DOI：10.7652/xjtuxb201710009

软件定义网络流表溢出脆弱性分析及防御方法

周亚东，陈凯悦，冷俊园，胡成臣
（西安交通大学智能网络与网络安全教育部重点实验室，710049，西安）

摘要：软件定义网络交换机非常有限的流表容量使其存在严重的流表溢出脆弱性问题，为此利用软件定义网络易于管理路由规则的新特性，提出一种基于装箱优化的路由聚合算法，并进一步提出了流表溢出攻击的防御方法。采用传统的基于基数树的路由聚合算法产生初步聚合后的流表项节点，将其划分为包含不同数量节点的若干个流表项规则组，并基于装箱优化问题求解得到每个流表项规则组的新转发地址，再将转发地址修改后的流表项规则进行二次聚合，从而有效减少交换机流表中的流表项数量，达到防御流表溢出攻击的效果。实验结果表明：流表聚合率达到了54.9%，优于传统的基于基数树的路由聚合算法，并使得达成流表溢出攻击的攻击数据包数增加了125.8%；该方法可显著增加流表溢出攻击的实现难度，有效缓解流表溢出脆弱性问题，提升软件定义网络对该类攻击的防御能力。

关键词：软件定义网络；流表溢出；装箱优化；防御方法
中图分类号：TP391 **文献标志码**：A **文章编号**：0253-987X(2017)10-0053-06

Analysis and a Defense Method for Overflow Vulnerability of Flow Tables in Software Defined Networks

ZHOU Yadong, CHEN Kaiyue, LENG Junyuan, HU Chengchen
(MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: The capacity of flow tables of software defined network switches is very limited and thus there exists a serious problem of flow table overflow vulnerability. A routing algorithm based on packing optimization is proposed to solve the problem and the algorithm uses new characteristics of software defined networks. A method to defense overflow attacks of flow tables is also presented based on the proposed route aggregation algorithm. Firstly, the traditional algorithm of route aggregation based on radix tree is used to generate initial aggregated nodes of flow tables. Then, the nodes are divided into several different groups with flow table rules, and the new forwarding address for each group is then obtained based on the solution of a packing optimization problem. Finally, the flow table rules are aggregated again after modifying the forwarding addresses, so that the number of flow entries in flow tables of a switch is effectively reduced, and the effect of defensing overflow attack of flow tables is achieved. It is found from experimental results of the proposed defense method that the aggregation rate of flow tables is 54.9%, and is better than that of the classical algorithm based on the radix tree, and that the number of attack packets reaching the overflow attack increases 125.8%. The experimental

收稿日期：2017-03-15。 作者简介：周亚东(1982—)，男，讲师。 基金项目：国家自然科学基金资助项目(61572397)；陕西省自然科学基础研究计划资助项目(2016JM6066)。
网络出版时间：2017-08-19 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20170819.1125.002.html

results show that the proposed method significantly increases the difficulty to reach the flow table overflow attack, so that the problem of flow table overflow vulnerability is effectively alleviated, and the defense ability to related attacks is enhanced.

Keywords: software defined networks; flow table overflow; packing optimization; defense method

随着互联网规模的不断扩大和业务类型的不断扩展,传统的 TCP/IP 网络架构体系暴露出越来越多的问题,例如网络扩展性不足、网络安全威胁、网络管理复杂等。为了解决这些问题,各种新型网络技术在不断发展,新概念、新技术、新协议也在不断引入。21 世纪初,以美国为首的发达国家启动了多个未来网络技术研究项目,比较有代表性的有美国的 GENI 项目、欧盟的 FIRE 项目等[1],中国未来网络技术的研究也已成为重点研究领域。

在各种未来网络技术中,软件定义网络(SDN)是对现有网络技术的一种颠覆性创新,为未来网络的发展提供了一个重要的方向[2]。SDN 解除了网络中数据转发平面与网络控制平面之间的紧耦合关系,形成了应用、控制、转发三层分离的架构,并通过标准化的北向接口为开发者提供了良好的网络编程接口,使得网络管理获得了更好的灵活度和敏捷度。2012 年,谷歌公司在开放网络峰会上公布了 SDN 在数据中心网络中的实施案例 B4[3],标志着 SDN 从学术界走向了工业界,并推动了 SDN 的研究热潮。

与 SDN 的研究与应用热潮形成鲜明对照的是现有的 SDN 交换机普遍存在严重的性能瓶颈,特别是流表容量瓶颈。现有的商用 SDN 交换机内的流表均采用三态内容寻址存储器(TCAM)制成,TCAM 缓存的高成本及 OpenFlow 架构下 TCAM 的低利用率,使得 SDN 交换机中的流表容量极为有限。与其较小的流表容量相对应的是 SDN 交换机承载的网络通信极为繁忙,当攻击者通过大量伪造数据包对 SDN 交换机进行攻击时,极易导致 SDN 交换机的流表溢出,从而引发一系列严重后果。

针对这些问题,本文对 SDN 网络的流表溢出脆弱性进行了研究,分析了该脆弱性的产生原因和攻击后果,提出了基于路由聚合的流表溢出攻击防御方法,通过实验验证了该方法可以有效提升 SDN 网络对流表溢出脆弱性攻击的防御能力。

1 流表溢出的脆弱性分析

1.1 流表溢出脆弱性产生原因

目前,商用 SDN 硬件交换机内的流表均采用 TCAM 缓存制成。TCAM 缓存的成本较高,使得 SDN 交换机中的流表容量普遍较小,例如 NEC 公司的 PF5820 交换机的流表容量仅为 750 条,盛科 V350 SDN 交换机流表容量也仅为 2 000 条。交换机需要处理的网络通信极为繁忙,据统计,通常数据中心内的网络流速可以达到 $5\sim10\times10^3$ 条/s[4],使得网络流速远超过 SDN 交换机中的流表容量。此外,防火墙、负载均衡等网络应用所需要的交换机流表容量也远超过现有的商用 SDN 交换机的流表容量,因此在 SDN 网络的实际使用中,现有的 SDN 交换机流表容量无法满足需求。

在网络运行过程中,SDN 交换机需要根据每条流的特征向流表中写入对应的规则。当网络流量较大时,这种精确匹配的流表生成机制极易造成流表被完全占用。流表被完全占用后,如果有新的网络流到达,且该网络流在流表内尚无对应规则进行处理,根据 OpenFlow 协议,SDN 交换机需要与控制器进行频繁交互,将网络流信息封装在 PacketIn 消息内发送给控制器,并由控制器计算下发新的流表项。此时,交换机与控制器之间的频繁交互会造成网络性能的严重降低,甚至导致该 SDN 交换机所负责的数据平面内网络完全瘫痪。

1.2 流表溢出脆弱性攻击后果

攻击者可以通过产生随机目的地址和伪造源地址的网络流向流表中插入大量流表项,造成流表溢出,这将使得 SDN 交换机的数据包转发性能严重降低,并达成针对 SDN 交换机的流表溢出攻击。基于这一漏洞,较易实现针对 SDN 交换机的拒绝服务攻击[5],并且在作者的前期研究中还发现,采用网络流量生成、网络流量注入、网络性能测量、网络参数推断的攻击路径,可以对存在该漏洞的 SDN 交换机的硬件流表容量及其所在网络中的实时网络流量特征进行较高精度的推断,从而达成网络关键信息泄露攻击[6]。

2 流表溢出攻击的防御方法

针对流表溢出脆弱性攻击防御的问题,本文提出基于装箱优化的路由规则聚合算法。该算法借鉴经典的基于基数树的路由聚合算法,采用装箱优化方法,进一步提高了流表项的聚合效率。通过采用

该算法可有效节省交换机的 TCAM 流表容量消耗，缓解 SDN 交换机流表溢出的脆弱性问题，使攻击者达成流表溢出攻击的难度加大，从而实现对 SDN 交换机流表溢出攻击的防御。

2.1　基于基数树的路由聚合算法

随着 IP 网络的高速发展，网络路由规则数量持续增长，在互联网中转发路由表（FIB）已经达到几十万条路由规则的数量级。FIB 的膨胀为传统网络的可扩展性带来了严重问题。为减少交换机的 FIB 空间消耗，提高网络的可扩展性，越来越多的研究人员开始关注流表项聚合问题[7-9]。已有的流表项聚合算法在聚合效率方面仍有不足，需要针对 SDN 网络特性，提出改进的聚合算法。

基于基数树的路由聚合算法是一种经典的路由聚合算法[9]。基数树是一种特殊的二叉树，它的节点由匹配前缀和相应的下一跳网络地址两部分组成。由流表项构建基数树的过程为：将路由匹配前缀转换为二进制字符串；然后对该二进制字符串进行遍历，如果遇到某位是 1，则将其作为二叉树的左节点插入，如果遇到某位是 0，则将其作为二叉树的右节点插入；重复以上步骤直至二进制字符串遍历完成。基于基数树的路由聚合算法常用的聚合策略包括父子节点聚合、兄弟节点聚合、引入额外路由空间的非兄弟节点聚合和不引入额外路由空间的非兄弟节点流表聚合，但在该聚合算法中，未能在聚合之后对路由规则的转发地址做进一步优化，使得路由聚合效率偏低，影响了其对流表溢出攻击的防御能力。

2.2　基于装箱优化的路由聚合算法

在 SDN 网络中，管理者可以通过控制器中的路由规则管理算法，很方便地对路由规则的转发地址进行实时调整，这是 SDN 网络中流表聚合问题与传统网络中流表聚合问题的重要区别，因此可以通过对路由规则的转发地址做进一步聚合和实时调整，提高 SDN 网络中路由规则的聚合效率。基于这一思路，本文提出了一种基于装箱优化的路由聚合算法，包括以下 3 个主要步骤。

步骤 1　划分流表项规则组。对于采用基于基数树的路由聚合算法初步聚合后得到的所有流表项节点，将其中目的地址属于相同网络应用且匹配前缀不可聚合的若干节点划分为一个流表项规则组，从而得到包含不同数量节点的多个流表项规则组。

步骤 2　调整流表项规则组的转发地址。对于步骤 1 得到的每个流表项规则组，将作为流表项转发目标的网络设备作为箱体，将设备的处理能力、网络流量负载等作为箱体的容积限制，依照路由规则组所包含的节点数，将每一个流表项规则组作为尺寸不同的物品，则可将路由规则组与转发目标设备之间对应的流表项规则聚合问题描述为装箱优化的问题，再基于装箱优化问题求解，得到每个流表项规则组的调整后转发地址。

步骤 3　聚合流表项规则组中的各项规则。通过改写流表项规则组中每个节点的转发网络地址，使其指向调整后的目标网络设备，并再次采用基于基数树的路由聚合算法进行二次聚合，最终使得转发地址改变后形成的新的流表项规则组所包含的流表项数量减少。

需要注意的是，装箱优化问题是一种典型的组合优化问题，属于 NP 问题，即在多项式时间内无法求得最优解，因此装箱优化问题的求解一般采用近似的启发式算法，这样可以较快得到满意解。常见的装箱优化问题求解算法包括：下次适应算法、首次适应算法、最佳适应算法、降序首次适应算法、降序最佳适应算法等[10]。本文基于降序最佳适应算法对该问题进行求解，具体求解过程为：首先，对每一个路由规则组中所包含的路由规则数量及每一个路由规则组中所包含的路由规则所转发的网络总流量进行统计；然后采用降序最佳适应算法求解该装箱问题，对每个规则组转发的网络总流量降序排序，检查资源剩余的网络设备，找到一个能够承载当前网络流量并且承载此流量后剩余资源最少的设备，并将这些流量的下一跳网络地址进行调整，指向此设备，否则将这些流量引导向一个资源未被占用的新的设备；最后根据求解结果对于每一个路由规则组中各个节点的下一跳网络地址进行统一调整。

对于基于装箱优化的路由聚合算法的合理性和可行性问题，进行如下分析。

首先，对算法中划分流表项规则组需满足的 2 个条件进行分析。条件 1：划分至同一流表项规则组的所有路由规则的下一跳网络地址需属于同一网络应用。条件 2：所有路由规则对应节点的匹配前缀不可再基于基数树进行聚合。其中，条件 1 保证同一流表项规则组中的路由规则具有进一步聚合的可能性，条件 2 保证已采用基于基数树的路由聚合算法对初始路由规则进行了充分聚合。

其次，对同一流表项规则组中的路由规则可进行再次聚合的合理性进行分析。在实际网络中，存在两种情况：①如果路由规则的目的网络设备是服

务器，通过保证一个流表项规则组中规则的多个下一跳网络地址对应同一网络应用，则当对其中路由规则的转发地址进行调整时，保证其调整后下一跳地址仍对应该网络应用，此时其目的服务器相当于该应用进行负载均衡的多台服务器，可保证其功能性，因此具有合理性；②如果路由规则的目的网络设备是交换机，在进行基于所提方法的路由聚合后，通过保证流表项规则组中流表规则的下一跳网络地址属于同一应用，只是使得流量的路径数量减少，但最终仍然可到达该应用对应的服务器，因此也可以保证其功能性，使其具有合理性。

最后，对同一流表项规则组中的路由规则可进行再次聚合的可行性进行分析。在采用基于基数树的路由聚合中，需要节点之间的转发地址相同才可以根据其匹配前缀情况进行聚合。本文所提的基于装箱优化的路由聚合中将同一流表项规则组中的路由规则的转发地址进行了调整，使得同一应用对应路由规则的转发地址数量减少，从而可将大量原基数树无法聚合的路由规则再次采用基于基数树的路由聚合算法进行二次聚合，从而显著增加了路由规则的聚合率，缩短了流表使用的长度。

基于基数树的流表聚合算法的时间复杂度为 $O(n)$[9]，其中 n 是未经基数树聚合时的流表项的数量。关于本文方法中采用的基于降序最佳适应算法的装箱算法，时间复杂度为 $O(n\mathrm{lb}n)$[10]。在基于基数树聚合的实验中，最差情况下流表项可减少 30%～50%；最好情况下，可以减少 90%。在这里采用较差的情况进行分析，即未经装箱算法优化处理的流表项数为初始未经基数树算法优化的流表项数的 70%，则装箱算法的时间复杂度是 $O(0.7n\mathrm{lb}(0.7n))$。由于基于装箱优化的路由聚合算法是依次执行这 2 个算法，因此算法总的近似时间复杂度为 $O(n\mathrm{lb}n)$。通过分析可知，所提方法增加了时间消耗，但仍可满足实际应用的需要。

2.3 流表溢出攻击的防御方法

基于 2.2 节所述的基于装箱优化的路由聚合算法，提出流表溢出攻击防御方法，包括 4 个步骤：①实时监测 SDN 交换机流表占用情况，当 SDN 交换机流表容量占用超过预先设定的阈值时，启动路由聚合；②将 SDN 交换机中的流表项构建为基数树，然后采用基于基数树的路由聚合算法对路由规则进行初次聚合；③将聚合后的流表项依照其前缀和后缀，划分成不同的流表项规则组，采用装箱优化算法对不同的流表项规则组与转发目标网络设备之间的对应转发关系进行计算调整，实施二次聚合；④根据计算结果对相应的路由规则进行下发更新。以下基于图 1 所示的实例对该防御方法进行说明。

由交换机流表项构建的基数树中有多个节点，根据以下规则将这多个流表项划分为若干个流表项规则组：同一个流表项规则组中所有流表项的目的网络地址属于同一网络应用；同一个流表项规则组中所有流表项的匹配前缀不可聚合。划分得到的若干个流表项规则组用 S_1，S_2，…，S_n 表示。

将流表项规则组划分完毕后，本文算法对每一个流表项规则组中所包含的流表项数及每一个流表项规则组中所包含的流表项规则所转发的网络总流量进行统计，然后采用降序最佳适应算法求解该装箱问题，根据求解结果对于每一个流表项规则组中各个节点的下一跳网络地址进行统一调整。

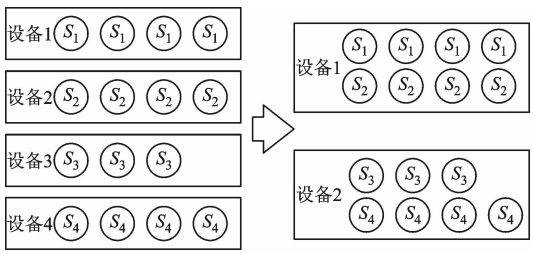

图 1 流表聚合前后对比示意图

图 1 给出了聚合前后流表项规则组与网络设备对应转发关系变化的示意图。图 1 中左边为优化前的流表项规则组，S_1、S_2、S_3、S_4 所包含节点被分别分配至设备 1、设备 2、设备 3 和设备 4。图 1 中箭头右侧为优化后的流表项规则组，S_1 与 S_2 所包含的 8 个节点被分配至设备 1，S_3 与 S_4 所包含的 7 个节点被分配至设备 2。在这一过程中，流表项被进一步聚合，从而有效降低了对流表容量的占用，提升了 SDN 交换机对流表溢出攻击的防御能力。

3 实验分析

3.1 实验环境

实验环境由若干台 PC 机构成，其中 PC_1 配有 Intel i7 3770 处理器与 16 GB 内存，安装 Windows 10 操作系统，用于运行 SDN 控制器 Floodlight，在负载均衡相关测试中实现网络控制平面的功能；PC_2 配有 Intel i5 2400 处理器与 8 GB 内存，用于运行 SDN 控制器 POX，在流表溢出脆弱性攻击相关测试中实现网络控制平面的功能；PC_3 配有 Intel Xeon 1235 v3 处理器与 8 GB 内存，安装 Windows Server 2008 操作系统，用于运行 VMware。

3.2　防御能力测试

图 2 给出了经典的基于基数树的流表聚合算法防御实验效果。实验中利用 SDN 流表溢出攻击方法[6]，向目标 SDN 交换机注入攻击流量，记录最终使得目标 SDN 交换机发生流表溢出时注入的数据包总数。实验共重复进行 10 次，目标 SDN 交换机的流表容量设定为 2 000 条。由图 2 可以看出，由于使用基于基数树节点聚合的流表聚合算法，大量攻击流量产生的流表项被聚合，平均流表聚合率为 35.9%。为了使流表容量为 2 000 条的 SDN 交换机流表溢出，平均需要发送的数据包数为 3 173 个，即攻击者平均需要发送 3 173 个攻击数据包，才能使得流表容量为 2 000 条的 SDN 交换机发生流表溢出。实验结果验证了基于基数树的流表聚合算法在防御流表溢出脆弱性攻击方面具有一定有效性。

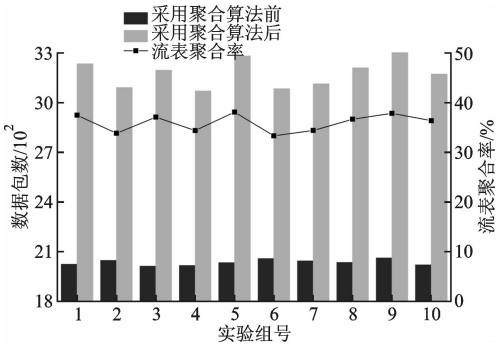

图 2　基于基数树节点聚合的防御效果图

图 3 给出了本文提出的基于装箱优化聚合的防御效果实验结果。实验方法与基于基数树节点聚合的防御效果实验相同。由于本文方法针对 SDN 网络中的路由聚合问题进行了针对性的改进，使得经过装箱优化的调整之后，可聚合的流表项数增加，平均流表聚合率达到 54.9%。通过与图 2 所示结果对比发现，与经典的基于基数树节点聚合的流表聚合算法相比，本文算法可更有效地防御针对 SDN 交换机的流表溢出攻击。采用本文方法后，为达成流表溢出攻击所需的攻击数据包数平均为 4 601 个，比不采用该防御方法的攻击数据包数增加了 125.8%，从而增加了攻击难度，提升了 SDN 网络对该类攻击的防御能力。

为对本文所提方法的防御效果做更深入的分析，分别采用流实体生存周期、频繁主机动态更新和流实体回收算法 3 种常见的流表维护算法进行防御效果的横向对比分析，实验方法与图 2、图 3 所示 2 项防御实验相同，结果如图 4～图 6 所示。在基于

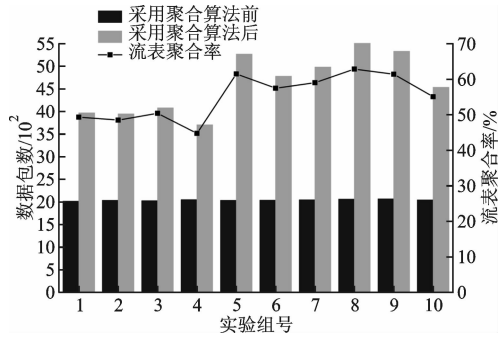

图 3　基于装箱优化聚合的防御效果图

流实体生存周期的防御实验中，根据网络管理中的实际情况，分别设置生存周期为 10、30 和 60 s，达成攻击平均所需数据包数为 2 036 个，具体如图 4 所示。在基于频繁主机动态更新的防御实验中，分别设置更新周期为 10、30 和 60 s，达成攻击平均所需数据包数为 2 042 个，具体如图 5 所示。在基于流实体回收算法的防御实验中，分别采用了 FIFO 算法(first in first out)、LFU 算法(least frequently used)和 LRU 算法(least recently used)3 类常用的回收算法，达成攻击平均所需数据包数为 2 033 个，如图 6 所示。通过对比可知，所提方法的防御效果均明显优于上述 3 类方法。这些常见流表维护方法的设计主要针对正常网络中的流表使用场景，在面

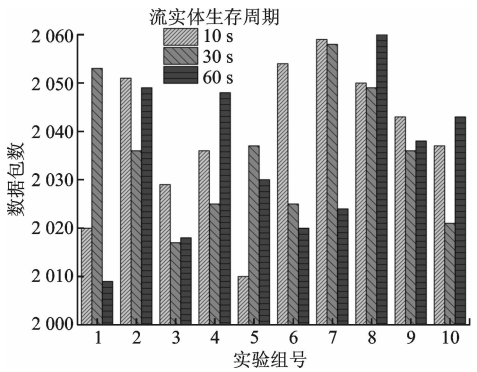

图 4　基于流实体生存周期的防御效果图

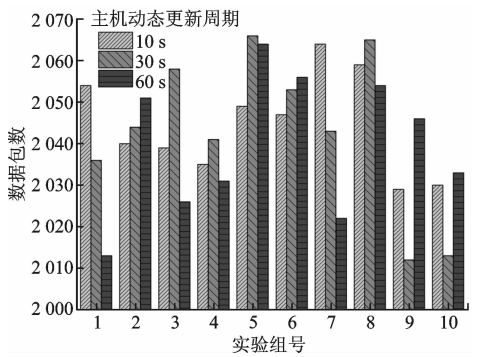

图 5　基于频繁主机动态更新的防御效果图

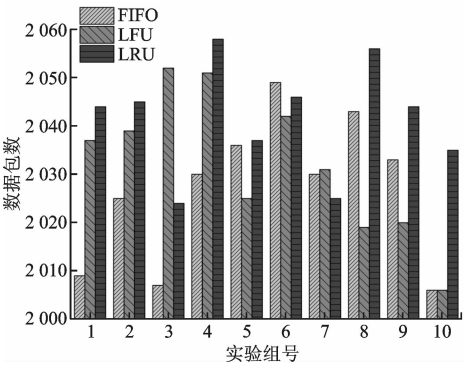

图 6 基于流实体回收算法的防御效果图

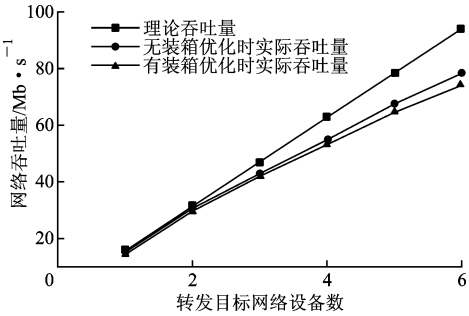

图 7 网络吞吐量影响分析图

对恶意构造的短时大规模攻击流量时，防御效果较弱，难以抵御 SDN 网络中的流表溢出攻击。

由于采用路由聚合算法会增加交换机和控制器的资源开销，因此本文进一步分析引入路由聚合算法是否会对交换机的吞吐量产生较大影响。图 7 给出了采用 2 种路由聚合算法和不采用聚合算法的网络吞吐量对比实验结果。首先测量了只有单台转发网络设备时的网络吞吐量，然后逐渐增加设备数量，从 1 台增加至 6 台，分别测量采用基于基数树的路由聚合算法、采用基于装箱优化的路由聚合算法和不采用路由聚合算法 3 种情况下的网络吞吐量。实验结果表明，采用基于装箱优化的路由聚合算法，仅使得网络吞吐量发生少量下降，且与基于基数树的路由聚合算法相差很小，从而验证了本文所提路由聚合算法对网络性能不会产生显著影响。

4 结 论

本文分析了当前 SDN 网络交换机流表溢出的脆弱性问题的产生原因和攻击后果，并提出了基于路由聚合思路的防御方法。在防御方法中，提出了基于装箱优化的流表聚合算法，充分利用了 SDN 的网络特性，在传统路由聚合算法的基础上，通过装箱优化再次聚合流表。实验结果表明，所提方法可有效提高 SDN 网络对流表溢出脆弱性攻击的防御能力。在网络应用数量较多且拓扑较为复杂的网络环境中，实现本文所提方法较为困难，因此在下一步研究中将研究约束条件更为宽松的路由聚合方法，以提出更便利的 SDN 交换机流表溢出防御方法。

参考文献：

[1] ELLIOTT C. Geni：exploring networks of the future [EB/OL]. (2012-08-24)[2016-09-10]. http：//www.ieice.org/～nv/2nd-chip.pdf.

[2] 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.

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

[4] BENSON T, AKELLA A, MALTZ A. Network traffic characteristics of data centers in the wild [C]// Proceedings of the 10th ACM Sigcomm Conference on Internet Measurement. New York, USA：ACM, 2010：267-280.

[5] ZHANG P, WANG H, HU C, et al. On denial of service attacks in software defined networks [J]. IEEE Network, 2016, 30(6)：28-33.

[6] LENG J, ZHOU Y, ZHANG J, et al. An inference attack model for flow table capacity and usage：exploiting the vulnerability of flow table overflow in software-defined network [EB/OL]. (2015-04-13)[2016-07-16]. https：//arxiv.org/abs/1504.03095.

[7] UZMI Z, NEBEL M, TARIQ A, et al. SMALTA：practical and near-optimal FIB aggregation [C]// Proceedings of the 7th Conference on Emerging Networking Experiments and Technologies. New York, USA：ACM, 2011：1-12.

[8] LI Q, XU M, CHEN M. NSFIB construction & aggregation with next hop of strict partial order [C]// Proceedings of the 2013 IEEE INFOCOM. Piscataway, NJ, USA：IEEE, 2013：550-554.

[9] ZHAO X, LIU Y, WANG L, et al. On the aggregatability of router forwarding tables [C]// Proceedings of the 2010 IEEE INFOCOM. Piscataway, NJ, USA：IEEE, 2010：1-9.

[10] PAPADIMITRIOU C, STEIGLITZ K. Combinatorial optimization：algorithms and complexity [M]. Englewood cliff, NJ, USA：Prentice-Hall, 1982：220-251.

（编辑 武红江）