软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(4):969−992 [doi: 10.13328/j.cnki.jos.005020]

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

软件定义网络:安全模型、机制及研究进展*

王蒙蒙, 刘建伟, 陈 杰, 毛 剑, 毛可飞

(北京航空航天大学 电子信息工程学院 信息与网络安全实验室,北京 100191)

通信作者: 王蒙蒙, E-mail: wangmm@buaa.edu.cn

摘 要: 软件定义网络(software defined networking,简称 SDN)初步实现了网络控制面与数据面分离的思想,然而在提供高度开放性和可编程性的同时,网络自身也面临着诸多安全问题,从而限制了 SDN 在很多场景下的大规模部署和应用.首先对 SDN 的架构和安全模型进行分析;其次,从“SDN 特有/非特有的典型安全问题”和“SDN 各层/接口面临的安全威胁”两方面,对 SDN 中存在的典型安全威胁和安全问题进行分析和归纳;随后从 6 个方面对现有 SDN 安全问题的主要解决思路及其最新研究进展分别进行探讨,包括 SDN 安全控制器的开发、控制器可组合安全模块库的开发和部署、控制器 DoS/DDoS 攻击防御方法、流规则的合法性和一致性检测、北向接口的安全性和应用程序安全性;最后对 SDN 安全方面的标准化工作进行了简要分析,并对 SDN 安全方面未来的研究趋势进行了展望.

关键词: 软件定义网络;OpenFlow;安全模型;安全威胁;控制器安全;北向接口安全协议

中图法分类号: TP309

中文引用格式: 王蒙蒙,刘建伟,陈杰,毛剑,毛可飞.软件定义网络:安全模型、机制及研究进展.软件学报,2016,27(4):969−992. http://www.jos.org.cn/1000-9825/5020.htm

英文引用格式: Wang MM, Liu JW, Chen J, Mao J, Mao KF. Software defined networking: Security model, threats and mechanism. Ruan Jian Xue Bao/Journal of Software, 2016,27(4):969−992 (in Chinese). http://www.jos.org.cn/1000-9825/5020.htm

Software Defined Networking: Security Model, Threats and Mechanism

WANG Meng-Meng, LIU Jian-Wei, CHEN Jie, MAO Jian, MAO Ke-Fei

(Laboratory of Information and Network Security, School of Electronic and Information Engineering, BeiHang University, Beijing 100191, China)

Abstract: Software defined networking (SDN) facilitates rapid and open innovation by decoupling the control plane from the data plane, thus enabling high degree of openness and programmability in network protocols and applications. However, the dynamism of programmable networks also introduces new security challenges, which limit the large-scale application of SDN in many places. This paper presents a comprehensive survey on the security of SDN. First, SDN architecture and the security model of SDN are reviewed. Next, typical security threats and security issues of SDN are summarized and classified from the following two aspects: SDN specific and non-specific threats, and the security issues associated with the SDN framework. Then an in-depth analysis is provided on the latest developments in how to build a secure and dependable SDN from the following six aspects: Building a secure SDN controller or network operating system, the modular composable security services for SDN, DoS/DDoS flooding attack prevention and detection for SDN controllers, conflict resolutions and consistency resolutions for flow rules in SDN, the security of northbound application programming interface (API), and the security of applications in SDN. Finally, a brief analysis of the standardization work on SDN security is provided, along with a discussion on future research trends in building more secured SDN.

* 基金项目: 国家重点基础研究发展计划(973)(2012CB315905); 国家自然科学基金(61272501, 61402029, 61370190)

Foundation item: National Key Basic Research Program (973) (2012CB315905); National Natural Science Foundation of China (61272501, 61402029, 61370190)

收稿时间: 2015-05-18; 修改时间: 2015-08-17; 采用时间: 2015-12-22; jos 在线出版时间: 2016-01-05

CNKI 网络优先出版: 2016-01-06 15:15:55, http://www.cnki.net/kcms/detail/11.2560.TP.20160106.1515.001.html

Key words: software defined networking; OpenFlow; security model; security threats; controller security; security protocol of northbound application programming interface

软件定义网络(software defined networking,简称 SDN),将传统封闭的网络体系解耦为数据平面、控制平面和应用平面,在逻辑上实现了网络的集中控制与管理.SDN 的突出特点是开放性和可编程性,目前已在网络虚拟化[1,2]、数据中心网络[3−5]、无线局域网[6−8]和云计算[9,10]等领域得到应用.

逻辑上的集中控制和数据转发分离是 SDN 架构的基本思想,这种思想前期已在学术界引起广泛关注,典型工作包括 ForCES[11]、4D 架构[12]、RCP[13]、SANE[14,15]和 Ethane[16,17]等.SDN 将网络的控制平面从嵌入式节点中独立出来,以开放、可编程的软控制平面代替了传统的基于系统嵌入的控制平面,并由软件驱动的中央控制器来控制整个网络.这种架构有效地简化了网络的管理工作,为用户提供了良好的网络可编程性,但同时也引入了单点失效、南/北向接口协议的安全性难以控制等问题.此外,SDN 动态、开放的可编程架构也间接地为攻击者提供了便捷、有效的攻击手段和方式,使得他们只要通过软件编程的方式便能够轻易地对网络发起攻击,从而降低了攻击者对 SDN 的攻击门槛,使得开发和部署相应的 SDN 安全服务面临着诸多挑战.随着业界对 SDN 架构开发和部署的不断深入,安全性问题逐渐成为制约 SDN 发展的关键因素.

集中管控机制和开放的编程接口虽然增加了 SDN 管理、运营等方面的灵活性,但同时也向攻击者呈现出了难以阻挡的诱惑,使得 SDN 的安全防护面临更多挑战,主要体现在:(1) 集中式的管控架构使得 SDN 的“智慧”集中在控制器上,控制器能够实时地获取网络的全局信息,并通过南向接口将网络配置和安全服务等控制信息下发到交换机和其他网络设备中,从而实现对全网的统一管控.因此,在大多数控制器的设计和开发之初,研究人员主要关注的是资源调度和规则下发等功能,并未将控制器自身的安全问题作为重点研究内容.目前,针对 NOX[18]和 Floodlight[19]等开源控制器,虽然国内外学者已提出部分安全扩展方案[20−23],但这些方案大多针对 SDN 中某个具体的安全威胁进行设计,目前尚无相对完善的控制器安全防护策略.(2) 可编程性是 SDN 实现统一管理、配置网络设备的重要依托,但同时它也为攻击者提供了便捷的攻击渠道.(3) 由于交换机会将无法匹配的流请求信息均发送至控制器,由控制器为其提供相关的应答策略,这种工作模式极大地增加了控制器遭受 DoS/DDoS 攻击的可能性[24−28].此外,由于 SDN 的网络状态和配置信息动态变化,网络中的信息流是否流经某个安全设备以及何时流经该安全设备,这些均由控制器下发的流规则决定.如果攻击者能够伪造来自控制器的流规则,便能够控制网络流量的路径,进而绕开 SDN 中部署的各种安全设备.

本文在分析 SDN 基本架构、工作流程和安全模型的基础上,从 SDN 特有/非特有的典型安全问题和 SDN 各层/接口面临的安全威胁两方面,对 SDN 面临的典型安全威胁进行分析和归纳,并重点对 SDN 安全问题现有的解决思路和最新进展进行总结和探讨,以期为 SDN 网络的安全性研究提供参考.

本文第 1 节对 SDN 的基本架构和安全模型进行分析.第 2 节对 SDN 面临的典型安全问题和安全威胁进行总结和归纳.第 3 节探讨并分析 SDN 安全问题的主要解决思路及其最新研究进展,并对各类方案的安全性和特点分别进行归纳.第 4 节对 SDN 安全的标准化工作进行简要分析.第 5 节总结全文,并对 SDN 安全方案未来的研究方向进行展望.

1 SDN 架构与安全模型

针对 SDN 带来的网络安全模型变化问题,在 RSA 2014 会议上,Check Point 公司的代表 Hinden 从物理设备无法主动执行安全操作、应用层和控制层之间的信任关系匮乏等方面对其进行了简要分析,并提出了全网统一的安全策略、控制与安全相关的信息流、隔离受感染的主机等应对方法[29].本节首先对 SDN 的基本架构和基于 OpenFlow 的 SDN 基本工作流进行探讨,在此基础上,将 SDN 安全模型与传统网络安全模型进行对比,并从网络信息流的控制方式和网络安全态势信息的感知方式两方面,对 SDN 给网络安全模型带来的冲击和变化进行分析.

1.1 SDN架构简介

SDN架构的核心思想是逻辑上集中控制和数据转发分离,基于OpenFlow协议的网络架构初步实现了SDN的原型设计思想,最初萌芽于斯坦福大学的Clean Slate项目组,是SDN技术的典型部署实例[30,31].随着SDN概念的不断推广,不同的研究机构和标准化组织分别从用户和产业需求等角度出发,提出了SDN的其他参考架构,如欧洲电信标准化组织(European Telecommunications Standards Institute,简称ETSI)从网络运营商的角度出发,提出了NFV架构[32];思科、IBM、微软等设备厂商和软件公司从SDN的具体实现和部署的角度出发,共同提出了OpenDaylight[33].目前,开放式网络基金会(Open Networking Foundation,简称ONF)[34]作为业界非常活跃的SDN标准研究机构,正致力于SDN的发展和标准化,并对SDN的定义、架构和南/北向接口规范等内容不断地加以完善.

如图1所示,ONF提出的SDN架构主要分为基础设施层、控制层和应用层[35].基础设施层由网络底层的转发设备组成,主要负责数据的处理、转发和状态收集.控制层集中维护网络状态,一方面,它通过自身与基础设施层之间的接口获取底层基础设施信息,对数据平面的资源进行编排;另一方面,它对全网的拓扑和状态等信息进行实时维护,并为应用层提供可扩展的编程接口.应用层位于SDN架构的顶层,主要包括不同类型的业务和应用.此外,按照接口与控制层的位置关系,ONF分别定义了SDN架构中的南向接口和北向接口.在南向接口方面,ONF定义了开放的OpenFlow协议标准[36].然而,由于应用层中各类业务和应用的复杂性和多样性,控制层与应用层之间的北向接口目前尚无统一的规范和标准.

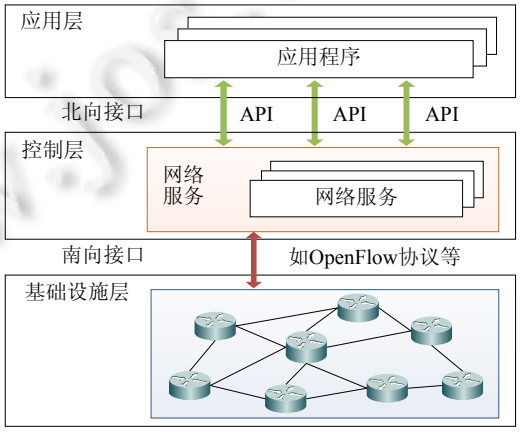


Fig.1 SDN architecture[35]
图1 SDN架构图[35]

1.2 基于OpenFlow的SDN工作流程

基于OpenFlow协议的SDN架构初步实现了SDN的原型设计思想,是SDN技术的典型实例.如图2所示,在基于OpenFlow协议的SDN网络中,当终端*A*初次和终端*B*进行通信时,其基本通信流程大致包括7步[37].

Step 1. 终端*A*加入网络,并向交换机1发送数据包.

Step 2. 交换机1查询自身的流表,若流表中没有与该数据包匹配的表项,则交换机1通过Packet-In事件将该数据包转发给控制器.在向控制器发送消息时,交换机1可以通过TCP协议直接将数据包发送给控制器,也可以采用安全传输层协议(transport layer security,简称TLS)[38]对数据包进行加密传送.

Step 3. 控制器收到交换机1的请求信息后,生成相应的应答策略,并通过Packet-Out事件下发至交换机1的指定端口.

Step 4. 交换机1执行控制器下发的应答策略,将数据包转发至交换机2.

Step 5. 若交换机2的流表中无该数据包匹配项,与Step 2处理方式相似,交换机2将通过Packet-In事件把收到的数据包信息转发给控制器;若交换机2的流表中含有匹配项,则跳转至Step 7,即交换机2按流表中相应

的转发规则将数据包转发至终端 *B*.

Step 6. 与 Step 3 相似,控制器根据交换机 2 的请求信息,下发相应的应答策略至交换机 2 的指定端口.

Step 7. 交换机 2 执行控制器下发的应答策略,数据包被转发至终端 *B*.

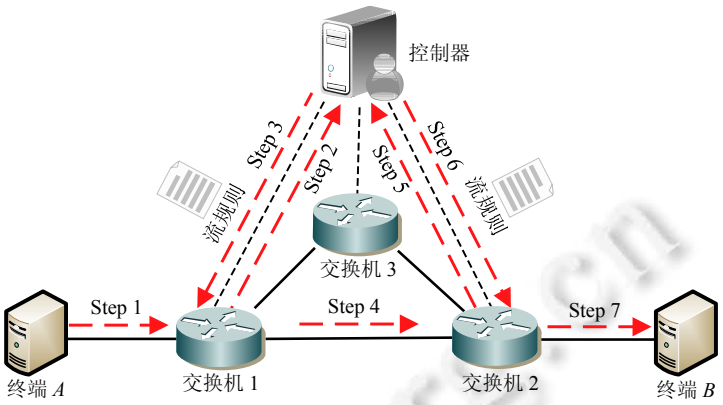


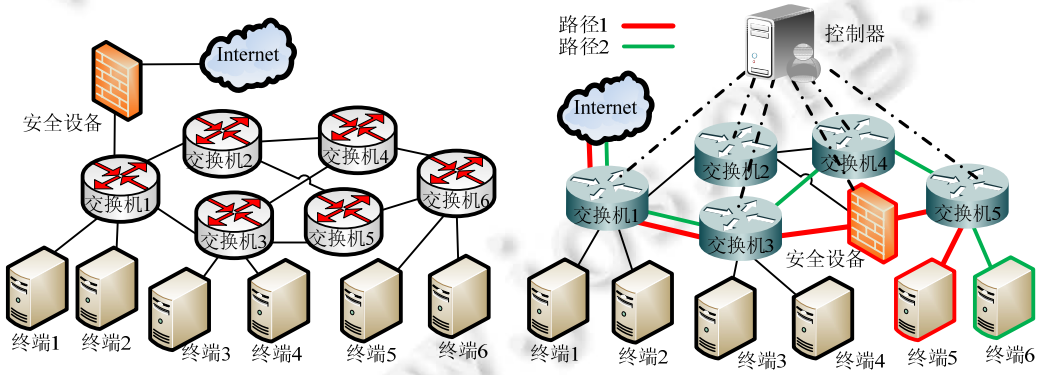
Fig.2 Working principle of OpenFlow-based SDN

图2 基于 OpenFlow 的 SDN 工作流程

1.3 SDN安全模型与传统网络安全模型对比

由图 2 可知,集中管控使得控制器对网络流量具有极强的控制能力,在各项配置策略的细粒度、实时推送方面具有独特优势,但这种工作模式也给网络的安全模型带来了较大冲击,主要体现在两个方面.

(1) SDN 与传统网络对信息流的控制方式不同.在传统网络环境中,如图 3(a)所示,防火墙等安全设备被部署在网络的关键位置,信息流被强制性地从这些安全设备中流过,以便安全设备可以对其进行实时监控和检测.相对于传统网络,SDN 是一个流规则驱动型网络.SDN 中的信息流是否流过某个安全设备以及何时流过该安全设备,均由控制器下发的流规则决定,物理的安全设备自身并不具有决定权.如图 3(b)所示,若控制器下发的流规则指定某些数据包的转发方式按路径 1(红色路径)执行,因路径 1 中的数据包流经 SDN 的安全设备,所以安全设备可根据相关的安全策略对数据包进行检查.相反地,若数据包的转发方式按路径 2(绿色路径)执行,则它们便能够绕过 SDN 中的安全设备,从而导致预先部署的安全防护措施失效.



(a) 传统网络安全模型图 (b) SDN 架构下的网络安全模型

Fig.3 Security model of traditional network and security model of SDN

图3 传统网络安全模型与 SDN 安全模型对比

(2) SDN 与传统网络对网络安全态势信息的获取方式不同.在传统网络中,如图 3(a)所示,当管理员需要获取当前网络的安全态势信息时,由于网络缺乏统一的控制中心,管理员需要同时向多个设备发送状态请求信息,在对收到的状态信息进行综合评估后,才能得出网络当前的安全态势信息.而在 SDN 中,如图 3(b)所示,控制器

作为整个网络的“指挥控制中心”,已为整个网络建立了全局视图,能够实时获取全网的各种状态信息.因此,网络的安全态势信息可以直接从控制器中轻易获取.然而,SDN 这种便捷的网络态势信息获取方式也极易被攻击者利用,使他们能够轻易地获取到网络的安全状态信息,并伺机发起大规模的攻击.

2 SDN 典型安全问题分析

随着业界对SDN架构开发和部署的不断深入,安全性问题逐渐成为制约SDN发展的关键因素.本节首先对SDN 特有和非特有的典型安全威胁进行总结和分类,在此基础上,对 SDN 各层和南、北向接口面临的主要安全问题分别进行探讨和分析.

2.1 SDN特有/非特有的典型安全问题

针对 SDN 中存在的各种安全问题,国内外学者分别从控制器的脆弱性[22,24,25,39−47]、流规则的合法性和一致性[20,44,48−53]、南向接口 OpenFlow 协议的脆弱性[37,41,54−58]、北向接口的安全性及标准化问题[20,22,44,59−65]等方面进行了初步研究和分析.此外,Kreutz 等人[66]还分别对部分 SDN 特有和非特有的安全威胁进行了描述.在 Kreutz 等人国内外已有研究成果的基础上,本节总结和归纳了 SDN 中特有和非特有的 8 种典型安全威胁及其主要表现形式(见表 1),并对南向接口协议 OpenFlow 的安全性进行分析和探讨.

(1) SDN 特有/非特有的典型安全威胁分类

Table 1 Eight main potential threat vectors in SDNs

表 1 SDN 中 8 种典型的安全威胁

名称	是否 SDN 特有	描述/主要表现形式
流规则的合法性和一致性问题	是	(1) 控制器上同时运行着多个自定义或第三方提供的应用程序,这些应用程序生成的流规则之间可能出现相互竞争、彼此冲突或覆盖的情况 (2) 流规则在下发时,由于时延或被攻击者恶意篡改等原因,使得控制器和交换机流规则不一致 (3) 不同控制器之间缺乏有效、安全的流规则同步方案
控制器的脆弱性	是	(1) SDN 中最严重的威胁、故障或恶意的控制器可使整个 SDN 网络受到威胁 (2) 常规 IDS 技术很难发现 SDN 中某个具体攻击的发起者,尚不足以保证 SDN 的安全
控制器和应用程序之间缺乏信任机制	是	(1) 控制器和应用程序之间缺乏有效的信任评估和信任管理机制 (2) 验证网络设备是否安全的技术和验证应用程序是否安全的技术并不相同 (3) 恶意应用程序可以轻易地被开发,已授权的合法应用程序也可能被篡改,并应用于控制器上
控制层-基础设施层之间的威胁(如 OpenFlow 协议的安全性)	是	(1) 主要指南向接口协议面临的安全威胁,如 DoS/DDoS 攻击或数据窃取等 (2) TLS/SSL 加密的基础是 PKI,不足以保证交换机与控制器之间的安全通信,部分文献已分析了 TLS/SSL 协议的安全问题[67,68]
管理站的脆弱性	否	(1) 攻击者可利用管理站的脆弱性,接入到 SDN 的控制器 (2) 管理站的脆弱性在传统网络中也存在,但由于 SDN 的集中管控方式,导致这种脆弱性可迅速扩展至整个网络
交换机的脆弱性	否	某个交换机受到攻击后,使网络中的数据包出现部分异常,若攻击者通过受攻击的交换机向控制器或其他交换机发送虚假请求,可将威胁迅速扩展到全网
取证和修复等缺乏有效的可信任资源	否	SDN 的故障修复需要安全和可靠的取证机制和可信任的资源,以确保网络的快速恢复
伪造/虚假的网络数据	否	由非法用户或设备产生,如伪造的流规则等

(2) OpenFlow 协议安全性

OpenFlow 协议是 ONF 标准化组织唯一确定的 SDN 南向接口通信规范,目前协议版本已更新至 OpenFlow1.5.1[37,69].依据 OpenFlow 协议,控制器和交换机之间通过安全通道进行连接,安全通道采用安全传输层协议 TLS 对消息进行加密和认证.参照 TLS 1.2 协议标准[38],控制器和交换机建立 TLS 连接的基本认证过程如图 4所示.

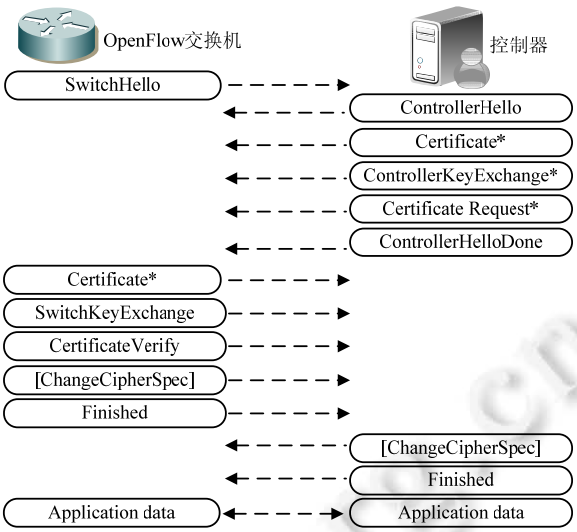


Fig.4 SSL/TLS communication example of OpenFlow switch-controller
图4 OpenFlow 交换机-控制器 SSL/TLS 通信实例

当交换机启动时,首先尝试连接至用户指定的TCP端口或控制器的6653 TCP端口,双方通过交换数字证书相互进行认证.同时,每个 OpenFlow 交换机至少需要配置两个证书,一个用于认证控制器的合法性,一个用来向服务器证实自身的合法身份.交换机和控制器证书的生成和分发过程如下[37,54].

(1) 由证书管理机构生成站点范围内的证书;

(2) 生成控制器证书;

(3) 生成交换机证书;

(4) 使用站点范围内的私钥对证书进行签名;

(5) 将密钥和证书分别发放到各个设备中.

由于 TLS 协议的认证过程涉及多次握手及信息确认步骤,操作较为繁琐,因此,继 OpenFlow 协议 1.3.0 版本之后,新版本的 OpenFlow 协议将 TLS 协议设为可选的选项,即允许控制通道不采取 TLS 加密处理.近年来,虽然有关 SDN 的安全方案不断被提出,但 OpenFlow 协议仍面临着一些重要的安全问题.

(1) 安全通道可选.由于 OpenFlow 1.3.0 版本之后的协议将 TLS 设为可选的选项,这使得缺乏 TLS 协议保护的 SDN 网络非常易遭到窃听、控制器假冒或其他 OpenFlow 通道上的攻击.

(2) TLS 协议本身的脆弱性.OpenFlow 协议 1.5.1 之前的版本并未指定 TLS 加密使用的参考规范和协议版本号,版本号的不一致或错误也可能会导致一些交互操作的失败,给 SDN 带来一些新的安全问题[56].同时,TLS 协议自身的脆弱性也使得 OpenFlow 协议面临着中间人攻击等安全隐患[67,68].此外,TLS 协议虽然能够增加交换机和控制器之间通道的安全性,但却无法阻止交换机修改流规则.

(3) 缺乏多控制器之间通信的安全规范.现有的 OpenFlow 协议仅给出了控制器和交换机间的通信规范,但并未指定多个控制器之间通信的具体安全协议和标准,因而多个控制器之间的通信仍面临着认证、数据同步等方面的安全问题.

2.2 SDN各层/接口面临的安全问题

依据控制与转发分离的逻辑架构,可将 SDN 面临的安全问题分为 5 个方面,如图 5所示.

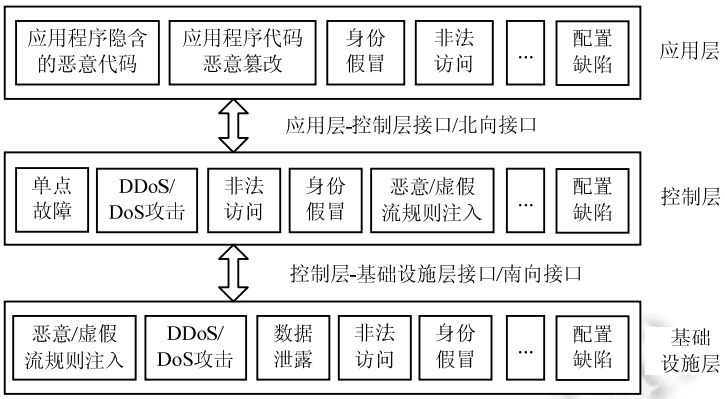


Fig.5 Main threats in each layer of SDN
图5 SDN 各层面临的主要安全问题

(1) 应用层安全

应用层主要包括各类应用程序.在 SDN 中,除管理员制定的流规则外,一些流规则还将由 OpenFlow 应用程序、安全服务类应用程序和一些其他的第三方应用程序制定,并通过控制器下发至相关的交换机和网络设备.目前,针对应用程序自身的安全性保护机制并不健全,由于基础设施层的各种交换机和网络设备对控制器下发的流规则完全信任,且不假思索地执行,一旦这些参与制定流规则的应用程序受到篡改和攻击,将给 SDN 带来难以预估的危害.因此,应用层面临的安全威胁主要包括:应用程序隐含的恶意代码、应用程序代码的恶意篡改、身份假冒、非法访问以及应用程序自身的配置缺陷等.

(2) 控制层安全

控制器是 SDN 的核心,也是安全链中最薄弱的环节.SDN 通过控制器对网络进行集中管控,接入到控制器的攻击者,将有能力控制整个网络,进而给 SDN 带来难以预估的危害.控制层的典型安全问题是集中式管控带来的控制器单点故障问题,主要体现在两个方面:① DoS/DDoS 攻击:攻击者制造一系列非法的访问致使控制器产生过量负荷,从而导致控制器系统资源在合法用户看来是无法使用的.由于新的流请求到达交换机之后,交换机上若没有与之匹配的流规则,便会将整个数据包或数据包的头部转发给控制器,由控制器来制定相应的应答流规则.因此,一些攻击者会利用 SDN 中的一些交换机向控制器发起大量虚假的请求信息,导致控制器负载过重而中断合法交换机的请求服务.② 控制器在逻辑上或物理上遭到破坏:这主要是指 SDN 中的关键控制器在物理上或逻辑上遭到破坏,致使用户的合理请求服务被拒绝.此外,控制层面临的安全威胁主要还包括非法访问、身份假冒、恶意/虚假流规则注入以及控制器自身的配置缺陷等.

(3) 基础设施层安全

基础设施层由交换机等一些基础设备组成,主要负责数据的处理、转发和状态收集,对控制器下发的流规则绝对信任.因此,该层面临的主要安全威胁包括:恶意/虚假流规则注入、DDoS/DoS 攻击、数据泄露、非法访问、身份假冒和交换机自身的配置缺陷等.此外,基础设施层还可能面临着由虚假控制器的无序控制指令导致的交换机流表混乱等威胁.

(4) 南向接口安全

这主要是指由 OpenFlow 协议的脆弱性而引发的安全性威胁.OpenFlow 安全通道采用 SSL/TLS 对数据进行加密,但由于 SSL/TLS 协议本身并不安全[38,67,68],再加上 OpenFlow 1.3.0 版本之后的规范均将 TLS 设为可选的选项,允许控制通道不采取任何安全措施,因而南向接口面临着窃听、控制器假冒等安全威胁.

(5) 北向接口安全

北向应用程序接口(northbound application programming interface,简称 Northbound API)的标准化问题已成为 SDN 讨论的热点[61−63,70−73].由于应用程序种类繁多且不断更新,目前北向接口对应用程序的认证方法和认证

粒度尚没有统一的规定.此外,相对于控制层和基础设施层之间的南向接口,北向接口在控制器和应用程序之间所建立的信赖关系更加脆弱,攻击者可利用北向接口的开放性和可编程性,对控制器中的某些重要资源进行访问.因此,对攻击者而言,攻击北向接口的门槛更低.目前,北向接口面临的安全问题主要包括非法访问、数据泄露、消息篡改、身份假冒、应用程序自身的漏洞以及不同应用程序在合作时引入的新漏洞等.

此外,针对 SDN 各层的安全问题,Scott-Hayward 等人[74]也给出了相应的分类方法,见表 2.

Table 2 Categorization of the security issues associated with the SDN framework by layer/interface affected[74]

表 2 SDN 架构中的安全问题(按相关层和接口)[74]

安全问题/攻击	SDN 中受影响的层或接口				
	应用层	应用层与控制层之间的接口	控制层	控制层与数据层之间的接口	数据层
未经授权的访问等					
未经授权的控制器访问			√	√	√
未经身份认证的应用程序	√	√	√		
数据泄露等					
流规则发现(侧信道攻击)					√
转发策略发现(包处理时序分析)					√
数据修改					
修改数据包的流规则修改机制			√	√	√
恶意应用程序					
虚假规则注入	√	√	√		√
控制器劫持			√	√	
拒绝服务攻击					
控制器交换机之间通信的洪泛攻击			√	√	√
交换机流表的洪泛攻击					√
配置问题					
缺少 TLS 协议或其他认证技术			√	√	√
策略生成问题	√	√	√		

3 SDN 安全机制

针对 SDN 中存在的各种典型安全问题,现有解决方案的主要思路可分为 6 类:(1) SDN 安全控制器的开发;(2) 控制器可组合安全模块库的开发和部署;(3) 控制器 DoS/DDoS 攻击防御;(4) 流规则的合法性和一致性检测;(5) 北向接口的安全性;(6) 应用程序安全性.本节主要对上述 6 类解决方案的主要思路、原理和特点进行探讨和分析.

3.1 SDN安全控制器的设计与开发

控制器是 SDN 的核心,它对整个网络的状态和拓扑等信息进行集中管控,一旦受到攻击,会导致 SDN 网络的大面积瘫痪.因此,完善的安全机制对 SDN 控制器而言尤为重要.目前,学术界和商业界已开发出的控制器功能各具特色,种类繁多.按照控制层的基本架构,可将现有控制器分为集中式和分布式两类.其中,集中式控制器主要包括 Floodlight[19,75]、NOX[18]、POX[76]、Beacon[77]、Maestro[78]、Meridian[79]、MuL[80]、NOX-MT[81]、PANE[82]、ProgrammableFlow[83]、Rosemary[84]、Ryu NOS[85]、OpenIRIS[86]、Trema[87]等,分布式控制器主要包括 Onix[88]、OpenDaylight[33]、ONOS[89]、DISCO[90]、Fleet[91]、HP VAN SDN[92]、HyperFlow[93]、Kandoo[94]、NVP Controller[95]、SMaRtLight[96]、yanc[97]等.

大多数 SDN 控制器在设计和开发之初,主要关注的是网络资源的调度和控制问题,如链路发现、拓扑管理、策略制定和表项下发等方面,基本没有将控制器自身的安全问题作为核心研究内容.随着 SDN 的不断推广,攻击者对控制器的攻击手段和攻击方式逐渐增多,集中式管控带来的扩展性[42]、单点故障[24]等缺点不断凸显.因此,安全控制器的开发和设计逐渐引起了学术界和产业界的广泛关注.目前,在 SDN 安全控制器的设计和开发方面,现有的研究思路可以归纳为两类:演进式安全控制器的开发和革命式安全控制器的开发.

3.1.1 演进式安全控制器

演进式安全控制器的主要研发思路是在现有开源控制器的基础上,改进已有的安全服务或开发部署新的安全模块.目前,已有诸多研究者对 Floodlight、NOX、POX 等控制器进行安全功能方面的扩展,并设计和开发了相应的安全架构和安全应用,如 FortNOX[20]、SE-Floodlight[22,64]等.

FortNOX 架构是 Porras 等人针对开源控制器 NOX 设计的一种安全内核.如图 6所示,FortNOX 在 NOX 控制器上分别增加了基于角色的数据源认证模块(role-based source authentication module)、状态表管理模块(state table manager)、流规则冲突分析模块(conflict analyzer)和流规则超时回调模块(flow rule timeout callback)等安全功能.其中,基于角色的数据源认证模块主要用于对每个流规则进行签名,并为候选流规则指定相应的特权类别;状态表管理模块主要对 NOX 控制器总流表中流规则的插入、删除等操作进行管理;流规则冲突分析模块可以根据 NOX 控制器当前总流表中流规则的状态,对每一个候选流规则进行评估;若流规则冲突分析模块提供的检测结果显示某个候选流规则与总流表中的流规则无冲突,则该候选流规则将被顺利转发给相应的 SDN 交换机,同时被更新到 NOX 控制器的总流表中,并由状态表管理模块对其进行管理.当 SDN 交换机的流规则过期时,FortNOX 将启动流规则超时回调接口模块,对 NOX 控制器的总流表进行更新.

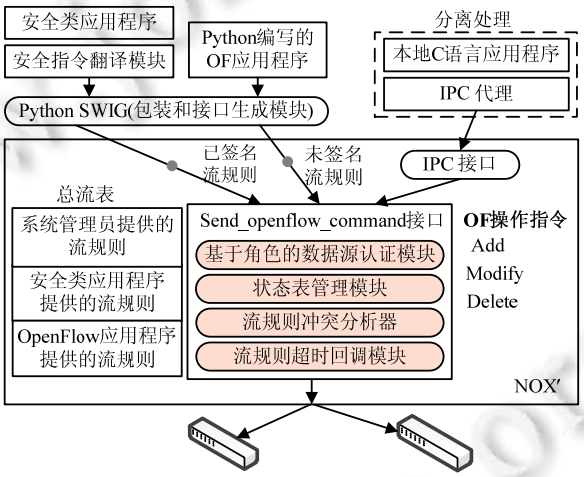


Fig.6 FortNOX implementation[20]

图6 控制器安全内核 FortNOX 架构示意图[20]

FortNOX 安全内核是对 NOX 控制器安全性能的有效扩展,通过为 NOX 控制器提供基于角色的数据源认证、状态表管理、流规则冲突分析、流规则超时回调等安全功能,使得 NOX 控制器在流规则冲突检测方面的安全性能得到了较大提升.在 NOX 控制器安全内核研究的基础上,Porras 等人对开源控制器 Floodlight 也进行了类似的安全扩展,设计了 Floodlight 控制器的安全增强版本 SE-Floodlight[22,64].如图 7所示,SE-Floodlight 控制器继承了 FortNOX 安全内核在数据源认证、状态表管理、流规则冲突分析管理和流规则超时回调管理等方面的基本设计思想,并分别对这些模块的安全服务和安全功能进行了扩充和完善.此外,SE-Floodlight 增加了应用程序证书管理模块(app credential management)、安全审计子系统(security audit subsystem)、权限管理(permission mediator)等新的安全模块.其中,安全审计子系统主要负责对与控制器安全相关的事件进行审计和跟踪,如应用程序的身份认证与授权、流规则修改、流规则冲突处理、交换机配置信息变更、控制器配置信息

变更、审计系统的启动与关闭等;应用程序证书管理模块主要对与控制面交互的应用程序及其身份凭证进行审核与认证;权限管理模块主要对数据面和控制面之间除流规则以外的交互消息进行管理,该模块仅允许由管理员授权的应用程序访问控制面资源.

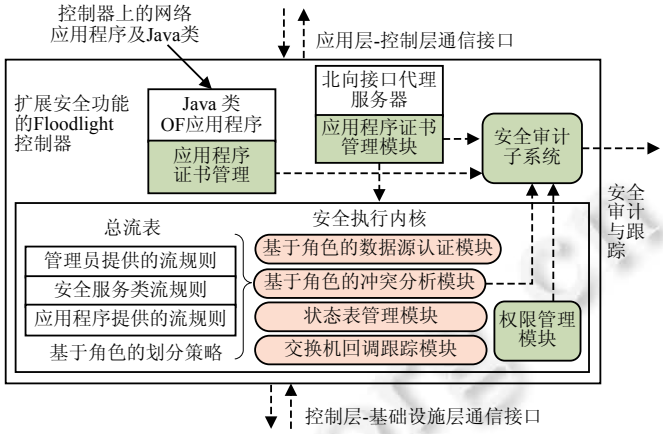


Fig.7 Overview of the SE-Floodlight architecture[22]
图7 SE-Floodlight 基本架构示意图[22]

在 SDN 的安全机制研究方面,目前,以 FortNOX、SE-Floodlight 等为代表的演进式安全控制器的开发已成为学术界和产业界的重要研究方向之一.然而,由于演进式控制器的开发工作主要是在已有开源控制器的基础上改进和扩充新的安全服务.因此,其开发工作受控制器自身的系统架构、编程语言和预留的安全接口等方面的影响较大.同时,由于现有的控制器种类较多,其编程语言和系统架构等方面也均存在差异.目前,虽然以 FortNOX、SE-Floodlight 等为代表的诸多控制器安全应用和安全架构被提出,但这些研究成果大多是针对特定控制器设计的,并不能实现安全应用在不同控制器之间的平滑过渡.因此,演进式安全控制器的开发工作在安全应用的移植和推广方面仍面临着诸多问题.

3.1.2 革命式安全控制器

针对现有控制器中存在的各种安全问题,通过附加安全机制或修补安全漏洞的方式,使得研究者和开发人员比较被动,很难在短时间内有效提升控制器的安全性.因此,一些研究者提倡在 SDN 控制器的设计和开发之初,便将安全性作为其核心问题之一进行考虑,从而突破已有控制器在系统架构、编程语言和预留接口等方面的限制,开发出全新的、内嵌安全机制的 SDN 控制器,以较好地提高 SDN 控制器的安全性能.目前,革命式安全控制器开发工作的代表性成果包括 Rosemary[84]和 PANE[82]等控制器.

由于 Floodlight、OpenDaylight、NOX、POX 和 Beacon 等常见控制器在安全性和健壮性等方面均面临着不同程度的问题,Shin 等人通过对常见控制器的安全性进行测试和分析后,设计了一种新型、内嵌安全机制的集中式控制器/网络操作系统 Rosemary[84].RoseMary 控制器由 C 语言编写,总代码约 2 万行.如图 8所示,与 SE-Floodlight、FortNOX 等演进式安全控制器的设计思想不同,RoseMary 控制器主要由数据抽象层(data abstraction layer)、RoseMary 内核(RoseMary kernel)、系统库(system libraries)和资源监控器(resource monitor)这 4 部分组成.其中,数据抽象层主要用于收集数据层中各种网络设备的请求信息;资源监控模块主要对 RoseMary 中正在运行的应用程序及其行为进行监控,并及时终止具有异常行为的应用程序;而 RoseMary 内核则被细分为资源管理模块(resource manager)、安全管理模块(security manager)、系统日志管理模块(system log manager)和内核程序区这 4 个主要部分,为网络的应用层提供资源控制、安全管理和系统日志记录等基本服务.同时,RoseMary 还为应用程序提供了各种类型的系统库,如调度和日常管理类系统库、内部存储管理类系统库和 I/O 接口管理类系统库等;当应用程序需要使用这些系统库时,可根据实际需要向 RoseMary 内核发送申请消

息,由 RoseMary 内核对其使用权限进行授权.

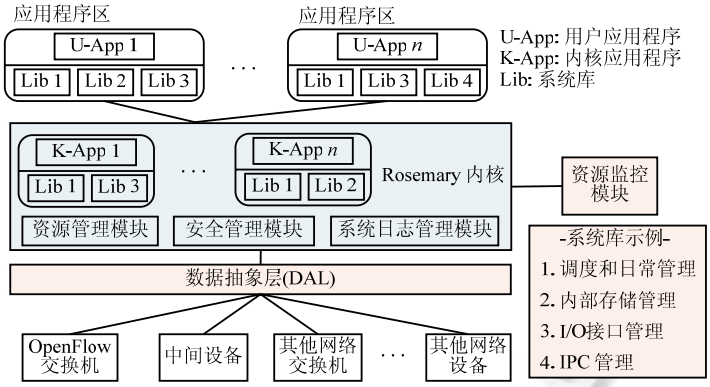


Fig.8 Overall architecture of Rosemary: A robust and secure controller or network operating system[84]
图8 具有安全特性的控制器操作系统 Rosemary 示意图[84]

RoseMary 作为革命式安全控制器开发方面的典型实例之一,与已有的演进式安全控制器相比,它对应用程序的行为监控更加严格.如图 9所示,RoseMary 内核将所有应用程序运行在一个封闭的应用程序区内,实时监控各个应用程序的行为,并通过检查各个应用程序的签名信息判定其是否为合法应用程序.当某个应用程序启动高于自身权限的系统调用功能时,RoseMary 权限审核模块(system call access check module)会首先对该应用程序的角色和权限进行审核,及时拒绝应用程序的未授权操作.然而,由于 RoesMary 系统对应用程序的签名方式是基于角色的签名机制,并将整个应用程序作为一个整体对其进行签名,因而无法较好地对应用程序各个模块的访问权限进行细粒度控制.

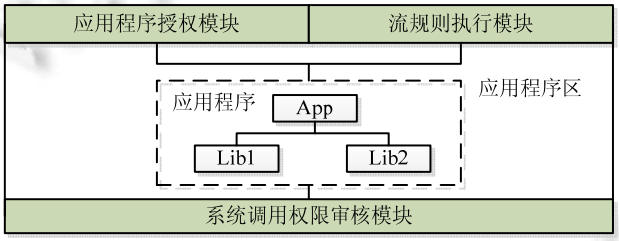


Fig.9 Internal architecture of application zone in Rosemary[84]
图9 Rosemary 中的应用程序区结构[84]

Ferguson 等人[82]开发了内嵌安全机制的 PANE 控制器,用于解决 SDN 中不可信用户访问请求之间的冲突问题.PANE 控制器允许管理员根据网络资源制定相应的访问控制策略,同时,通过 PANE 提供的 API 接口,终端用户可以动态、自主地请求网络资源.

Rosemary 和 PANE 等革命式安全控制器在设计和开发之初,便将安全机制内嵌于控制器系统,这种开发思想突破了已有控制器在系统架构、编程语言和预留接口等方面带来的诸多限制.因此,在控制器安全模块和安全服务的开发和部署方面,基于这种开发思想设计的 SDN 安全控制器,其系统架构的灵活性相对较高.然而,在革命式安全控制器的设计和研发方面,由于不同的研发机构其侧重点并不相同,各个 SDN 标准化组织也尚未发布关于 SDN 控制器设计方面的正式安全标准和规范,而现有的一些控制器(如 Onix、ONOS、DISCO、HP VAN SDN、Ryu 等)目前已在一些数据中心网络和云计算网络中得到部署.因此,综合控制器侧重的主要功能、用户在安全方面的实际需求和转换成本等多方面的因素,革命式安全控制器的开发和部署仍处于实验室测试阶段,并未得到大规模的部署和应用.

此外,由于分布式控制器可以应对跨域的 SDN 通信问题,相对于集中式控制器,其应用环境更加广泛.在分

布式安全控制器的开发方面,目前,仅 Onix[88]、ONOS[89]和 SMaRtLight[96]这 3 种控制器提供了多控制器之间的数据一致性安全模型.Kreutz 等人[66]从信息备份、设备之间的信任关系和安全域等方面出发,也提出了多控制器协作的安全方案.然而,在由分布式控制器构成的 SDN 网络中,由于不同控制器之间进行通信的东西向接口尚无明确的安全标准,分布式控制器在安全通信和资源调度方面均面临着诸多安全挑战.

3.2 可组合安全模块库的开发和部署

控制器的安全是 SDN 网络能够安全运行的基础.SDN 通过控制器对网络进行集中管控,这种工作模式虽然在安全策略的细粒度、实时推送和流量监控等方面具有较大优势,但同时也使得控制器成为被攻击的焦点.因此,在控制器上开发和部署相应的可组合安全模块库,通过这些安全模块库的相互组合和协作,不断增强控制器的安全防护能力,是 SDN 安全的一个重要研究方向.

Shin 和 Porras 等人设计了一种面向 SDN 控制器的可组合安全模块开发框架 FRESCO[21].这种安全框架允许开发人员在控制器上创建新的安全模块,且这些安全模块之间可以相互组合并协同工作,从而加快了控制器安全模块库的设计和开发.FRESCO 集成了大量 API 接口,并能够与传统的安全工具(如 BotHunter 等软件)进行通信.如图 10所示,FRESCO 部署在开源控制器 NOX 之上,由应用层和安全执行内核两部分组成.同时,它提供了由 Python 脚本语言编写的 API 接口,使得研究人员可以自己编写具有安全监控和威胁检测功能的 Module 安全模块.Module 安全模块是 FRESCO 安全模块库的基本处理单元,不同的 Module 模块用于提供不同的安全功能.同时,这些模块可以被共享或组合,以提供更加复杂的安全防护功能.

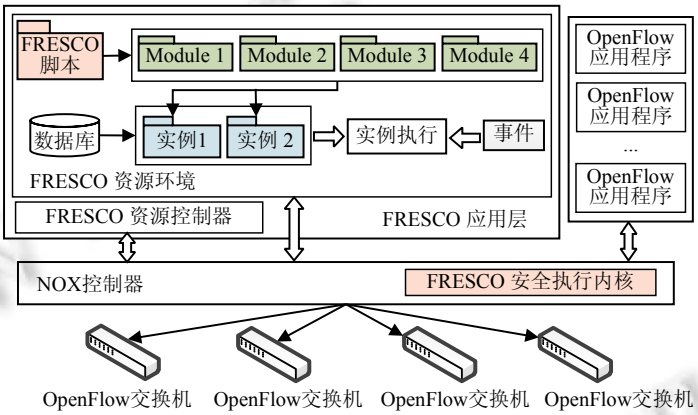


Fig.10 High-Level over view of the FRESCO architecture in NOX[21]
图10 NOX 控制器上的可组合安全模块开发框架 FRESCO[21]

如图 11所示,每个 Module 模块提供 5 个接口:输入(input)、输出(output)、事件(event)、参数(parameter)和动作(action).其中,输入和输出是接收和转发信息的接口,参数用于定义 Module 的配置信息和初始化信息,动作是对数据包的基本处理方式,事件用于通知 Module 执行相应的操作.

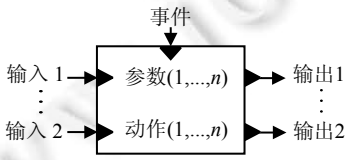


Fig.11 Illustration of FRESCO module design[21]
图11 FRESCO 架构-Module 模块示意图[21]

FRESCO 的资源控制器用于监控 OpenFlow 交换机的状态,并删除交换机流表中废弃的流规则,及时清理流表空间.此外,FRESCO 还可以与传统的网络安全工具,如 BotHunter[98]等结合使用,根据这些传统网络安全工具

提供的检测报告,对 Module 安全模块进行重新组合,制定出新的安全策略,并更新至控制器中的流表,从而为控制器建立起一种高效、可重构的安全防护平台.

然而,FRESCO 虽然可以对不同的 Module 安全模块进行组合,生成一些更加复杂的安全防护策略,但不同的安全模块在组合后,其安全强度是否真的有所增加以及增加的具体安全强度,目前还没有完善的评估机制.同时,不同的安全模块在组合后是否会引入新的安全漏洞也有待检验.

3.3 控制器DoS/DDoS攻击防御

DoS/DDoS 攻击主要是指攻击者通过傀儡主机消耗攻击目标的计算资源,阻止目标为合法用户提供服务.对于 SDN 控制器而言,DoS/DDoS 是简单且行之有效的攻击方式.当新的流请求到达交换机之后,若交换机的流表中没有与之匹配的流规则,该交换机会将请求信息转发给控制器,由控制器制定相应的应答策略.如图 12所示,当攻击者通过不同的交换机持续地向控制器发送大规模的虚假请求信息时,会使得控制器一直忙于应答攻击者的非法请求,而无暇响应合法用户的正常请求.当控制器在短时间内接收到的虚假请求信息超过一定规模时,它便可能因计算资源和内存资源消耗过度而无法正常工作,致使整个 SDN 网络处于瘫痪状态.

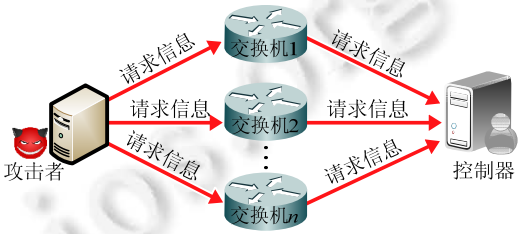


Fig.12 Launching denial-of-service (DoS) attacks to a SDN controller
图12 控制器遭受 DoS 攻击示意图

针对 SDN 控制器面临的 DoS/DDoS 攻击问题,现有解决方案的主要思路分为 3 种:(1) 基于流量变化特征对控制器 DoS/DDoS 攻击行为进行检测;(2) 基于连接迁移机制对 DoS/DDoS 攻击进行防范;(3) 基于 STRIDE、UML 等威胁建模方法,对 SDN 中潜在的 DoS/DDoS 攻击行为进行评估和预测.

3.3.1 基于流量特征变化的控制器 DoS/DDoS 攻击检测方法

DoS/DDoS 攻击的明显特征是短时间内流量的大幅度增加,因此,基于流量特征变化来检测 DoS/DDoS 攻击行为是 SDN 中常用的方法.Braga 等人[26]利用 SDN 集中管控的特点,提出了一种基于流量特性的轻量级 DDoS 洪泛攻击检测方法.他们通过自组织映射算法 SOM(self organizing maps)对网络中的信息流进行分类,并利用信息流中与 DDoS 攻击特征相关的六元组信息(信息流中数据包的平均数量、信息流的平均字节数、信息流的平均持续时间、配对信息流的百分比、单信息流的增长率、不同端口的增长率)对攻击行为进行检测.相较于传统的流量特征提取方法,该方案在 DDoS 攻击检测的流量特征提取方面,具有消耗低和检测率较高的特点.此外,Hong 等人基于流量特征和拓扑变化等技术,对控制器 Floodlight 进行安全性扩展,也设计了一种面向网络拓扑变化的实时、动态监测模型 TopoGuard[27].该模型通过在控制器上部署“网络拓扑检验”和“交换机端口监测”等模块,可对数据面和控制面之间的 DoS 等攻击行为进行有效防御.

3.3.2 基于连接迁移机制的 DoS/DDoS 攻击防范方法

SDN 控制器遭受 DoS/DDoS 的直接原因主要是短时间内收到了大量来自交换机的请求信息,因此,若在控制器和交换机的通信过程中增加一个流信息过滤模块(如图 13所示),在控制器收到流请求之前,由该过滤模块对无效的 TCP 会话信息进行检测和过滤,转移非法和虚假的流请求信息,仅将合法的流请求信息传送至控制器,便能够有效降低控制器遭受 DoS/DDoS 攻击的可能性.基于连接迁移机制的 DoS/DDoS 攻击防范方法正是基于这个原理对 SDN 控制面和数据面之间的 DoS/DDoS 攻击进行防范.

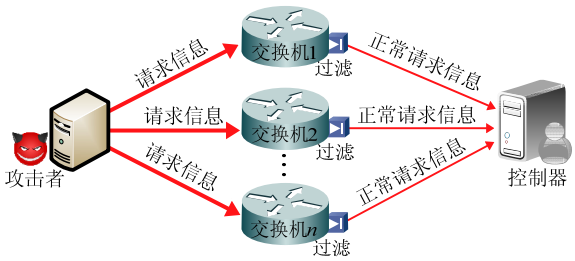


Fig.13 Connection-Migration based DoS/DDoS prevention in SDN
图13 基于连接迁移机制的 DoS/DDoS 攻击防范方法

Shin 等人基于连接迁移机制对 SDN 的数据面进行扩展,提出了一种可以有效检测 DDoS 攻击的安全架构 AVANT-GUARD[28].如图 14所示,AVANT-GUARD 在数据面已有统计和收集服务的基础上,新增了连接迁移(connection migration)和激励触发(actuating trigger)两个模块.当新的数据包到达 SDN 的交换机时,交换机首先检测流表中是否存在与该数据包匹配的流表项.若流表中包含与该数据包匹配的流表项,则交换机按照相应的流表项转发该数据包.否则,AVANT-GUARD 便启动连接迁移功能,将相关的数据包信息迁移至连接迁移模块.连接迁移模块在收到数据包信息后,首先判断该数据包是否为 TCP 会话中的 SYN/ACK 数据包.如果该数据包为 TCP 会话中的 SYN/ACK 数据包,连接迁移模块会将该数据包判定为正常的请求信息,并将其转发至 SDN 控制器.否则,连接迁移模块将其视为异常请求信息,并拒绝转发该数据包.此外,AVANT-GUARD 中的激励触发模块用于收集数据面当前的网络状态,并将相关信息及时报告给控制器,从而可以加快控制器对数据面流量变化的察觉和回应.

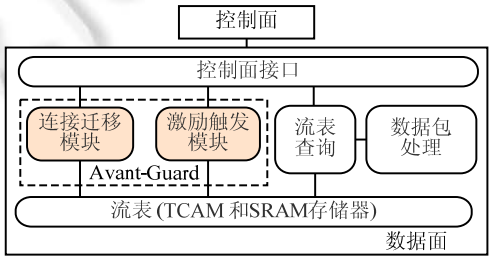


Fig.14 Conceptual architecture of AVANT-GUARD[28]
图14 AVANT-GUARD 体系结构示意图[28]

基于连接迁移机制的 AVANT-GUARD 架构可以有效地减少控制层和数据层之间的互操作,并将网络状态信息及时报告给控制器,以降低控制器遭受 DoS/DDoS 攻击的可能性. 然而, 由于 AVANT-GUARD 中的连接迁移模块仅对 TCP 会话中的数据包进行检测和过滤,因此,这种架构仅适用于对基于 TCP 的 DoS/DDoS 攻击进行防范,却无法抵御基于其他协议的 DoS/DDoS 攻击行为.

Wang 等人基于数据包迁移(packet migration)和数据面缓存(data plane cache)机制,设计了一种轻量级 DoS 攻击防御模型 OF-Guard[40],并在此基础上,结合主动流规则分析技术(proactive flow rule analyzer),提出了一种新型 DoS 攻击防御架构 FloodGuard[25].当检测到 DoS/DDoS 攻击行为时,FloodGuard 中的数据包迁移机制将产生一个通用的流规则,通过该流规则将非法数据包进行重新定向,转移至数据面的临时缓存中,从而对控制面和数据面之间的 DoS/DDoS 攻击行为进行防御.FloodGuard 弥补了 AVANT-GUARD 架构的不足,可以对 SDN 控制层和基础设施层之间更多类型的 DoS/DDoS 攻击行为进行检测和防御.

3.3.3 基于威胁建模的 DoS/DDoS 攻击评估方法

威胁建模是网络安全的重要评估方法和决策机制之一,它可帮助系统设计者清晰地理解和识别系统中潜在的安全威胁、攻击和漏洞.基于威胁建模的 DoS/DDoS 攻击评估方法的基本思想是在 SDN 网络部署之前,使

用信息系统和网络安全的一些威胁分析模型,如 STRIDE、UML 等,提前对系统中潜在的 DoS/DDoS 威胁进行刻画和评估,防患于未然.

针对 SDN 中的 DoS/DDoS 攻击等安全威胁,Kloti 等人[55]结合 STRIDE 威胁检测模型和攻击树技术,设计了一种可对 OpenFlow 安全性进行有效分析的威胁检测模型.该威胁检测方法的基本过程包括如下 4 步:(1) 基于 STRIDE 威胁分析模型对 OpenFlow 系统进行建模,列举系统潜在的脆弱性,如 DoS/DDoS 攻击、可扩展性和自适应性等方面的安全威胁;(2) 基于 STRIDE 模型建立 OpenFlow 各系统的数据流图,包括基本的处理过程、数据存储、有条件的数据流和信任边界等;(3) 分析系统的详细数据流图,从拒绝服务、欺骗、篡改、不可否认、信息泄露和特权提升等方面,对系统各组成部分的脆弱性进行检查;(4) 通过向控制器发送大量的数据来模拟 DoS/DDoS 攻击,对方案的性能进行验证.图 15为基于 STRIDE 模型对 OpenFlow 交换机的安全性进行威胁建模的一个基本实例,图中分别给出了需要执行转发工作的一系列数据路径和OpenFlow模块的相关活动,并刻画出了 SDN 交换机流表中潜在的 DoS 攻击威胁.

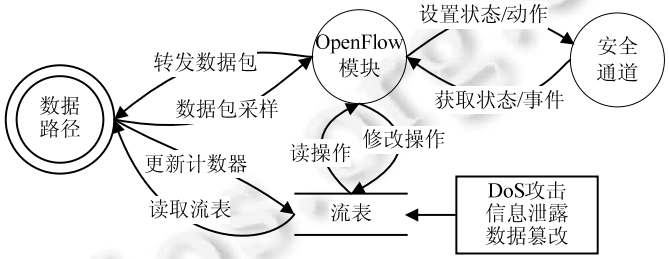


Fig.15 Simplified DFD for an OpenFlow switch, showing relevant vulnerabilities[55]
图15 基于 STRIDE 的 OpenFlow 交换机数据流程图(简化版本)[55]

3.4 流规则的合法性和一致性检测

SDN 是典型的流规则驱动型网络,流规则的合法性和一致性是保证 SDN 正常、有效运行的基础.SDN 中的流规则通常由系统管理员、OpenFlow 应用程序、安全服务类应用程序和一些其他的第三方应用程序共同制定后,并通过控制器下发至基础设施层的网络设备中.这些流规则是 SDN 交换机执行转发、数据包处理等操作的依据.由于基础设施层的网络设备对控制器下发的流规则完全信任,一旦由虚假控制器或恶意应用程序提供的流规则被执行,将使 SDN 的安全性面临严重威胁.因此,对流规则的合法性和一致性进行检查,防止恶意和非法流规则的扩散,并确保各类流规则的正确下发和执行,对 SDN 的安全运行至关重要.

针对 SDN 中流规则的合法性和一致性检测问题,目前研究者给出的解决方案的主要思路包括两种:(1) 基于应用程序的角色和优先级,对流规则的等级进行划分;(2) 采用形式化和数学分析方法,对不同流规则之间的一致性和冲突性进行分析.

3.4.1 基于角色和优先级划分的流冲突检测方法

如图 16所示,若将 SDN 中流规则的生命周期简单定义为生成、运行和废弃这 3 个阶段,则基于角色和优先级划分的流冲突检测方法通常应用于流规则的生成阶段.该检测方法主要通过数字签名、角色划分和功能分类等技术,对参与制定 SDN 流规则的实体(包括系统管理员、各类应用程序等)进行分级,在流规则生命周期的源头(未写入控制器流表前)对其合法性和等级进行确认.这种思想的代表性研究工作为 Porras 和 Shin 等人设计的 SDN 控制器安全增强内核 FortNOX 架构[20].

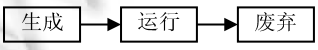


Fig.16 The simplified life-cycle model of flow rules in SDN
图16 SDN 中流规则生命周期简图

FortNOX 是对开源控制器 NOX 的一个安全扩展,它要求系统管理员和各类应用程序对自身生成的流规则

进行签名,从而确保了流规则的来源具有可认证性.在具体执行过程中,FortNOX 将参与流规则制定的主体划分为 3 类角色:(1) 系统管理员:在冲突处理的过程中,系统管理员的流规则插入请求具有最高优先权;当控制器向交换机下达流规则时,管理员的流规则优先级最高.(2) 安全类应用程序:由于安全类应用程序会基于实时的威胁(如恶意链接、内部感染的主机等)产生一些新的流规则,而这些规则可能与管理员的一些流规则冲突或相互限制,因此,在系统默认情况下,安全类应用程序的优先级低于管理员.(3) 与系统安全无关的 OpenFlow 应用程序:这部分应用程序的等级最低,其他没有被签名的流规则均为最低权限.

各实体生成的流规则在被插入控制器流表前,FortNOX 会根据流规则的签名内容判断其优先级和执行顺序,进而将不同的流规则按优先级更新到控制器流表中.若用 P_New 表示新流规则签名者的优先级,P_Old 表示总流表中与新流规则相冲突的某项流规则的优先级,则在冲突处理过程中,系统遵循的规则如下:

(1) 若 P_New 高于 P_Old,则旧的流规则被新的流规则所覆盖;

(2) 若 P_New 低于 P_Old,新流规则的插入请求被拒绝,系统将错误信息报告安全应用平台;

(3) 若 P_New 与 P_Old 相同,则提交系统高级管理员处理.

针对控制器上流规则的冲突问题,FortNOX 提供了一种按照实体的优先级对其生成的流规则进行等级划分的方法.然而,这种方法虽然在一定程度上缓解了流规则冲突问题,但由于 FortNOX 仅将参与流规则制定的实体划分为管理员、安全类应用程序和其他应用程序 3 类角色,而实际情况中参与流规则制定的实体更为复杂,如同为安全类应用程序的多个应用程序实体,它们生成的流规则之间也可能存在冲突.因此,还需要采用更细粒度的优先级划分机制对流规则的优先级和一致性进行约束.

3.4.2 基于形式化方法的流规则冲突检测机制

基于形式化方法的流冲突检测机制首先对待检测的流规则进行形式化描述,然后将描述信息和目标流规则一同输入到指定分析工具中,根据分析工具给出的分析结果对流规则的冲突情况进行判断.在 SDN 网络中,通常采用基于模型检测、约束求解等形式化分析方法,对 SDN 流规则的一致性和合法性进行检验.目前,该方法主要应用于 SDN 流规则生命周期的第 2 阶段,即流规则生成后至被废弃前,代表性的研究工作包括:(1) 基于二元决策图技术对 SDN 流规则进行检测的 FlowChecker 检测系统[50];(2) 基于 Yices SMT 求解器对 SDN 流规则进行检验的 FLOVER 约束求解系统[51]等.

Al-Shaer 等人[50]通过二元决策图技术(binary decision diagrams,简称 BDD)对 OpenFlow 流表的配置信息进行重新编码,并结合模型检测(model checking)技术,提出了 FlowChecker 检测系统,旨在通过形式化的方法对不同交换机和控制器之间的流规则一致性进行检验.另外,通过与 FlowVisor[99]网络切片技术相结合,FlowChecker 还可以对新协议的正确性、可达性和安全性等问题进行验证.

针对应用程序生成的流规则是否符合当前 SDN 的安全策略问题,Son 等人[51]结合 Yices SMT 求解器[100,101],提出了 FLOVER 模型检测系统.该系统将 OpenFlow 的流规则和网络安全相关的策略分别转化为相应的断言,并将断言信息输入到 SMT 求解器,进而根据 SMT 求解器给出的分析结果来判断某些流规则是否符合当前 SDN 的安全策略.

此外,Reitblatt 等人[52]针对多个交换机之间流规则的一致性问题,通过数据包一致性检测机制和数据流一致性检测机制等技术,对 SDN 中数据包的一致性和流规则的一致性也分别进行了有效检测.Khurshid 等人[53]针对流规则的合法性检测问题,也提出了流规则的 VeriFlow 检测模型.当转发类型的流规则被插入流表时,VeriFlow 可以对全网范围内的违规行为进行检查,实时定位和处理违规的流规则.

3.5 北向接口的安全性

在 SDN 控制层和应用层的安全交互方面,北向接口的安全性扮演着重要角色.目前,现有的各类控制器并没有为北向应用程序提供标准化的安全接口.为满足不同控制器的安全需求,开发人员需根据控制器类型的不同,重新对应用程序及其接口进行开发.由于与控制器进行交互的应用程序种类多样,版本也在不断更新,攻击者可通过北向接口的安全漏洞直接向控制器发起攻击,从而使得 SDN 控制器对应用程序的认证和安全管理等

工作变得更加复杂.

ONF 于 2013 年 10 月成立了 SDN 北向接口工作小组(Northbound Interface Working Group,简称 NBI-WG),旨在通过北向接口协议的标准化推进 SDN 的广泛应用.不同的厂商和参与者也分别从用户需求和商业运营等方面出发,提出了一些应用于特定环境的北向接口安全协议和方案.在学术界,目前北向接口安全方面的研究工作主要集中在控制器应用程序的访问控制方面,代表性成果除了 FortNOX[20]、SE-Floodlight[22, 64]等安全架构中涉及到的北向接口安全方案之外,还包括应用程序权限管理系统 PermOF[59]和 OperationCheckpoint[62],以及应用程序的细粒度访问控制机制[63]等.

FortNOX[20]、SE-Floodlight[22,64]分别设计了基于角色的北向应用程序访问权限管理模块,并将应用程序的角色划分为系统管理员、安全类应用程序和与安全无关的 OpenFlow 应用程序这 3 类,依据应用程序的具体角色,对不同流规则插入请求的优先级和权限进行划分.然而,由于部署于控制器上的应用程序种类较多,仅将它们划分为 3 类,无法对应用程序的访问权限进行细粒度的控制.此外,在北向接口的访问控制方面,SE-Floodlight 还提供了一个具有认证功能的北向应用程序接口,由管理员预先对应用程序的 Java 类进行签名,以便它们在运行时可以顺利通过安全内核的认证.但是,由于 SE-Floodlight 以整个应用程序包的形式对 Java 类进行签名,而非对应用程序各个权限模块或功能分别进行签名,因而对程序的验证粒度也较粗.

Wen 等人[59]将控制器上的应用程序和控制器内核进行隔离,基于应用程序访问权限最小化的思想,设计了一个细粒度的控制器应用程序访问权限管理系统 PermOF.PermOF 分析了控制器上应用程序的 18 种访问权限,并引入访问控制层(access control layer),以限制非法应用程序对控制器内核资源的直接访问.相对于 SE-Floodlight 对整个应用程序进行签名的方式,PermOF 可对应用程序的访问权限进行更细粒度的认证和管理,但这种细粒度的认证和管理方式却要以牺牲系统的处理时间和效率为代价.

基于 Wen 等人[59]的工作,Scott-Hayward 等人通过对应用程序的 Read 权限进行扩展,设计了一个更加灵活的控制器应用程序访问权限管理系统 OperationCheckpoint[62].在初始化时,系统先为每个应用程序分配一个唯一的 ID,并将该 ID 和应用程序的访问权限集合映射到一个安全的存储结构中.当应用程序试图启用权限集合中的访问权限时,OperationCheckpoint 会先获取该应用程序的 ID,并查询安全存储结构中与该 ID 对应的访问权限;若应用程序试图使用的权限满足权限集合的要求,则请求可顺利通过.否则,OperationCheckpoint 会将应用程序的请求信息视为无效请求,并写入系统异常行为监测日志中.在该应用程序的生命周期内,Operation Checkpoint 严格按照应用程序访问权限集合对每个应用程序的访问请求进行检验,且可以根据系统需要动态地调整某个应用程序的访问权限集合,因而对应用程序访问权限的管理粒度更细,灵活性也较高.然而,由于该方案将应用程序的 ID 和相应的访问权限集合映射到一个存储表中,一旦该存储表的信息被泄露或恶意篡改,将导致 OperationCheckpoint 权限检测模块失效.

此外,Klaedtke 等人[63]从资源分类、个性化安全需求和权限委托等方面出发,设计了一种支持权限委托的控制器应用程序访问控制模型.该模型为每个访问主体分配的访问权限包括读取状态信息、请求客体资源、修改客体状态和颁发权限这 4 类,并对客体资源进行细粒度的划分,以限制应用程序对它们的访问权限.同时,该模型还增加了权限委托机制,父节点可将访问权限委托给自己的子节点,以简化权限管理的操作.然而,方案中并未给出对应用程序自身合法性进行认证的方法,一旦恶意应用程序通过身份假冒等技术获得系统的访问权限后,将有机会把系统访问权限委托给更多的非法实体,从而给 SDN 安全带来更多的威胁.

3.6 应用程序安全性

在 SDN 网络中,应用程序自身的安全性主要是指 SDN 核心设备中的一些应用程序本身是否存在安全漏洞,以及这些应用程序是否会因为受到某些攻击而使得 SDN 中一些关键信息的安全性受到威胁.

SDN 中的一些流规则是由 OpenFlow 应用程序、安全服务类应用程序和一些其他的第三方应用程序共同制定的,这些应用程序自身的安全程度直接决定着流规则能否正常生成和更新,而流规则又是 SDN 基础设施层对数据包处理的基本依据.因此,SDN 中一些重要应用程序的安全性也是保障 SDN 安全的重要因素之一.针对应用程序自身的安全性问题,Ball 等人[102]提出了一种验证 SDN 控制器应用程序是否正常运行的工具 VeriCon;

Canini 等人[103]也提出了 NICE 检测模型,并使用该模型对真实的 Python 应用程序进行了漏洞测试和验证.然而,由于目前 SDN 网络的部署和应用仍处于初级阶段,大多数用户和开发者在判断某个应用程序是否符合要求时,通常偏重于应用程序的功能性验证,而忽略或较少考虑应用程序自身的安全性验证,再加上 SDN 的开放性和可编程性特征,使得攻击者在很多情况下都能有机可乘.

除了上述 6 类 SDN 安全防护方案之外,在 SDN 安全方面的研究工作中,还有一些研究者致力于对网络防火墙[23,48,104,105]、虚拟网络切片[99,106,107]、进程级纵深防御体系结构[108]等方面的研究,以期不断提升 SDN 的安全性.然而,随着各类安全方案的不断出现,SDN 面临的安全问题虽然在一定程度上得到了缓解,但一些关键的安全问题仍然存在,如集中管控带来的控制器单点失效问题,可编程性和开放接口带来的控制层和应用层之间信任关系脆弱问题,以及多控制器之间的安全交互问题等,这些问题依然从很大程度上限制着 SDN 在诸多场景中的应用和部署.

4 SDN 安全的标准化工作进展

随着 SDN 技术的不断发展,其安全的标准化问题日益突出,已逐渐成为阻碍 SDN 发展的重要因素之一.目前,开放网络基金会(ONF)、欧洲电信标准协会(ETSI)、Internet 工程任务组(IETF)、国际电信联盟(ITU)和中国电信标准化协会等多个标准化组织,以及 ONOS、OpenDayLight、OPNFV 等开源组织均在积极开展 SDN 安全领域的标准化研究工作(见表 3).然而,目前在 SDN 安全领域的行业标准和规范制定方面,各个标准化组织的侧重点并不相同.而 SDN 作为一种新型的网络架构,其安全标准又涉及 SDN 架构的安全性、控制器的安全性、南/北向接口协议的安全性、应用程序的安全性等多个方面,因此,截止到目前为止,国内外尚未有明确的关于 SDN 安全方面的行业标准被正式发布.

Table 3 Brief analysis of the standardizations work on SDN security

表 3 SDN 安全的标准化工作进展

组织名称	各个标准化组织在 SDN 安全标准化方面的主要工作	安全标准状态
开放式网络基金会(ONF)	(1) 设置 SDN 安全组,对 OpenFlow 协议的安全性、SDN 架构的安全能力等问题进行讨论[34] (2) 发布了 SDN 安全相关的一些技术文档[109]和安全方案介绍[110];OpenFlow 协议各个版本也涉及部分加密和认证操作	已发布相关安全方案的简介,尚无详细、正式的 SDN 安全标准和规范
互联网工程任务组(IETF)	(1) IETF 早期便有与 SDN 思想类似的 FoCES 项目工作组,并召开专门的工作组会议,对路由系统接口与 SDN 安全的关系、SDN 应用与控制器之间的安全需求等问题进行讨论 (2) 发布了针对 OpenFlow 协议的安全性分析草案[56],并陆续公布了一些略提到 SDN 安全注意事项的草案,如文献[111−113]等	已陆续发布一些与 SDN 安全相关的草案,尚未公布正式的 SDN 安全标准
国际电信联盟标准化组织(ITU-T)	(1) 设置不同的课题组,提出了在电信网中实现 SDN 架构的初步思路,并对 SDN 通用功能、通用实体标准制订,以及 SDN 在未来网络、NGN 和云计算等场景中的应用和功能需求进行研究 (2) 开展了与 SDN 安全相关的信令架构、信令需求、QoS 等研究工作[114]	已完成部分与 SDN 安全相关的标准立项,尚未公布正式的 SDN 安全标准
欧洲电信标准化协会(ETSI)	(1) 成立虚拟化标准工作组 NFV ISG,着重从电信运营商的角度,对虚拟化架构、可靠性与可用性、安全性等方面对与 SDN 相关的问题进行研究 (2) 陆续发布了一些与 NFV 相关的行业规范[115]	已发布部分与 NFV 相关的行业规范,尚未发布正式的 SDN 安全标准
中国通信标准化协会(CCSA)	(1) 组织国内运营商、设备商及科研院所开展 SDN 相关标准的研究工作,成立以软件定义为核心的未来数据网络(FDN)特别工作组(SWG3)和软件化智能型通信网络子工作组(SVN) (2) 从网络虚拟化、体系和协议架构、OpenFlow 协议、东西向接口协议等方面对 SDN 及其相关的安全问题进行研究,已发布部分与 SDN 相关的行业标准[116]	已发布部分与 SDN 相关的行业标准,尚未发布正式的 SDN 安全标准

5 结论与展望

SDN 将控制面与数据面解耦,在逻辑上实现了网络的集中管控,简化了网络的管理,但同时也引入了诸多安全问题.本文对 SDN 的基本架构、安全模型和典型安全问题进行了归纳,并从 SDN 安全控制器的开发、控制器

可组合安全模块库的开发和部署、控制器 DoS/DDoS 攻击防御、流规则的合法性和一致性检测、北向接口的安全性和应用程序安全性等方面,分析探讨了现有 SDN 安全防护方案的主要研究思路和最新进展,最后对 SDN 安全的标准化工作进行了简要分析,以期为 SDN 安全性研究工作的推进提供参考.目前,随着各类安全方案的不断出现,SDN 面临的安全问题虽然在一定程度上得到了缓解,但一些关键的安全问题仍然存在,未来 SDN 的安全研究还将重点关注以下几个方面.

(1) 面向安全的新型控制器/网络操作系统的设计与开发

大多数 SDN 控制器在设计之初主要关注的是网络资源的调度和控制,如链路发现、拓扑管理、策略制定和表项下发等,对控制器自身的安全问题考虑得较少.随着 SDN 架构的推广和应用,控制器的安全问题不断暴露.未来,结合操作系统安全管理思想和密码学等知识,设计新型、内嵌安全机制的 SDN 控制器将是 SDN 安全领域的一个重要研究方向.

(2) 控制器跨域协同安全通信问题

为缓解单控制器导致的单点失效、扩展性差等问题,OpenFlow 1.3 版本之后的各规范分别增加了多控制器的部署策略.但由于多个控制器可能分布在不同的自治域,不同控制器之间需要进行身份切换和资源调度等操作,如何保证多个控制器之间安全、实时地通信,将是未来 SDN 安全领域需要解决的一个重要问题.

(3) 北向接口安全协议的标准化

北向接口使得控制层对应用层更加开放,攻击者可通过北向接口间接实现对控制器的攻击,从而增加了北向接口遭受攻击的可能性.目前,SDN 北向接口安全协议的标准化方面还面临着诸多问题.一方面,不同类型的控制器分别定义了自身的应用程序编程接口,且这些接口的编程语言和具体功能并不相同,从而限制了北向接口安全协议的移植性和可扩展性.另一方面,由于北向接口大多由软件应用程序控制,不同的应用程序,如安全应用、路由应用和金融应用等,分别具有不同的安全需求,单一功能的北向接口安全通信协议并不适合所有的应用程序.因此,北向接口安全协议的设计和标准化也将是未来 SDN 面临的一个重要安全问题.

(4) 控制器 DoS/DDoS 攻击检测与防范技术

SDN 采用逻辑上集中控制的方式对整个网络进行管理,将网络的“智慧”集中在控制器上,因而增加了控制器遭受 DoS/DDoS 的可能性.已有的研究结果表明,针对控制器实施 DoS/DDoS 攻击是非常有效的.因此,需针对 SDN 的架构特征,不断研究和设计新型的 DoS/DDoS 检测与防范技术.

致谢: 感谢 SDNAP 联盟的各位专家、老师和同学,尤其是北京邮电大学未来网络理论与应用实验室各位老师和张健男、李呈等同学,在 SDN 工作流程和技术等方面提供的无私培训和指导,这些工作为本文的撰写提供了很大帮助,在此表示衷心感谢.同时,也十分感谢审稿专家对本文提出的宝贵意见和建议.

References:

[1] Yang M, Li Y, Jin D, Su L, Ma S, Zeng L. OpenRAN: A software-defined ran architecture via virtualization. In: Proc. of the ACM SIGCOMM 2013 Conf. on SIGCOMM. Hong Kong: ACM, 2013. 549−550. [doi: 10.1145/2486001.2491732]

[2] Mijumbi R, Serrat J, Gorricho J, Bouten N, De Turck F, Boutaba R. Network function virtualization: State-of-the-Art and research challenges. IEEE Communications Surveys Tutorials, 2016,18(1):239−262. [doi: 10.1109/COMST.2015.2477041]

[3] Kannan K, Banerjee S. Scissors: Dealing with header redundancies in data centers through SDN. In: Proc. of the 8th Int’l Conf. on Network and Service Management. Laxenburg: Int’l Federation for Information Processing, 2013. 295−301.

[4] Ghobadi M, Yeganeh SH, Ganjali Y. Rethinking end-to-end congestion control in software-defined networks. In: Proc. of the 11th ACM Workshop on Hot Topics in Networks. Washington: ACM, 2012. 61−66. [doi: 10.1145/2390231.2390242]

[5] Li D, Chen GH, Ren FY, Jiang CL, Xu MW. Data center network research progress and trends. Chinese Journal of Computers, 2014,(2):259−274 (in Chinese with English abstract).

[6] Suresh L, Schulz-Zander J, Merz R, Feldmann A. Demo: Programming enterprise WLANs with ODIN. ACM SIGCOMM Computer Communication Review, 2012,42(4):279−280. [doi: 10.1145/2377677.2377730]

[7] Yi G, Lee S. Fully distributed handover based on SDN in heterogeneous wireless networks. In: Proc. of the 8th Int’l Conf. on Ubiquitous Information Management and Communication. ACM, 2014. 1−7. [doi: 10.1145/2557977.2558047]

[8] Lee J, Uddin M, Tourrilhes J, Sen S, Banerjee S, Arndt M, Kim K, Nadeem T. meSDN: Mobile extension of SDN. In: Proc. of the 5th Int'l Workshop on Mobile Cloud Computing & Services. New Hampshire: ACM, 2014. 7−14. [doi: 10.1145/2609908.2609948]

[9] Jin D, Nicol D M. Parallel simulation of software defined networks. In: Proc. of the 2013 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation. ACM, 2013. 91−102. [doi: 10.1145/2486092.2486104]

[10] Jain R. OpenADN: Mobile apps on global clouds using software defined networking. In: Proc. of the 3rd ACM Workshop on Mobile Cloud Computing and Services. Lake District: ACM, 2012. 1−2. [doi: 10.1145/2307849.2307851]

[11] Yang L, Dantu R, Anderson T, Gopal R. Forwarding and control element separation (ForCES) framework. 2004. https://www.rfc-editor.org/rfc/rfc3746.txt.

[12] Greenberg A, Hjalmtysson G, Maltz D A, Myers A, Rexford J, Xie G, Yan H, Zhan J, Zhang H. A clean slate 4D approach to network control and management. ACM SIGCOMM Computer Communication Review, 2005,35(3):41. [doi: 10.1145/1096536.1096541]

[13] Caesar M, Caldwell D, Feamster N, Rexford J, Shaikh A, van der Merwe J. Design and implementation of a routing control platform. In: Proc. of the 2nd Conf. on Symp. on Networked Systems Design & Implementation. Berkeley: USENIX Association, 2005. 15−28.

[14] Akella A, Boneh D, Mazieres D, McKeown N, Rosenblum M. SANE/inSANE: Designing secure networks from the ground-up. 2006. http://www.yuba.stanford.edu/sane/

[15] Casado M, Garfinkel T, Akella A, Freedman MJ, Boneh D, McKeown N, Shenker S. SANE: A protection architecture for enterprise networks. In: Proc. of the 15th Conf. on USENIX Security Symp. Berkeley: USENIX Association, 2006. 1−15.

[16] Casado M, Freedman M, Pettit J, Luo J, Gude N, McKeown N. Ethane: A security management architecture. 2006. http://yuba.stanford.edu/ethane/index.html

[17] Casado M, Freedman MJ, Pettit J, Luo J, McKeown N, Shenker S. Ethane: Taking control of the enterprise. In: Proc. of the 2007 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications. Kyoto: ACM, 2007. 1−12. [doi: 10.1145/1282380.1282382]

[18] Gude N, Koponen T, Pettit J, Pfaff B, Mart, Casado N, McKeown N, Shenker S. NOX: Towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 2008,38(3):105−110. [doi: 10.1145/1384609.1384625]

[19] Project Floodlight. 2016. http://www.projectfloodlight.org/

[20] Porras P, Shin S, Yegneswaran V, Fong M, Tyson M, Gu G. A security enforcement kernel for OpenFlow networks. In: Proc. of the 1st Workshop on Hot topics in Software Defined Networks. Helsinki: ACM, 2012. 121−126. [doi: 10.1145/2342441.2342466]

[21] Shin S, Porras P, Yegneswaran V, Fong M, Gu G, Tyson M. FRESCO: Modular composable security services for software-defined networks. In: Proc. of the ISOC Network and Distributed System Security Symp. (NDSS). San Diego: Internet Society, 2013. 1−16.

[22] Porras P, Cheung S, Fong M, Skinner K, Yegneswaran V. Securing the software-defined network control layer. In: Proc. of the 2015 Annual Network and Distributed System Security Symp. (NDSS 2015). San Diego: Internet Society, 2015. 1−15.

[23] Wang J, Wang J, Jiao HY, Wang Y, Chen SY, Liu SH, Hu RX. A method of openflow-based real-time conflict detection and resolution for SDN access control policies. Chinese Journal of Computers, 2015,38(4):872−883 (in Chinese with Engliah abstract).

[24] Shin S, Gu G. Attacking software-defined networks: A first feasibility study. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot topics in Software Defined Networking. Hong Kong: ACM, 2013. 165−166. [doi: 10.1145/2491185.2491220]

[25] Wang H, Xu L, Gu G. FloodGuard: A DoS attack prevention extension in software-defined networks. In: Proc. of the 45th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN 2015). Rio de Janeiro, 2015. [doi: 10.1109/DSN.2015.27]

[26] Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: Proc. of the 35th IEEE Conf. on Local Computer Networks (LCN). Denver, 2010. 408−415. [doi: 10.1109/LCN.2010.5735752]

[27] Hong S, Xu L, Wang H, Gu G. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In: Proc. of the 2015 Annual Network and Distributed System Security Symp. (NDSS 2015). San Diego: Internet Society, 2015. 1−15.

[28] Shin S, Yegneswaran V, Porras P, Gu G. AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. Berlin: ACM, 2013. 413−424. [doi: 10.1145/2508859.2516684]

[29] Hinden RM. SDN AND SECURITY: Why take over the hosts when you can take over the network. 2014. http://www.rsaconference.com/events/us14/agenda/sessions/1021/sdn-security-why-take-over-the-hosts-when-you-can

[30] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008,38(2):69−74. [doi: 10.1145/1355734.1355746]

[31] Clean Slate Program. 2007. http://cleanslate.stanford.edu/

[32] ETSI. Network Functions Virtualisation. 2014. http://www.etsi.org/index.php/technologies-clusters/technologies/nfv

[33] OpenDaylight: Open source network controller. 2013. http://www.opendaylight.org/

[34] ONF. Open Networking Foundation. 2013. https://www.opennetworking.org

[35] ONF. Software-Defined Networking (SDN) Definition. https://www.opennetworking.org/sdn-resources/sdn-definition

[36] ONF. OpenFlow Switch Technical Library. https://www.opennetworking.org/sdn-resources/technical-library

[37] ONF. OpenFlow Switch Specification (Version 1.5.1), ONF TS-025. 2015. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

[38] RFC. The Transport Layer Security (TLS) Protocol Version 1.2. 2008. http://tools.ietf.org/html/rfc5246

[39] Porras P. Toward a more secure SDN control layer. 2013. http://sdn.wpengine.com/education/toward-secure-sdn-control-layer/2013/10/

[40] Wang H, Xu L, Gu G. OF-GUARD: A DoS attack prevention extension in software-defined networks. In: Proc. of the Poster Session of the Open Networking Summit 2014. Santa Clara: USENIX, 2014. 1−2.

[41] Lara A, Kolasani A, Ramamurthy B. Network innovation using OpenFlow: A survey. IEEE Communications Surveys & Tutorials, 2014,16(1):493−512. [doi: 10.1109/SURV.2013.081313.00105]

[42] Yeganeh SH, Tootoonchian A, Ganjali Y. On scalability of software-defined networking. Communications Magazine, 2013,51(2):136−141. [doi: 10.1109/MCOM.2013.6461198]

[43] Fei H, Qi H, Ke B. A survey on software-defined network and openflow: From concept to implementation. Communications Surveys & Tutorials, 2014,16(4):2181−2206. [doi: 10.1109/COMST.2014.2326417]

[44] Dai B, Wang HY, Xu G, Yang J. Opportunities and threats coexist in SDN security. Application Research of Computers, 2014,(8):2254−2262 (in Chinese with English abstract). [doi: 10.3969/j.issn.1001-3695.2014.08.003]

[45] Zhang CK, Cui Y, Tang HY, Wu JP. State-of-the-Art survey on software-defined networking (SDN). Ruan Jian Xue Bao/Journal of Software, 2015,26(1):62−81 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4701.htm [doi: 10.13328/j.cnki.jos.004701]

[46] Zuo QY, Chen M, Zhao GS, Xing CY, Zhang GM, Jiang PC. Research on OpenFlow-based SDN technologies. Ruan Jian Xue Bao/Journal of Software, 2013,24(5):1078−1097 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4390.htm [doi: 10.3724/SP.J.1001.2013.04390]

[47] Klingel D, Khondoker R, Marx R, Bayarou K. Security analysis of software defined networking architectures: PCE, 4D and SANE. In: Proc. of the AINTEC 2014 on Asian Internet Engineering Conf. Chiang Mai: ACM, 2014. 15−22. [doi: 10.1145/2684793.2684796]

[48] Wang J, Wang Y, Hu H, Sun Q, Shi H, Zeng L. Towards a security-enhanced firewall application for openflow networks. In: Proc. of the 5th Int'l Symp., CSS 2013. Zhangjiajie, 2013. 92−103. [doi: 10.1007/978-3-319-03584-0_8]

[49] Xia W, Wen Y, Foh CH, Niyato D, Xie H. A survey on software-defined networking. Communications Surveys & Tutorials, 2015,17(1):27−51. [doi: 10.1109/COMST.2014.2330903]

[50] Al-Shaer E, Al-Haj S. FlowChecker: Configuration analysis and verification of federated openflow infrastructures. In: Proc. of the 3rd ACM Workshop on Assurable and Usable Security Configuration. Chicago: ACM, 2010. 37−44. [doi: 10.1145/1866898.1866905]

[51] Son S, Seungwon S, Yegneswaran V, Porras P, Guofei G. Model checking invariant security properties in OpenFlow. In: Proc. of the 2013 IEEE Int'l Conf. on Communications (ICC). Budapest: IEEE, 2013. 1974−1979. [doi: 10.1109/ICC.2013.6654813]

[52] Reitblatt M, Foster N, Rexford J, Walker D. Consistent updates for software-defined networks: Change you can believe in. In: Proc. of the 10th ACM Workshop on Hot Topics in Networks. Cambridge: ACM, 2011. 1−6. [doi: 10.1145/2070562.2070569]

[53] Khurshid A, Zhou W, Caesar M, Godfrey PB. VeriFlow: Verifying network-wide invariants in real time. In: Proc. of the 1st Workshop on Hot Topics in Software Defined Networks. New York, 2012. 49−54. [doi: 10.1145/2342441.2342452]

[54] Benton K, Camp LJ, Small C. OpenFlow vulnerability assessment. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. Hong Kong: ACM, 2013. 151−152. [doi: 10.1145/2491185.2491222]

[55] Kloti R, Kotronis V, Smith P. OpenFlow: A security analysis. In: Proc. of the 21st IEEE Int'l Conf. on Network Protocols (ICNP). Goettingen, 2013. 1−6. [doi: 10.1109/ICNP.2013.6733671]

[56] Wasserman M, Hartman S. Security analysis of the open networking foundation (onf) openflow switch specification. IETF Documents, 2013.

[57] Kreutz D, Ramos FMV, Esteves VP, Esteve RC, Azodolmolky S, Uhlig S. Software-Defined networking: A comprehensive survey. Proc. of the IEEE, 2015,103(1):14−76. [doi: 10.1109/JPROC.2014.2371999]

[58] Nunes BAA, Mendonca M, Nguyen X, Obraczka K, Turletti T. A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications Surveys & Tutorials, 2014,16(3):1617−1634. [doi: 10.1109/SURV.2014.012214.00180]

[59] Wen X, Chen Y, Hu C, Shi C, Wang Y. Towards a secure controller platform for openflow applications. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. Hong Kong: ACM, 2013. 171−172. [doi: 10.1145/2491185.2491212]

[60] Sezer S, Scott-Hayward S, Chouhan P, Fraser B, Lake D, Finnegan J, Viljoen N, Miller M, Rao N. Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine, 2013,51(7):36−43. [doi: 10.1109/MCOM.2013.6553676]

[61] Brazil J. The Northbound API is the key to OpenFlow's Success. 2012. https://www.sdxcentral.com/articles/contributed/the-northbound-api-is-the-key-to-openflows-success/2012/11/

[62] Scott-Hayward S, Kane C, Sezer S. OperationCheckpoint: SDN application control. In: Proc. of the 22nd Int'l Conf. on Network Protocols (ICNP). IEEE, 2014. 618−623. [doi: 10.1109/ICNP.2014.98]

[63] Klaedtke F, Karame GO, Bifulco R, Cui H. Access control for SDN controllers. In: Proc. of the 3rd Workshop on Hot Topics in Software Defined Networking. Chicago: ACM, 2014. 219−220. [doi: 10.1145/2620728.2620773]

[64] SRI Team, Texas A&M Team. Openflowsec.org. 2013. http://www.openflowsec.org/Technologies.html

[65] Tasch M, Khondoker R, Marx R, Bayarou K. Security analysis of security applications for software defined networks. In: Proc. of the AINTEC 2014 on Asian Internet Engineering Conf. Chiang Mai: ACM, 2014. 23−30. [doi: 10.1145/2684793.2684797]

[66] Kreutz D, Ramos FMV, Verissimo P. Towards secure and dependable software-defined networks. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. Hong Kong: ACM, 2013. 55−60. [doi: 10.1145/2491185.2491199]

[67] Giesen F, Kohlar F, Stebila D. On the security of TLS renegotiation. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security (CCS 2013). New York: ACM, 2013. 387−398. [doi: 10.1145/2508859.2516694]

[68] Das ML, Samdaria N. On the security of SSL/TLS-enabled applications. Applied Computing and Informatics, 2014,10(1-2):68−81. [doi: 10.1016/j.aci.2014.02.001]

[69] ONF. OpenFlow Technical Specifications. https://www.opennetworking.org/component/content/article/42-sdn-resources/2046-technical-resources

[70] SDxCentral. What are SDN Northbound APIs? https://www.sdxcentral.com/resources/sdn/north-bound-interfaces-api/

[71] Matsumoto C. ONF Will Tackle SDN's Northbound Interface. https://www.sdxcentral.com/articles/news/onf-decides-tackle-sdns-northbound-interface/2013/10/

[72] Oktian YE, Lee S, Lee H, Lam J. Secure your Northbound SDN API. In: Proc. of the 7th Int'l Conf. on Ubiquitous and Future Networks (ICUFN). 2015. 919−920. [doi: 10.1109/ICUFN.2015.7182679]

[73] ONF. Real Time Media NBI REST Specification (Version 1.0). 2015. https://www.opennetworking.org/images/stories/down-loads/sdn-resources/technical-reports/Real_Time_Media_NBI_REST_Specification.pdf

[74] Scott-Hayward S, O'Callaghan G, Sezer S. SDN security: A survey. In: Proc. of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS). Trento, 2013. 1−7. [doi: 10.1109/SDN4FNS.2013.6702553]

[75] Floodlight documentation. http://www.projectfloodlight.org/display/floodlightcontroller/Floodlight+Documentation

[76] POX. Python network controller. http://www.noxrepo.org/pox/about-pox/

[77] Erickson D. The beacon openflow controller. In: Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. Hong Kong: ACM, 2013. 13−18. [doi: 10.1145/2491185.2491189]

[78] Cai Z, Cox AL, Ng TSE. Maestro: A system for scalable openflow control. TSEN Maestro-Technical Report, TR10-08, 2011. http://www.cs.rice.edu/~eugeneng/papers/TR10-11.pdf

[79] Banikazemi M, Olshefski D, Shaikh A, Tracey J, Wang G. Meridian: An SDN platform for cloud network services. Communications Magazine, 2013,51(2):120−127. [doi: 10.1109/MCOM.2013.6461196]

[80] Saikia D, Kong S, Malik N, Kim D. OpenMuL: High Performance SDN. http://www.openmul.org/

[81] Tootoonchian A, Gorbunov S, Ganjali Y, Casado M, Sherwood R. On controller performance in software-defined networks. In: Hot-ICE 2012 Proc. of the 2nd USENIX Conf. on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. San Jose: USENIX, 2012. 1−6.

[82] Ferguson AD, Guha A, Liang C, Fonseca R, Krishnamurthi S. Participatory networking: An API for application control of SDNs. In: Proc. of the ACM SIGCOMM 2013 Conf. on SIGCOMM. Hong Kong: ACM, 2013. 327−338. [doi: 10.1145/2486001.2486003]

[83] NEC. ProgrammableFlow Controller. http://www.nec.com/en/global/prod/pflow/controller.html

[84] Shin S, Song Y, Lee T, Lee S, Chung J, Porras P, Yegneswaran V, Noh J, Kang BB. Rosemary: A robust, secure, and high-performance network operating system. In: Proc. of the 2014 ACM SIGSAC Conf. on Computer and Communications Security. Scottsdale: ACM, 2014. 78−89. [doi: 10.1145/2660267.2660353]

[85] Ryu SDN Framework Community. Ryu: Component-Based software defined networking framework. 2014. http://osrg.github.io/ryu/

[86] IRIS Research Group. OpenIRIS: The recursive SDN OpenFlow controller by ETRI. http://openiris.etri.re.kr/

[87] Trema: Full-Stack OpenFlow framework in Ruby and C. http://trema.github.io/trema/

[88] Koponen T, Casado M, Gude N, Stribling J, Poutievski L, Zhu M, Ramanathan R, Iwata Y, Inoue H, Hama T, Shenker S. Onix: A distributed control platform for large-scale production networks. In: Proc. of the 9th USENIX Conf. on Operating Systems Design and Implementation. Vancouver: USENIX Association, 2010. 1−6.

[89] Berde P, Gerola M, Hart J, Higuchi Y, Kobayashi M, Koide T, Lantz B, O'Connor B, Radoslavov P, Snow W, Parulkar G. ONOS: Towards an open, distributed SDN OS. In: Proc. of the 3rd Workshop on Hot Topics in Software Defined Networking. Chicago: ACM, 2014. 1−6. [doi: 10.1145/2620728.2620744]

[90] Phemius K, Bouet M, Leguay J. DISCO: Distributed multi-domain SDN controllers. In: Proc. of the 2014 IEEE Network Operations and Management Symp. (NOMS). 2014. 1−4. [doi: 10.1109/NOMS.2014.6838330]

[91] Matsumoto S, Hitz S, Perrig A. Fleet: Defending SDNs from malicious administrators. In: Proc. of the 3rd Workshop on Hot Topics in Software Defined Networking. Chicago: ACM, 2014. 103−108. [doi: 10.1145/2620728.2620750]

[92] HP. HP SDN Controller Architecture. 2013. http://h17007.www1.hp.com/docs/networking/solutions/sdn/devcenter/06_-_HP_SDN_Controller_Architecture_TSG_v1_3013-10-01.pdf

[93] Tootoonchian A, Ganjali Y. HyperFlow: A distributed control plane for openflow. In: Proc. of the 2010 Internet Network Management Conf. on Research on Enterprise Networking. Berkeley: USENIX Association, 2010. 1−6.

[94] Yeganeh SH, Ganjali Y. Kandoo: A framework for efficient and scalable offloading of control applications. In: Proc. of the 1st Workshop on Hot Topics in Software Defined Networks. Helsinki: ACM, 2012. 19−24. [doi: 10.1145/2342441.2342446]

[95] Koponen T, Amidon K, Balland P, Mart, Casado N, Chanda A, Fulton B, Ganichev I, Gross J, Gude N, Ingram P, Jackson E, Lambeth A, Lenglet R, Li S, Padmanabhan A, Pettit J, Pfaff B, Ramanathan R, Shenker S, Shieh A, Stribling J, Thakkar P, Wendlandt D, Yip A, Zhang R. Network virtualization in multi-tenant datacenters. In: Proc. of the 11th USENIX Conf. on Networked Systems Design and Implementation. Seattle: USENIX Association, 2014. 203−216.

[96] Botelho F, Bessani A, Ramos FMV, Ferreira P. On the design of practical fault-tolerant sdn controllers. In: Proc. of the 3rd European Workshop on Software Defined Networks (EWSDN). 2014. 73−78. [doi: 10.1109/EWSDN.2014.25]

[97] Monaco M, Michel O, Keller E. Applying operating system principles to SDN controller design. In: Proc. of the 12th ACM Workshop on Hot Topics in Networks. College Park: ACM, 2013. 1−7. [doi: 10.1145/2535771.2535789]

[98] Gu G, Porras P, Yegneswaran V, Fong M, Lee W. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In: Proc. of the 16th USENIX Security Symp. on USENIX Security Symp. Berkeley: USENIX Association, 2007. 167−182.

[99] Sherwood R, Gibb G, Yap K, Appenzeller G, Casado M, McKeown N, Parulkar G. Flowvisor: A network virtualization layer. OpenFlow Switch Consortium, Technical Report, 2009.

[100] Dutertre B, de Moura L. Integrating simplex with DPPL(T). 2006. http://yices.csl.sri.com/papers/sri-csl-06-01.pdf

[101] Dutertre B, de Moura L. A fast linear-arithmetic solver for DPLL(T). Computer Aided Verification, 2006,4144:81−94. [doi: 10.1007/11817963_11]

[102] Ball T, Bj N, Rner, Gember A, Itzhaky S, Karbyshev A, Sagiv M, Schapira M, Valadarsky A. VeriCon: Towards verifying controller programs in software-defined networks. In: Proc. of the 35th ACM SIGPLAN Conf. on Programming Language Design and Implementation. Edinburgh: ACM, 2014. 282−293. [doi: 10.1145/2594291.2594317]

[103] Canini M, Venzano D, Pere P, Ni, Kosti D, Rexford J. A NICE way to test openflow applications. In: Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation. San Jose: USENIX Association, 2012. 1−14.

[104] Hu H, Han W, Ahn G, Zhao Z. FLOWGUARD: Building robust firewalls for software-defined networks. In: Proc. of the 3rd Workshop on Hot Topics in Software Defined Networking. Chicago: ACM, 2014. 97−102. [doi: 10.1145/2620728.2620749]

[105] Suh M, Park SH, Lee B, Yang S. Building firewall over the software-defined network controller. In: Proc. of the 16th Int'l Conf. on Advanced Communication Technology (ICACT). IEEE, 2014. 744−748. [doi: 10.1109/ICACT.2014.6779061]

[106] Sherwood R, Gibb G, Yap K, Appenzeller G, Casado M, McKeown N, Parulkar G. Can the production network be the testbed. In: Proc. of the 9th USENIX Conf. on Operating Systems Design and Implementation. Vancouver: USENIX Association, 2010. 1−6.

[107] Sherwood R, Naous J, Seetharaman S, Underhill D, Yabe T, Yap K, Yiakoumis Y, Zeng H, Appenzeller G, Johari R, McKeown N, Chan M, Parulkar G, Covington A, Gibb G, Flajslik M, Handigol N, Huang T, Kazemian P, Kobayashi M. Carving research slices out of your production networks with OpenFlow. ACM SIGCOMM Computer Communication Review, 2010,40(1):129−130. [doi: 10.1145/1672308.1672333]

[108] Cui JS, Guo C, Chen L, Zhang YN, Huang DJ. Establishing process-level defense-in-depth framework for software defined networks. Ruan Jian Xue Bao/Journal of Software, 2014,25(10):2251−2265 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4682.htm [doi: 10. 13328/j.cnki.jos.004682]

[109] ONF. Principles and practices for securing software-defined networks. 2015. https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Principles_and_Practices_for_Securing_Software-Defined_Networks_applied_to_OFv1.3.4_V1.0.pdf

[110] ONF. SDN security considerations in the data center (Solution Brief). 2013. https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-security-data-center.pdf

[111] IETF. Policy architecture and framework for NFV infrastructures. 2015. https://tools.ietf.org/html/draft-irtf-nfvrg-nfv-policyarch-02

[112] IETF. SPRING OpenFlow interworking requirements. 2015. https://tools.ietf.org/html/draft-khc-spring-openflow-interworking-req-01

[113] IETF. Verification of NFV services: Problem statement and challenges. 2015. http://www.potaroo.net/ietf/html/ids/draft-irtf-nfvrg-service-verification-00.txt

[114] ITU. ITU Telecommunication Standardization Sector. http://www.itu.int/en/ITU-T/Pages/default.aspx

[115] European Telecommunications Standards Institute. http://www.etsi.org/

[116] China Communications Standards Association. http://www.ccsa.org.cn/

附中文参考文献:

[5] 李丹,陈贵海,任丰原,蒋长林,徐明伟.数据中心网络的研究进展与趋势.计算机学报,2014,(2):259−274.

[23] 王鹃,王江,焦虹阳,王勇,陈诗雅,刘世辉,胡宏新.一种基于 OpenFlow 的 SDN 访问控制策略实时冲突检测与解决方法.计算机学报,2015,38(4):872−883.

[44] 戴彬,王航远,徐冠,杨军.SDN 安全探讨:机遇与威胁并存.计算机应用研究,2014,(8):2254−2262. [doi: 10.3969/j.issn.1001-3695.2014.08.003]

[45] 张朝昆,崔勇,唐嚣嚣,吴建平.软件定义网络(SDN)研究进展.软件学报,2015,26(1):62−81. http://www.jos.org.cn/1000-9825/4701.htm [doi: 10.13328/j.cnki.jos.004701]

[46] 左青云,陈鸣,赵广松,邢长友,张国敏,蒋培成.基于 OpenFlow 的 SDN 技术研究.软件学报,2013,24(5):1078−1097. http://www.jos.org.cn/1000-9825/4390.htm [doi: 10.3724/SP.J.1001.2013.04390]

[108] 崔竞松,郭迟,陈龙,张雅娜,Dijiang Huang.创建软件定义网络中的进程级纵深防御体系结构.软件学报,2014,25(10):2251−2265. http://www.jos.org.cn/1000-9825/4682.htm [doi: 10.13328/j.cnki.jos.004682]

王蒙蒙(1988－),女,河南商丘人,博士生,CCF 学生会员,主要研究领域为软件定义网络安全,密码学.

毛剑(1978－),女,博士,讲师,主要研究领域为信息与网络安全(云安全,Web 安全,移动安全).

刘建伟(1964－),男,博士,教授,博士生导师,主要研究领域为信息与网络安全,密码学.

毛可飞(1977－),男,博士生,主要研究领域为网络安全协议,优化算法,软件仿真.

陈杰(1985－),男,博士生,主要研究领域为软件定义网络安全.