计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.241100173

网络协议模糊测试技术研究进展

韩陆超[1]　张　伟[2]
1 国家自然科学基金委员会　北京 100085
2 中建电子信息技术有限公司　北京 100123

摘　要　模糊测试作为自动缺陷检测技术之一,通过持续向被测目标输入随机或半随机变异的数据从而导致被测目标发生异常或崩溃的方式来进行测试,近年来挖掘大量基础软件缺陷,是当前软件测试领域的研究热点。文中聚焦针对网络协议软件实现的模糊测试技术工具,针对近年来网络协议模糊测试相关研究成果进行了系统分析和总结。首先,以网络协议模糊测试的基本流程为牵引,分别阐述模糊测试技术在协议报文预处理、测试用例生成、模糊测试执行、测试异常监控等测试阶段的工作原理及其代表性研究工作进展;然后,通过主流协议模糊测试工具在多种网络协议的大规模集成测试应用中,实现了对当前主流网络协议模糊测试工具的评估验证;最后,对网络协议模糊测试技术未来的发展方向和面临的挑战进行了总结与展望。

关键词:模糊测试;漏洞挖掘;网络协议;测试用例生成;协议逆向工程
中图分类号　TP393

Survey on Fuzz Testing Techniques for Network Protocols

HAN Luchao[1] and ZHANG Wei[2]
1 National Natural Science Foundation of China,Beijing 100085,China
2 CSCEC Electronic Information Technology Co.,Ltd.,Beijing 100123,China

Abstract　Fuzz testing,as one of the automatic bug detection techniques,has found a large number of bugs in recent years by continuously inputting random or semi-random variant data to the target under test,leading to anomalies or crashes of the target under test. This paper focuses on fuzz testing for network protocol software implementation,and systematically analyses and summarises the research results related to network protocol fuzz testing in recent years. First,the basic process of network protocol fuzz testing is taken as the traction,and the working principle of fuzz testing technology in the testing phases of protocol message pre-processing,test case generation,fuzz test execution,and test anomaly monitoring and other testing phases are elaborated,as well as the progress of its representative research work. Then,through the application of mainstream protocol fuzz testing tools in the large-scale integrated testing of multiple network protocols,the evaluation and validation of mainstream network protocol fuzz testing tools are realised. Finally,the research results in recent years are analysed and summarised. Then,the evaluation and validation of the current mainstream network protocol fuzz testing tools are achieved through the application of mainstream protocol fuzz testing tools in the large-scale integrated testing of multiple network protocols;finally,the future development direction and challenges of network protocol fuzz testing technology are summarised and outlooked.

Keywords　Fuzz testing,Vulnerability detection,Network protocols,Test case generation,Protocol reverse engineering

1　引言

网络协议是一套预先制定的通信规则。协议双方在进行网络通信时,通过遵循对应网络协议对交互过程的规定和约束,以确保交互过程中信息的安全性和正确性。随着网络协议的飞速发展,网络协议以繁复庞大的体量呈现于各标准、规范文本中,并被植入芯片、操作系统以及各种网络信息设备中,对于国民经济发展具有重大意义,其质量安全必须得到保障。

模糊测试作为当今主流的漏洞挖掘技术之一,通过持续向被测目标对象输入随机或半随机变异的数据,从而导致被测目标发生异常或崩溃,来进行漏洞检测和挖掘[1-2]。模糊测试技术成为了今年软件质量保证领域的研究热点,大量的模糊测试技术研究涌现,需要系统归纳评估以指导当前研究人员未来工作与工程实践选型。当前,模糊测试技术相关综述文章大多以整体的视角对模糊测试技术的发展历程[1]、研究类别[2-3]、模糊测试工具[4-5]、研究挑战[6]或者如物联网领域[7]和车联网[8]等行业特定模糊测试应用等方面进行梳理总结,对针对网络协议的模糊测试相关的系统梳理工作较少。而协议模糊测试是应用最广泛的模糊测试应用领域之一,当前研究人员已提出大量测试优化方法及其实践测试工具,对聚焦网络协议的模糊测试技术发展进行系统性调研,对于协议测试人员工作、保障网络质量安全具有重要意义。

本文重点针对近年提出的网络协议模糊测试技术的相关工作进行归纳与应用评估。首先,概述了网络协议模糊测试技术分类、测试流程等基本概念;然后,以网络协议模糊测试的具体运作流程为脉络,介绍了当前代表性模糊测试关键技术研究进展;最后,对当前主流协议模糊测试工具进行综合评

通信作者:韩陆超(hlc2024@126.com)

估，并对协议模糊测试未来研究方向进行了展望与总结。

2 协议模糊测试概述

2.1 待测目标网络协议分类

模糊测试技术能较好地利用网络协议特性，当前被广泛应用于网络协议的质量保障过程中。协议模糊测试首先需要确定目标协议特性，从而根据目标协议特性选取模糊测试方法。

协议模糊测试将待测网络协议分为无状态协议和有状态协议。这两者之间的区别在于协议数据包的传输顺序是否影响对应服务的执行。其中对于无状态协议，如 HTTP，UDP 协议，通信双方不关心对方的状态，一次请求和响应构成一个独立的事务，不同事务之间没有直接联系。而对于有状态协议，如 TCP，FTP 协议，下一次传输的协议数据包和本次传输的数据包需要遵循一定的顺序，以此完成相应的服务。因此，有状态协议通常需在通信两端维持一个状态机，根据收到的协议数据包进行对应的操作和状态迁移。

2.2 基于用例生成的协议模糊测试

模糊测试工具按测试用例的生成方式主要分为变异式协议模糊测试和生成式协议模糊测试。

1）变异式协议模糊测试。变异式协议模糊测试可以在无需了解目标协议具体规范的情况下，对已有合法数据样本应用变异算法，不断修改其中的部分数据段，从而自动化生成大量变异的测试数据以供后续测试使用。传统的变异式协议模糊测试工具大多采用随机变异策略，该方法实现简单、自动化程度高，在发展初期发挥了一定的作用。但随机变异产生的测试用例存在规模庞大、结构性不强方面的不足。而网络协议通常严格要求输入数据的结构和格式，不符合协议格式要求的数据包大多被直接丢弃。因此，变异式网络协议模糊测试工具测试覆盖率相对较低，无法进行有效漏洞挖掘。相比之下，生成式协议模糊测试工具有明显的优势。

2）生成式协议模糊测试。生成式协议模糊测试通常依赖人工编写针对目标协议的测试脚本，创建包含数据模型和状态模型的网络协议测试模型，以便自动生成更具针对性和结构化的测试数据。然而，传统生成式协议模糊测试技术依赖人工分析建模，要求测试者对目标协议的数据结构和具体规约有足够的认识和理解。测试建模过程不仅耗时耗力，且需要使用者针对不同的目标协议进行完整的分析和模型构建。虽然漏洞检测过程快速准确，但数据模型构建部分的工作可重用性不高。其次，此方法仅适用于已知网络协议的模糊测试，无法对未知网络协议进行分析检测。

2.3 基于协议状态的协议模糊测试

按对待测网络协议实体内部运行时情况的可获取程度与测试反馈方式，模糊测试工具可分为黑盒协议模糊测试和灰盒协议模糊测试。

1）黑盒协议模糊测试。黑盒协议模糊测试技术无法通过观察协议实体内部执行信息判断其当前状态，故常常依赖引导类报文确定实体状态或将协议引导至相应状态，以便针对确定状态生成相应类测试用例。引导类报文指可被目标程序正常接收处理，引起协议实体正常状态迁移并输出期望响应报文的一类合法报文。但引导类报文仅作为控制测试流程走向的辅助类报文使用，无法进行实际的漏洞挖掘工作。

2）覆盖率引导的灰盒协议模糊测试。灰盒模糊测试通常在测试执行过程中获取协议实体内部执行信息，以状态测试覆盖率为衡量标准，以最大化测试覆盖率为测试目标引导模糊测试过程，在有状态网络协议测试过程中应用广泛。其中，覆盖率指在测试过程中测试对象被覆盖到的数目占总数的比例。高覆盖率也常常意味着发现更多隐藏漏洞的高可能性，因此众多相关研究常以覆盖率为反馈信息指导数据生成阶段的进化，以期得到覆盖率的提升。基于反馈机制的网络协议模糊测试技术可在迭代测试过程中进行自我学习，动态调整模糊变异方案，以不断修剪输入数据的搜索空间，使模糊测试工具关注更可能探索新执行空间的输入子空间，挖掘未知零日漏洞。

2.4 协议模糊测试的基本流程

协议模糊测试的基本流程可分为以下阶段：预处理、数据生成、测试执行、异常监控。下文基于各个测试阶段对协议模糊测试研究现状进行介绍分析。

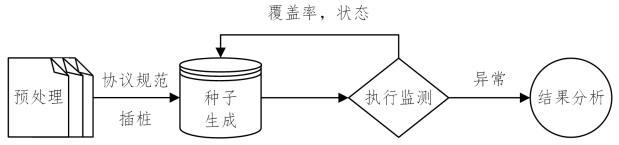


图 1 模糊测试的流程

Fig. 1 Process of fuzz testing

2.4.1 预处理

在协议模糊测试预处理阶段，需要针对协议模糊测试工具类型准备测试输入。其中，变异式协议模糊测试仅需要典型报文作为初始种子，输入构建较为简单。

对于生成式协议模糊测试，当前 SPIKE，Peach，Sulley 等典型工具及其派生工具已发展出成熟的数据构造 API 和协议描述语言，用于预处理阶段的测试输入建模，允许测试者根据对目标协议的掌握自定义协议数据模型与状态模型。其中，协议数据模型可包含字段长度、类型属性、字段间依赖关系和语义等详细信息。模糊测试工具可根据字段属性选取对应变异策略。如对于整数型字段的变异一般采用边界值操作，或有符号数据与无符号数据之间的转换，试图挖掘缓冲区溢出等漏洞。对于字符串型字段，利用超长字符串或格式化字符串进行替换，尝试触发服务程序的异常处理等等。

生成式协议模糊测试工具 SPIKE[9] 首次提出基于块的协议描述方式，结构化地构造协议的数据模型，允许用户根据需求灵活创建针对不同协议的模糊测试工具。此外，SPIKE 提供大量内置字典，包含更容易引发协议实体异常的数值或字符串等，用于替换、覆盖特定字段。相较传统变异式协议模糊测试工具，SPIKE 生成的测试用例结构性更强，数据通过率显著提高。此后发展的生成式协议模糊测试工具如 Peach[10]，Sulley[11] 等大多以基于块的构造方式为基础进行扩展开发。其中，通用模糊测试工具 Peach 利用 XML（Extensible Markup Language）语言对目标协议的数据结构进行定义，描述协议中各字段的数据类型和长度等信息，生成 Peach Pits 配置文件以指导测试用例的生成。

针对有状态协议，Peach 提出状态模型概念，允许用户按需配置状态机。Sulley 提供一套完善的数据原语用于定义协议模型。模糊测试工具根据有状态协议规则，自动生成会话图指导模糊测试有序进行。编写测试脚本的方法逐步发展成

熟，极大地简化了用户自定义模糊测试工具的过程。多数生成式有状态协议模糊测试工具在已知协议状态模型的基础上，可以控制模糊测试工具遍历测试目标协议的每个状态。

2.4.2 种子生成

预处理之后，变异式模糊测试根据预处理阶段得到的初始种子数据选取生成测试用例的具体变异策略进行种子生成，用于后续测试执行；生成式模糊测试基于数据模型生成种子，用于后续测试执行。在模糊测试过程中，测试用例的质量直接决定了漏洞挖掘的效率和能力，因此种子生成是最重要的环节之一。传统网络协议模糊测试工具主要采用随机变异策略，产生海量测试用例以增加检测到漏洞的概率。在漏洞挖掘过程中，由于缺少输入数据的结构语义信息等，这种暴力傻瓜式的测试技术常常产生大量无效测试报文，不仅浪费测试资源还会拖慢测试进度。因此，研究人员当前致力于从各个角度对目标协议进行特征提取，制定合适变异策略，以提高测试用例的生成质量。由于网络协议通常对输入数据格式要求严格，因此在生成测试用例时，模糊测试工具若根据一定的约束规则制定恰当的变异策略，则可在很大程度上提高测试用例的通过率。

2.4.3 测试执行

模糊测试过程中的测试执行模块主要负责执行测试用例，即通过某种方式自动将测试用例输入目标程序中以进行测试。该阶段一般与数据生成阶段并行进行。基于网络协议的交互特性，网络协议模糊测试过程中执行测试用例最常见的方式是通过与目标协议实体进行网络 Socket 连接，即“IP 地址＋端口”的方式进行协议绑定，发送测试用例至目标协议实体。协议实体接收测试用例，并根据具体解析情况执行相关操作。其中，目标实体的 IP 地址和端口号等信息需人工配置。然而，基于网络 Socket 连接执行测试用例的方式受限于网络带宽、连接数量等因素，限制了模糊测试的吞吐率。

2.4.4 异常监测

模糊测试工具需要利用系统原有的异常机制或自定义异常范式，监测种子执行以指导测试过程。反馈机制主要通过对目标程序进行实时监控并记录相关信息，来分析推断输入数据和程序行为之间的关系或对程序执行产生的影响等，然后将分析结果量化为某一度量指标，并在测试迭代中作为反馈信息以指导变异方案的优化。若无法进行有效及时的异常监测和记录，那么整个模糊测试过程将变得毫无意义，因此异常监控模块的研究与发展也相对丰富且成熟。监控手段依据获取信息的方式不同主要分为以下 4 种。

1）基于输入输出的异常监控：由于无法获取目标程序源码，黑盒模糊测试框架主要通过观察目标程序的输入输出，以分析协议实体的行为或状态是否异常。基于输入输出的异常监控方式实现简单，符合网络协议的交互特性，可有效定位目标协议的逻辑漏洞等，是黑盒网络模糊测试工具最常用的异常监控方式。然而，仅通过观察目标程序输入输出流量推断目标程序异常行为的方式往往需要人工定义异常行为模式，且由于网络延时等因素很难做到实时监控，可导致漏报情况，影响对模糊测试工具漏洞挖掘效率的判断。

2）基于调试的异常监控：基于调试的异常监控方式主要指在调试模式下启动目标程序，在测试过程中对程序内部运行情况进行收集和记录，可达到对异常进行及时检测和精准定位的目的。在协议模糊测试过程中，模糊测试工具和目标程序往往是两个独立的系统。模糊测试工具通过与目标主机中的调试器通信以及断点设置，可获取异常发生时目标程序的执行地址、各寄存器值等重要信息，以便后续定位和分析异常漏洞。基于调试的异常监控方式因其高度可操作性，得以广泛应用。

3）基于插桩的异常监控：插桩技术在保证目标程序原有逻辑完整性的基础上，在程序中特定位置插入探针。基于插桩的异常监控方式通过探针的执行，收集程序运行时的动态上下文信息，以分析程序的异常行为和特征。上述基于反馈机制的数据生成优化大多是通过插桩技术来完成覆盖信息采集工作。基于插桩的异常监控方式在监测内存泄漏、内存溢出等不会直接导致程序崩溃或逻辑行为异常的安全漏洞方面，有独特的优势。常见的基于插桩的动态检查工具有 Purify，Valgrind，AppVerifier，AddressSanitizer 等。基于插桩的异常监控方式基于业务调用上下文进行分析，无需重放，检测准确率高，误报率极低。但对程序进行插桩以分析实时信息的方式可能导致系统性能有所降低。

4）基于快照的异常监控：在虚拟环境下运行目标程序，不仅可控制目标程序的执行过程，还可对异常情况进行快照保存。测试者可根据快照信息分析程序运行情况和异常行为，还可基于快照快速重现目标协议测试状态，避免了重建测试过程导致的开销。

3 协议模糊测试技术研究进展

3.1 协议模糊测试输入建模优化

3.1.1 基于状态机的协议状态生成优化

当前，大量研究人员将注意力集中在优化协议迁移测试路径上，进而减少冗余数据的生成。有限状态机 FSM（Finite State Machine）常被用于有状态网络协议报文交互过程的建模，以形式化地描述网络协议实体的状态迁移情况。基于协议有限状态机，很多研究[12-14]引入图遍历算法来求解优化协议状态迁移测试路径。Zhang 等[12]将有状态协议实体的迁移状态遍历过程转化为有向图中的邮路问题，求解遍历所有协议状态迁移的最短路径，以快速有效地对协议的各个状态迁移进行测试。Li 等[14]应用深度优先搜索算法进行状态遍历，并设置最大遍历深度以防在遍历复杂协议时效率低下。而对于协议状态图存在循环路径的情况，Li 等[13]应用 Floyd 算法，将协议状态机转化为有向无回路图。

生成式协议模糊测试工具在对有状态协议进行模糊测试时，其有效性很大程度取决于给定状态模型的完整性，无法在测试过程中通过自学习以拟合实际的协议状态模型。针对这一问题，研究人员应用模型学习算法如经典的 L^* 算法[15]及其改进算法 TTT 算法[16]等，通过模拟观察协议实体的输入输出序列，来动态推断构建协议实现的有限状态机 Mealy Model。而以 SGPFuzzer[17]，AFLnet[18]，PNFUZZ，StateAFL 为典型的部分研究，在动态推断协议状态的同时保留对应的消息序列以及状态转移情况，以自动构建协议的状态模型。在持续的测试迭代过程中，模糊测试工具利用反馈的状态覆盖率指导生成新的消息序列以扩展新的状态，不断完善协议状态模型。

3.1.2 基于协议逆向的未知协议建模

当前协议模糊测试工作大多基于已知协议，难以处理私

有协议等未知协议的测试任务。为此,研究人员基于网络协议逆向分析技术,利用网络流量推断网络协议的结构特征,用于测试输入建模。该类方法通过 Wireshark,Tcpdump 等抓包工具收集协议交互流量,并应用报文聚类、多序列对比和特定域识别等算法进行分析处理,自动化识别各数据域并推断其数据类型或语义信息,最终得到目标协议的通用数据格式[17]。其中,Blumbergs 等[19]提出位感知的网络协议模糊测试框架,结合逆向工程技术将抓包文件转换为二进制表示文件并标识位字段的可变异属性,并引入香农熵和频繁项集挖掘算法 LogHound 以学习协议特征结构。APREFuzz[20]将协议消息看作字节序,应用多变点检测算法根据字节序的统计特征识别协议字段边界,并通过基于位置的发生概率测试进行进一步分析,以识别关键字字段、数据字段和不确定字段。之后涉及到的机器学习技术也大多基于网络追踪的逆向技术进行处理学习。然而,基于网络追踪技术的分析结果往往对捕捉到的网络流量有较强的依赖性,且无法处理对网络协议报文进行加密传输的情况。

3.1.3 基于机器学习的协议测试建模

研究人员开始思考如何将网络协议模糊测试过程中的协议模型构建工作从人工方式转为机器自学习方式,通过机器学习提升协议模糊测试漏洞挖掘的性能。Snipuzz[21]引入层次聚类算法分析协议交互过程中收集的实时响应报文,以寻找数据中内在分布结构及描述对象关系信息,推断出协议数据中代表相同语义的消息片段,从而优化测试建模;PN-FUZZ[22]则利用层次聚类算法实时推断协议实体状态,以有效指导测试用例生成,避免状态不符的无效交互,但以上聚类学习的引入也带来了不可避免的时耗问题。Seq2seq 模型是编码器解码器的架构,可基于上下文信息对输入输出进行建模,LSTM(Long Short-Term Memory)擅长学习时序数据中的长期依赖关系,因此基于 LSTM 的 Seq2seq 模型常用来学习提取有状态协议的数据格式和状态机,以指导生成模糊测试用例[23-26]。在此基础上,为得到更完整的协议特征,Gao 等[24]引入注意力机制对基于 LSTM 的 seq2seq 模型进行改进,但也导致了计算量的明显增加。Jero 等[27]提出引入轻量级 Zero-Shot Learning 技术,通过分析协议 RFC 文档文本与协议字段和属性的相似度,建立特征与语义空间之间的映射,对未知协议进行语法提取。Hu 等[28]提出通过准循环神经网络 QRNN 学习网络协议样本得到协议的相关结构特征,用于过滤无效测试用例。然而,当前基于机器学习的协议测试建模大多以协议流量为数据集,其学习结果很大程度取决于数据集质量,且面对新协议时需重新训练模型。

3.2 协议模糊测试种子质量精化

3.2.1 基于程序分析的种子变异优化

在实体程序源码或其二进制可执行文件可得的情况下,可以应用程序分析技术或程序逆向技术提取特定协议字段的语义信息(合法值的变异范围、数据字段对程序执行的影响等)。基于语义的变异策略在很大程度上减小了生成模糊数据的盲目性,加速了漏洞挖掘过程。

田一峭[8]考虑到不同字段语义对其所提供网络服务的重要性的差异,如对协议地址字段的篡改可能构成伪冒攻击,对数据长度字段的变异可能导致接收方处理程序的缓冲区溢出漏洞等等,通过分析协议脆弱点即可能导致协议实体崩溃的协议字段及其变异类型,将漏洞信息提取成字段形成语料库,并在此基础上提出基于权重的变异策略。该策略以语料库为依据赋予各字段不同的变异权重,权重大的字段可以优先产生更多的变异数据。HFuzz[29]学习协议文档制定相关规则,构建消息结构树(Message Structure Tree),并结合层次感知的变异策略指导测试用例的生成。SPFuzz[30]提出三层变异策略,分别对消息头部、内容和序列进行变异再重组消息,以弥补单维变异策略的不足。Li 等[31]利用逆向工具 IDA Pro 分析定位目标程序中敏感函数和常量,制定启发式变异规则,着重对发现漏洞至关重要的字段进行突变。同时根据分析得到相关参数矩阵,赋予各类测试用例不同优先级,优化测试用例的传输顺序。Polar[32]结合静态分析和轻量级动态污点分析技术对工控协议进行模糊测试,通过定位分析目标程序中功能代码和安全敏感函数,自动提取协议数据的语义信息,对筛选出的合法固定的字段值集不进行变异,避免盲目穷举导致的不必要能耗。Diane 等[33]融合动静态分析技术以定位程序控制流图中的重要函数,从而产生有效但未严格约束的测试用例,以通过数据验证阶段,并最大化代码测试覆盖率。

3.2.2 基于相似性度量的种子生成优化

分析实体程序对协议数据包的处理过程,可将其简化为 3 个步骤:语法解析、语义解析和程序执行。有效的测试用例需要依次通过实体程序的各级筛选验证,以最终进入程序的执行阶段,进行深层的漏洞挖掘工作。为提高测试用例的有效性,需要将测试用例和合法协议报文的差异性保持在一定的范围内,利用两者之间的相似度顺利通过前期的验证阶段。

SeqFuzzer[25]应用基于 LSTM 的 seq2seq 模型中编码器提取协议特征,并指导解码器生成有效的测试用例。其中使用 N-gram 衡量真实数据与畸形数据之间的相似度,在不断地迭代训练后产生以假乱真的畸形测试用例。为进一步提升数据生成模块的稳定性,BLSTM-DCNNFuzz[26]结合生成对抗模型 GAN,其生成器用于生成类似原始种子数据的测试用例,判别器则用于区分判断测试用例的合法性。两个网络在相互博弈进化中使生成器变异生成以假乱真的畸形数据。针对有状态网络协议,Pulsar[34]基于捕获的协议抓包文件,应用聚类算法提取协议结构特征,应用马尔可夫模型动态学习目标协议的有限状态机。在此基础上,针对马尔可夫模型的每个状态,关联一系列相关的协议格式定义模版。利用编辑距离 Levenshtein Distance 评估当前状态可接收消息和相关协议格式定义模版之间的相似度,以挑选出最相近的定义模版指导当前状态测试用例的生成。使用这种方式增加当前状态的测试用例执行通过率,以开展高效的漏洞挖掘工作。ICP-Fuzzer[35]应用 LSTM 模型和 Needleman-Wunsch 序列对比算法来自动定义协议数据的可变异区域,并结合有效的变异策略提高了测试用例生成质量。

3.3 基于协议状态的协议模糊测试

3.3.1 基于实体状态的测试执行性能优化

对有状态网络协议进行模糊测试时,往往需要根据协议实体的当前协议状态选取需生成的测试用例集类型,以避免大量无效交互。而如何确定协议状态成为了有状态模糊测试工具生成测试数据的重要一步。

为改善滥用辅助类报文导致的测试时间开销问题,近年来许多研究倾向于使用对目标实体响应情况进行实时监控分

析的方式来确定实体当前状态或状态转移情况，以动态调整后续的测试用例传输顺序[12-13,18]。此类方法有效减少了辅助类报文的使用，提高了有效测试用例在测试用例集中的占比。为加速对响应报文的分析过程，Pnfuzz[22]对收集到的响应报文应用层次聚类，自动推断协议实体状态从而根据协议实体状态选择变异的测试用例类型。与上述通过分析协议流量推断实体状态的做法不同，StateAFL[36]通过分析每次独立模糊测试交互过程中目标程序的长期内存信息推断实体状态的方式来达到更准确的效果。aBBRate[37]通过计算比较往返平均吞吐量的变化确定协议状态，提高了针对基于拥塞控制算法BBR的TCP协议的测试效果。

3.3.2 基于硬件仿真的测试执行性能优化

将模糊测试技术应用到物联网终端等实际硬件环境时，会受限于其实际硬件配置等因素，无法达到理想的测试吞吐率。多项研究成果表明，对物联网设备采用全仿真系统执行测试数据，可很大程度上提升吞吐量。IoTHunter[38]利用仿真器Avatar2结合固件镜像完成全仿真。FIRM-AFL[39]在全仿真系统的基础上，启用POSIX兼容固件结合增强过程仿真技术，进一步提升测试执行效率和兼容性。然而重复输入的模糊数据可能会导致FIRM-AFL系统陷入无限循环或重启。为提高仿真系统的稳定性，FIRM-COV[40]提出优化过程仿真系统。一般情况下，在用户模式仿真执行目标程序，异常发生时转换到全仿真系统进行处理。BaseSAFE[41]则通过选择性地仿真部分蜂窝基带，提高其测试吞吐率，优化运行性能。该方法同样适用于低级内核系统和固件。虚拟技术和仿真技术的发展，为网络协议模糊测试技术在物联网领域的应用带来了极大便利。

3.3.3 协议模糊测试反馈机制的构建与优化

当前部分研究人员在已有主流协议模糊测试工具基础上扩展其覆盖率反馈机制，或优化其覆盖率评估方式，以提高模糊测试效果。Peach*[42]在Peach的基础上进行扩展，利用轻量级插桩技术计算模糊测试过程中输入数据导致的代码覆盖率，以此作为语义信息反馈到下一轮的变异过程。然后，Peach*将输入数据拆分为片段作为语料库，结合语义感知的生成策略生成高覆盖率的测试用例，以更快的速度达到同等的漏洞挖掘效果。同样利用插桩技术计算覆盖率信息，Pnfuzz[22]直接以覆盖率为导向选取种子数据，应用AFL变异策略生成协议请求数据。FuSeBMC[43]以函数覆盖率为指标，使用基于路径的符号执行以扩展测试范围至未覆盖到的函数。HFuzz则构建了协议的消息结构树，并应用调度算法分配更多资源给触发更多测试路径的变异层次，以集中变异有效协议数据域。TCP-Fuzz[44]应用基于依赖的二维变异策略，依据消息间依赖性产生消息序列以及考虑系统调用间依赖性生成系统调用序列，并以分支转换覆盖率为导向不断优化生成数据质量。为得到进一步的性能提升，除了分析覆盖率外，VulFuzz[45]应用程序分析技术对目标程序中敏感函数进行加权处理，然后利用插桩技术得到输入数据的路径覆盖率和权重函数的调用次数，并以此作为反馈信息赋予各变异策略不同权重。Wen等[46]利用动态分析得到目标程序中的函数和基本块地址，并使用调试器在函数和基本块处设置断点以计算代码覆盖率，同时引入遗传算法设计以覆盖率为导向的适应度函数不断筛选出高覆盖率的测试用例。

在网络物理系统类异构环境下进行网络协议模糊测试，通过应用程序分析技术计算覆盖率的方式显然是困难且不合适的。因此，部分研究引入了Delta Time概念[47]，通过分析目标程序的输入输出推断输入数据是否触发了新的执行路径，并以此为反馈信息更新待变异的种子数据池。该方式使模糊测试工具更可能识别较少被执行的路径。但Delta Time很容易受到操作系统配置和网络抖动的影响而出现偏差，故可作为折中的测试方案应用于测试环境复杂的情况。

针对有状态网络协议，AFLnet，PNFUZZ，StateAFL均通过使用保留消息序列的状态覆盖和代码覆盖信息，将模糊测试过程引导至更广更深的测试空间，快速高效地挖掘潜在的协议漏洞。此外，覆盖率还可作为关键指标显示测试进度。

3.4 测试执行异常监控优化

异常监控主要负责监控目标的异常行为，并记录触发异常的测试用例。在模糊测试过程中，对异常的检测、跟踪和分类反映出模糊测试漏洞挖掘能力。如何对检测结果进行去重、自动化整合崩溃检测结果，实现异常监控模块的自动化，将测试人员从繁复的监视工作中解脱出来，是模糊测试技术实用化的一个重要目标。研究人员在异常监控构建与优化方面开展了大量研究。

3.4.1 基于输入输出的异常监控优化

田一峭[8]和Li等[14]通过间隔发送TCP探测包，尝试与目标程序指定端口建立TCP三次握手连接，以判断是否因程序异常导致该端口关闭。针对有状态协议的状态迁移特性，协议模糊测试工具可通过观察目标实体的响应情况，具体地，若模糊测试工具或协议客户端无法及时接收响应报文或收到非期望响应报文，则可推断目标实体发生崩溃或异常状态迁移[12-13,30,35]。

3.4.2 基于调试的异常监控优化

成熟的模糊测试框架Peach和Sulley等均提供基于调试的异常监控方式，且近年来许多涌现的协议模糊测试工具多基于前者进行开发调整。其中，Sulley基于调试器PyDbg开发进程监控代理，监控目标程序状态并记录异常崩溃信息到日志文件以便后续漏洞重现。在此基础上，Sulley可以对发现的异常漏洞进行检测、跟踪和分类。此外，Sulley结合自动化虚拟技术，将目标程序运行在VMware等虚拟环境下，通过虚拟控制代理VMControl执行目标程序的启动、停止和记录快照等操作。当目标程序在模糊测试过程中发生崩溃时可对其立即重启，一次性无缝运行数日，持续检查目标程序对模糊输入的异常反应。针对物联网协议实现的异构环境，Fwfuzz[48]结合反汇编框架Capstone和GDB进行异常监控。

3.4.3 基于插桩的异常监控优化

Polar使用动态数据流分析工具LLVM DataFlowSanitizer进行插桩，追踪程序运行期间的标签数据传播，收集测试输入导致的程序数据流，以实现准确的异常监控。Fan等[23]则利用AppVerifier以实时监测目标程序运行过程中缓冲区溢出等内存损坏漏洞，同时将安全漏洞定位到代码行。TCP-Fuzz利用AddressSanitizer监控运行时内存访问情况，以检测内存漏洞。同时集成数据检测器和差分检测器，检测导致TCP堆栈数据传输错误的语义错误。

3.4.4 基于快照的异常监控优化

Sanislav[49]结合QEMU和VirtualBox的优势，优化快照

功能。根据快照信息重新执行异常数据并确认该异常是否误报，判定发现的异常是否为一个可利用、可重现的安全漏洞。而相比于传统虚拟机，Casteur 等[50]认为部署 Docker 是更轻量化、更高效的选择。与虚拟技术的结合，加速了模糊测试过程，使模糊测试的自动化更为健全。而对于物联网领域，目标实体程序常部署于低级内核系统或嵌入式固件中。BaseSAFE 引入仿真技术，基于快照的模糊测试引入了新的精度级别并实现了高覆盖率和高执行速度。

4 典型协议模糊测试工具评估分析

为探究当前网络协议模糊测试工具的实际效果与部署应用开销，本文部署应用了主流生成式与变异式协议模糊测试工具，并针对当前主流网络协议进行了测试评估。本章首先介绍了实验环境、测试工具、测试目标协议、评估指标等内容，然后对实验结果进行了介绍与对比，最后归纳总结了当前主流工具部署应用过程的经验。

4.1 实验环境

本实验运行在配备 AMD Ryzen 9 7945HX with Radeon Graphics CPU 的机器上。该机器拥有 32 个逻辑内核，运行频率为 2.5GHz，主内存为 16GB。本实验使用的操作系统为 Ubuntu 20.04.2 LTS。为保证数据真实有效，将每个实验独立进行 10 次，每次运行 24h。

4.2 测试工具

本文一共选取了 6 种目前最为流行和先进的模糊测试工具，其中包含 3 种生成式模糊工具和 3 种变异式协议模糊测试工具。生成式协议模糊测试工具选取了工业界运用最为广泛的 Peach 和基于 Peach 改进的工具 Peach* 以及 SPFuzz[51]，变异式协议模糊测试工具选取了基于 AFL 修改优化的主流协议模糊测试工具 AFLnet、ChatAFL 以及 AFL 的改进版本 AFLnwe，支持通过网络套接字进行模糊测试。

4.3 待测网络协议

如表 1 所列，本文选取了 9 个网络协议实现作为测试目标。这些协议实现覆盖了文件传输、域名解析、加密通信、Web 服务、物联网通信、实时数据分发、消息队列传输等领域。

表 1 测试目标

Table 1 To-be-tested target

协议	协议实现	描述
FTP	LightFTP	轻量级文件传输服务器
DNS	Dnsmasq	轻量缓存域名系统代理服务
SSL	OpenSSL	TLS/SSL 加密通信软件包
DDS	cyclonedds	开源高性能 DDS 协议实现
MQTT	mosquitto	MQTT 协议服务器开源实现
ZMTP	libzmq	基于消息队列的多线程库
SMTP	exim	邮件传输代理服务器
RTSP	live555	开源实时流媒体服务项目
SIP	kamailio	开源会话初始协议服务器

4.4 评估指标

结合当前的研究，本文选取了 3 个最具代表性的评估指标来比较模糊测试工具的实际性能。其中，分支覆盖是模糊测试中最通用且常见的评估指标，状态覆盖是协议模糊测试最具特色的评估指标，而运行速度则是衡量模糊测试工具效率的最具代表的评估指标。

1)分支覆盖：表示测试用例覆盖的代码分支数与代码总分支数的比例。

2)运行速度：达到相同分支覆盖数需要运行的时间。在实验中，本文设置的相同分支覆盖数需要运行的时间为 Peach 实现最大分支覆盖所花费的时间以及 SPFuzz 达到相同覆盖数所花费的时间。

4.5 实验结果

4.5.1 模糊测试工具分支覆盖

如表 2 所列，本文对比了 3 种不同的生成式协议模糊测试工具在多种协议实现上的分支覆盖数。实验结果表明，PFuzz 在 5 种协议实现上(lightftp，openssl，cyclonedds，mosquitto 和 libzmp)的覆盖分支数均高于 Peach 和 Peach*。尤其是在 mosquitto 和 libzmp 上，SPFuzz 的覆盖分支数相比 Peach 分别提升 26.3% 和 9.2%，相比 Peach* 分别提升 16.7% 和 11.4%，测试效果提升尤为显著。SPFuzz 测试效果优于其他工具的原因如下。因为 SPFuzz 是一种基于状态路径的并行模糊测框架，利用协议状态和数据模型生成有状态路径，然后将这些路径划分为离散任务，并根据复杂性和多样性均衡分配工作负载，从而提高了模糊测试工具对协议实现的覆盖分支数。而 Peach 和 Peach* 主要依赖静态配置和预定义的路径生成策略，任务分配相对静态，可能导致某些测试路径的重复性高，一些路径被忽视，影响整体覆盖效果。

表 2 生成式协议模糊测试工具分支覆盖结果

Table 2 Branch coverage of the generated-based fuzzers

协议实现	AFLnet	AFLnwe	ChatAFL
LightFTP	353	162	110
Dnsmasq	904.5	807	—
OpenSSL	10104	6357	—
exim	2909	1092	3644
live555	2753	2526	2881
kamailio	9830	8236	9924
forked-daapd	2288	1675	2243

如表 3 所列，本文对比了 3 种不同的变异式协议模糊测试工具在多种协议实现上的分支覆盖数。

表 3 变异式协议模糊测试工具分支覆盖结果

Table 3 Branch coverage results of the mutation-based fuzzers

协议实现	Peach	Peach*	SPFuzz
LightFTP	173	141	174
Dnsmasq	892	900	893
OpenSSL	6069	6174	6222
cyclonedds	23168	22747	23852
mosquitto	5186	5615	6552
libzmq	7920	7759	8647

实验结果表明，AFLnet 在大多数测试用例中表现凸出，其在 lightftp，openssl，exim，openssh，dcmtk，live555，kamailio 以及 forked-daapd 协议实现上的覆盖率均优于 AFLnwe。尤其在 OpenSSL 与 exim 上，AFLnet 的覆盖分支数相比 AFLnwe 分别提升了 58.9%和 166%，测试效果提升尤为显著。这是因为 AFLnet 使用协议模型来生成测试用例，可以在网络协议的各个状态之间进行全面的测试，而 AFLnwe 采用智能输入生成策略来测试网络服务的各种情况，更侧重于模拟实际网络请求和服务交互。其中，ChatAFL 在特定协议如 Exim，Live555 和 Kamailio 上的覆盖分支数均高于 AFLNet，其在覆盖分支数上分别提升了 25.3%，4.6%和 1%，主要因为 ChatAFL 专门针对这些协议进行了优化，能够深入理解和处理这

些协议的复杂特性，包括协议的状态管理、消息格式和状态机行为，使得 ChatAFL 能生成更符合协议规范的测试用例，从而更全面地覆盖协议的各个分支。

4.5.2 模糊测试工具测试执行速度

本文对多个协议实现进行了并行模糊测试，并比较其运行速度。具体来说，为了实现并行机制，本文将每个工具在 4 个模糊实例下连续运行 24 h，每个实验重复执行 10 次，以确定结果的统计显著性，最终代码覆盖情况如表 4、表 5 所列，测试过程中的代码覆盖率动态变化情况如图 2 所示。

表 4 模糊测试工具在 24 h 内实现的平均代码分支数

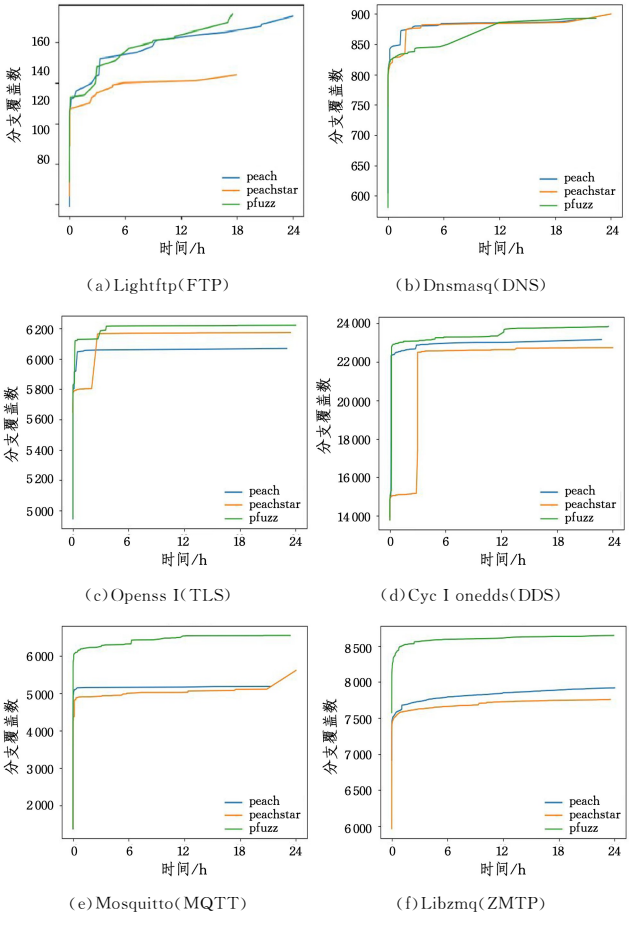
Table 4 Average branch coverage by different fuzzers in 24 h

协议实现	Peach-p	SPFuzz	分支覆盖提升/%
LightFTP	173	174	0.60
Dnsmasq	892	893	0.10
OpenSSL	6069	6222	2.50
cyclonedds	23168	23852	2.90
mosquitto	5186	6552	26.30
libzmq	7920	8647	9.20

表 5 模糊测试工具 24 h 内实际平均执行测试时间

Table 5 Average fuzzing times by different fuzzers in 24 h

协议实现	Peach	SPFuzz	速度提升
LightFTP	81925	59784	1.3X
Dnsmasq	86401	68828	1.2X
OpenSSL	70563	669	105.4X
cyclonedds	81581	16625	4.9X
mosquitto	212	9	23.5X
libzmq	3861	7	551.5X



(a) Lightftp(FTP) (b) Dnsmasq(DNS)

(c) Openss l(TLS) (d) Cyc l onedds(DDS)

(e) Mosquitto(MQTT) (f) Libzmq(ZMTP)

图 2 模糊测试工具测试分支覆盖变化情况

Fig. 2 Covered branch coverage by different fuzzers in 24 h

根据实验结果可知，SPFuzz 在所测试的多个协议实现中普遍展现出了相较于 Peach 更高的分支覆盖数和更快的分支覆盖速度。平均来看，SPFuzz 相较于 Peach 在速度上提升了 111.4 倍，SPFuzz 在保持相似或更高的分支覆盖率的同时，实现了 1.2 倍至 551.5 倍的速度提升，特别是在处理逻辑较为复杂和状态较为丰富的网络协议时，其测试效果优势更为明显。SPFuzz 通过利用协议状态和数据模型生成有状态路径并基于它们分配任务，避免了冲突，平衡了工作负载，并显著加快了并行模糊测试的速度。

4.6 协议模糊测试经验总结

本文总结了在主流协议模糊测试工具应用过程中遇到的主要挑战与经验。

1）人工编写生成式协议模糊测试工具输入存在开销，需要根据目标协议合理选择测试工具类型。生成式协议模糊测试需要 pit 文件格式的协议测试模板，然而在实际测试中，部分协议难以获取开源 pit 模板，需要人工编写 pit 文件。对于此类没有公开 pit 模板的协议，可采用变异式协议模糊测试方法。例如，COAP 协议没有开源的 pit 文件，因此可采用变异式；而对有大量开源 pit 模板的 modbus 等协议可直接采用生成式模糊测试工具。

2）开源 pit 文件质量缺乏保障，需要人工审核优化。在对 lightftp 进行实际测试的过程中，我们发现了现有 pit 文件内容与当前 Peach 版本之间存在不兼容性，这一不匹配导致了 Peach 测试的执行失败。鉴于此，在实际的测试操作中，必须细致地依据协议规范对 pit 文件进行审核和优化，以防止测试执行失败和测试效果不佳的情况发生。

3）同类型模糊测试工具之间支持的协议种类也存在差别，需要根据具体测试协议类型版本进行选择。StateAFL 和 ChatAFL 作为在 AFLnet 基础上进行改进的测试工具，其协议支持范围存在一定的局限性。尽管 AFLnet 已通过更新扩展了对 17 个不同协议的支持，但 StateAFL 和 ChatAFL 目前仅支持其中的 11 个协议，尚未支持 MQTT，DNCP，TFTP，SNTP，NTP，SNMP 等 AFLnet 已经支持的协议。因此，测试人员需要及时跟进开源测试工具更新，根据具体工具支持协议合理选择测试工具。

4）测试工具依赖软件版本不同，需要合理选型保证工具兼容。以 Peach 为例，因为 mono 6 以上版本并兼容 4.1.8 版本，在安装 Peach 时首先需要 6 以上版本编译然后在 4.1.8 版本上运行。此外相关依赖包的缺乏更新可能导致安装过程复杂化，导致部署难度增加。因此，在实际使用时尽可能采用容器方式以避免重复配置环境。

5 研究挑战及未来发展方向

经过近年来的飞速发展，网络协议模糊测试技术积极吸收融合其他领域先进知识技术，已渐渐脱离传统的纯随机暴力破解方式。但在融合过程中，其技术适应性有待商榷，新技术在网络协议模糊测试研究领域还有很大发展空间。与机器学习技术的融合，可加速网络协议模糊测试自动化的进程，是近年来热门的研究方向。此外，网络协议模糊测试技术的开发部署和社区建设同样是不可忽略的发展方向。

1）引入机器学习技术辅助协议模糊测试。机器学习技术在图像识别及自然语言处理领域取得了显著的成就，将其与

模糊测试技术结合以实现针对目标协议的测试数据的自动化生成是一个非常值得探索的领域。目前看来，基于机器学习模型自动化数据生成过程的方法仍面临许多挑战。首先是训练数据的收集。部分研究者通过收集生成式协议模糊测试工具产生的大量测试用例作为训练数据，训练机器学习模型，以证明其方法的可行性，但无法说明其脱离协议文档的有效性。另一部分研究人员将收集的实际交互报文作为训练数据，但数据的不平衡问题成为了阻碍其发展的主要原因。其次就是模型训练导致的时耗成为一个需要解决的问题。目前为止，应用于模糊测试领域的协议自动化分析技术有所发展，但整体尚未成熟，其自动化带来的便利尚未能抵消协议分析过程导致的能耗问题，且需进一步提升机器学习等人工智能算法分析的准确率。但是随着大语言模型等人工智能相关技术的发展，网络协议模糊测试的全自动化技术获得了越来越多的关注，可以预见该方面的研究将持续发展。

2)协议模糊测试持续集成。当前谷歌，微软等公司已提出 OSS-Fuzz[52]，ClusterFuzz 等针对通用模糊测试的持续模糊测试平台，通过云服务等计算集群运行模糊测试工具，使模糊测试获得更高执行效率和漏洞挖掘效率，在提高开源项目安全性的同时降低了模糊测试成本。当前有越来越多的安全研究者和开发者开始加入到模糊测试研究中，相关模糊测试技术社区也正逐步壮大。在此基础上，如何构建持续性协议模糊测试框架及其测试平台，是网络协议模糊测试实用化的一个重要挑战和研究方向。

结束语 模糊测试是网络协议漏洞挖掘主流技术之一，本文对协议模糊测试研究进展进行了系统性介绍，首先概述了协议模糊测试技术基础与基本流程，然后对当前协议模糊测试研究进展进行了介绍和分析，最后对主流协议模糊测试工具进行了应用评估，并对协议模糊测试工具应用经验与未来发展方向分别进行了总结和展望。

参 考 文 献

[1] MANÈS V J M, HAN H S, HAN C, et al. The art, science, and engineering of fuzzing: A survey[J]. IEEE Transactions on Software Engineering, 2019, 47(11): 2312-2331.

[2] XU W, LI P, ZHANG W B, et al. Survey of network protocol fuzzing [J]. Application Research of Computers, 2023, 40 (8): 2241-2249.

[3] GODEFROID P. Fuzzing: Hack, art, and science[J]. Communications of the ACM, 2020, 63(2): 70-76.

[4] REN Z, ZHENG H, ZHANG J, et al. A Review of Fuzzing Techniques[J]. Journal of Computer Research and Development, 2021, 58(5): 944.

[5] LIANG H, PEI X, JIA X, et al. Fuzzing: State of the art[J]. IEEE Transactions on Reliability, 2018, 67(3): 1199-1218.

[6] BOEHME M, CADAR C, ROYCHOUDHURY A. Fuzzing: Challenges and Reflections[J]. IEEE Software, 2021, 38(3): 79-86.

[7] ECEIZA M, FLORES J L, ITURBE M. Fuzzing the Internet of Things: A Review on the Techniques and Challenges for Efficient Vulnerability Discovery in Embedded Systems[J]. IEEE Internet of Things Journal, 2021, 8(13): 10390-10411.

[8] 田一崛. 智能汽车网络漏洞检测技术的研究与实现[D]. 成都：电子科技大学, 2019.

[9] SPIKEFuzzer Platform[EB/OL]. (2022-02-12). http://www.immunitysec.com.

[10] Peach Fuzzer Platform [EB/OL]. (2022-02-12). http://www.peachfuzzer.com/products/peach-platform/.

[11] Sulley: A pure-python fully automated and unattended fuzzing framework [EB/OL]. (2021-01-23). https://www.github.com/OpenRCE/sulley.

[12] ZHANG H Z, HONG Z, ZHOU S L, et al. Fuzzing Optimization Method Based on Protocol State Migration Traversal[J]. Computer Engineering and Applications, 2020, 56(4): 82-91.

[13] LI Y H, HONG Z, LIN P H. Fuzzing Test Case Generation Method Based on Depth-first Search [J]. Computer Science, 2021, 48(12): 85-93.

[14] LI J L, CHEN Y L, LI Z, et al. Mining RTSP Protocol Vulnerabilities Based on Traversal of Protocol State Graph[J]. Computer Science, 2018, 45(9): 171-176.

[15] AICHERNIG B K, MUŠKARDIN E, PFERSCHER A. Learning-based fuzzing of IoT message brokers[C]// 14th IEEE Conference on Software Testing, Verification and Validation. 2021: 47-58.

[16] FITERAU-BROSTEAN P, JONSSON B, MERGET R, et al. Analysis of DTLS Implementations Using Protocol State Fuzzing [C]// 29th USENIX Security Symposium. 2020: 2523-2540.

[17] YU Y, CHEN Z, GAN S, et al. SGPFuzzer: A state-driven smart graybox protocol fuzzer for network protocol implementations [J]. IEEE Access, 2020, 8: 198668-198678.

[18] PHAM V T, BÖHME M, ROYCHOUDHURY A. AFLnet: a greybox fuzzer for network protocols[C]// IEEE 13th International Conference on Software Testing, Validation and Verification. 2020: 460-465.

[19] BLUMBERGS B, VAARANDI R. Bbuzz: A bit-aware fuzzing framework for network protocol systematic reverse engineering and analysis[C]// IEEE Military Communications Conference. 2017: 707-712.

[20] LUO J Z, SHAN C, CAI J, et al. IoT Application-Layer Protocol Vulnerability Detection using Reverse Engineering[J]. Symmetry, 2018, 10(11): 561.

[21] FENG X, SUN R, ZHU X, et al. Snipuzz: Black-box fuzzing of IoT firmware via message snippet inference[C]// ACM SIGSAC Conference on Computer and Communications Security. 2021: 337-350.

[22] HE H H, WANG Y J. PNFUZZ: A stateful network protocol fuzzing approach based on packet clustering [J]. Computer Science & Information Technology, 2020: 61-69.

[23] FAN R, CHANG Y. Machine learning for black-box fuzzing of network protocols[C]// International Conference on Information and Communications Security. 2017: 621-632.

[24] GAO Z, DONG W, CHANG R, et al. The Stacked Seq2seq-attention Model for Protocol Fuzzing[C]// IEEE 7th International Conference on Computer Science and Network Technology. 2019: 126-130.

[25] ZHAO H, LI Z, WEI H, et al. SeqFuzzer: An industrial protocol fuzzing framework from a deep learning perspective[C]// 12th IEEE Conference on software testing, validation and verification. 2019: 59-67.

[26] LV W, XIONG J, SHI J, et al. A deep convolution generative ad-

versarial networks based fuzzing framework for industry control protocols[J]. Journal of Intelligent Manufacturing，2021，32(2)：441-457.

[27] JERO S，PACHECO M L，GOLDWASSER D，et al. Leveraging textual specifications for grammar-based fuzzing of network protocols[C]// The AAAI Conference on Artificial Intelligence. 2019：9478-9483.

[28] HU Z H，PAN Z L. Testcase Filtering Method Based on QRNN for Network Protocol Fuzzing[J]. Computer Science，2022，49(5)：318-324.

[29] LIU X，CUI B，FU J，et al. HFuzz：Towards automatic fuzzing testing of NB-IoT core network protocols implementations[J]. Future Generation Computer Systems，2020，108：390-400.

[30] SONG C，YU B，ZHOU X，et al. SPFuzz：a hierarchical scheduling framework for stateful network protocol fuzzing[J]. IEEE Access，2019，7：18490-18499.

[31] LI M，HE L，TENG Y X，et al. Research on network protocol vulnerability discovery based on fuzz testing[C]// IEEE 2nd Information Technology，Networking，Electronic and Automation Control Conference. 2017：1354-1358.

[32] LUO Z，ZUO F，JIANG Y，et al. Polar：Function code aware fuzz testing of ics protocol[J]. ACM Transactions on Embedded Computing Systems，2019，18(5s)：1-22.

[33] REDINI N，CONTINELLA A，DAS D，et al. DIANE：identifying fuzzing triggers in Apps to generate under-constrained inputs for IoT devices[C]// IEEE Symposium on Security and Privacy. 2021：484-500.

[34] GASCON H，WRESSNEGGER C，YAMAGUCHI F，et al. Pulsar：Stateful black-box fuzzing of proprietary network protocols[C]// International Conference on Security and Privacy in Communication Systems，2015：330-347.

[35] LIN P Y，TIEN C W，HUANG T C，et al. ICPFuzzer：proprietary communication protocol fuzzing by using machine learning and feedback strategies[J]. Cybersecurity，2021，4(1)：1-15.

[36] NATELLA R. StateAFL：Greybox Fuzzing for Stateful Network Servers[J]. Empirical Software Engineering，2022，27(7)：191.

[37] PETERSON A，JERO S，HOQUE E，et al. aBBRate：Automating BBR Attack Exploration Using a Model-Based Approach[C]// International Symposium on Research in Attacks，Intrusions and Defenses. 2020：225-240.

[38] YU B，WANG P，YUE T，et al. Poster：Fuzzing IoT firmware via multi-stage message generation[C]// ACM SIGSAC Conference on Computer and Communications Security. 2019：2525-2527.

[39] ZHENG Y，DAVANIAN A，YIN H，et al. FIRM-AFL：High-Throughput Greybox Fuzzing of IoT Firmware via Augmented Process Emulation[C]// 2019 USENIX Security Symposium. 2019：1099-1114.

[40] KIM J，YU J，KIM H，et al. FIRM-COV：high-coverage greybox fuzzing for IoT firmware via optimized process emulation[J]. IEEE Access，2021，9：101627-101642.

[41] MAIER D，SEIDEL L，PARK S. Basesafe：Baseband sanitized fuzzing through emulation[C]// 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks. 2020：122-132.

[42] LUO Z，ZUO F，SHEN Y，et al. ICS protocol fuzzing：Coverage guided packet crack and generation[C]// 57th ACM/IEEE Design Automation Conference. 2020：1-6.

[43] ALSHMRANY K，CORDEIRO L. Finding security vulnerabilities in network protocol implementations[J]. arXiv：2001.09592，2020.

[44] ZOU Y H，BAI J J，ZHOU J，et al. TCP-Fuzz：Detecting Memory and Semantic Bugs in TCP Stacks with Fuzzing[C]// 2021 USENIX Annual Technical Conference. 2021：489-502.

[45] MOUKAHAL L J，ZULKERNINE M，SOUKUP M. Vulnerability-Oriented Fuzz Testing for Connected Autonomous Vehicle Systems[J]. IEEE Transactions on Reliability，2021，70(4)：1422-1437.

[46] WEN S，MENG Q，FENG C，et al. Protocol vulnerability detection based on network traffic analysis and binary reverse engineering[J]. PloS One，2017，12(10)：e0186188.

[47] KIM S J，CHO J，LEE C，et al. Smart seed selection-based effective black box fuzzing for IoT protocol[J]. The Journal of Supercomputing，2020，76(12)：10140-10154.

[48] GAO Z，DONG W，CHANG R，et al. Fw-fuzz：A code coverage-guided fuzzing framework for network protocols on firmware[J]. Concurrency and Computation：Practice and Experience，2022，34(16)：1-15.

[49] SANISLAV F S. Development of fuzzing methodologies for testing the resilience of the SATA protocol[D]. Politecnico di Torino，2020.

[50] CASTEUR G，AUBARET A，BLONDEAU B，et al. Fuzzing attacks for vulnerability discovery within MQTT protocol[C]// International Wireless Communications and Mobile Computing. 2020：420-425.

[51] YU J Z，LUO Z X，XIAF S Y，et al. SPFuzz：Stateful Path based Parallel Fuzzing for Protocols in Autonomous Vehicles[C]// Proceedings of the 61st ACM/IEEE Design Automation Conference(DAC'24). Association for Computing Machinery，New York，NY，USA，2024：1-6.

[52] SEREBRYANY K. OSS-Fuzz：Google's continuous fuzzing service for open source software[EB/OL]. (2021-11-09). https://github.com/google/oss-fuzz/.

HAN Luchao，born in 1992，Ph.D. His main research interests include network and information security.