

渐进式的协议状态机主动推断方法

潘雁, 林伟, 祝跃飞

(信息工程大学, 河南 郑州 450001)

摘要: 主动协议状态机推断的理论基础为主动自动机学习, 所面临的核心问题是字母表的抽象与映射器的构建。同一类型消息取值的多样性可能导致同一类型的数据包存在不同的响应类型, 从而导致当前使用类型作为字母表的方法会丢失状态或状态转移。对此, 依据不同的响应将协议类型细化为子类型, 提出一种渐进式主动推断方法。基于已有协议数据提取协议状态字段, 构建初始字母表与映射器, 基于主动推断方法得到初始状态机; 对数据进行确定性变异, 若输入输出类型序列与当前状态机不符, 则将变异后数据视为协议子类型, 并添加至字母表, 同时依据新的字母表进行新的状态机推断。此外, 为减少协议实际交互次数, 依据协议特性, 在主动推断算法的缓存机制基础上提出一种基于前缀匹配的预响应查询算法。实现了开源框架 ProLearner, 并以 SMTP 和 RTSP 为对象, 通过扩展协议子类型获得了更为详细的协议行为, 验证了所提方法的有效性; 此外, 实验结果表明预响应查询算法可有效减少实际交互的次数, 平均降低的实际交互次数约为 10%。

关键词: 协议逆向分析; 主动自动机学习; 协议状态机推断; Mealy 自动机; 映射器

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023023

Progressive active inference method of protocol state machine

PAN Yan, LIN Wei, ZHU Yuefei

Information Engineering University, Zhengzhou 450001, China

Abstract: Protocol state machine active inference is a technique that relies on active automata learning. However, the abstraction of the alphabet and the construction of the mapper present critical challenges. Due to the diversity of messages of the same type, the response types of the same type are different, causing the method of regarding the message types as the alphabet will result in the loss of states or state transitions. To address the issue, message types were refined into subtypes according to the different responses and a progressive active inference method was proposed. The proposed method extracted the state fields from the existing protocol data to construct the initial alphabet and the mapper, and obtained the initial state machine based on active automata learning. It then mutated the existing messages to explore the response sequences, which were inconsistent with the current state machine. The mutated message was regarded as a protocol subtype and added to the alphabet, and a new state machine was

收稿日期: 2022-10-27; 修回日期: 2023-02-05

通信作者: 潘雁, z11pany@qq.com

基金项目: 国家重点研发计划 (2019QY1300)

Foundation Item: The National Key R&D Program of China (2019QY1300)

引用格式: 潘雁, 林伟, 祝跃飞. 渐进式的协议状态机主动推断方法[J]. 网络与信息安全学报, 2023, 9(2): 81-93.

Citation Format: PAN Y, LIN W, ZHU Y F. Progressive active inference method of protocol state machine[J]. Chinese Journal of Network and Information Security, 2023, 9(2): 81-93.

inferred progressively based on the new alphabet. In order to reduce the interactions, a pre-response query algorithm was proposed based on prefix matching for the caching mechanism in the active automata learning. The ProLearner tool was utilized to evaluate the proposed method in the context of the SMTP and RSTP protocols. It is verified that the pre-response query method can effectively reduce the number of actual interactions, with an average reduction rate of about 10%.

Keywords: protocol reverse analysis, active automata learning, protocol state machine inference, Mealy automata, mapper

0 引言

网络协议规范是实现计算机网络中互相通信的对等实体之间交换信息时所必须遵守的规则集合,其定义了通信数据的格式以及协议实现处理数据包时的状态切换。协议格式规范分析,又被称为协议逆向工程,它通过对未知协议的通信数据或处理数据的协议实体进行分析,以获取协议的格式与行为。其通常分为3个层次:语法、语义及时序。语法指报文的格式及编码,语义指报文的具体含义(包括控制信息、错误处理等),时序指报文之间的通信序列。协议状态机推断对应时序分析,通常是在报文格式、语义信息已知的前提下,根据协议实体接受和拒绝的报文序列,推断其处理报文序列过程中的状态迁移过程,通常采用有限状态机、概率状态机^[1]、前缀树接受器^[2]、马尔可夫模型^[3]等描述状态转移。

协议状态机推断根据学习的方式可分为被动推断与主动推断。

被动推断是基于离线数据包提取代表报文的 状态信息,并分析生成协议对应的处理逻辑;主动推断是将格式良好的数据包发送到服务器,基于查询-响应的方式构建状态机,能在一定程度上克服被动学习仅具有正样本的缺点。

主动推断的理论基础是主动自动机学习,但其广泛使用的 MAT (minimally adequate teach) 框架^[4]仅适用于输入字母表较小的系统,而现实世界的系统和软件的输入空间是巨大的,如协议实现的输入输出为协议消息。现有的方法通常是对协议消息进行分类,依据协议的已知格式将消息划分为不同的类型,即消息集合可以划分为多个互不相交的子集合,并将类型作为输入字母。

在现有的文献中,协议逆向工程和协议状态机主动推断对消息类型均没有明确的定义,对协议进行基于类型的划分所依赖的假设为目标协议

具有状态相关字段,即具有特定的字段标识协议类型;同一类型的协议消息具有相近的格式,且与其他类型的消息相似性较弱^[5]。因此可基于关键词分析和相似性的方式对协议消息进行分类。例如,实时流协议(RTSP, real-time streaming protocol)的关键词为 DESCRIBE、SETUP、PLAY、TEARDOWN 分别表示不同的协议类型。

在实际情况下,同一类型消息取值的多样性(如不同参数的选择)可能导致同一类型的数据包存在不同的响应类型。以 RTSP 为例,SETUP 类型对应的消息所包含的参数不同会导致响应类型不同。因此,直接将类型作为字母表会导致状态或者状态转移丢失。

针对上述问题,Cassel 等^[6]提出了寄存器自动机,它是一种能够表达数据对控制流影响的自动机模型,每种状态有一个寄存器用以存储数值,可对下一次输入进行相等性比较,从而直接推断出一部分数据值对控制流的影响,其表达能力相较于 Mealy 状态机有一定的提升,但参数的选定和模型的复杂程度都带来了更大的分析难度,且只能针对数值型的参数进行判断。

Székely 等^[7]基于已有数据包提取子类型,子类型是指在每种状态下使服务器产生的响应类型相同的消息集合。文献[7]基于自动协议逆向工程对已有数据包进行分析并获得候选子类型,在初始模型的基础上依次添加候选子类型判断是否为子类型,若是则将其添加至模型中,否则进行类型的合并。

被动学习同样面临该问题,Sun 等^[8]提出一种基于概率的 Mealy 机描述协议行为,若同一种类型导致了不同类型的响应,则统计已有数据中不同类型的比例,通过概率描述不同的转移过程。

综上,当前方法中的寄存器自动机仅针对数值型的参数进行判断,且参数的选定和模型的复杂程度都带来更大的分析难度;Székely 提出的方法中缺乏对已有数据包的详细分析,且缺少异常数

据的扩展。针对此问题，本文首先借鉴被动学习的预处理过程，将数据包转化为类型序列；而后基于类型相关字段对已有数据包进行分类提取；最后，细化并改进文献[7]的具体算法，逐步学习状态机。同时，由于正常通信采集的数据包消息格式都是标准格式，缺少异常数据，因此在主动学习的过程中添加变异模块，探索更多的异常子类型。

本文的主要贡献如下：

1) 针对协议状态机推断存在的字母类型与消息一对多的对应关系，基于已有的数据包进行变异，提出了一种主被动结合的渐进式协议状态机推断方法；

2) 基于协议特点，改进协议主动推断中的缓存机制，提出一种基于前缀的预响应查询算法；

3) 以 RTSP、简单邮件传送协议(SMTP, simple mail transfer protocol) 为对象，成功实践了所提方法的技术思路，验证了该方法的有效性。

1 相关工作

本文所涉及的相关工作主要为协议状态机的被动推断和主动推断。

1.1 被动推断

被动推断是基于离线数据包提取代表报文的状态信息，并分析生成协议对应的处理逻辑。Leita 等^[9]最早使用 Moore 状态机描述协议行为，同一类输入可能造成不同的输出，构成了状态的标签数组。Comparetti 等^[10]先将数据转化为带标签的增强型前缀树接受器，而后将其转化为 Moore 状态机。Wang 等^[11]基于已有数据统计状态类型之间的次序关系和转移概率值，并使用概率状态机描述上述统计数据。Krueger 等^[11]将协议行为建模为隐马尔可夫模型，隐状态为协议内部逻辑状态，输出为协议消息，但协议行为本质上并不满足输出独立性假设。上述基于 Moore 状态机和隐马尔可夫模型描述方法的前提是输出仅与当前状态有关，丢失了输入输出的关联性。Lee 等^[12]考虑了输入输出的关联性，采用 Mealy 状态机描述行为，提升了模型的表达能力。Lin 等^[13]生成状态机时引入数据流信息，考虑同一类型的协议消息中的参数差异形成的不同状态转移，类似于寄存器状态机，但其必须提前选定关注的参数，同时面临

着状态爆炸的问题。Sun 等^[8]则使用带有概率的 Mealy 状态机对同一类型下的差异进行描述。被动推断的缺点在于数据包中仅包含正样本，而仅基于已有数据构建最小化状态机是一个 NP 完全问题^[14]，因此基于被动学习的模型是不够完整的。

1.2 主动推断

主动推断的理论基础是主动自动机学习，其广泛使用的框架是 Angluin 于 1987 年提出的 MAT 框架^[4]，框架包含一个学习器和一个预言机，学习器通过向预言机提出查询来学习未知的模型，预言机了解待测系统(SUL, system under learning)的所有内容，而学习器只知道 SUL 的输入/输出字母表。为了学习未知自动机，学习器提出成员查询，即发送一个报文序列至 SUL，若 SUL 接受则返回“是”，否则返回“否”；而后基于一系列查询及其响应尝试构造一个行为与目标自动机相匹配的未知自动机，并提交至预言机，由预言机判断自动机是否正确并给出反例，称这一过程为等价查询。

目标状态机以 Mealy 状态机为主，其可以定义为在输入输出字母表 I, O 上的四元组 $\mathcal{A} = (S, s_0, \delta, \lambda)$ ，其中 $I = \{i_1, i_2, \dots, i_n\}$ 为有限的输入字母表， $O = \{o_1, o_2, \dots, o_k\}$ 为有限的输出字母表， S 是状态集合， $s_0 \in S$ 是初始状态， $\delta: S \times I \rightarrow S$ 是状态转移函数， $\lambda: S \times I \rightarrow O$ 是状态输出函数，可写为 $\delta(s_0, i_1) = s_1$ ， $\lambda(s_0, i_1) = o_1$ 。此外，由字母组成的有序序列，称为串，记为 w ，串构成的集合记为 I^* ，串的连接记为 $w_1 \cdot w_2$ ，特别地， ε 为空串。状态转移函数和状态输出函数可针对串扩充为 $\delta: S \times I^* \rightarrow S$ ， $\lambda: S \times I^* \rightarrow O^*$ 。

当目标系统为协议实现时，首先可定义消息的分类函数 $\text{class}: \mathbf{M} \rightarrow T$ ，其中 $T = \{t_1, t_2, \dots, t_l\}$ 为协议类型的集合。由该分类函数构建的等价关系为 $\sim_{\text{class}} \in \mathbf{M} \times \mathbf{M}$ ，如果 $\text{class}(m_1) = \text{class}(m_2)$ ，则有 $m_1 \sim_{\text{class}} m_2$ ，由此等价关系形成的等价类 $[m]_{\text{class}} = \{m_i \in \mathbf{M} \mid m_i \sim_{\text{class}} m\}$ 将协议消息划分为多个消息类型，若 $\text{class}(m) = t_i$ ，将 $[m]_{\text{class}}$ 记为 M_i 。协议实现在初始状态的输出函数可记为 $\lambda_A: \mathbf{M}^* \rightarrow \mathbf{M}^*$ ，由于本文只关注协议实现的响应类型，因此本文中将协议实现的输出函数简化为 $\lambda_A: \mathbf{M}^* \rightarrow O^*$ 。

主动自动机推断需要将上述输出函数进一步抽象为 $\lambda_A: I^* \rightarrow O^*$, 因此通常将协议类型作为协议输入字母表, 在构建具象化模块时生成的数据包与输入字母是一一对应的关系。Cho 等^[15]以僵尸网络的命令控制协议为对象进行研究, 通过手动逆向分析命令控制协议的结构, 将请求数据包分为多个类型, 将获得的类型作为输入字母表, 而在具体的学习阶段, 需要将字母表具象化为具体的数据包, 并将响应数据抽象为输出字母, 抽象和具象的实现极大地降低了学习的代价。Aarts^[16]形式化地定义了抽象化和具象化, 将其称为映射器, 其位于学习器和 SUL 之间, 实现字母与实际数据的转化, 现有的映射器都是基于协议技术规范或已有的协议数据包, 通过协议逆向工程分析获得协议的类型与具体数据的对应关系, 手动构建抽象和具象函数, 且考虑的类型较为粗略。

综上所述, 将主动学习算法应用于实际系统, 实现流程如图 1 所示。首先需要对输入空间进行抽象化, 将输入空间划分为特定的类型, 形成字母表; 而后在 MAT 框架中, 需要构建映射器, 映射器包括对输入字母表的具象化模块和对输出字母表的抽象化模块。在实际实现中, 映射器与目标系统之间具有缓存机制, 对所有的查询进行缓存。在当前的研究中^[17-19], 字母表均是提前确定的, 且是手动分析的结果, 通常以消息类型表示字母表, 且具象化模块是确定的, 即该类型的数据包输入目标系统所得到的响应类型是相同的。

在此框架基础上, 研究人员针对不同协议展开研究。Ruiter 等^[17]基于 LearnLib 开源框架^[20]构建了传输层安全协议 (TLS, transport layer security) 的映射器, 推断了多个 TLS 开源实现, 并且通过手动对比状态机和协议规范来分析逻辑漏洞。申莹珠等^[18]基于 LearnLib 开源框架对 TLS 模式下的 OpenVPN 协议进行了状态机推断, 并提出了一种基于时间压缩模型的状态机化简方法, 以便进行手动的对比分析。Fiterau-Brosteau 等^[19]针对数据包传输层安全协议 (DTLS, datagram transport layer security) 的特性实现了主动状态机学习, 并发现了相应的漏洞; 同时针对传输层控制协议 (TCP, transmission control protocol) 实现开展了相关的研究^[21], 并将协议技

术规范和协议状态机转化为 NuSMV (new symbolic model verifier) 语法, 实现自动化的模型检查, 后续基于寄存器自动机对 TCP 的滑动窗口特性进行了学习^[22]。Guo 等^[23]构建了 IPSec 的输入输出字母表, 分析发现了 IPSec 系统实现中的一些漏洞。王辰等^[24]融合被动推断和主动学习, 充分利用已有数据, 构建扩展前缀树接收器实现查询的预响应, 并提出了一种基于变异的等价查询算法以便更有效地发现反例。

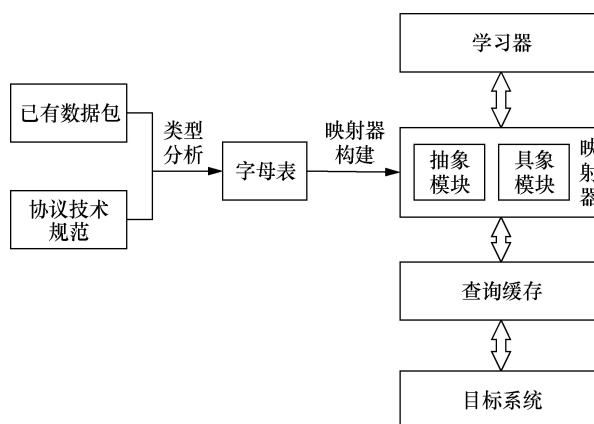


图 1 协议状态机主动推断实现流程
Figure 1 The flow chart of protocol state machine active inference

综上所述, 由于被动的状态机推断方法中数据包仅包含正样本, 因此其模型的泛化是不够完整的, 存在一定的局限性, 而模型学习技术的发展促进了协议状态机主动学习的应用, 如何融合被动和主动学习是值得研究的。同时, 模型学习技术应用于实际系统所面临的问题包括: 实际系统具有较大的输入输出字母表, 应当探讨更有效地构造映射器的方法; 实际系统学习过程中时间占比较高的是交互时间, 降低实际交互的次数尤为重要。

除此之外, 主动自动机学习理论的研究主要关注学习算法^[25]和等价查询算法的改进^[26], 目标模型类型的研究等^[27]。

2 本文方法

针对模型学习应用于协议状态机推断中存在的问题, 本文提出了一种主被动结合的渐进式协议状态机推断方法。

由于需要将较大的消息空间依据其响应类型分类为不同的子类型, 因此, 首先定义消息的子类型 (协议类型) 的细分。由于构成数据包的参

数各不相同，同一个类型对应的数据产生的响应类型并不一定相同。

定义 1 给定 λ_A 且 $\forall m, m' \in M_i, \forall \text{seq} \in M^*$ ，若满足 $\lambda_A(\text{seq} \cdot m) = \lambda_A(\text{seq} \cdot m')$ ，则 m, m' 属于同一子类型，即同一子类型的消息在任意消息序列发送后所得的响应类型相同。

通过遍历 M^* 对子类型进行判断是不现实的，因此在已有数据包的前提下提出一种渐进式学习算法，渐进式主动协议状态机推断流程如图 2 所示，图中黄色阴影部分为本文的改进点。

第 1 步，基于协议逆向工程方法或 RFC 规范对已有协议数据包进行类型分析，得到消息类型与协议消息的对应关系，该步充分利用已有数据包的知识，对行为相近的数据包进行分类去重，降低整个算法的输入数量。

第 2 步，以消息类型为字母表，构建对应的映射器，基于主动自动机学习推断协议实现的状态机，并基于协议特点，提出一种基于前缀的预

响应查询算法。

第 3 步，将已有的协议数据包都视为候选子类型，以流为单位将流转化为子类型序列，将其与初始状态机进行比对，若输出序列是一致的，则认为该候选子类型是重复的，否则认为该候选子类型是子类型，需将该流所包含的所有候选子类型添加至字母表中，此处简化了文献[7]中流序列与状态机相符的判断算法；而后基于扩充后的字母表执行第 2 步中的学习算法，获得新的状态机后，判断添加的候选子类型是否导致了新的状态或新的状态转移，若未导致新状态或转移，则将该候选子类型从状态机中去除。

第 4 步，遍历第 1 步所获得的数据包，循环第 3 步。同时，遍历已有数据包后，为扩充异常子类型，将第 3 步更改为对已有数据包进行确定性变异，若出现了新的输出类型或新的输出序列，则同样将所有的候选子类型序列添加至字母表中进行学习，变异一定的次数后循环终止，从而获得最终状态机。

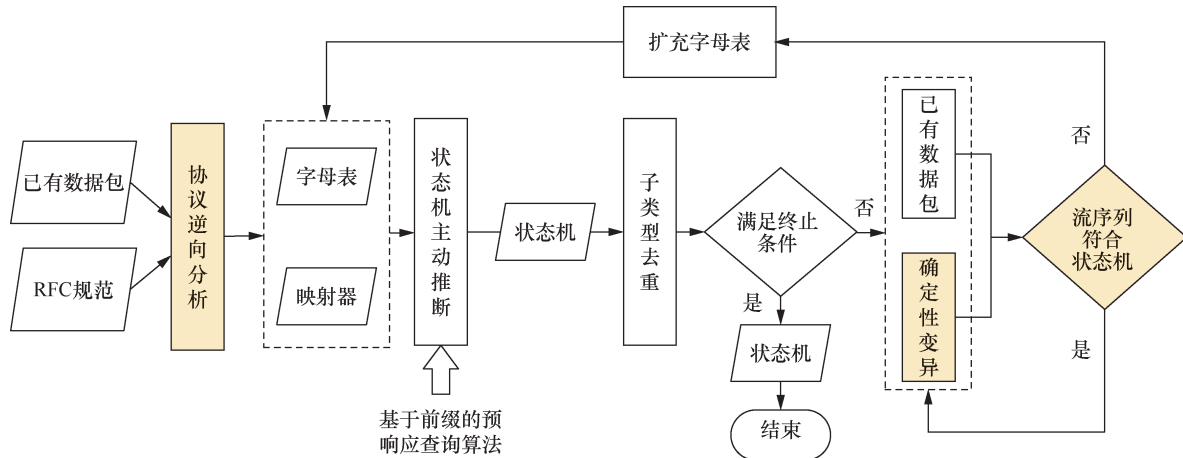


图 2 渐进式主动协议状态机推断流程
Figure 2 The flow chart of the progressive learning method on protocol state machine active inference

2.1 被动学习

被动学习的前提条件为协议具有特定的字段标识协议类型，同一类型的协议消息具有相近的格式，且与其他类型的消息相似性较弱。在此条件下，可基于现有的协议逆向实现协议消息的类型划分。在此基础上，研究者可从协议类型的角度生成协议状态机，如 Wang 等^[1]定义并使用概率状态机描述协议行为，但其输入输出是分离的；Sun 等^[8]融合分析了输入输出，使用一个概率状态机描述协议行为，但概率只在大量的重复数据

下具有统计学意义。

本文方法基于被动学习的思想将数据流进行分类，可认为是预处理过程。先将每条消息转化为输入输出类型序列，而在实际中，可能序列中存在输入为空或者输出为空的情况，如输入输出序列 $\{i_1 \cdot i_2 \cdot i_3, o_1 \cdot o_2 \cdot o_3\}$ 可能对应的为 $\{(\varepsilon, o_1), (i_1, o_2), (i_2, \varepsilon), (i_3, o_3)\}$ ，其中 ε 代表空输入或空输出。针对该问题，借鉴文献[8]中的标签修正算法实现转化过程，将每条消息转化为一一对应的输入输出类型序列，若输入为空或输出为空，

则用空串 ε 表示。

在此基础上,对类型序列集合进行分类去重。分类去重的规则如下。若序列 $w=w' \cdot v$, $w',v \in I^*$, 且其对应的输出序列类型满足 $\lambda_A(w)=\lambda_A(w' \cdot v)$, 则 w 与 w' 属于同一类序列,且最终类型序列集合中只保留序列 w 。仍以 RTSP 为例,为便于表达使用类型的首字母表示类型,若存在两个数据包的类型序列分别为 $\langle D,S,P,Q,T;200,200,200,200,200 \rangle$ 与 $\langle D,S,P,Q;200,200,200,200 \rangle$, 则只保留第一个序列。而若两个序列为 $\langle D,S,P,Q,T;200,200,200,200,200 \rangle$ 与 $\langle D,S,P,T;200,200,200,200 \rangle$, 则无法认定二者为同一类。基于该分类去重规则进行预处理能有效降低第3步中需循环处理的数据包的数量。

2.2 映射器自动化构建

当前主动学习算法中映射器在学习的过程中不会进行修改,而算法在学习过程中需要依据协议消息更新字母表,因此映射器需要自动化地构建。

由于输出关注的仍然是协议类型,因此抽象化模块无须修改,具象化模块可简单地视为提取类型的逆函数。由于在网络领域数据包的重放是有意义的,因此在实际实现过程中依据协议消息进行子类型的具象化同样是合理的。

值得注意的是,协议中通常会用一些与会话相关的字段来标识会话,在重放的过程中如果不调整这些字段,而是直接重放已有的协议消息,则会导致待测系统无法到达期望的状态,从而使得后续会话失败。Fang 等^[28]总结相关字段如下。

1) 序号:代表该消息在会话中的位置,通常是递增的。

2) 会话 ID:会话的唯一标识,一旦生成该 ID 值,后续交互中的消息将携带此值。

3) 时间戳:部分协议消息中会对时间戳进行验证。

4) 挑战-响应:用户将发送带有随机数的挑战,待测系统将根据挑战生成响应,如果响应不正确,会话将终止。

在具象化模块中添加字段更新模块,实现了序号和会话 ID 的更新,此更新模块需基于协议的分析进行构建,如 SMTP 中无相关字段,RTSP

中需要更新序号和会话 ID 字段。

2.3 渐进式主动学习

若直接将所有数据包转化为子类型,并将所有子类型作为字母表,会导致字母表非常庞大,而实际上很多子类型并未造成新的状态转移或者新的状态。因此,本文采用渐进式主动学习的策略,如图2所示,依次将每条流转化为现有的类型序列,将输入序列输入当前的状态机,判断状态机的输出序列与流的输出序列是否一致,若不一致则将该输入序列中的每一个类型都视为子类型,加入字母表中并扩充映射器,同时更新当前状态机。以 RTSP 为例,渐进学习过程中的字母表和映射器如表1所示,经过自动机学习获得的协议状态机,即第一阶段 RTSP 状态机如图3所示。对已有数据包中的流进行类型提取,提取的输入输出序列分别为 D-S-S-P-T 与 200-200-200-200-200,而状态机对应的输出序列为 200-200-Closed,两者不一致,因此需要将该流的类型序列视为新增的子类型序列,即表1中第二阶段所示,扩充字母表和映射器,更新状态机,并基于当前状态机依据定义1去除重复的子类型。

表1 渐进学习过程中的字母表和映射器
Table 1 The alphabet and mapper of the progressive learning process

阶段	字母表	协议消息
第一阶段	Describe01	DESCRIBE XX aacAudioTest
	Setup01	SETUP XX aacAudioTest/track1
	Play01	PLAY XX aacAudioTest
	Teardown01	TEARDOWN XX aacAudioTest
第二阶段	Describe02	DESCRIBE XX matroskaFileTest
	Setup02	SETUP XX matroskaFileTest/track1
	Setup02	SETUP XX matroskaFileTest/track2 session
	Play02	PLAY XX matroskaFileTest
	Teardown02	TEARDOWN XX matroskaFileTest

一条流中的子类型序列与当前状态机不一致,其中的候选子类型可能与已有的子类型重复。依据子类型的定义,需在当前状态机的基础上进行判断。去重算法如算法1所示,算法输入为当前状态机 $\mathcal{M}$,当前流 $flow$,上一阶段的字母表 I_{prev} ,需对该流的所有子类型进行遍历判断,对上一阶段字母表中与该子类型同属一个类型的所

有子类型进行一致性判断，即在每个状态下两个子类型的响应类型是否一致，若完全一致，则该子类型是重复的，从字母表中去除。

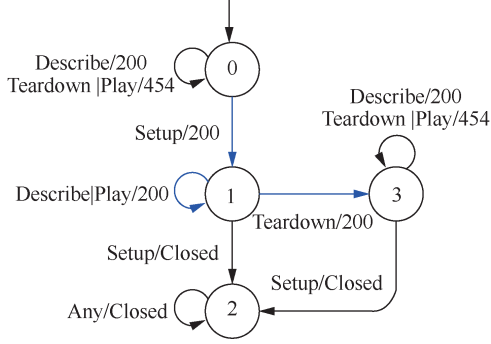


图 3 第一阶段 RTSP 状态机
Figure 3 The state machine of RTSP after the first stage

算法 1 去重算法

输入 $\mathcal{M}, \text{flow}, I_{\text{prev}}$

输出 $\mathcal{M}$

- 1) $\text{AccessSet} = \text{GetAccess}(\mathcal{M})$
- 2) $\text{typeseq}_i, \text{typeseq}_o = \text{GetType}(\text{flow})$
- 3) **foreach** subtype **in** typeseq_i **do**
 $\text{flag} = \text{true}$
- 4) **foreach** a **in** I_{prev} and $a \sim_{\text{class}}$ subtype **do**
- 5) **foreach** access **in** AccessSet **do**
- 6) **if** $\lambda(\text{access}, \text{subtype}) \neq \lambda(\text{access}, a)$
- 7) $\text{flag} = \text{false}$;
- 8) **break**;
- 9) **end if**
- 10) **end for**
- 11) **end for**
- 12) **end for**

2.4 确定性变异

如 2.1 节所述，由于已有数据流中大多是正常访问数据，对于异常行为的覆盖极少，因此本文提出基于确定性变异策略对现有种子进行变异，观察其类型的输入输出序列与当前状态机的一致性，若不一致，则将变异生成的种子作为新的子类型序列进行处理，处理方式与算法 2 中第 10)~16) 行的方式一致。

本文的变异策略主要是将 AFL 中的确定性变异策略应用到消息的变异，依次选取消息序列中的消息进行变异，变异策略主要有字节翻

转、整数叠加、字典替换等。设种子为 $\text{seq} = m_1 \cdot m_2 \cdot m_3$ ，依次对第 1 至 3 个消息进行变异，如第 1 个消息变异后的序列可记为 $\text{seq}' = \text{mutate}(m_1) \cdot m_2 \cdot m_3$ ，并将该序列输入至 SUL，得到对应的输出序列，对输出序列的判断标准与已有数据流的判断相同。由于本文采用的策略是每次仅对一个消息进行变异，因此在该策略下每次子类型的添加是确定的，即需进行子类型的去重。

算法 2 推断算法

输入 pcap

输出 $\mathcal{M}$

- 1) $\text{TypeSeqSet} = \emptyset$
- 2) $\text{Flows} = \text{GetFlow}(\text{pcap})$
- 3) $\text{Flows} \cup = \text{Mutate}(\text{Flows})$
- 4) **foreach** flow **in** Flows **do**
- 5) $\text{typeseq}_i, \text{typeseq}_o = \text{GetType}(\text{flow})$
- 6) $\text{TypeSeqSet} = \text{UniqueAdd}(\text{typeseq}_i, \text{typeseq}_o)$
- 7) **end for**
- 8) $\text{typeseq}_i, \text{typeseq}_o = \text{GetMaxSeq}(\text{TypeSeqSet})$
- 9) $I, \text{Mapper} = \text{Construct}(\text{typeseq}_i)$
- 10) $\mathcal{M} = \text{MAT}(I, \text{Mapper})$
- 11) **foreach** $\text{typeseq}_i, \text{typeseq}_o$ **in** TypeSeqSet **do**
- 12) **if** $\text{typeseq}_o \neq \mathcal{M}(\text{typeseq}_i)$ **then**
- 13) $I, \text{Mapper} \cup = \text{Construct}(\text{typeseq}_i)$
- 14) $\mathcal{M} = \text{MAT}(I, \text{Mapper})$
- 15) $I, \text{Mapper}, \mathcal{M} = \text{Dedup}(\mathcal{M})$
- 16) **end if**
- 17) **end for**

综上所述，核心算法如算法 2 所示，其中第 2)~7) 行是基于被动学习的预处理过程，第 2) 行是对数据包文件进行预处理，以流为单位进行存储，第 3) 行是基于已有数据进行变异得到半合法的数据，第 4)~7) 行是依据已有知识提取消息类型，生成每条流的输入输出类型序列，并对具有相同的输入输出序列或互相包含的序列进行剔除。第 8)~17) 行是渐进式主动学习的过程，其中第 8)~10) 行是初始化，首先选取其中包含类型最丰富的流，并依据此构建字母表和映射器，基于 MAT 框架学习初始状态机 $\mathcal{M}$ ；第 11)~17) 行是循环对每条流进行分析，首先判断其是否符合当

前状态机，不符合则需要将消息流中的所有子类型加入字母表和映射器中，重新推断状态机，并对子类型进行去重，最后遍历所有的数据流，包括通过确定性变异获得的数据。算法的最终输出为状态机。

本文方法的处理过程如图4所示，输入为按流划分的协议数据，其中淡蓝色为请求数据，黄色为响应数据。预处理过程分为两步：首先依据协议逆向工程或RFC规范提取每条流的类型序列；而后依据类型序列进行分类，并将重复序列剔除，每类数据流仅包含一条流与其对应的类型序列，如图中流1、3、6属于同一类，则只保留一条流代表该类。状态机推断过程分为初始化过程和渐进学习过程，初始化时选取一条类型最多的流构建字母表和映射器，并基于主动学习获得初始状态机，如图

中状态机①所示，其中使用“0”标记的为初始状态。蓝色箭头标识的是渐进学习过程，算法对每条流进行分析，判断其输入输出类型序列是否符合当前状态机，不符合则需扩充字母表和映射器，并重新学习获得状态机②，其中虚线部分是新增的子类型造成的新的状态转移，需经过子类型去重算法对子类型进行判断，若造成了新的状态转移，则该子类型保存，否则将删去该子类型，经确认后得到状态机③，虚线部分经确认后或删除或变为实线，表明造成了新的状态转移。完成对已有数据分析后，可基于变异生成新的输入数据，如图中橙色部分，对流2的第2个消息进行变异，可能得到新的响应类型，将其视为子类型，并进入渐进式学习过程。对已有种子遍历完成后，生成最终的状态机④。

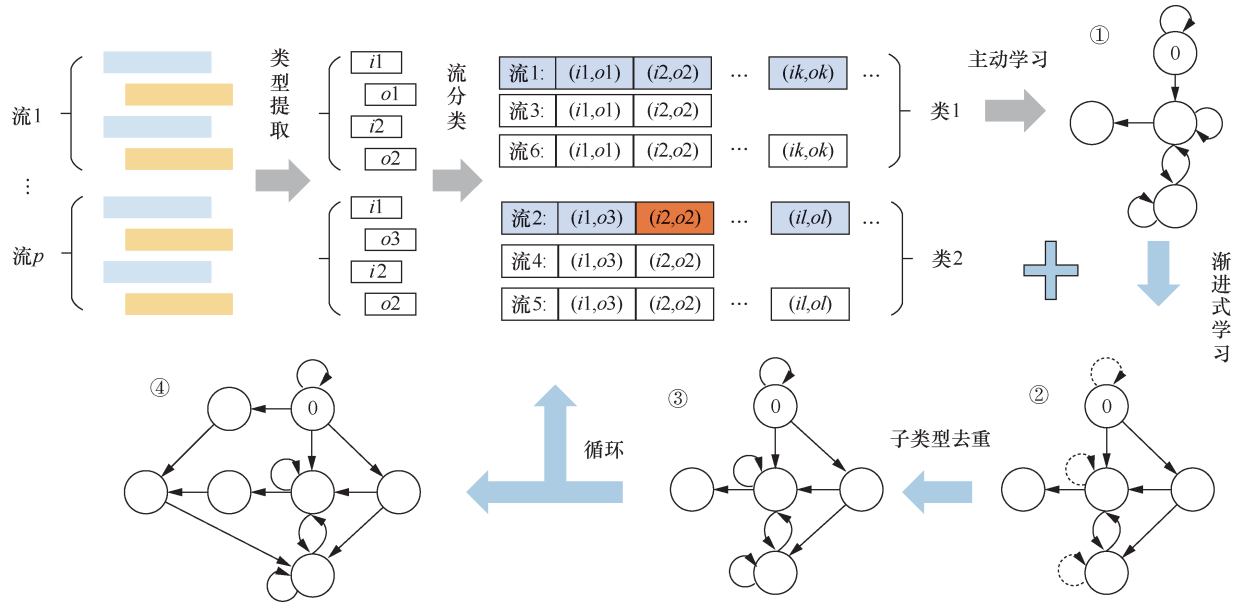


图4 本文方法的处理过程

Figure 4 The processing procedure map of the proposed method

2.5 预响应查询

除此之外，依据协议的特点，针对学习过程提出一种基于前缀匹配的预响应查询机制。实际系统的主动自动机学习效率决定因素为实际查询次数，即与协议实现交互的次数。因此，为尽可能地减少实际查询次数，LearnLib 框架^[20]在映射器与目标系统之间提供了缓存机制，对所有的查询进行缓存，相同的查询无须进行实际交互。与 LearnLib 的缓存类似，王辰等^[24]通过构建扩展前

缀树接收器实现成员查询的精准匹配；Yang 等^[29]通过已有数据构建扩展前缀树接收器辅助寻找反例，减少等价查询次数，Ruiter 等^[17]改进等价查询，若查询过程中 TCP 连接关闭，则该次查询的后续消息的输出统一设定为“Closed”。

基于当前研究，本文提出一种基于前缀匹配的预响应查询算法，适用于成员查询和等价查询，对所有查询进行预响应，降低实际交互次数。原理为 $\exists w \in I^*, \text{s.t. } \lambda_A(w). \text{contain}(\text{closed}) = \text{true}$ ，则

有 $\forall v \in I^*, \lambda_A(w \cdot v) = \lambda_A(w) \cdot \text{Closed}^{|v|}$ ，其中， $\text{Closed}^{|v|}$ 表示 $|v|$ 个 Closed 的连接。

基于上述原理，若某次查询中连接关闭，则将该查询序列记录在指定结构体中，在实现查询时基于前缀匹配判断该查询的前缀序列是否在指定结构体中，若不存在，则进行实际的交互查询；否则，直接基于上述原理进行构造响应。基于前缀匹配的预响应查询能有效减少实际交互的次数。

3 实验评估

本文方法基于 LearnLib 开源框架^[20]实现。本节共设置两组实验，实验一比较引入变异模块后的渐进式学习算法、文献算法和传统的主动学习算法是否包括更多的子类型来验证算法的有效性；实验二通过比较 LearnLib 中已有算法和预响应查询算法所需时间和实际交互次数来验证预响应查询算法的有效性。

实验对象为 RTSP 与 SMTP 及其开源实现。RTSP 可基于 TCP 或 UDP，本文实验是基于 TCP 的。SMTP 协议是邮件传输协议，实验忽略了协议认证的过程。两个协议提取类型的函数是基于分隔符进行提取的，而在确定输出类型时是提取响应数据的前 3 个字节。本文选取的协议实现为 Live555 (version: 20200809、20210824) 和 EXIM (version: 4.93、4.94)，实验环境为 Ubuntu16.04+LearnLib0.17.0。

即使通过去重删除了部分子类型，添加依据数据包和变异所生成的子类型后协议状态机仍非常复杂，为了易读性和易理解性，剩余的子类型可依据是否产生了新状态分为两类。若子类型的加入导致了状态数的增加，则认为该子类型是更有意义的，否则可能只是协议消息经过变异后生成的消息格式服务器无法识别，从而使得协议实现的响应是一种新的类型（错误或报警），将其称为无效子类型。

如果在学习的过程中对无效子类型进行去除，会增加后续学习的成本，因为后续的子类型可能与该子类型具有相同的响应类型。因此在后续结果展示中，是在学习结束后对此类无效子类型进行去除。后文中图中的状态机均已去除无效子类型。

对比实验的方法主要包括手动构建映射器算法（后文中称为传统算法^[16]）和文献[7]中的算法（后文中简称为文献算法）。传统算法的基本思路是基于 RFC 规范提取协议消息的输入输出类型，依据类型构造输入输出字母表，对于每一种消息类型，选取一个典型的数据包作为其在映射器中字母对应的具体消息，同时在实际交互时对会话相关的字段进行相应的更新；文献[7]不包含对数据包的变异过程。

3.1 实验一

(1) 以 RTSP 及其实现 Live555 为对象

图 3 是传统的主动学习算法获得的 Live555 状态机，图 5 是基于文献[7]算法所推断的状态机，相较于传统算法新增了 **Setup** 的一种子类型。图 6 是基于渐进式学习算法所获得的状态机，其中蓝色部分是正常逻辑所执行的状态转移，其相较于仅考虑类型时所获得的状态机有一定的差别，每个类型都具有独特的子类型，子类型的数量如表 2 所示，每个子类型都具有与之对应的具体消息，其中第 2 行是未经过有效子类型去重的结果，所生成的子类型是具有一定数量的；第 3 行是剩余的有效子类型，可见变异后的消息会造成新的状态。相较于文献算法，其能扩充更多的异常子类型，造成新的状态。

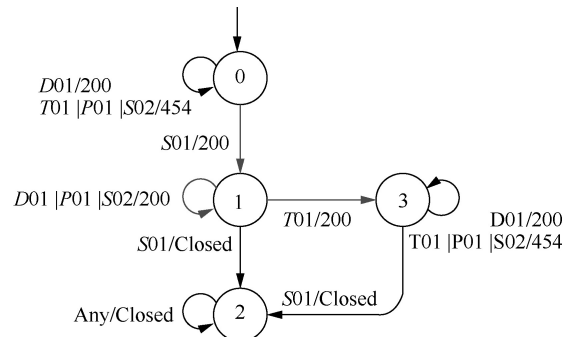


图 5 基于文献[7]算法的 Live555(20210824)协议状态机
Figure 5 The state machine of Live555(20210824) based on the algorithm of the paper[7]

而对于版本 20200809 的学习所获得的状态机与之有所区别，如图 7 所示。分析两者的差异，从大的差异上看，老版本的状态机多了一个状态，多出的状态 6 与状态 1 的差别就在于对 **Setup02** 处理的差异，分析发现老版本实现收到两个 **Setup02** 的数据包后会导致连接关闭，进一步结合

已有的漏洞信息和实现源码分析发现差异是老版本中漏洞 CVE-2021-38381 导致的。

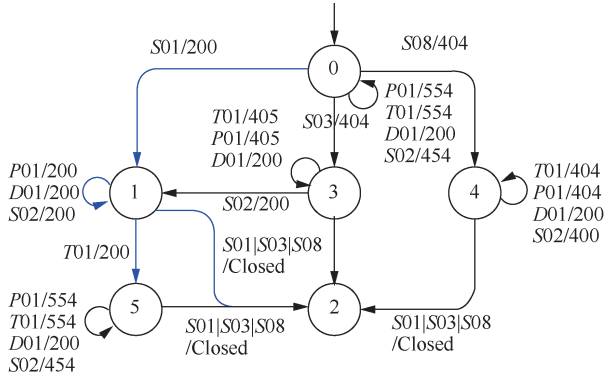


图 6 基于渐进式算法的 Live555(20210824)协议状态机
Figure 6 The state machine of Live555(20210824) based on the progressive algorithm

表 2 RTSP 实现 Live555(20210824)子类型数量
Table 2 The subtype number of RTSP in Live555(20210824)

字母	数量	有效数量
Describe	4	1
Setup	8	4
Play	6	1
Teardown	6	1

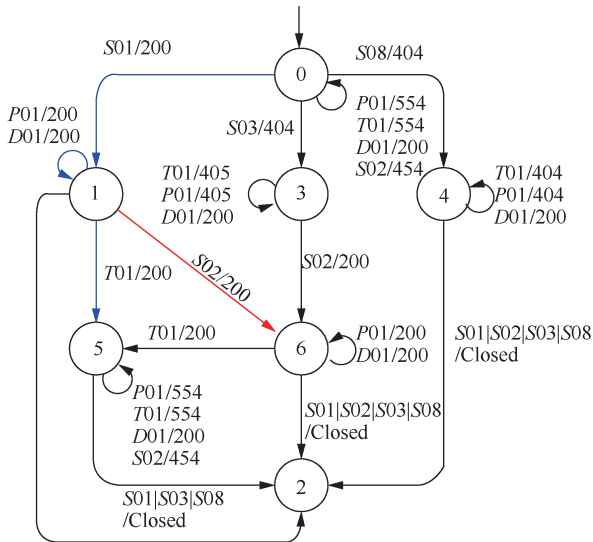


图 7 Live555(20200809)协议状态机
Figure 7 The state machine of Live555(20200809)

(2) 以 SMTP 及其实现 EXIM 为对象

图 8 是基于传统的主动学习算法所获得的 SMTP 实现 EXIM4.93 状态机，为了便于展示将命令 BDAT 删除，因为 BDAT 的存在会导致状态机非常庞大。图中使用 RS 代表 RSET 命令，Msg

表示消息命令，其他则均对应命令的首字母。同样地，蓝色部分是正常逻辑所执行的状态转移，从状态 0-1-3-4，若其中出现了 QUIT 命令则会立即停止。

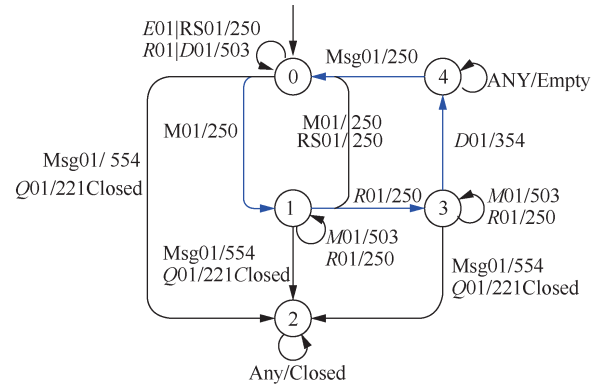


图 8 基于传统算法的 EXIM4.93 协议状态机
Figure 8 The state machine of EXIM4.93 based on the conventional algorithm

由于 SMTP 中并不存在 RTSP 中有多种文件类型的情形，实验仅选取了一个标准数据包作为已知数据包进行实验，因此，文献算法所获得的状态机与传统算法的状态机是一致的。

图 9 是基于渐进式学习算法所获得的状态机，由于对种子进行了一些限制，因此所展示的状态机较为简单。相较于原始的状态机，其对于 Mail From 命令的一个变异会导致一个新状态的产生，具体变异得到的是 Mail From 后续的邮箱未在邮箱服务器注册，其在发送时服务器并不会立即校验邮箱的可用性，而是在下一条 RCPT 命令之后进行校验，并响应 550（要求的邮件操作未完成，邮箱不可用）。具体子类型数量如表 3 所示，由于 DATA、RSET、QUIT 命令均是无参数的，因此没有产生新的子类型，而 MSG 命令是所发送的数据，简单的确定性变异并不会产生新的子类型。对比 RFC 规范，4.93 版本有一处明显违反规范，即对 EHLO 命令的处理，如图中状态 0 的自循环转移（红色标注），RFC 规范中规定协议必须以 EHLO 开始，而 4.93 版本并未严格进行处理，其在 4.94 版本中对该错误进行了更新，如图 10 中的状态机，由于原始的状态机过于复杂，图中的状态机是经过简化展示的。除此之外，其对于 MSG 的处理也有不同之处。

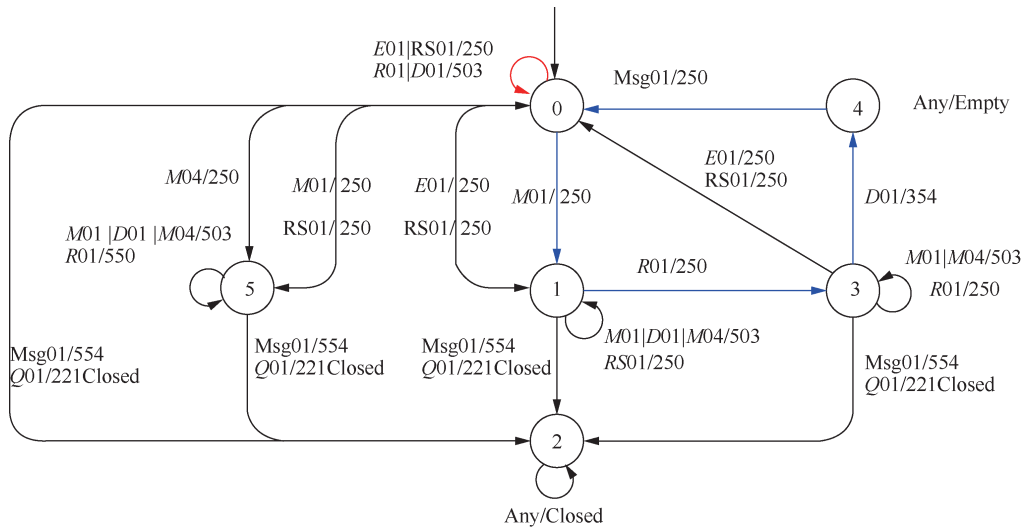


图 9 基于渐进式算法的 EXIM4.93 协议状态机
Figure 9 The state machine of EXIM4.93 based on the progressive algorithm

表 3 SMTP 实现 EXIM4.93 子类型数量
Table 3 The number of subtype of SMTP in EXIM4.93

字母	数量	有效数量
EHLO	2	1
MAIL	4	2
RCPT	3	1
DATA	1	1
MSG	1	1
RSET	1	1
QUIT	1	1

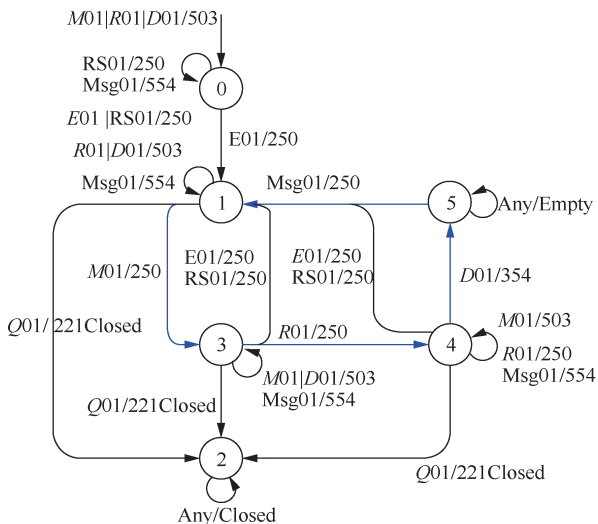


图 10 EXIM4.94 的协议状态机
Figure 10 The state machine of EXIM4.94

3.2 实验二

为分析预响应查询算法的改进效果，以

LearnLib 中的 LogOracle 算法为参照，分别以 RTSP 和 SMTP 为对象进行学习，所需的时间、查询数、实际交互数如表 4 所示，其中“原始”表示 LearnLib 开源框架中的原始算法，“改进”为预响应查询算法。独特查询是指 LearnLib 开源框架本身就提供了缓存查询机制，即相同的输入序列会由缓存查询提供响应，而无须实际交互；学习程序与协议实体的实际交互次数是决定学习过程时间的关键。由表 4 可知，RTSP 的改进算法和原始算法在等价查询过程所花费的时间基本相同，整个学习过程的总独特查询数相同，验证了学习算法的确定性；在学习过程中所花费的时间分别为 1 195 s 和 1 754 s，实际交互次数分别为 8 942 和 10 412，改进算法降低了 14% 的实际交互次数。SMTP 的实际查询数分别为 2 786 和 3 006，降低了 7% 的实际交互次数。上述结果验证了预响应查询算法降低实际交互次数的有效性。

3.3 讨论

相比原有的方法，渐进式学习方法对子类型的初步探讨，是一种启发式算法，基于已有数据包的渐进式学习方法可学到更完善的状态机，尽可能将正常通信行为进行建模。基于变异的策略对类型对应的消息进行多样性扩充，以期获得不同的响应类型，这是一种对子类型的探索。该方法可扩充的子类型空间更大，但较难扩展需要组合消息才能触发的差异。此外，基于当前状态机

表4 预响应查询算法效果比较
Table 4 The comparison of the pre-response query and the original algorithm

协议状态机	算法	学习过程 所需时间/s	等价查询过程 所需时间/s	独特查询数	学习程序与协议实体 的实际交互数
Live555(20210824)	原始	1 754	3 128	11 512	10 412
	改进	1 195	3 124	11 512	8 942(14%)
EXIM4.93	原始	1 226	3 433	4 058	3 006
	改进	1 105	3 445	4 058	2 786(7%)

对于类型是否重复进行判断是局部最优的，并不能完全保证子类型是重复的，若不进行去重则会导致状态数增加较快，从而极大地增加了查询的次数。

与被动推断相比，本文方法得到的状态机更复杂，且对于同一类型的不同消息的跳转是确定性的，子类型与消息的对应关系是明确的，而文献[8]仅能通过概率对不同跳转进行描述，且概率是基于已有数据包的，不同的数据集合会导致不同的概率。

相较于文献[7]，本文方法基于被动学习改进了对数据包的预处理过程，对数据流与状态机的一致性判断进行了简化，并增加了确定性变异模块，进一步探索异常的子类型。但变异模块单次仅对一个消息进行变异，无法探索不同的消息组合所引发的不同响应。

对比寄存器自动机，寄存器自动机可对部分整数型参数进行判断，且现有的寄存器自动机只能对参数进行相等判断，即若参数符合要求，则跳转至正常状态，否则跳转至另一个状态，若包含子类型的 Mealy 状态机仅考虑该参数的变异，也可同样获得类似的状态机，表达方式不同，但具有相同的表达能力，该设想有待证明。而当考虑协议所有字段的变异时，包含子类型的 Mealy 状态机具有更强的表达能力。

4 结束语

本文提出的主被动结合的渐进式协议状态机推断框架可有效对当前的协议状态机推断方法进行扩充，更有效地描述协议的异常行为，提升协议状态机的表达能力。同时，针对协议特点提出的基于前缀匹配的预响应查询算法，能有效降低实际交互的次数，平均降低 10% 的实际交互次数。

局限性在于本文的实现仅在 RTPS、SMTP 这两种文本协议上进行，两种协议的抽象化和具象化工作是较为简单的，可基于 RFC 规范进行提取类型字段，若针对复杂协议或未知协议则需依赖协议逆向工程进行协议的分析；同时，映射器的构建也相对简单，对于加密协议，映射器的自动化构建则需对协议的通信过程进行详尽的分析，解析握手过程中的密钥交换并实现对应的加解密模块，这也是未来工作之一。

参考文献：

- [1] WANG Y P, ZHANG Z B, YAO D F, et al. Inferring protocol state machine from network traces: a probabilistic approach[C]//Proceedings of Applied Cryptography and Network Security. 2011: 1-18.
- [2] HSU Y, SHU G Q, LEE D. A model-based approach to security flaw detection of network protocol implementations[C]//Proceedings of 2008 IEEE International Conference on Network Protocols. 2008: 114-123.
- [3] KRUEGER T, GASCON H, KRÄMER N, et al. Learning stateful models for network honeypots[C]//Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence. 2012: 37-48.
- [4] ANGLUIN D. Learning regular sets from queries and counterexamples[J]. Information and Computation, 1987, 75(2): 87-106.
- [5] KLEBER S, MAILE L S, KARGL F. Survey of protocol reverse engineering algorithms: decomposition of tools for static traffic analysis[J]. IEEE Communications Surveys & Tutorials, 2019, 21(1): 526-561.
- [6] CASSEL S, HOWAR F, JONSSON B, et al. A succinct canonical register automaton model[J]. Journal of Logical and Algebraic Methods in Programming, 2015, 84(1): 54-66.
- [7] SZÉKELY G, LÁDI G, HOLCZER T, et al. Protocol state machine reverse engineering with a teaching-learning approach[J]. Acta Cybernetica, 2021, 25(2): 517-535.
- [8] SUN F H, WANG S, ZHANG H L. A progressive learning method on unknown protocol behaviors[J]. Journal of Network and Computer Applications, 2022, 197: 103249.
- [9] LEITA C, MERMOUD K, DACIER M. ScriptGen: an automated script generation tool for Honeyd[C]//Proceedings of 21st Annual Computer Security Applications Conference (ACSAC'05). 2005: 203-214.

- [10] COMPARETTI P M, WONDRAK G, KRUEGEL C, et al. ProspeX: protocol specification extraction[C]//Proceedings of 2009 30th IEEE Symposium on Security and Privacy. 2009: 110-125.
- [11] KRUEGER T, GASCON H, KRÄMER N, et al. Learning stateful models for network honeypots[C]//Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence. 2012: 37-48.
- [12] LEE C, BAE J, LEE H. PRETT: protocol reverse engineering using binary tokens and network traces[C]//Proceedings of ICT Systems Security and Privacy Protection. 2018: 141-155.
- [13] LIN Y D, LAI Y K, BUI Q T, et al. ReFSM: Reverse engineering from protocol packet traces to test generation by extended finite state machines[J]. Journal of Network and Computer Applications, 2020, 171: 102819.
- [14] GOLD E M. Language identification in the limit[J]. Information and Control, 1967, 10(5): 447-474.
- [15] CHO C Y, C D B, SHIN E C R, et al. Inference and analysis of formal models of botnet command and control protocols[C]//Proceedings of the 17th ACM Conference on Computer and Communications Security. 2010: 426-439.
- [16] AARTS F D. Tomte: bridging the gap between active learning and real-world systems[J]. Model Based System Development, 2014.
- [17] RUITER J, POLL E. Protocol state fuzzing of TLS implementations[C]//Proceedings of 24th USENIX Security Symposium. 2015: 193-206.
- [18] 申莹珠, 顾顺祥, 陈熹, 等. 基于模型学习的 OpenVPN 系统脆弱性分析[J]. 软件学报, 2019, 30(12): 3750-3764.
SHEN Y Z, GU C X, CHEN X, et al. Vulnerability analysis of OpenVPN system based on model learning[J]. Journal of Software, 2019, 30(12): 3750-3764.
- [19] FITERAU-BROSTEAN P, JONSSON B, MERGET R, et al. Analysis of DTLS implementations using protocol state fuzzing[C]//Proceedings of 29th USENIX Security Symposium. 2020: 2523-2540.
- [20] RAFFELT H, STEFFEN B, BERG T. LearnLib: a library for automata learning and experimentation[C]//Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems. 2005: 62-71.
- [21] FITERAU-BROSTEAN P, JANSSEN R, VAANDRAGER F. Combining model learning and model checking to analyze TCP implementations[C]//Proceedings of Computer Aided Verification. 2016: 454-471.
- [22] FITERAU-BROSTEAN P, HOWAR F. Learning-based testing the sliding window behavior of TCP implementations[C]//Proceedings of Critical Systems: Formal Methods and Automated Verification. 2017: 185-200.
- [23] GUO J X, GU C X, CHEN X, et al. Model learning and model checking of IPSec implementations for internet of things[J]. IEEE Access, 2019, 7: 171322-171332.
- [24] 王辰, 吴礼发, 洪征, 等. 一种基于域知识的协议状态机主动推断算法[J]. 计算机科学, 2015, 42(12): 233-239.
WANG C, WU L F, HONG Z, et al. Domain-specific algorithm of protocol state machine active inference[J]. Computer Science, 2015, 42(12): 233-239.
- [25] ISBERNER M. Foundations of active automata learning: an algorithmic perspective[D]. Technische Universität Dortmund, Dortmund, 2015.
- [26] CENGİZ TÜRKER U, HIERONS R M, JOURDAN G V. Minimizing characterizing sets[J]. Science of Computer Programming, 2021, 208: 102645.
- [27] HOWAR F, STEFFEN B, JONSSON B, et al. Inferring canonical register automata[C]//Proceedings of International Workshop on Verification, Model Checking, and Abstract Interpretation. 2012: 251-266.
- [28] FANG D L, SONG Z W, GUAN L, et al. ICS3Fuzzer: a framework for discovering protocol implementation bugs in ICS supervisory software by fuzzing[C]//Proceedings of ACSAC: Annual Computer Security Applications Conference. 2021: 849-860.
- [29] YANG N, ASLAM K, SCHIFFELERS R, et al. Improving model inference in industry by combining active and passive learning[C]//Proceedings of 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering. 2019: 253-263.

[作者简介]



潘雁(1995-), 男, 安徽安庆人, 信息工程大学博士生, 主要研究方向为网络协议分析、软件逆向。



林伟(1986-), 男, 湖南常德人, 信息工程大学讲师, 主要研究方向为软件保护与分析、协议逆向。



祝跃飞(1962-), 男, 浙江兰溪人, 信息工程大学教授、博士生导师, 主要研究方向为网络安全、密码学。