软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(12):3750−3764 [doi: 10.13328/j.cnki.jos.005612]

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

基于模型学习的 OpenVPN 系统脆弱性分析*

申莹珠[1], 顾纯祥[1,2], 陈 熹[1], 张协力[1], 卢政宇[1]

[1](信息工程大学 网络空间安全学院,河南 郑州 450001)

[2](河南省网络密码技术重点实验室,河南 郑州 450001)

通讯作者: 顾纯祥, E-mail: gcx5209@sohu.com

摘 要: OpenVPN 在现实网络中有广泛应用,对其安全性进行评估具有重要的现实意义.基于自动机理论中模型学习的方法,利用协议状态模糊测试的技术对 OpenVPN 系统进行黑盒测试分析,自动化推演出目标 OpenVPN 系统的状态机.提出了状态机时间压缩模型并进行冗余状态和迁移化简,可以准确得到协议状态机中的行为特征.发现了多条期望行为路径外的特别行为路径及可能的安全隐患,为 OpenVPN 的安全性评估提供了新的思路与方法,同时对类似缺少协议规范但应用广泛的安全协议的内部设计细节分析具有重要参考意义.

关键词: OpenVPN;模型学习;状态模糊测试;脆弱性分析;时间压缩模型

中图法分类号: TP311

中文引用格式: 申莹珠,顾纯祥,陈熹,张协力,卢政宇.基于模型学习的 OpenVPN 系统脆弱性分析.软件学报,2019,30(12):3750−3764. http://www.jos.org.cn/1000-9825/5612.htm

英文引用格式: Shen YZ, Gu CX, Chen X, Zhang XL, Lu ZY. Vulnerability analysis of OpenVPN system based on model learning. Ruan Jian Xue Bao/Journal of Software, 2019,30(12):3750−3764 (in Chinese). http://www.jos.org.cn/1000-9825/5612.htm

Vulnerability Analysis of OpenVPN System Based on Model Learning

SHEN Ying-Zhu[1], GU Chun-Xiang[1,2], CHEN Xi[1], ZHANG Xie-Li[1], LU Zheng-Yu[1]

[1](Information Engineering University, Zhengzhou 450001, China)

[2](He'nan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China)

Abstract: OpenVPN is widely used in the real network, the assessment of its security has important practical significance. In this study, technology of state fuzzing is used to carry out black box test on OpenVPN implementation to infer state machine of the target system automatically based on model learning method in automata theory. Time compression model is proposed and state machine of OpenVPN is simplified to remove the redundant states and transitions. Then, the behavior characteristics of the protocol state machine will be obtained accurately to discover a number of special behavior paths and potential security risks outside the expected behavior path. It provides a new idea for the security evaluation of OpenVPN and has important significance for obtaining the internal design details of similar security protocols with little specification but widely used.

Key words: OpenVPN; model learning; protocol state fuzzing; vulnerability analysis and detection; time compression model

近年来,网络安全协议相关应用的安全性问题越来越受到重视.2014 年,开源加密库 OpenSSL 的重大安全漏洞"心脏出血"(CVE-2014-0160)引起了业界广泛关注,攻击者可以从服务器内存中读取包括用户名、密码和信用卡号等隐私信息在内的数据,影响波及大量互联网公司.2016 年,OpenSSL 高危漏洞"DROWN"(CVE-2016-0800)以及 2017 年的 OpenSSL 拒绝服务漏洞(CVE-2017-3731)也同样为网络协议的安全性敲响了警钟.因此,如

* 基金项目: 国家自然科学基金(61502533); 河南省自然科学基金(162300410335)

Foundation item: National Natural Science Foundation of China (61502533); Natural Science Foundation of He'nan Province of China (162300410335)

收稿时间: 2017-12-07; 修改时间: 2018-04-22; 采用时间: 2018-05-29

何对网络安全协议的安全性进行评估、尽快尽早发现其协议实现中的脆弱点,保护用户隐私数据安全可靠就显得格外重要.OpenVPN 是一款基于 OpenSSL 库的应用层虚拟专用通道(VPN),在 TLS 之上建立安全的数据传输隧道,实现身份认证、数据加密、完整性保护和访问控制.OpenVPN 在真实网络中被广泛部署且常应用于大型企业之中,对其进行脆弱性分析有着重要的现实意义.

协议脆弱性分析检测技术也叫协议漏洞挖掘技术.传统的漏洞挖掘技术主要依赖于安全人员的人工分析与测试,而模糊测试技术简单、有效、自动化程度比较高,是目前进行安全测试最有效的方法,被广泛应用于 Web、系统、应用程序的漏洞挖掘.由于它一般属于黑盒测试,通过构造有效的畸形数据进行测试,因此该技术的代码覆盖率相对较低.2015 年,Gascon 等人[1]提出了可对私有协议进行态式黑盒模糊测试的 PULSAR 系统,通过将模糊测试与协议逆向、模拟自动化执行技术结合,提高了协议状态探索空间,能够挖掘协议实现中的深层漏洞;2016 年,Ma 等人[2]优化测试数据生成方法,提出了使用基于规则的状态机和状态规则树来生成模糊测试的数据,与传统模糊测试方法相比,使用更少的测试数据找到脆弱点,提高了测试效率;2017 年,Kang 等人[3]提出了一种结合静态分析和动态分析的智能模糊测试系统,在提高检测有效性的同时,减少了误报率和漏报率.

协议状态机推断是脆弱性分析非常关键的内容.2015 年,de Ruiter 提出了通过模型学习的方法分析具体 TLS 实现的安全性[4],该方法在仅采用黑盒测试的情况下,应用状态机学习自动推断出了协议实现的状态机,并通过观察推断出来的状态机来检测可能由程序逻辑漏洞引起的异常行为;Beurdouche 等人[5]也提出了类似的通过系统测试非正常 TLS 消息流以检测协议实现中是否存在脆弱性的方法,并在测试中发现了新漏洞.2016 年,Ruiter 带领团队在其前期工作的基础上继续进行了许多相关研究:Ruiter 进一步对过去 14 年以来的 OpenSSL 以及 LibreSSL 的具体实现进行了并行自动化协议状态模糊测试,通过自动化学习,为 145 个不同版本的服务器端和客户端构建了状态机,并分析相关实现的安全性[6];Verleg 则针对 OpenSSH 推断出了 6 个 SSH 服务器的状态机,验证了协议状态机推断方法的通用性和可行性[7].同年,Somorovsky 提出了基于已知漏洞专家库的分阶段模糊测试框架 TLS-attacker[8],能够提供简单接口支持定制 TLS 消息流,并允许任意修改消息中的内容,从而实现对 TLS 库安全性的评估.2017 年,Ruiter 团队的 Lenaerts[9]对 Verleg 的方法[7]进行了改进,将 Verleg 分别对 SSH 协议的 3 个子层协议进行模型学习与检测的方法整合为统一的模型学习与检测,提出了对 SSH 服务器模型进行状态安全检测的新方法;同年,Veldhuizen 针对 IPSEC 也进行了协议状态机推断并分析[10].Novickis[11]在 2016 年将模型学习的方法扩展到 OpenVPN 协议进行研究分析,试图推断出 OpenVPN 系统的状态机,但在过程中遇到了许多困难最终并未实现目标.

本文基于模型学习的方法,利用协议状态模糊测试技术,对 OpenVPN 2.0.9 服务器端进行了模型学习和一致性检测,自动推演出目标系统的协议实现状态机,并对其状态机模型进行化简,对其实现逻辑及过程进行详细分析;提出了一种状态机时间压缩模型,与原始状态机及同版本 OpenSSL 状态机进行对比分析.结果表明:利用经化简后的模型可以更方便迅速地识别协议状态机中正常的和特殊的状态迁移路径,从而提高 OpenVPN 脆弱性检测与分析的效率.

1 基础知识

1.1 OpenVPN协议

VPN(虚拟专用通道)是企业与企业或者个人与企业之间安全数据传输的隧道,提供了身份认证、数据加密、完整性保护和访问控制安全服务.OpenVPN 是一款基于 OpenSSL 库的应用层 VPN 实现,由于其简单易用的特性而被广泛部署.OpenVPN 依赖 OpenSSL 的安全性,在客户端和服务器端通过指定端口建立 TCP/UDP(一般默认使用 UDP)安全隧道,然后在该 TLS 隧道中加密通信数据,隧道示意如图 1 所示.

- OpenVPN 协议 TLS 模式下的实现

OpenVPN 提供了两种完全不同的认证模式:TLS 模式和预共享静态密钥(PSK)模式.预共享静态密钥模式使用预共享静态密钥认证身份并加密.TLS 模式采用使用证书的 SSL/TLS 协议进行身份验证、建立安全隧道、交换对称会话密钥,并使用会话密钥加密数据隧道.TLS 模式由于能够保证安全地分发和更新对称密钥,进而在

现实应用中具有更强的安全性.因此,本文主要研究基于 TLS 认证模式的 OpenVPN 协议.

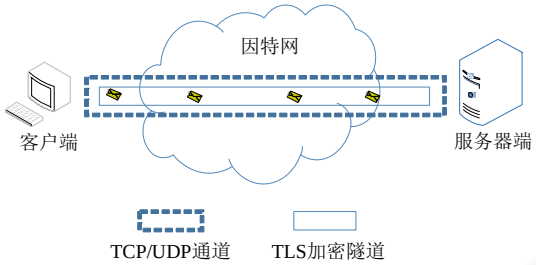


Fig.1 OpenVPN communication tunnel

图 1 OpenVPN 通信隧道

OpenVPN 安全通信过程如图 2 所示,依赖于以下子协议.

(1) OpenVPN 握手协议,类似于 TCP 的 3 次握手过程,握手数据包包括:由客户端发起的握手请求数据包 P_CONTROL_HARD_RESET_CLIENT_V1/V2(V1/V2 代表后续两种不同的密钥协商方式,分别对应 PSK 认证模式和 TLS 认证模式,OpenVPN 2.0 以上版本默认使用 P_CONTROL_HARD_RESET_CLIENT_V2),服务器端的请求响应数据包 P_CONTROL_HARD_RESET_SERVER_V1/V2(密钥协商方式与请求一致),客户端对服务器端响应的确认 P_ACK_V1;

(2) OpenVPN 控制协议,控制协议包括 TLS 握手和密钥协商两个阶段,封装在 P_CONTROL_V1 数据包中.当 TLS 握手完成后,TLS 加密隧道已建立,会话密钥协商信息将被封装在 TLS 记录层中安全传输.需要时,可使用 P_CONTROL_SOFT_RESET_V1 请求会话密钥冲协商;

(3) OpenVPN 记录协议,建立安全隧道和密钥交换完成后,双方进行加密数据通信 P_DATA_V1.

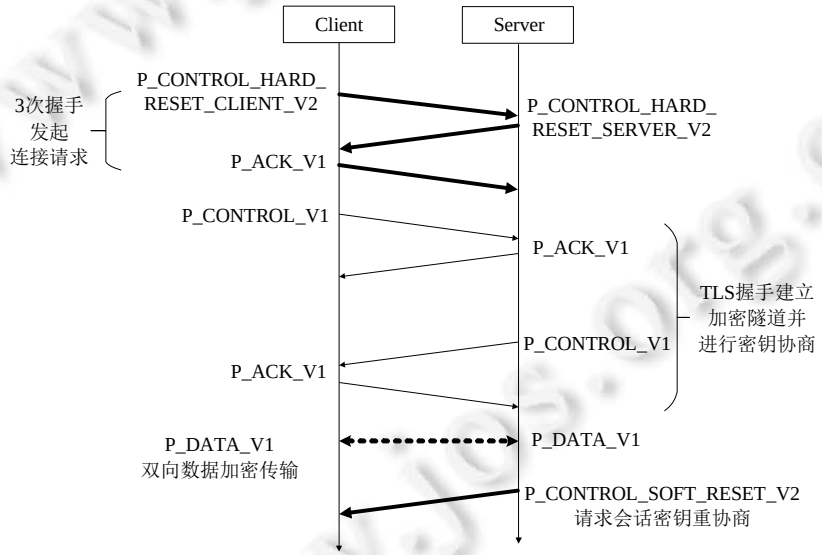


Fig.2 OpenVPN communication process in TLS mode

图 2 TLS 模式下 OpenVPN 通信过程

OpenVPN 没有详细的官方规范,2016 年,Tomas Novickis 方法[11]中给出的在 TLS 模式下期望的 OpenVPN 状态机如图 3 所示.Novickis 试图基于 LearnLib 得出真实的 OpenVPN 实现的状态机,但其在研究过程中主要遇到了以下困难:(1) 使用 python 库 Scapy 可以快速构造数据包,但仅尝试利用之前 Wirshark 捕获的数据包载荷进行复用,未根据协议原理精心设计每个数据包字段,导致无法通过证书验证或 HMAC 校验;(2) P_CONTROL_

V1 数据包载荷较多需要分块传输时还存在一些问题.因此,作者得到 OpenVPN 实现状态机的目标并未实现.

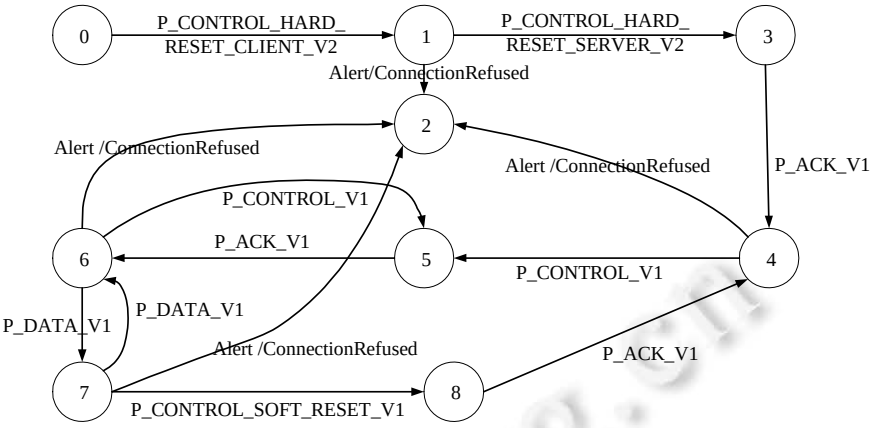


Fig.3 The expected OpenVPN state machine

图 3 期望的 OpenVPN 状态机

1.2 模型学习与一致性检测

为了解决使用常规技术难以分析检测安全协议的具体实现与协议标准之间差异性的问题,采用 Angluin 的 MAT(minimally adequate teachers)框架[12],使用成员查询和等价查询进行模型学习;通过测试查询进行一致性测试,确保模型的推断结果符合要求;最后,进一步分析检测协议实现是否存在脆弱点.在 MAT 框架中,学习器必须通过询问预言机来推断目标安全协议的状态机.图 4 为 MAT 框架示意图.

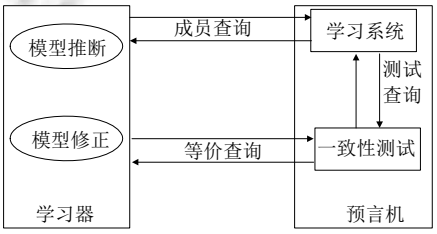


Fig.4 MAT framework

图 4 MAT 框架

基于 MAT 框架,模型推断与一致性检测系统可抽象为图 5 所示,学习器提供了一个可以发送给测试目标(SUT)的消息列表(输入表)以及一条重置测试目标到初始状态的命令.测试程序(test harness)可以将输入表中的抽象消息转换为可以发送给 SUT 的具体消息,也可以将 SUT 反馈的响应转换为学习器可识别的抽象消息.因此,测试程序相当于 MAT 框架中在学习器和预言机之间进行翻译转换的映射器(mapper).而实现测试程序就需要我们知道目标协议使用的具体消息集.

具体过程如下.

a. 成员查询:通过发送一系列消息和重置命令,学习器使用如最早由 Angluin 提出的 L^*算法[12]、改进 L^*后去除大量冗余成员查询的 TTT 算法[13]等经典有限状态机学习算法,通过从 SUT 返回的响应推断出状态机模型;

b. 等价查询:采用近似等价查询算法进行一致性检测,如 Chow 提出的 W-method 算法[14]、Ruiter 改进的 wmethod[4]等.通过有限数量的测试查询,检测该推断是否与实际的状态机等价.如果不等价,将会返回一个反例,学习器使用该反例重新进行推断,即进行模型修正.如果没有找到反例,则认为当前状态机推断近似等价于真实实现,得到协议实现的状态机.

Fig.5 System architecture

图 5 系统架构

2 基于模型学习的 OpenVPN 状态机推断

本文在模型学习与一致性检测技术的基础上,基于 LearnLib 平台[15,16]开发了针对 OpenVPN 的状态机推断框架,本文只考虑 TLS 模式下的 OpenVPN 状态机推断问题.

- 实验环境:Intel core i7-4790 处理器、8G 内存、Ubuntu 16.04-64 位系统、Wireshark 软件;
- 协议版本:OpenSSL 1.0.2h,OpenVPN 2.0.9;
- 编程语言:Java,Python;
- 环境配置:采用桥接模式,在两台虚拟机中分别配置本文系统与 OpenVPN 2.0.9 服务器端环境进行测试.其中,OpenVPN 通信采用默认的 UDP 连接方式(在 OpenVPN 协议的可靠传输要求下,即使采用 TCP 连接,也同样需要实现确认机制).

2.1 输入/输出表构造

本文主要测试 OpenVPN 服务器端状态机,因此根据 OpenVPN 协议中客户端消息集构造输入表以及服务器端响应消息集构造输出表,输入/输出表的符号表示及消息含义见表 1.

Table 1 Symbol representation and message meaning in input/output table

表 1 输入/输出表符号表示及消息含义

	符号表示	消息含义
输入表	PHRCV2	客户端向服务器端请求连接 P_CONTROL_HARD_RESET_CLIENT_V2
	PACK	确认消息
	PCH	TLS 隧道连接请求 ClientHello 消息
	PCC	TLS 隧道客户端证书 ClientCertificate 消息
	PCKE	TLS 隧道密钥交换 ClientKeyExchange 消息
	PCV	TLS 隧道证书校验 CertificateVerify 消息
	PCCS	TLS 隧道改变密码套件通知 ChangeCypherSpec 消息
	PF	TLS 隧道建立完成 Finished 消息
输出表	PHRSV2	服务器端对客户端请求的响应 P_CONTROL_HARD_RESET_Server_V2
	PACK	确认消息
	PSH	TLS 隧道请求响应 ServerHello 消息
	PC	TLS 隧道服务器端证书 Certificate 消息
	PCR	TLS 隧道请求客户端证书 CertificateRequest 消息
	PSHD	TLS 隧道建立阶段 ServerHelloDone 消息
	PCCS	TLS 隧道改变密码套件通知 ChangeCypherSpec 消息
	PF	TLS 隧道建立完成 Finished 消息
	CLOSED	连接关闭

转换程序 Test harness 基于 OpenVPN 底层协议,对学习器的消息输入表中的消息进行精心构造,按照学习器中的状态机推断与近似等价算法与测试目标进行交互,并将服务器端返回的响应消息识别、抽象,送入学习器继续分析推断,直到完成真实的 OpenVPN 服务器端状态机的学习与检测;通过比对协议实现状态机与协议标准状态机(或期望的状态机)之间的差异性,找到可能存在的攻击路径.

2.2 数据包构造及状态机推断策略

网络安全协议的执行路径中往往存在许多条件控制,这也正是限制传统模糊测试进行脆弱性分析检测能力与效率的一个重要方面.因此在状态模糊测试中,数据包构造是整个测试的基础[17].OpenVPN 协议中只有正

确的证书验证、HMAC 校验等才能顺利完成整个连接过程.

本文参考 RFC 2246 及 Wireshark 捕获的 OpenVPN 真实通信数据,精心构造转换程序中使用的数据包,并按照期望的路径进行测试,保证构造的消息集合产生的消息流能够生成正确的执行路径.

在 TLS 模式下,OpenVPN 使用 TLS 协议认证、建立安全隧道,并交换安全隧道的会话密钥.基于 TCP/UDP 的 OpenVPN 报文格式如图 6 所示.

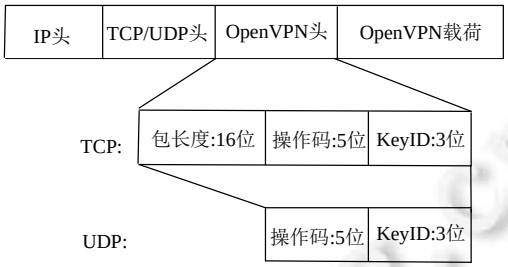


Fig.6 Message format

图 6 报文格式

OpenVPN 报文主要包括 IP 头、TCP/UDP 头、OpenVPN 头和 OpenVPN 载荷字段.其中,基于 TCP 和 UDP 协议的 OpenVPN 头部有细微差异,UDP 包含 5 位操作码字段以及 3 位 KeyID(密钥标号),基于 TCP 连接的报文的 OpenVPN 头部信息还包括 16 位的包长度字段.本文实验采取 OpenVPN 默认的基于 UDP 连接的方式.数据包构造示例如下.

OpenVPN 初始化时的第 1 个消息是客户端向服务器端请求连接的 PHRCV2(P_CONTROL_HARD_RESET_CLIENT_V2)消息,其操作码为 0X07,keyID 为 0,因此,其 VPNType(由 5 位操作码与 3 位 KeyID 串联而成)为 MSG_TYPE_P_CONTROL_HARD_RESET_CLIENT_V2=0X38,代码如下:

```
public class PControlHardRestClientV2 {
  protected byte VPNType;          //opcode(5 bit)+keyid(3 bit)  1byte
  protected byte[] sessionId;      //8 bytes
  protected int p_id_arry_len;     // number of packets to ack   1 byte
  protected byte[] p_id_arry_len_Byte;
  protected byte[] p_id ;          //this packet's number
  … }
```

按照上述方法依次构造数据包,有效性测试结果如图 7 所示.可以看出,本文构造的数据包在测试中能够突破条件控制的限制.

OpenVPN 协议没有标准的协议规范,现有的可参考研究资料很少,系统中映射程序只能根据 Wireshark 工具捕获真实数据包结合读 OpenVPN 源码来实现.结合 Open VPN 系统特点,在状态机推断过程中本文制定以下策略.

(1) OpenVPN 按消息生成顺序为数据包编号,因此在进行状态模糊测试的过程中需要正确处理消息序号,否则会造成正确路径难以正常执行;

(2) 为了防止确认机制导致的 ACK 数据包过多而出现无限状态机,因此在测试程序 test harness 中每每收到服务器端发回的响应都对其进行确认,且认为状态不改变;

(3) 测试程序每次执行重置 reset 指令后,为了防止由于异常数据流导致抛出空指针异常,要对所有内部变量进行初始化;

(4) 出于应用场景及安全性考虑,OpenVPN 建议默认配置下服务器要对客户端身份进行验证,因此测试程序对 TLS 中为可选项的客户端证书验证过程进行了实现,同时也对客户端证书为空时服务器端具

体实现的逻辑行为进行了测试;

(5) 网络传输消息的拆分与组包,以及对每个拆分后数据包响应的识别;

(6) OpenVPN 密钥协商子协议是在 TLS 加密隧道中完成的,整个过程都为密文封装,因此不再进行后续分析.

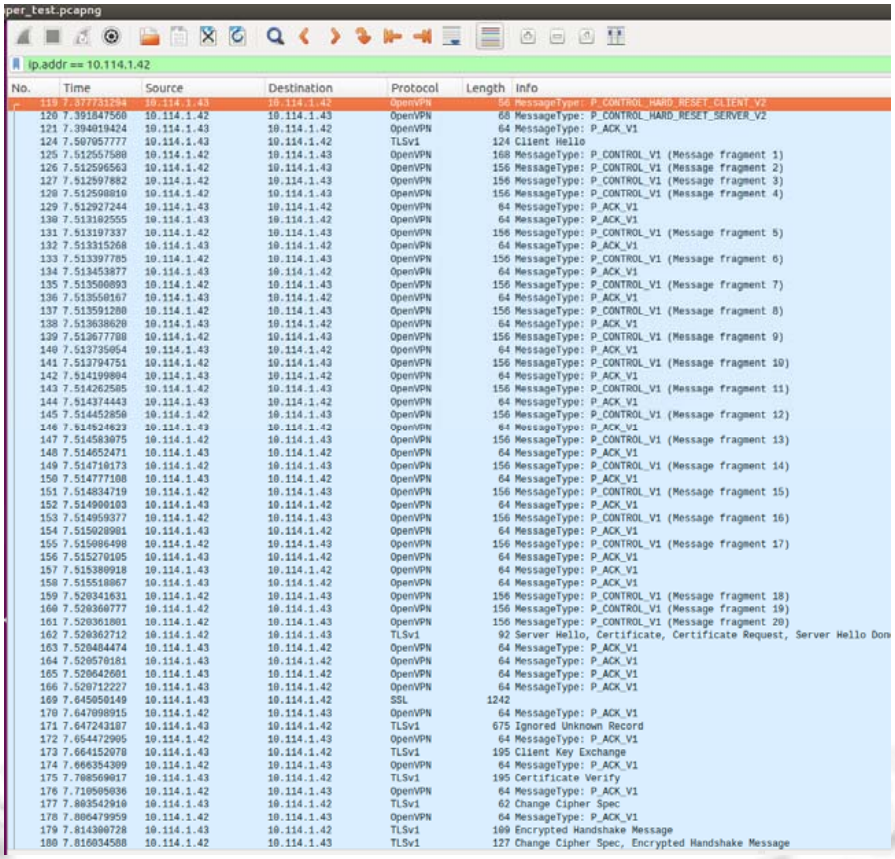


Fig.7 Packet construction test

图 7 数据包构造测试

2.3 状态机推断与化简

实验中,状态机推断采用经典的 L^*[12]算法,一致性测试采用改进的 W[4]方法,用时 19 小时 53 分钟,经过 76 轮次的推断与修正,推演出 OpenVPN 系统的状态机见附录 1.在本文所示的状态机图中,节点代表状态,*S*0 为起始状态,*S*2 为连接关闭状态.边上 I/O 形式的输入/输出符号对代表输入消息以及目标系统返回的响应消息.如 *S*0→*S*1,表示向目标 OpenVPN 系统发送 PHRCV2 消息后得到响应 PHRSV2,由状态 *S*0 迁移至状态 *S*1.

推断得到的原始状态机共存在 18 个不同状态、133 条状态迁移,状态迁移数量较多.过于复杂的状态迁移使安全性分析非常困难.对推断出的状态机进一步分析发现:一方面,在 TLS 隧道建立过程中,任意握手数据包或 TLS 数据流的错误都会导致连接关闭,产生大量关闭连接的状态迁移;另一方面,由于 OpenVPN 的确认机制,大量 ACK 数据包会导致状态机迁移冗余.然而这些状态迁移对协议实现的安全性分析并无影响,因此对原始状态机进行化简:将 PCH/CLOSED,PCC/CLOSED,PCKE/CLOSED,PCV/CLOSED,PCCS/CLOSED,PF/CLOSED 合并为 Other/CLOSED,同时将服务器端接收不同消息并确认合并为状态迁移 PF/PACK||PCCS/PACK||PCV/PACK||PCKE/PACK||PCH/PACK(“||”表示或的关系),化简后得到较为清晰简洁的状态机,如图 8 所示.

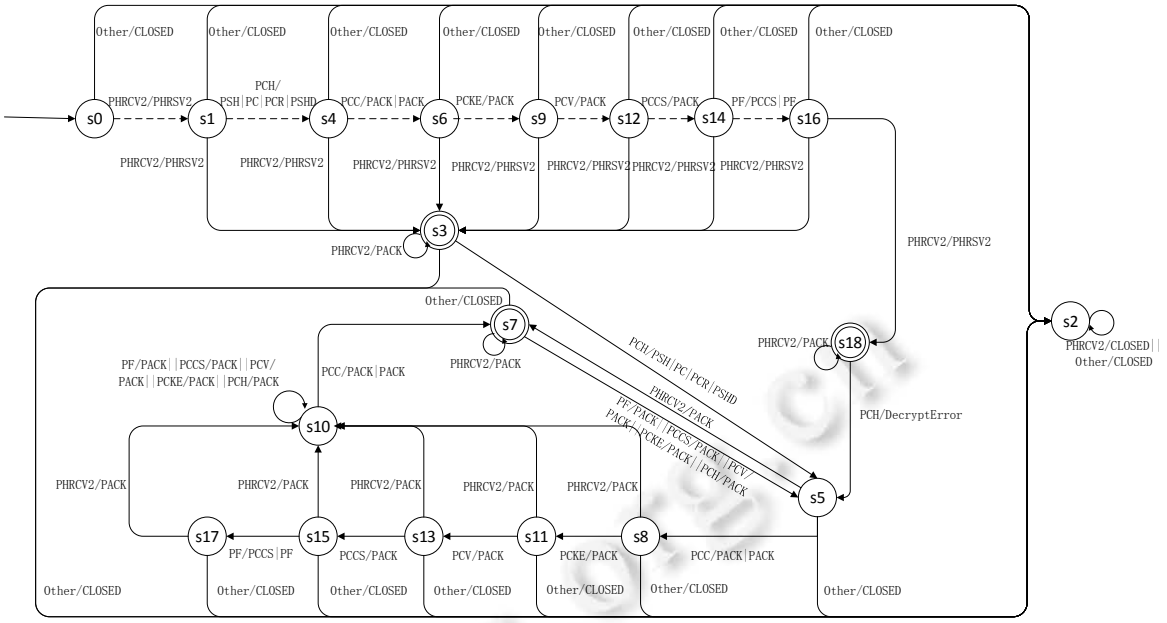


Fig.8 The simplified state machine

图 8 化简状态机

2.4 时间压缩模型

由本文第 2.3 节中推断并化简后的状态机可以看出:通过模型学习得到的 OpenVPN 系统状态机,除了正确路径外还存在相似行为路径,这说明状态机在一定程度上可能存在状态及路径的近似等价.由于状态机主要是描述对象在它的生命周期内所经历的状态序列,因此协议状态机也具有时间特性.我们考虑将状态机在时间轴上进行压缩,对比其状态及路径的近似等价部分的差异性.

状态融合技术是状态机推断相关研究中的一部分重要内容.正则语言推断中的 RPNI 算法[18]可根据样本集构建初始状态机,并不断融合冗余状态,最终推断出与给定样本相一致的有限状态自动机.但该算法在检测候选状态融合正确性时会产生大量的无效主动推断测试请求.因此,Lang 等人[19]提出了依据状态之间相似度对候选状态排序的解决方案 Blue-Fringe 算法.本文借鉴王辰等人[20]对原始的 Blue-Fringe 算法相似度规则进行扩展的方法,提出状态机时间压缩模型.

由于协议实现都是根据输入消息进行响应的网络交互系统,而协议状态机描述了协议实体间的消息序列及状态迁移,因此特别适合使用确定型的 Mealy 机来形式化描述协议状态机模型[21].

定义 1. 协议状态机模型定义为一个六元组 $M=(Q,I,O,\delta,\lambda,q_0)$,其中,$Q$ 为非空有限状态集,I 为有限输入符号集,O 为有限输出符号集,$q_0\in Q$ 为初始状态,δ:$Q\times I\to Q$ 为状态迁移函数,λ:$Q\times I\to O$ 为输出函数.

针对协议系统是对一系列通信报文序列进行交互处理的特点,将定义 1 中的状态转移函数δ和输出函数λ的输入从单个符号 $i\in I$ 扩展至符号序列 $w\in I^*$,对应的输出也为迁移状态序列$\mathbb{Q}\in O^*$和输出符号序列$\mu\in O^*$.例如:对于状态 $q_1\in Q$,输入字符序列 $w=i_1\ldots i_k\in I^*$,输出符号序列 $\mu=\lambda(q_1,w)=o_1o_2\ldots o_k$,中间经历的迁移状态 $\mathbb{Q}=q_2\ldots q_{k+1}\in O^*$.

定义 2. 状态后缀定义为一个输入/输出符号序列集 $L=\{l_1,l_2,\ldots,l_n\}$,输入/输出符号序列定义为 $l=i_1/o_1i_2/o_2\ldots i_k/o_k$,其中,$i_n/o_n\in I/O$.

状态后缀是从该状态出发的所有迁移路径上的 I/O 序列的集合,其代表着该状态对不同的输入序列所响应的输出序列,是该状态可能具有的不同行为的集合.

定义 3. 两个状态之间的相似度为其最长共同后缀的长度.

根据定义 2 和定义 3,状态后缀描述了一个状态在协议系统中后续可能出现的所有行为特征.相似度就是两个不同状态最长的相同后续 I/O 序列的长度值,描述了两个状态后续行为特征集合中最为相似的行为路径的相似程度.例如图 9 所示,*S*1 与 *S*2 的最长共同后缀为 $I_4/O_4|I_5/O_5$("|"表示序列符号之间的连接关系),相似度为 2.

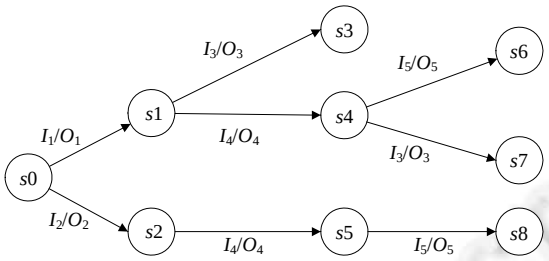


Fig.9 Example of similarity calculation

图 9 相似度计算示例

根据定义 3 可知:相似度可以作为一种衡量标准,表明两个不同状态后续行为的近似程度.相似度越高,两个状态后续的行为集合中就具有越相似的行为.因此,本文考虑压缩协议状态机在时间轴上的近似等价行为,提出时间压缩算法.设定相似度阈值变量 *depth* 反映对近似等价状态合并的严苛程度,*depth* 值越大,满足合并条件的状态数越少,说明要求能够合并的状态近似等价程度越高;反之,近似程度越低.

算法 1. 时间压缩算法.

输入:推断得到的状态机模型 *SM*,相似度阈值 *depth*;

输出:经过状态融合后得到的 SM^*.

1. **for each** s_i, s_j **in** *SM* **do**
2. *ComputeSimilarity*(s_i,s_j); //计算状态两两之间的相似度
3. **end for**
4. **if** *GetMaxValue*(*similaritymatrix*)>=*depth* **then** //相似度最大值不小于阈值
5. (s_1,s_2)=*MaxValuePair*(*similaritymatrix*);
6. *Merge*(s_1,s_2); //取相似度最高的两个状态融合
7. *ChildState*={*child*(s_1/s_2)}; //得到两个融合状态后续状态的集合
8. **for each** $childstate_m$, $childstate_n$∈*ChildState* **and not in** *dirtyset* **do** //对后缀状态及路径进行处理
9. **If** *path*(*to_childstate*$_m$).*input*==*path*(*to_childstate*$_n$).*input* **then**
10. *dirtyset*=*dirtyset*∪{$childstate_m$,$childstate_n$}; //遇到过的状态加入已处理集合
11. *ProcessPath*(*path*(*to_childstate*$_m$),*path*(*to_childstate*$_n$)); //合并迁移路径
12. *Merge*($childstate_m$,$childstate_n$); //融合后继状态
13. *ChildState*={*child*($childstate_m$/$childstate_n$)};
14. **goto** 8;
15. **end if**
16. **end for**
17. $SM=SM^{\#}$ //$SM^{\#}$为当前轮合并结束后得到的类状态机模型
18. **goto** 1;
19. **end if**
20. **return** SM^*

在该算法中需要注意:

- *S*0 为起始状态,不考虑与其他状态的压缩;

- 因状态机存在循环,因此在计算共同后缀时,若遇到已经计算过的状态节点,则其后缀长度不再增加;
- 当两个不同状态经过相同的路径长度后转移到同一状态时,计算的共同后缀长度不再增加;
- 该算法旨在针对状态机时间特性进行压缩,在每轮融合当前状态后,处理后缀迁移路径及状态节点时,只根据输入符号判断是否可合并,从而可能存在一条状态迁移上输入相同输出不同的情况,因此最后得到的是类状态机模型而非严格的 Mealy 机;
- 虽然 *depth* 值越高,合并后不改变其他状态机特性的可能性越大,但由于协议实现过程中的行为路径往往与期望行为路径存在一定差异,*depth* 值过高会导致对特别行为的容忍度降低,可能会出现合并不完全的情况.基于以上考虑,本文取 *depth* 值为 3.

以本文第 2.3 节中推演化简得到的状态机作为初始状态机 *SM*,按照算法 1 对化简状态机进行时间压缩后的模型如图 10 所示.

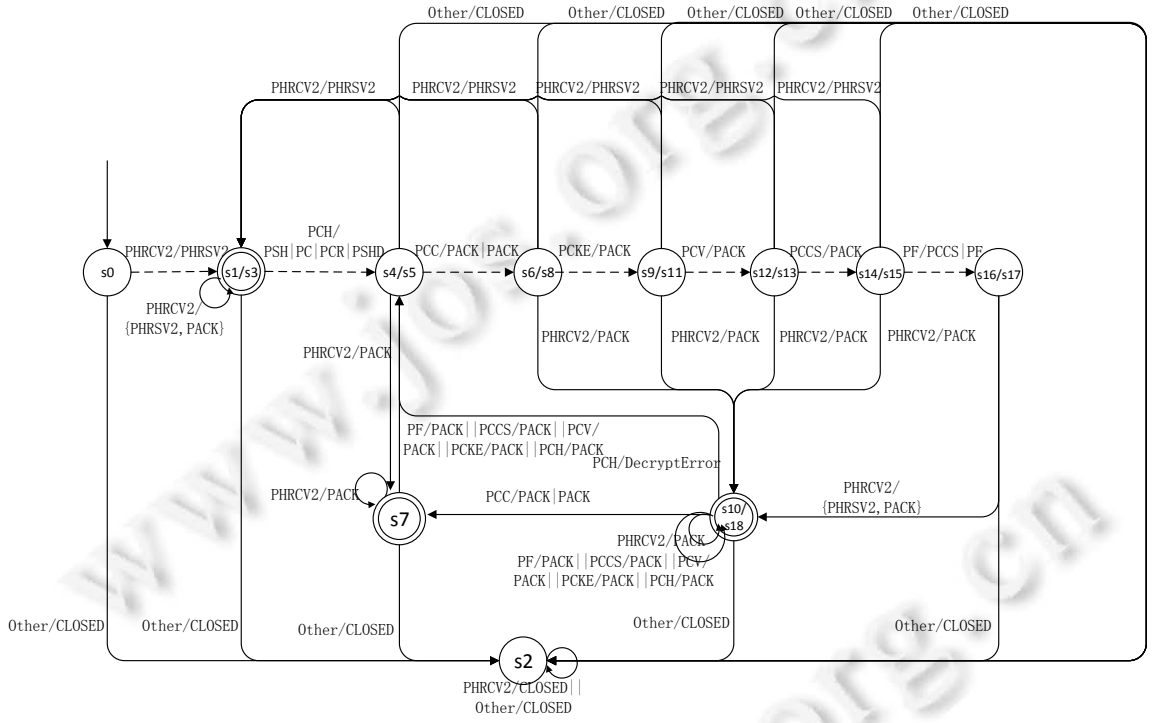


Fig.10 Model of the simplified state machine after time compression

图 10 对化简状态机时间压缩后的模型

3 OpenVPN 状态机脆弱性分析

基于模型学习的状态机推断方法通过主动询问的方式,确保了推断结果的正确性,并且基于主动学习型算法 L^*[15]可以保证推断的状态机是最小且完备的[1].在此基础上,我们对化简状态机与经过时间压缩后的模型进行进一步分析.

3.1 对化简后状态机分析

我们将状态机中的状态变迁称为 OpenVPN 系统的行为.由图 8 可以看出,推演并化简后的 OpenVPN 系统状态机中除了期望的正常行为外,还存在一些比较特别的状态与行为.

- 期望的行为

状态迁移路径:*S*0→*S*1→*S*4→*S*6→*S*9→*S*12→*S*14→*S*16,如图 8 中的虚线所示.

由初始状态 *S*0 开始,经过 OpenVPN 连接请求与响应消息交互 PHRCV2/PHRSV2 到达 *S*1 后,开始协商 TLS 加密隧道:通过 PCH/PSH|PC|PCR|PSHD 到达 *S*4("|"代表连续的数据包),PCC/PACK|PACK 到达 *S*6(由于 PCC 消息包含了客户端证书,消息内容较长将分割为多个数据包在网络中传输,因此对其确认的数据包也为多个),PCKE/PACK 到达 *S*9,PCV/PACK 到达 *S*12,PCCS/PACK 到达 *S*14,PF/PCCS|PF 到达 *S*16,此时加密隧道建立完成,之后开始以密文进行子密钥协商.

可以发现,*S*16→*S*18→*S*5→*S*8→*S*11→*S*13→*S*15→*S*17 也是一条成功的加密隧道建立路径.

- 特别的行为

(1) 服务器在收到两次 PHRCV2 连接请求后,从第 3 次开始不再回应 PHRSV2 消息,而是基于 OpenVPN 协议的确认机制返回 PACK 消息,因此会形成自循环,如 *S*3,*S*7,*S*18,图 8 中双圆节点.这是由于 SSL 握手协议需要一个可靠的下层,从而采用确认机制;

(2) 由 *S*7→*S*5 这个状态迁移的过程可以看出:在完成两次加密隧道建立的情况下,发送 TLS 握手消息都转移到状态 *S*5,此状态可继续顺利完成握手,但双方不再需要进行 TLS 握手的 ClientHello,ServerHello 等消息交换;

(3) *S*1(或 *S*4,*S*6,*S*9,*S*12,*S*14)→*S*3→*S*5→*S*8→*S*11→*S*13→*S*15→*S*17 也是成功的隧道连接路径,这正是由于特别行为 1 允许服务器端在收到重复的客户端连接请求 PHRCV2 后,对服务器端响应两次 PHRSV2 以确保可靠传输;

(4) 在非正常数据流中,服务器端会对客户端的乱序数据包给予一定程度的确认响应,如经过 PCH/PACK,PF/PACK,PCCS/PACK,PCV/PACK,PCKE/PACK 等消息对的交互,但当前状态并未发生改变,如 *S*10;

(5) 由于网络传输数据包大小有一定的限制,因此存在消息拆分与组装问题,这就造成了出现类似 PCC/PACK|PACK 这样的消息对,即,客户端发送的证书被拆分为两个数据包,相应地也就得到了两个服务器端的响应消息;

(6) *S*18→*S*5 的状态迁移上出现了 PCH/DecryptError,这是因为在 *S*18 的前一个状态 *S*16 已经完成了隧道建立,此时再次发送 PHRCV2 时会重新请求协商加密隧道,但 test harness 认为仍在加密隧道中而进行解密,从而出现解密失败.

综合以上分析可以看出,OpenVPN 系统实现状态机的特别行为可能导致攻击路径的存在.例如,依据 RFC 2246 可知:传统的 ClientHello,ServerHello 消息格式中分别包含了客户端和服务器端产生的随机数 random,而在特别行为 2 中,完成两次加密隧道建立后,再次握手重连时不再进行 ClientHello,ServerHello 消息交互,这与规范的 TLS 握手过程不同,导致至少通信双方的随机数没有更新.另一方面,由于 OpenVPN 缺乏详细的官方规范,前期 Novickis[11]在 2016 年做过相关研究,但其最终也并未实现推断出 OpenVPN 实现状态机的目标,因此本文推断状态机发现的期望行为和特殊行为,为 OpenVPN 安全研究提供了参考依据.

3.2 与期望的状态机对比

与第 1 节基础知识中借鉴的 Novickis 给出的期望的状态机[14]作比较可以发现:本文测试得到的 OpenVPN 实现的状态机具有 18 个不同状态,与期望的状态机相比更为复杂;真实实现的状态机完成隧道建立的路径不止一条,而期望的状态机未考虑 OpenVPN 确认机制带来的影响;真实实现的状态机在任何时候都接受 PHRCV2 消息重新建立连接,但期望的状态机只在起始状态接受该连接请求并相应;由于本文测试目标为 OpenVPN 协议的客户端实现,无法测得服务器端发送 P_CONTROL_SOFT_ RESET_V1 消息请求与客户端进行重协商的过程,因此不存在期望的状态机中 *S*7→*S*8 的状态迁移.

综上所述,虽然真实测得的状态机较为复杂,但其完整显示了协议实现的重要过程,且根据状态机路径能够得到一些协议实现的具体细节,如服务器端对 OpenVPN 连接请求响应两次后转为 ACK 确认.这对于类似缺少协议规范但应用广泛的安全协议的分析具有重要的参考意义.

3.3 与TLS状态机比对

由于 OpenVPN 安全性是以 SSL/TLS 加密隧道为基础的,因此将推断得到的 OpenVPN 2.0.9 系统状态机与其对应的 OpenSSL 1.0.2g 系统状态机进行对比分析.基于 Statelearner 开源框架推断出 OpenSSL 1.0.2g 的状态机如图 11 所示.由于 SSL/TLS 协议是基于 TCP 的可靠连接,握手过程不需要确认机制,因此会出现在发送 ClientKeyExchange,ChangeCipherSpec 消息时没有收到服务器端的响应(记为 Empty),但状态已发生迁移的情况.另一方面,由于 TLS 协议中用于检测连接是否保持的心跳请求数据包只有在握手阶段完成后才会得到响应,且其会导致出现无限状态机模型[4],会对本文的协议脆弱性分析工作造成困难,且对安全连接阶段的检测分析没有影响,因此在输入表中将其删除.

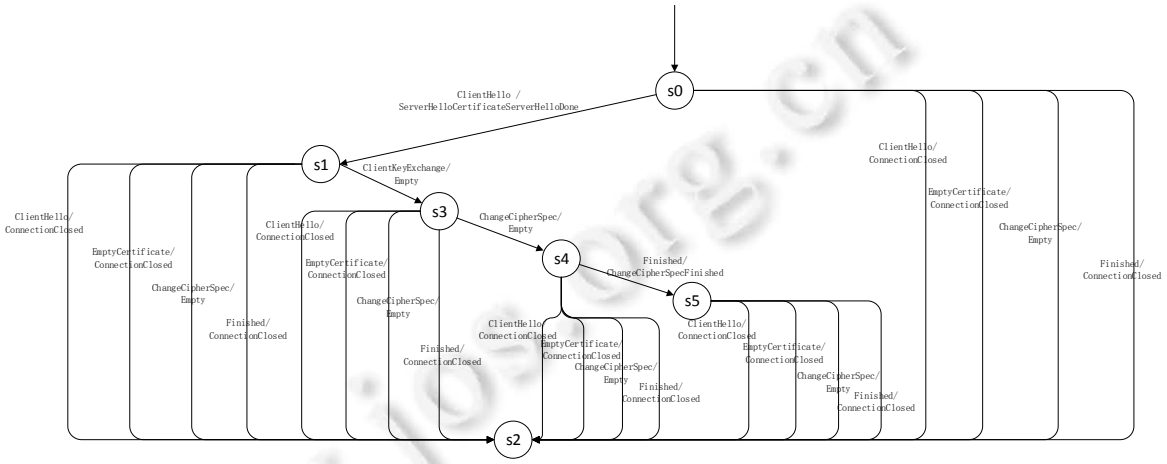


Fig.11 Server side state machine of OpenSSL 1.0.2g

图 11 OpenSSL 1.0.2g 服务器端状态机

由图 8 与图 11 对比可知:虽然 OpenVPN 基于 OpenSSL 建立加密隧道进行安全传输,但其由于特有的确认机制以及开发者对于重链接的具体实现,导致其具体实现的状态机较为复杂,且拥有不止一条成功路径,这使得 OpenVPN 通信双方在后续的数据加密密钥协商及数据加密过程中都存在一定的安全隐患.

由图 10 与图 11 对比可知:通过对 OpenVPN 实现状态机进行时间压缩,期望中应该是相对应版本 OpenSSL 实现状态机的扩充(如增加 OpenVPN 连接请求与响应等),且能很清晰地看出不同轮次连接建立中的差异.如 *S*1/*S*3 自循环与 *S*16/*S*17→*S*10/*S*18,这两条状态迁移就是由于加密隧道建立过程中因为连接请求次数不同而响应不同,符合第 3.1 节中描述的特别行为 1.除此之外,还存在 *S*7 这样特殊的状态,其产生的原因正是 OpenVPN 协议对消息拆分产生多次响应,与分析的特别行为 5 一致.

3.4 对时间压缩模型分析

图 10 显示了对化简状态机进行时间压缩后的模型中共有 11 个状态节点、36 条不同的状态迁移.与图 8 化简状态机 18 个状态节点、50 条迁移相比更加精简.但需要注意的是,采用时间压缩算法得到的模型并不是 Mealy 机.这是由于在时间轴上进行了压缩,可能出现某一状态的状态迁移上输入相同、输出不同,却迁移至同一状态的情况.如图 10 中的由原状态 *S*1 和 *S*3 合并得到的节点,其输入 PHRCV2 消息后,在不同时间点可能得到的响应为 PHRSV2 或 PACK.

- 时间压缩模型对期望的系统行为的刻画

图 10 中虚线所示路径即为期望中的加密隧道建立路径:

*S*0→*S*1/*S*3→*S*4/*S*5→*S*6/*S*8→*S*9/*S*11→*S*12/*S*13→*S*14/*S*15→*S*16/*S*17,说明该时间压缩模型能够表示推断并化简得到的状态机图中的期望路径.

- 时间压缩模型对系统特别行为的刻画

(1) 对于不同轮次的连接请求 PHRCV2,时间压缩模型通过状态合并以及采用集合{PHRSV2,PACK}表示可能收到的响应消息,可以清楚地展示出 OpenVPN 系统服务器端对 PHRCV2 消息在不同时间点响应是具有差异性的;

(2) 双圆节点的自循环行为以及 *S*10/*S*18 节点反映了 OpenVPN 系统的确认机制;

(3) *S*7,*S*10,*S*18 等具有特别行为的状态经过时间压缩算法后仍能够很好地保持其特征.

由上面的分析可以看出来:经过时间压缩后的状态机模型合并了对协议进行安全性分析没有影响的近似等价的迁移路径与状态,与原状态机相比更加清晰且能够充分反映出原模型的各种特征——期望的路径与特殊的路径.因此,通过对状态机时间压缩模型所进行的分析,是寻找协议实现中可能存在的攻击路径的一种简单可行的方法.

4 结 语

本文主要研究了对网络安全协议的实现逻辑脆弱性进行自动化分析的问题.基于模型学习的方法,对 OpenVPN 系统实现逻辑进行黑盒测试分析,自动推演出系统的实现状态机,发现了多条期望路径外的特别路径及可能的安全隐患,为目标系统的脆弱性分析和攻击路径发现提供依据,提出了针对协议实现状态机的时间压缩模型及算法,提高了脆弱性分析和攻击路径发现的效率.本文研究成果为大型应用的安全协议的脆弱性分析提供了理论和技术支持.

References:

[1] Gascon H, Wressnegger C, Yamaguchi F, *et al*. Pulsar: Stateful black-box fuzzing of proprietary network protocols. In: Proc. of the Security and Privacy in Communication Networks. Springer Int'l Publishing, 2015. 330−347. [doi: 10.1007/978-3-319-28865-9_18]

[2] Ma R, Wang D, Hu C, *et al*. Test data generation for stateful network protocol fuzzing using a rule-based state machine. Tsinghua Science & Technology, 2016,21(3):352−360. [doi: 10.1109/tst.2016.7488746]

[3] Kang J, Park JH. A secure-coding and vulnerability check system based on smart-fuzzing and exploit. Neurocomputing, 2017,256: 23−24. [doi: 10.1016/j.neucom.2015.11.139]

[4] Ruiter JD, Poll E. Protocol state fuzzing of TLS implementations. In: Jung J, ed. Proc. of the Usenix Conf. on Security Symp. Berkeley: USENIX Association, 2015. 193−206.

[5] Beurdouche B, Bhargavan K, Delignat-Lavaud A, *et al*. A messy state of the union: Taming the composite state machines of TLS. In: Butler K, ed. Proc. of the Security and Privacy. Piscataway: IEEE, 2015. 535−552. [doi: 10.1109/sp.2015.39]

[6] Ruiter JD. A tale of the OpenSSL state machine: A large-scale black-box analysis. In: Proc. of the Secure IT Systems. Springer Int'l Publishing, 2016. 169−184. [doi: 10.1007/978-3-319-47560-8_11]

[7] Verleg P, Poll E, Vaandrager FW. Inferring SSH state machines using protocol state fuzzing [MS. Thesis]. Nijmegen: Radboud University Nijmegen, 2016.

[8] Somorovsky J. Systematic fuzzing and testing of TLS libraries. In: Ahn GJ, Yung M, Li N, eds. Proc. of the ACM SIGSAC Conf. on Computer and Communications Security. New York: ACM Press, 2016. 1492−1504. [doi: 10.1145/2976749.2978411]

[9] Lenaerts T, Vaandrager F, Poll E. Improving protocol state fuzzing of SSH [Bachelor Thesis]. Nijmegen: Radboud University Nijmegen, 2016.

[10] Veldhuizen B. Automated state machine learning of IPsec implementations [Bachelor Thesis]. Nijmegen: Radboud University Nijmegen, 2017.

[11] Novickis T. Protocol state fuzzing of an OpenVPN [MS. Thesis]. Nijmegen: Radboud University Nijmegen, 2016.

[12] Angluin D. Learning regular sets from queries and counterexamples. Information & Computation, 1987,75(2):87−106. [doi: 10.1016/0890-5401(87)90052-6]

[13] Isberner M, Howar F, Steffen B. The TTT algorithm: A redundancy-free approach to active automata learning. In: Kifer M, ed. Proc. of the Int'l Conf. on Runtime Verification. Cham: Springer-Verlag, 2014. 307–322. [doi: 10.1007/978-3-319-11164-3_26]

[14] Chow TS. Testing software design modeled by finite-state machines. IEEE Trans. on Software Engineering, 2006,SE-4(3):178–187. [doi: 10.1109/tse.1978.231496]

[15] Raffelt H, Steffen B, Berg T. LearnLib: A library for automata learning and experimentation. In: Baresi L, Heckel R, eds. Proc. of the Int'l Conf. on Fundamental Approaches to Software Engineering. Berlin: Springer-Verlag, 2006. 377–380. [doi: 10.1007/11693017_28]

[16] Isberner M, Howar F, Steffen B. The open-source LearnLib. In: Proc. of the Int'l Conf. on Computer Aided Verification. Cham: Springer-Verlag, 2015. 487–495. [doi: 10.1007/978-3-319-21690-4_32]

[17] Xiao B. Design and implementation of protocol conformance test system [MS. Thesis]. Beijing: Beijing University of Posts and Telecommunications, 2015 (in Chinese with English abstract).

[18] Oncina J, Garcia P. Inferring regular languages in polynomial updated time. In: Sanfeliu A, Blanca NPDL, Vidal E, eds. Proc. of the Pattern Recognition and Image Analysis: Selected Papers from the IVth Spanish Symp. Singapore: World Scientific, 1992. 49–61. [doi: 10.1142/9789812797902_0004]

[19] Lang KJ, Pearlmutter BA, Price RA. Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In: Honavar V, Slutzki G, eds. Proc. of the Int'l Colloquium on Grammatical Inference. Berlin: Springer-Verlag, 1998. 1–12. [doi: 10.1007/bfb0054059]

[20] Wang C, Wu LF, Hong Z, *et al*. Method of protocol state machine inference based on state merging. Journal of PLA University of Science and Technology (Natural Science Edition), 2015,(4):322–329 (in Chinese with English abstract). [doi: 10.7666/j.issn.1009-3443.201409019]

[21] Pan F, Wu LF, Hong Z, *et al*. Network Protocol Reverse Analysis and Application. Beijing: National Defend Industry Press, 2016. 229–238 (in Chinese).

附中文参考文献:

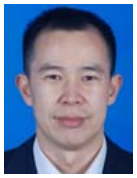
[17] 肖冰.协议一致性测试系统的设计与实现[硕士学位论文].北京:北京邮电大学,2015.

[20] 王辰,吴礼发,洪征,等.一种基于状态融合的协议状态机推断方法.解放军理工大学学报:自然科学版,2015,(4):322–329. [doi: 10.7666/j.issn.1009-3443.201409019]

[21] 潘璠,吴礼发,洪征,等.网络协议逆向分析及应用.北京:国防工业出版社,2016.229–238.

申莹珠(1989－),女,陕西西安人,硕士生,主要研究领域为网络信息安全.

张协力(1992－),男,博士生,主要研究领域为网络密码,安全协议.



顾纯祥(1976－),男,博士,教授,博士生导师,主要研究领域为密码学.

卢政宇(1993－),男,硕士生,主要研究领域为网络空间安全.

陈熹(1988－),男,讲师,主要研究领域为网络信息安全.

附录 1.