基于 OpenFlow 交换机端口混淆的移动目标防御机制

张昭俊,韩 俐
(天津理工大学 计算机科学与工程学院,天津 300384)

摘 要:软件定义网络中数据平面节点非法接入会导致拒绝服务攻击或实施中间人攻击进行信息的窃取和篡改。为防范 OpenFlow 伪交换机对网络服务造成危害,提出了一种利用网络中数据报文的包头信息,动态混淆 OpenFlow 交换机端口的移动目标防御机制。交换机根据控制器流规则的下发次数进行动态端口混淆,并且根据端口生存时间以当前处理的数据报文包头中的端口和 IP 地址等信息作为下一轮端口混淆的种子信息,保证混淆端口具有充分的随机性,有效提高网络的动态性。还提出了利用控制器和交换机间的通信消息 flow_mod 实现控制器与交换机之间的端口混淆同步,不仅保证双方数据转发端口的一致性,而且省却了双方在同步方面的开销。经过仿真实验验证,伪交换机无法正确解析出包含混淆端口号的流表项,从而有效阻止伪交换机发动网络攻击。

关键词:软件定义网络;数据平面;移动目标防御;OpenFlow;端口混淆
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2020)12-0106-06
doi:10.3969/j.issn.1673-629X.2020.12.019

Moving Target Defense Mechanism Based on OpenFlow Switch Port Obfuscation

ZHANG Zhao-jun, HAN Li
(School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300384, China)

Abstract: Illegal access to data plane nodes in software defined network can lead to Denial of Service attack or Man-in-the-Middle attack which causes theft and tampering. In order to prevent the OpenFlow unauthorized switch from accessing the network and causing harm to network services, we propose a moving target defense mechanism, which uses the header information of packets in the network to dynamically obfuscate ports of OpenFlow switch. The switch dynamically obfuscates port number based on times flow rules are delivered by controller, and uses the information such as the port and IP address in the currently processed packet header as the seed information for the next round of port obfuscation based on the port lifetime to ensure sufficient randomness of obfuscated port number and improve network dynamics. We also propose to implement port obfuscation synchronization between the controller and the switch by using the communication message flow_mod message, which not only ensures the consistency of the data forwarding ports of both parties, but also saves the cost of synchronization. Simulation results show that unauthorized switch cannot correctly parse flow entries with obfuscated port number, which can effectively prevent unauthorized switch from launching network attacks.

Key words: software defined network; data plane; moving target defense; OpenFlow; port obfuscation

0 引 言

近年来,新型网络威胁呈现出全球蔓延态势,新型攻击更隐蔽、手段更先进、持续性更强,要求网络安防系统的响应时间更短。传统防护手段,如防火墙、安全网关、安全软件等静态被动式防御手段,具有对威胁感知天然的滞后性和局限性,只有当网络攻击已经或正在发生才能采取防范行动,难以应对大规模有组织、有意图的突发性攻击。当前网络环境中,安全防御所需的花费与攻击者攻破网络的努力严重不对等。防御者需要花费大量的资源为系统添加各种安全防御手段,而攻击者只需找到系统中存在的某一个漏洞,就能发动有效的攻击,使得防御者始终处于不利地位。为应

收稿日期:2019-12-30 **修回日期**:2020-04-30
基金项目:国家自然科学基金项目(61702366,61802281);天津市自然科学基金项目(18JCZDJC30700,16JCYBJC41500,19JCYBJC15800)
作者简介:张昭俊(1994-),男,硕士研究生,CCF 会员(97882G),研究方向为软件定义网络、网络空间安全;通信作者:韩 俐(1983-),女,副教授,博士,CCF 会员(63585M),研究方向为软件定义网络、网络空间安全、下一代网络安全。

对传统网络安全防护的被动局面,移动目标防御(moving target defense,MTD)[1]作为一种颠覆性的安全防御理念,尝试改变游戏规则的主动防御机制被提出。MTD 的思想是通过动态改变基础设施属性,持续地改变攻击面,使网络配置具备随机性和动态性。MTD 包含多种实现方式,其中包括对软件多样性的实现[2]、底层指令的变化[3]和网络层的移动目标防御[4]等,能自动地改变一个或多个系统属性,从而使系统攻击面对攻击者是不可预测的。攻击者必须调用大量的资源,分析、探测不断变化的攻击面来获取有用信息,且探测和分析难度随着时间的推移而增大。这无疑增加了攻击者对网络信息获取的复杂度,增加攻击者的攻击难度和代价,打破了现有攻防双方信息的不对称性,使攻防天平向防御方倾斜,让攻击者和防御者处于同等地位,有效提升网络空间安全性。

软件定义网络(software defined network,SDN)[5]是斯坦福大学 Clean slate 实验室提出的一种新型的基于软件定义的网络架构及技术。SDN 的设计理念是将网络的控制平面和数据转发平面进行分离,逻辑上的集中控制层面具备网络全局视角,进行资源的全局调配和优化,提升网络控制的便捷性。SDN 重构传统网络架构及安全体系,可以实现高效网络管控和资源调度,其控制与转发平面分离、集中控制、开放可编程、流表转发等特性,有利于提升安全防护灵活性、智能性和协同性,推动网络能力便捷调用,支持网络业务创新,给网络安全领域的发展带来了新机遇。

SDN 架构虽然有很多优点,但其所面临的安全性问题成为限制 SDN 广泛应用的关键因素,数据平面的网络设备只是简单的转发元素,容易引发诸多安全问题[6-7]。在 OpenFlow 协议[8]中,转发设备需要在控制平面下发流规则前进行数据缓存,因此容易受到存储器饱和攻击。在遭受 DoS 攻击时,流表存储空间迅速耗尽,后续数据流被丢弃,导致拒绝服务。对 OpenFlow 交换机的窃听和篡改攻击,非法篡改控制器下发的流规则,导致流规则不一致,丧失可靠性,使网络不稳定。伪交换机长期存在网络中,可以窃听用户通信数据,截获用户口令,错误或拒绝执数据流处理规则,造成拒绝服务攻击。

端口跳变是一种典型的移动目标防御技术[9],传统的端口跳变技术往往是通过算法不断地改变用户数据报协议(user datagram protocol,UDP)/传输控制协议(transmission control protocol,TCP)的端口号,通信双方不断地通过新端口建立连接,使攻击者无法探测到网络通信双方实时有效的通信端口,导致攻击失效。但是这种防御方式只能应对针对终端主机的攻击,对交换机的攻击(如非法交换机的接入)不具备任何防御能力。

该文将 MTD 和 SDN 技术优势有机结合,提出基于 OpenFlow 交换机端口动态混淆的移动目标防御机制,增强网络设备配置参数的随机性和动态性,实现数据在 SDN 网络架构下的通信安全,增加攻击者的实施难度,降低攻击成功率。

1　相关工作

针对计算机网络信息面临巨大的安全威胁,端口跳变技术应运而生。Jajodia 等[10]定义了移动目标防御中攻击面的相关概念,并提出了空间随机化、指令集随机化、数据随机化等多种攻击面移动方法。Hong 等[11]总结了 3 种移动目标防御策略,包括混乱化、多样化和冗余化。Kampanakis 等[12]提出了 SDN 架构下针对网络映射和侦察的解决方法。该方法通过对恶意数据报文进行检测,并对这些恶意数据报文进行随机化处理操作。但是该方法建立在有效识别出恶意数据报文的基础之上,如果没有识别出攻击者的恶意数据报文,那么该方法将不会产生任何有效的防御效果。G. Badishi 等[13]为缓解 DoS 攻击,提出了随机端口跳变(RPH)算法,其主要思想是通信双方之间存在两条通信链路,一条用于双方的数据通信,另一条用于传递通知进行端口跳变的 ACK 消息,此方法中用于跳变同步的 ACK 消息暴露在通信链路中,容易被攻击者截获和篡改,导致跳变不同步,造成通信中断。Kousaburou Hari[14]通过分析 ACK 消息丢失或被截获等不同情况,提出了改进的 RHP 算法,在通信双方部署两套随机端口跳变策略,其中一套策略遭受攻击后,通信双方仍能通过备用端口进行通信。范晓诗等[15]提出了可变时隙端口跳变算法,通过在正常传输条件下延长端口传输时隙,减少端口跳变造成的通信开销,当发现可疑入侵行为时则加快跳变的频率,达到抵抗攻击的效果。Lee 等[9]为了缓解 DoS/DDoS 攻击所带来的危害,提出了一种端口跳变技术,其主要思想是服务器和交换机之间共享一个秘密,保证合法数据流的服务质量水平在可接受范围内的前提下,服务器通信所使用的 TCP/UDP 端口号随时间和共享秘密的变化而变化。唐秀存等[16]将移动目标防御中的端口跳变技术应用到 SDN 网络中的 DoS 攻击防御中,将服务端的端口跳变模块前移到 SDN 控制器,SDN 控制器作为 DoS 攻击的过滤网关,检查网络连接是否符合当前时隙开放端口,控制数据流进入网络,抵御 DoS 攻击。胡毅勋等[17]提出了基于 OpenFlow 的网络层移动目标防御方案,其主要思想是域内通信的网络流量在所经链路上的每一跳交换机都进行源和目的 IP 地址的改变,实现网络地址的逻辑移动。域间通信流量使用端口跳

变,实现跨域通信节点的隐藏。

基于上述事实,SDN 与 MTD 相结合能够有效地防御外部攻击者对网络的攻击和破坏,但是对于 SDN 内部数据转发层的核心设备——OpenFlow 交换机的安全防御却相对较少,SDN 内部 OpenFlow 交换机的安全问题日益突出。该文提出的基于 OpenFlow 交换机端口混淆的移动目标防御机制,能有效防范伪交换机接入网络造成的破坏。

2 端口混淆防御机制

2.1 机制设计

控制器与 OpenFlow 交换机初次连接时进行端口变换值和端口变换值生存时间的初始化。控制器和交换机建立连接时,双方在握手过程中通过系统预设的值对端口变换值及端口变换值生存时间进行初始化,同时将真实端口号和变换后的端口号及其生存时间存储在端口变换表中,端口变换表结构如表 1 所示,以保证控制器和交换机进行端口混淆的同步性和数据转发端口的一致性。

表 1 端口变换表结构

交换机 ID	真实端口号	端口变换值	端口变换值生存时间
dpid	port	tport	pst

端口变换表主要包含 4 部分:交换机 ID(dpid)、真实端口号(port)、端口变换值(tport)和端口变换值生存时间(pst)。交换机真实端口号是交换机创建时系统默认端口号,端口变换值是通过端口混淆算法生成的混淆端口值,用于在流表项中显示的 output 端口号。端口变换值生存时间是当前生成的端口变换值的存在时间,决定是否对当前端口变换值进行新一轮混淆。

控制器每次处理 Packet_in 消息进行规则生成时,根据正确的数据转发端口在端口变换表中查找对应的端口变换值,随后生成相应的流规则,下发到交换机。同时,相应端口号的生存时间衰减。OpenFlow 交换机端接收到控制器下发的流规则后,根据流规则中的变换端口号在端口变换表查找到对应的数据转发端口,并执行流规则进行数据转发处理,同时将端口的生存时间衰减。当控制器和交换机端口的生存时间为零,则依据刚处理的数据报文中的相应信息,调用端口混淆算法生成新的端口变换值并存储在端口变换表中,端口变换值的生存时间也一并更新。随后对端口混淆前相应流规则进行删除,确保数据流能根据混淆后的端口信息进行正确转发,保证网络通信正常。控制器端和交换机端处理流程如下所示。

控制器端处理流程:

(1)接收到 Packet_in 消息。

(2)查找端口变换表,根据正确转发端口确定变换端口。

(3)端口生存时间衰减,判断其是否为零,不为零跳转到步骤(7)。

(4)提取数据流信息,流规则生成并设置超时时间,通过 flow_mod 消息下发到交换机。

(5)调用 2.3 节端口混淆算法生成新的端口变换值。

(6)更新端口变换表,跳转到步骤(8)。

(7)流规则生成并通过 flow_mod 消息下发到交换机。

(8)结束。

交换机端处理流程:

(1)接收到控制器下发的 flow_mod 消息。

(2)端口变换表中查找相应的端口号。

(3)根据查找到的端口号执行相应的流规则,端口生存时间衰减。

(4)判断端口生存时间是否为零,不为零则跳转到步骤(9)。

(5)提取数据流缓存信息,调用 2.3 节端口混淆算法生成新的端口变换值。

(6)更新端口变换表。

(7)删除过期流表项。

(8)流规则超时删除对应流表项。

(9)结束。

2.2 基于 flow_mod 消息的同步方式

端口混淆对控制器和交换机之间混淆同步性要求较高,控制器和交换机之间必须使用相同的混淆算法和混淆规律,以及在进行混淆时使用的种子信息,从而保证网络通信正常。否则很容易造成控制器下发的流规则失效,网络通信中断。

传统的端口跳变技术中主要有严格的时间同步机制[9]、ACK 报文同步机制[12]和使用基于时间戳的同步方式[18]。严格的时间同步机制在网络拥塞状态和高时延状态下将失去作用,导致跳变不同步,网络通信异常。ACK 报文同步机制将同步信息封装在 ACK 报文中,ACK 报文易丢失或被攻击者截获。Zhang 等[19]提出了基于端口跳变的 SDN 环境中 DoS 缓解机制(PHS-DM),利用时间戳反馈机制实现端口的同步跳变,但此方法部署难度大,实现复杂。

该文提出的端口混淆机制中控制器和交换机之间的同步既没有使用严格的时间同步机制,也没使用 ACK 报文同步机制,而是采用控制器下发给交换机的流规则消息 flow_mod 消息的数量实现同步问题。

在控制器与交换机在建立连接过程时,双方通过

系统内置的默认值对相应的端口变换值生存时间进行初始化。控制器进行流规则生成时,根据数据转发时交换机的真实端口号查询端口变换表中的变换端口号,随后将此端口变换值生存时间减一,并检查是否为零,如果为零,提取数据包中相关信息,将生成的含有变换端口号的流规则通过 flow_mod 消息下发到对应的交换机,调用端口混淆算法,生成新的端口变换值,更新端口变换表中的端口变换值及其生存时间。交换机接收到控制器下发的包含流规则的 flow_mod 消息后,根据流规则中的端口号查找端口变换表,得到真实的端口号,并根据端口号进行数据转发,随后将端口的生存时间减一,判断是否为零,如果为零则根据交换机内缓存的数据流信息提取端口混淆算法中的种子信息,进行端口混淆,更新端口变换表中的端口变换值和生存时间。

该文提出的利用交换机和控制器之间的 flow_mod 消息实现双方端口混淆同步方法,省却了双方的同步开销。通信信道内只有控制器与交换机间的控制消息,没有其他的数据流干扰,有效降低了 flow_mod 消息的丢失率。此外,部署与实施更加方便灵活。

2.3　端口混淆算法

控制器和交换机根据进入数据流信息生成端口变换值 $Port_{update}$。$Port_{update}=f(IP_{seed},Port_{old},Port_{seed})$,其中 f 是伪随机数生成算法。$Port_{seed}$ 为以数据报文中源、目的端口号等信息所生成的端口种子,IP_{seed} 为从数据报文中提取出的源、目的 IP 地址等信息所生成的地址种子。其流程图如图 1 所示。

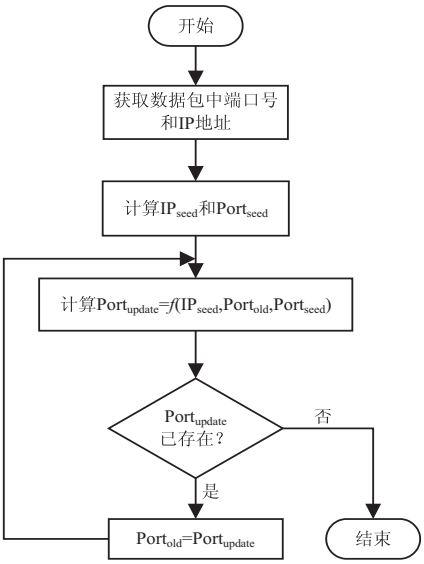

图 1　端口混淆算法流程

对于数据报文中源 IP 地址为 $IP_{src}=192.168.1.X$,目的 IP 地址为 $IP_{dst}=192.168.2.Y$ 的端口变换过程如下:

(1)计算端口种子 $Port_{seed}$,如式(1)所示,其中 $Port_{old}$ 为上一次端口变换后的通信端口,$Port_{src}$ 为此次端口变换的数据报文中的源端口号,$Port_{dst}$ 为此次端口变换的数据报文中的目的端口号。

$$Port_{seed}=Port_{old}^{(Port_{src}+Port_{dst})} \bmod 65536 \tag{1}$$

传统的端口跳变技术中 65 536 个可能端口中有 1 024个标准端口被占用[20],而交换机端口混淆中不存在这种现象,在端口混淆中的可用端口个数为 65 536个。

(2)计算 IP 地址种子 IP_{seed},如式(2)所示。

$$IP_{seed}=(X+Y)^{(Port_{src}+Port_{dst})} \bmod 256 \tag{2}$$

(3)根据以上计算的端口种子 $Port_{seed}$ 和 IP 种子 IP_{seed} 计算得出端口变换值 $Port_{update}$,如式(3)所示。

$$Port_{update}=(IP_{seed}\times Port_{old}+Port_{seed}) \bmod 65536 \tag{3}$$

(4)端口变换值的检测。检测新生成的端口变换值是否已经被其他策略使用,如已被使用,则将此端口号作为 $Port_{old}$,重新调用端口混淆算法进行新的端口号的生成,直到不再重复为止。

端口混淆算法充分利用网络中通信双方的数据报文中源、目的 IP 地址和端口号等信息作为端口混淆的种子信息,交换机每次进行端口混淆时种子信息是不确定的,混淆后端口是随机的,网络配置信息动态化、随机化变化。

2.4　防御能力分析

伪交换机非法接入网络,控制器向交换机下发流表项时,流表项中包含的动作端口号是经过混淆算法计算出来的,伪交换机中不存在端口变换表和端口混淆算法,无法解析出正确的端口进行数据转发,攻击者必须在端口生存时间内计算出正确的端口号并进行转发才能长期潜伏在网络中造成更大的破坏。假设攻击者的计算能力为 S_c,n 为端口混淆时所有可用端口数量,端口生存时间为 t,待处理数据流数为 m,攻击者计算出真实端口的成功率 R,则:

$$R=\frac{S_c\times t}{n\times m} \tag{4}$$

从式(4)中可以看出,端口的生存时间越短,当前网络中待处理的数据流越多,攻击者计算出真实端口号的成功率越低。

假设攻击者分析第 i 个端口时的计算开销为 S_{cal_i},单个端口的计算开销为 S_{cal},如果端口混淆算法计算所生成的端口号与交换机的真实端口号对应是顺序递增的,则攻击者的计算开销最小为:

$$S_{min}=\sum_{i=1}^{n}S_{cal_i}=nS_{cal}=O(n) \tag{5}$$

如果通过端口混淆算法生成的端口号对应的交换机的真实端口号是逆序的,则此时攻击者的计算开销

最大为：

$$S_{\max} = \sum_{i=1}^{n} iS_{\mathrm{cal}_i} = O(n^2) \tag{6}$$

综合式(5)、式(6)可知，攻击者的总体开销 S_{sum} 为 $O(n) \leqslant S_{\mathrm{sum}} \leqslant O(n^2)$ 。

从系统安全性考虑，可以适当减小端口变换值的生存时间，加快端口变换速率，增强系统防御能力。此外，交换机动态进行端口混淆，在不同的时间内其转发端口也是不同的，攻击者很难确定下次数据转发的端口号，攻击者需要不停地计算转发端口，消耗大量资源，降低了攻击成功率。

3 仿真实验

3.1 算法性能测试

使用 MATLAB Performance Test 测试框架对 2.3 节提出的端口混淆算法运行时间性能进行测试，解析基于类的测试得到的原始数据，对每一次算法运行时间绘制折线图，如图 2 所示。

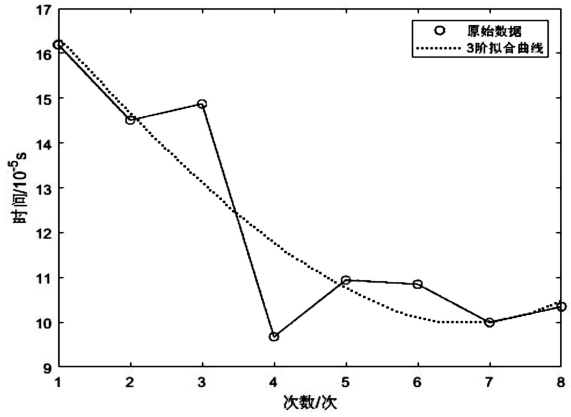

图 2 算法运行时间测试

MATLAB 通过测量数据集是否达到统计目标决定运行次数，从图 2 的结果中可以看出，一共对算法进行了 8 次性能测试。其中前 4 次测试结果是预备代码[21]，后 4 次测试结果是对代码的正式测试。对测试结果进行 3 阶拟合，可以明显看出正式测试与预备代码在运行时间上有着不同的分布，预备代码测试时间明显高于正式测试，随着预备代码，算法运行时间变短，正式测试代码运行时间趋于平缓。为避免代码在运行过程中的编译和优化对代码测试性能造成影响，剔除预备代码测试结果，对正式测试结果求均值得到统计意义上的结果，如图 3 所示。可以看出代码运行时间极短，端口混淆不会增加过多的数据处理时延，能有效满足当前数据传输的需要。

Name	GroupCount	mean_MeasuredTime
TestportupdateTimeCost/testTime	4	0.00010532

图 3 算法运行平均时间

3.2 防御机制有效性测试

为了测试端口混淆主动防御机制的有效性，该文使用 Mininet 网络模拟器[22]模拟并搭建简易 SDN 网络测试网络环境，采用 Ryu[23]作为 SDN 控制器。网络拓扑为两台主机 h1、h2 连接到同一台 OpenFlow 交换机，主机 h1 和 h2 进行通信。通过 Mininet 模拟器模拟实验网络环境，交换机初始化时默认使用的端口号为 1 和 2，主机 h1 与 h2 通信，主机间能正常通信，查看交换机中流表，其结果如图 4 所示。

```
root@jun-virtual-machine:~# ovs-ofctl dump-flows s1
 cookie=0x0, duration=9.623s, table=0, n_packets=13, n_bytes=1046, priority=0 ac
tions=CONTROLLER:65535
root@jun-virtual-machine:~# ovs-ofctl dump-flows s1
 cookie=0x0, duration=7.057s, table=0, n_packets=10, n_bytes=868, priority=1,in_
port="s1-eth2",dl_src=da:34:30:d5:0c:0e,dl_dst=72:ac:0b:20:bb:6c actions=output:
1024
 cookie=0x0, duration=6.840s, table=0, n_packets=7, n_bytes=630, priority=1,in_p
ort="s1-eth1",dl_src=72:ac:0b:20:bb:6c,dl_dst=da:34:30:d5:0c:0e actions=output:9
25
 cookie=0x0, duration=23.684s, table=0, n_packets=19, n_bytes=1438, priority=0 a
ctions=CONTROLLER:65535
```

图 4 交换机流表项

从图 4 中可以看出交换机 s1 中存在 3 条流表项，最后一条流表项为漏表项，用于将没有匹配到流表项的数据报文通过 Packet_in 消息上传到控制器，由控制器决定转发策略。前两条流表项为处理主机 h1 与 h2 通信的流规则，从中可以看出两条流表项中动作 output 的端口号均不是 1 或 2，而是交换机中并不存在的端口号，两条流表项均正确匹配并转发数据报文，两主机正常通信。

伪交换机加入网络后，主机 h1 和 h2 通过伪交换机进行通信，图 5 显示的是伪交换机中的流表项。伪交换机中只有两条流表项，第二条为漏表项，第一条是针对主机 h2 发往主机 h1 的应答包的流处理规则，伪交换机不能正确解析出 actions 中的端口号，不能进行正确的转发，从而防止伪交换机对网络的破坏。

```
root@jun-virtual-machine:~# ovs-ofctl dump-flows s1
 cookie=0x0, duration=72.177s, table=0, n_packets=17, n_bytes=714, priority=1,in
_port="s1-eth2",dl_src=8e:10:71:19:6b:8c,dl_dst=4e:7b:bc:de:48:50 actions=output
:11024
 cookie=0x0, duration=77.225s, table=0, n_packets=33, n_bytes=1914, priority=0 a
ctions=CONTROLLER:65535
```

图 5 伪交换机流表项

从以上实验结果可以看出，没有端口混淆防御机制的伪交换机不能正确解析出控制器下发的含有混淆端口号的流规则，无法根据流表项中的端口号获取网络拓扑，即使攻击者构造出流表项将所有通过伪交换机的流量发往某特定一链路或主机进行 DOS 攻击，由于无法获取网络拓扑相关信息不能进行有效的网络攻击。而部署端口混淆防御机制的交换机能正确解析出流表项中的端口号进行数据转发，网络通信正常。

4 结束语

针对软件定义网络中所面临的伪交换机安全问

题,提出了通过动态混淆交换机端口号的方法实现移动目标防御的机制,该机制充分利用数据报文内 IP 地址和端口号信息作为端口混淆种子信息,使交换机进行端口混淆时具有充分的随机性。交换机和控制器间通过流规则下发次数实现端口混淆同步,不需要严格的时间同步,也不需要发送额外的同步报文,进一步增强了防御机制的安全性,增加了攻击者获取网络信息的难度,能有效地防范非法交换机对网络的攻击。

参考文献:

[1] ZKIK K,SEBBAR A,BADDI Y,et al. Secure multipath mutation SMPM in moving target defense based on SDN[C]// International symposium on machine learning and big data analytics for cybersecurity and privacy. Leuven, Belgium: ELSEVER,2019:977-984.

[2] VIKRAM S,YANG C,GU G. Nomad:towards non-intrusive moving-target defense against web bots[C]//IEEE conference on communications and network security (CNS). National Harbor,MD,USA:IEEE,2013:55-63.

[3] LUCAS B,FULP E W,JOHN D J,et al. An initial framework for evolving computer configurations as a moving target defense[C]//The 9th annual cyber and information security research conference. Oka Ridge,TN,USA:ACM,2014:69-72.

[4] APPLEGATE S D. The principle of maneuver in cyber operations[C]//4th international conference on cyber conflict (CYCON 2012). Tallinn,Estonia:IEEE,2012:1-13.

[5] 左青云,陈　鸣,赵广松,等. 基于 OpenFlow 的 SDN 技术[J]. 软件学报,2013,24(5):1087-1097.

[6] 郭中孚,张兴明,赵　博,等. 软件定义网络数据平面安全综述[J]. 网络与信息安全学报,2018,4(11):1-12.

[7] 王　月,吕光宏,曹　勇. 软件定义网络安全研究[J]. 计算机技术与发展,2018,28(4):128-132.

[8] 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.

[9] LEE H C J,THING V L L. Port hopping for resilient networks[C]//60th IEEE vehicular technology conference. Los Angeles,CA,USA:IEEE,2004:3291-3295.

[10] AJODIA S,GHOSH A K,SWARUP V,et al. Moving target defense:creating asymmetric uncertainty for cyber threats[M]. New York:Springer Science & Business Media,2011.

[11] HONG J B,KIM D S. Assessing the effectiveness of moving target defense using security models[J]. IEEE Transactions on Dependable and Secure Computing,2016,13(2):163-177.

[12] KAMPANAKIS P,PERROS H,BEYENE T. SDN-based solutions for moving target defense network protection[C]// Proceeding of IEEE international symposium on a world of wireless, mobile and multimedia networks. Sydney, NSW, Australia:IEEE,2014:1-6.

[13] BADISHI G,HERZBERG A,KEIDAR I. Keeping denial of service attackers in the dark[J]. IEEE Transactions on Dependable and Secure Computing,2007,4(3):191-204.

[14] HARI K, DOHI T. Sensitivity analysis of random port hopping[C]//7th international conference on ubiquitous intelligence & computing and 7th international conference on autonomic & trusted computing. Xi´an, Shaanxi, China: IEEE,2010:316-321.

[15] 范晓诗,李成海,王　昊. 基于可变时隙与动态同步的端口跳变技术研究[J]. 计算机工程与设计,2013,34(10):3465-3469.

[16] 唐秀存,张连成,史晓敏,等. 基于端口跳变的 SDN 网络防御技术[J]. 计算机应用研究,2016,33(10):3083-3087.

[17] 胡毅勋,郑康锋,杨义先,等. 基于 OpenFlow 的网络层移动目标防御方案[J]. 通信学报,2017,38(10):102-112.

[18] LIN K,JIA C,WENG C. Distributed timestamp synchronization for end hopping[J]. China Communications, 2011, 8(4):164-169.

[19] ZHANG L,GUO Y,YUWEN H,et al. A port popping based dos mitigation scheme in SDN network[C]//12th international conference on computational intelligence and security. Wuxi,China:IEEE,2017:314-317.

[20] 赵子郁,郭渊博,刘　伟. 软件定义网络中基于加密的端口跳变技术研究[J]. 计算机应用与软件,2017,34(4):322-328.

[21] 王　建,赵国生. Matlab 数学建模与仿真[M]. 北京:清华大学出版社,2016.

[22] DE OLIVEIRA R L S,SCHWEITZER C M,SHINODA A A,et al. Using mininet for emulation and prototyping software-defined networks[C]//IEEE colombian conference on communications and computing. Bogota, Colombia: IEEE, 2014:1-6.

[23] 张朝昆,崔　勇,唐嵩祎,等. 软件定义网络(SDN)研究进展[J]. 软件学报,2015,26(1):62-81.