

软件定义网络数据平面安全综述

郭中孚¹, 张兴明¹, 赵博¹, 王苏南²

(1. 国家数字交换系统工程技术研究中心, 河南 郑州 450002;

2. 深圳职业技术学院, 广东 深圳 518000)

摘要: 软件定义网络将数据平面与控制平面解耦, 旨在更快地引入网络创新, 并从根本上实现大型网络的自动化管理。架构创新带来了挑战与机遇, 安全问题限制了软件定义网络的广泛采用。针对数据平面的攻击可能会损毁整个软件定义网络, 首先介绍了数据平面结构与发展趋势; 然后分析了数据平面安全风险, 指出漏洞, 确定潜在的攻击场景, 并给出2种具体解决方案, 讨论其意义与局限性; 最后展望未来的安全研究方向。

关键词: 软件定义网络; 数据平面; 有状态数据平面; 软件定义网络安全; 数据平面安全

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2018087

Survey of software-defined networking data plane security

GUO Zhongfu¹, ZHANG Xingming¹, ZHAO Bo¹, WANG Sunan²

1. National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China

2. Shenzhen Polytechnic, Shenzhen 518000, China

Abstract: The software-defined network decouples the data plane from the control plane, aiming to introduce network innovation faster and fundamentally automate the management of large networks. Architecture innovation brings challenges and opportunities. Security issues limit the widespread adoption of software-defined networks. Attacks on the data plane may damage the entire software-defined network. The data plane structure and development trends were introduced, data plane security risks were analyzed, vulnerabilities were pointed out, and potential attack scenarios were identified. It also presents two specific solutions, discusses the significance and limitations, and looks forward to future security research directions.

Key words: software-defined network (SDN), data plane, stateful SDN data plane, SDN security, data plane security

1 引言

传统的互联网协议网络以硬件为中心, 网络中底层设备配置管理需使用设备供应商的特定命令, 繁杂耗时, 此外, 数据平面与控制平面的捆绑, 使龙头厂商的垄断地位坚不可破, 严重阻碍

了网络基础设施的发展; 云计算、物联网等新技术使联网设备数量爆炸式增长, 从发展趋势看, 传统网络无法应对需求。软件定义网络(SDN, software defined network)是一种新兴的网络架构, 控制与数据平面分离。SDN有逻辑上集中但物理上分布的控制机制, 监控整个网络, 并对数

收稿日期: 2018-08-24; 修回日期: 2018-09-10

通信作者: 郭中孚, gzf0309@mail.dlut.edu.cn

基金项目: 高安全等级网络基础设施关键装备核心芯片及软件研发基金资助项目(No.2017ZX01030301)

Foundation Item: High Level Security Network Infrastructure Key Equipment Core Chip and Software Development Fund Funding Project (No.2017ZX01030301)

据平面中转发设备的分组转发做出决策。数据平面只是简单的转发元素,没有嵌入式智能来自主决策,因此易引发诸多全新的安全问题。本文旨在研究如何保护 SDN 平台本身,重点关注 SDN 数据平面相关安全问题。

2 SDN 数据平面

2.1 SDN 架构

SDN 的核心架构如图 1 所示,分为 3 层。顶层应用平面定义规则并提供不同服务(如负载均衡、流量监控、入侵检测/入侵防御、服务质量等)。应用平面通过北向应用程序设计接口(API, application programming interface)向 SDN 控制平面分发策略。第二层控制平面是网络拓扑的抽象。控制器负责建立流量表 and 数据处理策略,通过南向 API 抽象网络细节,并保持最新的网络整体视图。OpenDaylight^[1]和开放式网络操作系统(ONOS, open network operating system)^[2]实现分布式控制,多个控制器之间通过东向/西向 API(如 HyperFlow^[3])交换数据平面中的流量控制信息。最底层称为数据平面,提供网络设备,如真实或虚拟的交换机、路由器和接入点。

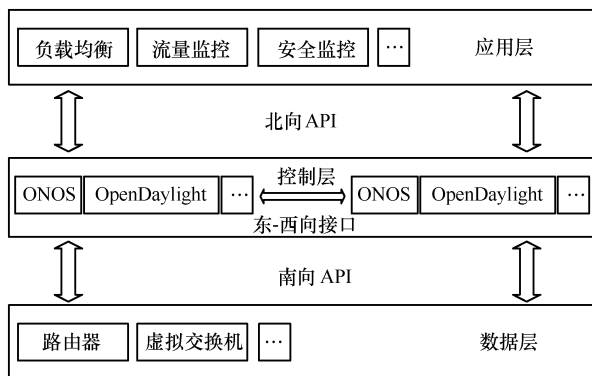


图 1 SDN 的核心架构

2.2 数据平面

数据平面由网络设备组成,如专门用于分组转发的交换机和路由器。与传统网络不同,这些只是简单的网元设备,没有嵌入式智能来自主决策。这些设备通过标准的 OpenFlow 接口与控制器进行通信,确保了不同设备之间的配置和通信兼容性以及互操作性。OpenFlow 转发设备具有转发表,它由 3 部分组成:① 规则匹配;② 为匹

配分组而执行的操作;③ 用于统计匹配分组数量的计数器。规则匹配字段包括交换机端口、源媒体访问控制(MAC, media access control)地址、目标 MAC 地址、以太网类型、虚拟局域网(VLAN, virtual local area network) ID、源 IP、目标 IP、传输控制协议(TCP, transmission control protocol)源端口、TCP 目标端口。流规则可以定义为这些字段的组合。最常见的操作包括:将数据分组转发到传出端口、封装并转发给控制器、丢弃、排队、修改字段等。最常见的情况是安装一个默认规则指示交换机将数据分组转发给控制器做出决定。

2.3 数据平面研究的最新进展

最初的 OpenFlow 标准过于严格,出现了各种提案,增加了额外的灵活性。它们可以分为 3 个不同的方向:① 转发设备添加多个流表;② 改进的匹配规则;③ 定制的 OpenFlow 扩展。

2.3.1 将多个流表添加到转发设备

最初在 OpenFlow 1.0^[4]中提出的单个流表模型会导致 2 个主要问题。首先,它会迫使开发者部署 OpenFlow 规则,这些规则结合了所需匹配的笛卡尔积,这种指数增长的部署规则将导致流量表容量使用效率非常低,通常需要昂贵且实际受限的三态内容寻址存储器(TCAM, ternary content addressable memory)来实现。其次,许多应用程序自然受益于两阶段处理模型^[5],其中,在第一阶段首先基于某些分组特征对分组进行标记,然后进行下一步的匹配和处理。因此,流水线流表是针对早期 OpenFlow 版本的一项重要改进,其通过一系列流表定义如何处理数据分组,可以显著节省 TCAM 内存,同时拥有更简洁的配置。

在标准化进程的同时,研究界提出了更进一步的硬件解决方案,以实现更灵活的多重匹配。此方面的突出成就是 2013 年提出的可重构匹配表(RMT, reconfigurable match table)^[6],该设计通过同一物理 TCAM 阵列非常灵活地映射任意数量的具有不同宽度和深度(逻辑)的匹配表,通过先前匹配计算或提取的参数定义匹配规则。

2.3.2 提高匹配规则的灵活性

不同类型网络应用程序操作显然要与不同类

型的匹配相关联,在最初的 OpenFlow 版本 1.0 中仅限于 12 个匹配字段, OpenFlow 版本 1.3.4^[7]已经支持 40 个匹配字段,最近 OpenFlow 版本 1.5.1^[8]中此数字增为 45,其中包括细粒度的匹配,如 TCP 标志等。2013 年,协议不经意转发(POF, protocol oblivious forwarding)提案^[9]通过<offset, length>组从数据分组头字段提取任意位的匹配,灵活性显著增强。

2.3.3 定制的 OpenFlow 扩展

从 OpenFlow v1.0 至今已提出多种 OpenFlow 协议的扩展,以适应普通流表或 OpenFlow 操作无法处理的特定需求。组表是 OpenFlow 扩展最显著的例子,它引入了有状态操作支持。例如, OpenFlow v1.0 中,当链路/端口发生(物理)故障时, OpenFlow 交换机会请求遥远的控制器来实例化一个新的转发规则(例如,故障链路/端口处理的所有流量由备用链接转发)。但是,控制器需要一些时间才能做出反应,因此这段时间内,所有发往故障端口的数据分组都将丢失(而在大型网络中,该数量可能非常庞大:对于 10 Gbit/s 链路上的控制器,100 ms 的响应时间将导致多达 3×10^7 个数据分组丢失)。为解决这个问题,新版本 OpenFlow 标准引入了一种快速故障转移技术。一个快速故障转移组包含多个操作集合,这些操作按照故障转移组定义的顺序进行评估,直到选出一个可用的方案。尽管解决了重要的问题,但快速故障转移不仅是最新的 OpenFlow 规范中的备选项,且关键细节的实施不在 OpenFlow 规范内。有状态 OpenFlow 扩展的其他重要例子如速率控制仪表、用于支持学习型功能的同步表格等,详细信息见 OpenFlow 版本 1.5.1^[8]。

2.4 数据平面的发展趋势

如 2.3 节所讨论的, OpenFlow 标准的发展证实在交换机内引入一些有状态的操作是必要的,以减少远程控制器实时参与(explicit involvement)带来的过度延迟。本节所讨论的内容也是当前数据平面的主要发展趋势,普遍被称为有状态的数据平面(stateful data plane)^[10,12,13]。

一般而言,有状态的 SDN 数据平面提议有 2 个主要原则:① 使流的状态信息保存在转发设备内部,允许以编程的方式实例化分组级别的状态转化;

② 赋予交换机做转发状态更新的权利,基于本地流的状态信息而不需要同控制器进行通信。

主流的有状态数据平面核心平台有 OpenState、流级状态转换(Fast, flow-level state transitions)和状态数据平面抽象(SDPA, stateful data plane abstraction),3 个平台中使用的表格如图 2~图 4 所示,每个表格都被表示为一个矩形,包含表格名称和用“|”分隔的相应表格项目。

2.4.1 OpenState

OpenState^[10]通过扩展有限状态机(XFSM, eXtended finite state machine)向 SDN 的数据平面引入了可编程性。OpenState 的工作最近在文献[11]中得到了进一步的扩展,更加全面地支持 XFSM(与原始的 Mealy 状态机不同)。

OpenState 每个转发设备都有 2 个独立的表:状态表和 XFSM 表(如图 2 所示)。前者存储接收到的数据分组当前流的状态,这些状态与该数据分组流相关。后者根据数据分组的接收信息来定义规则。XFSM 由 4 元组($S; I; O; T$)建模,其中, S 是包括起始状态 S_0 的一组有限状态; I 是一组有限的输入(事件); O 是一组有限的输出(操作); T 是一个状态转换函数(规则),它将 $\langle \text{state}, \text{event} \rangle$ 与 $\langle \text{state}, \text{action} \rangle$ 相映射。使用 OpenState,转发设备中的数据分组处理是一个 3 步过程。

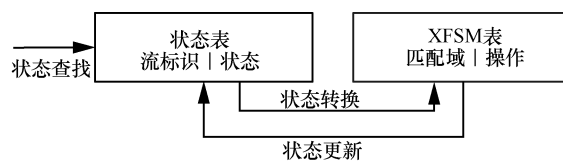


图 2 OpenState 架构中使用的表格

Step1 转发设备接收到数据分组后,在状态表中检索当前状态(它使用流 ID(如源 IP 地址)作为表检索的关键字),将其作为元数据字段附加到数据分组。若状态表查找未匹配,则转发设备会为传入数据分组分配默认状态 S_0 。

Step2 转发查询 XFSM 表以查找与“状态事件对” $\langle \text{state}, \text{event} \rangle$ 匹配的规则,根据 XFSM 表执行下一个状态字段预先定义的相关操作,并更新数据分组的状态字段。

Step3 转发设备基于从对应流 ID 的先前步骤中检索到的下一个状态字段来更新其状态表。

显然, 转发设备无须为每个接收到的分组联系控制器, 可以基于接收到数据分组的当前状态以及控制器对每个<state, event>对的预定义规则进行路由决策。

2.4.2 FAST

与 OpenState 类似, FAST^[12]还在每个转发设备中存储预装的状态机。但在 FAST 中, 每个转发设备可能有多个状态机实例, 每个实例都有特定的应用程序。而且, 数据平面并不是在 OpenState 中使用的 2 个表, 而是实现了包括状态机过滤器表、状态表、状态转换表、操作表 (如图 3 所示)。状态机所有实例共用一个状态机过滤器表, 但每个状态机都有自己的状态表、状态转换表和操作表。状态机过滤表用于选择与数据分组相关对应状态机。状态表是一个散列表, 将每个数据分组头映射到流状态, 每个状态值都存储在一个变量内, 就一个数字而言, 一些高位用于存储状态, 其他位表示相应的值。最后, 操作表根据数据分组头及其新状态指定转发设备应对数据分组执行相应操作。

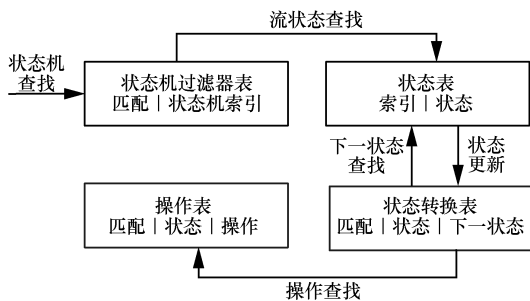


图 3 FAST 架构中使用的表格

FAST 控制平面有 2 个组件: 编译器和转发设备代理。前者是一个脱机组件, 负责将状态机转换为转发设备代理。后者是在线组件:

- ① 管理转发设备内部的状态机;
- ② 基于从转发设备接收到的更新执行某些本地计算;
- ③ 通过局部实现转发设备内部的状态机;
- ④ 处理转发设备与控制器之间的通信, 同时更新控制器关于交换机的本地状态。例如, 如果交换机检测到攻击者, 则更新交换代理, 随后更新控制器。

2.4.3 SDPA

SDPA^[13]由 3 个主表组成, 包括状态表、状态转换表和操作表 (如图 4 所示), 以及状态处理模块, 即转发处理器 (FP, forwarding processor)。状态表有 3 个字段值: 匹配、状态和指令。匹配值可以是数据分组头部字段的任意组合, 状态值是流的当前状态, 指令值可以由状态或数据分组指定。与前两者不同, SDN 控制器与 FP 模块通信并启动状态表。此外, 通过定期或特定的事件更新, 控制器通过与 FP 的通信维护状态表的完整控制和更新信息。通过 SDPA, 在接收到流的第一个分组后, 交换机将分组发送到控制器, 该控制器确定相应流状态应存储的状态表。来自该流的其他输入数据分组将在交换机内本地处理, 而无须与控制器通信。控制器有绝对控制权且能决定何时更新, 如接收 FP 更新 (即定期更新或由于特定事件触发更新) 或控制器决定何时接收更新。

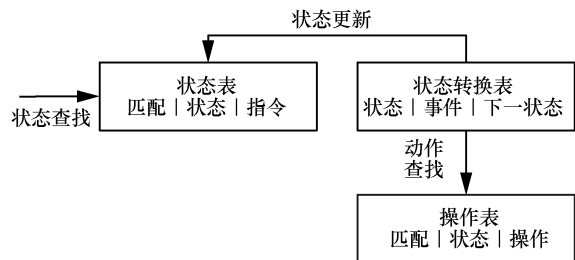


图 4 SDPA 架构中使用的表格

2.5 安全发展态势

当前 SDN 安全研究可以分为 3 个主要方向。

1) 引入已有的安全方案, 并在 SDN 架构内进行改进升级。例如, 在 SDN 中构建可靠防火墙^[14,15], 将防火墙、入侵防御系统 (IPS, intrusion prevention system)、(分布式) 拒绝服务攻击 ((D)DoS, (distributed) denial of service) 探测器、反射器网络等功能基于 SDN 集成起来增强网络安全方案^[16]。或者引入以太网^[17]、信息中心网络^[18]等一些久经检验的成熟技术, 该研究方向极大地推进了初期 SDN 安全的发展。

2) 基于 SDN 特性开发全新的安全服务, 这些服务在 SDN 出现前是无法开展或很难开展的, 如使用网络功能来保护物联网设备^[19]、网络取证^[20]、流量监控^[20], 基于云的环境预防 (D)

DoS 攻击^[21-22]。

3) 保证 SDN 平台本身安全的研究, SDN 平台本身因控制器的存在具有单点脆弱性, 2.4 节所述状态数据平面降低了数据平面对 SDN 控制平面的依赖性, 是现在主流的研究方向, 很多工作基于此展开, 可以开发多种应用程序, 包括状态故障恢复^[23]、HULA^[24]、端口敲门^[25]、SDN 通道状态检测^[26]和用户数据报协议 (UDP, user datagram protocol) 洪泛状态缓解^[27]。随着 SDN 的进一步发展, 其自身虽引入了新的漏洞, 但终将更安全, 详情参考文献[28]。

3 数据平面安全风险

第 2 节中讨论了现有 SDN 数据平面的结构与发展, SDN 优点很多, 但安全性问题成为制约 SDN 广泛应用的关键, 传统网络中转发设备普遍由不同设备商提供, 设备异构性本身也是一种安全防护。在 SDN 架构中有逻辑集中的控制平面, 通过开放、可编程的接口, 基于设备管理协议与 OpenFlow 一起, 对数据平面可编程转发设备进行配置和管理, 所以数据平面易受误配置信息影响。接下来从数据平面漏洞和攻击两方面讨论安全风险。

3.1 漏洞分析

在 OpenFlow 网络交换模型中, 控制器下发流规则到转发设备流表中, 流规则下发可分为 2 类: 被动式和主动式。被动工作模式: 用户发送数据分组后, 转发设备产生 packet-in 消息传递到控制平面, 控制器依据相应的网络状态下发流规则, 转发设备依据接收到的流规则完成转发。主动工作模式: 在用户发送数据报之前, 控制器已经下发流规则到转发设备流表空间, 需考虑转发设备流规则存储以及不同转发设备的流规则不一致等问题。

数据平面拒绝服务攻击。在转发设备流表存储空间有限的前提下, 要保证表项合理使用且不被恶意侵占。在 OpenFlow 协议中, 转发设备需要在控制平面下发相应流规则前缓存数据分组, 因此数据平面易受存储器饱和攻击。流表空间满足正常情况下转发设备工作需要, 但在遭遇 (D)DoS 攻击时, 攻击者产生高速率大容量的数据

分组, 流表空间迅速耗尽, 后续流被直接丢弃, 导致拒绝服务问题。

恶意流规则注入导致流规则不一致, 进而使 SDN 网络不稳定, 使流规则丧失可靠性。数据平面接收控制平面下发的流规则, 挑战在于如何隔离恶意或非正常流规则。转发设备处于监听模式时, 与控制平面连接无须认证, 因此转发设备被恶意控制器控制时流表信息易被篡改, 造成单点失效或信息泄露等问题。当涉及物理分布式控制平面^[29-31], 攻击者影响并强制流规则分发, 从而将网络驱动为不一致的状态, 进而导致控制平面的瘫痪。

在 SDN 转发与控制解耦的架构中, 数据平面基本功能依赖于与控制平面通信, 如果二者失去连接, 没有相应流转发规则的指导, 转发设施构成的节点网络会瘫痪^[32]。因此, 考虑数据平面安全问题, 本文也将分析针对南向 API 的攻击。

3.2 攻击分析

因为针对同一个漏洞有多种不同攻击方式, 无法与上文分析的漏洞一一对应, 本文按数据平面和南向 API 两方面叙述针对 3.1 所述漏洞的攻击。

3.2.1 数据平面

3 种不同的攻击可能被用来损害数据平面, 包括设备攻击、协议攻击和侧信道攻击。

设备攻击指通过 SDN 功能的交换机, 利用其软件或硬件漏洞来危害 SDN 的数据平面。攻击者可以针对转发设备的软件缺陷 (如固件攻击) 或硬件特征 (如 TCAM 存储器)。因 OpenFlow 交换机流量表容量有限, 推理攻击^[33]可以推断网络参数且准确度较高 (流量表容量和流量表使用情况)。

协议攻击是指利用转发设备中的网络协议漏洞 (如针对边界网关协议的攻击) 而展开 SDN 数据平面的攻击。Kloti 等^[34]详细研究了 SDN 中的拒绝服务攻击和信息泄露攻击, 这些攻击因 OpenFlow 的性质而加剧。如文献[35]中所述, 大多数 OpenFlow 交换机具有不同定制功能且在独立交换机固件上实现。例如, HP3800 型交换机不支持 TCAM 流表中对 OpenFlow 所指定 12 位字段组的匹配。交换机固件缺陷会被攻击者滥用导致木桶效应, 降低整体网络性能。恶意应用程序

可以安装针对性的流规则,用上述硬件不支持的匹配字段(MAC 匹配)覆盖现有流规则(IP 匹配)来实现此类攻击。

侧信道攻击是指攻击者通过分析转发设备的性能指标推断出网络的转发策略。例如,攻击者可以通过输入缓冲区来识别规则,进而分析数据分组处理时间,最终识别转发策略^[36]。

3.2.2 南向 API

针对 SDN 的南向 API 有 2 种主要攻击:窃听篡改攻击和(D)DoS 攻击。在窃听篡改攻击中,攻击者窃听控制和数据平面之间交换的信息,进而通过修改数据平面间的消息来破坏网络,如 Benton 等^[37]提出的篡改信息攻击。此类攻击可能存在潜伏期,掌握尽可能多的信息通路,在特定时刻彻底损毁网络。在(D)DoS 攻击中,南向 API 充斥着导致网络策略实施失败的请求,如文献[38]中所述,攻击者可以通过分析探测数据分组并将它们分类来评估延迟时间,从而通过探测数据分组推测 SDN 中的流量规则。已知流量规则的攻击者可以通过发送大量规则匹配的数据分组来触发(D)DoS 攻击,这些数据分组会触发大量 packet-in 数据分组,使控制器不堪重负而瘫痪。

4 数据平面安全措施

数据平面由网络转发设备组成,基础设施受损会导致大量信息泄露,各国情报机构在内的攻击者致力于在其上植入后门,即使是新手黑客也可以通过发送指定数据分组开启后门进而在路由器上执行任意代码^[39]。受损的转发设备可能会恶意分组丢失、网络减速、伪造网络流量等,以发起针对网络运营商及其用户的攻击,甚至取消整个软件定义网络^[40-41],相较传统网络而言,受损转发设备对 SDN 造成的损害要高很多。

保护 SDN 数据平面,要考虑以下 4 方面因素。

1) 现有解决方案不兼容性:用于传统网络的防御机制不适用于 SDN,为了将传统的防御措施引入 SDN 中,需要对 OpenFlow 协议进行基本的重新设计^[42]。

2) 控制平面对数据平面的依赖:SDN 控制器依赖数据平面反馈信息了解整个网络状况,这些未经验证的消息很容易被篡改,即使使用 TLS

身份验证,恶意转发设备也可能发送伪造的欺骗消息来破坏网络的控制器视图。

3) 软件交换机:可编程软件交换机(如 Open vSwitch)运行在主机服务器上,较物理交换机攻击面更大。

4) 有状态的 SDN 数据平面:2.4 节已提及,通过将流状态压入数据平面,使每个交换机都能够进行一定的本地自治和智能决策,从而提高网络的性能(这归功于数据平面和控制平面之间的通信流量减少)。尽管有状态的 SDN 数据平面好处显而易见,但这增大了攻击面,需要考虑状态一致性、流表存储空间等问题。

接下来就保护 SDN 数据层的基本要求出发(4.1 节),并结合有状态数据层讨论最新安全措施(4.2 节),列举 2 种具体的解决方案(4.3 节)。

4.1 解决方案基本要求

随着 SDN 组件的持续改进和变化,保护数据平面转发设备是一项具有挑战性的任务。为保护整个数据平面,需要一种能够自动检测受损转发设备并保护整个网络免受攻击的解决方案,这种检测应该基于转发设备的主要功能(而非任何协议、软件或攻击),要求如下。

扫描:保护机制能够自主优先检查转发设备,高效精准完成扫描任务。

识别恶意行为:为了保护网络,防护机制必须能够将恶意转发设备隔离,快速识别恶意行为尤其重要。

对威胁智能响应:保护机制需可编程,允许管理员指定网络策略且能智能响应。

支持有状态数据平面:必须将数据平面解决方案的最新进展设计为保护机制。

考虑到数据平面是网络基础设施的重要组成部分,所有提出的解决方案在检查和分析阶段都不得破坏网络性能。

4.2 有状态数据平面的安全意义

2.4 节讨论的有状态 SDN 数据平面是现在主流研究方向,因其增加了数据平面的复杂度,扩大了攻击面,体现为以下 3 方面。

流状态内存易受耗尽内存攻击:数据平面的智能性需要对流状态信息进行存储,攻击者可以发动内存耗尽攻击。

无身份认证机制: 控制平面的部分功能在数据平面上独立实现, 需要转发设备之间传递信息, 并在常规流量数据分组内捎带信息, 确保内部通信安全可靠是一个重要问题却很少被提及, 攻击者可能将虚假信息数据分组注入网络中, 更改特定设备的流量状态, 攻击者可以构造虚假的场景, 引发链路故障降低网络性能。

缺乏中央状态管理: 因为不存在管理转发设备内状态同步的中央实体, 分布式系统易发生状态不一致问题, 如接收到数据分组才会触发状态更改, 攻击者可以强制网络转变为状态不一致。

有状态数据平面的智能性使数据平面更加安全, 它可以减缓传统 SDN 中面临的安全威胁, 首先得益于数据平面与控制层间的通信减少, 如控制平面饱和攻击。这种攻击的目的是使控制平面无法工作, 这也是数据平面和控制平面之间频繁互动的结果。攻击者可以通过生成大量不同的流量(使用虚假 IP 地址)来发起控制平面饱和攻击, 这导致数据平面交换机将许多 packet-in 请求转发给控制器, 可能使通信链路或控制器内部资源饱和。尽管研究人员提出了一些数据平面方案^[43-45]来解决这个问题, 但在传统 SDN 环境中, 缓解控制平面饱和仍然是一个公开挑战。有状态的 SDN 设计缓解了这种威胁, 因为它减少了通信次数, 且有状态的数据平面减缓了常在传统 SDN 中出现的 2 个漏洞、流量信息泄露和 TCAM 流表耗尽, 传统 SDN 中攻击者刺探数控平面间的通信信息以获取配置信息(如安装规则的时间开销), 有状态的 SDN 交换机因为自主决定如何处理流量, 减少了依赖于定时信息攻击者的攻击面, 针对 TCAM 流表的攻击也因有状态的 SDN 交换机流表更改次数更少而缓解。

4.3 现有解决方案

制定保护机制需要清晰描述对手模型。本文所考虑的攻击模型是一个可以完全掌控一个甚至全部转发设备的对手。这也是 SDN 数据平面可能存在的最强大的对手。攻击者可以将全部或部分流量随机或选择性丢弃、错误路由、延迟、重新排序, 甚至生成伪造数据分组。

已知 SDN 数据平面需面对不确定的敌对模型^[46], 这将直接限制解决方案的采用。例如, 文

献[47-51]中提出的解决方案假定全部或大部分的转发设备是可信的。本文分析可以实施并在实际条件下进行评估的 SPHINX 和 WedgeTail。接下来, 首先讨论现有解决方案, 接着在 5.1 节和 5.2 节中概述 2 个最突出的解决方案, 并在 5.3 节中讨论它们的局限性。

有关分组转发异常检测的文献可分为 4 类:

① 密码机制; ② 流量统计; ③ 分组探测; ④ 基于确认的机制。这些方法在部署时受到两方面限制: ① 密码操作会显著增加计算开销; ② 需要修改 IP 分组格式。例如, 文献[52-56]等加密方案将签名嵌入分组中, 转发设备通过签名验证分组是否被正确路由转发。另一种加密方案是分析转发设备端口处的流量统计^[57-58]。但流量统计依赖于转发设备间严格的时间同步, 这在大规模网络中很难实现, 且该方法无法防御数据分组篡改攻击。分组探测方法, 如文献[59-61]中方法对探测分组进行采样和分析, 以检测转发异常。这些解决方案大多集中在网络的第一跳和最后一跳异常检测上, 并显著增加通信开销。基于确认的解决方案, 如文献[62-64]方法通过相邻转发设备之间的定期通信检测是否有分组丢失。此时, 由于每个转发设备需存储整个流的转发路径并周期性地收集确认分组, 所以转发设备计算和存储方面开销很大。

4.3.1 SPHINX

SPHINX 是一种检测 SDN 网络拓扑和数据平面转发攻击的框架。SPHINX 也是维护 SDN 数据平面安全的解决方案之一, 它不会假设转发设备是可信的。它通过监控所有 OpenFlow 的消息, 分析消息的功能, 并对所有网络更新进行验证, 是一种实时的解决方案。

SPHINX 体系结构如图 5 所示, SPHINX 利用流程图这一新颖抽象(与实际网络操作非常接近): ① 对网络更新和约束进行增量验证, 从而实时验证网络属性; ② 检测网络拓扑和数据平面转发中已知或未知的安全威胁且不影响性能。它分析 4 类消息(packet-in、stats-reply、features-reply 和 flow-mod)以了解网络拓扑和转发状态, 并为网络中观察到的每个流量构建流图。它不断更新并监控这些流程图中的更改, 当发现可疑变化时

发出警报, SPHINX 利用自定义算法逐步处理网络更新, 实时分析是否会导致异常行为并决定是否授权更新。网络行为通过 SPHINX 策略引擎进行验证。策略引擎使管理员能够验证增量流图。管理员指定的限制条件是使用策略语言编写的。

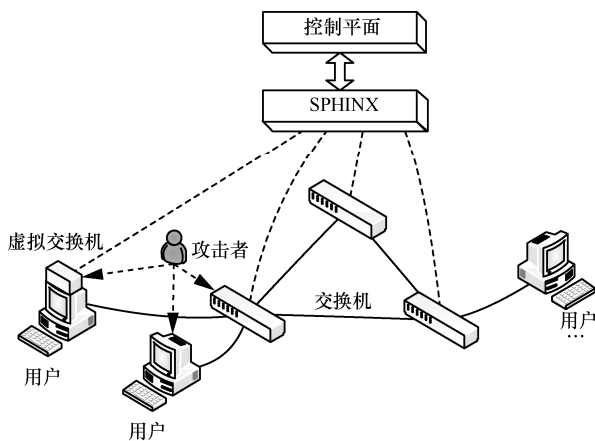


图 5 SPHINX 体系结构

4.3.2 WedgeTail

WedgeTail 是一个与控制器无关的入侵防御系统 (IPS), 旨在寻找未按预期处理数据分组的转发设备。它首先将转发设备映射为几何空间内的点, 并将数据分组视为其中的随机步行者。WedgeTail 在遍历网络时跟踪数据分组路径并生成相应的轨迹。此后, 为了检测恶意转发设备, 找到它们并识别具体的恶意行为 (如分组丢失、伪造等), 它将实际的分组轨迹与预期的分组轨迹进行比较 (即根据网络策略)。如果检测到恶意转发设备, 则 WedgeTail 按照管理员预定义的策略进行响应。例如, 即刻隔离策略可以由 2 个操作组成。首先指示潜在的恶意设备重置所有流量规则, 然后通过重复引起怀疑的数据分组以不同的间隔重新检查。如果恶意行为持续存在, 转发设备即刻与网络隔离。

为了增加成功查找恶意设备的概率, WedgeTail 依据转发设备的优先级进行检查。它采用无监督的轨迹采样^[65], 分析所有轨迹数据库, 依据分组轨迹中出现的累积频率将转发设备分成不同优先级的组。WedgeTail 拦截在控制和数据平面之间的 OpenFlow 消息, 并维护网络的虚拟副本。它使用这个虚拟副本计算预期的数据分组轨迹, 从而消除对任何转发设备的信任。为了计算

数据分组的实际轨迹, WedgeTail 依赖于 NetSight^[66], 并在数据分组穿越网络时查询它的历史记录。但当 NetSight 不可用时 (如在小型网络中), 可以使用该文中提出的定制数据分组跟踪机制来跟踪数据分组。

WedgeTail 由 2 个主要部分组成, 即检测引擎和响应引擎。检测引擎负责检索实际和预期的数据分组轨迹, 创建扫描区域并实施攻击检测算法。无论何时检测到受损设备, 响应引擎都会向控制器提交策略以保护网络。

如图 6 所示, 粗线显示的路径表示通过转发设备 fd (a) 将数据分组发送到转发设备 fd (f) 的预期路径。有虫子标记的是已被攻击者入侵的恶意设备。以虚线显示的路径为恶意转发设备将数据分组发送给攻击者的路径, 理想结果为 WedgeTail 找到并禁止这些转发设备继续恶意转发。

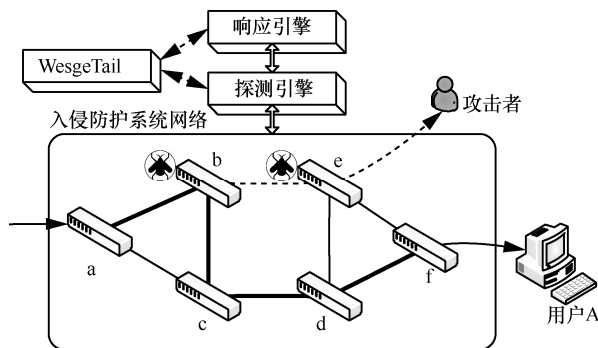


图 6 WedgeTail 架构

4.3.3 讨论

针对数据平面攻击的形式可以划分为两类。

1) 网络拓扑攻击: 如地址解析协议 (ARP, address resolution protocol) 污染攻击和链路层发现协议 (LLDP, link layer discovery protocol) 拓扑欺骗 (破坏物理拓扑状态), 来自恶意主机的以太网组管理协议 (IGMP, internet group management protocol) 消息建立虚假链路 (破坏逻辑拓扑状态)。

2) 数据平面转发攻击: 网络 DDoS 攻击 (一个或多个受感染的转发设备, 制造换路或者放大流量进而耗尽带宽); 网络向主机的 DoS 攻击 (向一个用户发送大量流量, 使其瘫痪); TCAM 耗尽攻击 (驱使控制器安装大量的流表信息); 交换机黑洞攻击 (丢弃或延时发送途径数据分组, 使

途经流量不能路由到目的地)。

SPHINX 和 WedgeTail 在前人工作^[67-70]基础上, 消除了转发设备必须可信的限制, 在有恶意转发设备的实验场景下, 攻击检测的准确性、时效性都符合实际应用的需求, 监测到攻击后阻止的成功率较高, 且资源消耗低未影响 SDN 的性能。SPHINX 特有优势是策略引擎可以由管理员编写, 在不同场景下可以保持适用性, WedgeTail 在一个相当大的网络中, 能够检测隐藏于最底层基础架构的恶意实体。

SPHINX 有 3 方面的主要局限性。首先, 系统不容忍拜占庭转发故障, 即它并不假定恶意转发设备可能会有任意行为, 因此, SPHINX 不是用来检测诸如数据分组丢失、伪造和延迟等特定恶意操作的。其次, 检测机制依赖于管理员定义的策略, 流程图组件不是根据控制器策略验证转发设备, 但与其随时间推移的行为之间相比较。因此, 如果转发设备从一开始就是恶意的, 将永远无法被检测到, 或者当网络配置发生根本性变化时, SPHINX 将产生大量误报。第三, SPHINX 不包括扫描机制, 并且在检查数据平面的威胁时未设优先级。而且, SPHINX 要求网络中大多数转发设备是可信的。虽然该假设符合现实, 但更好的解决方案不应如此。

WedgeTail 依靠网络快照来计算数据分组的预期行为。但是, 维护大型网络的快照有扩展性和性能方面的限制。WedgeTail 在检测针对不同用例和场景的攻击方面尚未被准确评估。此外, WedgeTail 与分布式控制器的兼容性尚未得到解决, 考虑到未来趋势可能是分布式控制器, 这可能是采用 WedgeTail 的主要障碍。而且, SPHINX 和 WedgeTail 都无法识别危害转发设备的漏洞(软件、硬件等)。

任何一种解决方案都要注意 SDN 转发设备硬件开销, 以及硬件延迟是否会影响这些解决方案在攻击检测中的实用性。

4.4 研究方向

考虑到针对 SDN 数据平面的攻击日新月异, 如传送攻击^[71], 攻击者能够绕过数据平面中关键网络功能, 违反安全策略。SPHINX 和 WedgeTail 都没有针对此类攻击进行评估, 现有

解决方案要预先考虑部署后可能面临的攻击, TopoGuard^[72]、SDNsec^[73]进行改进升级即可应对全新的、更狡猾的攻击者, 正是 SDN 这一架构为升级解决方案提供了必要的支持。此外, 未来需结合数据平面最新进展开展研究。例如, 解决方案对有状态数据平面的适用性, 研究有状态数据平面引入了哪些新的安全问题, 此类问题可与其他网络安全问题做类比, 如信息中心网络^[74-75]的兴趣表(PIT, pending interest table)和有状态的数据平面的状态表有相似性, 此类表项易受(D) DoS 攻击, 某些问题已经得到一定程度的解决, 如针对以路由器内部的 PIT 为目标的(D) DoS 攻击类型的缓解方法^[76]、本地基于监测的解决方案^[77]等。最后, 笔者认为 SDN 架构为开发独立于底层软件、硬件和协议的安全技术解决方案提供了必要基础, 这也正是推动安全部署这一开放、动态和不断发展的技术的不竭动力。

5 结束语

以性能为首要指标的 SDN, 安全研究方兴未艾, 鉴于数据平面在 SDN 中的基础性作用, 本文讨论了数据平面研究最新进展, 列举了针对 SDN 数据平面的攻击。分析攻击者如何利用漏洞并讨论防御方法。此后给出了本文安全方案的基本要求, 分析了攻击模型, 列举了 2 个具体解决方案, 并参考 SDN 数据平面安全要求讨论审查这些解决方案, 最后针对未来研究方案给出想法。随着 SDN 的广泛采用, 针对数据平面的安全讨论必将引人瞩目。

参考文献:

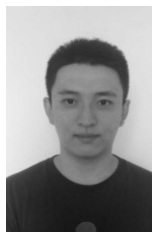
- [1] MEDVED J, VARGA R, TKACIK A, et al. Opendaylight: towards a model-driven sdn controller architecture[C]//2014 IEEE 15th International Symposium on. IEEE, 2014: 1-6.
- [2] BERDE P, GEROLA M, HART J, et al. ONOS: towards an open, distributed SDN OS[C]//The Third Workshop on Hot topics in Software Defined Networking. 2014: 1-6.
- [3] TOOTOONCHIAN A, GANJALI Y. HYPERFLOW: a distributed control plane for openflow[C]//2010 Internet Network Management Conference on Research on Enterprise Networking. 2010: 3.
- [4] Heller B. Openflow switch specification, version 1.0. 0[J]. Wire, 2009,12.

- [5] FOUNDATION O N. The benefits of multiple flow tables and ttps[R]. 2015.
- [6] BOSSHART P, GIBB G, KIM H S, et al. Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN[C]//ACM sigcomm Computer Communication Review. 2013, 99-110.
- [7] ZAL M, KLEBAN J. Performance evaluation of OpenFlow devices[J]. 2014.
- [8] BAKTIR A C, OZGOVDE A, ERSOY C. Implementing service-centric model with P4: a fully-programmable approach[C]//IEEE/IFIP Network Operations and Management Symposium. 2018: 1-6.
- [9] SONG H. Protocol-oblivious forwarding: unleash the power of SDN through a future-proof forwarding plane[C]//The Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. 2013: 127-132.
- [10] BIANCHI G, BONOLA M, CAPONE A, et al. OpenState: programming platform-independent stateful openflow applications inside the switch[J]. ACM sigcomm Computer Communication Review, 2014, 44(2): 44-51.
- [11] BIANCHI G, BONOLA M, PONTARELLI S, et al. Open packet processor: a programmable architecture for wire speed platform-independent stateful in-network processing[R]. 2016.
- [12] MOSHREF M, BHARGAVA A, GUPTA A, et al. Flow-level state transition as a new switch primitive for SDN[C]//The third Workshop on Hot Topics in Software Defined Networking. 2014: 61-66.
- [13] ZHU S, BI J, SUN C, et al. Sdpa: enhancing stateful forwarding for software-defined networking[C]//IEEE 23rd International Conference on Network Protocols (ICNP). 2015: 323-333.
- [14] HU H, HAN W, AHN G J, et al. FLOWGUARD: building robust firewalls for software-defined networks[C]//The Third Workshop on Hot Topics in Software Defined Networking. 2014: 97-102.
- [15] CHANG Y, LIN T. Cloud-clustered firewall with distributed SDN devices[C]//IEEE Wireless Communications and Networking Conference (WCNC). 2018: 1-5.
- [16] KIRAVUO T, SARELA M, MANNER J. A survey of Ethernet LAN security[J]. IEEE Communications Surveys & Tutorials, 2013, 15(3): 1477-1491.
- [17] AHLGREN B, DANNEWITZ C, IMBRENDA C, et al. A survey of information-centric networking[J]. IEEE Communications Magazine, 2012, 50(7).
- [18] ZIER L, FISCHER W, BROCKNERS F. Ethernet-based public communication services: challenge and opportunity[J]. IEEE Communications Magazine, 2004, 42(3): 88-95.
- [19] OLIVIER F, CARLOS G, FLORENT N. New security architecture for IoT network[J]. Procedia Computer Science, 2015, 52: 1028-1033.
- [20] KHAN S, GANI A, WAHAB A W A, et al. Software-defined network forensics: motivation, potential locations, requirements, and challenges[J]. IEEE Network, 2016, 30(6): 6-13.
- [21] CHOWDHARY A, PISHARODY S, HUANG D. SDN based scalable mtd solution in cloud network[C]//Proceedings of the 2016 ACM Workshop on Moving Target Defense. ACM, 2016: 27-36.
- [22] YAN Q, YU F R, GONG Q, et al. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges[J]. IEEE Communications Surveys & Tutorials, 2016, 18(1): 602-622.
- [23] CAPONE A, CASCONI C, NGUYEN A Q T, et al. Detour planning for fast and reliable failure recovery in SDN with OpenState[C]//2015 11th International Conference on the Design of Reliable Communication Networks (DRCN), 2015: 25-32.
- [24] KATTA N, HIRA M, KIM C, et al. Hula: scalable load balancing using programmable data planes[C]//The Symposium on SDN Research. 2016: 10.
- [25] BIANCHI G, BONOLA M, CAPONE A, et al. OpenState: programming platform-independent stateful openflow applications inside the switch[J]. ACM sigcomm Computer Communication Review, 2014, 44(2): 44-51.
- [26] ARASHLOO M T, KORAL Y, GREENBERG M, et al. SNAP: stateful network-wide abstractions for packet processing[C]//The 2016 ACM SIGCOMM Conference. ACM, 2016: 29-43.
- [27] ARASHLOO M T, KORAL Y, GREENBERG M, et al. SNAP: stateful network-wide abstractions for packet processing[C]//The 2016 ACM SIGCOMM Conference. ACM, 2016: 29-43.
- [28] DARGAHI T, CAPONI A, AMBROSIN M, et al. A survey on the security of stateful SDN data planes[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1701-1725.
- [29] LEVIN D, WUNDSAM A, HELLER B, et al. Logically centralized? state distribution trade-offs in software defined networks[C]//The First Workshop on Hot Topics in Software Defined Networks. 2012: 1-6.
- [30] PEREŠIĆ P, KUŽNIAR M, KOSTIĆ D. Rule-level data plane monitoring with monocle[C]//ACM sigcomm Computer Communication Review. 2015, 45(4): 595-596.
- [31] KUŽNIAR M, PEREŠIĆ P, KOSTIĆ D. What you need to know about SDN flow tables[C]//International Conference on Passive and Active Network Measurement. 2015: 347-359.
- [32] ZHANG Y, BEHESHTI N, TATIPAMULA M. On resilience of split-architecture networks[C]//Global Communications Conference, GLOBECOM. 2011: 1-6.
- [33] ZHOU Y, CHEN K, ZHANG J, et al. Exploiting the vulnerability of flow table overflow in software-defined network: attack model, evaluation, and defense[J]. Security and Communication Networks, 2018, 2018.
- [34] KLOTI R, KOTRONIS V, SMITH P. Openflow: a security analysis[C]//2013 21st IEEE International Conference on Network Protocols (ICNP), 2013: 1-6.
- [35] YOON C, LEE S, KANG H, et al. Flow wars: systemizing the attack surface and defenses in software-defined networks[J]. IEEE/ACM Transactions on Networking, 2017, 25(6): 3514-3530.
- [36] SCOTT-HAYWARD S, NATARAJAN S, SEZER S. A survey of security in software defined networks[J]. IEEE Communications

- Surveys & Tutorials, 2016, 18(1): 623-654.
- [37] BENTON K, CAMP L J, SMALL, C. Openflow vulnerability assessment[C]//The Second ACM SIGCOMM Workshop on Hot topics in Software Defined Networking ACM 2013: 151-152.
- [38] LIN P C, LI P C, NGUYEN V L. Inferring OpenFlow rules by active probing in software-defined networks[C]// 2017 19th International Conference on Advanced Communication Technology (ICACT), 2017: 415-420.
- [39] NIST: CVE-2014-9295 Detail[EB/OL]. <https://nvd.nist.gov/vuln/detail/CVE-2014-9295>, 2014
- [40] KLOTI R, KOTRONIS V, SMITH P. Openflow: a security analysis[C]//IEEE International Conference on In Network Protocols (ICNP), 2013:1-6.
- [41] KREUTZ D, RAMOS F, VERISSIMO P. Towards secure and dependable software-defined networks[C]//The Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. 2013: 55-60.
- [42] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.
- [43] KIM T H J, BASESCU C, JIA L, et al. Lightweight source authentication and path validation[C]//ACM sigcomm Computer Communication Review. 2014, 44(4): 271-282.
- [44] KIRKPATRICK K. Software-defined networking[J]. Communications of the ACM, 2013, 56(9): 16-19.
- [45] KLOTI R, KOTRONIS V, SMITH P. Openflow: a security analysis[C]//IEEE International Conference on Network Protocols(ICNP). 2013: 1-6.
- [46] SHAGHAGHI A, KAAFAAR M A, JHA S. Wedgetail: an intrusion prevention system for the data plane of software defined networks[C]//2017 ACM on Asia Conference on Computer and Communications Security. 2017: 849-861.
- [47] DHAWAN M, PODDAR R, MAHAJAN K, et al. Sphinx: detecting security attacks in software-defined networks[C]//NDSS. 2015.
- [48] KAZEMIAN P, CHAN M, ZENG H, et al. Real time network policy checking using header space analysis[C]//NSDI. 2013: 99-111.
- [49] KAZEMIAN P, VARGHESE G, MCKEOWN N. Header space analysis: static checking for networks[C]//NSDI. 2012: 113-126.
- [50] KHURSHID A, ZHOU W, CAESAR M, et al. Veriflow: Verifying network-wide invariants in real time[C]//The First Workshop on Hot Topics in Software Defined Networks. ACM, 2012: 49-54.
- [51] MAI H, KHURSHID A, AGARWAL R, et al. Debugging the data plane with anteater[C]//ACM SIGCOMM Computer Communication Review. 2011, 41(4): 290-301.
- [52] KIM T H J, BASESCU C, JIA L, et al. Lightweight source authentication and path validation[C]//ACM SIGCOMM Computer Communication Review. ACM, 2014, 44(4): 271-282.
- [53] LIU X, LI A, YANG X, et al. Passport: secure and adoptable source authentication[C]//NSDI. 2008, 8: 365-378.
- [54] NAOUS J, WALFISH M, NICOLOSI A, et al. Verifying and enforcing network paths with icing[C]//The Seventh Conference on Emerging Networking Experiments and Technologies. 2011: 30.
- [55] SASAKI T, PAPPAS C, LEE T, et al. SDNsec: Forwarding accountability for the SDN data plane[C]//25th International Conference on Computer Communication and Networks (ICCCN). 2016: 1-10.
- [56] ZHANG X, ZHOU Z, HSIAO H C, et al. ShortMAC: efficient Data-Plane Fault Localization[C]//NDSS. 2012.
- [57] AVRAMOPOULOS I, KOBAYASHI H, WANG R, et al. Highly secure and efficient routing[C]//Twentythird Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, 2004, 1.
- [58] MAHAJAN R, RODRIG M, WETHERALL D, et al. Sustaining cooperation in multi-hop wireless networks[C]//The 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2. 2005: 231-244.
- [59] AGARWAL K, ROZNER E, DIXON C, et al. SDN traceroute: Tracing SDN forwarding without changing network behavior[C]//The third Workshop on Hot Topics in Software Defined Networking. ACM, 2014: 145-150.
- [60] AWERBUCH B, CURTMOLA R, HOLMER D, et al. ODSBR: an on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks[J]. ACM Transactions on Information and System Security (TISSEC), 2008, 10(4): 6.
- [61] PADMANABHAN V N, SIMON D R. Secure traceroute to detect faulty or malicious routing[J]. ACM SIGCOMM Computer Communication Review, 2003, 33(1): 77-82.
- [62] LIU K, DENG J, VARSHNEY P K, et al. An acknowledgment-based approach for the detection of routing misbehavior in MANETs[J]. IEEE Transactions on Mobile Computing, 2007, 6(5): 536-550.
- [63] MARTI S, GIULI T J, LAI K, et al. Mitigating routing misbehavior in mobile ad hoc networks[C]//The 6th Annual International Conference on Mobile Computing and Networking. 2000: 255-265.
- [64] ZHANG X, JAIN A, PERRIG A. Packet-dropping adversary identification for data plane security[C]//2008 ACM CoNEXT Conference. ACM, 2008: 24.
- [65] PELEKIS N, KOPANAKIS I, PANAGIOTAKIS C, et al. Unsupervised trajectory sampling[J]. Machine Learning and Knowledge Discovery in Databases. 2010:17-33.
- [66] HANDIGOL N, HELLER B, JEYAKUMAR V, et al. I know what your packet did last hop: using packet histories to troubleshoot networks[C]//NSDI. 2014, 14: 71-85.
- [67] KAZEMIAN P, CHANG M, ZENG H, et al. Real time network policy checking using header space analysis[C]//The 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). 2013: 99-111.
- [68] KAZEMIAN P, VARGHESE G, MCKEOWN N. Header space

- analysis: Static checking for networks[C]//The 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12). 2012: 113-126.
- [69] KHURSHID A, ZOU X, ZHOU W, et al. Veriflow: Verifying network-wide invariants in real time[C]//The 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). 2013: 15-27.
- [70] MAI H, KHURSHID A, AGARWAL R, et al. Debugging the data plane with anteater[C]//ACM SIGCOMM Computer Communication Review. 2011: 290-301.
- [71] THIMMARAJU K, SCHIFF L, SCHMID S. Outsmarting network security with SDN teleportation[C]//IEEE European Symposium on Security and Privacy (EuroS&P). 2017: 563-578.
- [72] HONG S, XU L, WANG H, et al. Poisoning network visibility in software-defined networks: new attacks and countermeasures[C]//NDSS. 2015: 8-11.
- [73] XIA W, WEN Y, FOH C H, et al. A survey on software-defined networking[J]. IEEE Communications Surveys & Tutorials, 2015, 17(1): 27-51.
- [74] JACOBSON V, SMETTERS D K, THORNTON J D, et al. Networking named content[C]//The 5th International Conference on Emerging Networking Experiments and Technologies. 2009: 1-12.
- [75] AHLGREN B, DANNEWITZ C, IMBRENDA C, et al. A survey of information-centric networking[J]. IEEE Communications Magazine, 2012, 50(7).
- [76] ABDALLAH E G, HASSANEIN H S, ZULKERNINE M. A survey of security attacks in information-centric networking[J]. IEEE Communications Surveys & Tutorials, 2015, 17(3): 1441-1454.
- [77] COMPAGNO A, CONTI M, GASTI P, et al. Poseidon: mitigating interest flooding DDoS attacks in named data networking[C]// 2013 IEEE 38th Conference on Local Computer Networks (LCN). 2013: 630-638.

[作者简介]



郭中孚 (1994-), 男, 辽宁大连人, 国家数字交换系统工程技术研究中心硕士生, 主要研究方向为 SDN 网络、DDoS 攻击防御。



张兴明 (1963-), 男, 河南新乡人, 国家数字交换系统工程技术研究中心教授、硕士生导师, 主要研究方向为新型网络体系结构。



赵博 (1981-), 男, 吉林公主岭人, 博士, 国家数字交换系统工程技术研究中心助理研究员, 主要研究方向拟态防御架构。



王苏南 (1984-), 男, 河南洛阳人, 博士, 深圳职业技术学院副教授, 主要研究方向为网络信息安全、数据流分析。