电子学

MDPI

评审

网络协议模糊化技术发展综述

Zhaowei Zhang 1、Hongzheng Zhang 1、Jinjing Zhao 1、*和Yanfei Yin 2

信息系统安全科学技术国家重点实验室，北京100101;xavierzh0710@gmail.com（Z.Z.）;baiweidou31812631@gmail.com（中国）
2 中国航天系统科学与工程研究院，北京，100037;christmas720@163.com

*** 通讯地址：zhaojinjing@nudt.edu.cn**

摘要：网络协议作为计算机网络设备之间的通信规则，是网络正常运行的基础。然而，由网络协议中的设计缺陷和实施漏洞引起的安全问题给网络运行和安全带来了重大风险。网络协议模糊化是发现和减轻网络协议安全缺陷的有效技术。与其他安全分析技术相比，它具有无与伦比的优势，这归功于对目标先验知识的最低要求以及较低的部署复杂性。然而，测试用例生成的随机性、不可控的测试覆盖范围和不稳定的测试效率在确保测试过程和结果的可控性方面引入了挑战。为了全面考察网络协议模糊化技术的发展，分析其优势和存在的问题，本文从测试用例的生成方法和测试条件出发，对协议模糊化及其相关技术进行了分类和总结。具体而言，我们按照时间顺序概述了这些技术在过去20年中的发展轨迹和模式。在此基础上，进一步预测了模糊技术的发展方向。

关键词：漏洞发现;网络协议;起毛;网络安全;网络协议安全性

引文：
Zhang,Z.;Zhang,H.;Zhao,J.;Yin, Y.网络协议模糊化技术发展综述 *电子 2023,12,2904*。*https://doi.org/10.3390/electronics12132904*

学术编辑：安德烈 凯拉列夫

接收日期：2023年5月29日
修订日期：2023年6月22日
接受日期：2023年6月25日
发布日期：2023年7月1日

1. 引言

网络协议是计算机网络的基础。它们定义了通信实体之间消息交换的格式、含义、顺序和动作。随着网络应用的发展，网络协议中的漏洞成为威胁网络安全的重要因素。

2001 年，“Code Red”蠕虫利用 HTTP 协议实现中的漏洞获得 Microsoft IIS Web 服务器的超级用户权限。它感染了全球约360,000台服务器和100万台计算机，估计全球损失约26亿美元。2014年，公开披露并利用了OpenSSL中的“心脏出血”漏洞[1]。这一事件影响了大约500,000台互联网服务器。2021年，发现了一组名为“WRECK”的漏洞[2]，该漏洞与DNS协议的实现有关，可能导致拒绝服务或远程代码执行。仅在美国就有超过180,000台器械受到影响。

关于系统漏洞的发现，模糊概念由威斯康星大学的Barton Miller教授于1988年提出[3]。它已逐渐发展成为一种有效、快速和实用的技术[4–7]。主要思想是开发一种模糊工具，称为模糊器[8]，能够生成半有效数据（测试用例）并将它们提交给被测系统（SUT），以发现是否存在任何安全问题。半有效数据是指目标能够正确接收和处理的数据

系统，揭示了难以通过传统方法检测的深层漏洞[9-12]。模糊化的工作流程如图1所示，其中初始信息也称为种子。

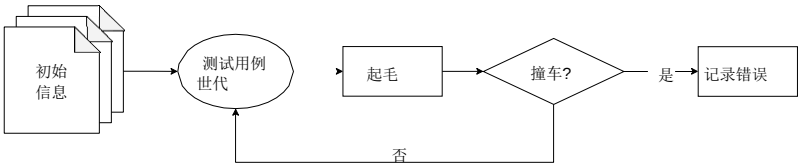

图1.网络协议模糊化技术的一般工作流程。

模糊化作为一种被广泛采用的漏洞检测技术，在协议安全测试领域得到了广泛的应用。目前，协议模糊化技术主要集中于测试协议实现。例如，在 CVE-2019-16519 漏洞中，在 BGP 协议通信过程中，发送消息长度足够长的紧密通信可能会导致路由守护程序出现缓冲区溢出错误。在协议模糊化中，如果模糊器生成并发送符合长度标准的紧密通信数据包，则可能会触发此漏洞。与传统模糊化技术的一个关键区别是，许多协议是有状态的，要求实现接收一系列消息请求并根据当前状态发送适当的响应。相比之下，传统的模糊工具不考虑软件的状态信息或待发送消息的结构和顺序。

在过去的两到三年中，没有专门针对网络协议模糊化技术的系统综述。如果我们再回顾几年，就已经有关于协议模糊化和传统模糊化的调查，其中提到模糊化技术在网络协议中的应用。Liang等人[7]阐述了传统模糊技术在每个阶段面临的关键挑战，并概述了为应对这些挑战而进行的研究。讨论了模糊化技术的不同应用场景，并简要介绍了两种网络协议模糊化器。Li等人[13]特别强调了基于覆盖的传统模糊化技术，并讨论了集成在模糊化中的技术。他们还提到了不同应用场景中的网络协议模糊化技术，仅强调了四种不同的模糊化器。Manes等人[14]将整个模糊化过程分解为几个阶段，并使用相关技术解释每个阶段的设计选择。在输入生成阶段，他们提到一些协议模糊化技术采用预定义的模型和基于推理的方法来生成测试用例，而没有为这些模糊器提供具体的介绍。Munea等人[15]从五个不同的角度对协议模糊化技术进行了分类和比较，但调查仅包括五个具体的模糊化器。Hu和Pan [16]按时间顺序总结了协议模糊化技术，并介绍了应用于网络协议模糊化的机器学习技术。然而，他们的研究在收集的相关技术和所进行的分析的全面性方面存在局限性。

本文旨在通过协议模糊化领域的调研来填补现有的空白。采用时间顺序的方法，根据测试用例的生成方法和测试条件，对约50种协议模糊化及相关研究技术进行了归纳和总结。这使得能够清楚了解自该技术开始以来不同阶段的发展轨迹和模式。在此基础上，进一步预测了协议模糊化技术的发展方向。

本文其余部分结构如下。在第2节中，我们将概述我们的文献检索方法。在第三节中，我们将介绍与网络协议模糊化技术相关的基础知识和常见的分类方法。在第四节中，我们将全面回顾过去二十年来网络协议模糊化技术的发展和进展，将它们划分为不同的阶段。第5节将检验和分析与网络协议模糊化技术相关的技术，并评估它们对该领域的贡献。在第6节中，我们将解决关键

并评估当前最新技术的优点和缺点。在第7节中，我们将全面分析协议模糊化技术中存在的瓶颈，并深入了解该技术的未来趋势。最后，我们将在第8节中进行总结。

2. 评审方法

为了对网络协议模糊化进行全面的研究，在接下来的章节中，我们将详细介绍我们的研究方法和收集的数据。

2.1. 研究问题

本综述的主要目的是回答以下关于网络协议模糊化的研究问题。

1. RQ1：协议模糊化研究的关键问题及相应的技术有哪些?
2. RQ2：最先进的技术及其利弊是什么?
3. RQ3：协议模糊化及相关技术的未来发展方向是什么?

RQ1在第6节中回答，它允许我们深入研究协议模糊化。建议在第6节中回答RQ2,以深入了解现有技术的比较和适当场景。最后，基于对前面问题的回答，我们期望识别未解决的问题以及协议模糊和相关技术的未来机会，以响应RQ3,在第7节中回答。

2.2. 检索策略

为了提供涵盖尽可能多相关论文的完整调查，我们通过三个步骤搜索相关技术。首先，我们检索了一些主要的在线存储库，如IEEE XPlore、ACM Digital Library、USENIX、Springer Online Library等，并进行了文献检索，以收集使用术语“fuzz testing”、“fuzzing”或“fuzzer”与“protocol”结合的论文，以及在其标题、摘要或关键词中包含“Protocol state machine”、“FSM”或“Protocol Modeling”的论文。其次，我们根据以下选择标准，使用所收集论文的摘要排除其中一些：

1. 与网络协议字段无关;
2. 非英文书写;
3. 无法通过 Web 访问。

第三，我们验证了所收集的论文的参考文献，以确定是否存在任何被忽略的技术。表1列出了从每个来源检索到的相关技术的数量。值得注意的是，“其他”类别包括许多可能尚未在研究论文中发表的重要技术，如Peach和AFL。

我们的检索仍可能无法完全涵盖所有相关论文，但我们确信本论文中的总体趋势是准确的，并提供了最先进技术的合理描述。

表1.出版商和相关技术的数量。

发布者	相关技术
IEEExpore数字图书馆	17
ACM数字图书馆	5
乌塞尼克	5
Springer在线图书馆	4
Elsevier ScienceDirect	3
其他	15

3. 网络协议模糊化技术的分类

3.1. 基于测试用例生成方法的分类

在网络协议模糊化技术中，测试用例主要是指被正确格式化但包含错误内容的协议数据分组。不同的网络协议模糊技术采用各种方法来生成测试用例，然后使用诸如套接字等机制将测试用例发送到被测协议实现以识别漏洞。在网络协议模糊化技术的分类标准中，测试用例的生成方法是最重要的因素之一。测试用例生成方法可大致分为基于突变和基于生成的方法，概述如下：

1. 基于突变的方法：在这种方法中，fuzzer最初获得一些具有适当格式和内容的有效数据。然后使用不同的方法修改这些数据，以创建相应的半有效数据。突变过程通常涉及四种方法：位翻转、算术突变、基于块的突变和基于字典的突变[14]：
    - 位翻转涉及翻转数据包中的特定位，将0更改为1并将1更改为0。
    - 算术变异选择字节序列，将其视为整数，执行算术运算，并生成新的整数值，然后将其插入原始字节序列。
    - 基于块的变异将给定长度的字节序列视为一个块，它被认为是数据包的基本单元。执行诸如添加、删除、替换和调整块的优先级的操作。
    - 基于字典的变异关注于包含加权字段的特定语义语句。它用预定义的数字或字符串替换权重或其他相关字段。
2. 基于生成的方法：在这种方法中，模糊器基于已知的规范或模板生成半有效数据。这些模板可以由测试人员自己定义，或者在fuzzer中预定义。

下面比较这两种方法在网络协议模糊化背景下的优缺点：

1. 基于变异的方法面临挑战，因为网络协议数据分组通常包含多种数据类型，并且不同的协议或分组类型具有不同的规范。很难找到合适的突变策略来生成在测试用例生成阶段能够正确接收的测试用例。如果采用随机突变策略，则需要付出很大努力来验证这些测试用例是否可以正确接收。
2. 另一方面，基于一代的方法也有其挑战。问题之一是在种子获取阶段期间获取网络协议规范所涉及的成本。获得网络协议的详细规范可能需要大量的资源和努力。此外，如果试验人员对方案的理解存在偏差，则采集的种子的质量可能会受到影响。对方案的理解不准确或不完整可能导致产生有缺陷或无效的测试用例，从而妨碍模糊过程的有效性。

3.2. 基于测试条件的分类

基于对协议实现的理解水平（指实现网络协议并处理协议消息的发送或接收的应用程序/软件或硬件过程），网络协议模糊可分为三类：黑盒、白盒和灰盒模糊。

1. 黑盒：黑盒模糊化，也称为随机测试，涉及模糊化工具不了解SUT的内部工作。它只能通过观察系统的输入和输出来推断其行为。因此，与其他方法相比，黑盒模糊化往往具有更低的代码覆盖率。
2. 白盒：白盒模糊需要理解目标程序的内部逻辑[17]。模糊工具收集和分析关于系统内部工作的信息，以生成测试用例。在网络协议模糊化的上下文中，这需要理解协议实现的特定代码和运行时信息。这种方法最初由Godefroid等人提出。以解决黑盒模糊在盲法和随机测试方面的局限性[18,19]。理论上，白盒模糊可以覆盖SUT中的所有代码路径。然而，实现100%的代码覆盖率仍然具有挑战性，尤其是对于大规模的协议实现。
3. 灰盒：灰盒模糊介于黑盒和白盒模糊之间。它根据从SUT获得的动态信息（如代码覆盖率、分支条件和内存状态）调整测试用例生成方法。它旨在生成覆盖更多执行路径或更高效地发现错误的测试用例，而不需要代码实现的特定知识。

在网络协议模糊化的背景下，测试人员访问协议实现的源代码通常具有挑战性。因此，黑盒和灰盒模糊化技术更常用，而白盒模糊化方法相对有限。与黑盒模糊化相比，灰盒模糊化具有基于所获得的协议实现信息来调整测试方向的优点，从而解决了在黑盒模糊化中遇到的盲目和随机测试的问题。

4. 网络协议模糊化技术的发展脉络

自从2001年模糊化技术应用于网络协议安全测试领域以来，网络协议模糊化技术的发展已经历了近20年。本节从测试时间线和测试条件出发，从测试用例生成方法和测试条件两个方面对网络协议模糊化技术的发展进行了回顾和分析。目的是概述该技术的发展轨迹和未来趋势。相关工作的具体分析如图2所示。

从图2可以看出，在2017年之前，大多数网络协议模糊化技术都采用了基于代数的黑盒模糊化方法。 然而，在过去的五年中，灰盒模糊技术经历了飞速的发展。此外，考虑到测试用例生成步骤的自动化水平和模糊器的目标等因素，网络协议模糊化技术的发展可分为三个阶段：初始阶段（2001-2009）、细化阶段（2009-2017）和开发阶段（2017-至今）。

4.1. 初始阶段

网络协议模糊技术的初始阶段始于2001年这些技术的出现，并持续到2009年左右。在此阶段，测试工具主要是通用的协议模糊化框架，并采用了黑盒模糊化技术。这些模糊化框架要么依赖于测试人员手工构建测试用例，要么利用协议规范指导的基于生成的方法来生成测试用例。

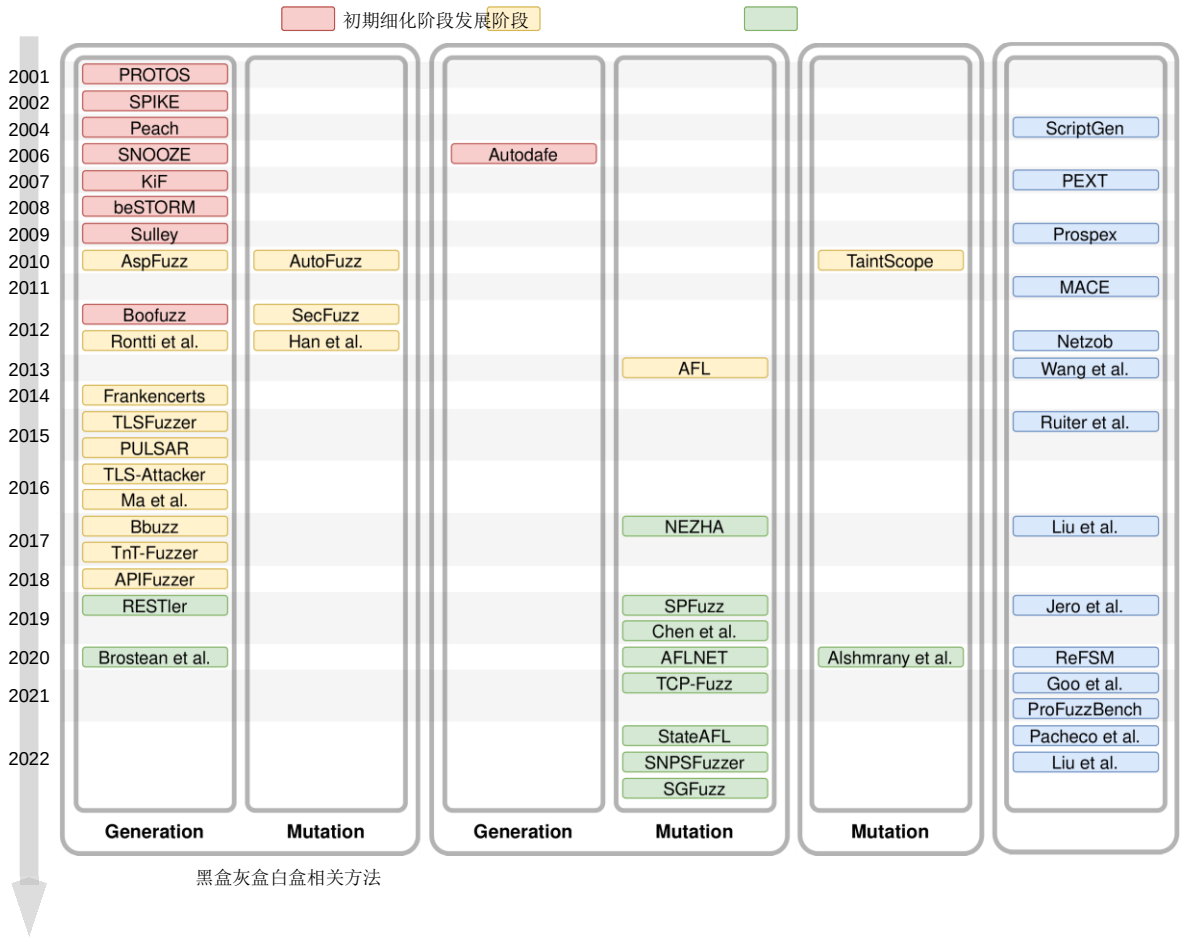

图2.网络协议模糊化技术相关研究。

4.1.1. 工作介绍

1. 黑匣子

2001年，Oulu大学的协议测试集研究项目PROTOS[20]介绍了模糊技术在网络协议安全测试中的应用。其目的是发现缓冲区溢出和字符串格式错误等漏洞。该方法引入了测试套件的概念，它涉及手动构建测试消息。这些消息基于协议规范的分析，考虑了支持的数据结构和每个字段的可接受值范围。然而，PROTOS具有局限性，因为它不提供用于构建自定义模糊化的API,并且不允许在不改变协议语法本身的情况下在测试用例中进行变化。

2002年，引入了基于C语言的通用协议模糊化框架SPIKE框架[21]。它提供了一个模糊数据库，其中包含各种格式错误的字符，如长字符串、带格式说明符的字符串、大整数和负数。SPIKE还提供了丰富的API集，并引入了基于块的协议模糊化的概念。使用SPIKE的测试人员可以直接使用提供的实用程序脚本执行fuzzing,或者通过利用框架提供的API的轻量级封装来创建自己的fuzzer。然而，SPIKE仍然需要用于构建测试用例的协议的先验知识，并且依赖于手动调整。另外，所提供的块抽象是相对低级的。这使得难以轻松地对有状态协议和复杂消息进行建模。

2004年，Deja vu Security发布了名为Peach的跨平台模糊框架[22]。它由诸如Datamodel（包括数据类型和mutator接口）、Statemodel（包括Datamodel接口、状态和操作）、代理、测试-

发动机等通过手动编写Peach Pit配置文件创建Fuzzer。Peach的初始版本是用Python开发的，随后分别在2007年和2013年发布了第二个和第三个版本。第三个版本用C#重新开发，支持文件格式、ActiveX、网络协议、API等的模糊化。2006年，Greg Banks等人开发了一个名为SNOOZE的网络协议模糊化工具[23]。它引入了基于场景的模糊化方法，允许测试人员描述协议中的有状态操作，并生成由每个状态中生成的消息组成的场景。该工具基于模糊场景生成相应的测试用例，从而实现有状态协议测试。SNOOZE还提供针对特定攻击的模糊原语，从而允许测试人员专注于特定类型的漏洞。该工具可以发现缓冲区溢出、整数溢出和SQL注入等漏洞。然而，一个限制是在模糊器中观察到的状态与SUT中的状态不同步。这需要测试人员进行进一步分析。

2007年，H.J.Abdelnur等人开发了工具KiF [24]，这是第一个不单独生成随机数据的SIP模糊器。KiF 根据 SIP 的场景和协议规范生成测试用例。它能够通过自我改进和跟踪目标设备的状态来进行自动攻击。KiF的实现依赖于使用真实网络跟踪训练攻击自动机的学习算法。这种攻击自动机可以在模糊阶段演化和更新。

2008年，Beyond Security发布了商业模糊工具beSTORM [25]。该工具将RFC文档中使用的BNF转换为攻击语言，将协议规范转换为自动化测试套件。beSTORM的体系结构由客户端和监控端两部分组成。客户端向SUT发送半有效报文，而监控端监控SUT的状态，记录异常，并返回客户端。

2009年，Fuzzing框架Sulley在GitHub平台上发布[26]。它由一个可定制的fuzzer和多个可扩展组件组成。它的优点包括简化数据传输、表示和目标监控过程。Sulley的具体特点包括：（1）监视网络通信并维护相关记录，（2）检测和监视目标的运行状态，具有使用各种方法恢复正常操作的能力，（3）检测、跟踪和分类崩溃，（4）并行执行模糊化，以及（5）自动确定触发错误的测试用例的顺序。Sulley通过将要模糊化的协议请求分解为块来实现模糊化，然后将分解后的请求链接到会话中，并在进行测试之前附加可用的监控代理。Sulley不再积极维护，2012年发布的Boofuzz [27]是Sulley框架的一个分支和延续。Boofuzz修复了Sulley中发现的问题，并提高了其可伸缩性。

2. 灰盒

2006年，Vuagnoux引入了Autodafe [28]，这是一种旨在发现缓冲区溢出漏洞的模糊工具。Autodafe分析SUT如何使用用户控制变量，并使用作为参数传递给安全敏感函数的变量对测试进行优先级划分。Autodafe的优点是它能够自动生成协议描述并使用模糊数据库进行测试。

4.1.2. 总结

表2概述了代表性技术。可以观察到，随着技术的发展，最初的技术仅限于测试无状态协议，而后来的技术支持有状态协议的测试。在此阶段，模糊技术严重依赖于构建测试用例过程中的人工干预，并且在测试灵活性方面存在一定的局限性。然而，诸如Peach、Sulley和Boofuzz等模糊工具至今仍被广泛使用。

表2.初始阶段代表性技术的总结。

工具测试用例生成方法测试		条件	状态支持	主要贡献
原型	代次	黑匣子	无国籍	网络协议模糊化的第一个工具。
加标	代次	黑匣子	无国籍	介绍基于块的协议测试策略。
桃	生成/突变	黑匣子	有状态	协议覆盖范围广，支持文件测试。
苏利	代次	黑匣子	有状态	支持并行模糊测试。

4.2. 精细化阶段

4.2.1. 工作介绍

1. 黑匣子

2010年，Gorbunov和Rosenbloom提出了一种名为AutoFuzz[29]的可扩展开源框架，用于测试网络协议实现。该框架捕获客户端和服务器之间的通信，以构造学习协议实现的有限状态自动机。通过应用生物信息学的知识，AutoFuzz学习单个协议消息的语法，包括消息字段和可能的类型。使用有限状态自动机作为指导，该框架智能地改变客户端和服务器之间的通信会话以执行测试。在此过程中，AutoFuzz记录所有操作，供测试人员审查和验证。然而，AutoFuzz缺乏对来自协议实现的真实反馈和来自协议状态机的理想反馈之间的分析和比较，以确定是否发生了特定类型的意外行为。同年，Kitagawa等人提出了基于应用层协议规范的状态感知模糊器AspFuzz[30]。先前的消息级模糊器仅考虑单个消息中的变化，而不改变消息传输的顺序。基于场景的Fuzzer,如SNOOZE和KiF,可以基于场景确定消息传输顺序，以避免消息级Fuzzer遇到的问题，但是场景创建过程很复杂。AspFuzz采用一种状态感知方法，允许在生成测试用例之后选择测试用例传输的顺序。测试用例可以以正确的顺序或错误的顺序传输。但是，AspFuzz依赖于测试协议的手动定义以及在测试过程中手动确定成功的攻击。

2012年，Tsankov等人引入了工具SecFuzz [31]，以解决某些协议实现的协议消息中的加密内容问题。SecFuzz为fuzzer提供了必要的密钥和加密算法，以便根据网络协议fuzzing正确改变加密消息。通过充当客户端和服务器之间的中间人来拦截消息，SecFuzz获得有效的输入，并根据三个自定义模糊运算符对其进行分类，然后对它们进行变异并将其转发到SUT。同年，Rontti et al.研究了下一代网络（NGN）fuzzer [32]，它使用协议规范创建测试用例。NGN网络是指集成所有类型的服务和媒体的网络。 该工具使用从协议规范导出的语法规则来增强协议描述，并且从现有的异常库中选择适当的异常级别以生成相应的测试用例。实验结果表明，Fuzzer能够有效地发现DoS或DDoS攻击中可被利用的漏洞。Han等人提出了一种基于关系分析和测试用例标记（RATM）模型的多域模糊数据集的模糊测试方法[33]。该方法通过分析协议中域之间的关系，直接对可能触发漏洞的对应数据包进行变异。它还可以分析测试结果，修改RATM参数，提高测试用例的质量。该方法对于基于MAC层的模糊化协议是非常有效的。

2014年，Brubaker等人设计、实现并应用了第一个大规模测试工具Frankencerts [34]，专门针对SSL/TLS中的证书验证逻辑

协议实现。该工具解决了两个主要问题：生成高质量的测试用例和验证接受/拒绝证书的合理性。关于测试用例生成，Frankencerts创建了一个包含大量证书的语料库。在测试用例生成期间，它将证书的手动构建部分与真实证书的随机组合部分进行组合，确保生成的测试用例具有可由协议处理的格式良好的语法。这些测试用例还违反了有效证书必须满足的约束和依赖关系，增加了触发协议漏洞的可能性。关于证书结果的合理性，该工具使用差异测试，使用多个独立的实现进行联合验证，以确保接受或拒绝证书的原因是正确的。如果大多数实现的结果不一致，则表明结果不正确。

同样，关注TLS协议实现确认，2015年，TLSFuzzer [35]（TLS协议的模糊工具）在GitHub上发布。与仅检查SUT是否崩溃的典型模糊器不同，TLSFuzzer还验证系统是否返回正确的错误消息。此工具验证TLS 协议中服务器的行为，检查 TLS 消息上的签名是否与服务器发送的证书信息匹配，而无需对协议证书执行任何检查。同年，Gascon等人引入了PULSAR [36]，这是一种用于专有网络协议的有状态黑盒模糊工具。该工具适用于没有可用的协议实现代码和协议规范的场景。PULSAR将模糊技术与自动协议逆向工程相结合，根据程序生成的一组网络数据包，自动推断出网络协议模型。学习的网络协议模型指导模糊化过程。PULSAR的缺点在于，从网络消息中学习的模型可能自然缺乏一些功能，这可能影响协议的测试过程。

2016年，Somorovsky等人开发了一个名为TLS攻击者[37]的开源框架，用于评估TLS库的安全性。该框架是使用Maven项目管理工具实施的。基于TLS攻击者，作者进一步提出了一种两阶段模糊化方法来评估TLS服务器行为，自动检测异常的填充oracle漏洞和溢出/过读。他们还建立了与TLS协议相关的测试套件。同年，Ma et al.提出了一种使用基于规则的状态机和有状态规则树生成模糊数据的方法[38]。该方法利用状态机作为网络协议状态的形式化描述，通过利用协议规则去除已知的安全路径来简化状态机，并使用有状态规则树来描述状态与消息之间的关系。使用各种生成算法来使用数据生成算法来有规律地突变初始种子，使得能够生成模糊数据。

2017年，Blumbergs等人引入工具Bbuzz,用于分析网络协议[39]，特别适用于军事环境。该工具在位级操作，过滤并在文件中存储所需的数据包数据。它通过计算香农熵来帮助识别场属性，从而能够分析哪些部分可以突变。因此，二进制协议的逆向工程结果变得更加准确。

对于代表性状态转移（REST）API,2017年和2018年在GitHub上发布了两个模糊工具：TnT-Fuzzer[40]和APIFuzzer[41]。 这两个工具都用Python编写，并利用Swagger规范解析HTTP请求，从而指导模糊化过程。

2. 灰盒

2013年，Zalewski等人提出了一种基于变异的通用模糊化工具，称为美国模糊Lop（AFL）[42]。该工具通过利用源代码编译插装和QEMU模式引入了代码覆盖率引导的模糊化。然而，AFL更适合于测试无状态项目，例如测试文件，并且缺乏协议实现的状态信息以及

测试网络协议。消息突变的过程是随机的，导致测试效率较低。

3. 白盒

2010年，Wang et al.呈现了TaintScope [43]，这是一种绕过校验和和确认的模糊工具。该工具使用细粒度的污染分析来识别流入关键系统调用或API调用的输入。它还引入了校验和感知模糊技术，该技术通过污点分析识别校验和测试指令，并修改SUT以绕过校验和验证。如果修改的SUT崩溃，TaintScope使用混合符号执行进一步修复校验和字段，并对原始SUT进行测试。此外，TaintScope 监控 SUT 如何访问和使用输入数据，以及如何定向突变敏感信息。但是，TaintScope 在处理与安全相关的完整性检查方案（如数字签名）方面存在局限性，并且在处理加密数据时效率较低。此外，它需要高质量的输入，这取决于格式正确和格式错误的输入格式。

4.2.2. 总结

表3总结了优化阶段的代表性技术。从表中可以看出，在细化阶段，模糊技术解决了初始阶段存在的最重要问题，即过度依赖人工干预。例如，Autofuzz和PULSAR可以通过分析协议通信数据包来自动生成协议状态机。此外，在细化阶段，模糊工具逐渐将焦点从一般协议转移到特定协议族。这些工具还能够在测试期间处理特定场景。例如，Frankencerts 和TLSFuzzer 等工具以 SSL/TLS 协议实现为目标，SecFuzz 专注于测试加密内容，而TaintScope 绕过 SUT 的完整性测试过程。

另一方面，在此期间，AFL作为一般模糊工具发布。AFL虽然在协议模糊化方面表现不佳，但提供了一种新的协议模糊化方法：利用灰盒模糊化技术来收集覆盖引导信息，提高网络协议模糊化的效率。

表3.细化阶段代表性技术的总结。

刀具生成方法	测试用例	试验条件	支持	主要贡献
SecFuzz	Mutation	Black-box	Stateful	自动模糊变异黑盒状态自动生成状态机。通过提供加密实现对加密数据的模糊化算法和Fuzzer的密钥。
Frankencerts	Generation	Black-box	Stateless	First Fuzzer for TLS 证书。PULSAR 生成黑盒状态自动生成状态机。
TaintScope	Mutation	白盒	状态	第一协议模糊器采用白盒模糊化方法。
AFL	突变	灰盒	状态	采用覆盖率反馈来指导测试过程，作为开发后续工具的基础。

4.3. 开发阶段

2017年至今的发展阶段的特点是灰盒模糊化技术占主导地位，该技术利用代码覆盖率来提高模糊化效率。

4.3.1. 工作介绍

1. 黑匣子

2019年，Atlidakis等人为REST API开发了第一个有状态模糊器，称为RESTler [44]。该工具分析云服务的OpenAPI规范，以提取REST

语法并推断不同类型的请求之间的依赖性。RESTler根据服务响应的动态反馈使用三种不同的搜索策略来帮助生成测试用例。此外，RESTler引入了分组方案来聚类类似的漏洞，帮助用户进行漏洞分析。

2020年，对于数据报传输层安全（DTLS）协议，Brostean等人扩展了TLS-Attacker框架，并为DTLS服务器构建了有状态模糊框架[45]。该工具采用模型学习和TTT算法对Mealy机器进行推理，并对DTLS进行了全面的协议状态模糊化。实验分析了13个DTLS实现的Mealy状态机模型，揭示了4个严重的安全漏洞。

2. 灰盒

2017年，Petsios等人修改了LibFuzzer框架[46]，并提出了一种称为NEZHA[47]的高效差分测试工具。NEZHA引入δ-多样性概念，以捕获多个SUT之间的行为不一致。NEZHA由运行时组件和核心引擎组成：运行时组件收集δ-多样性指导的所有必要信息，并传递给核心引擎，核心引擎通过变异生成新的输入，以揭示SUT之间的差异，并在δ-多样性指导下更新种子语料库。在测试期间，NEZHA根据SUT是否支持指令插入或二进制重写来确定采用黑盒还是灰盒方法。2019年，Song等人提出了SPFuzz[48]，一个有状态的协议模糊化框架，旨在建立一个灵活的覆盖引导方法。SPFuzz结合了Boofuzz的语言规范来描述协议规范、状态转换和依赖关系，以生成有价值的测试用例。它在会话状态下维护正确的消息，并通过及时更新消息数据来处理协议依赖性。SPFuzz采用三级变异策略（报头、内容和序列），并结合消息随机分配和变异策略的权重，以在模糊化过程中覆盖更多的路径。SPFuzz的一个限制是它需要协议的源代码和规范，协议规范依赖于人工构造。同年，为了在测试网络通信协议时实现更高的代码覆盖率，Chen et al.设计了有状态协议模糊化策略，并论证了无状态灰盒模糊器在协议测试中的局限性[49]。状态模糊策略由状态转换引擎和多状态分叉服务器组成。它采用深度优先搜索算法对不同的模糊状态进行搜索，并通过能量调度灵活地确定状态的级进和回归。该策略允许对协议程序中的不同状态进行灵活的模糊化。

2020年，Pham等人开发了AFLNET [50]，它使用响应代码作为网络协议程序的状态。AFLNET对网络协议程序的实际状态进行模糊化，并利用状态反馈来指导模糊化过程。AFLNET使用基于变异的方法，并使用客户端和服务器之间的消息交换序列作为初始种子。在测试期间，AFLNET充当客户端并重放请求消息序列的变体。它还应用覆盖指导技术来保留有效改进代码或状态覆盖的变体。与AFLnwe（AFL的网络版本）和基于代数的测试工具Boofuzz等覆盖范围引导的无状态测试工具相比，AFLNET显示出显著的改进。然而，AFLNET不适用于没有状态代码的协议，并且状态信息的提取依赖于手动编写的协议规范。

2021年，Zou et al.设计了TCP-Fuzz[51]，这是一种新颖的模糊框架，用于有效测试TCP堆栈并检测其中的错误。TCP-Fuzz采用基于依赖的策略，通过考虑系统调用与数据包之间的依赖关系生成有效的测试用例，生成系统调用序列。为了实现状态转移的有效覆盖，TCP-Fuzz采用了转移引导的模糊化方法，该方法利用称为分支转移的新代码度量作为程序反馈而不是代码覆盖。分支转换被表示为存储当前输入（数据包或系统调用）的分支覆盖以及当前输入（数据包或系统调用）之间分支覆盖的变化的向量。

输入和先前输入。该方法不仅描述状态，而且捕获相邻输入之间的状态转换。最后，为了检测语义错误，TCP-Fuzz使用差异检验器来比较多个TCP堆栈的输出与相同的输入。由于不同的 TCP 堆栈应遵守许多相同的语义规则（大部分在 RFC 文档中定义），因此输出中的不一致表明某些TCP 堆栈中可能存在语义错误。

2022年，Natella开发了StateAFL [52]，这是一种用于网络服务器的灰盒模糊器。StateAFL 执行编译时指令插入以检测目标服务器，并插入内存分配和网络 I/O 操作的探测器。在运行时，Fuzzer捕获长寿命内存区域的快照，并应用对位置敏感的散列算法将内存内容映射到唯一的状态标识符。这允许StateAFL推断SUT的当前协议状态，并逐渐构建协议状态机以引导模糊处理。定性分析表明，从内存推断状态比单独依赖响应代码能更好地反映服务器行为。同年，Li et al.提出了另一种称为SNPSFuzzer[53]的快速灰盒模糊器，它也使用快照。SNPSFuzzer构建在AFLNET之上，并引入了三个主要组件：基于快照的实例生成器、快照器和消息链分析器。当网络协议程序达到特定状态时，上下文信息被存储，并且可以在需要时被恢复。此外，Li等人设计了一个消息链分析算法，利用两个变量将消息链分解为前缀、中缀和后缀信息。此方法探索更深层的网络协议状态。与AFLNET相比，SNPSFuzzer在24小时内的网络协议模糊化速度提高了112.0%到168.9%，路径覆盖率提高了21.4%到27.5%。此外，在2022年，na et al.开发了SGFuzz [54]，这是一种自动分析协议状态并通过利用协议实现中变量名之间的规则性来执行状态测试的工具。SGFuzz使用模式匹配来识别使用枚举的状态变量。当为状态变量分配新值时，工具发送相应的通知并将新状态添加到构建的状态转换树（STT）中。SGFuzz还向种子库添加生成的输入，用于训练STT,并且集中于很少被访问并且具有更可能穿越不同路径的后代的节点，从而改进状态空间的覆盖。SGFuzz在生成状态序列、实现相同的分支覆盖以及在没有显式协议规范或手动注释的情况下发现有状态错误方面实现了显著改进。但是，SGFuzz需要协议实现的源代码来分析测试过程中的协议状态，而隐藏状态触发的漏洞分析仍然需要手动分析。

3. 白盒

2020年，Alshmrany等人提出了一种将模糊化和符号执行技术相结合的验证方法[55]。该方法基于AFL,利用模糊理论对网络协议进行初步探索。同时，采用符号执行来探索程序路径和协议状态。通过组合这些技术，可以自动生成用于网络协议实现的高覆盖率测试用例数据包。符号执行是使用路径探索和有界模型检查（BMC）方法来实现的。

4.3.2. 总结

表4总结了开发阶段的代表性技术。在此阶段，在AFL的基础上发展了几种灰盒模糊化技术，将AFL的无状态模糊化转化为适用于协议的有状态模糊化。AFLNET、StateAFL、SNPSFuzzer和SGFuzz都使用状态反馈来指导模糊化过程。它们通过记录消息响应代码、捕获内存快照、存储上下文信息以及分析协议源代码来识别特定数据类型并确定协议状态来实现这一点。Alshmrany等人[55]使用符号执行技术来分析程序路径并探索协议状态。

通常，在此阶段，重点是模糊化有状态协议，这些模糊化工具的主要目标是发现更深层协议状态中的漏洞。

表4.开发阶段代表性技术的总结。

工	测试用例生成方法	试验条件	国家支持	主要贡献
NEZHA	突变	灰盒	无状态	指导使用δ多样性更新种子库。
SPFuzz	Mutation	Gray-box	Stateful	使用代码覆盖率作为反馈来改进模糊效率。
AFLNET	突变	灰盒	状态	首个模糊器使用状态感知反馈。
StateAFL	Mutation	Gray-box	Stateful	通过内存快照表示协议状态。
SGFuzz	突变	灰盒	状态	利用现有协议实现用特定名称命名状态的趋势。

5. 网络协议模糊化技术的相关方法

除了上述网络协议模糊化工具/框架之外，还存在有助于网络协议模糊化的自动化和评估的其他相关工具和方法。

5.1. 协议状态机的自动生成技术

有限状态机（Finite State Machines,简称FSMs）是网络协议研究中的常用方法，用来对网络协议的消息交换过程进行建模，以描述协议实体的状态转换过程。为了为未知协议构造模糊测试用例，需要利用通过协议逆向工程提取的协议规范。对于无状态协议，可以根据协议格式生成测试用例，以确保每个字段突破目标程序的验证。然而，对于有状态的协议，为了避免由于状态不匹配而导致的大量无效测试用例，保证测试的深度和效率，需要有针对性地发送协议状态机可以接受的测试消息序列。传统的模糊测试并不包括上下文信息和消息序列中的所有状态，因此为每个状态生成的测试数据是离散的，并且可能不覆盖整个状态轨迹。因此，状态转换中的漏洞可能未被检测到，并且可能存在大量冗余测试数据。测试数据是随机生成的，缺乏规则。此外，基于生成的协议模糊化技术可以利用协议状态机来构造协议模板。因此，结合有限状态机生成技术，可以使协议模糊测试更具针对性和高效性，提供更好的测试覆盖率。

接下来，我们将介绍自动协议状态机生成技术的发展历史。

5.1.1. 基于被动推理的方法

被动推理是指在不依赖协议实体的指导的情况下，从给定的有限样本集合推断状态机的过程。它主要包括两个阶段：状态标记和状态机简化。在状态标记阶段，可以根据消息类型、长度和样本数据中的位置等特征分配分类标记，从而产生不同的状态。在状态机简化阶段，可以进一步简化标记状态，以生成简洁的状态机，用于后续分析和应用。

2004年，提出了第一个从网络数据流推断协议状态机的分析工具，称为ScriptGen [56]。该工具使用类似于PI项目的Needleman–Wunsch序列比对算法，以及捕获的网络流量的微聚类和宏聚类来构建协议状态机。然而，该工具具有某些限制，诸如其不能处理具有因果关联的不同消息会话以及其依赖于TCP分组标识符（例如，ACK、SYN、FIN）作为

优先解决与 TCP 数据包重组、重传和无序数据包相关的问题。因此，它只能分析几个特定的协议，并且不是一个稳健的、广泛适用的用于推断协议状态的解决方案。

2007年，Shevertalov等人最初提出了名为PEXT[57]的解决方案，该解决方案基于用于自动协议状态机推理的分组聚类。该解决方案基于分组a和分组b的最长公共子序列的长度来计算分组a和分组b之间的距离度量D（a,b）。然后，它基于D（a,b）对数据包进行聚类，并使用与初始状态转换序列相同的源地址和目的地址来注释数据包。随后，它合并具有公共前缀且没有同级节点的状态对，以获得整个样本集的状态机。但是，PEXT 有两个限制：（1）它没有充分考虑关键字字段在状态机转换中的作用，导致不太准确和精确的状态标记，以及（2）合并过程太简单，导致较不简洁的推断状态机。

2009年，Comparetti等人提出的Prospex[58]是一种基于二进制可执行代码推断协议状态的解决方案。该工具改进了他们先前关于基于行为的消息格式提取的工作[59]。它介绍了消息结构以及消息对服务器行为的影响。在会话分析阶段，Prospex使用动态污染分析来跟踪涉及从协议消息读取数据的所有操作。它将会话拆分为消息，将服务器收到的第一个输入字节视为消息的开始，并将所有后续输入视为该消息的一部分，直到服务器发送应答为止。重复该过程，直到所有被跟踪的数据分组被分段成消息。该方法比将每个分组视为消息更准确，特别是对于消息可能跨越多个数据分组的基于交互的协议。然后，它从样本分组中提取特征，对它们进行聚类以获得一组消息类型（M），将会话样本（S）表示为消息类型序列（MTS），并且
利用S.在APTA中，节点表示状态，边表示引起状态转移的输入ai∈M。还推导了每个类型mi M的前任类型Pi。 最后，利用Ex-bar算法进行融合
并提取最小确定性有限自动机（DFA）。然而，根据研究[60]，仅基于从网络捕获的正样本来推断准确的最小∈状态机是困难的。

5.1.2. 基于主动推理的方法

被动推理算法依赖于样本集的完备性。为了解决这一局限性，Angluin等人引入了L*算法，开始了对主动推理的研究。[61]在主动推理中，目标是利用人工学习系统来扩展原始样本集，以迭代地推断状态机。

L*算法将输入-输出示例分为两组：正样本和负样本。正样本是自动机可以正确处理的输入序列，而负样本是自动机不能正确处理的输入序列。L*算法假设存在能够提供准确答案的Oracle。有两种类型的查询：成员查询和等价查询。该算法使用这些示例来构造假设集OT并生成所有候选状态机M。随着算法的进展，使用等价查询针对真实状态机评估M，以逐渐缩小假设集并找到最小化状态自动机。L*算法确保在多项式时间内可以推断出完整的最小状态机，但关键的挑战是如何准确地回答查询。

2011年，Cho等人提出了基于L*算法的工具MACE[62]。MACE依靠动态符号执行来发现协议消息，并使用特殊的过滤组件来选择用于学习模型的消息。它使用学习模型来指导进一步的搜索，并在发现新消息时对其进行细化。这三个组件交替，直到过程收敛，自动推断协议状态机并探索程序的状态空间。MACE可以推断方案模型

探索程序的搜索空间，自动生成测试。在对4个程序的实验分析中，MACE发现了7个漏洞，并取得了良好的结果。

2012年，Bossert等人介绍了开源项目Netzob[63]，它由三个模块组成：词汇推理模块、语法推理模块和模拟模块。词汇推理模块采用PI[64]的多序列比对算法，并对L*算法进行了改进。它使用消息格式中的特征信息来推断 Mealy 机器。然后，在仿真模块中使用推断的词法和语法规范来仿真协议实体之间的通信，使得能够对未知协议进行智能模糊化。

2013年，Wang等人提出并设计了Veritas系统[65]。Veritas利用通过聚类分析获得的一组协议状态信息来构建概率协议状态机。对于每个协议状态信息，Veritas测量其发生频率以及它与其他协议状态信息之间的转换概率。然后，Veritas构造标记有向图，以使用这些统计结果来表示协议状态机，状态转移和转移概率值作为有向边上的标记。最后，Veritas将有向图转换为概率协议状态机。为了降低协议状态机的复杂性，Veritas使用Hopcroft–Karp算法对协议状态机执行最小化操作。Hopcroft–Karp算法将协议状态机中的等效状态合并为单个状态，减少了状态数量并提高了协议状态机的效率和可读性。

2015年，De Ruiter等人提出了一种使用状态机描述协议的方法[66]。他们采用了L*模型学习算法的改进版本，通过LearnLib来推断状态机，LearnLib提供了抽象的输入消息列表（也称为输入字母表）。De Ruiter等人使用测试工具将输入消息列表转换为发送到SUT的具体消息和接收的响应，然后将其转换为抽象消息类型。LearnLib分析了返回的消息类型，并对协议状态机进行了假设。分析不同的TLS实现会产生唯一且不同的状态机，表明该技术也可用于TLS指纹识别。这种方法的问题在于，在获得协议状态机之后，De Ruiter et al.手动分析状态机以发现特定实现中的逻辑漏洞，然后分析实现源代码以识别相应的问题，而不是使用自动化方法来测试协议实体。

2017年，Liu等人提出了一种技术，以解决在构造APTA树时，由于合并标记状态的错误而导致的状态机过度泛化的问题[67]。他们利用动态污点分析技术，依靠DECAF分析网络应用并构建APTA树。通过利用语义信息区分状态和合并相似状态，提高了APTA树中状态标注的准确性。用TCP协议和Agobot控制协议对该方法进行了测试，取得了较好的效果。

2019年，Pacheco等人研究了从文本规范自动学习协议规则（即，RFC），并将自动提取的协议规则应用于Fuzzer,以评估学习的规则[68]。评估表明，这种方法可以通过较少的测试用例发现与手动指定的系统相同的攻击。2022年，Pacheco et al.进一步提出了更全面的方法[69]。该方法包括三个关键步骤：（1）技术语言的大规模词表示学习，（2）协议文本到中间语言的零点学习映射，以及（3）从中间语言到特定协议有限状态机的基于规则的映射。他们从TCP、DCCP、BGPv4等协议规范中提取协议状态机，取得了良好的效果。

2020年，LI et al.提出了一种称为ReFSM[70]的协议状态推理方法，该方法通过考虑

实时分组捕获特征。该方法基于扩展有限状态机（EFSM），包括三个步骤：（1）消息类型识别，其使用先验关键词分析来提取协议关键词，并使用K-means算法来对消息进行分组以确定簇的数量，其中每个组被视为不同的消息类型。（2）EFSM构造和语义推理，构建协议转换自动机（PTA）树，该树接受所有协议会话，并使用K尾合并算法来简化协议状态机。（3）子数据集提取，提取包含消息中观察到的消息字段值的子数据集，用于进一步分析以搜索消息中字段之间的相关性。将状态相关场方法与聚类方法相结合，大大提高了提取的状态相关场的精度。然后，使用这些状态相关字段的信息来比较EFSM中的每个转换，并且通过确定所生成的A和B树是否具有相同的结构来完成状态机的合并操作。虽然ReFSM方法在状态相关字段的提取方面取得了一些改进，但仍然存在状态爆炸问题，由于协议状态机中存在大量的状态，影响了状态机推理的效率和准确性。

2021年，基于用于推断Mealy的L+算法的先前研究 $_{M}$ 机器，Goo等人提出了一种新的协议状态推断算法和解决方案[71]。他们通过对输入字符序列进行输入和等价查询，对客户机-服务器类型协议的Mealy机器进行了建模。在可行性测试中，该工具在Modbus和MQTT协议上进行了测试，结果良好。

5.1.3. 总结

总之，协议状态机可以系统地描述系统或网络协议的行为和状态转换，为模糊测试提供清晰的测试方向，并使模糊测试更具针对性和效率。利用协议状态机设计模糊测试用例，从结构上覆盖被测系统或网络协议的所有状态和转换，提高测试覆盖率。

从协议实现或网络流量中自动提取协议状态机，或者直接将协议行为映射到具有明确定义的状态和状态转换的状态机，是当前该领域最基本和重要的研究方向。一个重要的发展方向是基于主动推理，它利用人工学习系统不断地扩展原始样本集，反复推断状态机，从而减少对样本集完整性的依赖，提高状态机推理的准确性和效率。

5.2. 网络协议模糊化技术的评价

网络协议模糊化工具的数量不断增加，每个工具都集中于不同的测试目标和问题域。因此，为了确定哪种测试工具提供最好的结果，有必要评估网络协议模糊化技术。2021年，Natella et al.引入了用于有状态网络协议模糊化的基准测试套件ProFuzzBench [72]。基准测试套件包括一组具有代表性的开源网络服务器，用于流行的协议和自动化实验工具，如AFLNET和StateAFL。测试套件使用Docker实现，以实现可重复的实验，并支持不同模糊度的比较分析受控条件下的技术。

由于在有状态协议中并非所有状态都同等重要，并且模糊技术具有时间限制，因此需要一种有效的状态选择算法来过滤出具有较高优先级的良好状态。2022年，Liu et al.在ProFuzzBench基准测试套件上使用AFLNET工具评估了一组状态选择算法，并提出了一种改进的状态选择算法，称为AFLNETLegion[73]。

5.3. 总结

总之，当选择需要手动构建协议规范的模糊工具（如Peach和SNOOZE）时，可以根据现有协议相关信息选择自动化工具，以生成协议状态机并减少手动工作。另一方面，当测试人员需要找出哪些方法和模糊化工具在特定场景中更有效时，ProFuzzBench支持代表性工具的测试，并支持在受控条件下对不同的模糊化技术进行比较分析。

6. 最新技术水平

根据协议模糊化的一般过程、传统模糊化与第1节描述的协议模糊化的区别以及第5节介绍的协议模糊化技术的相关方法，需要考虑以下问题：

1. 如何生成或选择测试用例;
2. 如何根据SUT规范验证这些输入;
3. 如何指导SUT对更深的协议状态进行测试;
4. 如何提高协议模糊化的自动化水平。

在本节中，我们通过总结和比较当前基线和最新技术水平对上述协议模糊化问题的主要贡献，讨论RQ1和RQ2。

6.1. 测试用例生成和选择

在测试用例生成阶段，主要有两种方法：以Peach和Boofuzz为代表的基于生成的方法，以及以AFLNet为代表的基于突变的方法。

在基于生成的方法（例如Peach和Boofuzz）中，需要协议模板，该模板为协议定义所需的数据包格式。手动工作涉及获取协议模板，该模板指定了协议数据包的结构和预期行为。然后基于该模板生成测试用例，结合变化和突变来探索不同的输入场景。

另一方面，基于变异的方法，如AFLNet、StateAFL、SNPSFuzzer和SGFuzz,依赖于真实的网络流量来生成测试用例。通过捕获和分析实际的网络分组，应用预定义的技术来改变捕获的分组并生成期望的测试用例。此方法利用现有网络流量来探究潜在漏洞并测试协议实现的稳健性。

如第3节所述，基于生成的方法的主要挑战之一是依赖拥有方案先验知识的测试人员。它需要手动收集有关协议规范的信息，包括状态和转换条件。这种方法会导致大量的研究成本和专业知识需求。在基于突变的方法中，一个显著的挑战是当前的测试用例生成策略可能相当随机，其中通过突变生成的许多测试用例可能无法通过方案的确认，从而对基于突变的模糊技术的效率产生重大影响。

6.2. 输入确认

自动生成大量测试用例以触发SUT的意外行为的能力是模糊化的显著优点。但是，如果SUT具有输入验证机制，则这些测试用例很可能在执行的早期阶段被拒绝。在协议模糊化领域中，通常采用校验和和加密技术来确保输入数据分组的完整性和机密性。TaintScope 首先使用动态污染分析和预定义的规则来检测潜在的校验和点，然后改变字节以创建新的测试用例并更改校验和

以让所有创建的测试用例通过完整性验证。当某些测试用例可能使SUT崩溃时，它使用符号执行和约束求解来固定这些测试用例的校验和值。TaintScope 的主要挑战之一是它需要有关输入的大量信息，包括格式正确和格式错误的输入。输入的质量对检查点的识别有重大影响。因此，必须提供各种各样和全面的输入，包括各种格式和错误条件，以确保TaintScope 对检查点的全面测试和准确识别。

SecFuzz首先解决了协议消息中加密内容的问题。它为fuzzer提供了必要的密钥和加密算法，并通过充当客户端和服务器之间的中间人来改变加密的消息。目前，SecFuzz只能解决某些对称加密问题，不支持所有字段的变异。

6.3. 有状态协议模糊

在服务器-客户端模式下的协议实现上下文中，服务器是有状态的和消息驱动的。它从客户端接收一系列消息，处理这些消息，并发送适当的响应，服务器的响应取决于当前消息和当前内部服务器状态（由早期消息控制）。

在有状态协议模糊化中，基于生成和基于变异的方法使用不同的方法。

在基于生成的方法中，协议模板包含有关协议状态的相关信息。模糊器不仅生成测试用例，而且构造相应的状态机。在Peach中，协议状态机使用StateModel格式表示，而在Boofuzz中，协议状态机使用会话表示。一个值得注意的挑战是，在Peach和Boofuzz中，获取协议模板需要手动操作。

在基于变异的方法中，AFLNET首创了基于状态覆盖的灰盒模糊化，将自动状态模型推理和覆盖引导模糊化相结合。随后，StateAFL、SNPSFuzzer和SGFuzz进一步研究了基于状态反馈的协议模糊化。AFLNET利用状态码反馈构造协议状态机。在种子生成期间，如果生成的测试序列触发新的状态转换，则将其添加到种子语料库中，并且将新的状态合并到状态机中。当协议不提供状态代码时，AFLNET面临限制，使其无效。StateAFL和SNPSFuzzer通过对目标服务器的快照来推断协议的状态，并逐步构建协议状态机来指导模糊化过程。它们基于反馈度量（例如状态覆盖率）维护感兴趣的种子和状态。然而，它们的粒度处于进程级别，这可能导致状态机与实际协议状态不一致。SGFuzz自动识别程序代码中的状态变量，并通过构造状态转换树捕获协议状态变化。这种方法依赖于协议的源代码，不适用于闭源或专有协议的安全测试。

6.4. 自动化测试

第6.1节提到基于生成的协议模糊化技术（以Peach和Boofuzz为代表）在获取协议模板时需要人工帮助，并且它们的自动化水平相对较低。因此，可以采用自然语言处理（NLP）技术来帮助在协议模板提取阶段期间提取协议状态机。

在协议状态机提取阶段，有两种方法：被动推理和主动推理。以PEXT为代表的被动推理是基于消息聚类的协议状态机推理。主动推理研究主要基于L*（学习算法）算法，如LearnLib、Netzob、MACE等工具，假设存在能够提供准确答案的Oracle

成员资格查询和等价查询。首先，基于隶属度查询构造封闭且连续的观测表（OT），然后生成相应的候选状态机M。然后，使用等价查询来确定M是否与实际状态机一致。 如果是，则终止推理;否则，生成反例来重新推断。基于生成的方法的主要挑战之一是其效率取决于生成的查询的数量和Oracle对查询的响应速度。基于L*算法的主动推理需要生成大量的查询，导致效率低下。

近年来，协议状态推理的研究也集中于基于深度学习和机器学习技术的方法[70,74]。然而，这些方法仍然具有局限性和缺点。例如，需要大量的训练数据来训练模型，并且对于某些协议可能难以获得足够的训练数据，使得它们仅适用于简单的协议。

7. 技术发展背景和未来方向

在本节中，我们通过讨论协议模糊化和相关技术的一些可能的未来方向来回答RQ3。虽然我们不能准确预测协议模糊化研究的未来方向，但是我们可以在综述文献的基础上识别和总结一些趋势。

7.1. 开发背景

如前所述，自2013年发布以来，AFL在主流开源软件中发现了许多零日漏洞，并广泛用于测试不同目标。它的成功证明了代码覆盖引导技术在实际模糊化中的价值，并在模糊化技术的发展中发挥了重要作用。通过分析网络协议模糊化技术的发展趋势，可以看出AFL对网络协议模糊化技术的研究热点产生了重大影响。

参见图3,我们可以看到，在AFL发布之前，测试人员的研究重点是黑盒模糊技术。然而，在AFL发布后，灰盒模糊技术逐渐受到关注。基于这些发现，我们可以推测网络协议模糊化技术的发展趋势如下：

1. 随着时间的推移，测试人员可以根据状态覆盖率和代码覆盖率等反馈信息更容易地获取协议实现的相关信息，提高测试效率。因此，新发布的黑盒子模糊化工具的数量可能会逐渐减少，而灰盒子模糊化工具的数量将会增加，并主导网络协议模糊化格局。
2. 白盒模糊化工具的数量可能增加，因为随着一般网络协议模糊化技术变得更加成熟，对网络协议模糊化的具体要求也将出现;例如绕过加密密钥或完整性检查的模糊技术。完成这些特定的测试要求需要更全面地理解关于协议实现的相关信息，因此需要白盒模糊技术。

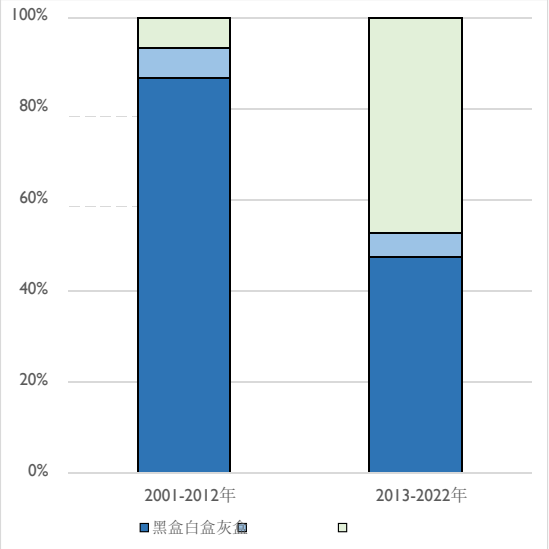

图3.AFL释放前后的技术百分比。

7.2. 未来方向

7.2.1. 基于发电的模糊化技术中的自动化

如前所述，网络协议模糊化中协议格式的分析需要相当大的努力，以区别于其他模糊化方法。人工分析网络协议格式在一定程度上提高了测试用例的有效性，但会带来大量的研究成本。此外，需要对不同的方案再次执行分析，从而导致较高的手动工作量。因此，手动访问网络协议格式的实际价值大大降低。解决此问题的可行解决方案包括：

1. 利用NLP等人工智能技术对协议规范文档和网络轨迹进行自动分析，从而提高协议格式分析的自动化水平和整体模糊自动化。然而，目前这种办法的实施还不是最佳的，而且还有许多未解决的问题。例如，在使用AI技术的自动分析之前，需要对协议文档进行手动注释，以帮助生成协议状态机。
2. 大型语言模型（LLM）是当前人工智能研究的一个热点领域，因为它们可以更好地理解和生成高质量的文本。在网络协议模糊化领域，需要考虑训练专用LLM来生成格式错误的协议分组。这种方法可以帮助克服现有的基于代的模糊技术中涉及的大量手工工作。

7.2.2. 模糊化技术的效率改进

在网络协议模糊化中，测试用例的有效性是发现未知漏洞的关键。有效的测试用例表现出高的突变率、接受率和代码覆盖率。优化测试用例选择策略，确保使用最小的测试用例集来发现尽可能多的潜在漏洞是未来网络协议模糊化研究的热点。可考虑的解决方案包括：

1. 通过改变变异策略，基于变异的协议模糊化技术可以生成更符合协议数据包特性的测试用例，从而提高验证的成功率。变异策略包括多种方法，如SPFuzz中使用的三级变异策略、上下文感知的结构化变异策略等。
2. 将神经网络模型与网络协议模糊化技术相结合，从海量数据中提取规则和知识。神经网络模型可以自动过滤掉无效的测试用例，从而提高模糊化的效率。

3. 在测试用例生成阶段应用定向模糊原则和施加某些约束，以生成覆盖目标代码/状态的测试用例。这些目标代码/状态段更有可能触发协议中的安全事件。测试这些代码/状态段可以导致更高的测试效率。

7.2.3. 网络协议模糊化中测试目标的多样性

在当前的网络协议模糊化技术中，测试目标主要遵循客户机/服务器体系结构。整个测试过程涉及fuzzer和协议实现之间的通信，fuzzer向协议实现发送测试用例。对多方协议或用户具有相对相同位置的协议的支持是有限的。此外，在网络层次结构方面，更侧重于测试简单的应用层协议，例如文件传输协议（FTP），而较少强调较低级别和更复杂的协议。

对于低级协议，相应的协议实现可能不存在。对于Fuzzer来说，在协议实现中观察异常行为（例如崩溃）以确定新漏洞的发现具有挑战性。然而，对于不同的协议，可以定义特定的安全事件，并且可以通过其他网络度量来间接评估测试的成功。例如，可以观察吞吐量的变化或路由表中的修改，以确定模糊器是否发现了漏洞。

7.2.4. 网络协议模糊化的相关技术

基于L*算法的主动学习方法目前广泛应用于协议状态机推理中，但仍面临诸多挑战。一个主要问题是过多的成员查询，这可能导致不完整的反示例样本集，并可能影响推断的协议状态机的准确性。另外，该方法在处理复杂协议时可能遇到泛化问题，这意味着它可能无法正确地推断协议状态机的某些行为。为了应对这些挑战，未来的研究可以探索以下需要扩展和改进的领域。

首先，将消息之间的顺序约束关系结合到协议状态机推理中可以更好地反映协议的实际行为，因为协议中的消息通常具有一定程度的时间顺序。其次，探索反例和正例之间的结构相关性，有助于更有效地生成反例，减少隶属查询的数量。此外，优化查询和反例的生成过程可以提高方法的性能，例如使用更有效的算法或策略来生成查询和反例。最后，将主动学习方法与其他形式化方法相结合，可以提高协议状态机推理的准确性和可靠性;例如，使用模型检验来验证推断出的协议状态机的正确性，或者将主动学习方法与其他机器学习技术相结合以提高推理的效率和准确性。

对于协议模糊化技术的评估，Profuzzbench目前仅支持代码覆盖率作为比较的度量，所包含的模糊化器不包含基于生成的模糊化技术。在未来的基准测试套件中，建议合并涉及基于生成的模糊技术的测试套件，并基于多个度量来评估协议模糊技术，这些度量包括发现的错误数量、测试相同错误所需的时间、状态覆盖率和其他相关指标。

8.结论

网络协议漏洞发现技术对于确保安全的网络通信至关重要。在各种漏洞发现技术中，模糊化由于其部署复杂度低、对SUT的先验知识要求极低而受到广泛关注。

本文回顾和总结了各种基于时间线的网络协议模糊化及相关技术的产生、发展和应用。本文首先介绍了网络协议模糊化技术的产生背景、工作原理和分类方法，然后从白盒、灰盒和黑盒模糊技术三个方面综述了网络协议模糊化技术的研究进展。这些观点中的每一个都包括对典型工具和方法的介绍。本文在分析近20年来约50篇相关论文的基础上，总结了网络协议模糊化技术的发展模式和存在的问题，并介绍了可与协议模糊化技术相结合的NLP技术。展望了该领域未来的研究方向，以期促进该领域研究工作的有效开展。

作者贡献：概念化，Z.Z.和J.Z.;方法学，Z.Z.和J.Z.;验证，Z.Z.，H.Z.，J.Z.和Y.Y.;形式分析，Z.Z.和H.Z.;调查，Z.Z.和H.Z.;资源，J.Z.;书写——原稿编制，Z.Z.和H.Z.;撰写——审稿和编辑，J.Z.和Y.Y.;可视化，Z.Z.;监督，J.Z.;项目管理，J.Z.所有作者均已阅读并同意手稿的已发表版本。

资金：本研究未获得外部资助。

数据可用性声明：本研究中提供的数据可根据通信作者的要求提供。

利益冲突：作者声明无利益冲突。

参考文献

1. CVE-2014-0160。可在线获取：https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160（2023年3月6日访问）。
2. 姓名：WRECK DNS 漏洞影响超过 1 亿台设备。可在线获取：https://www.bleepingcomputer.com/ news/security/name-wreck-dns-vulnerabilities-affect-over-100-million-devices/（访问日期：2023年3月6日）。
3. Miller, B.P.;弗雷德里克森湖;那么，B.UNIX实用程序可靠性的实证研究。*通讯ACM 1990,33,32–44*。[交叉引用]
4. Miller, B.P.;Koski,D.;Lee, C.P.;Maganty, V.;穆尔蒂河;Natarajan,A.;施泰德尔*Fuzz再次访问：UNIX实用程序和服务可靠性的再审视技术报告;威斯康星大学麦迪逊分校计算机科学系：*威斯康星州麦迪逊，美国，1995年。
5. Forrester, J.E.;米勒，B.P.使用随机测试对Windows NT应用程序的稳健性的实证研究。第4届USENIX Windows系统研讨会会议记录—第4卷，美国华盛顿州西雅图，2000年8月3日;WSS'00,第6.
6. Miller, B.P.;库克西，G.;穆尔，F.使用随机测试对macos应用稳健性的实证研究。2006年7月17日，美国密歇根州波特兰第一届随机试验国际研讨会论文集;第46-54页。
7. Liang,H.;Pei,X.;Jia,X.;Shen, W.Zhang,J.模糊：最新技术水平。*IEEE传输可靠性***2018年，67,1199-1218。**[交叉引用]
8. 奥赫勒特角违反模糊假设。*IEEE安全私人***2005年，第3页，第58–62页。**[交叉引用]
9. Viide, J.;Helin,A.;Laakso, M.;Pietikäinen,P.;Seppänen, M.;Halunen, K.;Puuperä，R.;伦宁模型推理辅助模糊化的经验。*WOOT 2008, 2, 1–2.*
10. Yang,H.;Zhang,Y.;Hu, Y.P.;Liu, Q.X.基于模糊化的IKE漏洞发现。*证券通讯网络***2013年，6, 889–901.**[交叉引用]
11. Yan,J.;Zhang,Y.;Yang,D.用于具有高度结构化输入的程序的基于结构化文法的模糊测试。*证券通讯网络***2013年6月1319-1330日。**[交叉引用]
12. Palsetia,N.;Deepa,G.;Khan, F.A.;Thilagam,P.S.;佩斯保护本机 XML 数据库驱动的 Web 应用程序免受 XQuery 注入漏洞的影响。*J.系统软件***2016年，122,93-109年。**[交叉引用]
13. Li,J.;Zhao,B.;Zhang、C.模糊：一项调查。*网络安全2018,1,1-13*。[交叉引用]
14. Manès, V.J.;Han,H.;Han,C.;Cha, S.K.;Egele, M.;Schwartz,E.J.;吴山模糊化的艺术、科学和工程学：一项调查。*IEEE传输软件工程师***2019,47,2312–2331。**[交叉引用]
15. Munea, T.L.;Lim,H.;肖恩角信息系统和应用程序的网络协议模糊测试：调查和分类学。*多媒体。工具应用***2016年，75,14745–14757。**[交叉引用]
16. Hu,Z.;潘网络协议模糊化技术系统综述。2021年IEEE第4届高级信息管理、通信、电子和自动化控制会议（IMCEC）会议记录，中国重庆，2021年6月18-20日;第4卷，第1000-1005页。
17. 德莫特不断发展的模糊艺术。美国内华达州拉斯维加斯DEF CON 14会议记录，2006年8月4日至6日;第1-25页。

18. Godefroid,P.;Levin, M.Y.;Molnar, D.A.自动白盒模糊测试。《网络和分布式系统安全研讨会论文集》，NDSS 2008,美国加利福尼亚州圣地亚哥，2008年2月8日;第8卷，第151-166页。
19. Godefroid,P.;Levin, M.Y.;莫尔纳尔SAGE：用于安全测试的白盒模糊。*通讯ACM 2012, 55, 40–44*。[交叉引用]
20. Kaksonen河;Laakso, M.;Takanen,A.通过规范突变和故障注入进行软件安全性评估。《新世纪的通信和多媒体安全问题会议录》：IFIP TC6/TC11 第五次通信和多媒体安全联合工作会议（CMS'01），德国达姆施塔特，2001年5月21日至22日;第173-183页。
21. 艾特尔湾*基于块的协议分析在安全测试中的优势Immunity Inc.*：佛罗里达州迈阿密海滩，2002年;第105卷，第106.
22. Peach Fuzzer。可在线获取：https://peachtech.gitlab.io/peach-fuzzer-community/（2023年3月2日访问）。
23. Banks, G.;Cova,M.;Felmetsger,V.;Almeroth, K.;Kemmerer河;维尼亚革SNOOZE：走向国家级网络协议。2006年8月30日至9月2日，第九届信息安全国际会议记录，ISC 2006,希腊萨莫斯岛;第4176卷，第343–358页。
24. Abdelnur, H.J.;州，R.;费斯托尔角KiF：有状态的 SIP 模糊器。第一届IP电信原理、系统和应用国际会议记录，美国纽约州纽约市，2007年7月19日;第47-56页。
25. 动态应用程序安全测试工具（DAST）。BESTORM。可在线获取：https://www.beyondsecurity.com/solutions/ bestorm-dynamic-application-security-testing（2023年3月2日访问）。
26. Pure-Python全自动无人值守的模糊框架。可在线获取：https://github.com/OpenRCE/sulley（2023年3月2日访问）。
27. Sulley Fuzzing框架的分支和继承者。可在线获取：https://github.com/jtpereyda/boofuzz（2023年3月2日访问）。
28. 瓦格努山Autodafe：软件折磨的行为。第22届混沌通信大会会议记录，混沌计算机俱乐部，德国柏林，2005年8月26日;第47-58页。
29. Gorbunov, S.;Rosenbloom,A.自动模糊：自动化网络协议模糊化框架。*Ijcsns 2010, 10, 239*。
30. Kitagawa, T.;Hanaoka,M.;科诺湾Aspfuzz：一个基于应用层协议的状态感知协议模糊器。IEEE计算机和通信研讨会会议记录，意大利里乔内，2010年6月22-25日;第202-208页。
31. Tsankov, P.;Dashti, M.T.;盆地，D.SECFUZZ：模糊测试安全协议。2012年第7届软件自动化测试国际研讨会会议记录，瑞士苏黎世，2012年6月2日至3日;第1-7页。
32. Rontti, T.;Juuso,A.M.;Takanen,A.使用基于前瞻性规范的模糊化来防止NGN网络中的DoS攻击。*IEEE通讯放大***2012年，50,164–170。**[交叉引用]
33. Han,X.;Wen,Q.;Zhang,Z.一种用于网络协议漏洞检测的基于变异的模糊测试方法。《2012年第二届计算机科学与网络技术国际会议记录》，中国长春，2012年12月29日至31日;第1018-1022页。
34. Brubaker,C.;Jana, S.;Ray, B.;Khurshid, S.;什马提科夫在SSL/TLS实现中使用frankencerts进行证书验证的自动对抗测试。2014年5月18日至21日在美国加利福尼亚州伯克利举行的2014年IEEE安全和隐私研讨会论文集;第114-129页。
35. SSL和TLS协议测试套件和Fuzzer。可在线获取：https://github.com/tlsfuzzer/tlsfuzzer（2023年3月2日访问）。
36. Gascon,H.;Wressnegger,C.;Yamaguchi, F.;Arp,D.;里克湖脉冲星：专有网络协议的状态黑盒模糊化。通讯网络的安全与保密学报：第11届EAI国际会议，SecureComm 2015,美国德克萨斯州达拉斯，2015年10月26日至29日;第330-347页。
37. 索莫罗夫斯基TLS库的系统模糊与测试。2016年10月24日至28日在奥地利维也纳举行的2016年ACM SIGSAC计算机和通信安全会议记录;第1492–1504页。
38. Ma,R.;Wang,D.;Hu,C.;Ji,W.;薛J.使用基于规则的状态机为有状态网络协议模糊化生成测试数据。*清华科学技术***2016年，21,352–360。**[交叉引用]
39. 布伦贝格斯湾;瓦兰迪河蜂鸣声：网络协议系统逆向工程与分析中的位感知模糊化框架。2017年10月23日至25日在美国马里兰州巴尔的摩举行的MILCOM 2017 IEEE军事通信会议（MILCOM）会议记录中;第707-712页。
40. 用 Python 编写的 OpenAPI 2.0（Swagger）Fuzzer。可在线获取：https://github.com/Teebytes/TnT-Fuzzer（2023年4月4日访问）。
41. HTTP API 测试框架。可在线获取：https://github.com/KissPeter/APIFuzzer（2023年4月4日访问）。
42. 美国Fuzzy Lop.可在线获取：https://lcamtuf.coredump.cx/afl/（2023年3月2日访问）。
43. Wang,T.;Wei,T.;Gu,G.邹湾TaintScope：用于自动软件漏洞检测的校验和感知定向模糊工具。2010年5月16日至19日，美国加利福尼亚州奥克兰，2010年IEEE安全和隐私研讨会论文集;第497–512页。
44. Atlidakis,V.;Godefroid,P.;波兰丘克山限制器：有状态rest api模糊化。在2019年IEEE/ACM第41届软件工程国际会议（ICSE）会议记录中，加拿大QC蒙特利尔，2019年5月25日至31日;第748-758页。
45. Fiterau-Brostean,P.;Jonsson,B.;Merget河;De Ruiter, J.;Sagonas, K.;索莫罗夫斯基使用协议状态模糊分析DTLS实现。第29届USENIX安全研讨会论文集，在线，2020年8月12日至14日;第2523-2540页。

46. libFuzzer - 用于覆盖率引导的模糊测试的库。可在线查阅：http://llvm.org/docs/LibFuzzer.html（2023年4月4日访问）。
47. Petsios, T.;Tang,A.;Stolfo, S.;Keromytis,A.D.;贾纳湾Nezha：高效的独立于域的差异测试。2017年5月22日至26日在美国加利福尼亚州圣何塞举行的2017年IEEE安全和隐私（SP）研讨会论文集;第615-632页。
48. 宋，C.;Yu,B.;Zhou,X.;Yang, Q.SPFuzz：用于有状态网络协议模糊化的分层调度框架。*IEEE Access 2019, 7, 18490–18499*。[交叉引用]
49. Chen,Y.;Lan, T.;Venkataramani,G.探索有效的模糊策略来分析通信协议。关于围绕软件转型形成生态系统的第三届ACM研讨会论文集，英国伦敦，2019年11月15日;第17-23页。
50. Pham, V.T.;Böhme, M.;Roychoudhury,A.AFLNet：用于网络协议的灰盒模糊器。2020年10月24日至28日在葡萄牙波尔图举行的2020年IEEE第13届软件测试、确认和验证（ICST）国际会议记录;第460-465页。
51. Zou, Y.H.;Bai,J.J.;Zhou,J.;Tan, J.;Qin,C.;胡S.M.TCP-Fuzz：使用Fuzzing检测TCP堆栈中的内存和语义缺陷。USENIX年度技术会议论文集，虚拟事件，2021年7月14日至16日;第489-502页。
52. 纳特拉河Stateafl：有状态网络服务器的灰盒模糊。 *呼气软件工程师***2022, 27, 191.**[交叉引用]
53. Li,J.;Li,S.;Sun,G.;Chen, T.;Yu, H.SNPSFuzzer：使用快照的状态网络协议的快速灰盒模糊器。*IEEE传输信息法证安全* **2022,17,2673–2687。**[交叉引用]
54. Ba,J.;Böhme, M.;Mirzamomen,Z.;Roychoudhury,A.状态灰盒模糊。在第31届USENIX安全研讨会（USENIX Security 22）的会议记录中，美国马萨诸塞州波士顿，2022年8月10日至12日;第3255–3272页。
55. Alshmrany, K.;科代罗湖在网络协议实现中查找安全漏洞。arXiv 2020年，arXiv：2001.09592。
56. 莱塔角;Mermoud, K.;达吉尔山Scriptgen：一个自动化的honeyd脚本生成工具。第21届计算机安全应用年会（ACSAC'05），美国亚利桑那州图森，2005年12月5日至9日;第页12.
57. Shevertalov,M.;曼科里迪斯湾用于提取网络应用协议的逆向工程工具。第14届逆向工程工作会议记录（WCRE 2007），加拿大不列颠哥伦比亚省温哥华，2007年10月28日至31日;第229-238页。
58. Comparetti, P.M.;Wondracek,G.;克鲁格尔角;科尔达湾Prospex：方案规范提取。2009年5月17日至20日在美国加利福尼亚州奥克兰举行的2009年第30届IEEE安全和隐私研讨会论文集;第110-125页。
59. Wondracek,G.;Comparetti, P.M.;克鲁格尔角;Kirda, E.;安娜，S.S.S.自动网络协议分析。《网络和分布式系统安全研讨会论文集》，NDSS 2008,美国加利福尼亚州圣地亚哥，2008年2月8日;第8卷，第1-14页。
60. 戈尔德，E.M.限制中的语言标识。 *信息控制 1967, 10, 447–474*。[交叉引用]
61. 安格林角从查询和反例学习规则集。 *信息计算***1987,75,87–106。**[交叉引用]
62. Cho, C.Y.;Babic´，D.;Poosankam,P.;Chen, K.Z.;Wu, E.X.;宋，D.MACE：协议和漏洞发现的模型推理辅助共栖探索。第20届USENIX安全会议论文集，美国加利福尼亚州旧金山，2011年8月8日至12日;第页10.
63. 协议逆向工程、建模和模糊。可在线获取：https://github.com/netzob/netzob（2023年3月2日访问）。
64. 贝多，文学硕士使用生物信息学算法的网络协议分析。*Toorcon 2004, 26, 1095–1098*。
65. Wang,Y.Zhang,Z.;Yao,D.D.;Qu,B.;郭湖从网络跟踪推断协议状态机：概率方法。第九届应用密码学和网络安全国际会议记录，西班牙内贾，2011年6月7日至10日;第1-18页。
66. De Ruiter, J.;波尔，E.TLS实现的协议状态模糊化。第24届USENIX安全研讨会（USENIX Security 15）会议记录，美国华盛顿特区，2015年8月12日至14日;第193-206页。
67. Liu,Y.Xie,P.;Wang,Y.Liu,M.基于动态污点分析的语义相关标记方法在协议状态机推理中的应用。2017年第3届IEEE计算机与通信国际会议（ICCC）会议记录，中国成都，2017年12月13日至16日;第258-262页。
68. Jero, S.;Pacheco,M.L.;Goldwasser,D.;尼塔罗塔鲁角利用文本规范对网络协议进行基于语法的模糊化。AAAI人工智能会议论文集，Honolulu,HI,2019年1月27日至2月1日;第33卷，第9478-9483页。
69. Pacheco,M.L.;von Hippel,M.;Weintraub,B.;Goldwasser,D.;尼塔罗塔鲁角通过从协议规范文档中提取有限状态机的自动攻击合成。2022年7月27日在美国加利福尼亚州旧金山举行的IEEE安全和隐私（SP）研讨会论文集;第51-68页。
70. Lin, Y.D.;Lai, Y.K.;Bui, Q.T.;赖，Y.C.ReFSM：从协议包跟踪到扩展有限状态机测试生成的逆向工程。*J.网络计算申请***2020, 171, 102819.**[交叉引用]
71. Goo, Y.H.;Shim, K.S.;Kim, M.S.基于自动逆向工程技术的协议规范提取方法。于2018年11月28日于中国北京召开的第二届电信与通信工程国际会议论文集;第16-21页。
72. Natella河;范，V.T.Profuzzbench：有状态协议模糊化的基准。第30届ACM SIGSOFT软件测试和分析国际研讨会论文集，New York, NY, USA, 2021年7月11日;第662-665页。

73. Liu,D.Pham, V.T.;Ernst, G.;Murray, T.;鲁宾斯坦，B.I.状态选择算法及其对有状态网络协议模糊化性能的影响。2022年IEEE软件分析、演化和再造国际会议（SANER）会议记录，Honolulu,HI,USA,2022年3月15日至18日;第720-730页。
74. Hagos, D.H.;Engelstad, P.E.;Yazidi,A.;Kure,Ø.基于机器学习技术的被动测量的一般TCP状态推断模型。*IEEE Access 2018, 6, 28372–28387*。[交叉引用]