计算机科学
COMPUTER SCIENCE

# SDN网络边缘交换机异常检测方法

赵扬，伊鹏，张震，胡涛，刘少勋

引用本文

赵扬，伊鹏，张震，胡涛，刘少勋SDN网络边缘交换机异常检测方法[J]．计算机科学, 2023, 50(1): 362-372.

ZHAO Yang, YI Peng, ZHANG Zhen, HU Tao, LIU Shaoxun. Anomaly Detection Method of SDN Network Edge Switch [J]. Computer Science, 2023, 50(1): 362-372.

---

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211100223

# SDN 网络边缘交换机异常检测方法

赵 扬[1] 伊 鹏[1,2] 张 震[1,2] 胡 涛[1] 刘少勋[2]
1 解放军战略支援部队信息工程大学信息技术研究所 郑州 450001
2 网络通信与安全紫金山实验室 南京 210000
(zy169719@163.com)

**摘 要** 软件定义网络(SDN)为网络赋予了可编程性,降低了网络管理的复杂性,促进了新型网络技术的发展。SDN 交换机作为数据转发与策略执行的设备,其权限不应被未经授权的实体窃取。然而,SDN 交换机并不总是执行控制器下发的命令,恶意攻击者通过侵蚀 SDN 交换机对网络进行隐秘而致命的攻击,严重影响用户的端到端通信质量。通信顺序进程(CSP)作为针对并发系统设计的建模语言,可对 SDN 交换机-交换机,以及交换机-主机间的交互进行准确的描述。文中使用 CSP 对 SDN 交换机、终端主机进行建模,对两种异常交换机定位方法进行理论分析,并在实例化的模型系统中验证检测方法在边缘交换机作为出口交换机恶意转发时的有效性,结果表明无法检测该异常行为。针对这一问题,提出了边缘交换机异常检测方法,主机记录统计信息并通过构造特殊的数据包触发 packet_in 消息完成与控制器之间的信息传递,控制器收集统计信息并利用边缘交换机与主机之间的统计信息一致性检测边缘交换机的异常传输行为。最后,基于 ryu 控制器在 mininet 平台上进行实验,实验结果表明,边缘交换机异常检测方法可以成功检测异常行为。

**关键词**:软件定义网络;数据平面安全;形式化认证与分析;通信顺序进程;受损交换机检测
**中图法分类号** TP391

## Anomaly Detection Method of SDN Network Edge Switch

ZHAO Yang[1], YI Peng[1,2], ZHANG Zhen[1,2], HU Tao[1] and LIU Shaoxun[2]
1 Institute of Scientific and Technical Information, People's Liberation Army Strategic Spport Force Information Engineering University, Zhengzhou 450001, China
2 Network Communication and Security Purple Mountain Laboratory, Nanjing 210000, China

**Abstract** Software-defined network gives programmability to the network, reduces the complexity of network management, and promotes the development of new network technology. As a device for data forwarding and policy enforcement, the permissions of SDN switches should not be stolen by unauthorized entities. However, the SDN switch does not always execute the commands issued by the controller. Malicious attackers attack the network covertly and fatally by eroding the SDN switch, which seriously affects the end-to-end communication quality of users. Communicationsequential process(CSP), as a modeling language designed for concurrent systems, can accurately describe the interaction between SDN switch-switch and switch-host. In this paper, CSP is used to model SDN switch and terminal host, and two abnormal switch location methods are analyzed theoretically. We verify the effectiveness of the two detection methods in the instantiated model system when the edge switch is maliciously forwarded as an egress switch, and the authentication results show that the abnormal behavior cannot be detected. In order to solve this problem, an anomaly detection method for edge switch is proposed in this paper. In this method, the host records the statistical information and triggers the packet_in message to complete the information transmission with the controller by constructing a special packet. The controller collects the statistical information and detects the abnormal forwarding behavior of the edge switch by analyzing the statistical information consistency between the edge switch and the host. Finally, based on the ryu controller, experiments are carried out on the mininet platform, and experimental results show that the edge switch anomaly detection method can successfully detect abnormal behavior.

到稿日期:2021-11-22 返修日期:2022-06-06
基金项目:河南省重大科技专项(智能网联汽车内生安全关键技术研究及示范应用 2022012);国家自然科学基金(61872382,62101598,61521003)
This work was supported by the Major Science and Technology Projects in Henan Province(Research and Demonstration Application of Key Technologies for Endogenous Safety of Intelligent Connected Vehicles 2022012) and National Natural Science Foundation of China(61872382,62101598,61521003).
通信作者:伊鹏(yp@mail.ndsc.com.cn)

**Keywords**　Software defined network，Data plane security，Formal authentication and analysis，Communication sequence process，Damaged switch detection

## 1　引言

软件定义网络（Software Defined Network，SDN）自诞生以来就一直受到学术界与工业界的广泛关注。SDN 将控制平面与数据平面分离，为网络管理者提供了集中的网络视图与灵活的可编程能力，大大简化了网络管理的复杂性，并且解决了传统网络环境下网络架构难以拓展的问题，与 5G、云计算、NFV、IOT 等技术交叉融合，促进了新型网络技术的发展。

尽管 SDN 有着许多优势，但是也引入了新的安全威胁，如何确保 SDN 的安全性成为其进一步发展的障碍。SDN 的安全性高度依赖于转发平面的完整性。一方面，控制器拥有的全局视图依靠交换机收集的统计信息来构建；另一方面，网络策略需要通过位于数据平面的交换机的转发行为才能执行。SDN 交换机应该始终能够接受控制器的指令并按照指示安装流规则，其控制权限不应被未经授权的实体窃取，这是 SDN 网络功能实现的基石[1]。

然而，数据平面交换机的权限会被未经授权的实体窃取。首先，SDN 交换机存在漏洞，攻击者能够获取交换机的控制权限[2]。有研究表明，即使重新安装了交换机操作系统，攻击者也能通过入侵交换机操作系统的引导加载程序获得对 SDN 交换机的持久控制权[3-4]；其次，北向通道缺乏保护。保护北向通道的 SSL / TLS 协议为可选协议，多数 SDN 交换机的供应商都没有配置此协议[5]；更糟的是，随着网络规模的不断扩大和软件交换机的不断普及，数据平面交换机出现故障的可能性不断提高，即使不存在恶意攻击者，网络中也可能存在与控制逻辑不同的异常转发行为[6]。

边缘交换机是终端主机在网络中通信的起点与终点，边缘交换机的任何行为都会影响用户的端到端通信。一方面，边缘交换机是终端主机连接网络的唯一接口，终端主机的任何信息都需要经过边缘交换机处理；另一方面，边缘交换机的异常行为无法被网络中的其他转发实体验证，边缘交换机的异常行为会严重影响用户的通信质量。

形式化方法采用严格的数学方法对系统进行描述，可以准确地表达系统性质。通信顺序进程（Communicating Sequential Processes，CSP）[7]是一种对并发系统进行描述的形式化建模语言，它将系统的行为描述为一些并发执行的进程，这些进程相互通信组成了一个系统。

已有许多方法对数据平面异常行为及异常交换机进行检测与定位，根据检测原理进行划分，主要有统计信息验证法与转发路径验证法两种。统计信息验证法通过分析流量统计信息间的一致性是否符合流量守恒原则来对异常交换机进行定位[2]，路径验证法则通过对比分析网络中的真实转发路径与逻辑转发路径间的不一致来定位异常交换机。尽管已经有了许多数据平面异常定位方法，却一直缺乏对数据平面异常定位方法原理的理论分析与形式化认证，尤其是边缘交换机被侵蚀时检测方法的有效性。

受文献[8]的启发，本文提出了新的 CSP 交换机模型与主机模型，实现了 SDN 数据平面 CSP 系统模型，对统计信息验证法与路径验证法的实现原理进行分析并在 CSP 模型系统中进行了有效性验证，验证结果表明现有的检测方法无法检测边缘交换机作为出口交换机时的异常转发行为。为弥补这一缺陷，本文提出了边缘交换机异常检测方法。与现有方法不同的是，该方法将终端主机的统计信息纳入网络一致性检测当中，利用边缘交换机与主机间的流量守恒原则对边缘出口交换机的异常转发行为进行检测。为了实现边缘交换机异常检测，本文解决了两个问题。1）主机如何通过受损交换机向控制器传递信息？本文通过将信息写入 IP 报头的审计字段中，触发 packet_in 消息传递信息。2）由于主机的特殊性，如何设计合理的一致性检测机制以避免误报？本文使用一定间隔内统计信息的增加量进行一致性检测，避免了丢包累计产生的误报。最后，本文基于 ryu 控制器在 mininet 平台上对边缘交换机异常检测方法进行实验，验证了拓扑、丢包率、恶意行为持续时间等因素对检测效果的影响。实验结果表明，在合理阈值下，边缘交换机异常检测方法可以成功检测边缘出口交换机的异常行为。

## 2　相关知识

### 2.1　SDN

SDN 将网络的逻辑控制功能与数据转发功能分离，网络管理者可以对网络进行集中化的网络控制与网络编程。网络智能集中在控制器中，控制器拥有全局视图，对网络中的流量与设备进行管理，并将路由策略转化为具体的流表项下发到相应的转发设备中。转发设备仅拥有少量智能，依照流表项对数据包进行高速转发，并收集控制器构造全局视图所需的统计信息。

### 2.2　CSP

通信顺序进程（Communicating Sequential Processes，CSP）是 Hoare 于 1978 年提出的形式语言，可对并发系统及系统的交互进行精确的描述与分析。CSP 将一个系统表示为描述其行为的一个进程，进程由原子动作和操作符组合而成，进程之间的交互行为被表达为进程之间的通信。CSP 的基本语法如下：

$$P;Q::=Skip\mid a\rightarrow P\mid c?x\rightarrow P\mid c!y\rightarrow P\mid P\mid\mid Q\mid P\square Q\mid P;Q\mid P\lhd b\rhd Q$$

其中，$P$ 与 $Q$ 代表进程，$a$ 是事件，$b$ 为布尔表达式，$c$ 是通道名称；$Skip$ 表示一个进程运行成功并进入终止状态，不再执行任何动作；$a\rightarrow P$ 表示先执行事件 $a$，完成后执行进程 $P$；$c?x\rightarrow P$ 表示从通道 $c$ 上接收一条消息并将内容存储在局域变量 $x$ 中，之后执行进程 $P$；$c!y\rightarrow P$ 表示从通道 $c$ 上发送一条消息 $y$，完成后执行进程 $P$；$P\mid\mid\mid Q$ 表示随机选择进程 $P$ 或进程 $Q$ 中的动作进行执行；$P\square Q$ 表示执行进程 $P$ 还是进程 $Q$ 将由两个进程中的第一个动作与外部环境决定；$P;Q$ 表示进程的顺序组合，依次执行进程 $P$ 和 $Q$；$P\lhd b\rhd Q$ 表示如果布尔表达式 $b$ 为真，那么就执行进程 $P$，反之则执行进程 $Q$。

### 2.3 PAT

PAT(Process Analysis Toolkit)[9]是在 CSP 的基础上设计的模型检测工具,由新加坡国立大学研发。其对 CSP 的语法进行了部分扩展,实现了多种模型检测技术,可对并发、分布式或实时系统进行多种性质的验证,如系统的无死锁性和线性时序逻辑(LTL)描述的性质等。

PAT 与 CSP 的语法基本相同,仅仅在一些符号表达上存在差别。PAT 的一些基本语法如下:

#*define cond* $v==0$:定义了一个名为 *cond* 的命题,该命题内容为变量 $v$ 的值为 0。

#*assert P reaches cond*:定义了一个断言,该断言判断进程 $P$ 在运行过程中是否存在一条执行序列,满足命题 *cond* 中的内容。

## 3 数据平面行为概述

### 3.1 转发行为

位于数据平面的交换机通过匹配控制器下发的流表项对数据包进行处理,对于成功匹配的数据包,按照流表项定义的处理指令处理数据包,并更新流表项的统计字段。通过获取统计字段信息,控制器对全局视图进行更新。对于在流表中没有匹配项的数据包,将会依照 table_miss 表中定义的操作来处理数据包,通常会将数据包封装在 packet_in 消息中转发至控制器,控制器决策出处理结果并将相应流表下发到数据平面交换机中。

OPENFLOW 协议 1.5 版本中规定的流表项组成如表 1 所列,流表项由匹配域、优先级、统计字段、处理指令集等字段组成[10]。匹配域依据数据报文的报文头部信息、数据包的接收入口等信息对数据包进行匹配,统计字段统计该条流表项处理过的数据包的数量或处理过的数据包的字节数等信息,优先级字段定义了流表项匹配的优先级,处理指令集字段定义了对成功匹配的报文进行处理时的动作。OPENFLOW 协议 1.5 版本中定义的交换机可以接收的指令如下:

Group:通过指定的组处理数据包。

Output:将数据包输出到指定端口。

Drop:丢弃数据包。

Push-tag/Pop-tag:压入、弹出标签,如 VLAN,MPLS 等。

Set-Field:修改报文的头字段信息。

Set-Queue:设置报文的队列 ID。

Meter:将数据包定向到指定的计数器。

Copy-Field:在任意长度数据包头部域之间复制数据。

表 1 OPENFLOW 流表项

Table 1 OPENFLOW flow table entry

| 匹配域 | 优先级 | 统计字段 | 处理指令集 |
|---|---|---|---|
| dw_dst=10.3.1.2 | 10 | 39 | output:"eth2" |
| nw_dst=1.0.0.3 | 1 | 17 | drop |
| dw_dst=2.0.0.3 | 1 | 13 | set_field:1.0.0.2->dst |
| dw_dst=1.0.0.2 | 5 | 6 | output:"eth3" |

### 3.2 恶意行为

数据平面交换机无智能,交换机被攻击者侵蚀后,其部分控制权被攻击者获取,交换机的逻辑与功能并没有发生变化。当网络中交换机的流表读写权限被攻击者获取时,攻击者通过修改受损交换机的流表项,可以对经过该交换机的数据包进行恶意转发、恶意丢弃、修改报头信息等操作。

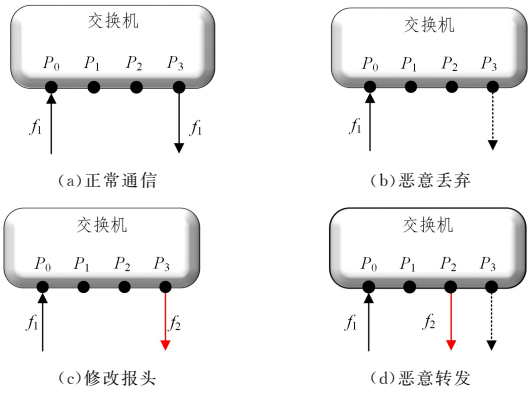


图 1 交换机恶意行为

Fig. 1 Switch malicious behaviors

(1)恶意丢弃

攻击者通过修改目标流表项来丢弃数据包。图 1(b)中控制器下发的流表项为将流 $f_1$ 从端口 $P_3$ 转发,攻击者修改流表项,将流 $f_1$ 的数据包丢弃,导致通信中断。

(2)修改报头

攻击者通过修改目标流表项来修改数据报头。图 1(c)中控制器下发的流表项为将流 $f_1$ 从端口 $P_3$ 转发,攻击者修改流表项,将流 $f_1$ 的数据包报头信息修改为流 $f_2$ 后从端口 $P_3$ 转发。攻击者可以通过这种行为对数据信息进行窃取。

(3)恶意转发

攻击者通过修改目标流表项来转发端口。图 1(d)中控制器下发的流表项为将流 $f_1$ 从端口 $P_3$ 转发,恶意攻击者修改流表项,将流 $f_1$ 的数据包报头信息修改为流 $f_2$ 后从端口 $P_3$ 转发。控制器的转发策略无法得到执行,造成信息泄露,从而影响端到端通信,甚至导致通信中断。

### 3.3 异常定位方法

已有许多工作对数据平面的异常交换机定位方法进行了研究,现有方法主要分为两类:转发路径验证方法与统计信息分析方法。

转发路径验证法通过采集数据包的真实转发路径,并将其与逻辑转发路径进行对比分析来定位异常交换机。SDNsec[3]预先计算需要验证的流的逻辑路径,并在路径的每个交换机上配置流表项,用以更新 MAC(数据包消息验证码)。出口交换机转发数据包至控制器后检查这些数据包的实际转发路径是否与逻辑路径一致。Rev[11]对 MAC 的大小进行压缩,出口交换机只需要向控制器上传流的单个数据包,降低了开销。WedgeTail[12]将转发设备视为几何空间内的点,存储数据包在网络中的转发路径,在采样前使用无监督的基于轨迹的采样机制对转发设备采样优先级进行排序,可以对恶意操作进行区分。Pazz[13]除了使用在网络中存在的流的数据包进行路径采样外,还会随机生成网络中不存在的流的数据包以检测数据平面转发行为,减少了资源消耗。

统计信息验证法利用流量守恒原则,通过收集和分析流量统计来检测转发异常。Ghannam 等[14]在交换机上安装

专用计数器规则，通过检查流量统计数据的一致性来检测恶意交换机。当相邻交换机的对应统计信息间的差值超过阈值时，可以认为网络中存在异常交换机。FADE[15]引入了规则路径的概念，并设计了算法来选择覆盖所有规则路径的最小数量的流，从而减少了配置专用流规则的开销。FlowMon[5]通过收集交换机端口统计信息来检测网络中的数据包是否被恶意丢弃或转发，但其无法检测攻击者精心设计的攻击。FOCES[2]从全局网络流量守恒的角度检测异常，使用流计数器矩阵(FCM)对控制器的视图(即预期的转发行为)进行建模，通过定期收集网络中所有流表项的统计信息，并检查统计信息与由 FCM 构造的流量计数器方程式是否相符来检测异常，不需要安装任何其他额外的专用计数规则就可以检测全网范围内的转发异常。

### 3.4　CSP 的优势

采用 CSP 对 SDN 进行形式化描述，有以下几个优势：

(1)CSP 可以直观、准确地描述 SDN 数据平面中每个对象的行为，并且 CSP 拥有成熟的模型检测工具 PAT(Process Analysis Toolkit)，实现方便。

(2)建模流程简单，模型可重用性高，验证转发行为的交换机模型被提出后，可以根据需求对模型进行更改，模拟其他数据平面行为。

(3)能准确描述特性，验证方法理论。PAT 会对系统的所有运行状态进行验证，验证是否存在一个抵达断言的状态。提出模型后，通过设置合适的断言，可以对数据平面异常定位方法进行理论分析与有效性验证。

## 4　数据平面形式化模型

介绍了 SDN 数据平面相关知识后，本节提出数据平面 CSP 系统模型，可用于对 SDN 数据平面行为进行形式化认证与分析，如受损交换机的恶意转发行为与受损交换机定位方法等。SDN 数据平面由控制器控制的若干 SDN 交换机组成，控制器将流表项下发到交换机中，交换机按照控制器安装的流表项对数据包进行转发、修改、丢弃、更新数据包标签等操作，并修改相应流表项的统计信息。主机产生并接收数据包，通过直连的交换机将数据包注入到网络中。

### 4.1　定义

数据包的格式如表 2 所列，由 SrcHid，DstHid 与 Tag 组成，其中 Tag 为数据包标签字段，在本研究中可以用来存储数据平面的真实转发信息。

表 2　数据包格式

Table 2　Packet format

| SrcHid | DstHid | Tag |
|---|---|---|
| 源交换机 ID | 目标交换机 ID | 数据包标签 |

表 3 列出了模型中用到的表的定义，本文定义的表都是从现实或标准中抽象出的合理的表，加粗的部分为流表项的主键。InterLink 表与 HostLoc 表分别存储交换机间的链路信息以及边缘交换机与主机间的链路信息，共同构成网络的拓扑信息。Flow Table 表存储各个交换机的流表项，记录对匹配流表项的转发、修改、丢弃、更新统计信息、更新标签字段等操作信息。

表 3　表项定义

Table 3　Table item definition

| 名字 | 定义 | 功能 |
|---|---|---|
| InterLink | Sid1 * Pid1 * Sid2 * Pid2 | 存储交换机间的链路信息 |
| HostLoc | Hid * Sid * Pid | 存储网络中的主机位置 |
| Flow Table | InPid * SrcHid * DstHid * Action * OutPort * Status * Value * STag | 存储交换机的流表项 |

### 4.2　数据平面交换机模型

OPENFLOW 协议中规定，SDN 交换机中数据包成功匹配到某条流表项时，首先更新该流表项对应的统计数据，然后依照流表项中的指令对数据包进行转发、丢弃、修改报头、更新标签等操作。由于提出本文模型的目的在于验证数据平面的转发行为，因此没有对交换机与控制器间的交互行为进行建模。本文认为流表项都已经安装到交换机当中，且所有触发 packet_in 的恶意行为都被视为已触发控制器的警报。

按照上述流程，本文提出了交换机模型，该模型中交换机总体建模如下：

$$Switch_x =_{df} PktHandler_x \,||\, (|||_{i\in\{1\cdots PNum-1\}} (PktReceiver_{xi} \,|||\, PktSendler_{xi})) \tag{1}$$

交换机进程 $x$ 由 $PktReceiver_{xi}$，$PktSender_{xi}$ 和 $PktHandler_x$ 模块组成，分别模拟数据包的接受、发送和处理过程。$PktReceiver_{xi} \,|||\, PktSendler_{xi}$ 代表交换机上 ID 为 $i$ 的普通端口上的转发行为，$PktReceiver_{xi}$ 接受数据包并将其发送至数据包处理进程，$PktSender_{xi}$ 发送从数据包处理进程处接收的数据包。在图 2 中，黑线表示交换机内部进程间的交互，蓝线表示交换机间进程的交互，红线表示主机与交换机间进程的交互。

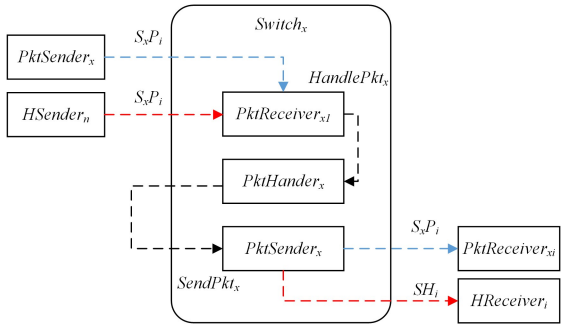


图 2　交换机模型(电子版为彩图)

Fig. 2　Switch model

交换机模型的整体建模如下：

$$PktReceiver_{xi} =_{df} S_xP_i?pkt \rightarrow HandlePkt_x!i.\ pkt \rightarrow PktReceiver_{xi} \tag{2}$$

$$PktSender_{xi} =_{df} SendPkt_{xi}?pkt \rightarrow (index = getSwitchPort(x,i)); (sw = index[0]; pt = index[1]; S_{sw}P_{pt}!\ pkt \rightarrow PktSender_{xi}) \lhd index!\ = EMPTY \rhd (ht = getHost(x,i); SH_{ht}!\ pkt \rightarrow PktSender_{xi}) \tag{3}$$

$$PktHandler_x =_{df} HandlePkt_x?i.\ pkt \rightarrow PktMatcher_x(i, pkt); (TranTag = true; PktHandler_x) \lhd out\_port == Drop \rhd (j = out\_port; SendPkt_{xj}!pkt \rightarrow PktHandler_x) \tag{4}$$

$$PktMatcher_x(i;pkt)=_{df}action=match\_pkt(i,pkt,Table_x);\ UpSta\ (sta_x);\ (UpTag\ (pkt))\triangleleft RcTag==True\triangleright(Skip);(out\_port=Drop)\triangleleft action==drop\triangleright(pkt=pkt\_mod(pkt,action);out\_port=match\_port(action))\triangleleft action==modify\triangleright(out\_port=match\_port(action))) \tag{5}$$

当交换机的某个端口 $i$ 收到一个数据包后，$PktReceiver_{xi}$ 进程将数据包通过内部通道 $HandlePkt_x$ 发送给数据包处理进程 $PktHandler_x$ 进行处理，$PktHandler_x$ 进程每次仅接收一个数据包进行处理。

数据包处理进程 $PktHandler_x$ 所定义的行为如图 3 所示。某个数据包成功匹配流表项后，对统计数据进行更新，随后查阅流表项的指令字段，并按照指令对数据包标签进行更新。当全局变量 $RcTag$ 设定为 true 时，将交换机 ID 添加进标志位 Tag 中；当指令字段为 drop 时，丢弃该数据包；当指令字段为 modify 时，修改数据包报头信息后，将数据包从设置的端口转发；当指令字段为 forwarding 时，交换机将数据包从设置的端口转发。

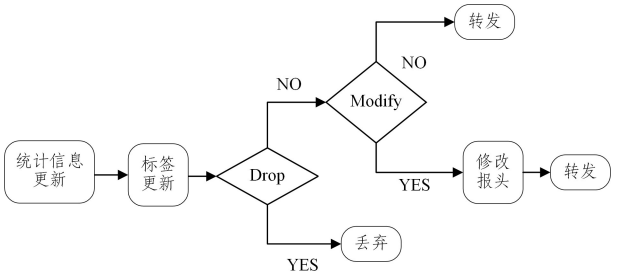


图 3 数据包处理流程

Fig. 3 Packet processing flow

为更好地了解该模型的意义，下面详细介绍 $PktHandler_x$ 进程中使用的一些函数的作用。$match\_pkt(i,pkt,Table_x)$ 函数依照报文头部信息及接收端口 ID 对交换机 $x$ 的流表 $Table_x$ 进行匹配，并将成功匹配到的流表项的操作字段存储在全局变量 $action$ 中；$pkt\_mod(pkt,action)$ 函数依照 $action$ 中存储的信息对数据包进行修改并将修改后的数据包返回；$match\_port(action)$ 函数依照 $action$ 中存储的信息查找数据包的转发端口号并将其存储在全局变量 $out\_port$ 中；$up(sta_x)$ 函数更新交换机 $x$ 的流表项统计信息，将成功匹配的流表项统计信息加 1；$UpTag(pkt)$ 函数将交换机 ID 添加到数据包的标签当中。

$PktSender_{xi}$ 进程通过交换机内部通道取得需要发送的数据包后，将数据包通过下一跳设备的接收通道发送至下一跳交换机。其中，$getSwitchPort(x,i)$ 函数通过查找表 InterLink 获取交换机 $x$ 的端口 $i$ 所连接的交换机及端口 ID，返回存储在一个数组 index 中。该数组的第一个元素为交换机 ID，第二个元素为端口 ID。若与交换机 $x$ 的端口 $i$ 连接的为主机端口，则返回一个空值，并通过函数 $getHost(x,i)$ 查找表 HostTable 获取与交换机 $x$ 的端口 $i$ 连接的主机 ID。

$TranTag$ 是一个布尔变量，用于确保网络中仅存在一个数据包。当网络中没有数据包时 $TranTag$ 被赋值为 True，此时主机可向网络中注入数据包；当网络中存在数据包时，$TranTag$ 被赋值为 False；当数据包被丢弃或者被终端主机接收时，$TranTag$ 会再次被赋值为 True，此时可以传输新的数据包。

数据平面交换机无智能，其功能不会因为场景的变化而变化，被攻击者侵蚀后，其变化也仅仅是其控制权被攻击者获取，流表读写权限被获取后，修改受损交换机的流表即可模拟受损交换机的行为。

### 4.3 主机模型

主机是网络中流量的产生者与接收者，本文中主机模型由 3 个进程组成：主机接收进程、主机发送进程与主机处理进程，模拟主机对数据包进行接收、发送与处理等操作。主机模型中的进程交互如图 4 所示，黑线表示主机模型内进程间的交互，蓝线表示主机与交换机间的进程交互。

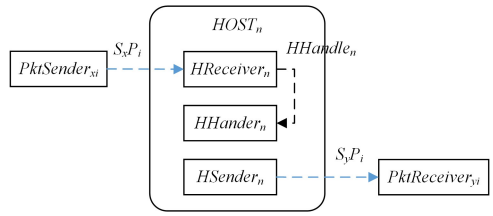


图 4 主机模型(电子版为彩图)

Fig. 4 Host model

主机模型的整体建模如下：

$$Host(x,i,n)=HReceiver_n\parallel HSender_n\parallel HHandler_n \tag{6}$$

$$HReceiver_n=_{df}S_xH_n?pkt?HHandle_n!pkt\rightarrow HReceiver_n \tag{7}$$

$$HSender_n=_{df}(HSender_n)\triangleleft TranTag==true\triangleright(TranTag=false;Hpkt=MakePkt(n);S_xP_i!Hpkt\rightarrow HSender_n) \tag{8}$$

$$HHandler_n=_{df}HHandle?\ pkt\rightarrow(TranTag=true;HHandler_n) \tag{9}$$

其中，$Host(x,i,n)$ 中的 $x,i,n$ 为常量，分别代表与主机相连的交换机 ID、交换机端口 ID、主机 ID。$MakePkt(n)$ 为数据包生成函数，生成由主机 n 产生的数据包。$upHnum(n)$ 更新主机收到的数据包的数量。

$TranTag$ 为一个布尔变量，初始值为 true，当某一主机开始发送数据包后，该布尔值被赋值为 false，此时任何主机均无法发送数据包。当网络中的某一主机接收到数据包，或者交换机丢弃了数据包后，$TranTag$ 被赋值为 true，此时允许主机发送数据包到网络中。这种设计能够确保模型可以模拟真实网络中正常状态下交换机统计信息的变化。

## 5 受损交换机检测原理认证

本节实例化数据平面 CSP 模型，分析现有两种主流异常交换机定位方法的原理，并依照原理在 CSP 模型中设定断言，验证两种受损交换机定位方法能否检测边缘交换机作为出口交换机时的错误转发行为。

### 5.1 原理分析

(1)使用路径验证方法定位异常交换机，即使其探测包生成方式和探测包覆盖范围有所不同。其检测数据平面异常的

核心原理在于利用数据包、网络探针等记录数据包在网络中实际的路由轨迹，将实际的路由轨迹与逻辑路由轨迹进行对比，分析两者之间的不一致定位异常交换机。

了解了原理后，在 CSP 数据平面模型中对其原理进行验证。在本文提出的交换机 CSP 模型中，数据包处理进程 $PktHandler_x$ 在处理数据包时可以根据需要将交换机 ID 增加到模拟数据包标签的 Tag 字段中。当数据包离开边缘交换机后，通过检查 Tag 字段中实际路由信息与逻辑路由信息之间的一致性来定位受损交换机，模拟路径验证方法。

(2)统计信息验证法通过分析网络流量统计信息是否符合流量守恒原则来定位受损交换机。流量守恒原则指网络中同一流在不同交换机的统计信息不应有太大的差异，应当相似或者一致。该方法的核心思想在于通过分析同一流不同交换机间统计信息的一致性来定位受损交换机。

本文提出的交换机 CSP 模型在执行指令前首先更新该流表项的统计信息，这与 OpenFlow 协议中规定的一致。在数据包完成传输后，通过分析各个交换机统计信息的一致性来定位受损交换机，模拟路径验证法。

### 5.2　验证拓扑

本文构建的 CSP 系统模型的验证拓扑如图 5 所示，$S_0$，$S_1$，$S_2$ 为数据平面交换机，其中受损交换机 $S_2$ 位于网络边缘。$H_0$，$H_1$，$H_2$ 为终端主机，$H_1$，$H_2$ 均通过受损交换机 $S_2$ 连接到网络中，其中 $H_2$ 为攻击者控制的主机。网络中存在主机 $H_0$ 到主机 $H_1$ 的单向通信，通信所需的流表都已经配置到交换机中。受损交换机 $S_2$ 的流表($table_2$)被攻击者所修改后，会对数据包进行丢弃、修改转发等恶意行为。

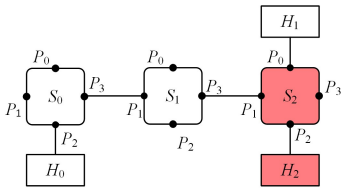


图 5　CSP 系统验证拓扑

Fig. 5　CSP system verification topology

丢弃：如图 6(a)所示，受损交换机 $S_2$ 丢弃由 $H_0$ 发送至 $H_1$ 的数据包。

修改转发：如图 6(b)所示，受损交换机 $S_2$ 将 $H_0$ 发送至 $H_1$ 的数据包的报头中目的地址字段修改为 $H_2$ 后，将数据包由 $P_2$ 端口转发至 $H_2$。

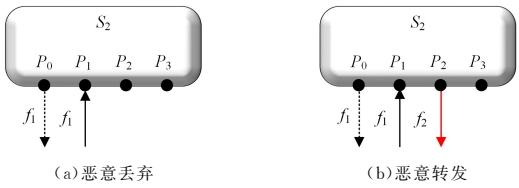


图 6　CSP 系统验证拓扑

Fig. 6　CSP system verification topology

综上所述，完整的 CSP 模型如下，其中 $S_2$ 的流表($table_2$)被攻击者所修改。

$$System =_{df} (\|_{x\in\{0..2\}} Switch_x) \| (\|_{n\in\{0..2\}} Host_n) \quad (10)$$

### 5.3　PAT 中实现

在 PAT 中实现模型系统后，通过设置断言，可以验证系统的状态空间是否存在满足断言的状态。PAT 与 CSP 的语法极其相似，只有一些符号的表达方式有所不同。

PAT 中的一些全局定义如表 4 所列，第一栏为在 PAT 中的定义，第二栏为该定义的意义，第三栏为该定义在 CSP 中的对应项。

表 4　PAT 中表项定义

Table 4　Table item definition in PAT

| 定义 | 含义 | 类比 CSP |
|---|---|---|
| channel SH[Hnum] 0 | 通道，表示边缘交换机到主机的链路 | $SH_u$ |
| channel SP[SNum * (PNum−1)] 0 | 交换机接收主机或其他主机通信的通道 | $S_xP_i$ |
| var InterLink[SNum][SNum] | 链路表，定义交换机间链路连接 | InterLink |
| var HostTable[Hnum][2] | 链路表，定义交换机到主机间的链路 | HostTable |
| var Stable[SNum][5] | 交换机流表，定义交换机流表项 | $Table_x$ |

以数据包接收进程 $PktReceiver_{xi}$ 为例说明 CSP 模型在 PAT 中的实现，如下所示，PAT 中的代码与在 CSP 中的进程定义基本一致，只有通道的格式有所不同。

$$\begin{aligned} PktReceiver(x,i) = {} & SP[x*(PNum-1)+i-1]?pkt \rightarrow \\ & HandlePkt[x]!i.\ pkt \rightarrow PktReceiver \\ & (x,i) \end{aligned} \quad (11)$$

### 5.4　断言

本文在 PAT 中定义了两个断言，用于判断统计分析法与路径验证法能否检测到边缘交换机的异常行为。

(1)统计分析法定位异常

统计分析法通过分析数据平面同一流中不同交换机间统计信息的一致性来定位异常，由于正常通信时网络中的一致性差值不大于 1，故设置断言如下：

$$\begin{aligned} & \|(STable[S_1][f_1]\text{-}STable[S_2][f_1]>2); \\ & \#assertSystemreachesStaCheck; \\ & \#define\ StaCheck \\ & (STable[S_0][f_1]-STable[S_1][f_1]>2) \\ & \|(STable[S_0][f_1]-STable[S_2][f_1]>2) \end{aligned} \quad (12)$$

当交换机间统计信息的差值大于 2，则视为网络中存在异常。

(2)转发路径验证定位异常

转发路径验证法通过实际转发路径与逻辑转发路径之间的对比来定位异常，若网络中存在转发路径不为预设转发路径时，则视为网络中存在异常。将转发路径写入 Tag 中，并与预设的路径进行对比，设置断言如下：

$$\begin{aligned} & \#assert\ System\ reaches\ RouCheck; \\ & \#define\ RouCheck \\ & (Tag[0]!=0 \| Tag[1]!=1 \| Tag[2]!=2) \\ & \&\&RecNum>0; \end{aligned} \quad (13)$$

### 5.5　结果与分析

在 PAT 中对断言进行验证，结果如图 7 所示。断言 RouCheck 与断言 StaCheck 均为无效断言，这意味着两种方法无法检测到受损交换机 $S_2$ 的异常行为。

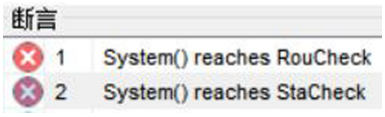


图 7 认证结果

Fig. 7 Verification result

对其原因进行分析，$S_2$ 位于网络边缘，在对数据包进行错误转发之前的行为逻辑均为正常行为。边缘交换机是受影响的流在到达主机前的最后一跳交换机，统计信息分析法需要通过交换机间统计信息的一致性来定位异常。依照交换机的正常处理逻辑，统计信息更新是在流表项中的动作执行前进行，$S_2$ 的统计信息并无异常，不存在不一致。而路径验证法记录的路径仅仅是网络中交换机的路由路径，而无法确定边缘交换机对数据包的后续转发行为。综上，我们需要研究新的方法以检测边缘交换机的异常行为。

## 6 检测方法

在 CSP 模型系统中，路径验证法与统计信息验证法的形式化认证结果表明，现有的数据平面异常交换机检测方法无法检测到边缘交换机作为出口交换机时的异常转发行为，因此，本文提出了新的边缘交换机异常转发检测方法，并在 CSP 模型系统中验证其理论有效性。

### 6.1 问题分析

如上文所述，位于流末端的边缘交换机异常行为无法被检测到的原因在于网络中不存在可以验证边缘交换机转发行为的实体，边缘交换机的转发行为仅自己可知，并且其行为不会影响其他转发设备。

边缘交换机的下一跳为终端主机，边缘交换机与终端主机之间的通信也需要满足流量守恒原则。可以将终端主机纳入网络一致性检测中，终端主机接收的数据包的数量应与边缘交换机发送的数据包数量一致。若两者间存在明显不一致，则可以判定边缘交换机为异常交换机。

但是上述方法有两个问题需要解决：一是主机不与控制器直接连接，如何确保主机的统计信息到达控制器？二是由于主机的特殊性，如何设计合理有效的统计信息一致性检验机制？

问题一 主机与控制器间的信息传递

对于问题一，可以通过 packet_in 机制向控制器传递信息。OpenFlow 规定交换机对于流表中没有匹配项的数据包依照 table_miss 表的配置处理，而 table_miss 表中往往配置为将该数据包封装至 packet_in 消息中并转发至控制器。主机将信息封装至交换机流表中没有匹配项的数据包，通过触发 packet_in 消息向控制器传递信息。

至于如何识别封装了主机统计信息的数据包，可以通过设置特殊的目标 IP 地址进行识别。该 IP 地址由网络管理者设置，不存在于网络中专门用于生成特殊数据包的地址，且网络中交换机均无转发此数据包的流表项。

Packet_in 消息可以封装完整的数据包，也可以仅封装 IP 报头，我们可以通过 IP 报头的标识字段携带信息。IP 报头的标识字段为 16 位随机数，并且交换机不会对其进行身份验证，即使交换机被恶意攻击者侵蚀也无法审查此字段，因此可以利用此字段携带主机统计信息。需要注意的是，若在标识字段中存储整数，统计信息大小不可超过 65 535，若统计数据的大小超过此值，可采用特殊浮点数的形式记录统计数据。

问题二 一致性验证机制的设计

对于问题二，网络中同一流的流量统计信息应当保持一致，同一时间间隔内统计信息的增加量也应该一致。控制器定期查询边缘交换机的统计信息，计算时间间隔 $T$ 内边缘交换机发送的数据包的增加量 $N_{tran}$。主机记录接收的数据包的数量，并定期查询统计信息，计算在时间间隔 $T$ 内主机接收的数据包的数量 $N_{rec}$，构造数据包将 $N_{rec}$ 写入数据包报头的标识字段，发送数据包至边缘交换机。需要注意的是，该数据包在边缘交换机中没有匹配的流表项。控制器得到 $N_{tran}$ 与 $N_{rec}$ 后计算两者之间的差值 $N_{thr}=N_{tran}-N_{rec}$，当 $N_{thr}$ 超过规定的阈值 $Thres$ 时判定边缘交换机存在异常行为，阈值 $Thres=N_{tran}*15\%$。

选用同一时间间隔内的统计信息变化量的一致性进行异常检测，既提高了检测机制的准确性，又有利于机制的实现。若直接使用统计信息的一致性对边缘交换机进行检测，随着网络运行时间的增加，边缘交换机处理的数据量不断累计，链路拥塞等原因导致的数据包丢弃的数量也在累计，只要运行时间足够，正常丢弃的数据包的数量也会到达阈值，此时会导致误报。此外，统计数据的无限增加使得主机向控制器传递信息变得困难，导致误报率更高。

阈值的设置也需要认真考虑，谷歌 B4 SDN 网络中，低优先级数据包的丢包率在 10%左右波动[16]；文献[17]认为正常 SDN 网络的丢包率小于 5%；文献[18]认为网络中的丢包率小于 5%。为确保准确性，我们将阈值设为 $N_{tran}*15\%$。

算法 1 总结了检测异常边缘交换机的方法。

**算法 1** 边缘交换机异常检测算法

```
Input:阈值 thres,时间间隔 T,特殊 ip 范围 P,主机接入表 Hosttable
Output:异常边缘交换机 ID(Sw)
1. / * 所有主机定期发送统计数据包 * /
2. SLEEP(T) / * 间隔时间 T 执行以下行为 * /
3.     主机记录统计信息 sta[h]
4.     计算主机统计信息增加量 Nrec[h]
5.     构造并发送特殊数据包 h. pkt
6. SLEEP(T) / * 控制器间隔时间 T 执行以下行为 * /
7.     控制器收集边缘交换机统计信息
8.     控制器计算边缘交换统计信息增加量 Ntran
9. 监听 packet_in 消息
10. Ifpacket_in. dst. ip∈P
11.     解析主机统计信息增加量 Nrec[h]
12.     Sw←Hosttable[h]. sw
13.     Pt←Hosttable[h]. pt
14.       if Ntran[Sw][Pt]-Nrec[h]>thres * Ntran[Sw][Pt]
15.       return Sw
```

(1)主机统计接收的数据包的数量，定期计算在时间间隔 $T$ 内接收的数据包的数量 $N_{rec}$，并将其存储在特殊构造的

数据包 IP 报头的标识字段中。将数据包发送至边缘交换机，该数据包在边缘交换机中没有匹配的流表项(第 1－5 行)。

(2)控制器以时间间隔 $T$ 定期收集交换机统计信息，计算在时间间隔 $T$ 内交换机发送给终端主机的数据包的数量 $N_{tran}$(第 6－8 行)。

(3)交换机接收到数据包后封装消息至 packet_in 中并将数据包传递至控制器。控制器监听 packet_in 消息，解析特殊目的 IP 的数据包，得到主机统计信息的增加量 $N_{rec}$(第 9－11 行)。

(4)控制器得到 $N_{tran}$ 与 $N_{rec}$ 后，计算两者之间的差值 $N_{thr}=N_{tran}-N_{rec}$，当 $N_{thr}$ 超过规定的阈值 $Thres$ 时判定边缘交换机存在异常行为，阈值 $Thres=N_{tran}*15\%$(第 12－15 行)。

### 6.2　定位方法形式化认证

边缘交换机异常检测方法检测一定时间内新增的统计信息的一致性，但在本文模型中进行形式化认证时不易实现。我们可以直接分析统计信息的一致性，因为在 CSP 认证模型中不存在丢包的情况，统计信息的一致性可以反应出一段时间内增加的统计信息的一致性。对比分析边缘交换机统计信息与主机的统计信息间的差值，当其差值超过阈值，则认为边缘交换机存在异常。

具体断言如下：

$$\begin{aligned}&\#assertSystemreachesH\_Sw\_Check;\\&\#define\ H\_Sw\_Check\\&STable[f_1][S_2]-Hnum[H_1]>2;\end{aligned}\tag{14}$$

### 6.3　认证结果

认证结果如图 8 所示，边缘交换机异常检测方法可以检测边缘交换机的异常转发行为。


图 8　边缘交换机异常检测方法验证结果

Fig. 8　Verification result of edge switch anomaly detection

## 7　模拟与认证

本文在 mininet 上实现了边缘交换机检测方法，并对其进行实验评估和验证。

边缘检测机制能否检测出边缘交换机的异常行为？

拓扑、网络规模、链路丢包率和恶意行为持续时间发生改变时，检测机制的准确性是否会受到影响？

### 7.1　检测机制的实现

本文方法既可集成于控制器中，也可采用增量式部署方式。为了减轻控制器的开销，我们采用增量部署方式，在控制平面与数据平面之间设置控制器代理。控制器代理为简化版本的 ryu 控制器，存储着网络拓扑，交换机向控制器发送的 OpenFlow 消息首先通过控制器代理，代理将从 OpenFlow 消息中提取边缘交换机统计信息与携带信息的 packet_in 消息以作进一步处理。在交互过程中，交换机发送的其他 OpenFlow 消息都直接发送给控制器，由控制器主动发起的 OpenFlow 消息也直接发送给交换机。

图 9 给出了边缘交换机异常检测方法的系统架构。1)边缘交换机信息收集模块定期收集边缘交换机的流量统计信息，并计算边缘交换机在一个时间间隔内新增的统计信息变化量 $N_{tran}$。2)主机信息收集模块监控 packet_in 消息，收集携带主机统计信息的 packet_in 消息，记录主机统计信息变换量 $N_{rec}$。3)一致性分析模块依照主机接入表将不同主机与边缘交换机的 $N_{tran}$ 与 $N_{rec}$ 进行匹配，计算差值 $N_{thr}$，判断一致性差值是否超过阈值，并将超过阈值的交换机 ID 上传至控制器。

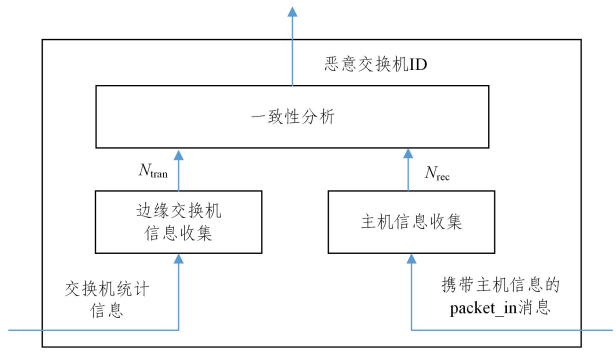


图 9　系统架构

Fig. 9　System structure

本文使用 python 代码实现边缘交换机检测算法，ryu(4.15)作为 SDN 控制器，控制器代理为简化版本的 ryu 控制器，在 mininet 中创建实验所需要的环境与拓扑。实验整体运行在 Ubuntu16.04 虚拟机当中，配置有 4 个 CPU 内核与 4GB 的 RAM。为模拟网络中的正常通信流量，使用 iperf 在一对交换机之间生成固定速率的流量，ryu 控制器按照最短路径算法计算流规则，并将流表项下发到交换机中。为了模拟边缘交换机的异常转发行为，在网络中随机选择边缘交换机，并将交换机中流表项的 action 字段修改为 drop。

### 7.2　功能测试

本文设置了如下的实验来测试边缘交换机异常检测算法能否成功检测出边缘交换机的异常转发行为。分别采用 Fat-Tree(4)拓扑与 ring(5)拓扑进行实验(考虑到实际网络中边缘交换机链接的主机数量有十几个甚至更多，实验采用 ring(5)验证边缘交换机检测方法在边缘交换机接入数量偏多时的检测正确性，每台边缘交换机连接 10 台终端主机)。边缘交换机与主机之间存在持续的稳定流量，检测时间间隔 $T$ 为 5 s，链路的丢包率为 5%。网络正常运行一段时间后(30 s)，依据威胁模型，攻击者持续随机地挑选部分交换机作为目标实施攻击，攻击成功后，恶意交换机随机选择一条流表项改变边缘交换机作为出口交换机时的转发端口，恶意流表项存在 5s。在仿真期间，记录边缘交换机异常检测方法的检测检测率，每次攻击时成功检测出交换机异常即为成功检测，检测率定义为在一次实验中攻击时成功检测出所有异常边缘交换机的次数与所有攻击次数的比率。分别在两种拓扑下进行 100 次实验，实验结果如图 10 所示。从图 10 中可以看出，边缘交换机异常检测方法可以成功检测边缘交换机作为出口交换机时的

异常行为。在 ring(5)拓扑下边缘交换机接入主机数量增多也不会影响检测方法的检测率的原因在于，方法检测原理简单，接入主机数量的增多，增加了检测方法的工作量，但不会影响检测的正确性。

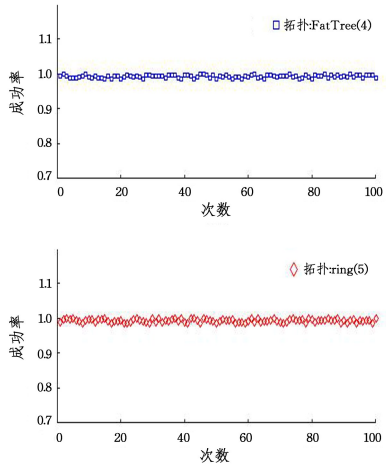


图 10 检测率

Fig. 10 Detection rate

由于本文方法是通过一致性的变化来检测异常，为了更详细地验证边缘交换机异常检测方法检测工作时网络中的一致性变化，设置以下实验。实验采用 FatTree(4)拓扑，边缘交换机与主机之间存在 10 M/s 的持续流量，该段链路的丢包率为 5%。实验总体运行 90 s，设置时间间隔 $T$ 为 5 s，随机选择一个边缘交换机作为攻击目标，实验运行 30 s 时恶意修改交换机目标流表项，实验运行 60 s 时恢复目标流表项，记录 $N_{thr}$ 与 $Thres$ 的大小变化，$N_{thr} > Thres$ 时触发警报。图 11 记录了一次实验中 $N_{thr}$ 与 $Thres$ 数值的变化，图 12 记录了 50 次实验中在 35 s 时产生的 $N_{thr}$ 与 $Thres$ 值的大小。

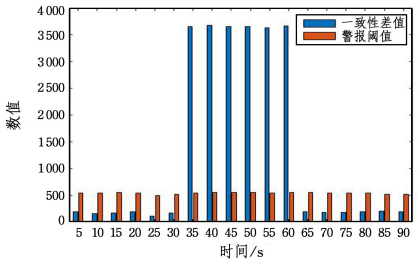


图 11 一致性差值与阈值变化

Fig. 11 Concordance difference and threshold change

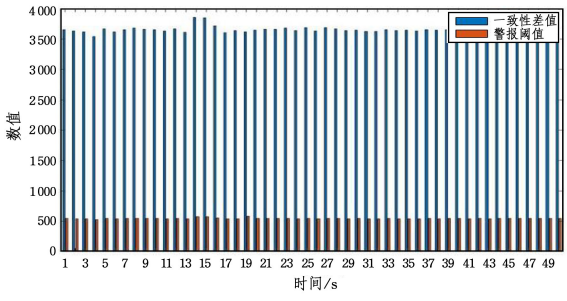


图 12 运行 35 s 时 $N_{thr}$，$Thres$ 的值(50 次实验)

Fig. 12 Value of $N_{thr}$ and $Thres$ when running for 35 s (50 experiments)

从图 11 可以看出，实验开始时，运行 5 s～30 s 时 $N_{thr} < Thres$，此时的 $N_{thr}$ 值由链路丢包产生；当边缘交换机的流表项被恶意篡改时(30 s)，$N_{thr}$ 会迅速增加，超过阈值 $Thres$；当流表项被修改为正常流表项时(60 s)，$N_{thr}$ 又会恢复为较低的状态。由于边缘交换机发送的流量保持稳定，因此阈值 $Thres$ 的值变化较小。由于 $N_{thr}$ 反映的是时间间隔 $T$ 内统计信息的差值，故在恶意转发发生时，$N_{thr}$ 值会迅速增大，而恶意转发行为消失时，$N_{thr}$ 值会迅速减小。

图 12 中每次修改流表项后(30～35 s) $N_{thr}$ 值都会迅速增大，并超过阈值触发警报，证明边缘交换机异常检测机制可以成功检测异常。

### 7.3 拓扑和网络规模

为验证拓扑的影响，分别在相同网络规模(25 个交换机)的线性、网状和树状拓扑结构下进行实验。为验证网络规模的影响，分别在具有不同网络规模( 10，25 和 50 个交换机)的线性拓扑下进行实验。边缘交换机与主机间存在 10 MB 的单向通信流量，链路丢包率为 5% 。实验总体运行 90 s，在 30 s 时恶意篡改边缘交换机处于流末端的转发规则，在 60 s 时恢复边缘交换机流规则。查询统计信息的时间间隔为 5 s。记录边缘交换机转发规则被修改后第一个检测时间 $T$(30～35 s)内边缘交换机与主机的一致性变化，结果如图 13 所示。

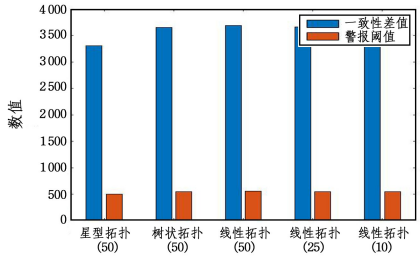


图 13 不同拓扑下 $N_{thr}$，$Thres$ 的变化

Fig. 13 $N_{thr}$ and $Thres$ changes under different topologies

从图 13 可以看出，在不同拓扑与网络规模下边缘交换机异常检测机制可以成功检测，$N_{thr}$ 与 $Thres$ 各自都没有很大的变化。其原因在于 $N_{thr}$ 与 $Thres$ 仅与边缘交换机和主机间链路上的数据量有关，即使拓扑与网络大小发生变化，只要链路中的流量没有变化，$N_{thr}$ 和 $Thres$ 的值就不会发生变化，故边缘交换机异常检测机制的检测结果不会受到影响。

### 7.4 丢包率

拓扑、网络规模不会影响边缘交换机的异常检测，边缘检测机制的关键在于边缘交换机与主机间统计信息的相对变化量的一致性，测量在不同丢包率、不同恶意条目存在时间下边缘交换机与主机之间的一致性。

丢包率可能引起误报，因此本文设置实验来测试在不同丢包率下边缘交换机检测机制的表现。实验采用 FatTree(4)拓扑，边缘交换机与主机之间存在 10 M/s 的单向流量，实验总体运行 90s，不对边缘交换机的转发规则进行篡改，共进行 5 次实验，每次实验的链路丢包率递增(0，5，10，15，20)，记录不同丢包率下 $N_{thr}$ 与 $Thres$ 的值。计算每次实验的 $N_{thr}$ 与 $Thres$，共进行 10 次实验，记录 $N_{thr}$ 与 $Thres$ 的均值，如图 14 所示。

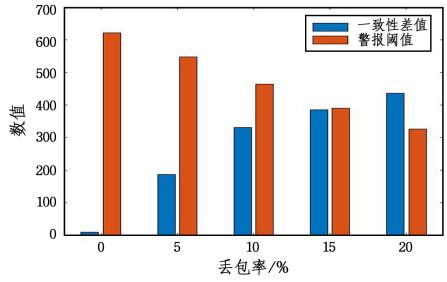


图 14　丢包率与一致性差值

Fig. 14　Loss rate and consistency difference

边缘交换机与主机之间链路的丢包率会影响边缘检测机制的成功率。如图 14 所示，丢包率为 15% 时，$N_{thr}$ 的均值仅略低于 *Thres* 值，随着丢包率的增加，$N_{thr}$ 超过了 *Thres*，此时将会产生误报。因此将警报阈值设为 15%，即丢包率大于阈值(15%)时会产生误报。

随着丢包率的增加阈值 *Thres* 会不断变小，这是因为丢包率提高后边缘交换机处理的数据包会变少，Thres 值与边缘交换机处理的数据包的数量成正比，从而导致 *Thres* 减小。

为验证丢包率与警报阈值一致时是否会产生误报，进行 4 次实验，丢包率与警报阈值一致，分别为 5%，10%，15%，20%，实验运行 90 s，时间间隔 $T$ 设为 5 s，不对流表项进行篡改，记录每个时间间隔 $T$ 内 $N_{thr}$ 与 *Thres* 的值，计算每次实验中 $N_{thr}$ 与 *Thres* 的均值，如图 15 所示。从图中可以看出，当丢包率与警报阈值一致时会产生误报。

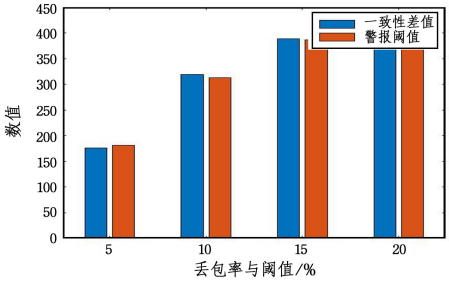


图 15　丢包率与阈值一致时一致性变化

Fig. 15　Consistency changes when packet loss rate is consistent with threshold

### 7.5　恶意流表项存在的时间

本小节测试在不同恶意行为持续时间下边缘交换机检测机制的表现。

实验采用 FatTree(4)拓扑，边缘交换机与主机之间存在 103 M/s 的单向流量，实验开始 30 s 时修改交换机的流表项，记录不同恶意流表项存在时间(0.1 s，0.2 s，0.3 s，0.4 s，0.5 s，0.6 s)下 35 s 处 $N_{thr}$ 与 *Thres* 的值，在每种恶意流表项存在时间下进行 10 次实验，共进行 60 次实验，$N_{thr}$ 与 *Thres* 的均值记录如图 16 所示。从图 16 中可以看出，随着恶意流表项存在时间的增加，$N_{thr}$ 的值会增大，存在时间短于 0.6 s 时不会触发一致性警报，原因在于恶意行为持续时间短，影响的数据包数量少，导致统计信息间的不一致无法到达阈值，故不会被检测出来。

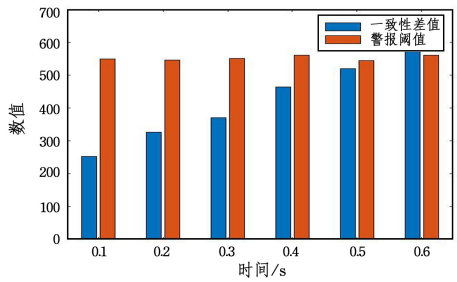


图 16　恶意行为持续时间与一致性差值

Fig. 16　Malicious behavior duration and consistency difference

### 7.6　总结与分析

网络拓扑类型与网络规模不会影响边缘交换机异常检测机制的检测。链路丢包率过高会导致误报，恶意流表项存在时间偏短会导致漏报。

将检测阈值设为 15% 时，过高的丢包率会导致误报，正常 SDN 网络的链路丢包率在 5% 左右，谷歌 B4 网络中低优先级数据包的丢包率也仅在 10% 左右，因此我们认为网络中极少出现因丢包率过高而产生的误报。

15% 的警报阈值下，5 s 内恶意行为持续时间少于 0.6 s 会导致漏报，由于链路中存在持续的稳定的流量，恶意行为的持续时间与恶意行为影响的数据包的数量成正比，故恶意行为影响的数据包的比例较少时不会触发一致性警报。即使出现了漏报，但恶意流表项影响的数据包数量较少，用户仍可以维持较为稳定的通信。

**结束语**　本文提出了新的 CSP 交换机模型与主机模型，在 PAT 中实现了 SDN 数据平面 CSP 系统模型，对统计信息验证法与路径验证法的实现原理进行分析，并在 PAT 中设置两种断言来验证方法在边缘交换机错误转发至终端主机时的有效性，验证结果表明两种方法均无法检测边缘交换机对主机的异常转发行为。针对这一问题，对失效原因进行分析，提出了边缘交换机异常检测方法，终端主机使用不会被交换机进行身份验证的 IP 报头标识字段携带的统计信息，完成与控制器间的信息传递，控制器利用边缘交换机与主机间的流量守恒原则对边缘交换机异常进行检测。该方法可以检测边缘交换机作为出口交换机时的异常转发行为。

本文基于 ryu 控制器在 mininet 平台上对边缘交换机异常检测方法进行实验，实验结果表明：1)通过设计合理的比例阈值，边缘交换机异常检测方法可以成功检测边缘交换机对主机的异常转发行为；2)拓扑种类与拓扑规模不会影响边缘交换机异常检测方法；3)丢包率过高会导致误报，但是在合理的阈值下，发生误报需要链路有很高的丢包率，这种情况很少发生；4)恶意行为持续时间偏短时会导致漏报，但在合理阈值下，恶意行为持续时间过短所影响的数据量较少，不会对通信产生严重影响。

本研究有两个延展方向：1)未来我们可以对不同网络场景下的阈值设置进行研究，阈值设置应当参考网络状态动态变化；2)扩展检测机制，可以对边缘交换机不同的恶意行为进行检测。

## 参 考 文 献


[1] 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.

[2] ZHANG P, XU S, YANG Z, et al. FOCES: Detecting forwarding anomalies in software defined networks[C]// 2018 IEEE 38th International Conference on Distributed Computing Systems(ICDCS). IEEE, 2018: 830-840.

[3] SASAKI T, PAPPAS C, LEE T, et al. SDNsec: Forwarding accountability for the SDN data plane[C]// 2016 25th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2016: 1-10.

[4] CHAO T W, KE Y M, CHEN B H, et al. Securing data planes in software-defined networks[C]// 2016 IEEE NetSoft Conference and Workshops(NetSoft). IEEE, 2016: 465-470.

[5] KAMISIŃSKI A, FUNG C. Flowmon: Detecting malicious switches in software-defined networks[C]// Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense. 2015: 39-45.

[6] YUAN B, JIN H, ZOU D, et al. A practical byzantine-based approach for faulty switch tolerance in software-defined networks [J]. IEEE Transactions on Network and Service Management, 2018, 15(2): 825-839.

[7] HOARE C A R. Communicating sequential processes[J]. Communications of the ACM, 1978, 21(8): 666-677.

[8] XIANG S, ZHU H, WU X, et al. Modeling and verifying the topology discovery mechanism of OpenFlow controllers in software-defined networks using process algebra[J]. Science of Computer Programming, 2020, 187: 102343.

[9] SUN J, LIU Y, DONG J S, et al. PAT: Towards flexible verification under fairness[C]// International Conference on Computer Aided Verification. Berlin, Heidelberg: Springer, 2009: 709-714.

[10] NYGREN A, PFAFF B, LANTZ B, et al. Openflow switch specification version 1.5.1[J/OL]. Open Networking Foundation. https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[11] ZHANG P, WU H, ZHANG D, et al. Verifying rule enforcement in software defined networks with REV[J]. IEEE/ACM Transactions on Networking, 2020, 28(2): 917-929.

[12] SHAGHAGHI A, KAAFAR M A, JHA S. Wedgetail: An intrusion prevention system for the data plane of software defined networks[C]// Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. 2017: 849-861.

[13] SHUKLA A, SAIDI S J, SCHMID S, et al. Toward Consistent SDNs: A Case for Network State Fuzzing[J]. IEEE Transactions on Network and Service Management, 2019, 17(2): 668-681.

[14] GHANNAM R, CHUNG A. Handling malicious switches in software defined networks[C]// NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2016: 1245-1248.

[15] PANG C, JIANG Y, LI Q. FADE: Detecting forwarding anomaly in software-defined networks[C]// 2016 IEEE International Conference on Communications(ICC). IEEE, 2016: 1-6.

[16] SHANG G, ZHE P, BIN X, et al. FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks[C]// IEEE INFOCOM 2017—IEEE Conference on Computer Communications. IEEE, 2017: 1-9.

[17] JAIN S, KUMAR A, MANDAL S, et al. B4: Experience with a globally-deployed software defined WAN[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 3-14.

[18] NAM H, KIM K H, KIM J Y, et al. Towards QoE-aware video streaming using SDN[C]// 2014 IEEE Global Communications Conference. IEEE, 2014: 1317-1322.


**ZHAO Yang**, born in 1997, postgraduate. His main research interests include advanced network and defense technology.

**YI Peng**, born in 1977, Ph. D, researcher, Ph. D supervisor. His main research interests include new network architecture, network security control and active defense technology.

(责任编辑:杨雪敏)