基于协议状态机遍历的模糊测试优化方法

张洪泽[1],洪 征[1],周胜利[2],冯文博[1]
1.中国人民解放军陆军工程大学 指挥控制工程学院,南京 210000
2.浙江警察学院 计算机与信息技术系,杭州 310000

摘 要:针对现有的协议模糊测试技术存在报文重复交互、输入盲目等问题,提出一种基于协议状态迁移遍历的模糊测试优化方法。该方法将协议状态迁移遍历问题转化为中国邮路问题,求解遍历所有协议状态迁移的最短路径,并依据该最短路径对各个状态迁移进行测试。在测试过程中,通过分析协议实体在执行测试用例后的响应报文,动态调整后续的报文输入,进而避免无效交互。同时利用UIO序列判断协议实体状态是否发生异常迁移,及时检测协议逻辑漏洞。实验结果表明,该模糊测试优化方法可以显著提高测试效率与漏洞挖掘能力。

关键词:模糊测试;协议状态机;协议状态迁移;中国邮路问题;UIO序列
文献标志码:A **中图分类号:**TP309 doi:10.3778/j.issn.1002-8331.1811-0190

张洪泽,洪征,周胜利,等.基于协议状态机遍历的模糊测试优化方法.计算机工程与应用,2020,56(4):82-91.
ZHANG Hongze, HONG Zheng, ZHOU Shengli, et al. Fuzzing optimization method based on protocol state migration traversal. Computer Engineering and Applications, 2020, 56(4):82-91.

Fuzzing Optimization Method Based on Protocol State Migration Traversal

ZHANG Hongze[1], HONG Zheng[1], ZHOU Shengli[2], FENG Wenbo[1]
1.Institute of Command and Control Engineering, Army Engineering University of PLA, Nanjing 210000, China
2.Department of Computer and Information Technology, Zhejiang Police College, Hangzhou 310000, China

Abstract: There are many problems such as repetitive message interaction, blind input and so on in the current protocol fuzzing techniques. This paper presents a fuzzing method based on protocol state migration traversal. The method transforms protocol state migration traversal into a Chinese postman problem, and obtains the shortest path traversing all protocol state transitions. The method then tests each state transition according to the shortest path. In the process of fuzzing, message input is dynamically adjusted through analyzing the response message of protocol entity so as to avoid invalid interaction. In addition, the UIO sequence is used to determine whether the protocol entity state is abnormally migrated or not in order to detect the protocol logic vulnerability in time. Experimental results show that the fuzzing optimization method can significantly improve the fuzzing efficiency and the vulnerability mining ability.

Key words: fuzzing; protocol state machine; protocol state migration; Chinese postman problem; UIO sequence

1 引言

协议漏洞挖掘是保证网络通信安全的重要手段。模糊测试是目前最常用的协议漏洞挖掘方法,它通过向协议实体输入变异的报文,监控协议实体的运行状况,分析协议实体发生的异常,从而发现潜在的安全漏洞。模糊测试具有自动化程度高、发现的漏洞实际可用等优点[1]。

根据输入报文相互之间是否存在关联关系,网络通信协议可分为无状态协议(Stateless Protocol)和有状态协议(Stateful Protocol)两类[2]。无状态协议是指报文发送方输出的各个报文之间没有关联性,例如ICMP协议的每个请求报文各自独立,相互之间没有关联关系。而对于有状态协议,协议实体会记录所接收到的报文信

基金项目:国家重点研发计划(No.2017YFB0802900)。
作者简介:张洪泽(1993—),男,硕士研究生,研究领域为网络信息安全,E-mail:zhz135@qq.com;洪征(1979—),男,博士,副教授,研究领域为网络信息安全;周胜利(1982—),男,博士,工程师,研究领域为网络信息安全;冯文博(1994—),男,硕士研究生,研究领域为网络信息安全。
收稿日期:2018-11-16 **修回日期:**2019-01-11 **文章编号:**1002-8331(2020)04-0082-10
CNKI **网络出版:**2019-03-21, http://kns.cnki.net/kcms/detail/11.2127.TP.20190319.1644.020.html

息，在处理报文后可能会出现协议状态的变化，例如FTP协议与SMTP协议都属于有状态协议。相比于无状态协议模糊测试，对有状态协议进行模糊测试更复杂。因为当测试用例与协议实体状态不匹配时，测试用例会被直接丢弃，为了保证测试用例尽可能被协议实体接受，测试时需要依赖协议的状态模型，根据协议实体所处的协议状态输入测试用例[3]。

网络通信协议的模糊测试最早可追溯到1999年芬兰Oulu大学研发的网络协议安全测试软件PROTOS[4]。PROTOS发现了不少的协议实体程序的安全漏洞，但是它不是通用的测试框架，软件灵活性较差，应用范围狭窄。2002年，Aitel开发了一款通用协议测试框架工具SPIKE[5]。SPIKE是一款可以定制的模糊测试器框架，能够方便地实现代码重用，但是不能灵活描述报文中字段之间的约束关系，并且SPIKE只适用于无状态网络协议的测试，应用范围受限。2013年，IOACTIVE发布了著名模糊测试框架Peach 3.0[6]。Peach 3.0是一个灵活的模糊测试框架，使用XML文件作为测试脚本来定义测试对象以及测试方法，充分利用了XML文件低耦合、易分离的优点，促进了模糊测试代码的重用。除了上述测试工具，研究者也提出不少模糊测试的优化方法。例如Kitagawa等人提出状态级(State-Level)变异的模糊测试方法AspFuzz[7]。AspFuzz可以自动构造与正常报文交互顺序相违背的测试序列，发现报文异常输入顺序引起的协议缺陷，但是其测试方法存在组合爆炸的问题。Zhao等人提出基于回归协议状态机的模糊测试方法RFSM-Fuzzing[8]。该方法通过状态引导对每个协议状态进行测试，对协议漏洞挖掘有较好的效果，但是测试过程存在冗余交互较多、测试效率低下的问题。Pan等人提出一种基于文法驱动的协议测试方法[9]。该方法利用高阶属性文法建立数据模型，通过遍历语义树指导测试用例生成，有效避免了协议字段的漏测或者重复测试。Ma等人提出基于规则的状态机(Rule-Based State Machine)与状态规则树(Stateful Rule Tree)来指导模糊测试序列生成的方法[10]。作者认为如果某个状态迁移对应的输入不会引发协议实体异常，那么则认为该迁移是安全迁移，在测试期间可以将其移除从而缩小测试空间，但是这种方法可能出现测试不完全的问题。总体上看，现有的模糊测试技术没有充分利用协议状态转换关系开展测试，也没有对输入报文和响应报文进行必要的相关性分析，在测试实施过程中较为盲目，测试效率低下，测试的覆盖率偏低。

本文旨在提高测试效率与测试覆盖率，提出一种针对有状态协议的模糊测试优化方法。

2　问题分析

2.1　有状态网络协议的模糊测试

有状态网络通信协议通常采用确定有限状态机(Deterministic Finite Automaton，DFA)作为协议交互的形式化描述模型。确定有限状态机的定义如下所示[11]。

定义1(确定有限状态机) 定义为一个六元组 $M=(S,I,O,\delta,\beta,T)$，其中：

$S=\{s_0,s_1,\cdots,s_n\}$ 是有限状态集合，$s_0\in S$ 表示 M 的初始状态，且在任意时刻，M 只能处于某一状态 s_i，有限状态机由 s_0 状态开始接收输入；

$I=\{i_1,i_2,\cdots,i_m\}$ 是有限输入符号集合；

$O=\{o_1,o_2,\cdots,o_m\}$ 是有限输出符号集合；

δ:$S\times I\rightarrow S$ 是状态迁移函数，它是一对一的映射；

β:$S\times I\rightarrow O$ 是状态输出函数；

T 是终结状态集合。

当DFA应用于网络协议描述时，$I=\{i_1,i_2,\cdots,i_m\}$ 表示协议实体可接受并正常处理的输入报文集合，$O=\{o_1,o_2,\cdots,o_m\}$ 表示协议实体输出的报文集合，以 M 表示协议状态机。以FTP协议[12]状态机为例，其输入输出报文类型的抽象符号如表1所示，M 包括7个状态，状态集合 $S=\{s_1,s_2,s_3,s_4,s_5,s_6,s_7\}$。状态机 M 如图1所示，其中 s_0 到 s_1 的迁移表示为 i_1/o_1，它表示协议实体处于 s_0 状态时，如果接受USER类型报文(用 i_1 表示)，

表1　FTP协议输入输出报文类型的抽象符号列表

输入报文类型	输入符号	输出报文类型	输出符号
USER	i_1	331	o_1
PASS	i_2	230	o_2
PWD	i_3	257	o_3
TYPE	i_4	200	o_4
STOR	i_5	150	o_5
MKD	i_6	250	o_6
RETR	i_7	221	o_7
RMD	i_8		
CWD	i_9		
DELE	i_{10}		
CDUP	i_{11}		
QUIT	i_{12}		

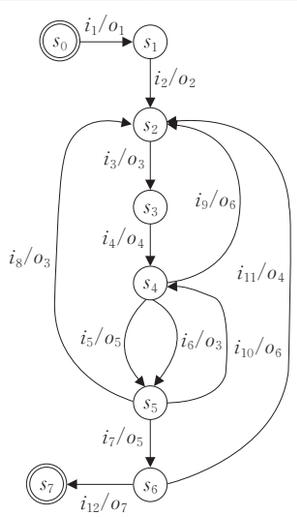

图1　FTP协议状态机

将输出331类型报文(用 o_1 表示),同时协议实体状态迁移至 s_1 状态。

对于有状态协议模糊测试,为了提高测试覆盖率,通常需要使协议实体处于某一指定状态,然后输入与状态相对应的测试用例进行测试。测试时需要依赖网络协议交互规则,根据协议状态机构造测试序列。传统的有状态协议模糊测试过程[8,13-14]主要包括三个步骤:首先,通过正常的报文交互,将协议实体引导至某个待测协议状态,这些正常交互报文所构成的序列称为前置引导序列。然后,向协议实体输入与被测状态相对应的变异报文,所涉及的变异报文被称为测试用例。如果检测到协议实体在处理测试用例后出现系统崩溃或者停止响应等异常情况,则需保存错误现场并进一步分析。最后,如果未检测到异常,还需按照特定顺序输入正常报文将协议实体引导至终止状态,准备下一轮测试。这些正常报文所构成的序列被称为回归序列。前置引导序列、测试用例与回归序列共同组成测试序列。如对于图1所示的FTP协议状态机,生成测试序列如表2所示,其中,$\sim i$ 表示测试用例。

表2 测试序列示例

被测状态	前置引导序列	测试用例	回归序列
s_0	—	$\sim i_1$	$i_1, i_2, i_3, i_4, i_5, i_7, i_{12}$
s_1	i_1	$\sim i_2$	$i_2, i_3, i_4, i_5, i_7, i_{12}$
s_2	i_1, i_2	$\sim i_3$	$i_3, i_4, i_5, i_7, i_{12}$
s_3	i_1, i_2, i_3	$\sim i_4$	i_4, i_5, i_7, i_{12}
s_4	i_1, i_2, i_3, i_4	$\sim i_5$	i_5, i_7, i_{12}
s_5	i_1, i_2, i_3, i_4, i_5	$\sim i_7$	i_7, i_{12}
s_6	$i_1, i_2, i_3, i_4, i_5, i_7$	$\sim i_{12}$	i_{12}

2.2 现有方法测试不足

评价模糊测试方法的优劣可以采用测试效率与测试覆盖率两个评价指标。现有的模糊测试方法进行有状态协议的模糊测试时,主要存在以下三方面的不足:

(1)辅助类型报文交互时间耗费高,测试效率低下。

网络协议模糊测试需要将报文通过网络传输到协议实体程序,报文在传输节点的等待、处理以及在网络中的传输都需要时间,每个发送的报文时间耗费是不可忽略的问题[15]。现有的测试方法通常只侧重于提高测试用例的有效性,没有考虑测试流程的优化,导致测试序列中只有少部分报文属于测试用例,其他大部分是前置引导序列、回归序列等辅助报文,这些辅助报文会产生额外的时间开销,使得单位时间内成功完成测试的测试用例数量很少,引发协议实体异常概率相应偏小,测试效率偏低。

(2)不能保证所有的状态迁移都进行测试,测试覆盖率不高。

在对有状态协议进行模糊测试时,需要考虑被测协议实体程序的代码覆盖率,程序代码覆盖得越充分,表明测试越完善[16]。对于程序源码未知的协议实体进行模糊测试,难以确定代码覆盖率,因此通常利用协议状态迁移覆盖率对测试方法进行评估,协议状态迁移覆盖得越充分,表明测试越完善。然而,现有的方法难以保证对所有的状态迁移都进行测试,协议实体程序代码测试覆盖不够充分,影响漏洞挖掘效果。

(3)无法保证输入报文与协议实体状态相对应,产生无效的报文交互。

协议实体程序在处理报文时,需要经过语法解析、语义解析与程序执行三个步骤[17]。测试用例触发协议实体程序异常或者正常处理输入报文必须首先通过前两步,即测试用例或者正常报文需要满足两个条件:协议格式语法约束以及与协议实体状态相对应。而目前模糊测试采用的主要是固定模式的测试方法,即在每次测试之前预先生成测试序列,测试时不再对测试序列进行调整。这种方法存在盲目性。因为测试用例被协议实体直接抛弃还是接收处理是不确定的,输入测试用例后协议实体状态是无法预知的,可能出现后续报文(例如回归序列)与协议实体状态不对应的情形,这导致报文交互是无效的。

针对上述分析,本文提出一种基于协议状态迁移遍历的模糊测试优化方法。首先将寻找遍历状态机所有迁移的路径问题转化为有向图的中国邮路问题,通过求解有向图的中国邮路问题获取协议状态机的最优遍历路径;然后遵循最优遍历路径的所有迁移依次进行测试;测试时根据响应报文信息推断协议实体状态,动态选择合适的测试用例或者正常报文,确保测试过程中发给协议实体的报文与其协议状态相对应,避免无效的报文交互,进而提高测试效率。此外,通过精心构造的唯一输入/输出序列(Unique Input/Output Sequences,UIO)检测协议实体异常,及时发现协议状态的异常迁移。

3 模糊测试优化策略

3.1 基本概念

在实施模糊测试时,协议实体对每个测试用例会给出相应的反馈信息,例如当测试用例不能被正确地接收处理时,协议实体会通过响应报文的形式告知。例如,FTP服务器会反馈包含“500 Syntax error unknown command”信息的报文;SMTP服务器会反馈包含“503 Bad sequence of commands”信息的报文。为方便表述,当协议实体处于状态 s_i,输入报文 i_m,将协议实体处理报文 i_m 后输出的响应报文 o_n 称为由输入 i_m 与状态 s_i 确定的期望响应报文。如果得到的输出不是正常的响应报文 o_n,而是诸如提示输入报文语法错误、错误命令序列、命令未实现、动作未完成等负面回答信息的报文,或者是非状态 s_i 的响应报文,统一称之为由输入 i_m 与状态 s_i 确定的非期望响应报文,表示为 $\overline{o_n}$。需要

说明的是，在测试时，若在设定时间阈值内未返回任何报文，也视为得到的是非期望响应报文。

网络协议模糊测试过程主要是为了发现能够触发协议实体异常的测试用例。然而很多测试用例并不会触发协议实体异常，这些测试用例主要可以分为两类：一类是由于测试用例存在畸形数据，如果测试用例无法通过语法验证，它们会被协议实体直接抛弃，没有进行程序执行，一般不会引发协议实体程序异常和状态的跳转，此时协议实体会输出非期望响应报文。另一类是测试用例符合协议格式规范，能够正常被程序执行处理，可以引发状态跳转并输出期望响应报文。例如，针对某些FTP协议实体程序进行模糊测试时，如果对MKD(创建目录)命令的参数变异生成的某些测试用例，这些测试用例可能仍属于正常报文，程序输出的是期望响应报文。正是因为这两类测试用例的存在，使得输入测试用例之前无法预知在输入测试用例之后协议实体状态的变化。

3.2　协议实体状态迁移检测

协议模糊测试为了检测协议实体程序能否正确处理畸形数据，判断协议实体程序是否有足够的健壮性，在模糊测试时，不仅需要利用调试器查看内存使用情况等方法发现协议实体内存访问异常等问题，还要考虑协议状态异常迁移问题，发掘协议实体程序的安全缺陷。

以协议实体处于 s_i 状态，输入正常报文 i_m 后协议实体迁移至状态 s_j 为例。所谓的协议状态异常迁移，在一致性测试领域[18]，指的是当协议实体接收到 i_m 后，协议实体状态未遵循协议状态机迁移至 s_j 状态，而是其他非 s_j 状态。这种情形说明协议实体间的交互存在偏差，是一种潜在错误。在对协议进行模糊测试时，当发送由正常报文 i_m 变异生成的测试用例后，如果利用查看调试器等方法未发现异常，但是协议实体状态既不处于 s_i 状态也不处于 s_j 状态，说明协议实体出现状态异常迁移。这种由测试用例引发协议状态异常迁移意味着协议实体程序不够健壮，通信过程不可靠，是一种安全缺陷。因此，协议实体状态异常迁移检测非常必要。为了及时发现协议状态是否异常迁移，需要在测试后准确地对协议实体所处的状态进行判断。

对于黑盒测试，无法通过观察协议实体内部动作判断协议实体状态，可以采用向协议实体发送UIO序列[19]判断协议实体所处的状态。所谓UIO序列，指的是当协议实体处于某一状态 s_i 时，向协议实体发送一个报文序列，报文序列表示为 $P=(i_x, i_y, \cdots, i_z)$，$i_x, i_y, i_z \in I$，协议实体输出报文组成的序列与其他非 s_i 状态下输入 P 而得到的输出报文组成的序列都不同，报文序列 P 称为状态 s_i 的UIO序列，记为 $\mathrm{UIO}(s_i)$。UIO序列可以唯一标识协议状态 s_i。例如在图1所示的状态机中，当协议实体处于 s_2 状态时，输入报文序列 (i_3, i_4) 会输出报文序列 (o_3, o_4)，而在其他任何状态下输入 (i_3, i_4) 构成的序列都不会输出报文序列 (o_3, o_4)，因此报文序列 (i_3, i_4) 为状态 s_2 的UIO序列。

文献[8]利用UIO序列，通过观察协议实体的输出报文序列检测协议实体所处的状态。具体而言，其方法在输入测试用例后，为了判断协议实体是否处于 s_i 状态，向协议实体输入 s_i 状态所对应的UIO序列。如果协议实体输出的报文序列与UIO序列对应的输出报文序列相符，则说明协议实体处于 s_i 状态，否则说明协议实体不处于 s_i 状态。这种方法在每次输入测试用例之后，都需要输入对应的整个UIO序列，使得测试时不得不发送与接收额外的报文，影响测试效率。此外，文献[8]方法也存在UIO序列与状态不对应而导致报文无效交互的问题。为解决此问题，本文提出了利用交叉迭代策略构造UIO序列。

3.3　模糊测试优化基本思想

模糊测试优化基本思想如下：首先需要在协议状态机上寻找遍历状态机所有状态迁移，并且尽可能短的路径，为方便叙述，该路径以OPT表示。测试过程从OPT对应的第一个状态开始实施，以协议状态 s_i 为例，状态 s_i 在OPT中邻接的下一个协议状态为 s_j，状态 s_i 与状态 s_j 之间的迁移对应的正常输入报文为 i_m。测试时向协议实体输入由 i_m 变异生成的测试用例，检测协议实体是否出现了异常，如果出现了异常，需要保存输入数据待进一步分析调试。如果没有出现异常，需要推断协议实体的状态，以决定如何实施下一步的测试。协议实体的状态推断主要依据协议实体的响应报文。对于输入的测试用例，如果协议实体的响应报文是协议状态 s_i 和输入 i_m 所对应的期望响应报文，那么表明测试用例被协议实体正常接收处理，协议实体状态已经处于 s_j 状态，可以输入下一个状态迁移所对应的测试用例进行下一步测试。如果协议实体的响应报文属于协议状态 s_i 和输入 i_m 所对应的非期望响应报文，那么协议实体仍可能处于 s_i 状态，也可能由于测试用例的作用，跳转至状态 s_i 之外的状态。在这种情况下，继续需要输入正常输入报文 i_m，观察协议实体的响应。如果响应报文是状态 s_i 和输入 i_m 所对应的非期望响应报文，那么在输入报文 i_m 之前，协议实体可能出现异常，需要保存输入数据进一步分析。如果响应报文是状态 s_i 和输入 i_m 所对应的期望响应报文，那么此时协议实体已经被引导至 s_j 状态，继续输入 s_j 状态与下一状态的迁移对应的测试用例进行测试。为了检测在输入 i_m 变异生成测试用例之后协议实体状态是否发生异常迁移，需要在输入测试用例之后输入状态 s_i 对应的UIO序列，那么在测试时输

入由 i_m 变异生成的测试用例之后，同时保证后续输入的其他报文可以构成状态 s_i 对应的UIO序列。如此依次推进，直至OPT的终止状态完成一轮测试。

在模糊测试优化实施过程中主要有两个问题：

(1)如何在协议状态机上寻找遍历协议状态机所有迁移，并且尽可能短的路径。

(2)在输入由 i_m 变异生成的测试用例后，如何保证后续输入的报文可以构成状态 s_i 对应的UIO序列。

下文将依次进行论述。

3.4 最优遍历路径获取

该节将解决3.3节提出的问题(1)。需要说明的是，之所以要在协议状态机上寻找遍历协议状态机所有迁移，并且尽可能短的路径，是因为如果依据OPT上状态迁移的顺序依次对每个状态迁移进行测试，可以保证所有状态迁移都能够被测试，尽可能短的路径可以减小异常定位分析的工作量。

有向图的中国邮路问题(Digraph Chinese Postman Problem，DCPP)是寻找经历有向图中每条有向边至少一次并且长度最短的遍历路线，这种遍历路线也被称为最优邮递路线[20]。可将协议状态机映射为有向图，然后再基于DCPP，寻找一条经历有向图中所有有向边并且长度最小的遍历路线，根据获取的有向图遍历路线，找到遍历协议状态机所有迁移的最短路径。依据该遍历路径上状态迁移的先后顺序进行测试，即可以对所有状态迁移进行测试，有效避免漏测问题。下面先给出有向图的定义。

定义 2(有向图) 以二元组表示一个有向图，记为 $DG=(V,E)$，其中 DG 是有向图的名称，$V=\{v_0,v_1,\cdots,v_n\}$ 是图的顶点集合，$E=\{e_0,e_1,\cdots,e_m\}$ 是图的有向边集合。

协议状态机 M 映射为有向图 DG 的过程如下：将状态集合 S 映射成有向图的顶点集合 V，使得任意状态均有唯一顶点与其对应。将状态的迁移映射为有向边，使得任意状态 s_i(对应顶点为 v_i)至 s_j(对应顶点为 v_j)的状态迁移(输入为 $i_k\in I$，输出为 $o_l\in O$)均有唯一有向边(以 v_i 为顶点，v_j 为终点)与其对应，并且该有向边以输入 $i_k\in I$ 和输出 $o_l\in O$ 进行标记，进而得到状态机 M 唯一对应的有向图 DG。

DCPP求解算法的适用条件是有向图为强连通图。强连通图是指有向图的每个顶点都可以到达其他任意顶点，然而协议状态机映射而成的有向图可能不是强连通图。例如，图1中FTP协议状态机映射而成的有向图中状态 s_2 所对应的顶点 v_2 无法到达 s_1 对应的顶点 v_1，该有向图不满足DCPP求解条件。为解决这一问题，采用了如下策略：在模糊测试过程中，一次会话结束后需要启动新的会话，因此将状态机 M 的终结状态返回初始状态作为一次状态迁移，将其称为虚拟迁移，将虚拟迁移添加到状态机中，添加虚拟迁移后的状态机用 $V\text{-}M$ 表示，如图2所示。在 $V\text{-}M$ 中，每个状态可以到达其他任意状态，$V\text{-}M$ 所对应的有向图为强联通图，其对应的DCPP可以求解。

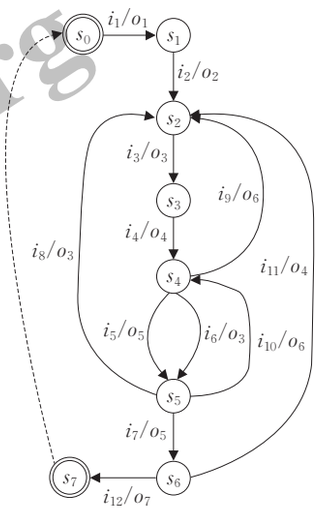

图2 添加虚拟迁移后的状态机 $V\text{-}M$

协议状态机映射为有向图 DG 后，将遍历所有状态迁移的最短路径的问题转化为DCCP问题进行求解。DCCP问题存在多项式时间解法。DCCP的经典求解算法是在有向图 DG 上增加有向边，使得每个顶点的入度和出度都相等，得到一个有向欧拉图 DG'，欧拉图 DG' 的欧拉回路就是遍历 DG 中所有迁移的最短路径。限于篇幅原因，算法细节不再赘述，详见文献[20]。在对 $V\text{-}M$ 遍历所有状态迁移的最短路径进行求解的过程中，以状态 s_0 对应的 v_0 结点出发，优先遍历每个顶点的自循环边。根据DCCP求解得到遍历有向图 DG 中所有有向边的最短路线，进而寻找到遍历协议状态机所有状态迁移的最短路径，称之为最优遍历路径(Optimal Traversal Path，OTP)。在模糊测试时，以遍历协议状态机对应的OTP为基础进行模糊测试，可保证对协议状态机中的所有状态迁移进行测试。

由图2对应的 $V\text{-}M$ 状态机求得最优遍历路径OTP，如图3所示。沿用协议状态机的概念，OTP中包含协议状态与状态迁移，其中OTP的第一个协议状态 s_0

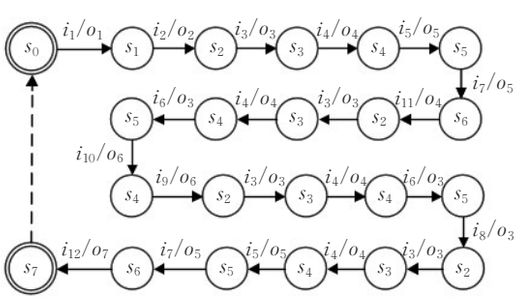

图3 最优遍历路径OTP

称为OTP的初始状态，最后一个协议状态 s_7 称为OTP的终止状态。从OTP的初始状态开始遍历至OTP的终止状态，期间经历了协议状态机 M 的所有状态迁移(包括虚拟迁移)。

3.5　利用交叉迭代策略构造UIO序列

该节通过设计交叉迭代策略解决3.3节问题(2)。交叉迭代策略基于这样一种思想：由3.3节可知，当协议实体处于状态 s_i，输入相应的测试用例之后，如果没有发现协议实体异常，将遵循OPT继续输入正常报文与测试用例执行测试。对于这些后续发送的报文，其中某些报文可组成报文序列 P'，如果报文序列 P' 是状态 s_i 的UIO序列，那么可利用 P' 检测状态 s_i 是否发生异常迁移，而无需再构造独立的UIO序列。

接下来介绍上述策略实施过程。以OTP的初始状态为起点，以OTP的终止状态作为终点，OTP上相邻的两个状态，以状态 s_i 与 s_j 为例，对状态之间的迁移进行如下判断：查看OPT中状态 s_i 与下一个 s_i 状态或者 M 的终止状态之间的迁移对应的报文序列，该报文序列记作 P。在实际测试过程中，在测试状态 s_i 与 s_j 之间迁移之后，如果未发现异常并继续测试，那么后续输入的报文序列 P' 是由报文序列 P 与测试用例构成。如果报文序列 P 是状态 s_i 对应的UIO序列，由定理可知，报文序列 s_i 也是UIO序列，此时可以保证在测试状态 s_i 与 s_j 之间的迁移后，接下来会输入相应的UIO序列用于检测协议实体是否发生异常迁移。为方便后续操作，此时需要对状态 s_i 与 s_j 之间迁移进行标记。如果报文序列 P 不是状态 s_i 对应的UIO序列，那么 P' 也不是UIO序列，状态 s_i 与 s_j 之间迁移测试后不会有UIO序列进行检测，那么不对状态 s_i 与 s_j 之间迁移进行标记。之所以查看OPT中状态 s_i 与下一个 s_i 状态或者 M 的终止状态之间的迁移对应的报文序列，是因为状态 s_i 在被测后，在协议实体处于下一个 s_i 状态或者会话结束前就要检测是否出现异常迁移。

定理　如果 $P=(i_x,i_y,\cdots,i_z)$ 是状态 s_i 的UIO序列，那么 $P'=((\sim i_x|-,i_x)\|\sim i_x,(\sim i_y|-,i_y)\|\sim i_y,\cdots,(\sim i_z|-,i_z)\|\sim i_z)$ 也是状态 s_i 的UIO序列，其中 $\sim i_x|-$ 表示要么输入 $\sim i_x$ 要么无输入，$(\sim i_x|-,i_x)$ 表示由 $\sim i_x|-$ 与 i_x 组成的输入序列。

证明　当协议实体处于状态 s_i 时，输入状态 s_i 的UIO序列 P，由UIO定义可知，协议实体必会有唯一输出 $(o_x,o_y,\cdots,o_z)$。而同样协议实体程序处于状态 s_i 时，输入 P'，那么协议实体必然输出 $((\overline{o_x}|-,o_x)\|o_x,(\overline{o_y}|-,o_y)\|o_y,\cdots,(\overline{o_z}|-,o_z)\|o_z)$，其中 $(\overline{o_x}|-,o_x)\|o_x$ 表示为 $\overline{o_x}|-$ 与 o_x 组成的输出序列。又因为输出 $(o_x,o_y,\cdots,o_z)$ 是唯一确定的，并且非期望响应报文 $\overline{o_i}$ 不会与其他任何期望响应报文相同，所以 $((\overline{o_x}|-,o_x)\|o_x,(\overline{o_y}|-,o_y)\|o_y,\cdots,(\overline{o_z}|-,o_z)\|o_z)$ 也是协议实体唯一确定的输出，则 P' 也是状态 s_i 的UIO序列。

上述策略实施过程也就是判断OPT上任意两个相邻的状态之间的迁移是否可以被标记，对整个OPT而言，详细的标记步骤如下：

步骤1　对于OTP中的某一个状态 s_i，以 s_j 表示其邻接状态。首先判断 s_i 是否为原始状态机 M 中的终止状态，如果是终止状态，那么状态 s_i 与状态 s_j 之间迁移是虚拟迁移，虚拟迁移不被标记，跳至执行步骤3；否则继续执行步骤2。

步骤2　从状态 s_i 出发顺序遍历，查看OTP中状态 s_i 与其之后的第一个 s_i 状态或者第一个原始状态机 M 终止状态之间的最短迁移路径。如果这段迁移路径对应的输入序列是 s_i 状态的UIO序列，那么对状态 s_i 至邻接状态 s_j 之间的迁移进行标记，然后执行步骤3；否则，该步骤不进行任何标记，直接执行步骤3。

步骤3　判断状态 s_j 是否为OTP的终止状态，如果状态 s_j 是OTP的终止状态，那么完成OTP的标记；否则，跳至执行步骤1，对状态 s_j 与其下一个邻接状态进行递归操作。

例如对图4最优遍历路径OTP进行标记，第一个 s_4 状态至其之后第一个 s_4 状态(即OTP的第二个 s_4 状态)之间迁移需要输入 i_5、i_7、i_{11}、i_3 与 i_4，依次经过状态 s_5、s_6、s_2 与 s_3，并且输出为 o_5、o_5、o_4、o_3、o_4，其中报文序列 (i_5,i_7,i_{11},i_3,i_4) 是状态 s_4 的UIO序列，因此对第一个 s_4 至与其下一个邻接状态 s_5 之间的迁移进行标记，然后再以同样的方法继续判断状态 s_5 以及其下一个邻接状态之间迁移是否应被标记，直至对所有状态都进行判断并标记相应迁移，标记后的状态迁移记作“|”，如图4所示。

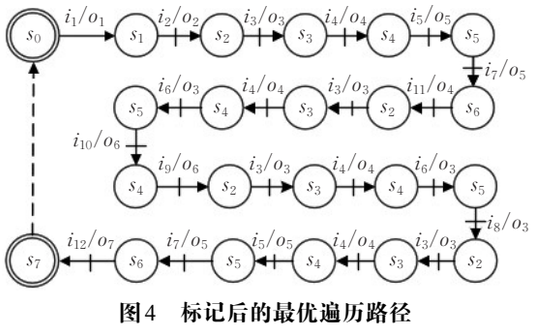

图4　标记后的最优遍历路径

4　模糊测试过程

基于前期分析与准备工作，本章将详细阐述动态模糊测试流程。遵循最优遍历路径OTP，以 s_i 和其邻接的下一个状态 s_j 之间的迁移为例，对应迁移的合法输入报文是 i_m，期望响应报文是 o_m，非期望响应报文记作 $\overline{o_m}$，报文 i_m 对应的测试用例的集合记作 $T_{i_m}=\{\sim i_{m1},\sim i_{m2},\cdots,\sim i_{mn}\}$，测试时，需要使用一个队列 Q 用以保存包含输入的报文信息。测试流程图如图5所示。

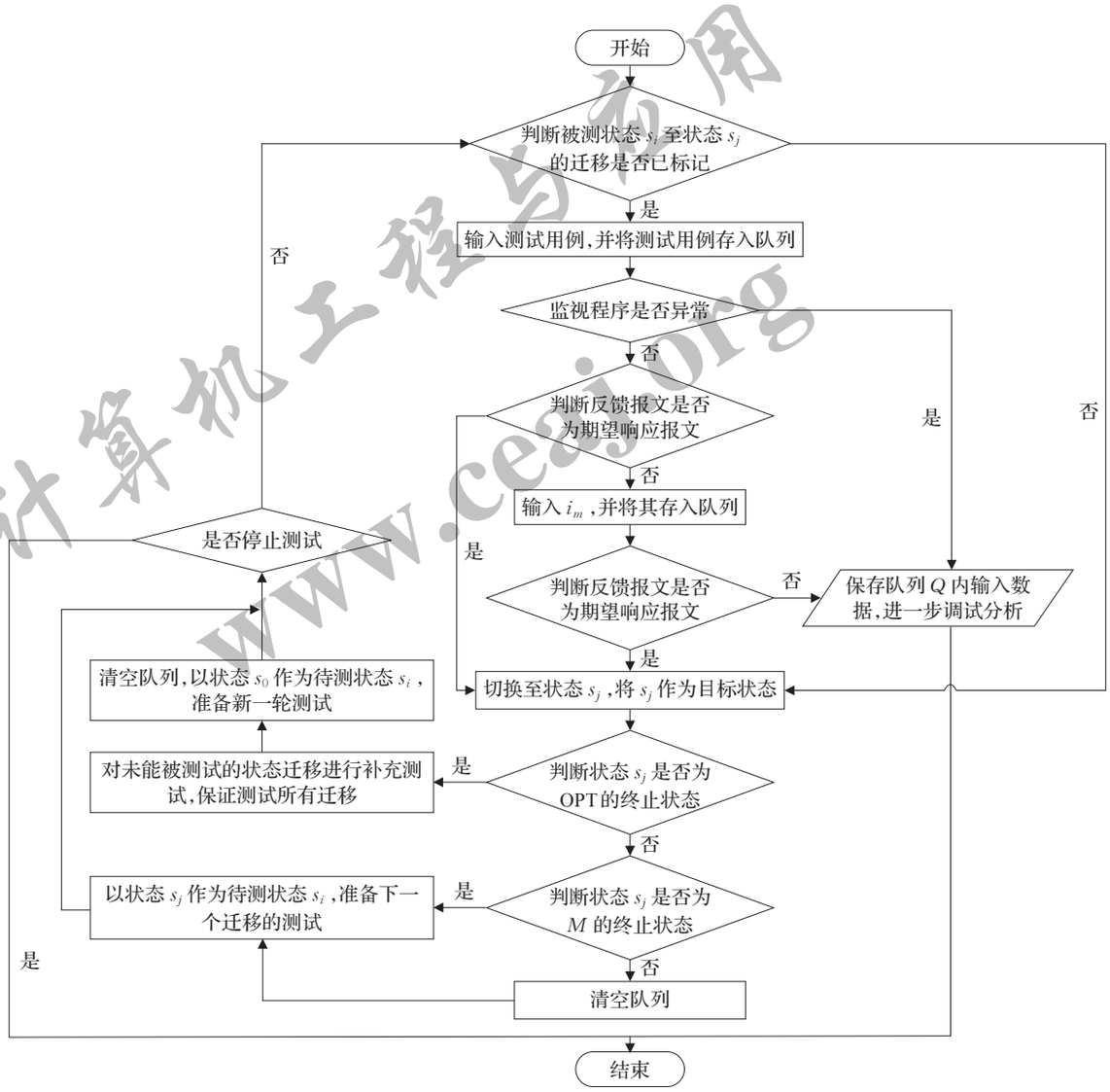

图5　模糊测试流程

动态模糊测试流程主要包括以下6个步骤：

步骤1　判断状态 s_i 至状态 s_j 之间迁移是否被标记过。如果该迁移被标记,那么满足后续输入报文可构成UIO序列的条件,执行步骤2进行测试;否则,执行步骤5。

步骤2　输入测试用例 $\sim i_{mj} \in T_{i_m}$,同时将测试用例信息记录在队列 Q 中,监测协议实体在处理测试用例后是否发生内存访问错误、程序崩溃等异常。如果出现异常,依据记录的测试用例信息进一步分析调试;否则,继续执行步骤3。

步骤3　判断协议实体在处理测试用例后的输出是否为 o_m。如果输出是 o_m,说明 $\sim i_{mj}$ 可以被协议实体接受并正常处理,此时协议实体的状态视为已迁移至 s_j 状态,跳转执行步骤5;否则,继续执行步骤4。

步骤4　继续输入 i_m,同时记录 i_m 信息到队列中,判断此时输出是否为 o_m。如果输出是 o_m,说明此时由于 i_m 的作用,协议实体的状态已经是 s_j,则执行步骤5;否则,对于正常的输入 i_m 没有得到期望响应报文 o_m,说明协议实体可能出现异常,可能是停止响应或者状态异常迁移,保存队列内输入报文信息,进一步分析调试。

步骤5　首先判断状态 s_j 是否为OTP的终止状态,如果是OTP的终止状态,那么执行步骤6,否则,判断状态 s_j 是否为状态机 M 的终止状态;如果状态 s_j 是终止状态,此时清空队列内所有已记录的报文信息,因为需要重新建立新的会话,旧会话阶段输入的报文对新会话的测试没有影响,然后返回步骤1重复该方法对状态 s_j 和其下一个状态之间的迁移进行操作;否则,直接返回步骤1对状态 s_j 和其下一个状态之间的迁移进行操作。

步骤6　由OTP迁移标记过程可知,可能某两个状态之间的迁移没有被标记,导致存在迁移未能被测试。此时,为了解决这种漏测问题,还需对未被测试的迁移进行补充测试,参考2.1节介绍的方法,输入前置引导序列将协议实体置于相应状态,再输入未测试过的状态迁移相应测试用例,检测协议实体是否异常。如果所有状态迁移都被测试,那么以OTP的始发状态作为状态 s_i,

返回至步骤1开始新一轮测试。

以图1的FTP协议状态机测试为例，对每个状态迁移都进行一次测试，需要向协议实体输入的报文依次是 $\sim i_1$、$i_1|-$、$\sim i_2$、$i_2|-$、$\sim i_3$、$i_3|-$、$\sim i_4$、$i_4|-$、$\sim i_5$、$i_5|-$、$\sim i_7$、$i_7|-$、$\sim i_{11}$、$i_{11}|-$、$\sim i_3$、$i_3|-$、$\sim i_4$、$i_4|-$、$\sim i_6$、$i_6|-$、$\sim i_{10}$、$i_{10}|-$、$\sim i_9$、$i_9|-$、$\sim i_3$、$i_3|-$、$\sim i_4$、$i_4|-$、$\sim i_8$、$i_6|-$、$\sim i_8$、$i_8|-$、$\sim i_3$、$i_3|-$、$\sim i_4$、$i_4|-$、$\sim i_5$、$i_5|-$、$\sim i_7$、$i_7|-$、$\sim i_{12}$、$i_{12}|-$。其中“|”表示选择关系，“-”表示不输入报文，如 $i_1|-$ 表示输入 i_1 或者不输入报文。输入该测试序列至多需要发送报文42次，最少需要交互21次，其中有21个测试用例进行测试，完成一次所有状态迁移的测试。

如果在进行测试时，协议实体出现异常，如协议实体出现内存访问错误，或者达到一定条件，如测试用例数目或者时间达到上限，则停止测试。协议实体异常可能是最后输入的测试用例所触发，也可能是前期测试用例引发协议实体状态异常迁移在此时才被检测出来。例如，当协议实体处于OPT上第一个 s_3 状态时，发送 i_4 后并未返回期望响应报文 o_4，可能是内存错误等问题导致协议实体停止服务，也可能是协议实体此时出现状态异常迁移，也可能是 s_3 状态（对应UIO序列是 i_4）测试后出现状态异常迁移，还可能是OPT上第一个 s_2 状态（对应UIO序列是 (i_3, i_4)）测试后出现状态异常迁移。因此，在发现协议实体出现异常时，需要对保存前期输入报文信息进行分析，验证捕获协议实体异常是否为误报，并进一步分析异常原因，判断是否存在可利用的漏洞。

总体而言，本章动态地调整后续输入的报文类型，避免无效的报文交互，利用交叉迭代思想减少报文交互次数，进而提高测试效率。在每次输入测试用例或者正常报文之后，对协议实体的响应报文进行分析，依据分析结果调整后续的输入是测试用例还是正常报文。如果后续输入是测试用例，那么又可以测试下一个迁移；如果后续输入是正常报文，那么充当前置引导序列的一部分，将协议实体引导至下一个状态继续测试，同时充当某些状态的UIO序列的一部分，达到检测协议实体是否发生状态异常迁移的目的。例如，基于FTP协议状态机进行测试时，正常报文 i_4 是状态 s_2 和 s_3 的UIO序列一部分，可用于检测协议实体是否发生状态异常迁移，i_4 又可作为状态 s_4、s_5 或者 s_6 的前置引导序列的一部分，i_4 的一次输入实际上发挥多个作用，减少了报文交互次数。

5　实验与结果分析

5.1　测试准备

Peach3.0是目前较为成熟的模糊测试框架，RFSM-Fuzzing是目前对有状态协议进行模糊测试的完整测试方案。本文方法PSTT-Fuzzing将与Peach3.0和RFSM-Fuzzing进行对比，三种方法采用的测试用例均以Peach 3.0变异策略生成，测试用例有效性相同。选取两种代表性的有状态协议作为测试对象——FTP与SMTP协议，其对应的协议实体程序信息如表3所示。

表3　测试对象的版本信息

协议类型	测试对象	版本
FTP	Quick’n Easy FTP Server	V4.0.0
SMTP	Quick’n Easy Mail Server	V3.3.0

5.2　测试结果分析

首先对实际环境下的测试效率进行评估，再以挖掘漏洞效率与挖掘能力统计结果进行对比分析。

5.2.1　测试效率评估

在测试过程中，即便是发送测试用例总数与发送报文的总数的比值很高，测试效率也未必高。因为如果测试用例未能与协议状态相对应，测试用例难以触发协议实体异常，仅仅考虑发送测试用例总数与发送报文总数的比值还不能准确评估测试效率。为了准确评估测试效率，本文引入有效测试报文发送效率概念。

假设某个测试用例 $\sim i_n$ 对应于状态 s_i，当协议实体处于被测状态 s_i 时，输入 $\sim i_n$ 进行测试，那么称该测试用例 $\sim i_n$ 为有效完成测试的测试用例。有效测试报文发送效率是指向被测协议实体输入有效完成测试的测试用例个数与发送报文的总数的比值，以 β 表示有效测试报文发送效率，$\sum Ac$ 表示有效完成测试的测试用例总数，$\sum m_j$ 表示发送报文的总数，β 的数学表达式为 $\beta=\sum Ac/\sum m_j$。β 值越大，说明发送辅助类型的报文越少，单位时间输入与状态相对应的测试用例数量比值较高，测试效率越高，采用 β 评估测试效率更具有合理性。

以Quick’n Easy FTP Server作为FTP协议的被测协议实体，以Quick’n Easy Mail Server作为SMTP协议的被测协议实体。PSTT-Fuzzing、Peach3.0、与RFSM-Fuzzing有效完成测试的测试用例总数（即 $\sum Ac$）与时间的关系如图6所示。

由图可知，在任意时刻PSTT-Fuzzing方法对应的有效完成测试用例数均要高于Peach3.0与RFSM-Fuzzing。Peach3.0尽管在测试时报文交互效率较高，单位时间输出的测试用例数量较多，但是Peach输入报文时盲目性较大，很多测试用例发送至协议实体时，协议实体状态实际上并未与测试用例相对应，$\sum Ac$ 值相对较低。RFSM-Fuzzing尽管采用完善的引导机制，使得多数测试用例可以在协议实体处于对应状态进行测试，但是测试期间前置引导序列等辅助报文的比重大，时间耗费

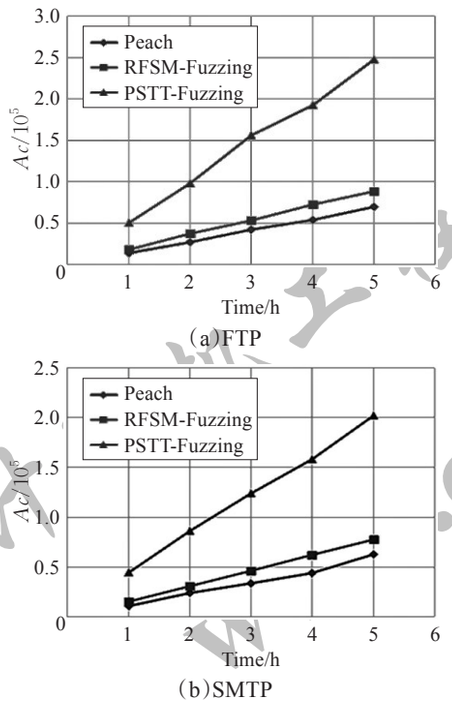

图6　有效完成测试用例数与测试时间的关系

大,在单位时间内的测试用例数目较少,$\sum Ac$ 值也相对较低。PSTT-Fuzzing在分析响应报文时存在时间耗费,但是该方法使用更少的交互引导报文,使得测试用例可在对应状态有效地实施测试,单位时间内有效完成测试用例比重更大,$\sum Ac$ 值更高。测试用例未能真正有效地实现测试。

需要说明的是,实际网络环境、服务器性能会影响报文传输的效率,因此有效完成测试用例数 $\sum Ac$ 值和时间的关系具有一定的随机性。为了更客观地评估测试效率,表4列出测试5小时有效测试报文发送效率 β 的统计结果。

表4　5小时报文交互有效率统计　%

测试对象	Peach	RFSM-Fuzzing	PFS
Quick'n Easy FTP Server	9.27	10.89	33.74
Quick'n Easy Mail Server	7.86	9.98	28.10

由表4可知,PSTT-Fuzzing对应的 β 高于Peach3.0与RFSM-Fuzzing对应的 β。当向协议实体发送报文的总数 $\sum m_j$ 相同时,相同的网络环境与服务器性能,发送时间耗费大致相同。由 β 的数学表达式可知,PSTT-Fuzzing对应有效完成测试的测试用例总数 $\sum Ac$ 更高。

5.2.2　漏洞挖掘效果

漏洞挖掘效率是指挖掘相同的漏洞所需时间耗费,时间耗费越少挖掘效率越高;漏洞挖掘能力是指在模糊测试正常执行条件下挖掘漏洞的数目。测试时间为5小时,漏洞挖掘效果统计结果如表5所示。

从表5可知,PSTT-Fuzzing漏洞挖掘能力和挖掘效率好于Peach与RFSM-Fuzzing。PSTT-Fuzzing挖掘出6个漏洞,Peach3.0挖掘出5个漏洞,RFSM-Fuzzing挖掘出4个漏洞,挖掘每个漏洞对应的时间耗费,PSTT-Fuzzing均要低于其他两种对比方法。

在挖掘能力方面,PSTT-Fuzzing效果好于Peach3.0与RFSM-Fuzzing。对于Quick Easy FTP Server V4.0.0服务器第二个未知漏洞,当变异的PASV报文发送至服务器时,服务器反馈命令错误的提示,然后输入正常PASV报文,在输入变异的LIST报文,再输入正常LIST报文时,服务器未返回任何响应报文,经检测此时服务器已经停止服务;Peach与RFSM-Fuzzing两种方法并未能挖掘该漏洞。对于Quick'n Easy Mail Server已知漏洞1,服务器短时间内接收到一定数量包含长字符串参数的多个HELO命令的报文时,程序本身没有错误提示,调试工具也未能捕获到运行异常,但是服务器未返回期望响应报文,经检测发现服务器已近停止服务,Peach与RFSM-Fuzzing测试方法使用调试器检测异常未能挖掘到该漏洞。在挖掘效率方面,从实验统计结果对比可知,PSTT-Fuzzing的测试效率高于Peach与RFSM-Fuzzing方法,验证了5.2.1小节测试性能评估的正确性。对测试性能与挖掘漏洞效果进行比较,相比于Peach与RFSM-Fuzzing测试方法,本文测试方法PSTT-Fuzzing具有明显的优势,本文模糊测试优化方法达到了预期的测试效果。

6　结束语

本文针对现有的有状态协议的模糊测试技术测试时存在辅助类型报文交互时间比重大,测试效率低下,测试覆盖率低等问题,提出一种基于协议状态迁移遍历的模糊测试优化方法。实验结果表明,在相同测试时间

表5　5小时漏洞挖掘效果统计结果　min

漏洞编号	测试对象	Peach	RFSM-Fuzzing	PSTT-Fuzzing
1	Quick Easy FTP Server V4.0.0	135	95	25
2	Quick Easy FTP Server V4.0.0	—	—	45
3	PCMan FTP Server 2.0.7	182	268	76
4	Quick'n Easy Mail Server 3.3	39	45	18
5	Quick'n Easy Mail Server 3.3	271	—	95
6	MailEnable 3.13 SMTP	248	295	71

情形下，本文方法可以针对性地发送更多测试用例，更易触发协议程序异常，测试效率更高。下一步研究将提高测试用例的有效性，结合本文优化方法，更好地提高模糊测试效率。

参考文献：

[1] 李舟军，张俊贤，廖湘科，等.软件安全漏洞检测技术[J].计算机学报，2015，38(4)：717-732.

[2] 张宝峰，张翀斌，许源.基于模糊测试的网络协议漏洞挖掘[J].清华大学学报(自然科学版)，2009，49(S2)：2113-2118.

[3] Munea T L，Lim H，Shon T.Network protocol fuzz testing for information systems and applications：a survey and taxonomy[J].Multimedia Tools & Applications，2016，75(22)：14745-14757.

[4] Oulu University Secure Programming Group.The PROTOS project[EB/OL].[2018-08-01].https://www.ee.oulu.fi/research/ouspg/Protos.

[5] Aitel D.An introduction to SPIKE[EB/OL].[2018-08-11].http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-aitel-spike.ppt.

[6] Eddington M.PeachFuzzer[EB/OL].[2018-08-12].http://peachfuzzer.com/.

[7] Kitagawa T，Hanaoka M，Kono K.AspFuzz：a state-aware protocol fuzzer based on application-layer protocols[C]//IEEE Symposium on Computers and Communications，2010：202-208.

[8] Zhao J，Chen S，Liang S，et al.RFSM-Fuzzing a smart fuzzing algorithm based on regression FSM[C]//8th International Conference on P2P，Parallel，Grid，Cloud and Internet Computing，2013：380-386.

[9] Pan F，Hou Y，Hong Z，et al.Efficient model-based fuzzy testing using higher-order attribute grammars[J].Journal of Software，2013，8(3)：645-651.

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

[11] Narayan J，Shukla S K，Clancy T C.A survey of automatic protocol reverse engineering tools[J].ACM Computing Surveys，2015，48(3)：1-26.

[12] Gascon H，Wressnegger C，Yamaguchi F，et al.PULSAR：stateful black-box fuzzing of proprietary network protocols[M]//Security and privacy in communication networks.Berlin：Springer International Publishing，2015：330-347.

[13] Cui B，Liang S，Chen S，et al.A novel fuzzing method for Zigbee based on finite state machine[J].International Journal of Distributed Sensor Networks，2014(3)：1-12.

[14] 康红凯，吴礼发，洪征，等.一种基于FSM的BGP-4协议模糊测试方法[J].计算机工程与应用，2017，53(6)：111-117.

[15] Sutton M，Greene A，Amini P.模糊测试：强制性安全漏洞发掘[M].北京：机械工业出版社，2009.

[16] 苏璞睿，应凌云，杨轶.软件安全分析与应用[M].北京：清华大学出版社，2017.

[17] Wang J，Chen B，Wei L，et al.Skyfire：data-driven seed generation for fuzzing[C]//2017 IEEE Symposium on Security and Privacy，2017：579-594.

[18] Provost J，Roussel J M，Faure J M.Generation of single input change test sequences for conformance test of programmable logic controllers[J].IEEE Transactions on Industrial Informatics，2014，10(3)：1696-1704.

[19] Sabnani K，Dahbura A.A protocol test generation procedure[J].Computer Networks & ISDN Systems，1988，15(4)：285-297.

[20] 高敬振，高勃.中国邮递员问题50年[J].运筹学学报，2013，17(1)：17-28.