

高效安全的软件定义网络拓扑发现协议

李冬¹, 于俊清^{1,2}, 谷永普², 赵鹏程²

(1. 华中科技大学网络与计算中心, 湖北 武汉 430074; 2. 华中科技大学网络空间安全学院, 湖北 武汉 430074)

摘要: 基于 OpenFlow 的软件定义网络 (SDN, software defined network) 控制器主要采用 OFDP (OpenFlow discovery protocol) 发现网络拓扑, 现有研究表明, OFDP 存在网络拓扑信息更新效率低、容易遭受网络拓扑污染攻击等问题, 为了提高网络拓扑发现协议的效率和安全性, 对 OFDP 的网络拓扑发现机理和安全问题进行了深入研究, 详细分析了软件定义网络拓扑建立和更新阶段的特点, 基于图论的最小顶点覆盖问题提出了一种改进的 OpenFlow 网络拓扑发现协议——Im-OFDP (improved OpenFlow discovery protocol)。Im-OFDP 基于 OFDP 网络拓扑发现的先验信息构建端口信息表和链路信息表, 然后建立网络拓扑图模型, 基于最小顶点覆盖算法筛选支撑网络拓扑的交换机, 再根据网络拓扑结构设计网络拓扑发现的多级流表, 由控制器下发至相应交换机。控制器发出的网络拓扑发现报文经多级流表转发处理后上报给控制器, 进而获取网络拓扑信息。针对安全问题, Im-OFDP 一方面基于拓扑发现获取的信息在 LLDP (link layer discovery protocol) 报文中采用动态检验码检测链路的真实性, 另一方面基于主机和交换机等网络设备的拓扑信息建立验证机制, 确保网络设备可信。实验结果表明, 部署 Im-OFDP 后, 控制器在网络拓扑发现的消息数量、带宽开销、CPU 资源负载显著降低, 节点失效响应时间、节点失效后链路恢复时间明显较短, 能够防御链路伪造、交换机伪造等多种形式的网络拓扑污染攻击。Im-OFDP 能够显著提高 SDN 拓扑发现的效率和安全性。

关键词: 软件定义网络; 网络拓扑; 网络安全

中图分类号: TP309

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023080

Efficient and safe software defined network topology discovery protocol

LI Dong¹, YU Junqing^{1,2}, GU Yongpu², ZHAO Pengcheng²

1. Network and Computation Center, Huazhong University of Science and Technology, Wuhan 430074, China

2. School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract: The network topology discovery in OpenFlow-based software-defined networks is mainly achieved by utilizing the OpenFlow discovery protocol (OFDP). However, it has been observed in existing research that OFDP

收稿日期: 2022-12-29; 修回日期: 2023-03-22

通信作者: 于俊清, yjqing@hust.edu.cn

基金项目: 国家重点研发计划 (2020YFB1805601); 中国高校产学研创新基金 (2021FNA02005)

Foundation Items: The National Key R&D Program of China (2020YFB1805601), China University IUR Innovation Fund (2021FNA02005)

引用格式: 李冬, 于俊清, 谷永普, 等. 高效安全的软件定义网络拓扑发现协议[J]. 网络与信息安全学报, 2023, 9(6): 20-33.

Citation Format: LI D, YU J Q, GU Y P, et al. Efficient and safe software defined network topology discovery protocol[J]. Chinese Journal of Network and Information Security, 2023, 9(6): 20-33.

exhibits low updating efficiency and is susceptible to network topology pollution attacks. To address the efficiency and safety concerns of the network topology discovery protocol, an in-depth investigation was conducted on the mechanism and safety of OFDP network topology discovery. The characteristics of network topology establishment and updating in OFDP were analyzed, and an improved protocol named Im-OFDP (improved OpenFlow discovery protocol) based on the minimum vertex covering problem in graph theory was proposed. In Im-OFDP, the switch port table and network link table were initially established using prior information obtained from OFDP network topology discovery. Subsequently, a graph model of the network topology was constructed, and the minimum vertex covering algorithm in graph theory was employed to select specific switches for the reception and forwarding of topology discovery link layer discovery protocol (LLDP) packets. Multi-level flow tables were designed based on the network topology structure, and these flow entries were installed on the selected switches by the controller to process LLDP packets. To address security issues, dynamic check code verification in LLDP packets was employed to ensure the safety of network links. Additionally, a network equipment information maintenance mechanism was established based on known network topologies to ensure the safety of the network equipment. Experimental results demonstrate a significant reduction in the number of network topology discovery messages, bandwidth overhead, and CPU overhead through the deployment of Im-OFDP. Moreover, the response time for node failures and link recovery time after mode failure is substantially reduced. Im-OFDP also effectively mitigates various network topology pollution attacks, such as link fabrication and switch forgery attacks. Overall, Im-OFDP has the capability to enhance the efficiency and safety of SDN topology discovery.

Keywords: software defined network, network topology, network security

0 引言

软件定义网络 (SDN, software defined network) 是一种集中式的网络架构, 将控制决策功能从传统设备中分离, 由控制器集中统一管理, 能够动态部署网络功能, 提高网络规模的伸缩性, 极大地降低了维护成本。SDN 一经提出, 便引起了学术界和产业界的高度重视^[1-2], 不仅在数据中心网络、物联网^[3-4]、车联网^[5]、卫星网^[6]、光网络^[7-8]和无线传感网络^[9]等领域受到高度关注, 而且进行了实际的部署和应用。

SDN 技术的应用能提高网络的管理能力和传统网络安全的防护能力^[10], 但是其本身面临新的问题, 其中, 网络拓扑发现是 SDN 最核心的功能^[11]。在基于 OpenFlow 的 SDN 中, OFDP (OpenFlow discovery protocol) 通过发送和接收携带 LLDP (link layer discovery protocol) 报文信息的 Packet-Out 和 Packet-In 实现拓扑链路发现^[12]。由于 OpenFlow 交换机本身无法发送、接收和处理 LLDP 消息, 因此, 需要控制器启动该过程。拓扑发现过程是定期执行的, 每轮链路发现间隔是固定的时间, IEEE 802.1AB 标准中建议间隔时间是 30 s。大多数控制器采用 OFDP, 如 NOX、

POX、Floodlight、ONOS、Ryu、Beacon 等。各个控制器对于 OFDP 实现的方式基本相同, 差异主要在于每轮拓扑发现的间隔时间。基于 OFDP 的网络拓扑发现主要存在以下问题^[13-16]。

1) 控制器处理的报文量大。OFDP 中控制器会每个周期给交换机的每个端口发送 Packet-Out 报文, 同时处理收到的 Packet-In 报文, 导致大规模网络中控制器网络拓扑发现开销大, 且效率低。

2) 网络拓扑容易受到污染攻击。OFDP 中 LLDP 报文是以广播的方式进行传播的, 恶意主机收到报文会伪造或重放 LLDP 消息发送至控制器, 控制器由于无法判断报文真伪, 会解析出伪造链路^[17], 交换机也会被恶意利用^[18], 进而破坏网络拓扑。

本文针对目前 OpenFlow 网络拓扑发现协议的效率和安全问题, 基于图论的最小顶点覆盖算法, 对 OFDP 进行改进, 由控制器定制下发网络拓扑发现流表, 实现了一种高效安全的 SDN 拓扑发现协议——Im-OFDP (improved OFDP), 并改进了安全功能, 能够有效阻止多种类型的 SDN 拓扑污染攻击。本文的主要贡献如下。

1) 基于最小顶点覆盖算法, 提出了改进的软件定义网络拓扑发现协议, 在首次发现网络拓扑结构后, 大幅降低后续拓扑发现和维持网络的消

耗,并验证了其有效性。

2) 改进安全机制,首次结合了3种安全机制(设备注册、HashMac、端口过滤),能够免疫目前已知的链路伪造、主机伪造等多种类型的网络拓扑污染攻击,具有较好的安全性。

3) 在多种类型的网络拓扑环境中测试发现,Im-OFDP在效率 and 安全性上都优于现有方法。

1 相关工作

Im-OFDP的主要目的是提高SDN拓扑发现协议的效率和安全能力,国内外学术界做了大量的相关研究工作。文献[19]的研究发现拓扑发现对SDN控制器的中央处理器(CPU, central processing unit)、内存等系统资源的占用率很高(文献实验为树形拓扑,交换机数量从5到17,主机数量从16到256, CPU 占用率从17.1%提升到43.4%);文献[20]的研究发现网络拓扑污染攻击对控制器性能的影响非常大,被污染的拓扑会造成控制器发送无效的包,影响到正常链路的通信,如果造成环路则会导致整个网络无法工作;文献[17-18]用多个实验证实了链路伪造攻击和恶意交换机的危害。这些研究揭示了OFDP存在的效率和安全问题亟须改进。根据OFDP发现网络拓扑的功能流程^[12],相关研究如下。

(1) 网络拓扑发现效率

效率提升的主要手段是减少网络拓扑发现的报文。OFDPv2^[21]通过部署专用交换机流表的方式将控制器下发至每个交换机网络拓扑发现的报文减少到1个,相比OFDP有较大提升;SOFTDP^[22]观察交换机的状态,仅当网络的连接状态更改时,控制器才执行OFDP拓扑发现,由于只检测更新链路,因此降低了控制器的负载;SLDP^[23]通过创建合法端口列表来避免攻击,同时由于端口数量减少,效率上相比OFDP有一定提升;ESLD^[24]通过在发包时过滤掉主机端口,避免发送冗余数据包,进而减少发包数量;OFDPv3^[25]通过LLDP报文广播实现整个网络拓扑的发现,控制器选择一个交换机下发LLDP包,交换机收到后转发给控制器并进行广播,减少了Packet-Out消息的数量;LADP^[26]在OFDPv3的基础上进行了改进,避免了广播风暴的产生;OFDP-PD^[27]采用标记端

口的方式,实现Packet-Out和Packet-In数量相比OFDP减半的效果,提升了网拓扑发现效率;NewOFDP+^[28]采用每条链路只检测一条单向链路的方式实现Packet-Out和Packet-In数量相比OFDP减半的效果。此外,文献[29-31]通过在交换机上设置功能代理,承载部分网络拓扑发现功能,将网络拓扑发现Packet-Out报文减少为0;文献[32-34]通过设计新的网络拓扑发现协议,规避OFDP效率和安全的缺陷。

(2) 网络拓扑发现安全

TopoGuard^[35]分析了SDN拓扑发现面临的安全问题,通过端口属性管理、主机探测、拓扑更新验证等机制建立控制器网络拓扑安全功能模块。之后,学术界针对TopoGuard做了许多补充加固工作,文献[36]建立TopoGuard的形式化验证模型;TopoGuard+^[37]增加了TopoGuard未能防御的两个攻击;SPV^[38]建立主动探测验证的机制,从而加强网络拓扑安全能力;文献[39]评测了TopoGuard、TopoGuard+、SPV等网络拓扑保护方法,发现即使部署了这些方法,恶意攻击者仍然能伪造链路,这些主要是SDN具有比传统网络更高的动态性导致的;SPHINX^[40]通过建立控制器控制流的Flow Graph模型验证网络拓扑更新的异常,检测针对网络拓扑的攻击;ESLD^[24]在提高协议效率的同时,提出在控制器进行链路解析时增加校验机制,采用与TopoGuard^[36]相似的方法在LLDP报文中加入HashMac动态校验码,防止LLDP包被伪造或者重放;文献[41]通过统计分析链路时延实施攻击检测;FlowMon^[42]通过设计算法分析网络中数据包丢失和交换异常来识别恶意交换机;文献[43]建立主机和链路验证机制来达到防御恶意攻击的目的;文献[44]在软件定义车联网中实现了针对网络拓扑的多种攻击手段,同时认为LLDP报文加密验证^[24]、链路时延分析^[42]等方法会导致网络时延,建立有效的验证机制是防范网络拓扑污染攻击最基本的方法;SecTop^[45]在总结相关的网络拓扑安全防御的研究基础上,提出了一种安全的动态管理接入网络主机的算法和安全的链路发现算法,能够以极小的开销实现拓扑污染防御。

综上所述,学术界围绕SDN拓扑发现的效率和安全研究主要有改进OFDP的网络拓扑发现流

程、在交换机上部署网络拓扑发现功能、在控制器上增加安全功能、设计新的网络拓扑发现协议等，其中改进 OFDP 的网络拓扑发现流程和在控制器上增加安全功能较容易与现有技术标准兼容，易于实施，本文从这两个方面进行改进，文献[46]是本文方法的前期研究，本文相比文献[46]，在 Im-OFDP 的阐述上更详实，分析了 Im-OFDP 相对于其他协议的增益，以及具体的 Packet-Out/In 消息的增益效率，并且补充了与 LADP、OFDP-PD 的对比实验。

1) 在效率方面，减少 Packet-Out 消息数量和 Packet-In 消息数量。由于 OFDP 在进行拓扑发现时，Packet-Out 消息数量等于拓扑中所有交换机端口，Packet-In 消息数量为链路数量的两倍，因此在效率上有较大的提升空间。其中，OFDPv2、ESLD、OFDPv3 等方法均减少了控制器 Packet-Out 消息数量，而 OFDPv3、OFDP-PD 等方法均减少了 Packet-Out 和 Packet-In 数量。

2) 在安全方面，增加合法校验机制。TopoGuard 建立了网络拓扑信息的验证机制，ESLD 采用对所有 LLDP 包增加动态校验码的方式，实现了 LLDP 伪造和重放攻击的防御，sOFDP 和 SLDP 通过创建端口合法列表，处理合法端口上传的数据包，过滤非法端口上传的数据包的方式来确保网络拓扑的安全。

本文研究一方面极大提升 OFDP 的网络拓扑发现效率，另一方面加强 SDN 拓扑污染攻击防御，实现效率和安全的双重提升。针对效率问题，研究发现通过建立网络拓扑的图结构模型，计算关键的支撑交换机节点，结合相应的交换机流表设计，网络拓扑发现的报文可以极大减少（相对 OFDP 至少减少 50% 的消息数量）。针对安全问

题，本文参考文献[47-49]的研究结论和 secTop^[29] 的防御机制，结合 Im-OFDP 的网络拓扑发现流程，在以下 3 个方面加强网络拓扑污染攻击的防御能力：一是在 LLDP 中加入 HashMac 动态校验码，且只对网络拓扑更新的链路实施验证，在防御攻击的同时保证网络拓扑发现效率；二是对主机和交换机等网络设备建立注册校验机制，确保网络设备真实可信；三是对端口进行标记，过滤来自主机端口的 LLDP 消息，防御来自主机的攻击，同时减少控制器处理的消息数量。

2 Im-OFDP

Im-OFDP 将 SDN 拓扑发现过程分为初始化和更新两个阶段。在初始化阶段，首先定义交换机端口数据结构表和网络链路数据结构表，然后控制器基于 OFDP 收集 SDN 拓扑信息，利用图论的最小顶点覆盖算法选择网络拓扑的支撑交换机，构造专用流表下发至交换机，接着通过下发 LLDP 报文收集网络拓扑信息。在更新阶段，根据收集的拓扑信息检测网络拓扑更新状况，如果有更新，则依据更新信息生成新的端口映射表和链路信息表，重新筛选出支撑交换机并构建下发专用流表，进入下一轮网络拓扑发现；如果没有更新，则直接进入下一轮的网络拓扑更新流程。

图 1 是 Im-OFDP 网络拓扑发现过程，网络中的交换机 S2 分别与交换机 S1、S3、S4 和 S5 相连。OFDP 中的控制器需要向每个交换机的存活端口发送网络拓扑发现的 LLDP 报文，数量大、负载重，Im-OFDP 旨在抽象出网络拓扑的图结构，让控制器向交换机 S2 发送携带 LLDP 的 Packet-Out 报文，S2 收到报文后根据预定流表通过指定的链路端口

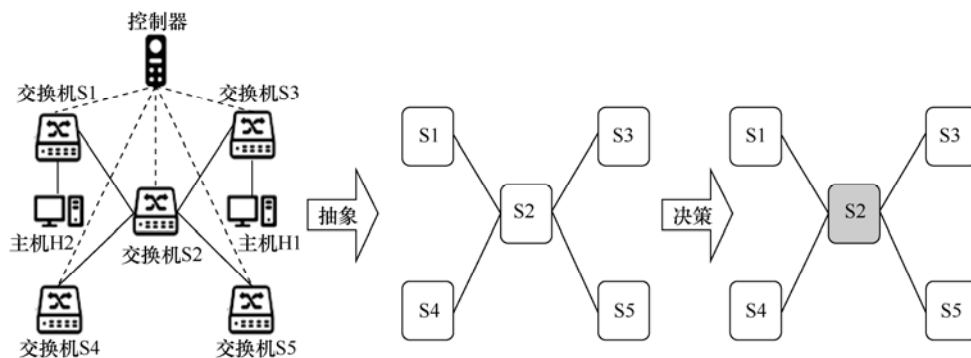


图 1 Im-OFDP 网络拓扑发现过程
Figure 1 Process of network topology discovery in Im-OFDP

向 S1、S3、S4、S5 转发 LLDP 报文，相应的交换机收到 LLDP 报文后，经过预定流表处理封装在 Packet-In 报文中转发给控制器，从而实现网络拓扑的发现，控制器不再向 S1、S3、S4、S5 发送 LLDP 报文，显著减少整个网络拓扑发现的报文数量。

2.1 网络拓扑信息表

Im-OFDP 将交换机端口分为 4 类：主机设备 (HOST) 类型、双向链路 (BIDI) 类型、单向链路 (UNDI) 类型、未注册 (UNRE) 类型。HOST 类型表示该端口所连接的设备为主机等终端设备；BIDI 类型表示该端口所连接的是交换机，且二者之间的链路为双向链路；UNDI 类型表示该端口所连接的是交换机，且二者之间的链路为单向链路；UNRE 类型表示该端口处于激活状态，暂时未确定其所连设备。交换机端口绑定表的构造方法如下。

第一步：控制器与交换机建立连接，监听并解析交换机的 feature_reply 消息，用于初始化端口信息。

第二步：控制器建立每个交换机的介质访问控制 (MAC, medium access control) 地址和端口 ID 之间一对一的映射关系，使得 MAC 地址成为有效的唯一端口标识符，并将端口类型初始化为未注册类型 UNRE，再将映射关系存入数据库中。端口信息数据结构如下。

端口信息数据结构

```
Class PortInfo {
    SwitchDPID sdpid; /* 交换机 DPID */
    PortNum pnum; /* 端口号 */
    MacAddress mac; /* 端口 MAC 地址 */
    Type ptype; /* 端口类型 */
}
```

控制器同时初始化链路信息表，数据结构如下。

链路信息数据结构

```
Class LinkInfo {
    SrcDPID sdpid; /* 源交换机 DPID */
    SrcPortNum spnum; /* 源端口号 */
    DstDPID ddpid; /* 目的交换机 DPID */
    DstPortNum dpnum; /* 目的端口号 */
    Type ltype; /* 端口类型 */
}
```

2.2 SDN 支撑交换机选择算法

假设 $G(V, E)$ 表示 SDN 拓扑， V 是顶点（交换机）的集合， E 是网络链路的集合。 S 表示网络中的一台交换机， V' 是 V 的一个子集， $S \in V$ ， $V' \subseteq V$ 。如果任意一条链路 $L \in E$ 的两个顶点中至少有一个顶点属于 V' ，则称 V' 为网络拓扑支撑交换机集合。如果 $S \in V'$ ，则称 S 为支撑交换机；网络拓扑支撑交换机的集合可能有多个，如果 V' 包含的交换机数量最小，则称 V' 为最小支撑交换机集合。

从 2.1 节的端口类型分析中可以看到，SDN 拓扑链路发现不需要对所有交换机的端口发送链路发现协议报文，可以根据网络当前的拓扑结构进行选取，只发送给部分交换机的链路类型端口，即可完成整个链路的探测。

最小支撑交换机集合即图论中的最小顶点覆盖，文献[47,50]研究表明，最小顶点覆盖问题是一个经典的非确定多项式 (NP, nondeterministic polynomially) 问题，Im-OFDP 中采用混合贪心算法来求解。SDN 拓扑静态支撑交换机筛选算法如算法 1 所示。

算法 1 SDN 拓扑静态支撑交换机筛选算法

输入 网络拓扑图 $G(V, E)$

输出 网络拓扑支撑交换机集合 V'

Step1 图 $G(V, E)$ ，计算集合 $\bar{V}$ 。记 $d^* = \max\{d(v) \mid \forall v \in V\}$ ， $\bar{V} = \{v \mid \forall v \in V, d(v) = d^*\}$ ； $d(v)$ 表示顶点 v 的度数

Step2 如果 $|\bar{V}| = 1$ ，对于 $v^* \in \bar{V}$ ，转至 Step4；

如 $|\bar{V}| > 1$ ，则转至 Step3； $||$ 表示集合的势

Step3: 记 $\bar{E} = \{(v1, v2) \mid \forall v1, v2 \in \bar{V}\}$ ，图 $G(\bar{V}, \bar{E})$ 中，记 $d^{**} = \max\{d(v) \mid \forall v \in \bar{V}\}$ ， $\bar{V} = \{v \mid \forall v \in \bar{V}, d(v) = d^{**}\}$ ，对于 $v^* \in \bar{V}$ ，转至 Step4；

Step4 将顶点 v^* 加入 V_S ；

Step5 $E = E - \{(v, v^*) \mid \forall v \in V\}$ ， $V = V - \{v^*\}$ ，

如果 $|E| = 0$ ，则转至 Step6；否则转至 Step1；

Step6 输出 V' 。

在大规模网络拓扑中，只有局部极少数链路发生变化的场景下，可以使用动态最小顶点覆盖算法，采用增量式迭代思路，利用原集合进行迭代，进而求解新覆盖集，达到速度快、变更开销小、变更范围小的效果。在“规模大，变动少”的情况下，相比静态支撑交换机筛选算法，动态

最小顶点覆盖算法的时间复杂度更低，可达到 $O(n)$ 。动态支撑交换机筛选算法如算法 2 所示。

算法 2 动态支撑交换机筛选算法

输入 网络拓扑图 $G(V, E)$ ，原覆盖集 V_n ，新增链路集合 E_{new} // V 表示网络中的交换机， E 表示网络链路

输出 网络拓扑支撑交换机集合 V_n'

Step1 初始化覆盖集 $V_n' = V_n$;

Step2 如果 $\forall e \in E_{\text{new}}, (v_1, v_2) \in e$ ，有 $v_1 \in V_n' \parallel v_2 \in V_n'$ ，则转至 Step6；否则转至 Step3；

Step3 如果 $|E_{\text{new}}| > 1$ ，设 $v \in V$ 且 $\forall (v_1, v_2) \in E_{\text{new}}$ ，都有 $v_1 = v$ 或 $v_2 = v$ ，将 v 加入 V_n' 中，转至 Step6；如果 $|E_{\text{new}}| = 1$ ，则转至 Step4；

Step4 $|E_{\text{new}}| = 1$ ，如果 $\forall (v_1, v_2) \in E_{\text{new}}$ ，且 $v_1 \in V_n' \parallel v_2 \in V_n'$ ，则转至 Step6； $V_{n+1}' = V_n'$ ，将 v_1 加入 V_{n+1}' ， $V_{n+1}'' = V_n$ ，将 v_2 加入 V_{n+1}'' ， $E_u = E_{\text{new}} \cup E$ ；

Step5 计算是否存在 $V^* = \{v \mid \forall v \in V_{n+1}', |V^*| = |V_{n+1}'| - 1\}$ ， $\forall (v_1, v_2) \in E_u$ ，都满足 $v_1 \in V^* \parallel v_2 \in V^*$ ，如果存在，则 $V_n' = V^*$ ，转至 Step6；计算是否存在 $V^* = \{v \mid \forall v \in V_{n+1}'', |V^*| = |V_{n+1}''| - 1\}$ ， $\forall (v_1, v_2) \in E_u$ ，满足 $v_1 \in V^* \parallel v_2 \in V^*$ ，如果存在，则 $V_n' = V^*$ ，转至 Step6； $V_n' = V_{n+1}'$ ，转至 Step6；

Step6 计算结束，输出 V_n' 。

OFDP 在节点/链路失效后，会进行节点/链路的删除，并重新进行链路发现。Im-OFDP 在节点/链路失效后，同样进行节点/链路的删除，并重新进行链路发现，但是 Im-OFDP 调用静态最小顶点覆盖算法，计算出新的支撑交换机，然后根据新的支撑交换机集合进行链路发现。

2.3 网络拓扑发现流表构造

通过支撑交换机选择算法得到支撑交换机集合，需要针对网络拓扑发现在这些交换机上配置相应的转发规则。Im-OFDP 构造的流表规则有 3 种。

规则①：协议类型为 LLDP，匹配优先级为 High，报文来源为控制器，Src MAC 字段为 0，动作为从指定端口转发。

规则②：协议类型为 LLDP，匹配优先级为 Middle，条件是如果 Src MAC 为 0，则将 Src MAC 的值替换为接收端口的 MAC 地址，动作为从接收端口转发。

规则③：协议类型为 LLDP，匹配优先级为 Low，动作为发给控制器。

规则①为在最小支撑交换机集合 V' 中安装的流表规则，而规则②、规则③为在每个交换机中均安装的流表规则。

针对最小支撑交换机集合 V' 中的交换机构造流表规则①的方法如下。

1) 遍历 V' 中的交换机，依次构造流表。如果未构造，则进行步骤 2)，否则进行步骤 6)。

2) 交换机 S 构造流表。S 的流表动作列表为 ACT (actions)，遍历交换机 S 的所有可用端口，检测是否还有未处理的端口，如果有，则执行步骤 3)；否则执行步骤 5)。

3) 对交换机 S 的端口 P，在端口信息表中查找 P 的类型。如果端口已经被标记，则进行步骤 2)，处理下一个端口；如果未被标记，则进行步骤 4) 处理。

4) 如果端口 P 的类型为 HOST，则不做处理；如果端口 P 类型为 UNRE，则对端口构造单向规则，即复制并重写 LLDP 的 Src MAC 字段，然后从端口 P 发出，将动作加入 ACT 中；如果端口类型为 UNID，则对端口构造单向规则，即重写 LLDP 的 Src MAC 字段，然后从端口 P 发出，将动作加入 ACT 中；如果端口类型为 BIDI，则对端口构造双向规则，即将 LLDP 包复制并直接从端口 P 发出，将动作加入 ACT 中，同时在链路信息表中查找 P 所在链路的另一个端口，将该端口进行标记。转到步骤 2)，处理下一个端口。

5) 交换机 S 的所有端口处理完毕，转至步骤 1)，处理下一个交换机。

6) V' 中所有交换机处理结束，对于 V' 中每一个交换机 S 都有对应的 ACT 流表动作列表与之对应。

7) 将流表构造完毕之后，下发到 V' 中相应的交换机。其中公共规则②、规则③为拓扑中所有交换机均下发，然后根据 ACT 给 V' 中的每个交换机构造流表作为规则①下发到 V' 中的各个交换机。

2.4 LLDP 数据包构造

计算得到支撑交换机集合后，控制器为每个支撑交换机构造一个 LLDP 数据包。OFDP 中的 LLDP 帧结构如图 2 所示。Im-OFDP 保持 OFDP 中 LLDP 的报文结构不变，但与 OFDP 不同的是，

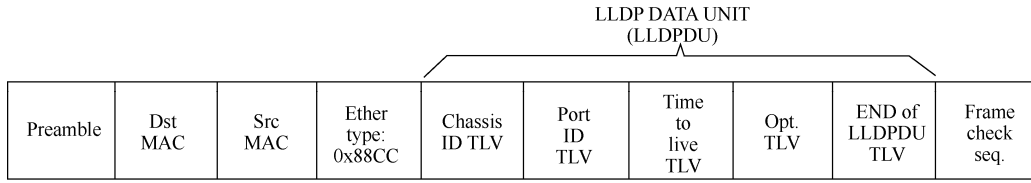


图2 LLDP 帧结构
Figure 2 Structure of LLDP frame

Im-OFDP 不再通过 Chassis ID 和 Port ID 解析链路端口, 因此统一将 Port ID 置为 0。Im-OFDP 主要依赖 Src MAC 和 Chassis ID 这两个字段的内容解析链路。通过分析 Src MAC 字段来识别链路的源交换机和端口, 在构造 LLDP 数据包时, 将 LLDP 报文中的 Src MAC 置为 0; 通过分析 LLDP 报文的 Chassis ID 字段判断链路的类型, 即单向或双向链路, 因此 Chassis ID 字段置为对应支撑交换机的 ID。

2.5 性能分析

假设在一个 SDN 拓扑中, 数据层由 SDN 交换机和主机设备组成, 任意两个交换机之间均存在双向通信链路, 即网络拓扑是一个连通图, 且交换机的存活端口连接的是主机或者双向链路。交换机的数量为 N ($N \geq 2$), 主机设备数量为 H ($H \geq 0$), 链路数量为 L ($L \geq 1$)。

Im-OFDP 仅向每个支撑交换机下发 1 个 Packet-Out 消息, 故消息数量为支撑交换机数量, 记为 N_x , 根据图论相关知识, 可得到 $1 \leq N_x \leq N-1$; 对于 Packet-In 消息, 每条双向链路通过 Im-OFDP 双向链路解析规则仅需一个 Packet-In 消息, 故 Packet-In 消息数量为 L 。

通过对相关研究工作的分析, 本文选择了 OFDP、OFDPv2、LADP、OFDP-PD 这 4 种最相关的网络拓扑发现协议进行对比分析, 各种网络拓扑发现协议消息数量对比结果如表 1 所示。

表1 各种网络拓扑发现协议消息数量对比结果
Table 1 Comparison results of message number in various network topology discovery protocols

网络拓扑发现协议	Packet-Out	Packet-In	消息数
OFDP	$H+2L$	$2L$	$H+4L$
OFDPv2	N	$2L$	$S+2L$
LADP	1	$2L$	$1+2L$
OFDP-PD	$H+L$	L	$H+2L$
Im-OFDP (含校验码)	$2L$	$2L$	$4L$
Im-OFDP (无校验码)	$N_x \in (1, N-1)$	L	N_x+L

为了比较各种协议的性能改进程度, 定义效率增益 G , 比较 Im-OFDP 相对其他 4 种拓扑发现协议消息数量的减少情况。 G_1 、 G_2 、 G_3 、 G_4 分别为 Im-OFDP 和 OFDP、OFDPv2、LADP、OFDP-PD 相比发现协议消息减少的数量与相应协议消息数量的比例。

$$G_1 = \frac{3 - (N-1)/L}{4} \geq 1 - \frac{L+L}{4L} = \frac{1}{2} \quad (1)$$

$$G_2 = 1 - \frac{N_x + L}{N + 2L} \geq 1 - \frac{(N-1) + L}{N + 2L} \geq \frac{1}{2} \quad (2)$$

$$G_3 = 1 - \frac{N_x + L}{1 + 2L} \geq \frac{1}{1 + 2L} > 0 \quad (3)$$

$$G_4 = 1 - \frac{N_x + L}{H + 2L} \geq 1 - \frac{L + L}{2L} = 0 \quad (4)$$

由式(1)~式(4)可知, Im-OFDP 在单个拓扑发现周期内消息总数相对 OFDP 和 OFDPv2 的理论效率增益均不小于 $1/2$, 而相对 LADP 和 OFDP-PD 的理论效率增益不小于 0。综上, Im-OFDP 相比其他拓扑发现协议能够有效减少拓扑发现过程中 Packet-Out 和 Packet-In 的消息数量。

2.6 复杂度分析

Im-OFDP 的算法复杂度主要来自交换机选取决策算法, 决策算法仅在拓扑变更时运行, 下面分别对时间复杂度和空间复杂度进行分析。设链路数量为 L , 交换机数量为 N , 主机数量为 H 。

(1) 时间复杂度

时间复杂度由以下几方面构成: 选择最大邻接度交换机 $O(N^2)$; 构造端口绑定表 $O(2L+H)$; 标记链路 $O(L)$, 累加得 $O(N^2)+O(3L)+O(H)$, 故时间复杂度为 $O(N^2+3L+H)$ 。

(2) 空间复杂度

空间复杂度由以下几方面构成: 存储端口表 $O(2L+H)$, 存储链路信息表 $O(L)$, 累加得 $O(3L+H)$, 故空间复杂度为 $O(3L+H)$ 。

从复杂度分析的结果可以看出,网络拓扑发现协议与网络拓扑规模高度相关,当网络达到一定规模时,拓扑发现的效率低是难以避免的。

2.7 安全机制改进

基于 2.3 节构造的网络拓扑发现流表,交换机不会向 HOST 类型端口转发 LLDP 包,因此主机无法接收到 LLDP 数据包,这在一定程度上提升了拓扑发现的安全性。但是仍然存在安全问题,若攻击者通过某种方式直接构造一个 LLDP 数据包,并通过 UNRE 端口转发给控制器,由于缺乏数据包校验机制,无法判断数据包是否合法,控制器会按照正常逻辑进行解析,得到一条伪造链路,从而网络视图错误。除链路伪造攻击存在外,由于没有对设备的注册消息进行校验,因此存在设备伪造攻击。

为了提升安全性,Im-OFDP 对于设备伪造攻击,建立设备注册检验机制,在控制器中构建设备信息表,通过监听主机/交换机的注册和注销等信息,对设备信息表进行维护,当设备发生伪造迁移时,在设备信息表中进行核对校验,从而实现设备伪造攻击的防御;对于链路伪造攻击,建立 LLDP 动态加密验证机制对首次生成的链路进行验证,保护链路安全,同时建立端口过滤、链路重用检测等异常检测机制确保链路生成之后的网络拓扑不被伪造。

(1) 设备信息表构造

设备信息表用于登记网络中的交换机和主机设备信息,也用于防御设备伪造攻击。设备信息表数据结构描述如下,主要包含 3 个字段。

设备信息表数据结构

```
Class DeviceInfo {
    DeviceID did; /* 设备 ID 编号 */
    MacAddres mac; /* 设备 MAC 地址 */
    DeviceType dtype; /* 设备类型 */
}
```

① 设备 ID 编号字段:记录主机或交换机的设备编号。对于交换机的设备编号,控制器通过与交换机握手时的 feature_replay 消息中获取;对于主机的设备编号,通过控制器自带的主机发现模块获取。

② 设备 MAC 地址字段:记录主机或交换机的设备 MAC 地址。控制器通过分析主机向控制

器发送的数据包请求和交换机向控制器发送的配置信息中获取。

③ 设备类型字段:用于标记主机或交换机设备类型,即 HOST 或 SWITCH。

(2) LLDP 数据包加密机制

由于 LLDP 数据包存在被伪造的可能,因此在 Im-OFDP 中对部分 LLDP 数据包进行加密校验。在原有 LLDP 帧结构的 LLDPDU 中加入 HMAC-MD5 校验码和时间戳两个 TLV 字段,校验码随时间动态变化,实现动态加密,从而防止 LLDP 被伪造,增加字段的 LLDP 帧结构示意图如图 3 所示。

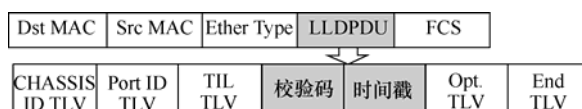


图3 增加字段的 LLDP 帧结构
Figure 3 LLDP frame structure with field

LLDP 动态校验码生成式

$$\text{HMAC}(K, M) = H[K \oplus \text{opad} | H(K \oplus \text{ipad} | M)] \quad (5)$$

其中, K 为密钥, M 为待加密的消息, M 消息格式为 PortMac-当前时间-交换机编号-交换机端口号, opad 和 ipad 为迭代参数,分别为 0x5c 字符串和 0x36 字符串, H 为加密算法,由于 HMAC-MD5 算法具有加密过程不可逆、密文长度固定等优势,因此 Im-OFDP 采用 HMAC-MD5 算法进行加密。

3 实验分析

本文的实验平台采用 POX 作为控制器,Mininet 作为仿真模拟器,OVS 作为 SDN 交换机,操作系统为 Ubuntu16.04,编程语言为 Python,搭建了 4 种典型拓扑结构,实验网络拓扑图参数如表 2 所示。其中, N 为交换机数量, P 为交换机端口数量, H 为主机数量, L 为链路数量, SN 为支撑交换机数量。实验同时针对 OFDP^[12]、OFDPv2^[21]、LADP^[26]、OFDP-PD^[27]这 4 种网络拓扑发现机制进行对比分析。

文献[32]方案将拓扑发现的功能代理到交换机上,控制器发出的 Packet-Out 报文降为 0,然而,该方案需要对现有的 OVS 进行 LLDP 模块的部署和运行,因此不能与上述 4 种网络拓扑发现机制运行在同一平台进行比较。文献[32]方案确实极大地降低了控制器的负载,然而它对基础设

施（交换机）的改动，导致它与现有的基础设施不兼容，也难以在同一平台对其进行比较。

表2列举的拓扑结构中，Topo1是树形拓扑，是深度为4的四叉树，Topo2是深度为6的二叉树，这两种树形拓扑的主机连接在树的最下层叶子交换机节点上。图4描述了深度为3的三叉树拓扑，方形表示交换机，椭圆形表示主机。Topo3是线形拓扑，由100个交换机构成，图5描述了3个交换机构成的线形拓扑。FatTree拓扑如图6所示。Topo5是由一个C++随机拓扑生成程序生成的随机拓扑，由20个交换机和20个主机组成。

表2 实验网络拓扑图参数
Table 2 Parameters of network topologies in experiment

拓扑	拓扑结构	N	P	H	L	SN
Topo1	Tree, $d=4, f=4$	85	424	256	84	17
Topo2	Tree, $d=6, f=2$	63	188	64	62	21
Topo3	Linear, $N=100$	100	298	100	99	50
Topo4	FatTree, $k=4$	20	80	16	32	8
Topo5	随机 Topo	20	72	20	26	10

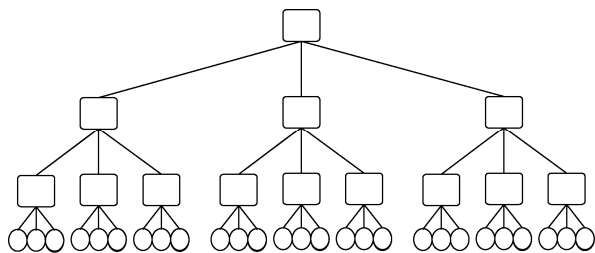


图4 深度为3的三叉树拓扑
Figure 4 Topology of a trinomial tree with depth 3

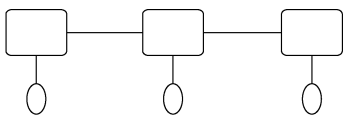


图5 3个交换机构成的线形拓扑
Figure 5 Linear topology consisting of three switches

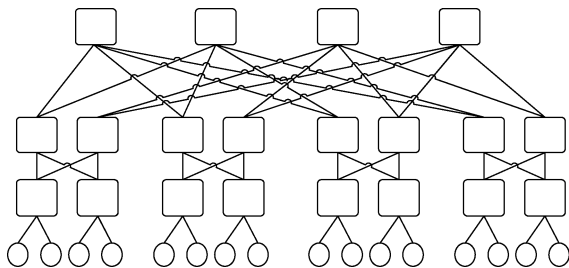


图6 FatTree 拓扑
Figure 6 Topology of FatTree

3.1 网络拓扑发现报文数量分析

网络拓扑发现消息数量表示在一个拓扑发现周期内，控制器需要下发的和接收的 LLDP 数据帧的数量，即携带 LLDP 报文的 Packet-Out 和 Packet-In 消息的数量和。拓扑发现的负载主要来源于控制器处理 LLDP 报文的数量，因此拓扑发现过程中所使用的报文数量直接影响拓扑发现的效率。

SDN 拓扑发现按周期进行轮询，单个拓扑发现周期的 LLDP 消息数量通过统计控制器的端口网卡上每个链路周期所携带 LLDP 报文的 OpenFlow 消息获得。表3中记录了单个拓扑发现周期不同拓扑发现协议的 LLDP 消息数量。

表3 不同拓扑发现协议的 LLDP 消息数量
Table 3 Number of LLDP messages in topology discovery protocols

拓扑	消息数量				
	OFDP	OFDPv2	LADP	OFDP-PD	Im-OFDP
Topo1	592	253	169	424	101
Topo2	312	187	125	188	83
Topo3	496	298	199	298	149
Topo4	144	84	65	80	40
Topo5	124	72	53	72	36

从表3可以看出，Im-OFDP在拓扑发现过程中相比其他4种协议所使用的消息数量大幅减少，减少比率最高在Topo1中，相比OFDP，最高减少达82.9%；相比OFDPv2最高减少60.1%；相比LADP最高减少40.2%；相比OFDP-PD最高减少76.2%。由此可见，Im-OFDP能够显著减少LLDP报文数量。

3.2 带宽占用率分析

在拓扑发现过程中，消息数量和带宽开销强相关，因为携带 LLDP 报文的 Packet-Out 和 Packet-In 消息的大小是已知且恒定的，通过 Packet-Out 和 Packet-In 消息的数量，结合包的大小，可以直接计算出一个拓扑发现周期内控制器与交换机进行拓扑发现所使用的通信带宽情况。为了研究通信带宽开销，使用 Wireshark 抓包工具测量了各种拓扑发现协议 Packet-Out 和 Packet-In 消息报文的大小，然后结合每个实验拓扑所使用的消息数量得到通信带宽开销，各种网络拓扑发现协议的带宽开销对比如图7所示。由图7可知，Im-OFDP相比OFDP带宽开销下降

70%以上 (Topo3), 相比 OFDPv2 下降 50%以上 (Topo3), 相比 LADP 下降 15%以上 (Topo3), 相比 OFDP-PD 下降 50%以上 (Topo3)。因此, Im-OFDP 可以有效降低在拓扑发现过程中控制器与交换机的通信带宽开销。

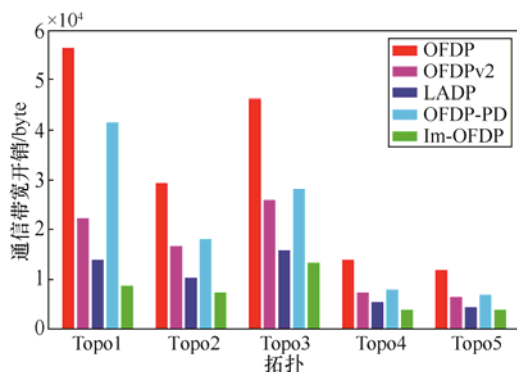


图7 各种网络拓扑发现协议的带宽开销对比
Figure 7 Bandwidth overhead of various network topology discovery protocols

3.3 网络拓扑发现鲁棒性分析

针对节点失效的问题, OFDP 和 Im-OFDP 均采用回显请求消息 (echo request message) 对节点的有效性进行持续探测; 针对链路失效的问题, 均采用检测链路的 timeout (在 POX 中, 这个时间为 20 s) 时间进行链路状态的维持。

图8是 OFDP 和 Im-OFDP 对于节点失效的响应时间对比, 图9是 OFDP 和 Im-OFDP 在节点失效后的链路恢复时间对比。由图8、图9可知, OFDP 在节点失效后进行响应和链路发现的速度更快。这是因为在链路恢复过程, Im-OFDP 相对 OFDP, 减少了链路发现的开销, 仅发送支撑交换机数量的 LLDP 报文 (OFDP 向所有端口发送报文), 但是相应增加了拓扑变化后计算支撑交换机的开销。因此在 Topo1、Topo2、Topo3 中, OFDP 的恢复时间明显低于 Im-OFDP, 因为它们的拓扑节点更多, 需要更多的计算开销。而在 Topo4、Topo5 中, OFDP 与 Im-OFDP 的节点/链路恢复时间没有显著区别, 因为它们的节点数量较少, 计算支撑交换机的开销更小。

以上分析表明: Im-OFDP 增加了网络拓扑结构的计算环节, 导致部分节点失效后的网络拓扑的恢复较慢, 恢复的速度与网络拓扑的规模有显著关系。

3.4 控制器 CPU 负载分析

CPU 时间开销能够直接反映拓扑发现过程中控制器的负载情况, 是评价拓扑发现方法效率和性能的重要指标。在 POX 中, 拓扑发现周期默认为 5 s, 本文统计不同拓扑发现协议在 5 种不同拓扑结构中进行 600 s 拓扑发现, 即 120 个拓扑发现周期, 记录拓扑发现过程中控制器累计需要的 CPU 时间开销, 各种网络拓扑发现协议的 CPU 时间开销对比如图10所示。

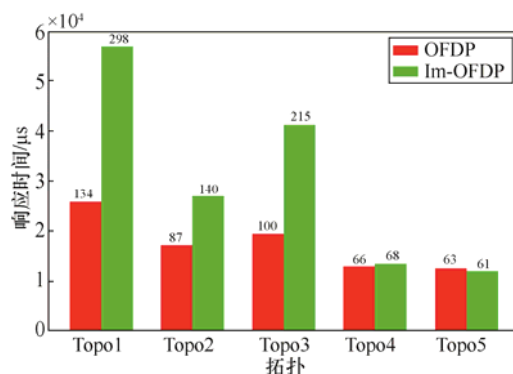


图8 节点失效的响应时间对比
Figure 8 Response time contrast of node failure

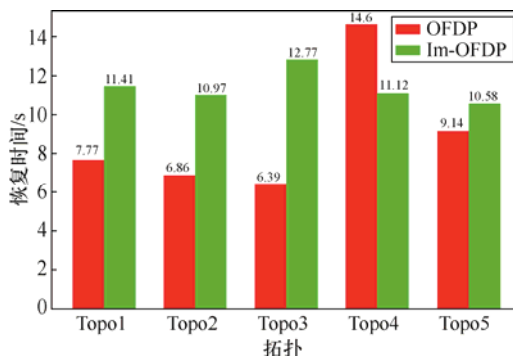


图9 节点失效后的链路恢复时间对比
Figure 9 Link recovery time contrast after node failure

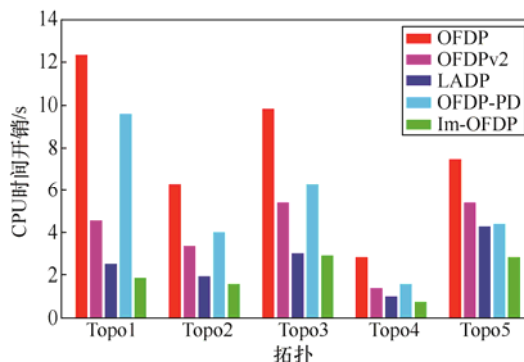


图10 各种网络拓扑发现协议的 CPU 时间开销对比
Figure 10 CPU time overhead contrast of various network topology discovery protocols

从图 10 看出,相比 OFDP、OFDPv2、LADP、OFDP-PD, Im-OFDP 所需要的 CPU 时间开销最小。相比 OFDP, Im-OFDP 的 CPU 开销减少 50% 以上,而对比其他几种优化协议,在各个拓扑中, Im-OFDP 的效率表现也是最优的。

在网络 Topo1 中,各种拓扑发现协议在 600 s 内的 CPU 累计时间开销对比如图 11 所示,从中可以看出, Im-OFDP 的 CPU 累计时间开销是最小的。

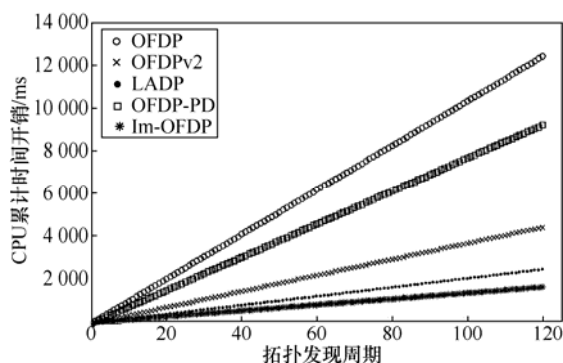


图 11 CPU 累计时间开销对比
Figure 11 CPU cumulative time overhead contrast

3.5 安全分析

现有的不同协议安全性对比如表 4 所示。

表 4 不同协议安全性对比
Table 4 Safety contrast of various protocols

协议	攻击方式			
	LLDP 重放	LLDP 伪造	交换机伪造	主机伪造
OFDP	×	×	×	×
OFDPv2	×	×	×	×
LADP	√	√	×	√
OFDP-PD	×	×	×	×
Im-OFDP	√	√	√	√

Im-OFDP、LADP 对于拓扑发现的过程增加了安全措施,其他协议则没有增加安全措施。LADP 采用“block-ports”拒绝来自恶意端口的 LLDP 报文保护拓扑,而 Im-OFDP 采用合格端口列表,拒绝来自列表外端口的 LLDP 报文,二者在收到来自主机端口的攻击时,都可以进行抵御。然而, LADP 没有设备注册机制,在收到交换机伪造的信息时,不能进行甄别,而 Im-OFDP 则可以有效识别到此类攻击。

为验证现有 OFDP 方法在拓扑发现中存在的漏洞,同时测试 Im-OFDP 对于拓扑污染攻击的防

御能力,本文采用的安全测试网络拓扑如图 12 所示,拓扑图包含 S1~S8 共 8 个交换机, h1~h5 共 5 个主机分别与交换机 S1、S2、S4、S7、S8 连接, 9 条链路。实验平台保持不变,配置并启动 POX 控制器,观察控制器的网络拓扑视图。

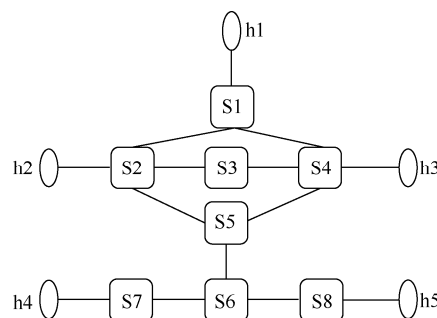


图 12 安全测试网络拓扑
Figure 12 Network topology for security testing

总结目前已知的各种网络拓扑污染攻击,分为 4 种方式: LLDP 报文重放攻击、LLDP 报文伪造攻击、交换机设备伪造攻击和主机设备伪造攻击,实验首先对 OFDP 进行测试,上述 4 种攻击方式都能够破坏网络拓扑视图;然后对 Im-OFDP 进行测试,具体方案如下。

1) LLDP 报文重放攻击。攻击的发起节点为 h4,在 h4 上使用 TCPdump 进行监听抓包,发现无法获取 LLDP 报文。

2) LLDP 报文伪造攻击。攻击的发起节点为 h4,伪造带校验码和不带校验码的 LLDP 数据包,分别向 HOST 类型和 UNRE 类型的交换机端口发起攻击,发现网络拓扑视图没有改变,图 13 是各种 LLDP 报文伪造攻击的防御日志。

```

openflow.discovery_in_OFDP] SrcPortInfo: port_type:host DPID:8 Port:11
[openflow.discovery_in_OFDP] DstPortInfo: port_type:host DPID:7 Port:11
[openflow.discovery_in_OFDP] Error: host port LLDP packet is illegal!
[openflow.discovery_in_OFDP] SrcPortInfo: port_type:bidl DPID:8 Port:1
[openflow.discovery_in_OFDP] DstPortInfo: port_type:host DPID:7 Port:11
[openflow.discovery_in_OFDP] Error: host port LLDP packet is illegal!
[openflow.discovery_in_OFDP] SrcPortInfo: port_type:unre DPID:8 Port:11
[openflow.discovery_in_OFDP] DstPortInfo: port_type:unre DPID:7 Port:11
[openflow.discovery_in_OFDP] Error: unre port LLDP packet without Check_Code TLVs!
[openflow.discovery_in_OFDP] SrcPortInfo: port_type:bidl DPID:8 Port:1
[openflow.discovery_in_OFDP] DstPortInfo: port_type:unre DPID:7 Port:11
[openflow.discovery_in_OFDP] ..... security verification .....
[openflow.discovery_in_OFDP] 20201229-215005
[openflow.discovery_in_OFDP] 20201229-215005
[openflow.discovery_in_OFDP] ciphertext_content_from_packet ==96d5424ddfe6048da035ertcd33d0
[openflow.discovery_in_OFDP] ciphertext_content_from_calculate ==96d5424ddfe6048da035ertcd33d0
[openflow.discovery_in_OFDP] Error:ciphertext content verification failed!
[openflow.discovery_in_OFDP] SrcPortInfo: port_type:bidl DPID:8 Port:1
[openflow.discovery_in_OFDP] DstPortInfo: port_type:unre DPID:7 Port:11
[openflow.discovery_in_OFDP] ..... security verification .....
[openflow.discovery_in_OFDP] 20201229-215005
[openflow.discovery_in_OFDP] 96d5424ddfe6048da035ertcd33d0
[openflow.discovery_in_OFDP] ciphertext_content_from_packet ==96d5424ddfe6048da035ertcd33d0
[openflow.discovery_in_OFDP] ciphertext_content_from_calculate ==96d5424ddfe6048da035ertcd33d0
[openflow.discovery_in_OFDP] True, ciphertext content verification succeeded!
[openflow.discovery_in_OFDP] Link port reuse detection: Link(dpid=8, port=1, dpid2=4, port2=3)
[openflow.discovery_in_OFDP] Link port reuse, the port(8:1) appears in Link(Link(dpid=8, port=1, dpid2=4, port2=3))
[openflow.discovery_in_OFDP] Add link failed,link-[7:11:8:1,False]

```

图 13 LLDP 报文伪造攻击的防御日志
Figure 13 Defense logs for LLDP message forgery attack

3) 交换机设备伪造攻击。交换机设备伪造攻

击的攻击对象是交换机 S6, 恶意交换机伪造一个携带交换机 S6 的 DPID 的 OFPT_FEATURES_REPLY 响应消息, 发送给控制器, 发现网络拓扑视图没有改变, 图 14 是交换机设备伪造攻击的防御日志。

```

[openflow.of_01] func_handle_HELLO
[openflow.of_01] func_handle_FEATURES_REPLY
[openflow.of_01] [00-00-00-00-00-10] connected.
[openflow.discovery_in_OFDP] ----- Device Attack Detect -----
[openflow.discovery_in_OFDP] Controller receive a switch connection request
[openflow.discovery_in_OFDP] Start to detect device_forge_attack...
[openflow.discovery_in_OFDP] Device info: (device id:6, device type:switch)
[openflow.discovery_in_OFDP] Existing switches:[4, 3, 8, 1, 2, 7, 5, 6]
[openflow.discovery_in_OFDP] Device_forge_attack: the switch(dpid=6) already exist!
[openflow.discovery_in_OFDP] -----

```

图 14 交换机设备伪造攻击的防御日志

Figure 14 Defense logs for switch device forgery attack

4) 主机设备伪造攻击。主机设备伪造攻击的攻击发起者是主机 h5, 伪造目标是主机 h1, 攻击者构造携带主机 h1 身份信息的数据包, 通过主机 h5 发送给交换机 S8, 发现网络拓扑视图没有改变, 图 15 是主机设备伪造攻击的防御日志。

```

[openflow.discovery_in_OFDP] ----- Device Attack Detect -----
[openflow.discovery_in_OFDP] Controller receive a host message from [Switch:8, Port:11, SrcMac: fa:cc:26:77:f3:9f]
[openflow.discovery_in_OFDP] Device info: (device id:fa:cc:26:77:f3:9f, device_type:host)
[openflow.discovery_in_OFDP] Existing hosts:[fa:cc:39:e7:00:5a, '08:ad:ed:b7:ed:b8', 'e2:4b:89:d9:f1:20', 'fa:cc:26:77:f3:9f']
[openflow.discovery_in_OFDP] Device_forge_attack: The host(mac=fa:cc:26:77:f3:9f) already exist!
[openflow.discovery_in_OFDP] -----

```

图 15 主机设备伪造攻击的防御日志

Figure 15 Defense logs for host device forgery attack

由于没有获取到 LADP 的可运行程序, 而其他协议相比 OFDP, 采取了相同的安全措施, 因此本文实验进行了与 OFDP 的安全性对比, 通过重现 OFDP 的 4 种拓扑污染攻击实验, 一方面验证了 OFDP 中存在的安全问题, 在遭受拓扑污染攻击时, 会导致控制器的网络拓扑视图被破坏; 另一方面验证了 Im-OFDP 防御网络拓扑污染攻击的能力。

4 结束语

高效的网络拓扑发现和安全防护能力一直是 SDN 亟须解决的关键问题。目前的主要思路一种是将拓扑发现功能从控制器移植到交换机上, 但是破坏了 SDN 交换机“白盒”架构, 另一种是改进现有的协议。本文基于第二种思路研究了现有的 SDN 拓扑建立和更新机制, 提出一种改进的拓扑发现协议——Im-OFDP。协议基于图论的最小顶点覆盖算法改进 OFDP 的网络拓扑发现流程, 能够显著提高大规模 SDN 拓扑更新效率, 同时在控制器上增加网络拓扑污染攻击防御功能, 防止 LLDP 报文重放和伪造、主机和交换机伪造等多种类型的恶意攻击。

实验结果表明, 在相对稳定的拓扑环境中, Im-OFDP 在控制器上实现了 SDN 拓扑发现和安全能力的提升, 具有网络拓扑发现报文数量少、占用带宽低、控制器 CPU 消耗少、网络拓扑发现快、安全性高等特点。同时, 由于 Im-OFDP 增加了网络拓扑结构的计算过程, 当网络拓扑的规模和动态性较大时, 拓扑的更新较慢。下一步研究将考虑网络拓扑动态变化继续改进协议, 提高 SDN 的网络拓扑发现和安全防护能力。

参考文献:

- [1] MCKEOWN N. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.
- [2] NUNES B A A, MENDONCA M, NGUYEN X N, et al. A survey of software-defined networking: past, present, and future of programmable networks[J]. IEEE Communications Surveys & Tutorials, 2014, 16(3): 1617-1634.
- [3] QIN Z, DENKER G, GIANNELLI C, et al. A software defined networking architecture for the internet-of-things[C]//Proceedings of IEEE Network Operations and Management Symposium. 2014: 1-9.
- [4] AHMED N, ROY A, MONDAL A, et al. SDN-based link recovery scheme for large-scale internet of things[C]//Proceedings of IEEE 22nd International Conference on High Performance Switching and Routing (HPSR). 2021: 1-6.
- [5] KU I, LU Y, GERLA M, et al. Towards software-defined VANET: architecture and services[C]//Proceedings of 13th Annual Mediterranean Ad Hoc Networking Workshop. 2014: 103-110.
- [6] GUO J M, YANG L, RINCÓN D, et al. SDN controller placement in LEO satellite networks based on dynamic topology[C]//Proceedings of IEEE/CIC International Conference on Communications in China (ICCC). 2021: 1083-1088.
- [7] MONTERO R, AGRAZ F, PAGES A, et al. Dynamic topology discovery in SDN-enabled transparent optical networks[C]//Proceedings of International Conference on Optical Network Design & Modeling. 2017: 1-6.
- [8] TARNARAS G, HALEPLIDIS E, DENAZIS S. SDN and ForCES based optimal network topology discovery[C]//Proceedings of IEEE Conference on Network Software. 2015: 1-6.
- [9] KIPONGO J, OLWAL T O, ABU-MAHFOUZ A M. Topology discovery protocol for software defined wireless sensor network: Solutions and open issues[C]//Proceedings of 27th International Symposium. Industrial Electronics. 2018: 1282-1287.
- [10] SHIN S, XU L, HONG S, et al. Enhancing network security through software defined networking[C]//Proceedings of 25th International Conference on Computer Communication and Networks, 2016: 1-9.
- [11] ZHU K, YAO Z, HE N, et al. Toward full virtualization of the net-

- work Topology[J]. IEEE Systems Journal, 2019, 13(2): 1640-1649.
- [12] OpenFlow discovery protocol and link layer discovery protocol[EB].
- [13] DACIER M C, KÖNIG H, CWALINSKI R, et al. Security challenges and opportunities of software-defined networking[J]. IEEE Security & Privacy, 2017, 15(2): 96-100.
- [14] AZZOUNI A, TRANG N T M, BOUTABA R, et al. Limitations of OpenFlow topology discovery protocol[C]//Proceedings of 16th Annual Mediterranean Ad Hoc Network Workshop. 2017:1-3.
- [15] KHAN S, GANI A, WAHAB A W A, et al. Topology discovery in software defined networks: threats, taxonomy, and state-of-the-art[J]. IEEE Communications Surveys & Tutorials, 2017, 19(1): 303-324.
- [16] TOUFGA S, ABDELLATIF S, OWEZARSKI P, et al. OpenFlow based topology discovery service in software defined vehicular networks: limitations and future approaches[C]//Proceedings of IEEE Vehicular Network Conference. 2018: 1-4.
- [17] SMYTH D, MCSWEENEY S, O'SHEA D, et al. Detecting link fabrication attacks in software-defined networks[C]//Proceedings of 26th International Conference on Computer Communication and Networks. 2017: 1-8.
- [18] GHANNAM R, CHUNG A. Handling malicious switches in software defined networks[C]//Proceedings of Network Operations & Management Symposium. 2016: 1245-1248.
- [19] OIKONOMOU N V, MARGARITI S V, STERGIOU E, et al. Performance evaluation of software-defined networking implemented on various network topologies[C]//Proceedings of 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference. 2021: 1-6.
- [20] KAUR N, SINGH A K, KUMAR N, et al. Performance impact of topology poisoning attack in SDN and its countermeasure[C]//Proceedings of the 10th International Conference on Security of Information and Networks. 2017: 179-184.
- [21] PAKZAD F, PORTMANN M, TAN W L, et al. Efficient topology discovery in OpenFlow-based software defined networks[C]//Proceedings of 8th International Conference on Signal Processing and Communication Systems. 2014.
- [22] AZZOUNI A, BOUTABA R, TRANG N, et al. sOFTDP: secure and efficient topology discovery protocol for SDN[C]//Proceedings of IEEE/IFIP Network Operations and Management Symposium. 2018: 1-7.
- [23] NEHRA A, TRIPATHI M, GAUR M S, et al. SLDP: a secure and lightweight link discovery protocol for software defined networking[J]. Computer Networks, 2018, 150(2): 102-116.
- [24] ZHAO X, YAO L, WU G. ESLD: an efficient and secure link discovery scheme for software-defined networking[J]. International Journal of Communication Systems, 2018, 31(10): 18.
- [25] HAO D. Research and implementation of OFDP based on multi-purpose address broadcast algorithm[D]. Kunming: University of Yun Nan, 2018.
- [26] JIA Y, XU L, YANG Y, et al. Lightweight automatic discovery protocol for OpenFlow-based software defined networking[J]. IEEE Communications Letters, 2020, 24(2): 312-315.
- [27] DONG S, ZHANG A L, HUANG D Y, et al. Optimization of link layer topology discovery in SDN based on OpenFlow[J]. Fire Control & Command Control. 2020: 45(305):150-155.
- [28] WANG Z Q. Research on topology discovery and link delay measurement of SDN network[D]. Huhehaote: University of Inner Mongolia, 2020.
- [29] CHANG Y C, LIN H T, CHU H M, et al. Efficient topology discovery for software-defined networks[J]. IEEE Transactions on Network and Service Management, 2020, (99): 1375-1388.
- [30] OCHOA-ADAY L, CERVELLÓ-PASTOR C, FERNÁNDEZ-FERNÁNDEZ A. eTDP: enhanced topology discovery protocol for software-defined networks[J]. IEEE Access, 2019, 7: 23471-23487.
- [31] TARNARAS G, ATHANASIOU F, DENAZIS S. An efficient topology discovery algorithm for software-defined networks[J]. IET Networks, 2017, 6(6): 157-161.
- [32] OCHOA-ADAY L, CERVELLÓ-PASTOR C, FERNÁNDEZ-FERNÁNDEZ A. Self-healing topology discovery protocol for software defined networks[J]. IEEE Communications Letters, 2018, 3(3): 1-4.
- [33] ROJAS E, ALVAREZ-HORCAJO J, MARTINEZ-YELMO I, et al. TEDP: an enhanced topology discovery service for software-defined networking[J]. IEEE Commun Lett, 2018, 22(8): 1540-1543.
- [34] ALJERI N, BOUKERCHE A. EDiPSo: an efficient scalable topology discovery protocol for software-defined vehicular network[J]. Computer Networks, 2021: 108342.
- [35] HONG S, XU L, WANG H, et al. Poisoning network visibility in software-defined networks: new attacks and countermeasures[C]//The Network and Distributed System Security Symposium(NDSS). 2015: 8-11.
- [36] XIANG S, ZHU H, WU X, et al. Modeling and verifying the topology discovery mechanism of OpenFlow controllers in software-defined networks using process algebra[J]. Science of Computer Programming, 2019, 187: 102343.
- [37] SKOWYRA R, XU L, GU G et al. Effective topology tampering attacks and defenses in software-defined networks[C]//International Conference on Dependable Systems and Networks(DSN). 2018: 374-385.
- [38] ALIMOHAMMADIFAR A, MAJUMDAR S, MADI T, et al. Stealthy probing-based verification (SPV): an active approach to defending software defined networks against topology poisoning attacks[C]//Proceedings of European Symposium on Research in Computer Security (ESORICS). 2018: 463-484.
- [39] MARIN E, BUCCIOL N, CONTI M. An in-depth look into SDN topology discovery mechanisms: novel attacks and practical countermeasures[C]//ACM Conference on Computer and Communications Security(CCS). 2019: 1101-1114.
- [40] DHAWAN M, PODDAR R, MAHAJAN K. SPHINX: detecting security attacks in software-defined networks[C]//Proceedings of the Network and Distributed System Security Symposium(NDSS). 2015: 1-15.

- [41] SMYTH D, MCSWEENEY S, O'SHEA D, et al. Detecting link fabrication attacks in software-defined networks[C]//Proceedings of IEEE 26th International Conference on Computer Communication and Networks (ICCCN). 2017: 1-8.
- [42] KAMISIŃSKI A, FUNG C. FlowMon: detecting malicious switches in software-defined networks[C]//Proceedings of ACM CCS workshop on Automated Decision Making for Active Cyber Defense. 2015: 39-45.
- [43] HUANG X, SHI P, LIU Y, et al. Towards trusted and efficient SDN topology discovery: a lightweight topology verification scheme[J]. Computer Networks, 2020, 170: 107119.
- [44] WANG J, TAN Y, LIU J. Topology poisoning attacks and countermeasures in SDN-enabled vehicular networks[C]//Proceedings of IEEE Global Communications Conference(GLOBECOM). 2020: 1-6.
- [45] ZHENG Z, XU M W, LI Q, et al. Defending against SDN network topology poisoning attacks[J]. Computer Research and Development. 2018, 55(1): 207-215.
- [46] GU Y P, LI D, YU J Q. Im-OFDP: an improved openflow-based topology discovery protocol for software defined network[C]//Proceedings of IFIP International Conferences on Networking. 2020: 628-630.
- [47] SOLTANI S, SHOJAFAR M, MOSTAFAEI H, et al. Real-time link verification in software-defined networks[J]. IEEE Transactions on Network and Service Management, 2023, 20(3): 3596-3611.
- [48] KONG D. Combination attacks and defenses on SDN topology discovery[J]. IEEE/ACM Transactions on Networking, 2023, 31(2): 904-919.
- [49] GAO Y, XU M. Defense against software-defined network topology poisoning attacks[J]. Tsinghua Science and Technology, 2023, 28(1): 39-46.
- [50] ZHANG H L, XIE Q M, LI W. Research on minimum convex covering problem[J]. China New Telecommunications, 2014, (13): 51-52.

[作者简介]



李冬（1979—），男，博士，湖北枣阳人，华中科技大学高级工程师，主要研究方向为计算机网络、软件定义网络、网络安全。



于俊清（1975—），男，博士，内蒙古赤峰人，华中科技大学教授、博士生导师，主要研究方向为数字媒体处理与检索、网络安全、多核计算与流编译。



谷永普（1995—），男，湖南永州人，华中科技大学硕士生，主要研究方向为网络安全、软件定义网络。



赵鹏程（1996—），男，湖北武穴人，华中科技大学硕士生，主要研究方向为网络安全、软件定义网络。