Journal of Computer Applications
计算机应用,2015,35(11):3270-3274,3307
ISSN 1001-9081
CODEN JYIIDU
2015-11-10
http://www.joca.cn

文章编号:1001-9081(2015)11-3270-05
doi:10.11772/j.issn.1001-9081.2015.11.3270

面向 OpenFlow 网络的访问控制规则自动实施方案

刘　艺[1,2*],张红旗[1,2],代向东[1,2],雷　程[1,2]
(1.信息工程大学,郑州 450001;　2.河南省信息安全重点实验室,郑州 450001)
(*通信作者电子邮箱 liuyi9582@126.com)

摘　要:针对 OpenFlow 网络数据平面频繁改变导致网络难以实时满足访问控制策略要求的问题,提出了面向 OpenFlow 网络的访问控制规则自动实施方案。首先,由实时构建的转发路径获得可达空间,并通过规则集动态合成算法消除访问控制规则间的冲突;之后,采用规则空间分割算法将合成后访问控制规则的拒绝空间与可达空间比较,以检测直接和间接违反访问控制规则的非法转发路径;在此基础上,结合网络更新事件与违反检测结果灵活采取自动的违反解决方法,包括规则更新拒绝、规则序列移除、基于线性规划(LP)的近源端规则部署和末端规则部署4种;最后转换访问控制规则形式。理论分析和仿真结果表明,方案可用于控制器上运行多个安全应用程序和交换机内存受限的情况,并且基于 LP 的近源端规则部署方法可以降低网络中的不期望流量。

关键词:转发路径;线性规划;网络数据平面;访问控制规则;OpenFlow 网络
中图分类号:TP393.08　　**文献标志码**:A

Automatic implementation scheme of implementing access control rules in OpenFlow network

LIU Yi[1,2*], ZHANG Hongqi[1,2], DAI Xiangdong[1,2], LEI Cheng[1,2]
(1. *Information Engineering University, Zhengzhou Henan* 450001, *China*;
2. *Henan Key Laboratory of Information Security, Zhengzhou Henan* 450001, *China*)

Abstract: Focusing on the issue that OpenFlow network can't meet access control policy constantly resulted from its data plane changing frequently, an automatic implementation scheme of implementing access control rules in OpenFlow network was proposed. Firstly, reachable space was obtained by building real-time forwarding paths, and conflicts among access control rules were resolved by using dynamical synthesis algorithm. Then, denied space was extracted from synthetic set of access control rules by using rule space division algorithm, which was compared with reachable space subsequently to detect direct and indirect violations. According to network update situations and violation detection results, automatic violation resolutions were adopted flexibly, such as rejecting rule update, removing rule sequence, deploying rule near source based on Linear Programming (LP) and deploying rule terminally. Lastly, the format of access control rule was converted. The theoretical analysis and simulation results demonstrate that the proposed scheme is applicable under the condition that multiple security applications are running on the controller and memory of switch is limited, and show that deploying rule near source based on LP can minimize unwanted traffic of network.

Key words: forwarding path; Linear Programming (LP); network data plane; access control rule; OpenFlow network

0　引言

近年来,软件定义网络(Software-Defined Networking, SDN)[1]逐渐成为学术界和业界的研究热点,基于 OpenFlow[2]实现 SDN 是目前主流趋势[3]。在 OpenFlow 网络中,运行在控制器上的流量管理、负载平衡、动态路由等应用程序通过向交换机中的流表插入、修改和删除规则来决定数据包在网络中的转发路径。然而,当多个应用程序或用户同时对同一物理网络进行操作时,它们可能因为彼此存在竞争、重写等关系而产生相冲突的规则,使得转发路径违背访问控制策略。因此,正确部署访问控制规则十分重要,其面临3个挑战:1)不同安全应用程序生成的访问控制规则之间可能存在冲突;2) OpenFlow 标准[4]允许在转发过程中修改数据包包头,导致转发路径间接违反访问控制策略,即动态流隧道[5]问题;3)网络数据平面的改变速率可达每秒上千次[6],需要及时调整访问控制规则。

由此,本文提出了一种面向 OpenFlow 网络的访问控制规则自动实施方案,它可以快速检测出直接和间接违反访问控制规则的非法转发路径,针对不同的违反情况采取不同的解决方法,达到最小化不期望流量[7]、减少所需安装规则数目等目的,并且在网络动态变化时自动部署访问控制规则,保证网络转发路径满足访问控制策略要求。

1　相关工作

随着 OpenFlow 网络的广泛部署,其安全性问题越来越受到重视。以访问控制策略为代表的安全策略作为传统网络安

收稿日期:2015-05-29;**修回日期**:2015-07-02。
基金项目:国家863计划项目(2012AA012704);郑州市科技领军人才项目(131PLJRC644)。
作者简介:刘艺(1991-),女,江西崇义人,硕士研究生,主要研究方向:信息安全、软件定义网络安全;　张红旗(1962-),男,河北唐山人,教授,博士生导师,博士,主要研究方向:网络安全、等级保护;　代向东(1977-),男,四川仁寿人,讲师,硕士,主要研究方向:安全策略管理;雷程(1989-),男,北京海淀人,硕士研究生,主要研究方向:网络与信息安全。

全保护的重要因素,在 OpenFlow 网络中同样必不可少。已有研究主要围绕实施和验证访问控制策略两方面展开。

对于实施访问控制策略,主要通过检测和消解访问控制策略与数据包转发策略之间的冲突来实现。Pyretic[8] 将这两类策略翻译为有序的规则集合,通过规则的串行操作解决转发策略直接违反访问控制策略的情况,但无法发现并解决由包头修改带来的间接违反问题。为解决这一问题,FortNOX[5] 设置别名集合记录规则关系,并逐一比较新规则与访问控制策略,但未考虑同一流表内和不同访问控制规则间的规则依赖关系,易造成误判,而且需要管理员参与部署访问控制规则。FLOWGUARD[9] 以全网转发路径为基础,能够在动态网络中准确检测和灵活解决转发路径违反访问控制策略的问题,但未考虑多个安全应用程序共同管理网络的情况,且规则部署方式不够优化。

对于验证访问控制策略,主要由网络属性验证工具实现,如 FlowChecker[10]、Anteater[11]、Hassel[12]、Veriflow[13] 和 NetPlumber[6],它们通过实时或非实时从网络数据平面获取与分析规则,验证转发路径是否满足访问控制策略,但不支持自动的、有效的违反解决方法。

综上所述,针对在 OpenFlow 网络中自动实施访问控制规则问题,上述研究各自存在不足,比如未协调多个安全应用程序各自生成的访问控制策略,规则部署方式有待优化,未提供自动的违反解决方法等。因此需要一种能够在动态 OpenFlow 网络中准确且快速地检测所有违反访问控制规则的转发路径,自动且高效地部署访问控制规则的方案,保证网络安全。

2　访问控制规则自动实施方案

本文将 OpenFlow 网络中的规则分为两类:转发规则,由负载均衡等与安全无关的应用程序生成,遵循 OpenFlow 标准格式,可以直接安装到交换机上;访问控制规则,由防火墙、入侵检测系统等安全应用程序生成,部署位置确定后需要转换为 OpenFlow 标准格式安装到相应交换机上。

依据 OpenFlow1.0 标准,转发规则 $r:=(Inport,C,pri,a)$,其中:$Inport$ 具有全网唯一性,表示规则对从特定端口进入其所在交换机的数据包起作用;$C=Pro\times IP_{src}\times Port_{src}\times IP_{dst}\times Port_{dst}\times Others$;$pri$ 是规则优先级;动作集合 $A=\{fwd,drop,set_{a\to b}\}$。一般的,访问控制规则 $r:=(C,pri,a)$,其中 $C=Pro\times IP_{src}\times Port_{src}\times IP_{dst}\times Port_{dst}\times Feature$;动作集合 $A=\{allow,deny\}$。假设 $Feature$ 与 $Others$ 所含字段相同。

面向 OpenFlow 网络的访问控制规则自动实施方案分为规则预处理、违反检测、违反解决和访问控制规则形式转换 4 阶段,如图 1 所示。首先,在转发规则和访问控制规则下发到交换机前,由转发规则构建实时转发路径,提取可达空间,并动态合成访问控制规则集,之后,基于可达空间和访问控制规则空间分割结果检测非法转发路径,随后,以违反检测结果为依据采用不同的违反解决方法,最后将访问控制规则转换为 OpenFlow 标准格式,安装到交换机上。

2.1　规则预处理

2.1.1　实时转发路径构建

数据包在 OpenFlow 网络中转发时,交换机可通过 set 操作改变包头字段,如源/目的 IP 地址、源/目的端口等,使得非法流绕过访问控制规则的检查而在网络中传播,即动态流隧道问题。如图 2 所示,网络包括 3 个主机和 2 个 OpenFlow 交换机。假设安全应用程序生成访问控制规则,规定主机 A(192.168.17.11)允许访问主机 B(192.168.17.12),但不允许访问主机 C(192.168.17.13),并安装在交换机 s_1 上,如表 1 所示,若交换机 s_2 上存在两条由 GOTO TABLE 指令链接的转发规则,使得所有从 192.168.17.11 发往 192.168.17.12 的数据包的目的 IP 地址改为 192.168.17.13,直接交付给主机 C,则当主机 A 发送源 IP 地址为 192.168.17.11、目的 IP 地址为 192.168.17.12 的数据包时,其最终可到达主机 C,这与访问控制规则相背。因此,在部署访问控制规则前需要获取完整的转发路径。本文采用网络属性验证工具 NetPlumber 实时获取所有转发路径,提取违反检测所需信息。

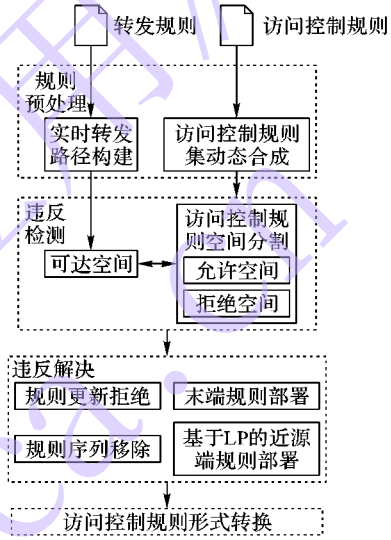

图 1　实施方案整体架构

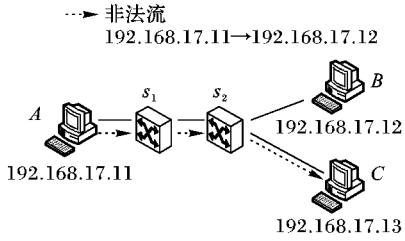

图 2　动态流隧道

表 1　规则信息

交换机	规则	
s_1	192.168.17.11→192.168.17.12	forward
	192.168.17.11→192.168.17.13	drop
s_2	192.168.17.11→192.168.17.12	set dst = 192.168.17.13
	*→192.168.17.13	forward

在 NetPlumber 中,对于每条规则,头空间 $P_r:=C\to\{0,1,x\}^L$ 表示其作用范围,可进行交(∩)、并(∪)、差(-)和补运算。为便于叙述,记源地址空间 $P^s:=\{Pro\times IP_{src}\times Port_{src}\}\to\{0,1,x\}^L$,目的地址空间 $P^d:=\{IP_{dst}\times Port_{dst}\}\to\{0,1,x\}^L$,特征空间 $P^o:=\{Others\}\to\{0,1,x\}^L$,由此,$P_r$ 可另记为 $\{P^s,P^d,P^o\}$。转发路径由规则序列 $r^{seq}=[r_0,r_1,\cdots,r_n]$ 构成,本文从中提取 3 类空间:源空间 $P_{src}=P_{r_0}:=\{P_0^s,P_0^d,P_0^o\}$ 表示初始流信息;目的空间 $P_{dst}=P_{r_n}:=\{P_n^s,P_n^d,P_n^o\}$ 表示经网络转

发后的最终流信息，由于规则存在 *set* 动作，P_{src} 与 P_{dst} 不一定相同；可达空间 $P_{pass} := \{P_0^s, P_n^d, P_n^o\}$ 表示实际传输的流信息，用于检测转发路径是否违反访问控制规则。

2.1.2 访问控制规则集动态合成

在 OpenFlow 网络中控制器上不同的安全应用程序依照自身策略生成各自的访问控制规则列表，称为“成员规则集”，它们之间可能存在冲突，因此需要合成为“合成规则集”，对流实施统一的访问控制。此外，安全应用程序会动态更新自身的规则列表，即当成员规则集变化时，规则合成算法应增量式更新合成规则集，尽可能小地改变已有规则，减少时间开销。

记成员规则集为 M 和 N，合成规则集为 R，规则 $m_i \in M$，$n_j \in N$，$r_k \in R$。记规则 r 优先级为 pri_r，规定：pri_r 越大，r 的优先级越高，且不同安全应用程序间的规则优先级设置互不影响，同一安全应用程序内的规则优先级具有唯一性。

定义 1 规则动作合成操作符 $+_a$。假设 $a_1 \in A, a_2 \in A$，有 $allow +_a allow = allow$，$a_1 +_a deny = deny, deny +_a a_2 = deny$。

性质 1 $+_a$ 满足交换率和结合率。

定义 2 规则合成操作符 $+_r$。$r_k = m_i +_r n_j$，满足 $P_{r_k} = P_{m_i} \cap P_{n_j}, a_{r_k} = a_{m_i} +_a a_{n_j}, pri_{r_k} = pri_{m_i} + pri_{n_j}$。

性质 2 $+_r$ 满足交换率和结合率。

定义 3 规则集合成操作符 $+_R$。$R = M +_R N$ 的构成算法如下：对于所有 $m_i \in M, r_k = m_i$；对于所有 $n_j \in N, r_k = n_j$；对于所有 $(m_i, n_j) \in M \times N$，若 $P_{m_i} \cap P_{n_j} \neq \varnothing$，则 $r_k = m_i +_r n_j$。

性质 3 $+_R$ 满足交换率和结合率。

性质 4 $(\bigcup_i P_{m_i}) \cup (\bigcup_j P_{n_j}) = \bigcup_k P_{r_k}$。

性质 1、2、3 保证了规则(集)合成算法与合成顺序无关，性质 4 保证了规则集合成算法不改变成员规则集的作用范围。

定理 1 设流的头空间为 h，$h \cap P_{m_i} \neq \varnothing$，且对于 $\forall i' \neq i, m_{i'} \in M$，若 $h \cap P_{m_{i'}} \neq \varnothing$，则 $pri_{m_{i'}} < pri_{m_i}$。同样的，$h \cap P_{n_j} \neq \varnothing$，且对于 $\forall j' \neq j, n_{j'} \in M$，若 $h \cap P_{n_{j'}} \neq \varnothing$，则 $pri_{n_{j'}} < pri_{n_j}$。对于合成规则集 $R = M +_R N$，假设 $\exists r_k \in R, r_k = m_i +_r n_j$，则 $h \cap P_{r_k} \neq \varnothing$，且对于 $\forall k' \neq k, r_{k'} \in R$，若 $(h \cap P_{r_k}) \cap P_{r_{k'}} \neq \varnothing$，则 $pri_{r_{k'}} < pri_{r_k}$。

证明 假设 $\exists k' \neq k, r_{k'} \in R, (h \cap P_{r_k}) \cap P_{r_{k'}} \neq \varnothing$，$pri_{r_{k'}} \geqslant pri_{r_k}$，若 $r_{k'} = m_{i'} +_r n_{j'}$，则 $h \cap P_{m_{i'}} \neq \varnothing, h \cap P_{n_{j'}} \neq \varnothing$。由式(1)和(2)

$$pri_{r_k} = pri_{m_i} + pri_{n_j} \tag{1}$$

$$pri_{r'_k} = pri_{m'_i} + pri_{n'_j} \tag{2}$$

可得式(3)和式(4)至少有一个成立：

$$pri_{m_{i'}} \geqslant pri_{m_i} \tag{3}$$

$$pri_{n_{j'}} \geqslant pri_{n_j} \tag{4}$$

若式(3)成立，与 $\forall i' \neq i, m_{i'} \in M$，若 $h \cap P_{m_{i'}} \neq \varnothing$，则 $pri_{m_{i'}} < pri_{m_i}$ 相矛盾；若式(4)成立，与 $\forall j' \neq j, n_{j'} \in M$，若 $h \cap P_{n_{j'}} \neq \varnothing$，则 $pri_{n_{j'}} < pri_{n_j}$ 相矛盾。故定理 1 成立。

定理 1 表明，只要成员规则集中不存在匹配相同数据包且优先级相同的规则，那么合成规则集对数据包的动作是明确的，即匹配同一数据包的多条规则中必存在唯一的最高优先级规则。此外，当成员规则集增加规则时，只需将新的规则与其他成员规则集中的规则进行合成，并不改变原有合成规则集中的规则信息，当成员规则集删除规则时，只需将原有合成规则集中的相关规则删除，对其他规则无影响。

2.2 违反检测

由于合成规则集的不同规则头空间仍然存在覆盖关系，每次违反检测时都需要按照优先级大小顺序比较，效率低，因此本文采用规则空间分割算法，按照规则动作将合成后的访问控制规则集转换为一组不相关的空间，即允许空间 $P_a = \{p_1, p_2, \cdots, p_n\}$ 和拒绝空间 $P_d = \{p_1', p_2', \cdots, p_m'\}$，对于 $\forall p_i \in P_a, \forall p_j' \in P_d, p_i \cap p_j' = \varnothing$。由于允许空间与拒绝空间相互独立，违反检测时无需考虑优先级。此外，允许空间和拒绝空间各自内部可能存在空间覆盖，可采用卡诺图和 Quine-McCluskey算法进行合并，减少离散空间数目，加快检测速度。

算法 1 规则空间分割算法。

```
foreach r ∈ R do                          /* 按 pri_r 从大到小的顺序 */
  if r.a = deny then
    if ∃p ∈ P_a, p ∩ P_r ≠ ∅ then P_r = P_r − p;
    end if
    P_d.add(P_r);
  else if r.a = allow then
    if ∃p ∈ P_d, p ∩ P_r ≠ ∅ then P_r = P_r − p;
    end if
    P_a.add(P_r);
  end if
end for
return P_d, P_a;
```

将访问控制规则的拒绝空间 P_d 和转发路径的可达空间 P_{pass} 进行比较，违反情况分两种：

1)完全违反：$P_{pass} \subseteq P_d$；

2)部分违反：$P_d \cap P_{pass} \neq \varnothing$，且 $P_d \not\subset P_{pass}, P_{pass} \not\subset P_d$。这存在两种可能：若非法转发路径上存在规则 r^* 满足 $r^*.a = set(P_{r^*} \rightarrow H^*), H^* \cap P_d \neq \varnothing$，同时，未部署访问控制规则阻止头空间为 $set^{-1}(H^* \cap P_d)$ 的流，或者阻止该流的规则在 r^* 之后，称为间接部分违反；若非法转发路径上不存在这样的 r^*，表明未部署访问控制规则阻止头空间为 $P_d \cap P_{pass}$ 的流，称为直接部分违反。

2.3 违反解决

OpenFlow 网络具备集中控制特性和细粒度网络控制[9]，控制器可统一安装规则，灵活选择访问控制规则的部署位置。因此，基于网络更新事件和违反检测结果，本文设计了灵活的违反解决方法，如图 3 所示。

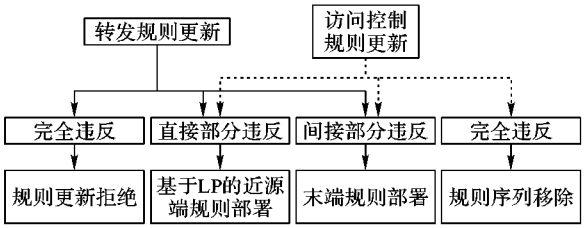

图 3 违反解决方法

1)规则更新拒绝。

它适用于由转发规则更新导致的完全违反情况，控制器

拒绝向交换机下发转发规则更新报文,并向生成该报文的应用程序返回更新拒绝消息。

2)基于 LP 的近源端规则部署。

它适用于由转发规则或访问控制规则更新导致的直接部分违反情况。当非法转发路径直接部分违反访问控制规则时,可在该路径上的任意交换机中安装 deny 规则阻断非法流。如图 4,假设访问控制策略规定主机 H_1、H_2、H_3 都不允许访问主机 H_4,则有两种方式实现:一是在交换机 s_2 上部署访问控制规则禁止从 H_1、H_2、H_3 到 H_4 的流;二是分别在交换机 s_1、s_2、s_3 上部署访问控制规则禁止从 H_1 到 H_4、H_2 到 H_4、H_3 到 H_4 的流。显然,第一种方式所要插入的规则数目少于第二种方式,而后者能够减小网络中不期望的流量,即阻止不允许的流(从 H_1 到 H_4 和从 H_3 到 H_4)通过部分网络,这对防止网络遭受 DoS 攻击十分重要。因此,结合二者的优势,本文采用线性规划方法,在给定交换机内存限制条件下,以最小化不期望流量为目标,综合衡量确定 deny 规则的部署位置。

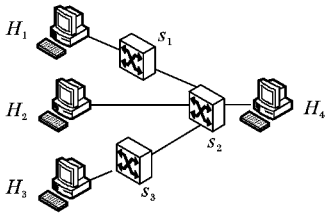

图 4　示例网络

假设交换机 $s_i(i \leqslant m)$ 上最多安装 c_i 条访问控制规则,拒绝空间 $P_d = \bigcup_{j \leqslant n} P_{d_j}$。若转发路径 k 的可达空间满足 $P_{\text{pass}} \cap P_{d_j} \neq \varnothing$,则该路径对应的交换机序列记为 $S_k^{Seq_j} = \{s_1, s_2, \cdots, s_{l_{j,k}}\}$,长度 $l_{j,k} = \| S_k^{Seq_j} \|$。由此,所有可达空间满足 $P_{\text{pass}} \cap P_{d_j} \neq \varnothing$ 的转发路径对应的交换机序列组成集合 $Seq_j = \{S_1^{Seq_j}, S_2^{Seq_j}, \cdots, S_{t_j}^{Seq_j}\}$,$t_j$ 为违反 P_{d_j} 的交换机序列的总数。$\omega_{i,j,k}$ 是 s_i 在 $S_k^{Seq_j}$ 中的位置,比如,若 $S_2^{Seq_1} = \{s_2, s_3, s_5\}$,则 $\omega_{3,1,2} = 2$。$\alpha_{i,j}$ 是 s_i 在 Seq_j 包含的交换机序列中出现的次数。二元变量 $x_{i,j} = 1(i \leqslant m, j \leqslant n)$ 当且仅当在 s_i 上安装规则阻止 P_{d_j} 对应的流,否则 $x_{i,j} = 0$。

$$\max: \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{t_j} \frac{l_{j,k} - \omega_{i,k} + 1}{l_{j,k}} x_{i,j} \tag{5}$$

$$\text{s.t.} \quad \forall_{i \leqslant m}: \sum_{j=1}^{n} x_{i,j} \leqslant c_i \tag{6}$$

$$\forall_{j \leqslant n}: \sum_{i=1}^{m} \alpha_{i,j} \cdot x_{i,j} \geqslant t_j \tag{7}$$

目标函数式(5)表示尽可能在非法转发路径的源头安装 deny 规则,以最小化不期望流量。限制条件式(6)表示每个交换机上安装的规则数不能超出其额定数目,式(7)表示所有违反 P_{d_j} 的转发路径上至少需要安装一条 deny 规则来阻止该流。

3)末端规则部署。

它适用于由转发规则或访问控制规则更新导致的间接部分违反情况。由于流在网络转发过程中可能经历多次改变,难以区分合法改变与非法改变,以及每次改变后的合法流与非法流,而且访问控制规则的部署位置不当会使得规则失效,导致动态流隧道问题,故从降低规则部署复杂度方面考虑,在转发路径末端交换机上部署访问控制规则。假设非法转发路径的交换机序列 $S = \{s_1, s_2, \cdots, s_l\}$,则在 s_l 上安装两条 deny 规则分别阻止 P_{d_j} 和 P_{dst} 对应的流,其中 P_{dst} 是该路径的目的空间。

4)规则序列移除。

它适用于由访问控制规则更新导致的完全违反情况,所有与非法转发路径相关的规则都将被移除。

2.4　访问控制规则形式转换

对于近源端和末端规则部署方法,在确定了访问控制规则部署位置后,需要将规则头空间表示形式转换为 Openflow 标准规则形式 $r := (Inport, C, pri, drop)$,安装到相应交换机上,其中 pri 通常为最高优先级。

2.5　时间复杂度分析

在网络新增一条转发规则和访问控制规则的情况下,分阶段分析方案的时间复杂度。在规则预处理阶段,实时转发路径构建的时间复杂度为 $O(t + spd)$[6],其中:t 是平均每张流表所含规则数;s 是管道图中附加的探针数,通常与网络入口交换机数目成比例;p 是管道图中每个规则节点的平均出边和入边总数,由于规则的匹配域通常聚合多个子网[14],p 较小;d 是网络直径。访问控制规则集动态合成的时间复杂度在最坏情况下为 $O(mn)$,此时新增访问控制规则需要与所有成员规则集(除自身外)中的规则进行合成,其中 m 是成员规则集的平均规则数,n 是运行在控制器上的安全应用程序数,通常较小。在违反检测阶段,假设已有 r 条合成规则,由于规则空间分割算法对规则的匹配域进行合并,允许或拒绝空间数目小于规则数,故分割新增合成规则头空间和检测新增可达空间的时间复杂度都小于 $O(r)$。在违反解决阶段,规则更新拒绝、末端规则部署和规则序列移除三种方式无需计算,时间复杂度可视为 $O(1)$,基于 LP 的近源端规则部署方法的本质是在非法交换机序列中选择交换机部署规则,搜索空间与该序列的长度相关,在最坏情况下它等于网络直径 d,时间复杂度为 $O(d)$。在访问控制规则形式转换阶段,时间复杂度为 $O(1)$。

3　仿真实验与评估

3.1　实验环境

仿真实验在 Mininet 上进行,开源的 Floodlight 作为控制器,负责安装规则到 Open VSwitch(OVS)交换机上,为与 FLOWGUARD[9] 进行性能对比,实验网络 T1 ~ T6 均采用 Stanford 主干网[15] 拓扑,包含 14 个运行区 Cisco 路由器,10 个以太网交换机和 2 个主干 Cisco 路由器,但它们所含的转发规则总数不相同,如表 2 所示。

表 2　各实验网络转发规则数目

网络编号	转发规则总数	网络编号	转发规则总数
T1	8 900	T4	24 500
T2	14 100	T5	29 700
T3	19 300	T6	34 900

3.2　实验结果分析

仿真实验在控制器上安装代理获取非安全应用程序生成的转发规则和安全应用程序生成的访问控制规则,并将其转换为各自的标准格式(如第 2 章所述),利用 NetPlumber 构建全网转发路径并提取违反检测信息,同时合成访问控制规则

并划分空间，当转发路径的可达空间与访问控制规则的拒绝空间相重叠时，分析违反类型，采取相应解决方案。此外，在方案运行结束后，利用 NetPlumber 的网络属性验证功能检验网络中是否存在非法转发路径。

1）方案效率。

实验采用网络 T6，先随机选取 80% 转发规则插入到交换机中构成初始网络，再由非安全应用程序将剩下的 20% 转发规则经由控制器下发到交换机，同时两个安全应用程序按照管理员制定的策略生成访问控制规则交付给控制器。由于 FLOWGUARD 没有提供访问控制规则合成算法，未考虑交换机内存受限，为进行方案效率对比，实验将合成后的访问控制规则作为 FLOWGUARD 的输入，且不限制交换机能够安装的规则数目，分阶段统计两种方案的时间开销，如表 3 所示。

表 3 方案各阶段时间开销统计表

阶段		本文方案	FLOWGUARD
转发路径构建	平均值/ms	0.300	0.320
	中值/ms	0.075	0.107
访问控制规则合成	平均值/ms	0.03	—
	中值/ms	0.02	—
违反检测	平均值/ms	0.06	0.05
	中值/ms	0.08	0.07
违反解决	平均值/ms	3.82	3.67
	中值/ms	2.01	3.01

实验结果表明，在两种方案运行结束后，网络中不存在非法转发路径。方案的主要时间开销在转发路径构建阶段和违反解决阶段。对于前者，可以采用分布式 NetPlumber 提高构建速度。对于后者，本文方案与 FLOWGUARD 的时间开销相当，但后者未区分直接和间接部分违反，只在入口和出口交换机上安装规则来解决部分违反情况，而本文提出的基于 LP 的近源端规则部署方法考虑了交换机内存限制，更加符合实际情况，而且可以减小不期望流量，对提高网络性能，防止 DoS 攻击十分重要。此外末端规则部署可以解决间接部分违反情况，但文献[9]对此没有提供明确方案。

2）方案可扩展性。

实验将从[15]中获得的实际访问控制规则转换为标准格式，分别采用本文方案和 FLOWGUARD 依次在网络 T1 ~ T6 中部署。因为违反检测时间对实施访问控制规则总时间影响小，因此只检测各网络中转发路径构建和违反解决时间，实验结果如图 5 所示。

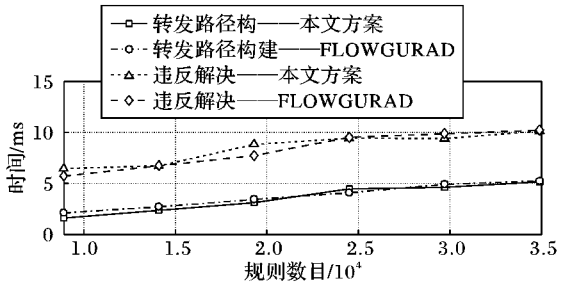

图 5 各实验网络时间开销对比

本文方案的转发路径构建和违反解决时间都与 FLOWGUARD 相当，且二者都随转发规则数目的增加而增加，增加速率趋于平缓，但当网络出现部分违反情况时，本文方案提供了更加实用的解决方法。此外与表 3 相比，本实验中的转发路径构建时间长，这是因为它们是 NetPlumber 的初始化时间，而在初始化完成后，当转发规则动态变化时 NetPlumber 能增量更新转发路径信息，大大提高构建速度。

3）降低网络不期望流量实验。

为了评估基于 LP 的近源端规则部署算法对网络中不期望流量的作用，采用累积分布函数描述不期望流量的作用范围，比如 $F(0.18) = 0.95$ 表示 95% 的不期望流量的作用范围为非法转发路径的前 18% 长度，即它们在经过非法转发路径的前 18% 长度之中被阻断。实验在网络 T1 上进行，采用 ClassBench 生成 2 000 条访问控制规则，假设每个交换机至多能额外安装 100 条访问控制规则，检测不期望流量，结果如图 6 所示。

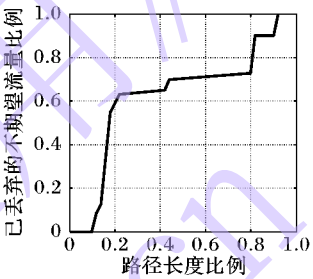

图 6 不期望流量实验结果

由图可知，在一半路径长度前能丢弃大约 80% 不期望流量，而且超过 60% 不期望流量会在路径 20% 长度前被丢弃。此外由 $F(0.92) \approx 1$ 可知所有不期望流量都会被最终丢弃。

4 结语

为保证动态 OpenFlow 网络时时满足访问控制策略的要求，本文提出了一种面向 OpenFlow 网络的访问控制规则自动实施方案。在规则预处理阶段，针对数据包包头在转发过程中发生改变导致访问控制规则失效的问题，采用 NetPlumber 构建实时转发路径，获取可达空间。此外，针对访问控制规则间存在冲突的问题，提出了规则动态合成算法，并证明了其正确性。在违反检测阶段，利用规则空间分割算法提取拒绝空间，通过空间比较得到所有直接和间接违反访问控制规则的非法转发路径，提高了检测的速率和准确性。在违反解决阶段，设计了 4 种自动的违反解决方法，其中基于 LP 的近源端规则部署和末端规则部署方法利用了 OpenFlow 网络的细粒度网络控制，提高了规则部署的灵活性和有效性。实验表明，本文方案在提供与已有研究相当的效率和可扩展性情况下，能够用于多安全应用程序和交换机内存受限的场景，并细分了部分违反情况，从减少所需安装的规则数目等方面优化了违反解决方法，而且基于 LP 的近源端规则部署方法可以将 80% 不期望流量阻断在一半路径长度前，有效降低了网络不期望流量。下一步需要研究访问控制规则下发过程，控制器和交换机之间存在时延有可能影响到规则在交换机上的安装次序，导致控制逻辑不一致，网络进入不安全状态。

参考文献：

[1] McKEOWN N. Software-defined networking[J]. International Conference on Computer Communications Keynote Talk, 2009, 17(2): 30 - 32.

（下转第 3307 页）

tershed segmentation algorithm for remote sensing image[D]. Nanjing: Nanjing University of Science and Technology, 2011. (席英. 遥感图像的 K-均值聚类和分水岭分割算法的研究与实现[D]. 南京: 南京理工大学, 2011.)

[16] YI L, ZHANG G, WU Z. A scale-synthesis method for high spatial resolution remote sensing image segmentation[J]. IEEE Transactions on Geoscience and Remote Sensing, 2012, 50(10): 4062 - 4070.

[17] GAO X, ZHANG S. Theory method and application of image segmentation[D]. Changchun: Jilin University, 2006. (高秀娟, 张树功. 图像分割的理论, 方法及应用[D]. 长春: 吉林大学, 2006.)

[18] LONG J, SHEN X, ZANG H, *et al*. An adaptive thresholding algorithm by background estimation in Gaussian scale space[J]. Acta Automatica Sinica, 2014, 40(8): 1773 - 1782. (龙建武, 申铉京, 臧慧, 等. 高斯尺度空间下估计背景的自适应阈值分割算法[J]. 自动化学报, 2014, 40(8): 1773 - 1782.)

[19] SUN Y, CAI Z. An improved character segmentation algorithm based on local adaptive thresholding technique for Chinese NvShu documents[J]. Journal of Networks, 2014, 9(6): 1496 - 1501.

[20] JAIN P, TYAGI V. An adaptive edge-preserving image denoising technique using tetrolet transforms[J]. Visual Computer, 2015, 31(5): 657 - 674.

[21] SAUVOLA J, PIETIK?INEN M. Adaptive document image binarization[J]. Pattern Recognition, 2000, 33(2): 225 - 236.

[22] LOWE D. Object recognition from local scale-invariant features [C]// Proceedings of the 7th IEEE International Conference on Computer Vision. Piscataway: IEEE, 1999: 1150 - 1157.

[23] LOWE D. Distinctive image features from scale-invariant keypoints [J]. International Journal of Computer Vision, 2004, 60(2): 91 - 110.

[24] SHARMA G, CHAUDHURY S, SRIVASTAVA J. Bag-of-features kernel eigen spaces for classification[C]// Proceedings of the 19th International Conference on Pattern Recognition. Piscataway: IEEE, 2008: 1 - 4.

[25] ARTHUR D, VASSILVITSKII S. k-means++: the advantages of careful seeding[C]// Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms Society for Industrial and Applied Mathematics. New York: ACM, 2007: 1027 - 1035.

[26] LAZEBNIK S, SCHMID C, PONCE J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories [C]// Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2006: 2169 - 2178.

[27] HUANG Q, WU G, CHEN J, *et al*. Automated remote sensing image classification method based on FCM and SVM[C]// Proceedings of the 2012 2nd International Conference on Remote Sensing, Environment and Transportation Engineering. Washington, DC: IEEE Computer Society, 2012: 1 - 4.

[28] HE S, WANG L, XIA Y, *et al*. Insulator recognition based on moments invariant features and cascade AdaBoost classifier[C]// Proceedings of the 2013 International Conference on Mechatronics and Control Engineering. Zurich: Trans Tech Publications, 2013: 362 - 367.

(上接第 3274 页)

[2] McKEOWN N, ANDERSON T, BALAKRISHNAN H, *et al*. OpenFlow: enabling innovation in campus networks[J]. ACM Special Interest Group on Data Communication Computer Communication Review, 2008, 38(2): 69 - 74.

[3] ZUO Q, CHEN M, ZHAO G, *et al*. Research on OpenFlow-based SDN technologies[J]. Journal of Software, 2013, 24(5): 1078 - 1097. (左青云, 陈鸣, 赵广松, 等. 基于 Open Flow 的 SDN 技术研究[J]. 软件学报, 2013, 24(5): 1078 - 1097.)

[4] OpenFlow switch consortium. OpenFlow switch specification version 1.0.0[J/OL]. [2015-03-01]. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf.

[5] PORRAS P, SHIN S, YEGNESWARAN V, *et al*. A security enforcement kernel for OpenFlow networks[C]// Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks. New York: ACM, 2012: 121 - 126.

[6] KAZEMIAN P, CHAN M, ZENG H, *et al*. Real time network policy checking using header space analysis[C]// Proceedings of the 10th USENIX Symposium on Networked System Design and Implementation. Berkeley: USENIX, 2013: 99 - 111.

[7] KANG N, LIU Z, REXFORD J, *et al*. Optimizing the one big switch abstraction in software-defined networks[C]// Proceedings of the 9th ACM Conference on Emerging Networking Experiments and Technologies. New York: ACM, 2013: 13 - 24.

[8] MONSANTO C, REICH J, FOSTER N, *et al*. Composing software defined networks [C]// Proceedings of the 10th USENIX Symposium on Networked System Design and Implementation. Berkeley: USENIX, 2013: 1 - 13.

[9] HU H, HAN W, AHN G J, *et al*. FLOWGUARD: building robust firewalls for software-defined networks[C]// Proceedings of the 3rd Workshop on Hot Topics in Software Defined Networking. New York: ACM, 2014: 97 - 102.

[10] AL-SHAER E, AL-HAJ S. FlowChecker: configuration analysis and verification of federated OpenFlow infrastructures [C]// Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration. New York: ACM, 2010: 37 - 44.

[11] MAI H, KHURSHID A, AGARWAL R, *et al*. Debugging the data plane with anteater[J]. ACM Special Interest Group on Data Communication Computer Communication Review, 2011, 41(4): 290 - 301.

[12] KAZEMIAN P, VARGHESE G, McKEOWN N. Header space analysis: static checking for networks[C]// Proceedings of the 9th USENIX Symposium on Networked System Design and Implementation. Berkeley: USENIX, 2012: 113 - 126.

[13] KHURSHID A, ZHOU W, CAESAR M, *et al*. VeriFlow: verifying network-wide invariants in real time[J]. ACM Special Interest Group on Data Communication Computer Communication Review, 2012, 42(4): 467 - 472.

[14] ZENG H, ZHANG S, YE F, *et al*. Libra: divide and conquer to verify forwarding tables in huge networks[C]// Proceedings of the 11th USENIX Symposium on Networked System Design and Implementation. Berkeley: USENIX, 2014, 14: 87 - 99.

[15] Header space library and NetPlumber [CP/OL]. [2015-03-04] https://bitbucket.org/peymank/hassel-public/