基于拟态防御的 SDN 控制层安全机制研究

丁绍虎, 李军飞, 季新生

国家数字交换系统工程技术研究中心, 郑州 中国 450002

摘要 软件定义网络(Software-Defined Networking, SDN)的集中式管控为网络带来了创新与便利, 但主控制器被赋予了足够的管理权限, 仅依赖其自身内部的防御技术, 难以确保其不发生异常, 以独裁的能力来危害整个网络。本文提出基于拟态防御的 SDN 控制层安全机制, 以一种多样化民主监督的方式, 使用多个异构的等价控制器同时处理数据层的请求, 通过对比它们的流表项来检测主控制是否存在恶意行为。其中, 重点研究了如何在语义层面对比多个异构控制器的流表项, 以解决它们在语法上的差异化问题。该安全机制不依赖于对恶意行为的先验知识, 实验结果验证了它检测恶意行为是有效的, 同时具有较好的性能。

关键词 软件定义网络; 控制器; 拟态防御; 网络安全; 监督
中图法分类号 TP393 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2019.07.06

Research on SDN Control Layer Security Based on Mimic Defense

DING Shaohu, LI Junfei, JI Xinsheng

National Digital Switching System Engineering and Technological Research Center, Zhengzhou 450002, China

Abstract Software-Defined Networking (SDN) brings innovation and convenience to the network benefiting from the centralized management. However, the master controller is given sufficient management authority and relies solely on its own internal defense technology. But this method is hard to ensure that it does not occur anomaly, and the entire network is under threat. We propose an SDN control layer security mechanism based on mimic defense. In a diversified democratic supervision mode, multiple heterogeneous equivalent controllers are used to simultaneously process data layer requests, and main control is detected by comparing their flow entries. We focus on how to compare the flow table items of multiple heterogeneous controllers at the semantic level to solve their grammatical differences. The security mechanism does not rely on prior knowledge of malicious behavior. The experimental results verify that it detects malicious behavior is effective and has good performance.

Key words software defined networking; controller; mimic defense; security; supervision

1 引言

软件定义网络(Software Defined Networking, SDN)提出了控制与转发分离的设计结构, 实现了开放的可编程网络接口, 为网络提供了更细粒度的管理, 引起了学术界和产业界的广泛研究。然而, SDN 的集中式控制在为网络应用带来创新与便利的同时[1], 也带来了安全性问题[2]。主控制器一旦被攻破, 由于目前协议中的主备机制[3]并不能应对拜占庭攻击, 将会危机整个网络的安全。更为严重的是, SDN 网络中的控制器实体是由相应的控制程序和其所部署的计算机构成, 其较高的复杂性也将引入更多的漏洞或后门, 遭受针对操作系统和控制器程序的多重攻击。且 Shalimov A 等人[4]对常见的多款开源 SDN 控制器进行了安全性测试, 包括 NOX、POX、Ryu 等, 结果表明它们都存在着不同程度的安全漏洞, 没有达到电信级的质量标准。

针对此, 研究者提出了多种多样的解决办法, 但均存在一定的缺陷, 可以归纳为以下四类: 1)引入传统防御技术, Zaalouk A 等人在文献[5]提出了一种基于网络监测和控制功能增强 SDN 安全的综合管理架构 *OrchSec*, Hu H 等人在文献[6]中也提出了

通讯作者: 李军飞, 博士, 讲师, Email: lijunfei90@qq.com
本课题得到国家网络空间安全专项(No. 2017YFB0803204); 国家自然科学基金创新群体项目(No. 61521003); 国家自然科学基金项目(No. 61802429, No. 61872382, No. 61702547, No.61502530)资助。
收稿日期: 2018-02-20; 修改日期: 2019-02-26; 定稿日期: 2019-06-06

FLOWGUARD 的安全架构, 它通过在控制层和数据层之间加入检测层, 构建可靠的防火墙。但这些技术需要明确的行为特征作为先验知识, 难以抵御未知的安全威胁; 2)采用安全机制, Jagadeesan N 等人在文献[7]中提出了一种基于多方计算(Multi-Party Computation, MPC)的安全控制框架, Li H 等人在文献[8]中研究了通过拜占庭容错(Byzantine Fault Tolerant, BFT)来提高 SDN 网络的安全性。但这些机制会造成较大的通信开销, 并不实用; 3)优化控制器部署, 文献[9]分别以链路保护和故障恢复为优化目标, 研究了多控制器的位置部署。但这些仅是解决 SDN 网络控制链路的弹性和容错问题。4)采用拟态技术[10], 拟态安全防御主要针对网络空间攻击成本和防御成本的严重不对称性。然而该技术在 SDN 场景中应用时受限于异构控制器的表项语法差异, 不利用拟态裁决输出。

SDN 中的主控制器被赋予了足够的管理权限, 仅依赖其自身内部的防御技术, 难以确保其不发生异常, 以独裁的能力来危害整个网络。为解决该问题, 基于拟态思想, 在 SDN 控制层实现多异构控制器, 共同完成对 SDN 网络的管理。其主要思想是在 SDN 中构建拟态的控制架构, 以一种多样化民主监督的方式裁决输出流表项。例如, 把当前下发的流表与多个等价控制器计算出的流表进行对比, 以民主监督的方式检测主控制器的行为。相比于 BFT, 该安全监督机制并不影响主控制器的功能和性能, 且能够及时预警网络中的安全威胁。另外, 多样性是有效实施拟态防御的基础, 这是由于控制器的异构度越大, 其攻击表面[11]的交集越小, 使得多个控制器由同一漏洞或后门导致同时异常的概率越小, 越有利于相互监督。

并且, 目前 SDN 中的相关协议和技术很大便利地支持实现上述想法, 而这在传统网络中是难以实现的。OpenFlow 规定了标准的南向接口, 便于收集和解析控制器和交换机间的交互数据。OpenFlow v1.2 提出了多控制器协同控制的方法, 等价控制器与主控制器具有相同的管理权限。这就使得在接收到交换机请求或由其他网络事件触发时, 等价控制器同样也可以计算输出的响应, 即 FLOW_MOD 消息, 作为与主控制器下发流表的对比数据。同时, 有许多不同类型的开源控制器可以使用, 如 OpenDayLight、ONOS 等, 且其上便于开发用户应用, 为民主监督的多样化环境提供了物质基础。

然而, 控制器的多样化也随之引入了结果比对的困难, 不能仅在语法层面上进行判决, 尽管它们采用标准的南向协议。具体来说, 即使多个异构控制器输出具有相同功能的流表, 但它们的表达形式(在内容和数量上)也可能存在着不同, 这是由流表编译器或应用具体实现的不同所造成。因此, 需要分析多组流表的语义, 进而判断它们之间是否存在矛盾, 这是基于拟态防御的 SDN 控制器安全机制的研究重点。从网络的本质着手, 将流表的语义表示为网络转发分布图, 借助网络管道图[12]的方法进行高效计算。这样可以将流表的比对转换为转发分布矩阵的比对, 使其具有可行性。进一步, 从对比结果中初步判断当前网络威胁的类型。

本文的主要贡献和创新工作总结如下:

- 提出基于拟态防御的 SDN 控制层安全机制, 以一种多样化民主监督的方式, 使用多个异构的等价控制器同时处理数据层的请求;
- 研究在语义层面对比异构控制器的流表项, 将流表的语义表示为网络转发分布图, 借助网络管道图完成流表比对, 从而检测主控制器是否存在恶意行为;
- 搭建仿真环境, 采用多种异构控制器作为执行体, 并在不同网络场景下仿真实验, 验证所提机制的安全防护性能。

文章的组织结构如下: 第 2 部分介绍当前 SDN 控制器的安全性以及拟态技术; 第 3 部分提出一种基于拟态防御的 SDN 控制层安全机制; 第 4 部分分析对比多个异构控制器流表项存在的难点, 并重点研究基于转发语义的对比算法; 第 5 部分介绍实验环境并分析测试结果; 第 6 部分总结并展望了本文的工作。

2 背景

文献[4]中对常见的 7 种 SDN 控制器进行了安全性测试, 结果如表 1 所示, 它们都存在着不同程度的安全漏洞。例如, 控制器 NOX 在接收到错误长度的 OpenFlow 消息时会发生崩溃, 并且对错误类型、畸形 Packet-in 等异常 OpenFlow 消息并不检测, 存在潜在的安全隐患。然而, 这些仅是被测试到的少数的已知漏洞, 可以通过改进控制器代码质量来弥补, 而更多潜在的未知漏洞却是安全防御的难点。

同时, 注意到不同控制器的安全漏洞不尽相同, 例如, 在接收到长度错误的 OpenFlow 消息时, NOX 会崩溃, POX 会断开与交换机的连接, 而 Ryu 能够成功处理。这就产生一个重要启示: 将多个多样化的异构控制器调度在一起工作, 有利于减少由同一漏洞或后门引起的异常。

表 1 SDN 控制器的安全漏洞[4]

Table 1 Security vulnerability of SDN controller[4]

	错误 长度	无效版 本号	错误 类型	畸形 Packet-in
NOX	崩溃	未检测	未检测	未检测
POX	断开连接	断开连接	未检测	正常
Floodlight	断开连接	未检测	未检测	未检测
Beacon	断开连接	未检测	未检测	未检测
MuL	断开连接	断开连接	断开连接	断开连接
Maestro	崩溃	未检测	未检测	未检测
Ryu	正常	正常	正常	未检测

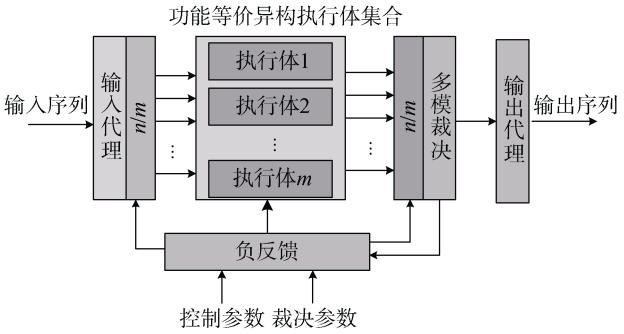

图 1 拟态防御架构

Figure 1 Mimic Defense Architecture

近年来, 邬江兴院士提出的拟态安全防御思想引起相关研究人员的关注。拟态技术的核心思想是通过引入动态异构冗余架构和负反馈机制提升系统应对未知威胁的能力。拟态防御的基础架构如图 1 所示。在该架构中, 输入代理需要根据负反馈控制器的指令将输入序列分发到相应的多个异构功能等价体; 异构执行体集合中接收到输入激励的执行体, 产生满足给定语义和语法的输出矢量; 多模裁决器根据裁决参数或算法生成的裁决策略, 研判多模输出矢量内容的一致性情况并形成输出响应序列, 一旦发现不一致情况就激活负反馈, 直至输出矢量不一致的情况在多模裁决环节消失。因此, 通过将拟态技术引入 SDN 控制器, 可以极大提升 SDN 网络的安全防护能力。

3 基于拟态防御的 SDN 控制层安全机制

拟态防御架构是一种通用技术, 在此将其引入到 SDN 网络中, 功能等价的异构执行体以异构控制器的形式进行实现, 让多个控制器处理网络事件以输出流表, 并获取这些流表进行对比, 同时通过设置控制器代理完成多模裁决和输出。基于拟态防御的 SDN 控制器安全机制如图 2 所示, 在控制层和数据层之间加入代理, 可以容易地获取它们之间的所有交互消息。同时, 在控制层部署多个异构的等价控制器, 设置其接收交换机的请求并输出响应流表。但仅有主控制器的流表被直接下发至交换机, 而等价控制器的流表被截留在代理中, 用作与主控制器的流表进行对比。该机制运行采用一种多样化民主监督的方式, 为了使得监督是公正有效的, 应该保证主控制器和从控制器的实现不同(它们攻击表面[13]的交集更小)但功能相同(响应流表项具有可对比性)。例如, 在 POX 和 ODL 上部署 OSPF 算法, 它们的具有相同的路由功能, 但一个采用 Python 实现, 另一个采用 Java 实现。

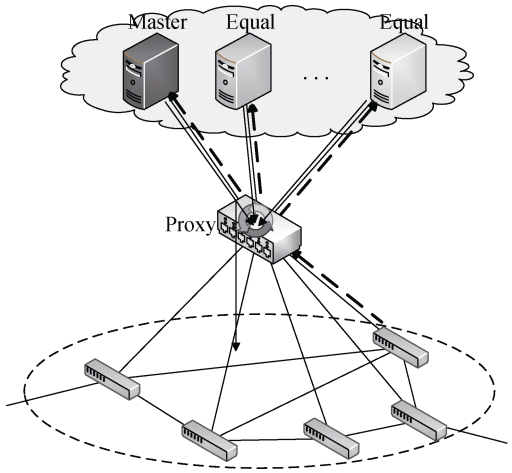

图 2 基于拟态防御 SDN 控制层安全机制

Figure 2 SDN Control Plane Secure Mechanism Based on Mimic Defense

代理的功能也易于实现, 主要包括: 1)消息分发: 当代理捕获到交换机发送的 Asynchronous Messages(通常仅主控制器接收)时, 会将其复制转发给等价控制器, 以使它们能够及时接收交换机的请求或更新网络视图; 2)响应汇集: 代理要汇集和保存控制器发出的控制类消息, 包括 Switch configuration、FlowTable configuration 等。这些消息影响数据层流量的转发, 需要被重点监督; 3)流表裁决: 通过语义对比的方式, 判断主控制器的控制流是否与其他大多数的控制流存在矛盾, 以感知网络中的威胁。其中, 流表判决是代理的核心部分, 将在第四部分阐述其原理和实现算法。

在性能上, 该安全机制无需改动当前 SDN 网络的架构和协议技术, 且对原有网络中的控制器和交换机是无感的, 可以增量部署至有较高安全需求的场景中。代理对于主控制器与交换机来说可以视作网桥, 其引入的通信延迟很小可以被忽略。且监督设

置在主控制器的旁路, 不对其性能产生影响。但整个网络增加了等价控制器和代理的资源开销。

在安全性上, 代理对于控制通道来说相当于网桥, 该部分功能是简单可靠的, 不容易出现故障或被攻破。代理的监督功能设置在控制通道旁路, 即使其出现异常, 也只能使得监督功能失效, 而不会影响网络的安全。另外, 相对于控制器, 代理更加简单, 不需要提供大量的北向接口和部署用户应用, 具有更小的攻击表面。因此, 可以假定代理是安全的。

4 多样化流表项的对比

4.1 问题描述

为了便于用户简洁高效地开发上层应用, 控制器需要对数据层进行抽象, 集成流表规则的自动生成功能, 提供更高级的编程接口。例如, Frenetic[14]提供状态查询、规则组合和一致性更新的 API 抽象, NetCore[15]支持通配符规则、规则的预生成和自定义, FatTire[16]能够自动生成响应链路故障的流表规则和组表规则, 还有 Pyretic、HFT 等。在实验中, 不同控制器集成的流表规则生成模块也不尽相同, 这就是使得尽管在多个异构控制器上层部署相同功能的应用, 且处理相同输入请求, 但其输出的响应流表也无法保证是相同的。

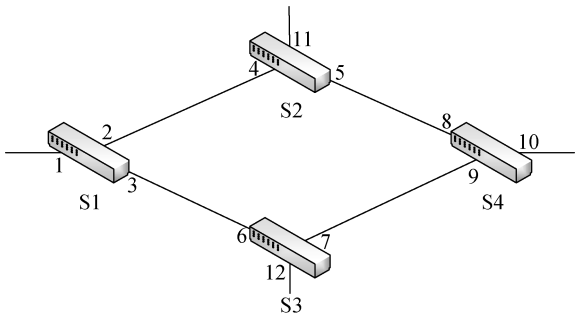

图 3 数据层网络拓扑

Figure 3 Data Plane Network Topology

图 3 中展示一个简单的网络拓扑, 负责数据层网络流量的转发。一个 OpenFlow 流表项可以简单描述为<match, action>两元组, 其中 action 可以是转发、重写、封装、解封装等。对于交换机 S4, 当采用控制器 A 下发的流表项时, 如图 4 中 Table 4_a 所示, 表示将从端口 8 接收到的匹配域为 101xxxxx 和 111xxxxx 及从端口 9 接收到的匹配域为 101xxxxx 和 111xxxxx 的数据包转发到端口 10。而控制器 B 中编译流表的智能化程度较高, 结合网络拓扑对流表进行了压缩, 消除其中的信息冗余, 下发如图 4 中 Table 4_b 所示的流表项。对比上述两者, 尽管流表项的内容和数量均不相同, 但交换机 S4 完成相同的转发功能。语义相同而语法不同, 是采用不同流表编译器导致的问题, 而并不意味着其中某一控制器出现了异常。

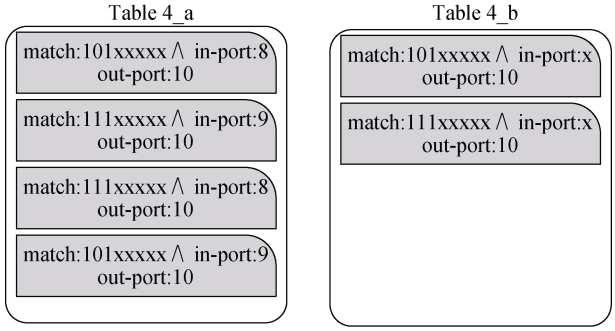

图 4 编译流表的差异

Figure 4 Compiling flow table differences

4.2 基本思想

SDN 网络的本质是转发用户的数据, 无论采用何种形式和方法, 需要确保将某一特征的数据包(可用匹配域表示)送至其目的地。基于此, 可以推理出: 如果多个异构主控制器正常工作, 无论其具体实现的方法如何不同, 无论其生成流表的内容如何不同, 但对于某一特征的数据包, 它被转发的目的地是不应改变的, 即从边缘交换机的出端口是确定的。从整个网络的数据流量来看, 尽管主控制器之间存在 4.1 小节中所述的问题, 但其控制下的边缘交换机转发出数据流量的分布却应是相同的, 这里称之为转发分布图(Forwarding Distribution Map, FDM)。基于此, 可以将流表项组的语义表示为转发分布图, 它表示各入端口进入的流量在各网络各出端口的分布。仅需要关心边缘端口, 也就是不与当前网络中交换机连接的端口, 由于它们是流量最终有效的归宿。综上可以得出只要多个异构的控制器正常且具有相同的功能, 尽管流表项的表示不同, 但它们对应的转发分布图是相同的。

然而, 当网络中的节点有 NAT 功能时, 这些 FDM 也可能存在差异。应该首先隔离这些节点, 将与其相连的端口都作为边缘端口。上述推理对具有 NAT 功能的交换机节点并不成立, 但就目前的应用而言, 该类节点基本都在网络的边缘处(子网的网关、核心骨干网的接入点), 这并不影响后续所述拟态裁决算法的正确性。而对于少数不处于边缘的具有 NAT 功能的交换机节点, 可以将与其有地址转换关联的节点合并在该节点内, 将其转化为边缘点。

4.3 基于转发语义的比对算法

以上述推理为前提, 研究拟态化 SDN 网络的拟态裁决算法, 流程如图 5 所示。首先, 根据汇集的流表项集合, 建立或更新每个控制器的网络管道图。然后, 将网络管道图作为计算“模板”, 生成或更新每个边缘端口的转发分布。最后, 由各端口转发分布构成的矩阵, 对比其中的元素, 判断主控制器是否存在异常行为。接下来将介绍其中的实现细节:

4.3.1 建立网络管道图

HSA 是斯坦福大学于 2012 年提出的用于网络状态建模和规则冲突检测的分析框架[17], 它与网络协议无关, 具有较好的通用性。2013 年, Peyman K 等人对 HSA 做了进一步的改进, 提出网络管道图的概念, 大幅减小其检测规则冲突的计算量。这里引入网络管道图, 主要目的是便于快速计算大量的边缘交换机输出端口的转发分布图, 即创建后续 FDM 的计算“模板”。

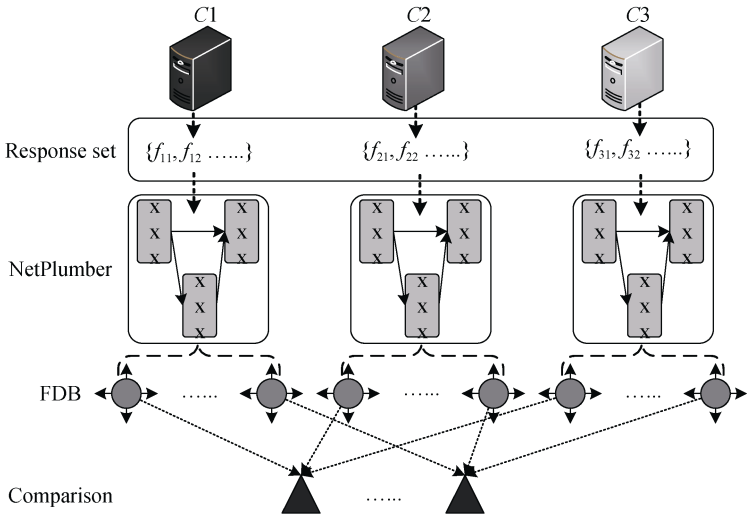

图 5 流表对比的流程

Figure 5 Flow chart of flow table comparison

以图 3 所示的简单网络拓扑为例, 假设其 4 个交换机中的流表如图 6 中的 Table 1–Table 4 所示, 且在每个流表中, 流表项的优先级由上到下顺序降低。首先分析表内的依赖关系, 即高优先级流表项对低优先级流表项的影响。例如, 在 Table 1 中, 两个流表项的入端口都是 1, 且其匹配域存在交集, 则两者存在依赖关系, 需要在管道图中用虚箭头标注, 并注明交集范围 1000xxxx。然后分析相邻交换机中流表项的下一跳关联, 即两个流表项能依次前后处理同一类网络流, 需要满足以下条件: 1)前者的出端口与后者的入端口存在实际的物理连接通道; 2)若前者不修改网络流的报文头部, 则两者的匹配域需要存在交

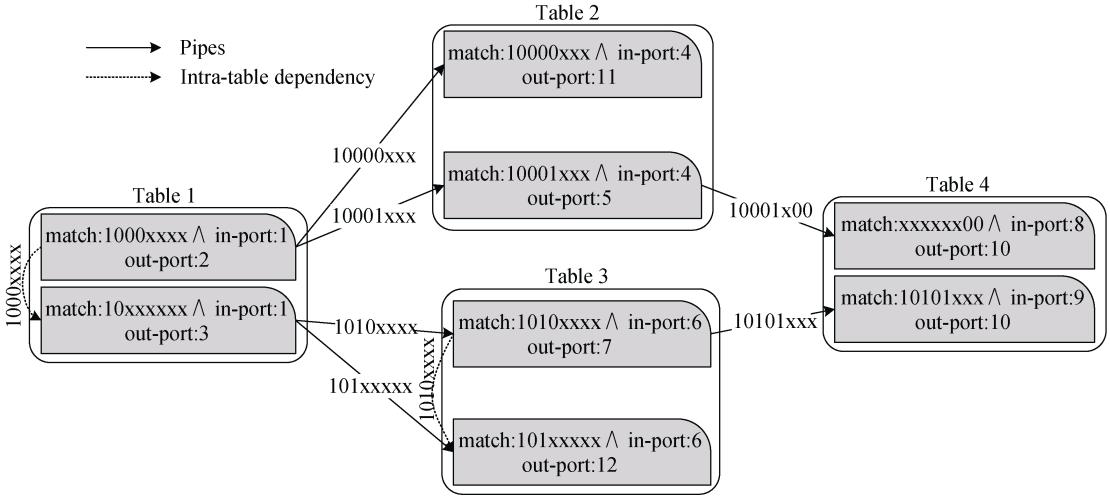

图 6 网络管道图

Figure 6 Network pipeline

集, 否则前者修改后的报头域与后者的匹配域存在交集。例如, Table 1 中的第一个流表项出端口为 2, 与 Table 2 中的第一个流表项入端口为 4, 两者有物理链路连接, 且两者匹配域存在交集, 需要在管道图中用实箭头标注, 并注明交集范围 10000xxx。参照上述两条规则, 分析表内的依赖关系以及相邻表间的管道关联, 建立如图 6 所示的网络管道图。在网络管道图建立之后, 当流表项发生变化, 如增加、删除、修改等, 或网路状态发生变化, 如链路断开、新交换机加入等, 网络管道图也要随之改变。但是, 并不需要依据新的流表项或网络状态重新再计算该网络管道图, 而是仅对其变化的部分做相应的更新, 具体参考文献[12]。

4.3.2 生成转发分布图

转发分布图是分析和标识数据流在网络中的转发流向, 当其报文头部已知, 且进入网络的端口确定, 无论多个“主控制器”编译的流表项有何差异, 则其最终流出网络的端口是相同的。因此, 为了全面检测多个“主控制器”之间的流表项有无冲突, 需要计算所有可能的数据流的转发分布, 即在每个边缘交换机的每个与外部连接的端口处注入报文头为 xxxxxxxx 通配域的数据流, 计算其最终流向。

对于图 3 所示的简单网络拓扑, 端口 1、10、11、12 与外部连接, 需要计算其转发分布图。以端口 1 为例, 如图 7 所示, 在注入报文头为 xxxxxxxx 通配域的数据流后, 报文头为 1000xxxx 的流向 Table 1 中的第一个流表项, 而报文头为 10xxxxxx 的流向第二个流表项。接下来, 数据流按照图 6 中的管道进行流动, 且要考虑表内的依赖关系。例如, 对于从 Table 1 中第二个流表项到 Table 3 中第一个流表项的数据流, 依据管道图的计算为 (10xxxxxx − 1000xxxx) ∩ 1010xxxx = 1010xxxx, 即报文头为 1010xxxx 的数据流流入 Table 3 中的第一个流表项。以此类推, 计算出端口 1 在端口 10、11、12 的数据流分布, 最终结果简化表示为表 2 所示。此外, 由于源端口之间的数据流向无前后依赖关系, 可以并行计算入端口为 10、11、12 的转发分布。

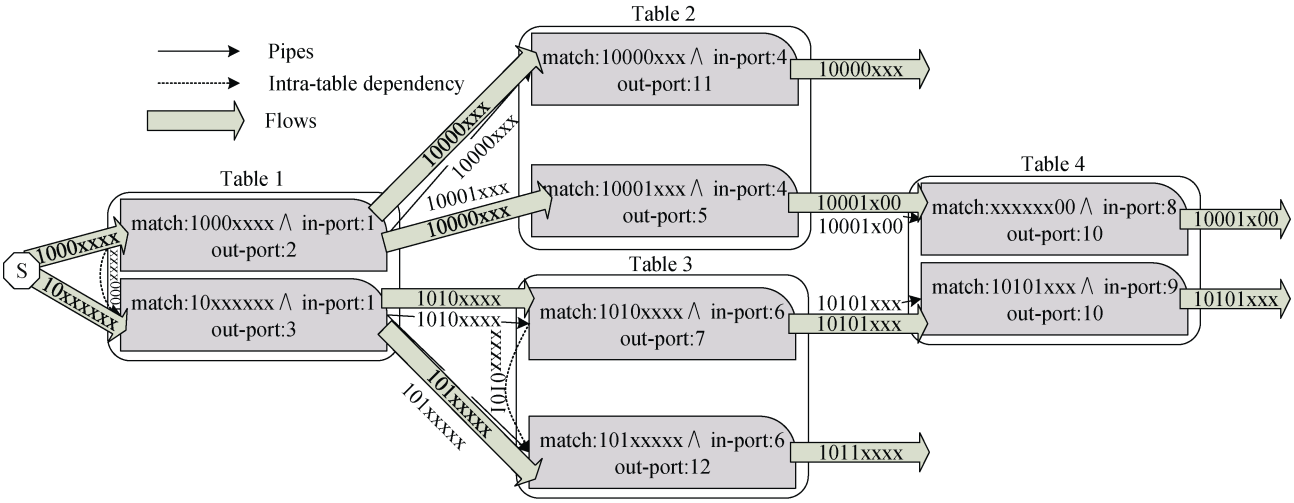

图 7 转发分布的计算

Figure 7 Computation of Forwarding Distribution

表 2 转发分布图

Table 2 Forwarding Distribution Graph

入口端	1	10	11	12
1	-	10001x00∪10101xxx	10000xxx	1011xxxx
10	-	-	-	-
11	-	-	-	-
12	-	-	-	-

同理, 在流表或网络状态发生变化时, 网络管道图随之更新, 这里仅需对相关的小部分数据流做计算, 更新转发分布图, 能够大幅度降低其计算量。

4.3.3 流表对比

通过对比有不同控制器流表项生成的的多张转发分布图, 发现和感知主控制器中的安全威胁。转发分布图的存储结构如表 2 所示, 是一个 $n\times n$ 的矩阵, 记为 f_k, 其中 n 表示数据层中所有与外部连接的端口数目, k 表示第 k 个控制器, 共有 m 个。另外, s_{ij}^{k} 表示 f_k 中的第 i 行第 j 列的元素, 是一个报文头空间的范围。

流表对比就是对 $f_1 - f_m$ 中的数据逐个进行比对, 即 $\forall i,j\in[1,n]$, 若 $s_{ij}^{x}=s_{ij}^{y}$, 则 $f_x=f_y$。这里, 采用严格

判等的方法来做 Compare($s_{ij}^1, s_{ij}^2, \ldots, s_{ij}^m$)运算, 选择 s_{ij}^k 中大多数相同的结果作为标准结果 r_{ij},

$\exists\ a \in [1, m]$, 若 $\forall\ b \in [1, m]$, $card(\{f_x \mid f_x = f_a\}) \geqslant card(\{f_x \mid f_x = f_b\})$, 则 $r_{ij} = s_{ij}^a$。

进一步地, 可以根据裁决结果初步判断出错主控制器的异常类型, 以便网络管理员定位安全威胁。对于出错的主控制器(其转发矩阵为 f_1), 若 $s_{ij}^1 \neq r_{ij}$, 则可简单分以下 3 种情况进行讨论, 这里用 $I(i)$表示 f_1 中第 i 行的入端口号, $O(j)$表示 f_1 中第 j 列的出端口号。

a) $\forall\ q \in [1, n], (s_{ij}^1 - r_{ij} \cap s_{ij}^1) \cap (r_{iq} - r_{iq} \cap s_{iq}^1) = \phi$

表示从端口 $I(i)$处进入的部分流量被额外转发至端口 $O(j)$处, 且该部分流量在正常转发中不应做处理, 意味着数据层中有非正常需求的转发路径存在, 可能有翻墙、规避检查、窃情等安全威胁。

b) $\forall\ q \in [1, n], (r_{ij} - r_{ij} \cap s_{ij}^1) \cap (s_{iq}^1 - r_{iq} \cap s_{iq}^p) = \phi$

表示从端口 $I(i)$处进入的且需转发至端口 $O(j)$处的部分流量被丢弃了, 意味着数据层中有正常需求的转发路径被破坏, 可能存在破坏用户网络、降低传输性能、虚假报文攻击等安全威胁。

c) $\exists q,\ h \in [1, n], (s_{ij}^1 - r_{ij} \cap s_{ij}^1) \cap (r_{iq} - r_{iq} \cap s_{iq}^1) \neq \phi$, 或 $(r_{ij} - r_{ij} \cap s_{ij}^1) \cap (s_{ih}^1 - r_{ih} \cap s_{ih}^p) \neq \phi$

表示从端口 $I(i)$处进入的部分流量, 从端口 $O(q)$被重定向至 $O(j)$处流出, 或从端口 $O(j)$被重定向至 $O(h)$处流出, 意味着网络中正常的转发路径被恶意修改, 可能存在中间人攻击、窃情等安全威胁。

上述严格判等的方法避免了假阴性误判, 也就是 SDN 控制层中如果存在能够反映于流表的相关安全威胁即可被发现, 但假阳性误判的小概率事件仍可能出现。在具体的应用场景中, 可以根据不同的安全防护要求, 制定不同严格程度的 Compare( )方法, 或设计多步裁决机制, 对有嫌疑的主控制器做更深入的分析检测。

4.4 复杂度分析

假设数据层的网络拓扑有 l 个顶点, t 条边, s 个边缘端口, 且每个交换机中的流表项数目平均为 r, 控制层中的控制器数目为 m。此时, 若初始化建立其网络管道图, 则每个交换机内需要对比 $r(r\text{-}1)/2$ 次, 以分析表内的依赖关系, 且对比的次数也是其依赖关系的数目上限(通常会远小于该值)。且每条边会产生 $2r^2$ 次相邻表间的对比, 以计算其关联管道, 同理每条边生成管道数目的上限是 $2r^2$(通常也会远小于该值, 与 r 同数量级)。因此, 初始化网络管道图的计算复杂度为 $O(lr^2+tr^2)$。在生成转发分布图时, 每个边缘端口的数据流最多按照依赖关系和关联管道遍历一遍, 因此, 其计算复杂度为 $O(lr^2s+tr^2s)$。最后, 流表对比仅对比 m 张转发分布图中的各个元素, 因此, 其计算复杂度为 $O(m^2s^2)$。

在流表或网络状态发生变化时, 管道图和转发分布图也要进行更新。以插入一个流表项为例, 则更新管道图的计算复杂度为 $O(rt)$, 随之更新转发分布图的计算复杂度为 $O(rts)$, 而对比转发分布图的计算复杂度保持不变。因此, 插入新的流表项, 流表对比过程的整体计算复杂度为 $O(rts+m^2s^2)$。

通过上述分析, 可以确定该流表对比过程的计算复杂度与网络规模呈多项式级关系, 而在实际计算过程中也往往小于上述理论分析的上限。并且, 上述计算过程存在较大的并行度, 可以利用现在 CPU 多核多线程的优势来进一步加速。因此, 在一定的网络规模下, 该流表对比算法的计算量是可以接受的, 且能够满足实时性要求。

5 实验验证与分析

5.1 实验环境

基于 C++实现了该代理, 部署上述的安全机制应用, 运行在配置为 Intel I7-4790 及 16Gb DRAM 的 Ubuntu 主机上。它与另外的两台主机连接, 分别仿真 SDN 网路中的控制层和数据层。控制层中有三个控制器, 且应用层采用相同的路由算法, OpenDayLight 作为主控制器管理数据层, POX 和 ONOS 作为等价控制器生成的流表项作为参考数据来与主控制器下发的流表进行对比。另外, 用户能够通过 OpenDayLight 的 feature DLUX 下发任意的流表项, 用来模拟主控制器的恶意行为。

在数据层中, 使用 mininet 仿真一个真实的网络: Stanford's Backbone Network[18], 如图 8 所示。为了测

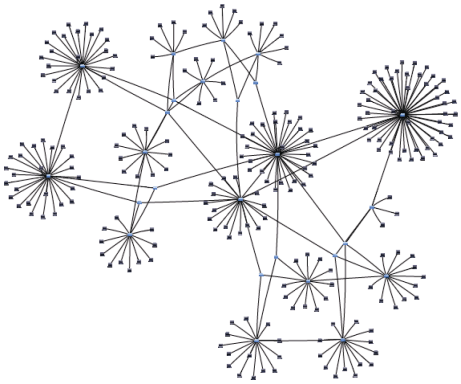

图 8 控制器中可视化的网络拓扑

Figure 8 Visual network topology in the controller

试该监督算法在不同流表规模下的效率, 通过划分子网、设置 VLANs、加入 ACL rules 等方式来改变网络的复杂度。通过批处理指令控制 OVSSwitch 加减链路或加减主机, 使得网络处于动态变化中, 以不断地产生新的 OpenFlow 请求。

5.2 结果分析

5.2.1 准确率

在上述实验平台中, 设计特定的异常流表项, 并通过 feature DLUX 下发给一些交换机, 可以视作主控制器的恶意行为。这些恶意行为分为三类: 1)篡改路径: 修改流表, 将用户的流量重定向至其他出口, 以发起会话劫持; 2)建立旁路: 添加高优先级的流表项, 使得一些流量能够绕过 ACL, 穿透防火墙; 3)切断通路: 删除有正常转发功能的流表项, 造成用户通信的中断。在不同的流表规模下, 对其中的 10 个中心交换机分别测试上述三种恶意行为, 每种恶意行为使用不同的异常流表项测试 20 次。实验中, 主控制器对数据层的传输路径共进行了 6000 次的破坏。

每次破坏均会使得主控制器生成的转发分布图明显与等价控制器的转发分布图不同, 监督代理能够识别异常并发出警告。因此, 该方法对上述测试例可以达到 0 漏报率的效果。

然而, 该监督代理会存在着一些短暂时间的误报。在网络变化较快时, 控制器随之生成的流表速率也会加快, 造成控制器当前检测的流表所对应的网络视图存在差异, 以致产生误报。但随着控制器下发流表的同步, 这些转发分布图很快会趋于一致, 异常警告也会消除。这并不意味着流表裁决算法有瑕疵, 可以通过异常持续时间来区分这类误报。

5.2.2 性能

该监督代理部署在控制通道的旁路, 不会干扰主控制器的控制流, 更不会影响数据层的转发。因此, 它以透明的方式存在, 不改变 SDN 网络原有的传输时延和吞吐率。异常检测时间是评价检测算法的重要指标, 该监督机制中它可以被定义为代理从接收到异常流表项到输出警告的时间间隔。另外, 控制器在一次检测中可能会更新多条流表项, 因此, 检测异常流表项时也涉及与其伴随的正常流表项。

分类统计测试数据, 实验结果如图 9 所示。其中, *X* 轴表示网络的复杂度, 由当前数据层总的流表项规模来反映。*Y* 轴表示异常检测的时间, 是多次试验(不同的交换机或不同的异常流表项)的平均值。同时, 每个图表中包含四个系列的数据, 表示一次异常检测中由主控制器下发的与异常流表项伴随的正常流表项的数量范围。图 9 a,b,c 分别表示三类网络异常, 对应 Section IV part C 中三类流表对比结果.从结果中可以得出: 1)检测时间没有较大的波动, 大致分布在[2*ms*, 9*ms*]范围内, 意味着该检测算法的性能比较稳定; 2)虽然检测时间随着网络复杂度的增加而增大, 但增长幅度较小。这意味着该检测算法具有较好的扩展性, 适用于大规模的网络; 3)伴随的正常流表项越多, 异常检测的时间越大, 意味着增加数据层网络变化速度会降低该检测算法的性能; 4)不同异常类型所需要的检测时间明显不同, 这是由于它们所修改网络管道图中的 Pipe 的数量不同。

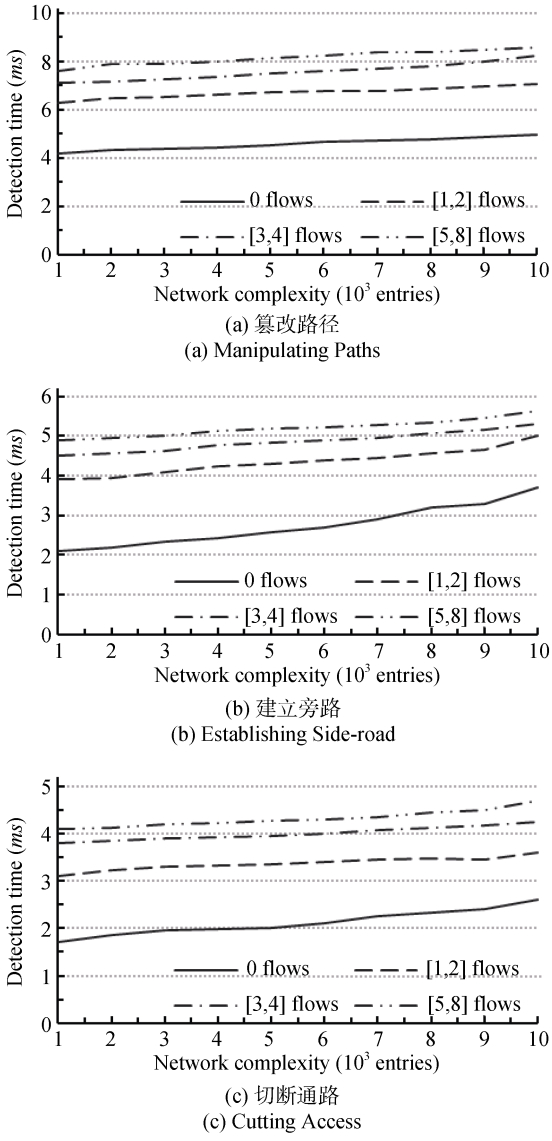

图 9 平均检测时间

Figure 9 Average Detection Time

相对于一些策略一致性检测的方法, 这个安全机制方法更加简单高效。例如, 从数据来看, 相对于 Pisharodys 等[19]提出的 Brew 和文献[20]中的 FlowGuard,

尽管所提方法网络更复杂且流表项更多, 但误报率更低, 且检测时间更短。尽管所提方法网络更复杂且流表项更多, 但误报率更低, 且检测时间更短。总之, 该检测方法不依赖于对恶意行为的先验知识, 通过对比具有多样性的多个控制器流表的语义, 来识别主控制器的恶意行为。实验结果验证了该检测算法的有效性, 而且也表明了它具有较高的实用价值。

6 总结和未来工作

针对 SDN 控制层的安全问题, 本文提出了一种基于拟态防御的 SDN 控制层安全机制, 以一种多样化的民主监督方式进行实施, 采用多个异构的等价控制器同时处理数据层的请求, 通过对比它们的流表项来检测主控制是否存在恶意行为。首先, 设计用于民主监督的代理, 部署在控制通道旁路, 不影响网络的性能。然后, 分析了对比多个异构控制器流表项存在的难点, 并研究了基于转发语义的对比算法。最后, 实验结果验证了该监督机制的有效性, 能及时准确检测出主控制器的恶意行为。在后续研究中, 将对该算法做并行化设计, 使之具有更好的实用性。

参考文献

[1] 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.

[2] Rawat D B, Reddy S R. Software defined networking architecture, security and energy efficiency: A survey[J]. *IEEE Communications Surveys & Tutorials*, 2017, 19(1): 325-346.

[3] Yoon S, Lee J, Kim Y, et al. Fast controller switching for fault-tolerant cyber-physical systems on software-defined networks[C]. *IEEE 22nd Pacific Rim International Symposium on. IEEE*, 2017: 211-212.

[4] Shalimov A, Zuikov D, Zimarina D, et al. Advanced study of SDN/OpenFlow controllers[C]. *Proceedings of the 9th central & eastern european software engineering conference*. ACM, 2013: 1.

[5] Zaalouk A, Khondoker R, Marx R, et al. Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions[C]. *Network Operations and Management Symposium (NOMS)*, IEEE, 2014: 1-9.

[6] Hu H, Han W, Ahn G J, et al. FLOWGUARD: building robust firewalls for software-defined networks[C]. *Proceedings of the third workshop on Hot topics in software defined networking. ACM*, 2014: 97-102.

[7] Agadeesan N A, Pal R, Nadikuditi K, et al. A secure computation framework for SDNs[C]. *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014: 209-210.

[8] Li H, Li P, Guo S, et al. Byzantine-resilient secure software-defined networks with multiple controllers in cloud[J]. *IEEE Transactions on Cloud Computing*, 2014, 2(4): 436-447.

[9] Hu Y, Wendong W, Gong X, et al. Reliability-aware controller placement for software-defined networks[C]. *IFIP/IEEE International Symposium*, 2013: 672-675.

[10] Wu Jiang Xing, Research on mimic defense of cyberspace [J]. *Journal of Information Security*, 2016: 62-82.

[11] Jajodia S, Ghosh A K, Swarup V, et al. Moving Target Defense[M]. *Springer* New York, 2011.

[12] Kazemian P, Chan M, Zeng H, et al. Real Time Network Policy Checking Using Header Space Analysis[C]. *NSDI*. 2013: 99-111.

[13] Ma D , Wang L , Lei C , et al. POSTER: Quantitative Security Assessment Method based on Entropy for Moving Target Defense[C]. *Acm on Asia Conference on Computer & Communications Security*. ACM, 2017.

[14] Foster N, Harrison R, Freedman M J, et al. Frenetic: A network programming language[C]. *ACM Sigplan Notices*. ACM, 2011, 46(9): 279-291.

[15] Schlesinger C, Greenberg M, Walker D. Concurrent NetCore: From policies to pipelines[C]. *ACM SIGPLAN Notices. ACM*, 2014, 49(9): 11-24.

[16] Reitblatt M, Canini M, Guha A, et al. Fattire: Declarative fault tolerance for software-defined networks[C]. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM*, 2013: 109-114.

[17] Kazemian P, Varghese G, McKeown N. Header Space Analysis: Static Checking for Networks[C]. *NSDI*. 2012, 12: 113-126.

[18] Header Space Library and NetPlumber. https://bitbucket.org/ peymank/hassel-public/

[19] Pisharody S, Natarajan J, Chowdhary A, et al. Brew: A Security Policy Analysis Framework for Distributed SDN-Based Cloud Environments[J]. *IEEE Transactions on Dependable & Secure Computing,* 2017, PP (99):1-1.

[20] Zaalouk A, Khondoker R, Marx R, et al. Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions[C]. *Network Operations and Management Symposium (NOMS)*, 2014: 1-9.

丁绍虎 现在国家数字交换系统工程技术研究中心网络空间安全专业攻读博士学位。研究领域为网络空间安全, 拟态防御。研究兴趣包括: SDN、流表分析。

李军飞 于 2018 年在国家数字交换系统工程技术研究中心信息与通信工程专业获得博士学位。现任国家数字交换系统工程技术研究中心助理研究员。研究领域为网络安全、人工智能。研究兴趣包括: SDN、拟态。Email: lijunfei90@qq.com

季新生 现任国家数字交换系统工程技术研究中心教授。研究领域为网络空间安全, 无线通信。研究兴趣包括: SDN、NFV, 5G。