Press, 2018. 1–11. [doi: 10.14722/BAR.2018.23017]

[64] Gao ZC, Dong WY, Chang R, Wang YS. Fw-fuzz: A code coverage-guided fuzzing framework for network protocols on firmware. Concurrency and Computation: Practice and Experience, 2022, 34(16): e5756. [doi: 10.1002/cpe.5756]

[65] Chen JY, Diao WR, Zhao QC, Zuo CS, Lin ZQ, Wang XF, Lau WC, Sun MH, Yang RH, Zhang KH. IoTFuzzer: Discovering memory corruptions in IoT through App-based fuzzing. In: Proc. of the 25th Annual Network and Distributed System Security Symp. San Diego: Internet Society Press, 2018. 1–15. [doi: 10.14722/ndss.2018.23159]

[66] Fang KM, Yan GH. Emulation-instrumented fuzz testing of 4G/LTE Android mobile devices guided by reinforcement learning. In: Proc. of the 23rd European Symp. on Research in Computer Security. Barcelona: Springer, 2018. 20–40. [doi: 10.1007/978-3-319-98989-1_2]

[67] Cui BJ, Feng SB, Xiao QS, Li M. Detection of LTE protocol based on format fuzz. In: Proc. of the 10th Int'l Conf. on Broadband and Wireless Computing, Communication and Applications. Krakow: IEEE Press, 2015. 187–192. [doi: 10.1109/BWCCA.2015.42]

[68] Munea TL, Kim IL, Shon T. Design and implementation of fuzzing framework based on IoT applications. Wireless Personal Communications, 2017, 93(2): 365–382. [doi: 10.1007/s11277-016-3322-9]

[69] Biyani A, Sharma G, Aghav J, Waradpande P, Savaji P, Gautam M. Extension of SPIKE for encrypted protocol fuzzing. In: Proc. of the 3rd Int'l Conf. on Multimedia Information Networking and Security. Shanghai: IEEE Press, 2011. 343–347. [doi: 10.1109/MINES.2011.143]

[70] Chandrasekar B, Ramesh B, Prabhu V, Sajeev S, Mohanty PK, Shobha G. Development of intelligent digital certificate fuzzer tool. In: Proc. of the 2017 Int'l Conf. on Cryptography, Security and Privacy. Wuhan: ACM Press, 2017. 126–130. [doi: 10.1145/3058060.3058070]

[71] Bratus S, Hansen A, Shubina A. LZfuzz: A fast compression-based fuzzer for poorly documented protocols. Technical Report, TR2008-634, Hanover: Dartmouth College, 2008.

[72] Yoo H, Shon T. Grammar-based adaptive fuzzing: Evaluation on SCADA modbus protocol. In: Proc. of the 2016 IEEE Int'l Conf. on Smart Grid Communications. Sydney: IEEE Press, 2016. 557–563. [doi: 10.1109/SmartGridComm.2016.7778820]

[73] Xiong Q, Liu H, Xu Y, Rao HY, Yi SW, Zhang BF, Jia W, Deng H. A vulnerability detecting method for Modbus-TCP based on smart fuzzing mechanism. In: Proc. of the 2015 IEEE Int'l Conf. on Electro/Information Technology. Dekalb: IEEE Press, 2015. 404–409. [doi: 10.1109/EIT.2015.7293376]

[74] Tu TF, Zhang H, Qin BQ, Chen Z. A vulnerability mining system based on fuzzing for IEC 61850 protocol. In: Proc. of the 5th Int'l Conf. on Frontiers of Manufacturing Science and Measuring Technology. Taiyuan: Atlantis Press, 2017. 589–597. [doi: 10.2991/fmsmt-17.2017.119]

[75] Park H, Nkuba CK, Woo S, Lee H. L2Fuzz: Discovering bluetooth L2CAP vulnerabilities using stateful fuzz testing. In: Proc. of the 52nd Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks. Baltimore: IEEE Press, 2022. 343–354. [doi: 10.1109/DSN53405.2022.00043]

[76] Li H, Zhang WS, Zhou WF, Su B. A novel vulnerability detection method for ZigBee MAC layer. In: Proc. of the 12th Int'l Conf. on Dependable, Autonomic and Secure Computing. Dalian: IEEE Press, 2014. 121–124. [doi: 10.1109/DASC.2014.30]

[77] Cui BJ, Wang ZY, Zhao B, Liang XB. CG-Fuzzing: A comprehensive fuzzy algorithm for ZigBee. Int'l Journal of Ad Hoc and Ubiquitous Computing, 2016, 23(3–4): 203–215. [doi: 10.1504/IJAHUC.2016.079267]

[78] Fowler DS, Bryans J, Shaikh SA, Wooderson P. Fuzz testing for automotive cyber-security. In: Proc. of the 48th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks Workshops. Luxembourg: IEEE Press, 2018. 239–246. [doi: 10.1109/DSN-W.2018.00070]

[79] Kim S, Jo W, Shon T. A novel vulnerability analysis approach to generate fuzzing test case in industrial control systems. In: Proc. of the 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conf. Chongqing: IEEE Press, 2016. 566–570. [doi: 10.1109/ITNEC.2016.7560424]

[80] Shu GQ, Hsu Y, Lee D. Detecting communication protocol security flaws by formal fuzz testing and machine learning. In: Proc. of the 28th Int'l Conf. on Formal Techniques for Networked and Distributed Systems. Tokyo: Springer Press, 2008. 299–304. [doi: 10.1007/978-3-540-68855-6_19]

[81] Liu XY, Cui BJ, Fu JS, Ma JX. HFuzz: Towards automatic fuzzing testing of NB-IoT core network protocols implementations. Future Generation Computer Systems, 2020, 108: 390–400. [doi: 10.1016/j.future.2019.12.032]

[82] Ma R, Ji WD, Hu CZ, Shan C, Wu P. Fuzz testing data generation for network protocol using classification tree. In: Proc. of the 2014 Communications Security Conf. Beijing: IET Press, 2014. 1–5. [doi: 10.1049/cp.2014.0748]

[83] Wang ZQ, Zhang YQ, Liu QX, Huang TP. Algorithm for discovering SNMP protocol vulnerability. Journal of Xidian University, 2015, 42(4): 20–26, 40 (in Chinese with English abstract). [doi: 10.3969/j.issn.1001-2400.2015.04.004]

[84] Ming L, Zhao G, Huang MH, Pang L, Li J, Zhang JZ, Li D, Lu SB. Remote protocol vulnerability discovery for intelligent transportation systems (ITS). In: Proc. of the IEEE 3rd Int'l Conf. on Data Science in Cyberspace. Guangzhou: IEEE Press, 2018. 923–929. [doi: 10.1109/DSC.2018.00147]

[85] Gu SJ, Song YY, Zhao X, Li WH. Fuzzing test data generation based on message matrix perturbation with keyword reference. In: Proc. of the 2011 Military Communications Conf. Baltimore: IEEE Press, 2011. 1115–1120. [doi: 10.1109/MILCOM.2011.6127448]

[86] Huang Y, Zou QW, Fan KF. Fuzzing test-based vulnerability mining for industrial control network protocol. Journal on Communications, 2018, 39(S2): 181–188 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-436x.2018254]

[87] Gascon H, Wressnegger C, Yamaguchi F, Arp D, Rieck K. PULSAR: Stateful black-box fuzzing of proprietary network protocols. In: Proc. of the 11th Int'l Conf. on Security and Privacy in Communication Systems. Dallas: Springer, 2015. 330–347. [doi: 10.1007/978-3-319-28865-9_18]

[88] Chau SY, Chowdhury O, Hoque E, Ge HY, Kate A, Nita-Rotaru C, Li NH. SymCerts: Practical symbolic execution for exposing noncompliance in X.509 certificate validation implementations. In: Proc. of the 2017 IEEE Symp. on Security and Privacy. San Jose: IEEE Press, 2017. 503–520. [doi: 10.1109/SP.2017.40]

[89] Fan R, Chang YY. Machine learning for black-box fuzzing of network protocols. In: Proc. of the 19th Int'l Conf. on Information and Communications Security. Beijing: Springer Press, 2018. 621–632. [doi: 10.1007/978-3-319-89500-0_53]

[90] Gao ZC, Dong WY, Chang R, Ai CW. The stacked Seq2Seq-attention model for protocol fuzzing. In: Proc. of the 7th IEEE Int'l Conf. on Computer Science and Network Technology. Dalian: IEEE Press, 2019. 126–130. [doi: 10.1109/ICCSNT47585.2019.8962499]

[91] Walz A, Sikora A. Maximizing and leveraging behavioral discrepancies in TLS implementations using response-guided differential fuzzing. In: Proc. of the 2018 Int'l Carnahan Conf. on Security Technology. Montreal: IEEE Press, 2018. 1–5. [doi: 10.1109/CCST.2018.8585565]

[92] Becker S, Abdelnur H, State R, Engel T. An autonomic testing framework for IPv6 configuration protocols. In: Proc. of the 4th Int'l Conf. on Autonomous Infrastructure, Management and Security. Zurich: Springer, 2010. 65–76. [doi: 10.1007/978-3-642-13986-4_7]

[93] Liu DG, Pham VT, Ernst G, Murray T, Rubinstein BIP. State selection algorithms and their impact on the performance of stateful network protocol fuzzing. In: Proc. of the 2022 IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering. Honolulu: IEEE Press, 2022. 720–730. [doi: 10.1109/SANER53432.2022.00089]

[94] Qin SS, Hu F, Ma ZY, Zhao BD, Yin TT, Zhang C. NSFuzz: Towards efficient and state-aware network service fuzzing. ACM Trans.

on Software Engineering and Methodology, 2023, 32(6): 1–26. [doi: 10.1145/3580598]

[95] Sivakorn S, Argyros G, Pei KX, Keromytis AD, Jana S. HVLearn: Automated black-box analysis of hostname verification in SSL/TLS implementations. In: Proc. of the 2017 IEEE Symp. on Security and Privacy. San Jose: IEEE Press, 2017. 521–538. [doi: 10.1109/SP.2017.46]

[96] Newsome J, Song DX. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In: Proc. of the 12th Annual Network and Distributed System Security Symp. San Diego: The Internet Society, 2005. 1–17.

[97] Boyer RS, Elspas B, Levitt KN. SELECT—A formal system for testing and debugging programs by symbolic execution. In: Proc. of the 1975 Int'l Conf. on Reliable Software. Los Angeles: ACM Press, 1975. 234–245. [doi: 10.1145/800027.808445]

[98] King JC. Symbolic execution and program testing. Communications of the ACM, 1976, 19(7): 385–394. [doi: 10.1145/360248.360252]

[99] Howden WE. Symbolic testing and the DISSECT symbolic evaluation system. IEEE Trans. on Software Engineering, 1977, SE-3(4): 266–278. [doi: 10.1109/TSE.1977.231144]

[100] Godefroid P, Klarlund N, Sen K. DART: Directed automated random testing. In: Proc. of the 2005 ACM SIGPLAN Conf. on Programming Language Design and Implementation. Chicago: ACM Press, 2005. 213–223. [doi: 10.1145/1065010.1065036]

[101] Cadar C, Ganesh V, Pawlowski PM, Dill DL, Engler DR. EXE: Automatically generating inputs of death. ACM Trans. on Information and System Security, 2008, 12(2): 10. [doi: 10.1145/1455518.1455522]

[102] Cadar C, Dunbar D, Engler D. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. In: Proc. of the 8th USENIX Conf. on Operating Systems Design and Implementation. San Diego: USENIX Association, 2008. 209–224.

[103] Zhao BD, Li ZM, Qin SS, Ma ZY, Yuan M, Zhu WY, Tian ZH, Zhang C. StateFuzz: System call-based state-aware Linux driver fuzzing. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX, 2022. 3273–3289.

[104] Li QM, Liu YZ, Meng SM, Zhang HR, Shen HY, Long HQ. A dynamic taint tracking optimized fuzz testing method based on multi-modal sensor data fusion. EURASIP Journal on Wireless Communications and Networking, 2020, 2020(1): 1–21. [doi: 10.1186/s13638-020-01734-0]

[105] Clause J, Li WC, Orso A. Dytan: A generic dynamic taint analysis framework. In: Proc. of the 2007 Int'l Symp. on Software Testing and Analysis. London: ACM Press, 2007. 196–206. [doi: 10.1145/1273463.1273490]

[106] Bosman E, Slowinska A, Bos H. Minemu: The world's fastest taint tracker. In: Proc. of the 14th Int'l Workshop on Recent Advances in Intrusion Detection. Menlo Park: Springer, 2011. 1–20. [doi: 10.1007/978-3-642-23644-0_1]

[107] Godefroid P, Levin MY, Molnar D. SAGE: Whitebox fuzzing for security testing. Communications of the ACM, 2012, 55(3): 40–44. [doi: 10.1145/2093548.2093564]

[108] Banabic R, Candea G, Guerraoui R. Finding trojan message vulnerabilities in distributed systems. ACM SIGPLAN Computer Architecture News, 2014, 42(1): 113–126. [doi: 10.1145/2654822.2541984]

[109] Wang TL, Wei T, Gu TF, Zou W. TaintScope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection. In: Proc. of the 2010 IEEE Symp. on Security and Privacy. Oakland: IEEE Press, 2010. 497–512. [doi: 10.1109/SP.2010.37]

[110] Li C, Wei Q, Peng JS, Wang QX. Network software test data generation based on decomposition and reconstruction. Computer Science, 2013, 40(10): 108–113 (in Chinese with English abstract). [doi: 10.3969/j.issn.1002-137X.2013.10.022]

[111] Natella R. STATEAFL: Greybox fuzzing for stateful network servers. Empirical Software Engineering, 2022, 27(7): 191. [doi: 10.1007/s10664-022-10233-3]

[112] Lu J, Yu B. WThreadAFL: Deterministic greybox fuzzing for multi-thread network servers. In: Proc. of the Asia-Pacific Workshop on Networking. Fuzhou: ACM Press, 2022. 1–2.

[113] Muench M, Stijohann J, Kargl F, Francillon A, Balzarotti D. What you corrupt is not what you crash: Challenges in fuzzing embedded devices. In: Proc. of the 25th Annual Network and Distributed Systems Security. San Diego: Internet Society Press, 2018. 1–15. [doi: 10.14722/ndss.2018.23166]

[114] Han H, Oh D, Cha SK. CodeAlchemist: Semantics-aware code generation to find vulnerabilities in JavaScript engines. In: Proc. of the 26th Annual Network and Distributed Systems Security Symp. San Diego: Internet Society Press, 2019. 1–15. [doi: 10.14722/ndss.2019.23263]

[115] Cha SK, Woo M, Brumley D. Program-adaptive mutational fuzzing. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. San Jose: IEEE Press, 2015. 725–741. [doi: 10.1109/SP.2015.50]

附中文参考文献:

[5] 赖英旭, 刘静, 刘增辉, 张靖雯. 工业控制系统脆弱性分析及漏洞挖掘技术研究综述. 北京工业大学学报, 2020, 46(6): 571–582.

[doi: 10.11936/bjutxb2019120008]
[12] 伊胜伟, 张翀斌, 谢丰, 熊琦, 向憧, 梁露露. 基于Peach的工业控制网络协议安全分析. 清华大学学报(自然科学版), 2017, 57(1): 50–54. [doi: 10.16511/j.cnki.qhdxxb.2017.21.010]
[32] 李伟明, 张爱芳, 刘建财, 李之棠. 网络协议的自动化模糊测试漏洞挖掘方法. 计算机学报, 2011, 34(2): 242–255. [doi: 10.3724/SP.J.1016.2011.00242]
[57] 张宝峰, 张翀斌, 许源. 基于模糊测试的网络协议漏洞挖掘. 清华大学学报(自然科学版), 2009, 49(S2): 2113–2118. [doi: 10.16511/j.cnki.qhdxxb.2009.s2.017]
[83] 王志强, 张玉清, 刘奇旭, 黄庭培. 一种简单网络管理协议漏洞挖掘算法. 西安电子科技大学学报(自然科学版), 2015, 42(4): 20–26, 40. [doi: 10.3969/j.issn.1001-2400.2015.04.004]
[86] 黄影, 邹颀伟, 范科峰. 基于Fuzzing测试的工控网络协议漏洞挖掘技术. 通信学报, 2018, 39(S2): 181–188. [doi: 10.11959/j.issn.1000-436x.2018254]
[110] 李程, 魏强, 彭建山, 王清贤. 基于分解重构的网络软件测试数据生成方法. 计算机科学, 2013, 40(10): 108–113. [doi: 10.3969/j.issn.1002-137X.2013.10.022]

喻波(1985－), 男, 博士, 副研究员, CCF 高级会员, 主要研究领域为软件安全, 系统安全.

刘润昊(1998－), 男, 博士生, 主要研究领域为网络协议软件安全, 物联网安全, 漏洞挖掘.

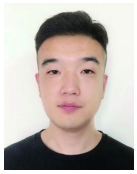
苏金树(1962－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为高性能网络, 互联网体系结构, 网络安全.

卢建君(1998－), 男, 硕士, 主要研究领域为自动化漏洞挖掘, 模糊测试.

杨强(1983－), 男, 博士, 副研究员, 主要研究领域为软件安全分析.

梁晨(1998－), 女, 硕士, 主要研究领域为 Web 安全, 渗透测试, 物联网安全.

黄见欣(1985－), 男, 博士生, 主要研究领域为网络协议, 网络安全, 系统安全.



陈晨(1992－), 男, 硕士, 主要研究领域为软件安全, 网络安全.

盛周石(1993－), 男, 硕士, 主要研究领域为机器学习, 知识图谱.

赵磊(1985－), 男, 博士, 教授, 博士生导师, 主要研究领域为软件及系统安全, 二进制程序的安全分析, 软件漏洞的自动化挖掘和分析.