

面向 SDN/NFV 环境的网络功能策略验证

陈浩宇^{1,2,3}, 邹德清^{1,2,4}, 金海^{1,2,3}

1. 大数据技术与系统国家地方联合工程研究中心, 华中科技大学计算机学院, 湖北 武汉 430074;
2. 服务计算技术与系统教育部重点实验室, 华中科技大学计算机学院, 湖北 武汉 430074;
3. 集群与网格计算湖北省重点实验室, 华中科技大学计算机学院, 湖北 武汉 430074;
4. 大数据安全湖北省工程研究中心, 华中科技大学网络空间安全学院, 湖北 武汉 430074)

摘要: SDN 与 NFV 技术带来了网络管理的灵活性与便捷性, 但 SDN 的动态转发策略可能导致网络功能策略失效, 同时不同网络功能的策略可能互相影响, 引起冲突问题。为了在基于 SDN/NFV 的云网络中对网络功能的策略进行验证, 分析了网络功能与 SDN 设备之间、跨网络功能之间的策略冲突, 建立了统一策略表达进行策略解析, 设计策略验证方案、框架并进行原型实现, 检验不同场景下的虚拟网络功能策略的正确性, 并与现有策略冲突验证方案对比, 用实验进行了有效性及性能分析。

关键词: 策略验证; 云网络; 软件定义网络; 网络功能虚拟化

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2021035

Verification on policies for network functions in SDN/NFV-based environment

CHEN Haoyu^{1,2,3}, ZOU Deqing^{1,2,4}, JIN Hai^{1,2,3}

1. National Engineering Research Center for Big Data Technology and System, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
2. Services Computing Technology and System Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
3. Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
4. Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract: Although the newly introduced SDN and NFV technologies bring flexibility and convenience in network management, the dynamic forwarding policies introduced by SDN may cause invalidation in the network function

收稿日期: 2020-12-14; 修回日期: 2021-01-21

通信作者: 金海, hjin@hust.edu.cn

基金项目: 国家重点研发计划 (2019YFB2101700); 广州市未来产业关键技术研发专题项目 (201902020016)

Foundation Items: The National Key R&D Program of China (2019YFB2101700), The Science and Technology Program of Guangzhou (201902020016)

论文引用格式: 陈浩宇, 邹德清, 金海. 面向 SDN/NFV 环境的网络功能策略验证[J]. 网络与信息安全学报, 2021, 7(3): 59-71.
CHEN H Y, ZOU D Q, JIN H. Verification on policies for network functions in SDN/NFV-based environment[J]. Chinese Journal of Network and Information Security, 2021, 7(3): 59-71.

policies, and the policies in different network functions may also cause conflicts due to their own behaviors. In order to verify the policies in SDN/NFV-based cloud network, the verification on policies between the network function and the SDN device, as well as across the network functions were considered. A unified policy expression for analysis was summarized, and policy verification scheme, framework and prototype implementation were proposed to verify the correctness of policies in different scenarios, then experiments were conducted to justify the effectiveness and performance.

Keywords: policy verification, cloud network, software-defined networking, network function virtualization

1 引言

现代网络技术正朝虚拟化的方向快速发展。软件定义网络 (SDN, software-defined network)^[1-2] 实现了网络转发设备的虚拟化, 通过将传统硬件交换机的控制层与数据层分离, 以集中化、可编程的方式部署并执行转发控制逻辑。网络功能虚拟化 (NFV, network function virtualization)^[3] 技术将传统的基于硬件的防火墙、入侵检测/防御系统 (IDS/IPS, intrusion detection/prevention system)、流量监控器等网络设备虚拟化, 以软件的形式运行在通用硬件 (如 x86 架构的服务器) 上, 从而使管理员能够实现动态的网络服务按需调配。SDN 和 NFV 技术得到了广泛认可, 并在实际的生产环境中得到了应用, 如将 SDN 运用在数据中心^[4], 将 SDN 和 NFV 结合运用于网络切片等^[5-6]。

然而, SDN 和 NFV 技术为网络管理带来灵活性与便捷性的同时, 基于 SDN/NFV 的网络所具有的动态性为策略验证带来了新的挑战。尤其在云环境中, SDN 使云网络拓扑结构能够随时改变, SDN 交换机中的网络转发规则可以根据用户需求进行动态调整^[5]; NFV 支持虚拟网络功能实例, 如软件防火墙、软件 IDS 等, 根据业务需求, 动态地生成并部署在网络中任意位置^[7]。这两个新特性给网络中的策略验证带来了两方面新的挑战。一方面, 基于 SDN 中转发策略处于动态更新的状态, 网络流量路径可能在更新转发策略后, 绕过原先布置的网络功能, 从而导致网络功能中的策略失效, 即 SDN 与网络功能的策略冲突; 另一方面, 不同于传统硬件网络功能部署在固定的位置上, NFV 能够在网络拓扑任意位置随时部署新的网络功能实例, 并且不同类型的网络功能 (如同时存在的网络代理、防火墙、IDS 等) 之间可能导致策略的失效, 即跨网络功能的策略冲突。

为了解决基于 SDN/NFV 中的策略验证问题, 本文设计了适用的动态策略验证方案, 对基于网络功能策略中网络数据包头部匹配域以及动作域的验证方案进行扩展, 同时在 SDN 架构中研究了动态转发规则更新可能引起的策略冲突问题以及验证方法, 并进一步研究了不同类型网络功能自身行为可能引起的策略冲突。此外, 基于 SDN 架构, 设计了专用的跨网络功能的策略验证框架, 并进行了原型实现。在实验方面, 基于 SDN 控制器 (RYU)、网络代理 (squid)、防火墙 (iptables) 以及入侵检测/防御系统 (Snort) 等代表性的开源工程实现原型系统并进行了网络功能的策略验证, 证明了方案的有效性与执行效率。

2 国内外研究现状

已有大量工作针对传统网络^[8]和 SDN^[9]中的转发策略进行了可达性和隔离性的验证, 主要研究在网络转发过程中, 交换机中的转发策略是否会导致网络中主机无法正常通信或引起非法通信。文献 [10-11] 实现了转发策略的实时验证, 即针对输入的策略进行动态、即时逐条验证。此外, 有研究指出网络中的网络功能数量已堪比转发设备^[12], 因此针对网络功能的策略验证必不可少^[13]。常见工作有专门针对防火墙^[14-16]的策略验证, 保证单个或多个防火墙策略的有效性; 或者对防火墙与 IPS 之间的策略冲突进行检测^[17]; 还有一些工作采用对网络功能的建模^[18,19]或生成测试流量^[20]的方案对网络功能中的策略进行验证。这些针对网络功能的策略验证主要进行可达性和隔离性的验证, 技术思路主要基于策略中关于网络数据包的头部匹配域以及动作域进行建模, 通过比较策略的匹配域是否重叠, 并判断重叠部分的动作域是否冲突来进行验证。

具体而言, 表 1 总结了目前不同场景下代表性策略验证工具的可用性。其中 Veriflow^[11]和其改进版本工具 Delta-net^[10]用于网络转发策略的验

表 1 不同场景下代表性策略验证工具的可用性
Table 1 Usability of representative policy verification tools in different scenarios

验证工具	网络转发策略验证	网络功能策略验证	SDN 交换机与网络功能策略验证	跨网络功能策略验证
Veriflow	●	○	○	○
Delta-net	●	○	○	○
Symnet	●	○	●	○
FAME	○	◎	○	◎
VFW	○	◎	●	◎
Plankton	●	◎	●	◎
NetSMC	○	●	○	●

注：● 表示支持，◎ 表示部分/单一支持，○ 表示不支持

证，主要先将网络转发策略进行原子化切割形成不同的域，再使用增量查找的方式快速验证实时改变的转发策略；但由于它们只支持转发策略，因此无法实现与网络功能相关的策略验证。Symnet^[9]则使用符号执行的方案，依据生成的符号模型产生测试流量来验证网络整体的可达性和隔离性，因此它能够实现 SDN 交换机与网络功能之间的策略验证，但是由于无法理解网络功能策略的语义，无法执行跨网络功能的验证。FAME^[14]和 VFW^[15]均支持防火墙策略验证，包括单一防火墙和跨防火墙之间的策略验证，但是并不支持其他类型的网络功能。同时，VFW 面向 SDN 环境，可以实现 SDN 交换机与网络功能的策略验证，但是仅支持防火墙这一种跨网络功能策略验证。Plankton^[21]面向协议进行策略验证，从而能够在支持转发策略验证的同时，兼顾多种网络功能的策略验证；但是由于其只理解网络协议而不理解网络功能的策略语义，无法解决跨网络功能策略冲突。NetSMC^[22]则专门针对有状态的网络功能进行有状态的策略验证，能够实现跨网络功能的策略验证；然而由于其并未考虑 SDN 的动态转发环境，因此无法实现 SDN 交换机与网络功能的策略验证。

3 SDN 架构及网络功能策略

本节对相关背景知识进行介绍，首先描述 SDN 架构，接着介绍各类网络功能的安全策略模型，最后介绍常见的策略冲突问题与解决办法。

3.1 SDN 网络架构

SDN 的基本架构如图 1 所示，分为数据层、

控制层和应用层 3 个层次。在数据层中，SDN 将传统交换机中的控制逻辑剥离出来，只留下基本的转发功能，即只基于流表 flowtable 进行流量转发的 SDN 交换机。控制层则由 SDN 控制器构成，通过南向接口的通信协议与数据层的 SDN 交换机进行通信，并通过控制器中北向接口的可调用函数入口支撑上层的 SDN 应用。应用层运行着开发者的 SDN 应用，可以获取全局的网络信息，进行交换机的功能、转发策略配置，以及接收并响应来自控制层的事件、通知等消息。

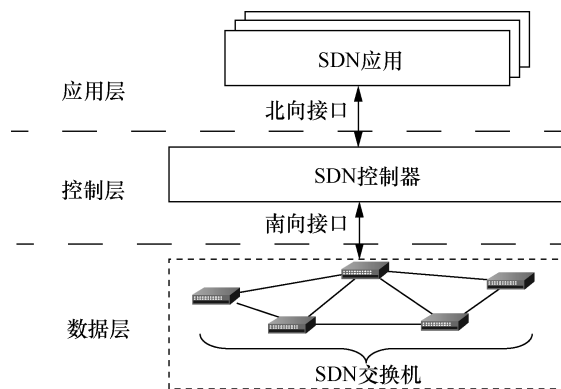


图 1 SDN 的基本架构
Figure 1 Basic architecture of SDN

目前，流行的 SDN 控制器包括 RYU、floodlight、OpenDayLight、ONOS 等，最主流的南向接口即 SDN 的通信协议，为 OpenFlow 协议，而北向接口则依据控制器的具体实现，由控制器的开发人员根据需要进行设计与定义。

3.2 防火墙策略模型

防火墙策略是一系列防火墙规则的集合。防

防火墙规则具有 $\langle \text{order} \rangle \langle \text{predicate} \rangle \rightarrow \langle \text{action} \rangle$ 的形式。其中, **order** 表示规则的优先顺序, 匹配数据报字段时将依照 **order** 的顺序对数据报依次进行匹配, 第一次匹配成功的规则即为将要执行的规则, 其后的规则不再进行匹配。**predicate** 是对数据报字段的描述, 它包括协议、源端口、目的端口、源 IP 地址、目的 IP 地址等, 字段满足谓词 **predicate** 的数据包即与该条规则匹配成功。**action** 表示对满足谓词 **predicate** 的数据包进行的具体操作, 包括两种操作: **deny** (丢弃数据包) 与 **accept** (允许通过数据包)。

防火墙检测的数据包具有报头, 它是表示数据包中可寻址元素的一系列字段集合, 同时它是与防火墙规则进行匹配时要考察的字段, 如式(1)所示。

$$h = \{h_1, h_2, \dots, h_n\} \quad (1)$$

一条过滤规则的过滤字段 f 也是一系列字段的集合, 如式(2)所示。

$$f = \{f_1, f_2, \dots, f_n\} \quad (2)$$

h 与 f 应具有相同形式, 对于一条报文, 如果 $\forall i, h_i \in f_i$, 则数据包与该规则匹配, 将按照这条规则的动作对数据包进行处理。对于一条防火墙规则, 它具有如式(3)的形式。

$$r = \langle i, f, \text{action} \rangle \quad (3)$$

其中, i 表示该条规则在防火墙规则集合中的匹配优先度, 当一个数据包到达时, 将依照自上而下的顺序对数据包进行匹配, 匹配成功则执行该条规则, 否则将继续匹配直到匹配成功或所有规则检查完毕。**action** 字段表示对数据包的操作。 f 为规则的过滤字段, 用来与报头进行匹配, 用如下五元组表示。

$$\langle \text{protocol}, \text{src_ip}, \text{src_port}, \text{dest_ip}, \text{dest_port} \rangle$$

其中, **protocol** 是协议类型, 包括 **tcp**、**icmp** 等。**src_ip** 是源 IP 地址, **src_port** 是源端口, **dest_ip** 是目的地址, **dest_port** 是目的端口, 与网络流的五元组定义相同。

3.3 IDS/IPS 策略模型

IDS/IPS 策略包含一系列检测规则, 不同厂商/开发者的规则格式不一, 但表达的内容大同小异。以代表性的 IDS/IPS——Snort 为例, 它采用

了一种简单的轻量级语言来描述规则, 一条完整规则的含义就是对具有特定特征的数据包进行匹配并操作, 允许通过或拒绝通过并发出警示。Snort 规则的语义模型可以用如下二元组表示。

$$\langle \text{packet}, \text{action} \rangle$$

其中, **packet** 表示满足给定条件的所有数据包, 这些条件包括源 IP 地址、目的 IP 地址、源端口、目的端口以及一些给定的特征值。**action** 则表示对满足给定条件数据包的具体操作, 包括允许通过、拒绝通过、发送响应报文 3 种, 其中允许通过与拒绝通过统称为控制动作 (**control_action**), 发送响应报文称为响应动作 (**resp_action**), 响应动作是指 Snort 向发送数据的源地址发送响应报文以告知它对数据的处理, 而允许通过与拒绝通过又分别包括特定动作, 具体情况如下。

Control_action={**permit**,**deny**}

Permit={**alert**,**log**}

Deny={**drop**,**reject**,**react**,**sdrop**,**resp**}

Resp_action={**resp**,**reject**,**react**}

Snort 规则分为如下两部分: 规则头 (**rule_header**) 和规则选项 (**rule_option**)。其中, 规则头包含五元组 (对数据包采取的动作, 源 IP 地址, 目的 IP 地址, 源端口, 目的端口)。而规则选项则包含数据包的具体特征, 特征的种类很多, 包括数据字段的值、**icmp_seq** 等, 而 IDS/IPS 的精髓在于其规则选项, 通过设定规则选项筛选出具有特定特征的数据包, 从而进行进一步处理。其模型如式(4)、式(5)所示。

$$\text{rule} = \langle \text{rule_header}, \text{rule_option} \rangle \quad (4)$$

$$\begin{aligned} \text{rule_header} = & \langle \text{action}, \text{src_ip}, \\ & \text{src_port}, \text{dest_ip}, \text{dest_port} \rangle \end{aligned} \quad (5)$$

以如下规则为例说明 Snort 规则。

Alert tcp 10.12.5.109 any->12.25.1.1 any (content:baf)

该规则含义为对于协议为 **tcp** 协议、从 10.12.5.109 任何端口发往 12.25.1.1 任何端口的包含字段“**baf**”的流量, 采取告警措施。其中, **alert** 表示动作, **tcp** 表示协议, **any** 表示源端口/目的端口, 10.12.5.109 和 12.25.1.1 表示源地址目的地址, “->”表示流量方向, **content: baf** 是规则选项, 用于筛选包含 **baf** 字段的流量。

3.4 统一策略表示

观察并总结防火墙、IDS/IPS 的策略模型后，本文提出一种统一策略表示，用于描述不同类型虚拟网络设备的策略并用于冲突检测。从上文的不同类型网络功能的策略模型可知，网络功能甚至网络交换机中的策略可以用四元组表示。

<priority,match,other,action>

具体总结如表 2 所示，其中，priority 字段通常为正整数；match 字段则通常包含网络流的五元组，或者根据策略表达的需求进行扩充；other 字段则依据不同网络功能的具体行为进行定义，如 IDS/IPS 中定义的数据包 content 匹配字符串或代理网关中定义缓存网站的域名字符串；action 字段则通常用枚举类型表示，指明动作代码。对于交换机和一般防火墙的策略，则包含 priority、match 和 action 域，按照策略顺序，匹配数据包并进行转发或者访问控制；而对于其他网络功能（如 Snort 或 Squid），还需要引入 other 字段以支撑更丰富的功能。本文基于统一策略表示对网络中的转发策略和网络功能策略进行解析，并进行冲突检测。

表 2 统一策略表示域名

Table 2 Field names of unified policy representation

域名	说明
priority	策略的匹配优先级
match	数据包头部匹配，一般包含网络流五元组
other	不同设备中特有的策略字段，如载荷匹配
action	对匹配到的数据包进行的操作

3.5 常见策略冲突分类

网络设备中常常部署着大量不同的策略，这些策略虽然格式、语义等各不相同，但是互相之间易产生冲突，因此需对其进行验证。策略冲突可能发生在单个设备内部，也可能发生在多个设备之间。单设备中常见的策略冲突主要分为以下三类。

(1) 互斥

每条策略都具有特定的作用域（如 IP 地址、端口号范围），而不同策略的作用域可能有交叉重叠。如果两条策略具有交叉的作用域，同时具有不同的行为（例如，防火墙的两条具有 IP 域交叉的策略，一条为 Allow，另一条为 Deny），则会发生策略互斥冲突。

(2) 冗余

两条策略如果对相同的作用域制定了相同的动作，则可以判定两条策略发生了冗余冲突，需要对策略集进行精简。

(3) 覆盖

一些访问控制设备（如防火墙）中的策略往往具有优先级，在成功匹配到一条策略后则不再匹配后续策略。如果具有高优先级的策略作用域完全覆盖了某一条具有低优先级的策略，那么该低优先级策略将永远无法被匹配到，此时发生策略覆盖冲突。

多设备间的策略冲突可以看作单设备情形的扩展，依据设备类型可以分为两类。① 同类设备策略冲突，即网络中可能同时部署着多个同类型的设备，这些设备中的策略发生了与单设备中类似的冲突。在验证这类冲突时，需要从全局角度出发，再参照单设备中可能发生的策略冲突情形进行验证。② 异类设备策略冲突，即不同类型的设备中发生策略冲突。不同类型设备往往使用不同格式、语义和目的的策略，因此该类冲突最为复杂，需要对不同的策略语义进行综合考量，并找出潜在的策略冲突。

本文工作针对 SDN/NFV 网络环境中的特有策略冲突场景提出验证方案，涉及的策略冲突属于多设备中的异类设备策略冲突。虽然现有一些工作已经针对各种各样的策略冲突提出了验证方案，但在 SDN/NFV 这种特定上下文环境中仍包含特有的策略冲突场景亟待解决。

4 SDN/NFV 中的策略冲突

本节用具体的案例对 SDN/NFV 中特有的 SDN 交换机与网络功能之间以及跨网络功能之间的策略冲突进行场景介绍并分析原因。

4.1 SDN 交换机与网络功能的策略冲突

SDN 中应用最广泛的协议是 OpenFlow 协议，用于 OpenFlow 交换机与控制器之间的通信。OpenFlow 交换机基于流表中的流表项进行转发，而一条流表主要包含匹配域、统计域与动作域 3 个域，如图 2 所示。其中，匹配域用于匹配数据包头，可以基于任意字段进行匹配，如 IP/MAC 地址、入端口或者协议等；统计域则针对匹配域所表示的数据流

记录统计信息,包括该流表项匹配到的数据包个数等;动作域则表示在 OpenFlow 交换机中对匹配到的数据包如何操作,如转发到某端口、镜像流量、广播流量等。

匹配域	统计域	动作域
-----	-----	-----

图 2 OpenFlow 交换机流表项示意
Figure 2 Illustration on an entry of flowtable in OpenFlow switches

在基于 OpenFlow 协议的 SDN 环境中,用户根据自己的需求设计并编写应用,并将应用运行在控制器上;转发策略由应用生成,并调用控制器的配置接口以安装到 OpenFlow 交换机中。虽然这种设定颇具灵活性,有利于自动化的转发策略生成与部署,但是带来了网络动态性,在配置转发策略时,很难综合考虑网络功能的策略,从而引起冲突,如配置 OpenFlow 交换机的转发策略时,可能会忽略网络功能(如防火墙的策略),使交换机的功能满足了需求,却造成防火墙策略失效。

在基于 SDN 的云网络中,内部网络的主机需要与外部的文件服务器进行通信来获取数据。假设在内部网络出入口的防火墙中定制了安全策略,允许文件服务器与主机 X 之间的通信,但是禁止文件服务器的数据发送给主机 Y。SDN 中的转发策略是动态下发的,由上层应用安装,因此可能出现如图 3 所示的情形。① 在 S2 交换机的流表中临时安装了一条规则,将入端口来自 S1 交换机的数据包进行镜像,同时转发给 S3 交换机和主机 X 所在的端口。② 同时在 S3 交换机的流表中临时安装了一条规则,将入端口来自 S2 交换机的流量转发给主机 Y 所在的端口。当文件服务器向主机 X 发送数据时,由于数据包经由 S1 交换机转发给了 S2 交换机,而 S2 交换机镜像转发给 S3 交换机的流量最终可以达到主机 Y,结合源地址来看,从文件服务器发出的流量除了抵达主机 X,还抵达了主机 Y。虽然它们之间的数据传输不是直接而是间接进行,但这依然绕过了防火墙策略,出现了策略失效的情况。该情况下,认为防火墙规则与交换机之间产生了冲突。同理,将防火墙替换为 IPS,依然会出现同样的策略失效。

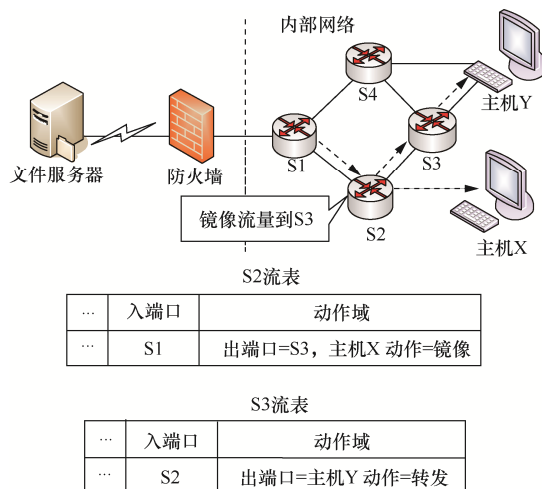


图 3 SDN 交换机与防火墙策略冲突场景
Figure 3 The scenario of policy conflicts between SDN switches and firewalls

SDN 交换机与网络功能的策略冲突产生的原因为交换机中的转发策略配置不当,导致出现了不合理的转发路径,因此需要根据网络功能中配置的策略,对比找出是否存在对应的非法转发路径。

4.2 跨网络功能策略的冲突

网络中可能由于具体业务的需求同时存在多个功能不同的网络功能,如云环境中为了优化主机对外部网站的访问,添加了可用于缓存的代理网关,并且在网络出入口处部署了访问控制设备。它们可能由于策略配置不当出现冲突。

考虑如图 4 所示的场景,在该场景中云网络中运行着两个不同的网络功能——代理网关和防火墙。云内主机对外部网站的访问均会经过代理网关和防火墙处理。该场景中,在策略配置时,代理网关被配置为可以对任意网页进行缓存,从而加快主机重复访问的速度,而防火墙则被配置为禁止主机 2 访问 ABC.com 这个网站,其他主机对该网站的访问则被放行。当主机 1 首次访问 ABC.com 时,防火墙放行了该主机的访问,并且代理网关中自动对该网站进行了缓存以加快再次访问的速度。然而,当之后主机 2 试图访问 ABC.com 时,代理网关发现缓存中存在该网站后将会直接返回给主机 2,因此主机 2 可以顺利访问到 ABC.com 的缓存版本。但是在防火墙策略中应当拒绝主机 2 对 ABC.com 的访问,这就导致了防火墙策略的失效。

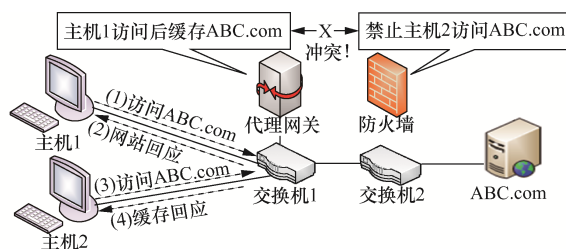


图4 代理网关与防火墙策略冲突场景

Figure 4 The scenario of policy conflicts between the proxy and the firewall

跨网络功能的策略冲突主要来自不同网络功能对于同一个流或者会话的策略定义了冲突的动作，如该场景中代理网关将 ABC.com 进行缓存，发送给主机 2，而防火墙则拒绝了 ABC.com 与主机 2 之间的通信。这两个网络功能的动作产生了矛盾，造成了策略冲突。

5 策略验证

本节针对第 4 节中提出的两种基于 SDN/NFV 云环境中特有的策略冲突场景适用的安全策略进行验证。

5.1 SDN 交换机与网络功能策略验证

依据 SDN 交换机（即 OpenFlow 交换机）与防火墙策略冲突产生的场景与原因，提出冲突检测算法，如算法 1 所示，寻找 OpenFlow 交换机与防火墙安全策略之间的冲突，即从一系列防火墙规则中寻找两条规则，它们满足如下约束条件。

- (1) $x[i].action == \text{deny}$
- (2) $x[j].action == \text{accept}$
- (3) $x[i].src_ip == x[j].src_ip$
- (4) 存在交换机 z ，使 $\text{findpath}(x[i].dest_ip, z) \geq 1$ 且 $\text{findpath}(x[j].dest_ip, z) \geq 1$

算法 1 OpenFlow 交换机与防火墙策略验证算法

BEGIN

For i from 0 to rule_number in 防火墙:

do: 寻找规则 $x[i]$ ，其动作 $x[i].action == \text{deny}$

for j from 0 to rule_number in 防火墙:

do: 寻找规则 $x[j]$ ，其动作 $x[j].action$

$== \text{accept}$

if ($x[i].src_addr == x[j].src_addr$):

for 交换机 z in 防火墙直连交

换机集合:

```
if (findpath( $x[i].dest\_ip, z$ )  $\geq 1$  &&
    findpath( $x[j].dest\_ip, z$ )  $\geq 1$ ):
    showpath( $x[i]$ );
    showpath( $x[j]$ );
```

END

针对以上约束条件进行说明。

(1) 防火墙规则 $x[i].action$ 动作为 **deny**，即本规则定义中的源地址与目的地址不能直接通信。

(2) 防火墙规则 $x[j].action$ 动作为 **accept**，即本规则定义中的源地址流出的流量能顺利抵达目的地址。

(3) 规则 $x[i]$ 和规则 $x[j]$ 的源 IP 地址 src_ip 相同，则意味着通信流量是从同一个源地址流出的。

(4) 存在某个直接连接防火墙的 OpenFlow 交换机 z ，同时满足两个条件：从源地址 $x[i].src_ip$ 流出的流量能通过该交换机顺利抵达规则 $x[j]$ 的目的地址 $x[j].dest_ip$ ，即 $\text{findpath}(x[j].dest_ip, z) \geq 1$ ；从源地址 $x[j].src_ip$ 流出的流量能通过该交换机顺利抵达规则 $x[i]$ 的目的地址 $x[i].dest_ip$ ，即 $\text{findpath}(x[i].dest_ip, z) \geq 1$ 。

此处规则 $x[i]$ 为不允许，即从源地址 $x[i].src_ip$ 发出，目的地址为 $x[i].dest_ip$ 的流量不能通过防火墙。但由于从源地址 $x[i].src_ip = x[j].src_ip$ 发出的目的地址为 $x[j].dest_ip$ 的流量可以通过防火墙，并且存在 OpenFlow 交换机能够使该流量经过该交换机或其后某个交换机时发生分流，导致该流量的一部分流向目的地址 $x[i].dest_ip$ ，这就间接导致源地址 $x[i].src_ip$ 和目的地址 $x[i].dest_ip$ 之间存在可行的流通信路径，违背了规则 $x[i]$ 的定义。此时，即使防火墙的规则正确执行，依然可能由于 OpenFlow 交换机的转发策略配置不当，使防火墙中定义为无法通信的两端出现可通信路径，造成策略失效，因此造成 SDN 交换机与防火墙策略冲突。策略验证的核心思路为先查找防火墙或 IPS 中每条动作为拒绝的策略，对交换机转发路径建立树状模型，寻找是否有可达路径，最终找出策略失效所在。算法中有两个关键函数。

(1) **findpath** 函数

该函数检查所有直接连接防火墙的 OpenFlow 交换机，旨在找出从某个直接连接防火墙的 OpenFlow 交换机抵达目的地址的所有路径，

并将路径存放在树中, 然后依据返回值路径的条数判断防火墙与交换机之间是否有冲突。函数使用两个重要参数: ①目的地址; ②直连防火墙的 OpenFlow 交换机设备号。如果该 OpenFlow 交换机到该目的地址具有一条或一条以上路径, 返回路径条数, 否则返回 0。

该函数在寻找路径的同时对路径树进行构建, 此处选择构建一棵前缀树。构建树形数据结构一方面是便于分析路径, 另一方面是为之后的打印路径做铺垫。因此, 该函数主体包括两部分: ①寻找路径, 具体过程如算法 2 所示, 算法主要使用递归的方式, 对流表项的下一跳地址反复查询, 直至下一跳地址为目的地址时, 递归结束。②构建前缀树, 如何把下一跳 OpenFlow 交换机的流表创建一棵树并将这棵树映射到下一棵树的地址。该过程由两个步骤完成: 其一是创建下一棵树, 给它分配存储空间; 其二是将下一跳 OpenFlow 交换机地址存入树的数据结构中。

算法 2 寻找路径算法

BEGIN

For each rule in OpenFlow 交换机流表:

If (该流表项下一跳地址==目的地址)
return 1;

else

result = result + findpath (目的地址,
该流表项下一跳设备号);

return result;

END

(2) showpath 函数

该函数的功能是打印某个 OpenFlow 交换机设备号及经由该交换机后抵达目的地址的所有路径上的设备号。当 findpath() 函数返回值大于或等于 1 时, 说明从直接连接防火墙的 OpenFlow 交换机到达目的地址的路径存在, 此时可以调用 showpath() 将 findpath() 构建的树中的所有路径打印出来。函数包含两个参数: ①要打印的树的根节点; ②树的深度 n , 利用 n 调整打印格式, 决定交换机设备号的缩进关系, 以便查看路径。

5.2 跨网络功能策略验证

跨网络功能的策略检测相对简单, 只需先找到匹配域相同的网络功能策略, 再分析并对比这

些策略的动作是否存在冲突, 具体的验证流程如图 5 所示, 共包含 5 个步骤。

(1) 策略解析。该步骤以网络功能策略作为输入, 用统一策略表达对规则进行解析, 分解成 priority、match、other 和 action 域。

(2) 对比 match 域。该步骤为了找出针对的流有重叠的规则, 因为它们可能对同一条流采用不同的动作。具体的提取算法可以采用已有工作的防火墙策略检测算法^[16]。

(3) 提取 action 域。该步骤找出有重叠 match 域的策略的动作域, 通常防火墙或者 IDS/IPS 包含 allow、deny、drop 等。

(4) 提取 other 域。该步骤是策略验证中容易忽略的部分, 与网络功能自身的功能相关。存在一些网络功能的策略中并不直接定义 action 域 (默认 allow), 但是在 other 域中隐含动作, 如代理网表中存在存储行为 (cache), 而 NAT (network address translation) 设备则可能存在包重写行为 (重新定义包头)。

(5) 交叉对比。对重叠策略中所提取出的 action 域和 other 域动作进行交叉对比, 并输出策略检测报告, 警告管理员是否存在冲突。

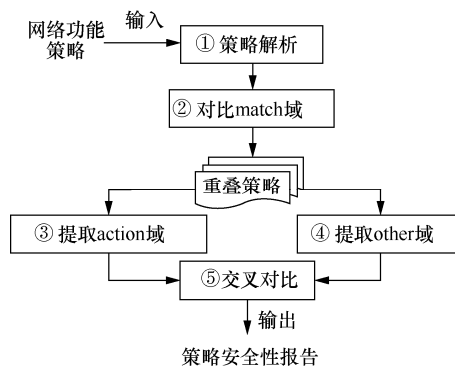


图 5 网络功能策略验证流程

Figure 5 The flow of policy verification across network functions

该策略冲突的解决方案通常需要管理员根据实际情况进行调整, 即对于引起策略冲突的流量进行人为决策。这是因为是放行还是阻止流量取决于实际网络环境以及该流量的用途。

6 基于 SDN 的策略验证框架

基于上文分析的策略验证需求, 本文提出了一种云环境中基于 SDN 的策略验证框架, 如图 6

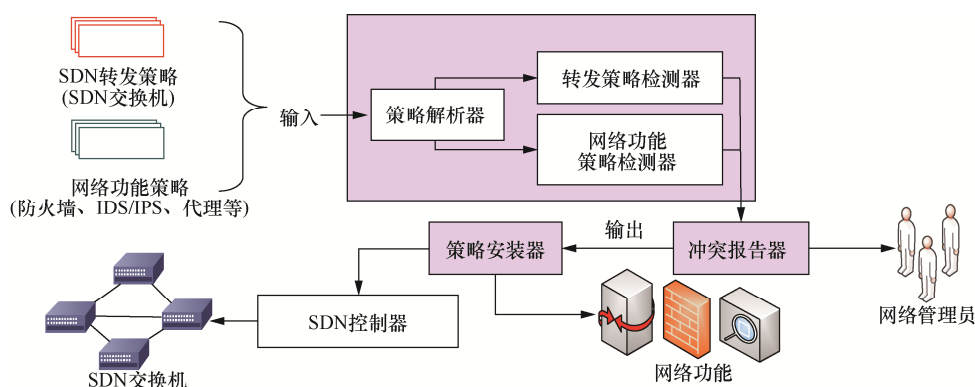


图6 基于SDN的策略验证框架
Figure 6 SDN-based policy verification framework

所示。该框架以 SDN 转发和网络功能策略为输入，包含策略检测引擎、冲突报告器和策略安装器 3 个部分，其中策略检测引擎包含策略解析器、转发策略检测器和网络功能检测器 3 个子模块；作为输出，将策略冲突警告报告给网络管理员，以及将合法网络功能策略直接安装到网络功能中，或通过 SDN 控制器将合法转发策略安装到 SDN 交换机中。

首先，SDN 转发策略（运行在 SDN 交换机中）以及网络功能策略（运行在防火墙、IDS/IPS、代理等网络功能中）作为策略检测引擎的输入。在策略检测引擎中，每条策略先被策略解析器进行解析，填充进 3.4 节中定义的统一策略表示所对应的数据结构中。接着，策略解析器根据策略的类型，将结构化的策略分别发送到转发策略检测器和网络功能策略检测器中进行检测。在转发策略检测器中，SDN 转发策略可以进行网络转发的可达性或隔离性检测，也可以对网络功能策略进行交叉检测（即第 5.1 节中的检测算法），保障 SDN 交换机中与网络功能中的策略不冲突；在网络功能策略检测器中，网络功能策略将会进行交叉检测（如第 5.2 节中的检测流程），确保网络功能中的策略不冲突。策略检测引擎中的两个策略检测器会将结果输出到冲突报告器中。冲突报告器策略检测引擎中检测到的策略冲突作为警告通知网络管理员，并且将合法策略作为输出发送给策略安装器，然后策略安装器通过 SDN 控制器安装合法转发策略，或直接将合法网络功能策略安装到网络功能中。

在该框架的原型实现中，策略检测引擎、冲突报告器、策略安装器均以 SDN 应用的形式运行在 SDN 控制器中。本文选用基于 Python 语言的 RYU 控制器作为 SDN 控制器。其中，策略检测引擎应用包含 3 个类 class，分别对应策略解析器、转发策略检测器和网络功能策略检测器；冲突报告器应用运行时直接在终端输出警告信息；策略安装器则采用 RYU 控制器提供的基于 OpenFlow 协议的消息接口——flow_mod 和 add_flow 安装 SDN 转发策略，并且直接利用 socket 通信调用网络功能的远程配置接口进行策略的安装。

7 实验测试与分析

本文利用上述策略验证框架的原型实现，对有效性和性能进行分析。实验环境由 2 台运行 ubuntu 16.04 操作系统的服务器组成，服务器的配置为英特尔至强 Xeon E3-1230 v3 CPU，32 GB 内存，100Mbit/s 网卡，250 GB Samsung EVO850 SSD+2 TB 西部数据硬盘。每台服务器使用 VMware Workstation 12.0 运行若干虚拟机，采用 ubuntu 16.04 操作系统，配置 2 GB 内存、60 GB 硬盘空间以及双核双线程的虚拟 CPU。此外，SDN 控制器选用 RYU 4.2 版本，虚拟交换机使用 OpenvSwitch 2.5.0，运行 OpenFlow 1.3 版本，并且支持开源 iptables-v1.4.21、Snort-2.9.8.2 的策略解析与验证。

图 7 为实验测试的网络拓扑与设备 id。其中，主机 7、15、16 分别使用一个虚拟机，iptables 运行在一个虚拟机中，而用于模拟所有

OpenFlow 交换机的 mininet 则单独运行在另外一个虚拟机中。表 3 为 OpenFlow 交换机中的转发策略。为了便于分析, 此处只列出会引起冲突问题的策略, 并且 IP 地址用主机 id 和交换机 id 表示。

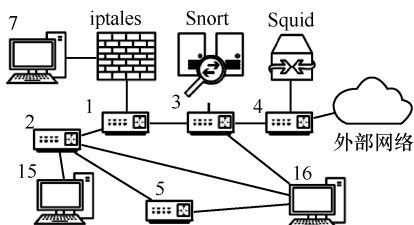


图 7 实验拓扑
Figure 7 Topology of experiment settings

表 3 交换机转发规则
Table 3 Forwarding rules deployed in switches

交换机设备号	目的地址	动作
1	15	转发到 2
1	16	镜像到 2、3
2	15	转发到 15
2	16	镜像到 5、16
3	16	转发到 16
5	16	转发到 16

7.1 有效性测试

7.1.1 SDN 交换机与网络功能策略验证

SDN 交换机与网络功能策略的冲突验证。实验中, 使用防火墙 iptables 作为网络功能的代表进行验证, 与部署在网络中的 OpenFlow 交换机一起进行策略验证。其中 iptables 规则如表 4 所示。

SDN 交换机与网络功能策略检测结果如图 8 所示。防火墙规则 2 允许通过的流量间接抵达规则 1 的目的地址, 此时防火墙规则与 OpenFlow 交换机之间产生了冲突。由打印序列可知, 从源地址 7 到目的地址 16 的路径有如下 3 种: ①1-2-5-16; ②1-2-16; ③1-3-16。而从源地址 7 到目的地址 15 的路径为 1-2-15。将两者的路径进行对比, 可以发现 1-2-5-16 与 1-2-15 具有公共部分交换机 1 和交换机 2, 说明数据经过这两个交换机时如果发生流量分流, 会导致部分流量流向另一个目的地址, 而这是与防火墙安全策略相违背的, 因为规则 2 允许流量抵达地址 15, 规则 1 不允许流量抵达地址 16, 现在流向 15 的流量却可能到达 16, 因此防火墙策略与 OpenFlow 交换机 1、2 之间产生了冲突。

表 4 防火墙 iptables 规则 (验证转发策略)

Table 4 Rules of iptables (verifying forwarding policies)

index	proto	src_ip	src_port	dest_ip	dest_port	action
1	tcp	7	24	16	any	deny
2	tcp	7	24	15	any	accept
3	udp	7	27	16	512	deny

```

There are conflicts between rule 1 and rule 2!
Here are the routes of 7 to 16:

1
2
5
16
16
3
16
16
Here are the routes of 7 to 15:

1
2
15

There are conflicts between rule 3 and rule 2!
Here are the routes of 7 to 16:

1
2
5
16
16
3
16
16
Here are the routes of 7 to 15:

1
2
15
  
```

图 8 SDN 交换机与网络功能策略检测结果
Figure 8 Results of policy verification between SDN switches and network functions

7.1.2 跨网络功能策略验证

对于跨网络功能的策略冲突, 实验中首先使用 Snort 和 iptables 进行测试。其中, iptables 使用表 5 中的规则, 而 Snort 规则为

- ① alert tcp 10.12.5.109 any -> 12.25.1.1 any (content: baf)
- ② drop tcp 5.2.14.66 any -> 19.48.5.182 any (id:5)
- ③ reject tcp 31.73.244.5 any -> 6.1.53.29 any (id:7)

表 5 防火墙规则 (验证跨网络功能策略)
Table 5 Firewall rules (verifying policies across network functions)

proto	src_ip	src_port	dest_ip	dest_port	action
tcp	10.12.5.109	any	12.25.1.1	any	deny
tcp	192.39.1.0	any	31.73.244.5	any	deny
udp	5.2.14.66	any	19.48.5.182	any	accept

图 9 为跨网络功能策略检测结果。其中，Snort 规则 1 与 iptables 规则 1 产生冲突，Snort 允许通过的数据报被 iptables 策略拒绝；Snort 规则 2 与 iptables 规则 3 产生冲突，iptables 允许通过的数据报被 Snort 策略拒绝；Snort 的规则 3 与 iptables 的规则 2 产生冲突，Snort 的动作是 reject，在拒绝数据报通过之外还要向数据报的发送者发送响应报文，而响应报文被 iptables 策略拒绝，产生了冲突。

```
Here is the conflict list!
IPS_rule 1 and firewall_rule 1 conflict! IPS permits while firewall denies!
IPS_rule 2 and firewall_rule 3 conflict! Firewall permits while IPS denies!
IPS_rule 3 and firewall_rule 2 conflict! Reaction from IPS is bicoked!
```

图 9 跨网络功能策略检测结果

Figure 9 Results of policy verification across network functions

7.2 性能测试

本节对第 6 节提出的策略验证框架进行了原型实现，并进行了性能测试。

(1) OpenFlow 交换机与网络功能策略验证性能

对于 SDN 转发策略与网络功能策略验证，其中转发路径需要构建前缀树，与交换机的实际数量、策略的具体定义以及策略的数目有关；网络功能策略需要两两对比，寻找两条满足约束条件的规则，因此分析时间是 $O(m^2)$ (m 为网络功能策略数)。图 10 为检测结果，用时符合预期：当转发路径策略数目不同时，网络功能策略数量越多，则验证时间越长，且增长时间与网络功能策略数的平方基本呈线性关系。

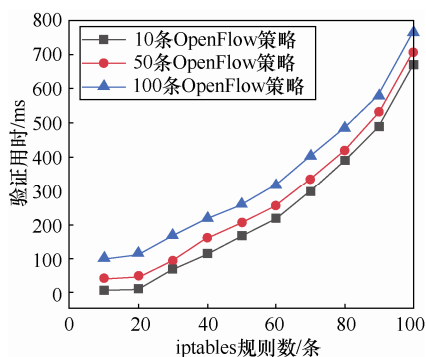


图 10 OpenFlow 交换机与网络功能策略检测开销对比

Figure 10 Comparison of time overhead in policy verification among OpenFlow switches and network functions with policies of different numbers

(2) 跨网络功能策略验证性能测试

对于跨网络功能的策略验证，由于两个网络

功能之间的策略需要两两进行交叉对比，因此算法的复杂度为 $O(mn)$ ，其中 m 、 n 分别代表两个网络功能的规则数目。表 6 为性能测试结果，用 iptables(n)和 Snort(m)，用时与 $m \times n$ 基本呈线性相关，但规则数目过多时，由于冲突数量大幅增加，结果略有偏差。

表 6 跨网络功能策略验证性能测试结果

Table 6 Performance overhead of policy verification across network functions

m /条	n /条	mn /条 ²	用时/ms
10	10	100	2
20	20	400	38
30	30	900	77
40	40	1600	129
50	50	2500	324

7.3 对比实验测试

如上文所述，现有工具并不具备同时检测 SDN 交换机与网络功能之间以及跨网络功能之间策略冲突的验证能力，因此主要对比策略验证性能。选取同样以策略为单位的验证工具 Veriflow^[11]和 Symnet^[9]进行对比，其中 Veriflow 专用于验证 SDN 交换机中的策略是否存在冲突，而 Symnet 则使用符号执行的模型，专用于网络功能中有状态的策略验证。对比实验考虑了策略总数从 0~200 的情形，图 11 为对比实验结果。从结果可以看出，相比 Veriflow 这种只针对 SDN 交换机规则的验证工具，本文工作验证时间更长（如 SDN+FW 曲线所示）；但是对比 Symnet 这种采用了更加复杂的验证方案的工具，本文工作的验证过程更为迅速（如 FW+IPS 曲线所示）。这是由于：① Veriflow 只涉及简单的交换机转发策略，而本文工作涉及网络功能中语义和结构更为复杂的策略，因此时间开销更大；② Veriflow 采用了增量验证方案，将转发策略变成等效类后再进行验证，提高了效率；③ Symnet 可以支持有状态的网络功能，因此功能实现更加复杂。由此可见，策略验证的性能与复杂度受多方面影响，但是本文工作仍然有优化空间，可以考虑加入 Veriflow 的增量策略验证方案，作为未来改进方向。

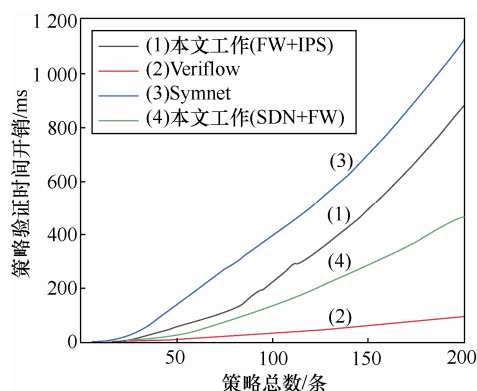


图 11 对比实验结果
Figure 11 Comparison on performance overhead between this work and related tools

7.4 实验结果总结分析

由实验结果可知,策略检测引擎可以正确检测出 SDN 转发策略与网络功能、跨网络功能策略之间的冲突问题;检测的性能符合复杂度预期,检测速度较快。但性能方面仍然具有改进空间,可以进一步优化。

8 结束语

由于 SDN 和 NFV 技术的引入,云环境下基于 SDN/NFV 的网络中,不仅网络结构变得更加灵活、复杂,网络转发策略也变得更加动态。动态策略的引入,出现了不同于传统网络中的动态策略冲突问题。为了进行基于 SDN/NFV 的云环境网络中的动态策略验证,本文分析了网络转发设备与网络功能之间、跨网络功能之间的策略冲突场景与原因,并提出了验证方案;此外,基于 SDN 的网络架构进行扩展,在 SDN 的基础上提出了策略验证框架,并基于实用的 SDN 平台进行了原型实现,支持多种流行的网络功能以及 SDN 交换机中的策略检测,最后在实验中验证了方案的有效性与原型实现的性能。

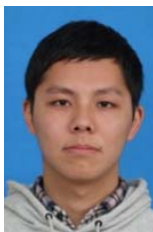
参考文献:

- [1] MANAR J, TARANPREET S, ABDALLAH S, et al. Software defined networking: state of the art and research challenges[J]. Computer Networks, 2014, 72: 74-98.
- [2] Software-defined network[EB].
- [3] RASHID M, JOAN S, JUAN-LUIS G, et al. Network function virtualization: state-of-the-art and research challenges[J]. IEEE Communications Surveys & Tutorials, 2016, 18(1): 236-262.

- [4] 饶少阳, 陈运清, 冯明. 基于 SDN 的云数据中心网络[J]. 电信科学, 2014(8): 33-41.
- RAO S Y, CHEN Y Q, FENG M. cloud data center based on SDN[J]. Telecommunications Science, 2014(8): 33-41.
- [5] 安琪, 刘艳萍, 孙茜, 等. 基于 SDN 与 NFV 的网络切片架构[J]. 电信科学, 2016(11): 119-126.
- ANQ, LIU Y P, SUN Q, et al. Network slicing architecture based on SDN and NFV[J]. Telecommunications Science. 2016(11): 119-126.
- [6] SDN & NFV use cases defined[EB].
- [7] WANG Y, LI Z, XIE G, KAVE S. Enabling automatic composition and verification of service function chain[C]//IEEE/ACM International Symposium on Quality of Service (IWQoS'17). 2017: 1-5.
- [8] PANDA A, ARGYRAKI K, SAGIV M, et al. New directions for network verification[C]//LIPIcs-Leibniz International Proceedings in Informatics 2015: 209-220
- [9] STOENESCU R, POPOVICI M, NEGREANU L, et al. Symnet: scalable symbolic execution for modern networks[C]//Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM'16). 2016: 314-327.
- [10] HORN A, KHERADMAND A, PRASAD M R. Delta-net: real-time network verification using atoms[C]//Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17). 2017: 735-749
- [11] KHURSHID A, ZHOU W, CAESAR M, et al. Veriflow: verifying network-wide invariants in real time[C]//Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI'13). 2013: 15-27.
- [12] SHERRY J, HASAN S, SCOTT C, et al. Making middleboxes someone else's problem: network processing as a cloud service[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 13-24.
- [13] PANDA A, LAHAV O, ARGYRAKI K, et al. Verifying isolation properties in the presence of middleboxes[J]. arXiv preprint, 2014, arXiv:1409.7687.
- [14] HU H, AHN G J, KULKARNI K. Detecting and resolving firewall policy anomalies[J]. IEEE Transactions on Dependable & Secure Computing(TDSC'12), 2012, 9(3): 318-331.
- [15] DENG J, LI H. On the safety and efficiency of virtual firewall elasticity control[C]//24th Network and Distributed System Security Symposium (NDSS'17). 2017.
- [16] MALDONADO-LOPEZ F A, CALLE E, DONOSO Y. Detection and prevention of firewall-rule conflicts on software-defined networking[C]//7th IEEE international Workshop on Reliable Networks Design and Modeling (RNDM'15). 2015: 259-265.
- [17] 邱松, 焦健, 张东阳. 面向防火墙和 IDS/IPS 协同防御的策略冲突检测算法[J]. 系统仿真学报, 2015, 27(11): 2770-2777.

- QIU S, JIAO J, ZHANG D Y. Policy conflicts verification algorithm towards collaborative defense with firewalls and IDS/IPS[J]. Journal of System Simulation, 2015, 27(11): 2770-2777.
- [18] PANDA A, LAHAV O, ARGYRAKI K, et al. Verifying reachability in networks with mutable datapaths[C]//Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17). 2017: 699-718.
- [19] WU W, ZHANG Y, BANERJEE S. Automatic synthesis of NF models by program analysis[C]//ACM Workshop on Hot Topics in Networks (HotNets'16). 2016: 29-35.
- [20] FAYAZ S K, YU T, TOBIOKA Y, CHAKI S, SEKAR V. BUZZ: testing context-dependent policies in stateful networks[C]//13th USENIX Conference on Networked Systems Design and Implementation (NSDI'16). 2016: 275-289.
- [21] PRABHU S, CHOU K Y, KHERADMAND A, et al. Plankton: Scalable network configuration verification through model checking[C]//Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI'20). 2020: 953-967.
- [22] YUAN Y, MOON S J, UPPAL S, et al. NetSMC: a custom symbolic model checker for stateful network verification[C]//Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI'20). 2020: 181-200.

[作者简介]



陈浩宇（1992-），男，江苏扬州人，华中科技大学博士生，主要研究方向为网络安全测试、网络模糊测试、软件定义网络、网络功能虚拟化、软件定义网络安全。



邹德清（1975-），男，湖南长沙人，华中科技大学教授、博士生导师，主要研究方向为虚拟化安全与云安全、网络攻防与漏洞检测、大数据安全、容错计算。



金海（1966-）男，上海人，华中科技大学教授、博士生导师，主要研究方向为计算机体系结构、虚拟化技术、集群计算与云计算、P2P 计算、网络存储以及网络安全。