计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20170389
54(10):2356-2368, 2017

基于网络资源管理技术的 SDN DoS 攻击动态防御机制

王 涛 陈鸿昶 程国振
(国家数字交换系统工程技术研究中心 郑州 450002)
(wangtaogenuine@163.com)

A Dynamic Defense Mechanism for SDN DoS Attacks Based on Network Resource Management Technology

Wang Tao, Chen Hongchang, and Cheng Guozhen
(*National Digital Switching System Engineering and Technological Research Center*, *Zhengzhou* 450002)

Abstract Software defined networking (SDN) has quickly emerged as a new communication network management paradigm and greatly changed the traditional network architecture. It provides fine-grained network management service by decoupling the control plane from the data plane. However, due to the separation of control plane from data plane, controller is easy to be the attacking target of DoS. To address this problem, we make a comprehensive research on DoS attacks in SDN, and propose MinDoS, a lightweight and effective DoS mitigation method. MinDoS mainly contains two key techniques/modules: simplified DoS detection module and priority manager. MinDoS can divide flow requests into multiple buffer queues with different priorities according to the users' trust values. For a better protection towards controller under DoS attacks, this method then uses the SDN controller to schedule processing these flow requests by a dual polling mechanism. In addition, the design of MinDoS is also combined with dynamic controller assignment strategy so as to minimize the average response time of the control plane and improve the quality of service. Finally, we evaluate the performance of MinDoS in the single controller experimental environment and multi-controller experimental environment respectively. The experimental results show that the defense effect of MinDoS works well and the designed system meets the design objective basically.

Key words software defined networking (SDN); denial-of-service (DoS) attacks; multi-priority queues; dual polling mechanism; quality of service (QoS)

摘 要 软件定义网络(software defined networking, SDN)已经迅速成为一种新的网络通信管理模式,极大地改变了传统网络架构.SDN 可以通过将控制层与数据层分离来实现更细粒度的网络控制与管理.但是,转控分离的 SDN 架构也使得控制器极易成为 DoS 攻击的目标.为解决这一问题,现对 SDN 中的 DoS 攻击进行全面的研究,并提出一种轻量有效的 MinDoS 防御机制,该机制主要由简化的 DoS 攻击探测模块和优先级管理模块这 2 个核心模块实现.该机制可以根据用户信任值将流请求分类并将其划分到具有不同优先级的多个缓冲队列,然后使用 SDN 控制器以双轮询机制来调度处理这些流请

收稿日期:2017-06-02;**修回日期**:2017-07-28
基金项目:国家自然科学基金创新群体项目(61521003);国家重点研发计划项目(2016YFB0800101);国家自然科学基金青年科学基金项目(61602509);河南省科技攻关计划项目(172102210615);信息工程大学新兴方向培育基金项目(2016610708)
This work was supported by the National Natural Science Foundation of China for Creative Research Groups (61521003), the National Key Research and Development Program of China (2016YFB0800101), the National Natural Science Foundation for Young Scientists (61602509), the Science and Technology Project of Henan Province (172102210615), and the Emerging Direction Fostering Fund of Information Engineering University (2016610708).

求,从而在 DoS 攻击下更好地保护控制器.另外,MinDoS 还结合了多控制器动态调度策略来降低全局响应时间,提高用户服务质量.最后,分别在 SDN 单控制器和多控制器实验环境中对 MinDoS 防御性能进行综合评估,实验结果表明:MinDoS 防御效果良好,系统设计满足预期目标.

关键词　软件定义网络;拒绝服务攻击;优先级队列;控制器双轮询机制;服务质量

中图法分类号　TP391

随着网络信息时代到来,网络群体规模不断扩大,用户需求及网络流量急剧增加,传统网络体系架构逐渐不能适应云计算、大数据、虚拟化等新型网络技术发展需求[1].在此背景下,软件定义网络架构(software defined networking, SDN)由美国斯坦福大学 Clean Slate 研究组设计研发,其核心思想是将传统网络中的控制平面与数据平面分离,提高网络可编程性和灵活性,便于实现细粒度网络控制管理.因此,SDN 作为一种新型网络体系架构近年来得到极大发展.当然,随着 SDN 在云数据中心等实际应用场景中广泛部署,相应安全问题[2]也逐渐显现,SDN 安全也成为近年来研究热点问题.其中,针对 SDN 逻辑中心化控制器的拒绝服务攻击[3](denial of service, DoS)因攻击影响大、解决难度高而广受关注.

针对 SDN 控制器的 DoS 攻击是传统 DoS 攻击的一种衍生攻击技术.在传统 DoS 攻击中,攻击者定向发送大量无效数据请求到指定目标,造成目标服务过载,从而达到攻击目的.同样,在 SDN 中,由于控制器作为 SDN 网络的"大脑"承担着网络请求策略制定等核心功能,因此攻击者也可以通过类似攻击方式大量消耗控制器计算资源,影响正常数据请求处理,从而使整个 SDN 网络"瘫痪".随着 SDN 技术广泛应用,针对控制器的 DoS 攻击态势逐年上升,造成的损失也逐渐增加,因此,提出一种专门针对 SDN 控制器的 DoS 攻击防御方案[4]迫在眉睫.

本文基于多优先级队列与控制器双轮询处理机制设计了一种针对 SDN 控制器 DoS 攻击的动态防御机制.该机制能够有效保证控制器免受大量无效恶意数据包影响,为网络内所有用户合理分配资源,动态实现网络服务质量(quality of service, QoS)最优化.本文的主要贡献有 5 个方面:

1) 建立针对 SDN 控制器的 DoS 攻击威胁模型;

2) 梳理现有防御机制发展趋势及脉络,并总结各解决方案优劣性;

3) 在控制器端设计了一种基于多优先级队列与控制器双轮询机制的 SDN DoS 攻击动态防御机制,在保证 QoS 的同时有效防止控制器过载,而且动态防御机制能够根据网络流量实际情况,动态调整网络部署,实现全局网络 QoS 最优化;

4) 在开源 SDN 控制器 Floodlight 上实现相应防御机制并在实际 SDN 环境下测试以验证防御效果;

5) 从控制器端设计并开发防御机制符合 SDN 安全发展的主流趋势,减少对数据平面更改,有利于该防御机制广泛部署,便于 SDN 标准化.

1　SDN 网络架构及 DoS 攻击安全威胁模型

SDN 网络架构将控制平面与数据平面分离,提高了网络可编程性和灵活性,降低了网络运维成本,加速相关网络产业及技术的发展.

SDN 的典型架构主要由"三层两接口"组成:三层按照架构由上至下(由南至北)的逻辑顺序分别为应用层、控制层和数据层;两接口分别为应用层与控制层之间的北向接口(northbound interface, NBI)和控制层与数据层之间的南向接口(southbound interface, SBI).其中,应用层包括各类 SDN 应用,主要负责拓扑发现、路由、负载均衡等网络策略制定;控制层主要通过控制器执行编排数据层资源、维护全局网络拓扑和状态信息等细粒度控制操作;数据层包括 SDN 交换机等基础网络设施,主要负责数据处理、转发和状态收集等.

OpenFlow[5]协议作为 SDN 网络架构事实上统一的协议标准较好地实现了 SDN 原型设计思想.OpenFlow 协议分为主动模式和被动模式 2 种工作模式.在主动工作模式下,控制器会在网络运行前将相应网络策略分解为流规则形式提前下发至指定交换机,交换机在运行过程中根据已有流表规则转发相应数据包.主动工作模式下,由于流表项需要根据预置网络策略提前下发,且在运行过程中不能动态调整转发策略,因此,主动模式下的 SDN 网络具有明显静态性,局限性较高,不适宜实际网络部署.在被动工作模式下,控制器会根据交换机发送的数据包请求,结合网络全局状态视图动态计算转发策略,

并下发流规则到指定交换机,从而完成数据包转发.被动工作模式更能体现 SDN 灵活的网络管理方式,因此在实际部署中应用更为广泛.当然,被动工作模式的特有属性使得 SDN 控制器直接面临 DoS 攻击风险.

为了更好地分析 SDN DoS 攻击安全威胁,我们基于 OpenFlow 被动模式下的工作流程详细说明 SDN DoS 攻击原理及过程.如图 1 所示,当正常用户与服务器通信时,发送者首先发送带有源地址及目的地址的数据包请求到所连接的交换机(Step1);当交换机接收到用户请求时对数据包包头域进行解析,并在自身流表中查询该数据包有无匹配流规则(Step2),如果匹配成功则转发到指定端口,否则将触发交换机产生 Packet-in 事件,并发送 Packet-in 数据包到控制器以请求下发流规则(Step3);控制器根据全局网络状态计算转发策略,并下发流规则到指定交换机(Step4);交换机根据流规则转发数据包到服务器(Step5).

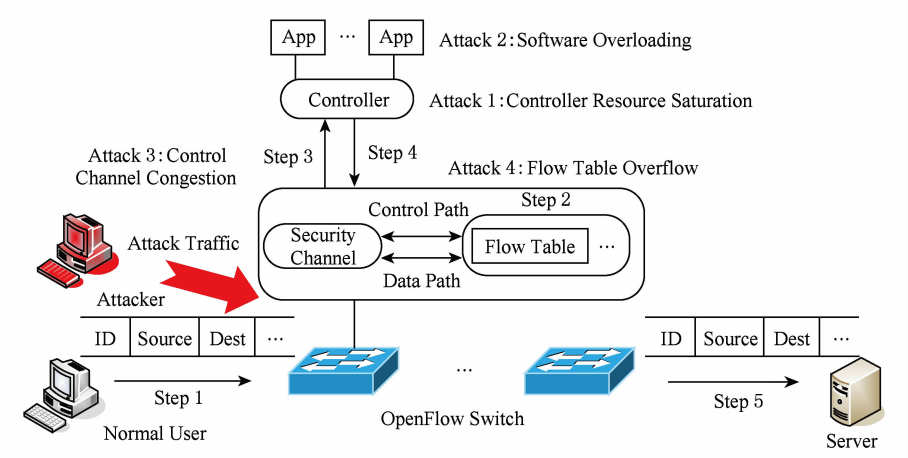

Fig. 1 SDN work processes and SDN DoS attacks

图 1 SDN 工作流程及 SDN DoS 攻击原理

在上述过程中,如果攻击者在短时间内大量发送无效虚假数据包(数据包头域以随机值填充)到交换机,交换机同样将不断被触发 Packet-in 事件并发送至控制器请求下发流规则.以上攻击过程将会造成 4 方面严重后果:

1) 控制器计算资源短时间内被大量无效虚假数据包请求消耗,正常用户数据包由于控制器过载而无法得到正常服务,造成网络服务中断(Attack 1);

2) 上层应用由于攻击请求数量过大而过载,同样会造成网络服务中断(Attack 2);

3) 控制器与交换机之间的通路因恶意攻击请求在短时间内触发大量 Pakcet-in 事件(Step3)而阻塞,正常数据包触发的 Packet-in 无法发送到控制器,从而产生拒绝服务攻击(Attack 3);

4) 交换机自身流表存储空间被大量相应攻击请求的无效流规则挤占,针对正常数据包的流规则无法安装到交换机,使正常网络用户服务中断(Attack 4).

从上述安全威胁分析中可知,针对控制器计算资源的拒绝服务攻击危害性最大.尤其对于目前广泛部署的 SDN 多控制器模型影响最为明显.随着 SDN 网络部署规模扩大,需要采用多控制器模型实现高性能和高可扩展性,然而多控制器模型更放大了此类 DoS 攻击的影响,极易造成多控制器间的连锁失败[6].

如图 2 所示,实心圆代表控制器,其处理能力为

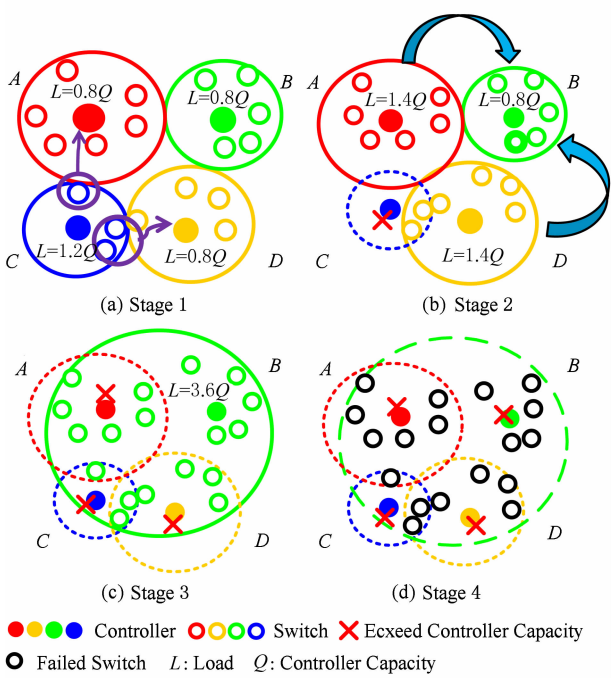

Fig. 2 A cascading failure in SDN multi-controller network

图 2 SDN 多控制器间的连锁失败

C;空心圆代表交换机.当 C 控制器因 DoS 攻击而过载时,则将其负载迁移至 A,D 控制器(状态 1 箭头).迁移后,A,D 控制器也因迁入流量而过载,因此又将负载迁移至 B 控制器(状态 2).此次迁移后,B 控制器负载大大超过其处理能力(状态 3),最终导致全部网络崩溃(状态 4).由此看出,DoS 攻击给 SDN 网络带来毁灭性灾难,解决该安全威胁刻不容缓.

2　相关工作

随着 SDN 技术的发展,业界对 SDN 安全的关注度逐渐增加.针对上述攻击场景,相关研究人员从多种角度设计防御机制,推动了 SDN 安全技术发展.

DoS 攻击在传统网络领域亦是热点及难点问题.其中一些方法也比较成熟,如异常流量检测机制[7],其一般流程包括信息收集、特征提取和分类识别,如果将该技术不经改变直接用于 SDN,则会产生一定局限性,如信息收集代价大且不灵活、需要重新定义特征向量等.客户端难题机制[8-9](client puzzle)通过难题分发、应答及验证机制,筛选出合法用户继续提供服务.此机制虽然需要消耗一定资源代价,但在传统网络中仍可有效防御 DoS 攻击.当然,此机制并不适用于 SDN 网络,客户端难题机制的分发、应答及验证过程,会产生更多 Packet-in 数据包消耗更多控制器资源,更容易造成控制器过载.因此,传统 DoS 解决方案因其局限性不能直接用于 SDN 环境下.

AVANT-GUARD[10]对 OpenFlow 数据平面进行扩展,引入了连接迁移模块和执行触发器模块.连接迁移模块可以代理 TCP 握手阶段,阻塞 TCP SYN Flood 攻击对控制器的影响.执行触发器模块可在数据平面统计数据满足一定条件时下触发插入预置的安全流规则,从而改善对动态变化流的快速处理能力.但是,AVANT-GUARD 未加强控制层安全防御机制,仅从数据平面扩展防御机制不符合 SDN 安全发展趋势.此外,该防御机制仅针对 TCP 型的 DoS 攻击有效,具有一定局限性.

FloodGuard[11]以增强控制平面安全机制为切入点,增加主动流规则分析模块和数据包迁移模块.主动流规则分析模块可以在发生 DoS 攻击时保证基本网络策略执行.数据包迁移模块可以及时缓存数据包并利用循环调度算法避免控制器过载.FloodGuard 相比于 AVANT-GUARD 能够防御各类 DoS 攻击,而且未涉及数据平面改动,有利于架构的现实部署.然而,由于 FloodGuard 控制平面安全机制对控制器负载需求增加,而其并没有扩展优化控制器处理能力,从一定程度上影响其防御效果.

文献[12-13]采用自组织图分类算法对所提取的网络流量特征分类,在识别正常流量与攻击流量后,采取相应的防御措施.虽然这些文献对流量特征及分类算法进行了优化,也取得了较高的准确率,但是此类方法性能开销较大,不能满足实时性检测要求.Yan 等人[14]基于模糊综合评价模型设计了攻击检测算法,虽然基于模糊的防御方案具有轻量级优势,但是检测准确率有待进一步增加.

通过梳理现有相关工作不难发现,针对 SDN 控制器的 DoS 攻击,目前尚没有一种防御机制能够在符合 SDN 安全发展趋势的同时满足轻量、高效、准确等特点,因此,本文基于多优先级队列与控制器双轮询处理机制设计了一种针对 SDN 控制器 DoS 攻击的动态防御机制,以推动该领域进展.

3　动态防御机制设计

为了突出整体防御架构设计思想,本文以图 3 为例详细介绍防御架构各部分功能作用.在如图 3 所示的 SDN 网络中由控制器、SDN 交换机、合法用户、攻击者和服务器 5 部分组成.为了便于叙述,3 个 SDN 交换机以线性拓扑方式连接.其中,交换机 1(S_1)下连接 3 个攻击者和 3 个合法用户,交换机 2(S_2)下连接 3 个合法用户,交换机 3(S_3)下连接 3 个合法用户及 1 台服务器.当攻击者发动 DoS 攻击时,发送大量虚假无效数据包至 S_1,S_1 产生大量 Packet-in 以消耗控制器计算资源.与此同时,3 台交换机下的合法用户均请求与服务器建立 TCP 连接.当然,由于在攻击中控制器过载可能会导致正常合法用户连接请求被丢弃.在以上攻击情景中,我们设计一种基于网络资源管理技术的 SDN DoS 攻击动态防御机制(下面简称为 MinDoS).

如图 3 所示,MinDoS 主要包括简化版 DoS 攻击检测模块(simplified DoS detection module)和优先级管理模块(priority manager)两部分.MinDoS 主要利用上述功能模块实现多逻辑队列划分、优先级管理和控制器双轮询处理机制,从而达到基本防御效果.MinDoS 工作流程如下:

Step1. 控制器划分多交换机逻辑队列；

Step2. 控制器执行基于时间切片策略的多交换机逻辑队列轮询机制(控制器第 1 阶段轮询机制)；

Step3. 网络用户优先级管理及针对每一交换机逻辑队列下多优先级逻辑队列划分；

Step4. 控制器针对每一交换机逻辑队列下的多优先级队列执行基于权重策略的轮询机制(控制器第 2 阶段轮询机制).

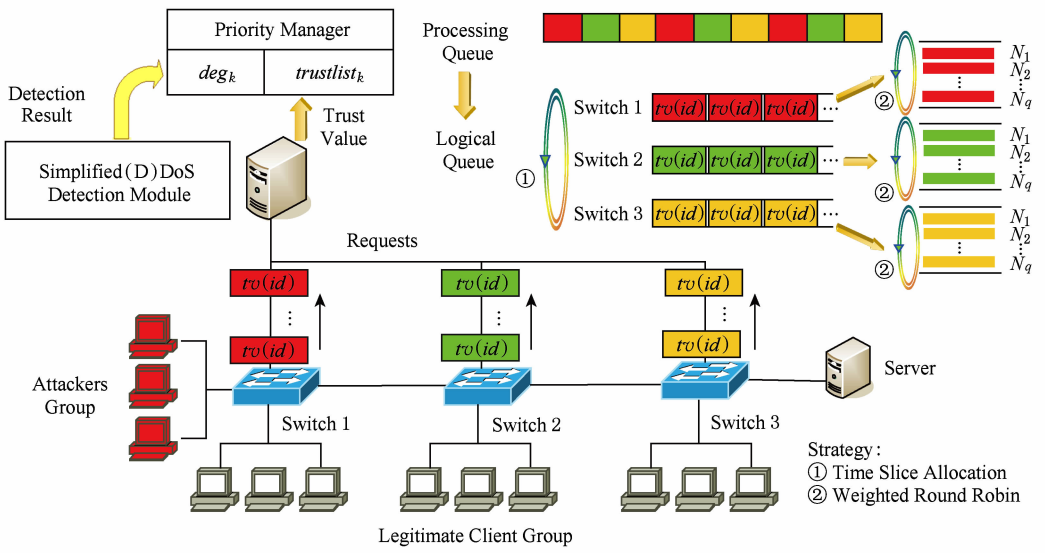

Fig. 3 The architecture and work process of MinDoS

图 3 MinDoS 架构及工作流程

3.1 控制器划分多交换机逻辑队列

传统 SDN 控制器在接收到 SDN 交换机请求消息后会将所接收的请求数据包存储在单物理队列[15](single-Q)中，并且按照“先进先出”(first come first service, FCFS)的处理原则对数据包请求回应，从而完成整个后续转发过程. 当然，此类机制具有明显漏洞，给 DoS 攻击者带来可乘之机.

传统的 single-Q 机制将网络内所有 SDN 交换机请求存储到一个单队列中，控制器再以 FCFS 机制对该队列请求处理. 在此过程中，如果 SDN 网络内任意一个交换机遭受 DoS 攻击后产生大量恶意 Packet-in 到该单队列，则控制器计算资源将会被此类恶意请求大量消耗，造成整体网络性能明显降低. 同时，由于 single-Q 缺少对不同交换机请求的隔离机制，使得 SDN 网络内其余正常交换机产生的正常请求与大量攻击请求混杂在同一队列中，使得控制器无法及时筛选并处理这些正常请求，加剧拒绝服务攻击效果.

为了增加不同交换机请求的隔离性，降低 DoS 攻击效果，本文改变传统 SDN 控制器单队列(single-Q)机制，将传统单队列划分为多逻辑队列(multi-Q)，其中，每一个逻辑队列对应一个 SDN 交换机. 通过 multi-Q 机制，不同交换机的请求排队到相应交换机的逻辑队列后，控制器根据一定策略对这些逻辑队列轮询处理(控制器第 1 阶段轮询机制在 3.2 节中详细介绍). 通过此方法，控制器可以避免因 FCFS 机制一直处理大量处理恶意请求而过载，同时，也可以将其他正常交换机的请求与被攻击交换机的攻击请求隔离，避免其他交换机受到被攻击交换机的影响，进一步提升防御效果.

3.2 基于时间切片策略的交换机队列轮询机制

控制器对于多交换机逻辑队列轮询机制需要保证全局资源分配的合理性，如果设计不当，可能会影响全局网络服务质量. 因此，本文提出一种基于时间切片策略的轮询机制，通过对全局网络进行恰当评估后进行合理资源分配，同时提升防御效果.

本文提出的时间切片策略需要结合简化版 DoS 攻击检测模块进行实现. 简化版 DoS 攻击检测模块是在现有 DoS 检测算法的基础上进行改进实现的. 通过第 2 节相关工作中可知，SGuard 通过设计全新的六元特征组向量对攻击流量进行特征提取后从而判断网络攻击状态，此方法虽然准确率较高，但负载仍不能满足我们对防御架构轻量级的要求；Yan 等人基于模糊综合评价模型设计了攻击检测算法，虽然基于模糊的防御方案具有轻量级优势，但是检测准确率有待进一步增加. 受到以上工作启发，我们结合 SGuard[12] 和模糊综合评价模型[14] 的优势，将六元特征组作为模糊综合评价的特征集，得出相应综

合评价因子,从而实现能够同时满足轻量级及准确率方面要求的简化版 DoS 攻击检测模块.该模块可针对 SDN 网络内所有交换机进行攻击评估,并对每一交换机计算出综合评价因子.综合评价因子的值介于 0～1 之间,0 表示该交换机未受到攻击者的 DoS 攻击,1 则表示该交换受到最为严重的 DoS 攻击.通过每个交换机的攻击综合评价因子,我们采取不同的时间切片分配策略,该策略表示为

$$FP_k=\begin{cases}\left\lfloor \dfrac{\overline{Deg_k}}{\sum_{i=1}^{N}\overline{Deg_i}}\times\mu \right\rfloor, & 1\leqslant k<N\\ \mu-\sum_{i=1}^{N-1}FP_i, & k=N\end{cases}, \tag{1}$$

$$\overline{Deg_k}=1-Deg_k.$$

其中,μ 表示控制器数据包请求处理能力,Deg_k 表示交换机 k 的攻击综合评价因子,FP_k 表示控制器为交换机逻辑队列 k 分配的计算资源.通过式(1)可以看出,当交换机 k 的攻击综合评价因子越小时,即交换机受到 DoS 攻击程度越低,控制器为该交换机逻辑队列分配的资源越大;反之,控制器为该交换机逻辑队列分配的资源越少.通过此策略,正常交换机下的用户 QoS 要高于被攻击交换机下的用户 QoS,基本达到了阻塞攻击者、保护合法用户的防御效果.当然,控制器第 1 阶段轮询机制在处理与攻击者连接同一交换机合法用户的请求时存在明显不对称性,需要进一步引入相应机制优化防御方案.

3.3　网络用户优先级及交换机逻辑队列管理机制

由于攻击者会造成所连接的交换机攻击综合评价因子升高,进而减少控制器对该交换机逻辑队列资源分配量,所以与攻击者连接同一交换机的合法用户所获得控制器的轮询处理能力也随之降低[15].为了解决这一矛盾,本文引入优先级管理机制,针对每一交换机逻辑队列内的请求进行优先级管理,从而进一步提升与攻击者连接同一交换机的合法用户的 QoS.Priority Manager 模块在此阶段进行网络用户信任评级后,根据用户请求数据包优先级标签将不同数据包划分至多优先级队列,从而为控制器轮询处理多优先级队列奠定基础.

3.3.1　优先级管理

为了解决控制器执行多交换机逻辑队列轮询机制时所产生的矛盾,本文需要引入优先级机制对同一交换机下的攻击者和正常用户提供差别服务.控制器通过动态调整用户信任值来区分不同用户优先级.优先级机制具体流程如算法 1 所示.

算法 1. 用户优先级(信任值)管理算法.

输入:用户 IP;

输出:用户优先级(信任值)列表 tl_i.

① in 每一时间间隔 t

② for 每一交换机 i do

③　for 用户 $s(ip)$的请求 r do

④　if 用户 $s(ip)$不在信任列表 tl_i then

⑤　为用户 s 在信任列表 tl_i 中新建表项〈ID id, IP ip, TrustValue tv〉;

⑥　关联 ip 与表项 id,并初始化 $tv(id)=1$;

⑦　else

⑧　$id\leftarrow index\ of\ (ip,tl_i)$;

⑨　end if

⑩　if 用户 $s(ip)$总请求数$>T_i$ then

⑪　$tv(id)=\lambda tv(id)-1$;

⑫　else

⑬　$tv(id)=\lambda tv(id)+1$;

⑭　end if

⑮　end for

⑯　for 优先级列表 tl_i 中用户 s do

⑰　if 用户 s 的 ip 未在时间 t 内出现 then

⑱　$id\leftarrow index\ of\ (ip,tl_i)$;

⑲　$tv(id)=\lambda tv(id)$;

⑳　end if

㉑　end for

㉒ end for

Priority Manager 为每个交换机 k 维护一个信任值列表 $trustlist_k$,列表中存储着该交换机下的所有用户(包括攻击者)的信任值信息.当用户 s 首次连接到 SDN 网络后,Priority Manager 将用户 s 的 IP 插入到相应交换机的优先级列表中,该列表主要包括表项 ID、所对应的用户 IP 和用户信任值 TrustValue 组成.首次接入到网络的用户(IP)将于信任列表中唯一的 ID 绑定,并且初始信任值设置为 1(行①～⑨).随着网络运行,信任列表中的用户信任值动态调整.当用户发送请求速率超过网络预置阈值时,意味着该用户可能成为潜在的攻击者,所以优先级管理模块对其信任值相应调整;相反,如果用户发送请求速率在网络合理范围区间内,意味着用户趋向为正常合法用户(行⑩～⑭).当然,阈值的设置具体可以根据控制器处理性能、网络拓扑大小及网络内用户需求进行综合衡量,其中文献[16]已经

对该问题进行详细叙述,在此我们不再赘述.另外,该信任评级动态调整算法与普通限速算法有着本质区别:限速算法在用户请求速率超过阈值后直接迅速采取惩罚措施,这样会使得大量正常请求被限速,误报率较高;而信任评级动态调整算法以信任值连续调整的方式调整优先级,避免不必要的惩罚,更具合理性.考虑到 DoS 攻击特性,如果用户 IP 一定时间内无任何请求,其优先级同样以一定衰减系数减低(行⑯~㉑).

3.3.2 交换机逻辑队列下多优先级队列划分机制

在优先级管理模块对用户优先级进行评定后,该用户发送的数据包均带有相应优先级标签,优先级管理模块根据发送用户的信任值标签将数据包划分至不同优先级的队列中,以便控制器第 2 阶段轮询处理.多优先级队列划分流程如算法 2 所示.

算法 2. 多优先级队列管理算法.

输入:优先级列表 tl_i、优先级队列长度 L;

输出:请求划分至相应优先级队列.

① in 每一时间间隔 t

② for 每一交换机 i do

③ for 用户 $s(ip)$的请求 r do

④ $id \leftarrow index\ of\ (ip, tl_i)$;

⑤ 请求 r 对应的优先级队列索引 N_r:

$$N_r = \left\lceil \frac{tv(id) - \min\{tv \text{ in } tl_i\}}{\max\{tv \text{ in } tl_i\} - \min\{tv \text{ in } tl_i\}} \times N_q \right\rceil;$$

⑥ if $\sum_{n=1}^{N_q} L_n < L_{\max}$ then

⑦ 将请求 r 划分至第 N_r 个优先级队列;

⑧ else

⑨ 丢弃最低优先级索引非空队列中的请求;

⑩ 将请求 r 划分至第 N_r 个优先级队列;

⑪ end if

⑪ end for

⑫ end for

优先级管理模块可将每一个交换机逻辑队列继续划分为 N_q 个优先级队列,当带有优先级标签的数据包到达时,计算该数据包所对应的优先级队列索引.其索引表达式为

$$N_r = \left\lceil \frac{tv(id) - \min\{tv \text{ in } tl_i\}}{\max\{tv \text{ in } tl_i\} - \min\{tv \text{ in } tl_i\}} \times N_q \right\rceil, \tag{2}$$

其中,N_r 表示该数据包所对应的优先级队列索引,$tv(id)$为该数据包所对应的用户信任值,$\min\{tv \text{ in } tl_i\}$表示在交换机 i 所对应优先级列表中用户信任值的最小值,$\max\{tv \text{ in } tl_i\}$表示在交换机 i 所对应优先级列表中用户信任值的最大值.从式(2)不难看出,用户信任值越高,对应的优先级索引越大(行③~⑤),当控制器进行第 2 阶段轮询时,相应索引对应的队列资源分配量越大.

另外,当新数据包插入优先级队列之前,需要先判断该交换机逻辑队列($L_{\max}$)长度是否满足要求,如果该交换机逻辑队列长度不足,则先删除最低优先级索引队列中的数据包再添加新数据包请求;如果满足长度要求,直接将请求添加到指定优先级索引队列(行⑥~⑩).通过优先级机制,正常合法用户的数据包优先级索引将会远大于攻击数据包优先级索引,因此,正常用户服务质量将会显著提升.

3.4 控制器基于权重策略的多优先级队列轮询机制

当优先级管理模块将每一交换机逻辑队列继续划分为多优先级队列后,控制器需要继续对多优先级队列中的请求进行轮询处理.为了提高控制器在单位时间切片内对高优先级队列处理量(控制器对优先级队列的资源分配量),改善与攻击者在同一交换机的正常合法用户的服务质量,本文设计了控制器基于权重策略的多优先级队列轮询机制(控制器第 2 阶段轮询机制).其工作流程如算法 3 所示.

算法 3. 基于权重策略的多优先级队列轮询算法.

输入:第 1 轮控制器资源分配量 FP_i、逻辑队列;

输出:优先级队列处理权重及相应资源分配量.

① in 每一时间间隔 t

② for 每一交换机 i do

③ for $n=1$ to N_q do

④ 计算每一优先级队列权重:

$$\omega_n = \left\lceil \frac{L_n}{L_{\max}} \delta^{n-1} \right\rceil;$$

⑤ end for

⑥ 控制器每一轮在 n_{th} 队列的处理请求数目:

$$FP_i \times \omega_n \Big/ \sum_{n=1}^{N_q} \omega_n;$$

⑦ end for

从算法 3 可知,为减少正常合法用户受到来自同一交换机下的攻击者的攻击影响,提高正常用户服务质量,减缓或者忽略攻击者攻击请求,控制器每轮对不同优先级队列的处理量与优先级队列权重成正比(行⑥).不同优先级队列的权重由优先级索引与队列长度共同决定:

$$\omega_n = \left\lceil \frac{L_n}{L_{\max}} \delta^{n-1} \right\rceil, \tag{3}$$

其中,ω_n 为优先级索引为 n 的优先级队列权重,L_n 为优先级索引为 n 的优先级队列长度,$L_{\max}$ 为该交换机逻辑队列总长度,δ 为权重比例因子. 从式(3)可以看出,优先级索引数越大,该优先级队列权重越大;优先级队列长度越大,该优先级队列权重越大. 优先级索引是队列权重首要决定性因素(指数级影响因素),当然,优先级队列长度也对权重起一定作用. 通过此种权重策略设计,可以兼顾优先级索引及优先级排队长度的影响,有助于提高用户服务质量.

4　控制器动态调度策略设计

随着云数据中心[17]等 SDN 网络部署规模不断扩大,业界普遍采用 SDN 多控制器模型[18-19]实现高性能及高可扩展性. 但是,SDN 多控制模型对于针对控制器的 DoS 攻击具有明显脆弱性(见第 1 节),因此更需要部署相关 SDN DoS 防御机制增强其可用性. 本文第 3 节所设计的防御机制完全适合部署于该 SDN 多控制器模型,可以有效防御针对控制器的 DoS 攻击,避免控制器单点故障[20]问题. 当然,该防御机制由于引入了优先级管理等安全机制,会相对增加控制器负载,另外,考虑到 DoS 攻击具有一定随机性(可以在随机时间、随机地点以随机攻击速率对随机控制器发动攻击),因此可能会造成多控制器间负载不均衡问题,影响全局网络平均响应时间,降低全局用户服务质量. 为进一步优化本文所提出的防御机制应用于 SDN 多控制器模型下的防御效果,本节同时提出一种控制器动态调度策略来保证全局网络状态最优化,提高攻击防御效率.

4.1　控制器动态调度模型建立

我们假设 SDN 网络内由 M 个控制器和 N 个交换机构成,控制器集合和交换机集合分别以集合 $C=\{c_1,c_2,\cdots,c_m\}$ 和 $S=\{s_1,s_2,\cdots,s_n\}$ 表示,其中 c_m 和 s_n 分别表示第 m 个控制器和第 n 个交换机. $y(t)_{ij}$ 表示第 i 个交换机与第 j 个控制器是否连接(1 表示连接,0 表示未连接),d_{ij} 则表示第 i 个交换机与第 j 个控制器距离(跳数). 控制器集合 C 中各控制器处理能力为 $\mu=\{\mu_1,\mu_2,\cdots,\mu_m\}$. 另外,为增加控制器可靠性及弹性,本文设置各控制器处理能力衰减系数 $\gamma=\{\gamma_1,\gamma_2,\cdots,\gamma_m\}$,其中 $\gamma\in(0,1)$.

4.1.1　全局控制器平均处理时间

对于 $i\in(1,N)$,若第 i 个交换机在时刻 t 以速率 $\upsilon(t)_i$ 向控制器发送请求时,控制器 j 在时刻 t 平均负载可表示为

$$\theta(t)_j = \sum_{i=1}^{N} \upsilon(t)_i y(t)_{ij}. \tag{4}$$

结合式(4),同时考虑到网络拓扑规模对控制器平均处理时间的影响,控制器 j 的平均请求处理时间为

$$\vartheta(t)_j = \frac{1}{\mu_j - \theta(t)_j} O(V^2), \tag{5}$$

其中,V 表示网络拓扑规模(即网络规模节点数).

根据式(4)、(5)可得全局控制器请求平均处理时间为

$$\xi(t) = \frac{\sum_{j=1}^{M} \theta(t)_j \vartheta(t)_j}{\sum_{j=1}^{M} \theta(t)_j}. \tag{6}$$

4.1.2　控制器动态调度模型

控制器动态调度模型的目的是在给定多控制器模型下,根据交换机请求负载变化情况,动态调整所连接的控制器,从而降低全局控制器平均处理时间,最优化全局用户服务质量. 该模型可以抽象为控制器动态调度分配问题:

$$\text{Minimize} \quad \xi(t) \tag{7}$$

$$\text{s.t.} \quad \theta(t)_j \leqslant \gamma_j \mu_j, \ \forall j, \tag{8}$$

$$\sum_{j=1}^{M} y(t)_{ij} = 1, \ \forall i, \tag{9}$$

$$y(t)_{ij} \in \{0,1\}, \forall i,j. \tag{10}$$

目标式(7)保证全局控制器平均处理时间最优;不等式(8)约束所有控制器均不过载;约束条件式(9)保证每个交换机必须连接且只连接一个控制器,确保有效性和唯一性.

4.2　控制器动态调度模型求解

由于控制器动态调度模型面向 SDN 多控制器模型,因此对于该模型求解需满足高效性以确保在网络在动态环境中迅速收敛,获得全局最优控制器-交换机映射关系,使得多控制器间负载相对平衡,提高网络性能及防御机制防御效果.

为求解最优控制器动态调度模型,本文首先令控制器与交换机进行“双向选择”形成初始映射关系,然后再利用迭代算法动态调整控制器-交换机映射关系实现全局网络性能最优化. 下面本节将以上求解算法进行详细介绍.

4.2.1 控制器与交换机“双向选择”机制

本文采用“双向选择”机制来构建控制器与交换机初始映射关系.具体来说,控制器和交换机各自维护一个偏好[21]列表,在满足全局约束条件的情况下,以各自偏好列表构建初始映射关系.

从交换机角度分析可知,交换机优先选择控制器处理能力高、处理时间短的控制器(如式(5)),但是该因素与其他交换机连接情况耦合性较大,不便于实际选取,因此本文以控制器最长处理时间为指标构建每一交换机偏好列表,第 j 个控制器最长处理时间为

$$\vartheta(t)_j^{\max}=\frac{1}{\mu_j-\gamma_j\mu_j}. \tag{11}$$

根据式(11),在第 i 个交换机的偏好列表 $\Gamma(s_i)$ 中所有控制器以其最长处理时间大小按照升序排列,在该偏好列表中索引越小,表明交换机 i 对该控制器偏好程度越高,相应的控制器最长处理时间也越短.即:

$$\begin{aligned}&\Gamma(s_i)=\{c_{j^*},c_j\},\forall j\neq j^*,\\&\vartheta(t)_{j^*}^{\max}<\vartheta(t)_j^{\max}.\end{aligned} \tag{12}$$

从控制器角度分析可知,控制器优先选择请求速率小且距离其跳数小的交换机,因此在第 j 个控制器的偏好列表中所有交换机以请求速率和距离第 j 个控制器跳数的乘积为指标按照升序排列,在该控制器偏好列表中,交换机所对应的索引值越小,其被控制器 j 所选中的优先级越高.即:

$$\begin{aligned}&\Gamma(c_j)=\{s_{i^*},s_i\},\forall i\neq i^*,\\&\upsilon(t)_{i^*}\times d_{i^*j}<\upsilon(t)_i\times d_{ij}.\end{aligned} \tag{13}$$

当然,在上述情况中,如果控制器 j 想将第 i^* 个交换机列入初始映射中,则控制器 j 需要满足在列入第 i^* 个交换机后不能超过控制器负载,即:

$$\forall s_{i^*},\theta(t)_j+\upsilon(t)_{i^*}\leqslant\gamma_j\mu_j. \tag{14}$$

综上所述,控制器与交换机“双向选择”机制算法流程如算法 4 所示.

算法 4. 控制器-交换机初始映射机制算法.

输入:$\forall i,\upsilon(t)_i,\forall j,\mu_j,\gamma_j$;

输出:$\forall i,j,y(t)_{ij}$.

① function $MutualChoice(\upsilon(t)_i,\mu_j,\gamma_j)$

② 每一交换机和控制器分别建立 $\Gamma(s_i),\Gamma(c_j)$;

③ while 交换机存在任意提议 do

④ if 所有提议不违反约束条件 8 then

⑤ 控制器接受交换机所有提议;

⑥ else

⑦ 控制器仅接受 $\Gamma(s_i)$ 优先级最高的提议;

⑧ 控制器拒绝其他交换机提议;

⑨ end if

⑩ end while

⑪ 转化映射 Θ 到 $y(t)_{ij},\forall i,j$

⑫ end function

4.2.2 最优化映射算法

通过上述控制器与交换机“双向选择”机制可以得到控制器与交换机的初始映射关系 Θ,该初始映射关系并不能满足最优化效果,因此,本节通过确定交换机迁移规则并利用迭代算法得到控制器与交换机最优化映射关系.

迁移规则:假设交换机 s 在初始映射关系 Θ 中与控制器 a 对应,在满足一定迁移规则后,交换机 s 在更新后的映射关系 Θ 中对应控制器 b.上述过程以 $transfer(s,a,b)$ 表示.在迁移前,控制器 a 所映射的交换机包括 s;迁移后,在控制器 a 所映射的交换机中删除 s,同时在控制器 b 所映射的交换机集合中添加 s.迁移规则根据全局控制器处理时间最优化目标确定.若根据迁移后的控制器与交换机映射关系,全局控制器处理时间降低,则满足迁移规则,同时更新控制器与交换机映射关系.上述迁移过程数学表达为

迁移前:$transfer(s,a,b)$

$$S_a=\Theta(a),\ S_b=\Theta(b); \tag{15}$$

迁移后:$transfer(s,a,b)$

$$\begin{aligned}S_{a^*}&=\Theta(a^*)=S_a\backslash\{s\},\\S_{b^*}&=\Theta(b^*)=S_b\cup\{s\};\end{aligned} \tag{16}$$

迁移规则:

$$TR(s,a,b)=\vartheta(t)_{a^*}+\vartheta(t)_{b^*}-[\vartheta(t)_a+\vartheta(t)_b]<0. \tag{17}$$

根据以上迁移规则,我们设计算法 5 动态调整控制器与交换机映射关系[22],在基于控制器-交换机初始映射关系及相关初始状态参数上求解出具有最小迁移规则值的迁移对(transfer pair)后,更新映射关系,实现全局控制器响应时间最优效果.

算法 5. 控制器-交换机映射动态调整机制算法.

输入:$\forall j,\mu_j,\gamma_j$、初始映射 Θ;

输出:$\forall i,j,y(t)_{ij}$.

① function $transfer(\Theta,\mu_j,\gamma_j)$

② for 控制器,$\forall j,c_j$ do

③ for 交换机 $s_i\in\Theta(c_j)$ do

④ for 控制器 $c_m\in C\backslash\{c_j\}$ do

⑤ 找到具有最小 $TR(s_i,c_j,c_m)$ 值的迁移对(s_i,c_j,c_m);

⑥　　end for
⑦　　if $TR(s_i,c_j,c_m)<0$ then
⑧　　update:$\Theta\leftarrow transfer(s_i,c_j,c_m)$
⑨　　end if
⑩　end for
⑪　end for
⑫　Transform Θ to $y(t)_{ij}$;
⑬ end function

5　实验验证与分析

为验证上述机制有效性,本文分别在 SDN 单控制器的模拟环境和基于 SDN 多控制器模型的实际数据中心下进行实验验证与分析.本节将详细介绍 2 种实验环境下实验设计思路并进行实验结果分析.

5.1　SDN 单控制器环境下实验及结果分析

在单控制器环境中,我们基于开源的 Floodlight 控制器部署本文所提出的安全机制.实验拓扑如图 3 所示,3 个 SDN 交换机以线性拓扑方式连接.其中,交换机 1(S_1)下连接 3 个攻击者和 3 个合法用户,交换机 2(S_2)下连接 3 个合法用户,交换机 3(S_3)连接 3 个合法用户及 1 台服务器.交换机 $S_1\sim S_3$ 下所连接的用户分别部署到 3 台服务器上,每台服务器均配置 Intel® Xeon® CPU x5690 3.47 GHz 和 48 GB RAM,运行 Ubuntu 16.04 server 版系统,交换机采用 pica8 SDN 交换机.表 1 列出了其余主要实验参数设置.

Table 1　Experimental Parameter Settings

表 1　实验参数设置

Parameters	Value
Controller Capacity μ/pkts·s^{-1}	1 000
Decay Factor of Trust Value λ	0.9
Request Rate Threshold T_i/pkts·s^{-1}	500
Number of Priority Buffer Queues N_q	5
Logical Queue Capacity of Each Switch L_{max}/pkts	1 000
Weight Scale Factor δ	2

Flooglight 控制器处理能力设置为 $\mu=1\,000$ pkts·s^{-1},为使模拟环境更加接近真实环境,正常合法用户在随机时刻以随机速率发送请求数据包,其速率范围设置为 $\mu/5\sim 3\mu/5$.我们分别在攻击者总攻击速率为 $1.5\mu,2\mu,4\mu,8\mu$ 的情况下,测试不同交换机下正常合法用户 TCP 连接失败率 R_f 及 TCP 连接建立的平均时延 D_s.为了横向对比实验效果,我们分别以先进先出机制(FCFS)、SGuard 防御机制、DoS 模糊综合评价防御机制(FSEDM)为对照,测试上述防御机制 TCP 连接失败率及平均时延.实验结果如图 4 所示:

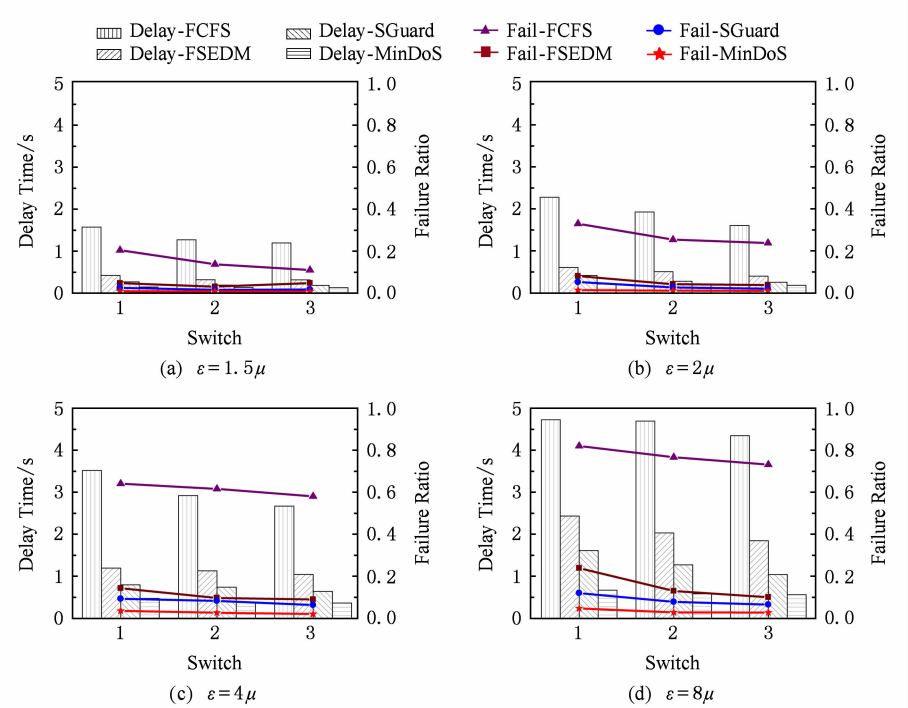

Fig. 4　Performance comparison between MinDoS and existing defense mechanisms under different DoS attack intensities

图 4　不同攻击速率下 MinDoS 与现有机制的防御性能比较

从以上实验结果可以看出,由于 FCFS 机制完全不具备防御功能,因此随着攻击速率不断增加时,其相应时延不断增加,TCP 连接失败率在 $\varepsilon=8\mu$ 时更是达到 80%,网络性能垂直下降. 相比之下,SGuard 和 FSEDM 在连接建立时延及失败率方面具有一定防御效果,在 $\varepsilon<4\mu$ 的攻击情景下,时延控制在 1 s 左右,连接失败率维持在 10%以下. 但是,SGuard 和 FSEDM 在更高强度的 DoS 攻击下,如在 $\varepsilon=8\mu$ 时,由于两者自身防御机制负载原因,时延和失败率均显著上升,其中,FSEDM 机制连接失败率甚至接近 20%,防御效果降低. MinDoS 由于采用优先级队列及控制器双轮询机制,整体防御机制较为轻量、高效,因此,在不同攻击速率下,防御性能可以基本稳定,即使在攻击强度较高的情况下,时延也能控制在 1 s 以内,失败率维持在 5%以下. 另外,在本实验中,由于交换机 1 下连接攻击者,所以在以上机制中,交换机 1 下的平均时延及连接失败率比其余交换机略高. 综上所述,MinDoS 相较于 SGuard, FSEDM 和 FCFS 机制,不管在时延还是失败率方面具有绝对优势,而且随着攻击速率不断增加,防御性能优势也逐渐明显.

5.2 SDN 多控制器模型下实验及结果分析

为检验基于多优先级队列与控制器双轮询机制的 SDN DoS 攻击动态防御机制在实际 SDN 环境中的部署效果,本文将该防御机制实际部署于数据中心中测试其防御效果. 通过 5.1 节单控制器环境下的实验结果分析可知,本文所提出的防御机制较其他机制具有明显优势,但是在单控制器环境下防御机制相对静态,没有体现出该机制应用于数据中心等实际环境中控制器动态调度策略对全局网络性能方面的动态优势,因此,本节对该防御机制应用于数据中心环境中的性能进行测试.

本文选择的数据中心拓扑为 24-pod fat-tree 结构,该拓扑下共有 720 个交换机和 3 456 个主机用户. 该数据中心内部署了 30 个 Floodlight 控制器. 各控制器弹性系数 γ_i 随机设置为 0.92～0.96. 其余实验参数设置如表 1 所示. 攻击者在该数据中心下以总用户数 5%的比例随机选取. 为了便于对比分析,本文在攻击者攻击速率分别为 1.5μ, 2μ, 4μ, 8μ 的情况下,测试并分析 MinDoS, SGuard 和 FSEDM 在该数据中心拓扑结构下的全局正常用户请求处理时间. 实验结果如图 5 所示. 其中,图 5(a)～(d)分别展示了不同攻击速率情况下的全局时间.

分析图 5(a)～(d)可知:在发动不同速率攻击之前(10 s),采用 MinDoS 控制器动态调度策略的响应时间明显低于 SGuard 和 FSEDM 机制,这是由于本文所提出的控制器动态调度策略能够有效均衡数据中心各控制器负载,使得全局响应时间最优化.

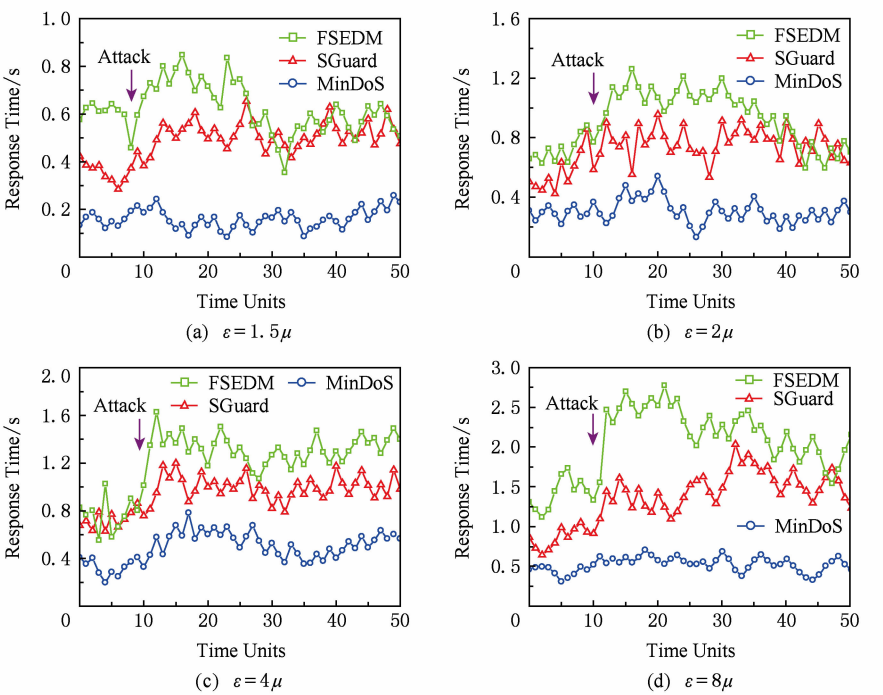

Fig. 5 Response time comparison between MinDoS and existing defense mechanisms in SDN multi-controller model under different DoS attack intensities

图 5 不同攻击速率下 SDN 多控制器模型中 MinDoS 与现有防御机制的响应时间比较

在 $t=10$ s 以后,当攻击者开始发动攻击时,图 5(a)~(d)在 SGuard 和 FSEDM 机制下的响应时间开始逐渐增加,并且该响应时间随着攻击速率升高而增加(响应时间增加幅度关系 $1.5\mu<2\mu<4\mu<8\mu$).当然,由于 SGuard 和 FSEDM 机制具有一定 DoS 攻击防御效果,因此,在攻击速率分别为 1.5μ,2μ,4μ,8μ 时,其响应时间峰值分别为{(0.7,0.85);(0.9,1.2);(1.16. 1.62);(2,2.6)},该响应时间虽然受到一定性能影响,但仍然控制在合理范围内.相较于以上 2 种防御机制,MinDoS 在该多控制器模型下具有显著优势,即使在不同攻击速率下,MinDoS 防御机制下的全局相应时间也基本维持在(0.2,0.6)区间内,浮动率较低.这是因为 MinDoS 机制从控制器处理方式角度出发,与攻击速率关联度较低,而且 MinDoS 采用控制器动态调度策略有助于优化全局网络性能.综上所述,在基于多控制器模型的小型数据中心场景中,MinDoS 在未发动攻击的条件下,其全局响应时间优于现有解决方案,证明控制器动态调度策略能够显著降低全局控制器相应时间,满足预期实验目标;MinDoS 在受到不同程度的 DoS 攻击情况下,其全局响应时间仍然优于现有解决方案,证明 MinDoS 在有效防御 DoS 攻击的同时能够显著提高系统整体性能,更加有力地证明了 MinDoS 有利于在实际真实环境中广泛部署.

6　总结和未来工作

本文为解决针对 SDN 控制器的 DoS 攻击问题,设计了一种基于网络资源管理技术的 SDN DoS 攻击动态防御机制.1)介绍了 SDN 基本架构及针对 SDN 控制器的 DoS 攻击原理,然后介绍和分析了现有针对该问题的解决方案及其自身的优缺点;2)针对现有解决方案的不足,创新性地提出优先级队列算法和控制器双轮询机制,同时,结合多控制器动态调度策略,进一步提升防御效果和网络性能;3)本文将该防御机制与现有防御机制分别在 SDN 单控制器模型和多控制模型下进行防御效果对比分析,实验结果表明:该防御机制可以有效防御针对 SDN 控制器的 DoS 攻击,并且防御性能优于现有解决方案,完全适合部署于实际 SDN 场景中.在未来的工作中,我们将继续将该防御机制部署于大型数据中心中,在实际网络流量环境中测试其防御效果,发现并解决可能存在的问题,进一步调整并优化相应防御机制.

参　考　文　献

[1] Jain R. Internet 3.0: Ten problems with current Internet architecture and solutions for the next generation [C] //Proc of MILCOM'06. Piscataway, NJ: IEEE, 2006: 1-9

[2] Sezer S, Scott-Hayward S, Chouhan P K, et al. Are we ready for SDN? Implementation challenges for software-defined networks [J]. IEEE Communications Magazine, 2013, 51(7): 36-43

[3] Kandoi R, Antikainen M. Denial-of-service attacks in OpenFlow SDN networks [C] //Proc of the 14th Int Symp on Integrated Network Management. Piscataway, NJ: IEEE, 2015: 1322-1326

[4] Li Baohui, Xu Kefu, Zhang Peng. pTrace: A counter technology of DDoS attack source for controllable cloud computing [J]. Journal of Computer Research and Development, 2015, 52(10): 2212-2223 (in Chinese)
(李保珲, 徐克付, 张鹏, 等. pTrace: 一种面向可控云计算的 DDoS 攻击源控制技术[J]. 计算机研究与发展, 2015, 52(10): 2212-2223)

[5] 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

[6] Yao Guang, Bi Jun, Guo Luoyi. On the cascading failures of multi-controllers in software defined networks [C] //Proc of the 22nd IEEE Int Conf on Network Protocols. Piscataway, NJ: IEEE, 2014: 1-2

[7] Kim M S, Kong H J, Hong S C, et al. A flow-based method for abnormal network traffic detection [C] //Proc of NOMS'04. Piscataway, NJ: IEEE, 2004: 599-612

[8] Waters B, Juels A, Halderman J A, et al. New client puzzle outsourcing techniques for DoS resistance [C] //Proc of the 11th ACM Conf on Computer and Communications Security. New York: ACM, 2004: 246-256

[9] Michalas A, Komninos N, Prasad N R, et al. New client puzzle approach for DoS resistance in ad hoc networks [C] //Proc of the 2010 IEEE Int Conf on Information Theory and Information Security. Piscataway, NJ: IEEE, 2010: 568-573

[10] Shin S, Yegneswaran V, Porras P, et al. AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks [C] //Proc of ACM CCS'13. New York: ACM, 2013: 413-424

[11] Wang Haopei, Xu Lei, Gu Guofei. FloodGuard: A DoS attack prevention extension in software-defined networks [C] //Proc of the 45th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks. Piscataway, NJ: IEEE, 2015: 239-250

[12] Wang Tao, Chen Hongchang. SGuard: A lightweight SDN safe-guard architecture for DoS attacks [J]. China Communications, 2017, 14(6): 113-125

[13] Alsulaiman M M, Alyahya A N, Alkharboush R A, et al. Intrusion detection system using self-organizing maps [C] // Proc of the 3rd Int Conf on Network and System Security. Piscataway, NJ: IEEE, 2009: 397-402

[14] Yan Qiao, Gong Qingxiang, Deng Fangan. Detection of DDoS attacks against wireless SDN controllers based on the fuzzy synthetic evaluation decision-making model [J]. Ad Hoc & Sensor Wireless Networks, 2016, 33(1): 275-299

[15] Zhang Peng, Wang Huanzhao, Hu Chengchen, et al. On denial of service attacks in software defined networks [J]. IEEE Network, 2016, 30(6): 28-33

[16] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance [J]. IEEE/ACM Trans on Networking, 1993, 1(4): 397-413

[17] Roy A, Zeng Hongyi, Bagga J, et al. Inside the social network's (datacenter) network [C] //Proc of the 2015 ACM Conf on Special Interest Group on Data Communication. New York: ACM, 2015: 123-137

[18] Yu Minlan, Rexford J, Freedman M J, et al. Scalable flow-based networking with DIFANE [C] //Proc of ACM SIGCOMM'10 Conf on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM, 2010: 351-362

[19] Koponen T, Casado M, Gude N, et al. Onix: A distributed control platform for large-scale production networks [C] // Proc of the 9th USENIX Conf on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2010: 351-364

[20] Tootoonchian A, Gorbunov S, Ganjali Y, et al. On controller performance in software-defined networks [C] // Proc of the 12th USENIX Conf on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. Berkeley, CA: USENIX Association, 2012: 10-10

[21] Liu Fangmin, Guo Jian, Huang Xiaomeng, et al. eBA: Efficient Bandwidth Guarantee Under Traffic Variability in Datacenters [J]. IEEE/ACM Trans on Networking, 2017, 51(1): 506-519

[22] Guo Jian, Liu Fangmin, Zeng Dan, et al. A cooperative game based allocation for sharing data center networks [C] // Proc of the 32nd IEEE Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2013: 2139-2147

Wang Tao, born in 1993. MS candidate at National Digital Switching System Engineering and Technological Research Center (NDSC) at Zhengzhou, China. His main research interests include software-defined networking and security.

Chen Hongchang, born in 1964. PhD, professor, PhD supervisor. His main research interests include future network communication and future network architecture (chc@mail.ndsc.com.cn).

Cheng Guozhen, born in 1986. PhD, assistant professor. His main research interests include network security (chengguozhen1986@163.com)