软件定义网络流的安全要素知识图谱研究

游瑞邦[1,2], 袁子牧[1*], 涂碧波[1,2], 孟 丹[1,2]

[1]中国科学院信息工程研究所, 北京 中国 100093
[2]中国科学院大学 网络空间安全学院, 北京 中国 100049

摘要 流表是软件定义网络控制平面与数据平面交互的核心组件, 也是实现安全策略全局协同及动态映射的关键。然而, 构建具备相关安全策略的流表却需应对流知识要素过于分散、不断扩充、难以通过独立应用或预设规则满足等诸多难点。针对这一现状问题, 本文通过采取在软件定义网络控制、数据和应用等三大平面之外新建知识平面的方式, 构建流表及其相关安全知识要素聚集的流知识图谱, 并基于此选择或生成流表规则。在流规则选择方面, 构建同源-目的地址单条/合成流规则合并的流规则搜索树并关联流知识图谱, 达到对已有流规则快速选择并决策的目的; 在流规则学习生成方面, 以流规则搜索树图融合的方式分裂生成流规则安全决策图, 以此根据流标记生成或选择流规则。在评估部分, 本文通过与应用平面交互、流规则选择、流规则学习等三个角度观察流知识图谱的实际应用方向及可能性, 并通过实验衡量了基于流知识图谱的关键算法性能。以流知识平面的图谱等为基础设施, 可近一步深入具体场景, 通过流安全标记与应用相结合的方式, 促进流规则演进等实践开展。

关键词 软件定义网络; 流表; 流规则; 流标记; 知识图谱
中图法分类号 TP309 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2019.07.05

Research on Security Elements Knowledge Graph of Flows in Software-Defined Network

YOU Ruibang[1, 2], YUAN Zimu[1*], TU Bibo[1, 2], MENG Dan[1, 2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract In software defined networks (SDN), flow table interacts as the core component between the control plane and data plane, and is also the key to achieve global coordination and dynamic mapping for implementing security policies. However, constructing such flow tables with security policies faces challenges that the source of the related knowledge elements are scattered over the network, need continuously expanding when flow applications differ, and it is almost impossible to implement all the security policies by preset rules or independent applications. To tackle these challenges, we propose to build a newly knowledge plane besides current planes in SDN. On this knowledge plane, we construct flow knowledge graph based on flow tables with the corresponding knowledge elements on policy adoptions and decisions, and choose or generate flow rules based on the constructed flow knowledge graph. On the aspect of choosing flow rules, we build a search tree based on homologous source-destination address of single or synthetic flow rules, and links the corresponding knowledge elements in flow knowledge graph. On the aspect of learning to generate flow rules, the decision graph of flow rules for a unit is generated by fusing the search trees from a set of targeted, training units, and the decision graph can be used to generate or choose the flow rules conforming to the security labels of a flow. In evaluation section, we assess the practicality of the flow knowledge graph (or say knowledge panel) through the view of its interactions with the application panel, choosing flow rules, and learning from the linked knowledge elements of flow rules, and conduct experiments on the performance of key algorithms. The built flow knowledge graph can be regarded as a base installation. With the flow knowledge graph, we can move into specific scenes, combining flow labeling with applications, to promote the performance of practices, such as dynamically evolving the flow tables under the dynamic SDN environment.

Key words software defined network; flow table; flow rule; flow label; knowledge graph

1 引言

软件定义网络(Software Defined Network, SDN)将数据转发与逻辑控制分离, 通过控制平面实现应用和数据的集中管控。开放网络基金会(Open Networking Foundation, ONF)是业界公认的 SDN 标准化

通讯作者: 袁子牧, 博士, 副研究员, Email: yuanzimu@iie.ac.cn。
本课题得到国家重点研发计划基金资助项目“网络空间安全”重大专项课题(No.2016YFB0801002), 国家自然科学基金(No. 61602470)和中国科学院信息工程研究所基础前沿项目(No.Y7Z0271116)资助。
收稿日期: 2017-11-03; 修改日期: 2018-02-13; 定稿日期: 2019-06-06

研究机构, 其为 SDN 制订了接口和架构上的定义规范与标准。ONF 定义的 SDN 模型包括应用平面、控制平面和数据平面; 控制平面为应用平面提供一系列可扩展编程管理接口(通常称为北向接口, Northbound Interface), 因应用及交互的多样性和复杂性, ONF 并未完成对其的标准定义; 在控制平面与数据平面之间, ONF 目前负责制订 OpenFlow 协议, 其定义了控制器与数据平面基础设施通信交互接口规范(也称为南向接口, Southbound Interface)。OpenFlow 根据应用定义或预设规则匹配并处理网络包, 其中所有的规则都被组织在不通的流表(FlowTable)中, 在流表中根据优先级适用规则。

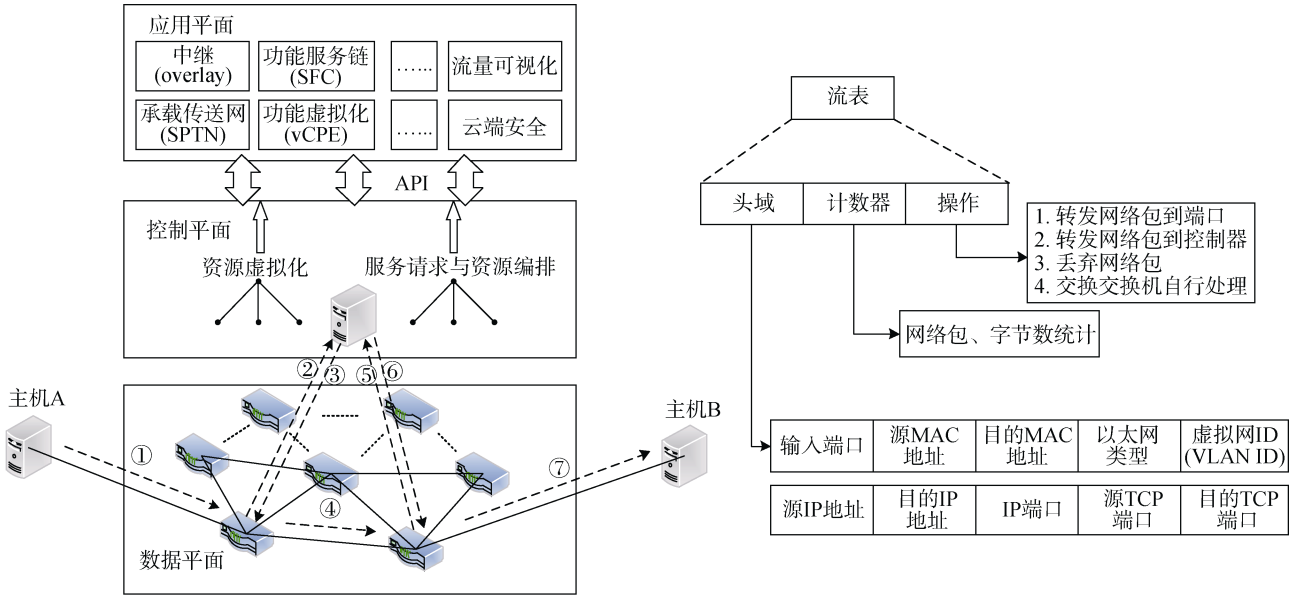


图 1 SDN 架构及流表示意

Figure 1 SDN architecture and flow representation

图 1 中①~⑦步骤示意 SDN 网络通过查询流表完成主机 A 与主机 B 初次通信的过程, 其中②③指网络包达到交换机 1, 交换机 1 查询之前下发至本地的流表, 发现并无相关条目; 随后, 交换机 1 通过传输层协议(TCP/TLS)将网络包发送给控制器(Packet-In 事件); 控制器决定转发端口并下发回交换机(Packet-Out 事件), 交换机 1 执行决策将网络包转发给交换机 2。⑤⑥步骤也类似, 最终初次通信网络包到达主机 B。

流表构建是 SDN 虚拟网络环境下实现安全策略全局协同及动态映射的核心步骤。然而, 流表构建知识却异常分散, 随要素信息的变换而不同, 且需不断扩充, 难以通过独立应用或控制平面的预设规则满足安全策略协同及映射的全部需求。具体来说:

(1) 自定义、第三方应用太多, 应用间的交互并无统一的方式, 各应用间所同步的流规则存在彼此冲突覆盖的现象; 解决这一现象问题需结合网络拓扑、架构、应用、分析人员决策选择等要素知识综合判断, 且相关知识随网络运行会动态变化及演进;

(2) SDN 网络运行异常的原因众多, 受到控制器脆弱性、管理站脆弱性、交换机脆弱性、控制器与应用间交互、控制器与基础设施间交互、伪造数据等多方面的因素影响; 脆弱性成因往往由具体程序实现问题产生, 因而模式并不固定; 从流规则决策的角度来说, 需要充分考虑各要素的历史状态、程序版本等知识;

(3) 安全策略的采用依赖于相关设施等要素知识。如相关安全策略应用设施是否存在, 是否采用导流方式等; 如表 1 所示, 如网络中部署有蜜网, 则可选择将恶意流量导流至蜜网, 或采取正常通过、简单丢弃等策略; 而探测到流违反网络既定策略时, 则可选择导流至处理或通知服务器。

知识有效聚集是应对 SDN 网络流决策要素信息庞杂、分散的必要手段。一方面, 如前所述, 应用过多、交互方式不定, SDN 运行异常原因众多, 流安全策略的采用依赖于众多知识要素, 通过预设安全规则或实现安全机制能解决部分可预先估计的流安全状况, 但本质上无法保证能应对并及时响应 SDN 网络动态变化、流决策要素不断扩充的所有情形。另一方面, 采用知识图谱等能有效聚集 SDN 流决策要素的手段, 将能从全局角度形成流决策知识要素关联底图, 不仅能采用等效实现预设规则或安全机制

的既有功能, 且能岁 SDN 网络和要素动态变化而共同演进, 如能在新型知识要素加入的情况下, 调整知识要素以适应新变化。

表 1　流安全标记及相应处理策略举例

Table 1　Flow security tag and corresponding processing strategy example

流安全标记	流安全策略
恶意流量	导流至蜜网(Honeynet)/正常通过/丢弃
违反策略	导流至处理服务器/正常通过/丢弃
网络流量异常	选择性过滤/重新配置导流资源/正常通过/丢球
主机感染	流隔离/正常通过/丢弃
拒绝服务攻击	导流正常流量/正常通过/丢弃
注入攻击	导流至沙网(Sandnet[1])/正常通过/丢弃
远程控制	导流输入或越界流量至处理节点/正常通过/丢弃
服务器异常	流量调整/流量隔离/导流至新服务器/正常通过/丢弃
网络探测	导流至蜜网/正常通过/丢弃
源信誉等级低	导流限制/反欺诈考验/正常通过/丢弃
网络隧道	临时中断/正常通过/丢弃

因此, 本文选择构造流表分析对象及要素信息知识图谱嵌入的知识平面, 智能化指导基于安全策略的流表规则构建。在第二部分介绍基于流知识平面的软件定义网络架构; 在第三部分描述流知识本体, 及举例相应的实体; 在第四部分设计安全属性聚集的流规则知识图谱并论述基于该图谱的流规则合成及选择; 在第五部分描述基于流规则安全知识的决策图构建及相关学习过程; 在第六部分说明功能场景并开展性能实验; 在第七部分论述相关工作; 在第八部分总结全文。

2　基于流知识平面的软件定义网络架构

D. Clark 等[2]在“A knowledge Plane for the Internet”一文中提出知识平面的概念, 其工作范式(paradigm)自动化(识别-动作)或推荐(识别-解释-建议)有极大促进网络操作智能化的潜在价值。本文将知识平面的概念具体应用于 SDN, 将网络流行为与流表安全要素集合的知识图谱相结合, 决策生成网络流规则。具体来说, 如图 2 所示, 知识平面定义流表相关安全要素聚集的本体及存储对应实体且可动态扩展的知识图谱库, 从控制平面获取网络流信息及可能的应用信息, 结合获取信息和知识图谱作为输入, 应用流安全要素指导的知识决策和学习算法生成流规则知识; 同时, 根据可能的人工审计结果, 对比是否根据新生成的流规则知识变更或更新知识图谱库中的数据, 将最终决策生成的网络流规则交付控制平面。

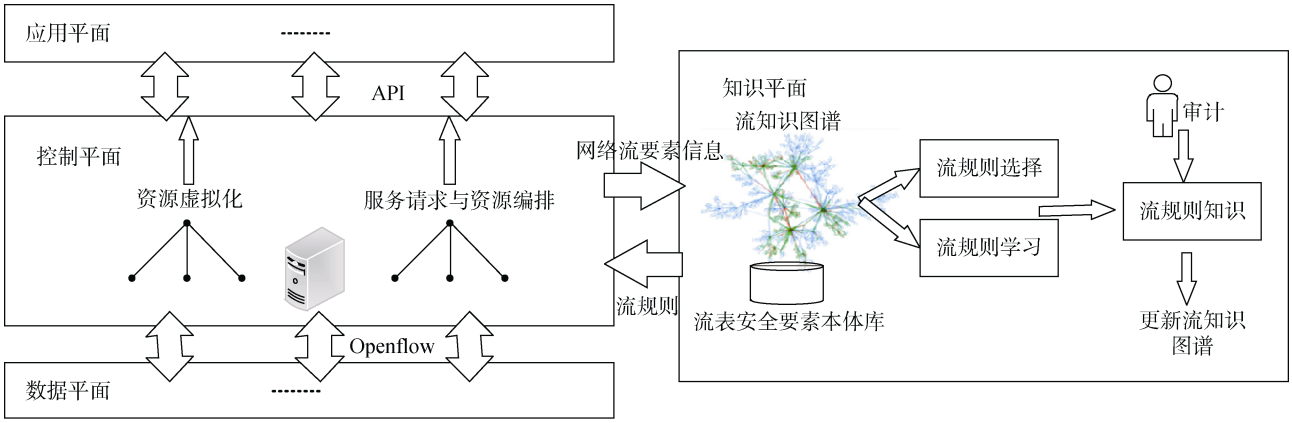


图 2　基于流知识平面的 SDN 架构

Figure 2　SDN architecture based on flow knowledge plane

本文将在第三部分重点描述流表安全要素本体库及其对应的实体举例, 流知识图谱则由众多实体以及实体间的关联组成。第四部分论述“流规则选择”过程, 具体指如何合成、选择并应用现有的流规则, 以及在这一过程中如何构建安全要素聚集的流知识图谱。第五部分论述“流规则学习”过程, 具体指基于流知识图谱学习并生成新的流规则。

3　流知识本体设计及对应实体

在流表安全要素本体及知识图谱库的数据来源方面, 既有流表数据、平面间的交换数据, 也有从应用获取的日志信息、人工录入的知识等, 其来源多种多样、格式不一。因此, 本文认为本体及知识图谱库不应局限于特定几种类型的数据, 应可动态扩展、持

续更新。为了阐明流表安全要素本体及相应实体的概念, 本文将以流表和应用与数据平面间数据交换为例叙述本体设计及相应实体例子。本体描述语言包括 RDF[3]、DAML[4]、OWL[5]等, 本文将以 RDF 格式为例介绍。

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="......"
  xmlns:ft="resource/ft#">
  <rdf:Description
    ref:about="resource/ft/1">
    <ft:输入端口>*</ft:输入端口>
    <ft:源MAC地址>*</ft:源MAC地址>
    <ft:目的MAC地址>*</ft:目的MAC地址>
    <ft:以太网类型>*</ft:以太网类型>
    <ft:虚拟网ID>600</ft:虚拟网ID>
    <ft:源IP地址>10.4.8.6</ft:源IP地址>
    <ft:目的IP地址>10.4.1.6</ft:目的IP地址>
    <ft:IP端口>*</ft:IP端口>
    <ft:源TCP端口>*</ft:源TCP端口>
    <ft:目的TCP端口>*</ft:目的IP端口>
    <ft:数据包统计>3</ft:数据包统计>
    <ft:字节数统计>294</ft:字节数统计>
    <ft:操作>转发端口3</ft:操作>
  </rdf:Description>
</rdf:RDF>

<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="......"
  xmlns:ft="resource/de#">
  <rdf:Description
    ref:about="resource/de/1">
    <de:流方向>应用平面->数据平面</de:流方向>
    <de:操作>流规则修改</de:操作>
    <de:控制器策略>流合规检测</de:控制器策略>
    <de:权限>应用级权限</de:权限>
  </rdf:Description>
</rdf:RDF>
```

图 3 流知识本体及对应实体示例

Figure 3 Flow ontology and corresponding entity examples

图 3 例举了流表以及数据交换两类本体设计。其中, 本体“流表项”使用“resource/ft#”标识, 包含输入端口、源 MAC 地址、目的 MAC 地址、以太网类型、虚拟网 ID、源 IP 地址、目的 IP 地址、IP 端口、源 TCP 端口、目的 TCP 端口、数据包统计、字节数统计、操作等属性字段; 流表项实体“resource/ft/1”表示该流表项处理源 IP 地址为“10.4.8.6”、目的地址为“10.4.1.6”路由, 其操作为“转发端口 3”, 当前虚拟网 ID 为 600, 总计有 3 个数据包, 共 294 个字节, 其他字段值为任意。本体“数据交换”使用“resource/de#”标识, 该本体用于表示 SDN 应用平面与控制平面的交互, 包含流方向、操作、控制器策略、权限等属性字段; 数据交换实体“resource/de/1”表示应用平面向数据平面提交流规则修改操作, 控制平面的控制器限定该流规则修改的权限为应用级, 并检测该流规则修改是否合规。

在实际使用中, 本文采取将所有 RDF 描述数据转化为三元组<头, 关系, 尾>(<head, relation, tail>)的表示形式, 如图 4 中流表实体“resource/ft/1”中属性源 IP 地址 10.4.8.6 将被转换为<“resource/ft/1”, “普通属性: 源 IP 地址”,“10.4.8.6”>; 在该三元组中, “resource/ft/1”和“10.4.8.6”均被转换为实体, “普通属性: 源 IP 地址”表示两者之间的关联。选择<头, 关系, 尾>的实现形式可直接应用知识图谱表示学习的成果(transE[6], transG[7]等), 将语义信息表示为稠密低维实值向量用于计算与推理。

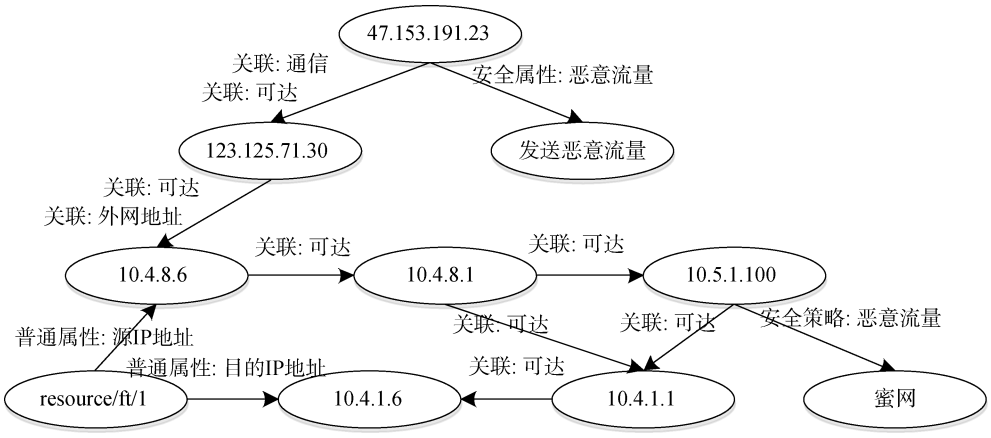


图 4 流知识图谱示例

Figure 4 Flow knowledge graph example

为了更直观的阐述流知识实体及其在流安全上的应用, 本文以图 4 为例说明。如前所述, 该图将所有的 RDF 描述数据转为<头, 关系, 尾>的三元组表达形式, 示例为 IP 地址为“47.153.191.23”的主机向内网地址为“10.4.1.6”的主机发送恶意流量, 其可通过“resource/ft/1”等多条流规则的组合沿

“123.125.71.30(10.4.8.6)” ->“10.4.8.1” ->“10.4.1.1” ->“10.4.1.6”路线达到目的地。同时，地址“123.125.71.30”所对应内网已部署蜜网，可引导恶意流量至蜜网入口“10.5.1.100”；通过新添加流规则并设置高优先级，可将所有经外网进入的恶意流量导流至蜜网做下一步引诱捕获处理。基于安全策略的流规则添加示例过程如算法 1 所示，其通过检索“安全策略”在流规则“安全属性”标记中是否存在，进一步决定是否添加新的流规则。本文将安全策略、安全属性以及其相关的决策知识统称为安全要素知识，流知识图谱可随着流规则及相关要素知识的不断添加而动态扩展。

算法 1：基于安全策略的流规则添加

输入：流知识图谱 *FG*，新增实体 *E*，流表 *FT*

输出：流表 *FT*

IF *E.contains*(“安全策略:*x*”) THEN

  FIFO=Empty queue;//空队列

  FOR *E’* IN *E.relatedBy*(“关联:可达”) THEN

push(*E’*,*FIFO*);

IF *E’.contains*(“安全属性:*x*”) THEN

  BuildFlowRule(*E*, *E’*, *FT*);

END IF

  END FOR

  BFS(*FIFO*,*FG*);//广度遍历

END IF

RETURN *FT*;

4 流规则选择

通常，知识图谱决策算法指通过对已有实体及其关联关系的推理计算建立输出实体间的关联。将其应用到流规则决策上，具体指在构建流规则知识图谱的基础上推理得出给定源-目的地址所有合成流规则或与源-目的地址间安全要素相似的流规则。目前，常用的知识图谱决策方式是在知识图谱嵌入(Embedding)至低维稠密空间(即采用 transE 方法等)之后，使用张量分解方法[8, 9]和路径推理方法[10, 11]等计算求解。张量分解方法将知识图谱当作一个庞大的张量，进而分解成小的张量片。但张量分解方法只考虑实体间的直接关联，忽略了多路径的可能。而对于一对源-目的节点来说，可能不存在流规则，也有可能存在一条或多条对应流规则，且存在多条流规则组合(即路径组合)成为以该对节点为起始和结束节点的合成流规则的情况。因此，本文采用路径推理的方式，一方面结合路径关系进行计算；另一方面针对现有路径推理算法不能解决长路径计算且效率不高的问题[12]，考虑通过合并流规则实体，构建安全属性聚集的流规则知识图谱。

给定一对源-目的节点，对于流规则决策来说，需找出(1)具备相同源-目的地址的单条流规则和合成流规则，此处合成流规则指多条流规则组合形成具有给定源-目的地址的虚拟流规则，可用于流规则选择；(2)哪些流规则的路由选择因素与当前指定源-目的路径上的因素相似，如可以采用一致的安全策略生成新的流规则。本文将(1)称为“流规则选择”，(2)称为“流规则学习”，将分别在本章节部分和第五部分阐述。

在流规则选择方面，本文主要考虑合并流规则实体，扩展安全属性聚集的流知识图谱，通过构建好的流规则实体间关联快速查找具有相同源-目的地址的单条或合成流规则，同时为后续流规则学习提供规整而充分的特征输入。

令任意流规则 r_i 和 r_j 间可以相互比较。如先根据流规则的源 IP 地址(*r.src*)进行比较，按 IP 地址从高位至低位依次进行数值比较，高位数值较小(认为 10.4.1.6<10.5.1.1)则对应的流规则也较小，如高位数值相等则继续进行低位数值比较；在源 IP 地址相同的情况下，比较目的 IP 地址；如目的 IP 地址也相等，则认为 $r_i=r_j$。根据流规则比较结果，将某流表 *FT* 里所有规则组成二叉搜索树图 *T*，即对于某节点，其左指针指向子节点所表达的流规则小于该节点表示的流规则，右指针指向子节点流规则大于该节点流规则。流知识图谱可在搜索树图 *T* 的基础上进一步扩展。

本文假设流知识图谱 *FG* 已存在(如图 4 示例所示，可包含流表数据、平面间的交换数据，应用日志信息、人工录入等知识来源)，且已对流表 *FT* 建立搜索树图 *T*。算法 2 说明了给定一组源-目的地址(即流规则集 *R*)，查找具备相同源-目的地址的合成流规则，并扩展流知识图谱 *FG* 的过程，具体如下：

- 已有流规则查找部分。对于流规则 $rj \in R$，在 *T* 中查找具有相同源-目的 IP 地址流规则节点；如找到该类型节点，则在 *T* 中将表示流规则 *rj* 的节点与该节点合并；如无，则往 *T* 中插入表示流规则 *rj* 的节点。*Merge*(*FG, T*)表示 *T* 中节点将继承合并前或插入前的流规则 *rj* 节点在流知识图谱 *FG* 中的关联关系。
- 合成流规则查找部分。对于流规则 $rj \in R$, $ri \in T$，其合成流规则包括(1)以 *ri.src* 为源，*rj.dest* 为目的地址，和(2)以 *rj.src* 为源，*ri.dest* 为目的地址。以(1)为例，其首先通过 *GenerateNode*()生成合成节点，该节点表示以 *ri.src, rj.dest* 分别为源和

目的的虚拟流规则; 其次, 以该合成节点为根, 调用 *BuildTree*()以遍历方式找到多组流规则构建树 T'; 在 T'中, 每一组完成流规则可组合为以 *ri.src* 为源和 *rj.dest* 为目的地址的合成流规则, 而一组非完成流规则指该组流规则无法组合为以 *ri.src* 为源和 *rj.dest* 为目的地址的合成流规则, 通过 *PruneUncompletedNode*()剪枝删掉部分无法组合的流规则; 最后, *Merge*(*FG, T'*)使得 T'中所有节点继承其原有流规则节点在流知识图谱 *FG* 中的关联关系, 并将合成流规则 *Update*()更新至流表 *FT* 和搜索树 *T*。

- 在合成流规则查找部分, *BuildTree*()首先构建以传入参数 *src* 和 *dest* 分别为源和目的地址的虚拟节点并作为子节点接入树 T', 如存在流规则 *rk* 且 *rk.src* 和 *rk.dest* 分别等于 *src* 和 *dest* 则表示已构建好一组完成流规则, 直接返回树 T'; 否则以 *ri.dest* 作为源, *rj.src* 作为目的地址继续在树 *T* 中分别搜索对应流规则集 *Ri* 和 *Rj*, 在不为空的情况下返回递归扩展构建的树 T'。

算法 2: 基于流规则查找及合成的流知识图谱扩展过程

输入: 流知识图谱 *FG*, 流表 *FT*, 待比较流规则集 *R*, 流表二叉搜索树图 *T*

输出: *FG*

// (1) 已有流规则查找合并

FOR each *rj* in *R* THEN

  Tj = BinarySearch(*rj*,*T*);//在 *T* 中查找非 *rj* 自身具有相同源-目的 IP 地址流规则

  IF *Tj* != *NULL* THEN

    T = *Merge*(*rj*, *Tj*, *T*);//合并属性

  ELSE THEN

T = *InsertNotExists*(*rj*, *T*);

  END IF

END FOR

FG = *Merge*(*FG*,*T*);

// (2) 合成流规则查找合并

FIFO=*Empty queue*;//空队列

// 查找 *ri.src rj.dest* 对应合成流规则(*ri* in *T*, *rj* in *R*)

FOR each *ri* in *T*, *rj* in *R* THEN

  T' = *BuildTree*(*ri*, *rj*, *ri.dest*, *rj.src*, *T*, *NULL*);

  T' = *PruneUncompletedNode*(*T'*);

  FG = *Merge*(*FG*,*T'*);

  [*FT, T*] = *Update*(*FT*, *T*, *FG*);

END FOR

// 查找 *rj.src ri.dest* 对应合成流规则(*ri* in *T*, *rj* in *R*)

FOR each *ri* in *T*, *rj* in *R* THEN

  T'= *BuildTree*(*rj*, *ri*, *rj.dest*, *ri.src*, *T*, *NULL*);

  T' = *PruneUncompletedNode*(*T'*);

  FG = *Merge*(*FG*,*T'*);

  [*FT*, *T*] = *Update*(*FT*, *T*, *FG*);

END FOR

RETURN *FG*;

//以 *src* 为源地址, *dest* 为目的地址, 搜索并合成规则

T' = *BuildTree*(*ri*, *rj*, *src*, *dest*, *T*, *T'*)

输入: 流规则 *ri* 和 *rj*, 源和目的地址 *src* 和 *dest*, 合成规则树 *T'*, *T*

输出: 合成规则树 *T'*

T'=*GenerateNode*(*ri*, *rj*, *src*, *dest*, *T'*);

IF *src*==*dest* THEN

  T' = *GenerateNode* (*ri*, *rj*, *T'*);

END IF

IF exist (*rk.src*= *src* && *rk.dest* =*dest* && *rk* in *T*) THEN

T' = *GenerateNode* (*rk*, *T'*);

END IF

WHILE not exist (*rk.src*= *src* && *rk.dest* =*dest* && *rk* in *T*) THEN

Ri = *BinarySearch*(*ri.dest* as source,*T*);

Rj = *BinarySearch*(*rj.src* as *destination*,*T*);

IF *Ri*==*NULL* or *Rj*==*NULL* THEN

  RETURN *T'*

END

FOR each *r'i* in *Ri*, *r'j* in *Rj* THEN

T'= *BuildTree* (*r'i*, *r'j*, *r'i.dest*, *r'j.src, T*, *T'*);

  END FOR

END WHILE

RETURN *T'*;

算法 2 中, 搜索树 *T* 的最终节点数为初始 *T* 节点数和 *R* 节点数之和, 可用$|T^{(0)}+R|$表示, *BinarySearch*()的比较次数为 $O(\log(|T^{(0)}+R|))$。算法 2 的复杂度为 $O(|T^{(0)}||R|\log(|T^{(0)}+R|))$。

图 5 为算法 2 关键数据知识部分示意。算法中 *ri* 在图中对应用“resource/ft/i”表示, 搜索树 *T* 以流规则间比对关系构成, 如“resource/ft/1”<“resource/ft/3”表示 $r_1.src < r_3.src$, 或 $r_1.src = r_3.src$, $r_1.dest < r_3.dest$; “resource/ft/2” 与 “resource/ft/2′ ” 位于同一节点则表示 $r2.src = r2'.src$, $r2.dest = r2'.dest$。合成流规则树 T'以“resource/ft/s1”为根节点, 其可扩展出多组流规则组合, 如“resource/ft/4”-“resource/ft/7”表示一组完成流规则, 其中 $rs1.src = r4.src$, $rs1.dest = r5.dest$; 而以“resource/ft/s2”为根节点的合成流规则树(虚线框)则表示一组非完成流规则, 则需通过 *PruneUncompletedNode*()剪枝删掉。最终, 剪枝后的 T'将更新至 *T*, 同时 *T* 和 T'均为流知识图谱 *FG* 中的一部分。

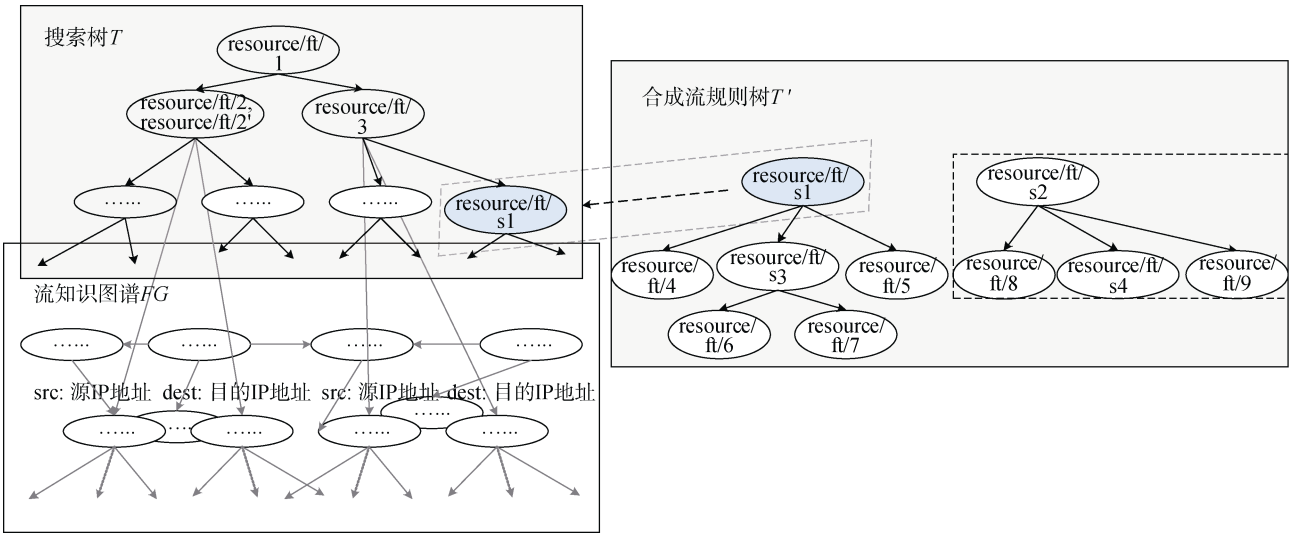


图 5　搜索树 *T* 和合成流规则树 *T'*示例

Figure 5　Search tree *T* and synthetic flow rule tree *T'* example

5　流规则学习

在考虑流安全的前提下, 无论流规则的选取采用、丢弃流包, 亦或是创建新规则导流等决策均与节点可实施的安全策略、流的安全属性、节点是否可达中的多项要素有关。而具体安全策略的采用以及如何应用到流规则上则受设计者或实施决策的分析人员影响, 且并无严格区分安全策略好坏的标准。因此, 应用安全策略的智能化流规则决策必不可少地需要引入先验决策知识以及在人工干预之后的重新调整。

每一节点或局域网络对具体流规则安全策略的选择并不相同。本文初始以局域网、核心路由器等可能单独应用安全策略的网络或节点为单元, 每一单元采用单独的决策树决定流规则的选择。人工干预之后会带来流规则选择的改变, 需重新调整决策树, 本文依据安全策略的优先级使用决策树属性值分裂的方法进行调整。此处, 决策树分裂与通常意义上所研究的基于信息增益等减少分类比较次数[13, 14]不同, 其过程是基于决策而逐步演进的过程, 使得决策树通过属性分裂适应分析人员的策略选择。

对于每一单元, 在初始构建流决策树时, 先扫描其所有节点中对应的所有流规则, 依据具备相同源和目的地流规则的优先选择顺序确定流决策树的构建形态。如算法 3 所示, 对于一个待建立流规则决策图的单元 *u*, 以算法 2 中所构建的流规则知识图谱 *FG* 和流表 *FT* 为输入, 最终返回流规则决策图 *RGu*, 其具体过程如下:

- 在初始化 *FTu*=*FT* 和 *RGu* 后, 遍历 *FTu* 中的所有流规则 *r*;
- 当用 *RuleContians*()函数判断到流规则 *r* 与单元 *u* 相关(即源或目的地在 *u*, 或经过 *u*), 则对流规则 *r* 进行处理;
- 通过 *GetBinaryTree*()搜索流规则知识图谱 *FG*(或搜索树 *T*)找到在算法 2 中生成与 *r* 同源和同目的地的所有流规则 *Tr*, 并用 *DeleteEntries*()将 *Tr* 中的流规则从 *FTu* 中删除以避免重复循环;
- 通过 *GetPriorList*()函数返回 *Tr* 中按优先级排序的流规则列表 *LT*, 列表中的流规则其按优先级从低到高排列;
- 依次遍历列表 *LT* 中的每一条流规则 *ri*, 调用 *SplitGraph*()应用决策图属性值分裂的方式构建 *RGu*, 如表 2 所示;
- 对于单元 *u*, 最终返回构建好的流规则决策图 *RGu*。

算法 3: 流规则决策图的构建

输入: 流规则知识图谱 *FG*, 流表 *FT*, 待建立流规则决策图的单元 *u*

输出: 流规则决策图 *RGu*

使得 *FTu*=*FT*;

初始化 *RGu*;

FOR each *r* in *FTu* THEN

　IF *RuleContians*(*r*,*u*) THEN

Tr=*GetBinaryTree*(*r*,*FG*);

DeleteEntries(*FTu*,*Tr*);

LT=*GetPriorList*(*Tr*);//*LT* 中的规则以优先级从低到高排列

FOR each *ri* in *LT* THEN

　RGu=*SplitGraph*(*RGu*, *ri*);

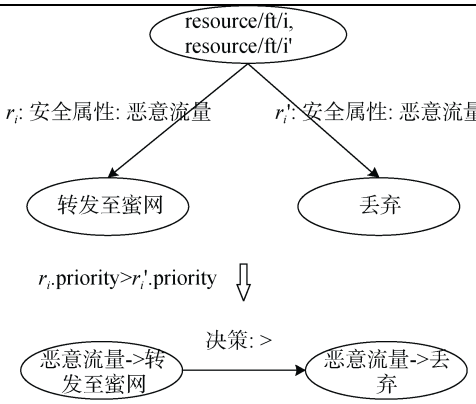
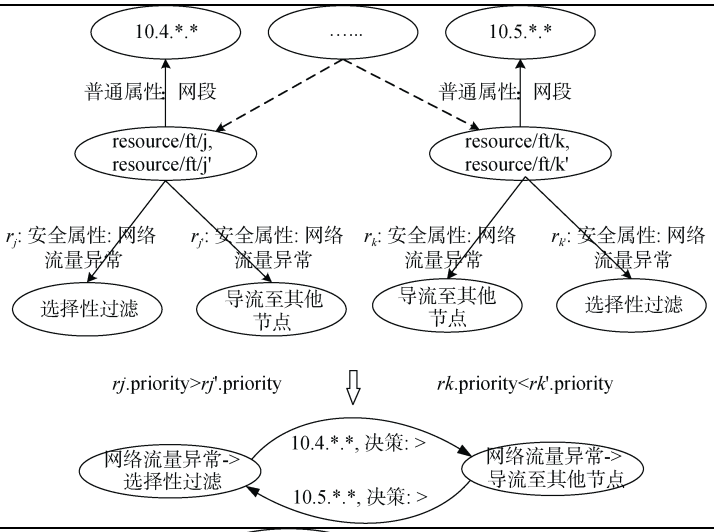
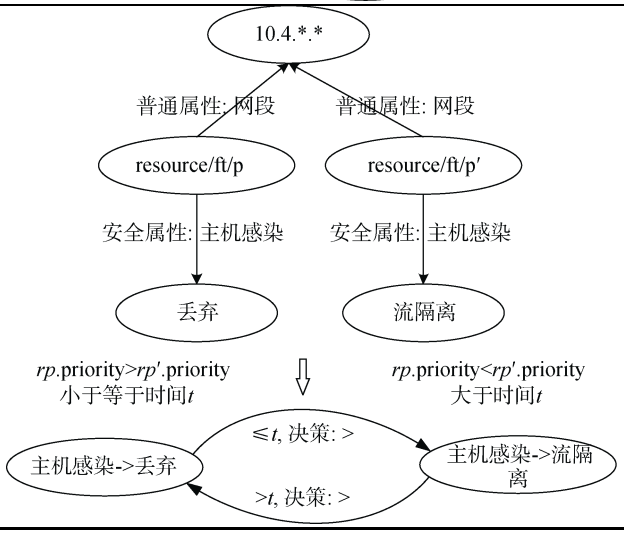
END FOR

END IF
END FOR

FG=Merge(*FG*, *RGu*);
RETURN *RGu*;

表 2　决策图属性值分裂示例说明

Table 2　Decision graph attribute value splitting example description

分裂规则优先级	示例图 (对于任意规则 *ri*, 其在图中对应用“resource/ft/i”表示)	示例解释
(1)当流规则 r_i 与 r_j 互斥时(即 $r_i.priority > r_j.priority$, 且执行 r_i 后不再执行 r_j), 分裂 r_i 与 r_j 所附加安全属性和相关转发动作, 分裂后的边连线声明相互的优先级关系;	 流规则所能实施的安全策略与网络部署有关。如无蜜网部署, 则依据优先关系将恶意流量包丢弃。FG=Merge(FG, RGu);表示 RGu 与 FG 合并, 继承原有流规则在 FG 中的边连接关系。	“resource/ft/i”和“resource/ft/I″”两条流规则用于处理恶意流量, 其所标记的安全策略分别为“转发至蜜网”和“丢弃”, 优先级 $r_i.priority > r_j.priority$. 在决策图上, “恶意流量->转发至蜜网”优先级高于“恶意流量->丢弃”。
(2)当(1)中的流规则 *ri* 与 *rj* 无附加安全属性或安全属性及策略冲突时, 如决策图单元 *u* 为网络, 则按子网划分分裂节点并标注优先级关系;		“resource/ft/j”和“resource/ft/j′”同源-目的地址, 在搜索树 *T* 位于同一节点, “resource/ft/k”和“resource/ft/k”也是如此。但两组流规则安全策略相互冲突, 分别用于 10.4.*.*和10.5.*.*等单元 *u* 下的两个子网段, 分裂时按网段划分。
(3)当(2)无法完成划分时, 即决策图单元 *u* 为单节点或无法划分的网络, 则按时间点并标注优先级关系。		“resource/ft/p”和“resource/ft/p”指向同一网段, 且相互冲突。通过(1)和(2)方法无法区分, 按照时间 *t* 前后的优先级关系标注。

算法 4 为在单元 u 内的流规则决策过程。其首先检查流所标记的安全属性, 如当前流是否为恶意流量; 其次, 提取安全标记集合 P, 在依据 RGu 解决流规则冲突的前提下提取或生成具备相应安全策略的流规则集 R。在 P 为空或允许多份转发的情况下, 如某流量包标记为恶意流量并复制一份转发至蜜网, 同时允许其按正常流转发, 将相应流规则加入集合 R。最终, 算法 4 向 SDN 控制平面返回流规则集合 R。

算法 4: 流规则决策

输入: 流 *flow*, 流规则决策图 *RGu*,

输出: 决策规则集 *R*

决策规则集 *R=NULL*;

安全属性集 *P=flow.labelsof*(“安全属性”);

FOR each *p* in *P* THEN

IF not *Conflict*(*p*, *P*, *RGu*) THEN

R=Union(*R*, *RuleRelated*(*p*, *RGu*));

ELSE IF *Prior*(*p*, *P*, *RGu*) THEN

R=Union(*R*, *RuleRelated*(*p*, *RGu*));

R=Subtract(*R*, *RuleRelated*(*Conflict*(*p*, *P*, *RGu*)), *RGu*);

END IF

END FOR

IF *AllowForward*(*RGu*) THEN

R=Union(*R*, *RuleNormal*(*flow*, *RGu*));

END IF

RETURN *R*;

当新加入一个节点时, 如其为某单元 u 中节点, 则可按照单元 u 的决策图 RGu 选择流规则; 当新加入节点或网络本身作为一个单元 u 时(如某一核心路由器或局域网初始应用安全流策略), 则可选择学习一组单元集合 U 的决策图 RG_U, 根据网络部署情况应用相关的安全策略并生成流规则。因此, 对于新加入的单元 u, 本文将通过合并一组单元集合 U 的决策图 RG_U 并应用学习策略生成其流规则决策图。如算法 5 所示, 在合并流规则决策图结合 RG_U 之后, 压缩删除无关节点并探测相互冲突的安全属性(即无法确定安全属性 A 与 B 的优先级, 如在单元 u_p 决策图中 *A.priority>B.priority*, 而在单元 u_q 决策图中 *A.priority<B.priority*), 通过学习新加入单元 u 的结构特征或考虑其他要素信息消减冲突得到单元 u 的决策图 RGu。

- 其中 *Merge*()简单合并决策图集合 RG_U, 多决策图中名称一致的节点合并为同一节点, 并继承原有边关系;
- *Condense*()删除与 *Learning2DeleteCycles* (*RGu*, *FG*, “*method*”)学习无关的节点, 如“*method*”为“*Count*”时, 将删除非安全属性节点及边连接关系和单元 u 中不具备的安全属性节点, 并让邻居节点继承其原有优先级边关系;
- *DetectCycles*()检查优先级相互冲突的安全属性, 并标记相关冲突属性;
- *Learning2DeleteCycles*()通过学习决策图要素特征解决安全属性相互冲突问题, 考虑其实现方法多样, 本文例举三种方法: “*Count*”统计优先级采用次数, “*Match*”匹配单元 u 决策要素与其他单元的相似度, “*Statistics*”采用多标签分类的方法。

算法 5: 流规则决策图合并学习

输入: 流规则决策图集合 RG_U, 流规则知识图谱 *FG*

输出: 决策图 *RGu*

RG_U=*Merge*(RG_U);

RGu =Condense(*RGu*, “*method*”);

RGu =DetectCycles(*RGu*);

Rgu =Learning2DeleteCycles(*RGu*, *FG*, “*method*”);

RETURN *RGu*;

学习方法

“*Count*”方法: 在 *Condense*()删除非安全属性之后, 统计安全属性两两优先关系被规则决策图集合 RG_U 所采用的次数。其具体方法为: 初始时将 *RGu* 中采用次数最多的安全属性优先关系加入 *RGu*(“*Count*”), 随即依次将 *RGu*(“*Count*”)邻接安全属性节点 *RGu* 中优先级关系采用次数最多的边及相关节点加入 *RGu*(“*Count*”), 并在 *RGu* 中删除所加入的节点和优先级边, 直至最后任何加入任何 *RGu* 中的优先级边均会于 *RGu*(“*Count*”)中的优先级关系冲突则返回, 如算法 6 所示。

算法 6: *Learning2DeleteCycles*(*RGu*, *FG*, “*Count*”)

输入: 单元 *u* 决策图 *RGu*, 流规则知识图谱 *FG*, 方法“*Count*”

输出: 解决冲突后的决策图 *RGu*(“*Count*”)

使得 *RGu*(“*Count*”)=*MaxCountEdge*(*RGu*);

WHILE *HasNoCycleEdge*(*RGu*) THEN

G = MaxCountConnectedEdge(*RGu*);

RGu(“*Count*”)= *RGu*(“*Count*”) *U G*;

RGu = RGu - G;

END WHILE

RETURN *RGu*(“*Count*”);

“*Match*”方法: 适用于掌握因变量关系的情形(即掌握决策安全属性间优先级的知识要素), 基于图

相似匹配的结果解决安全属性冲突问题。其具体方法为: 初始时将RG_u("*Match*")设为RG_u, 随即依次计算 $\underline{RG_u}$("*Match*")中冲突的安全属性优先级关系及其在流知识图谱 *FG* 关联的决策知识要素与现有决策图间的相似分数(如对于某两个安全属性的优先级关系及其决策知识要素, 如存在已有单元 *u'*, 其决策知识要素图与单元 *u* 决策知识要素图相似, 则认为其相似分数高, 否则认为其相似分数低), 并在RG_u("*Match*")中删除相似分数的优先级关系, 重复这一过程直至无优先级关系冲突, 如算法 7 所示。其中, 相似分数计算可直接应用众多已有的图相似性匹配算法[15-16]。

算法 7: *Learning2DeleteCycles*(*RGu*, *FG*, "*Match*")

输入: 单元 *u* 决策图 *RGu*, 流规则知识图谱 *FG*, 方法"*Match*"

输出: 解决冲突后的决策图 *RGu*("*Match*")

使得 *RGu*("*Match*")= *RGu*;

所有冲突的优先级关系 *CS*=*Cycles*(*RGu*("*Match*"));

WHILE *HasCycles*(*RGu*("*Match*")) THEN

RS = *SimilarityMatchScore*(*CS*, *FG*);

RGu("*Match*") = *DeleteCycleWithLowestScore*(*RS*, *RGu*("*Match*"));

CS=*Cycles*(*RGu*("*Match*"));

END WHILE

RETURN RG_u("*Match*");

"*Statistics*"方法: 适用于因变量关系不明确的情形, 采用多知识要素统计决策的方式解决安全属性冲突问题。对于流规则知识决策, 可将 *RGu* 中的每一对节点间优先级关系视为一个标签, 流知识图谱 *FG* 中每一关联知识要素视为一个特征, 转化为多标签分类问题。设知识要素集合为 *X*, 标签集为 *L*, 多标签分类问题目标为在给定数据集上确定分类函数 f: $X\rightarrow\{-1,0,1\}^{|L|}$, 其中 0 表示无法推断该标签所代表的优先级关系是否成立, 1 表示成立, –1 表示不成立。从本文关注的问题看, 标签之间不一定冲突, 如(1,0) < (1,1)表示前者存在属性间优先级 $A<B$, 后者 $A<B$ 与 $A<C$ 并存, 但两者并不冲突。因此, 可将标签集视为|L|-维空间图, 各分类结果用空间顶点 *v* 表示, 并建立边连接关系; 如对于任意 v_i, v_j, 如$\sum|v_i - v_j|=1$, 即仅有一个表征属性优先级的值不同, 且与其他边表征的优先级关系不冲突, 则建立边连接。该|L|-维空间图可通过压缩或变换映射到低维空间, 通过在低维空间向量上建立回归函数使得原|L|-维空间图建立边连接关系的或距离在 *N* 跳之内的两两节点距离在某阈值 *d* 之内, 而不在此范畴内的两两节点映射到低维空间距离则大于阈值 *d*(相应的方法如[6-17])。令 *V* 表示 *X* 在低维空间的投影, *Y* 为分类结果, $Y\in\{-1,0,1\}^{|L|}$。可让学习过程考虑如下优化过程:

$$\mathrm{Min}_{\{A,B,V\}}\ (1-a)\|X-VA\|^2+a\|Y-VB\|^2,\ V^TV=I$$

其中, *A* 为低维空间至|L|-维空间映射, *B* 为低维空间至的$\{-1,0,1\}^{|L|}$映射, $0\leqslant a\leqslant 1$ 调节 *X* 和 *Y* 在低维度空间至原有空间的重构误差比例。该式可用[18, 19]中描述的方法求解。最终应用式(1)得出的 *X* 与 *Y* 之间的关系, 以单元 *u* 的决策要素知识为输入得出分类结果; 值得注意的是, 分类结果并不能确保为非冲突的结果, 一旦冲突则需要重新学习, 如形成解决冲突后的训练集 *X'*与 *Y'*。其学习并分类过程如算法 8 所示。

算法 8: *Learning2DeleteCycles*(*RGu*, *FG*, "*Statistics*")

输入: 单元 *u* 决策图 *RGu*, 流规则知识图谱 *FG*, 方法"*Match*"

输出: 解决冲突后的决策图 *RGu*("*Statistics*")

应用流规则知识图谱 *FG* 得到训练集(*X*, *Y*), 采用式(1)训练

应用式(1)分类结果得到 *RGu*("*Statistics*");

WHILE *HasCycles*(*RGu*("*Statistics*")) THEN

解决冲突, 如 *RGu*("*Statistics*")=*Learning2DeleteCycles*(*RGu*, *FG*, "*Count*");

应用 *RGu*("*Statistics*")和 *FG* 得到训练集(*X'*, *Y'*), 采用式(1)训练;

应用式(1)分类结果得到 *RGu*("*Statistics*");

END WHILE

RETURN RG_u("*Statistics*");

6 功能及性能评估

该部分将对基于流知识平面的 SDN 开展功能及性能评估。表 3 列举了三类功能评估。其中流规则知识与应用平面类包括评估 1-应用越权操作和评估 2-应用安全评级, 流规则选择类包括评估 3-流隧道发现和评估 4-流规则更新, 流规则学习类包括评估 5-基于流规则的推荐和评估 6-安全事件成因分析。

评估 1: 应用越权操作。如图 6 所示, 设有某局域网, 其安全策略设置使得主机 B 禁止被外网所访问, 以避免可能引发的安全问题。具体来说, 该局域网通过设置流规则, 当有外网主机发送包给主机 B 时则丢弃, 如在局域网入口设置流规则 r_1: *->B, 丢弃。同时, 可通过控制平面授权修改的方式防止流规

则被恶意篡改[20]。在该局域网范围内, 知识平面可认为经该局域网授权的流规则优先级(r_1.*priority*)高于未经授权的流规则。当某恶意应用设法通过外网主机 A 访问主机 B, 其可采取两种方式。①新增流规则 r_2: *->B, 转发; 但 r_2.*priority*< r_1.*priority*, 所以流规则 r_2 永远不会被执行。②新增流规则 r_3: *->C, 重写 C 地址至 B 地址; 即当包到达主机 C 时, 写入一条发包往主机 B 的流规则, 这在流行的 SDN 开源架构(如 Floodlight[21]等)中并无机制避免这一情况发生; 知识平面可检测到 A->C, C->B 的合成流规则, 合成流规则取多条流规则中优先级最小的为其优先级, 则其优先级小于流规则 r_1 的优先级, 仍然不会被执行。这两种越权访问的方式均无法使外部流包达到被禁止外网访问的内部主机 B。

表 3 功能评估类别及说明

Table 3 Functional assessment category and description

功能评估类别	评估说明
流规则知识与应用平面	应用平面是 SDN 流规则知识的来源, 但恶意应用或多应用交互本身能带来众多安全问题。在流规则的操作方面, 评估知识平面对应用在流安全上的限制。
流规则选择	在本文第四部分, 流规则选择涉及具备相同源-目的地址单条/合成流规则及相关搜索树构建。评估流规则选择在查询、更新及安全检测上的应用效果。
流规则学习	在本文第五部分, 流规则学习涉及基于搜索树的安全策略决策图及相关流规则生成。评估其在安全问题及相关策略采用成因以及节点部署等流规则知识上的学习效果。

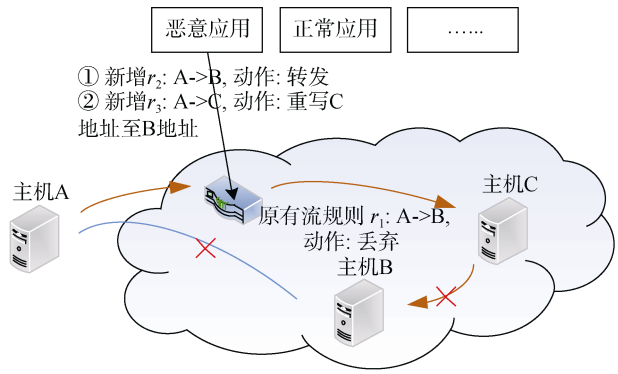


图 6 评估 1-应用越权操作示例

Figure 6 Evaluation 1 - application override operation example

评估 2: 应用安全评级。如评估 1 所示例, SDN 应用平面各应用的可靠性参差不齐, 流规则有可能被恶意应用或因安全性不高的应用被利用而被篡改或混淆。借助知识平面上流知识图谱的实体及边连接关系可对应用的安全性进行评定。如基于知识推理, 评估某应用所生成的流规则如被采用对安全现状的影响, 以图 6 为例, 恶意应用所生成的流规则 r_2, r_3 将会破坏局域网不允许外网访问主机 B 的安全策略, 可推理认为该应用存在恶意嫌疑, 给予较低的安全评级。基于流知识图谱, 可计算获取到更多的应用安全评级要素知识, 如某应用生成流规则被采用比例/被禁用比例, 基于知识连接判定待评级应用与已有评级高/低应用的远近关系等。

评估 3: 流隧道使用。由评估 1 的②新增流规则 r_3 可引申, 单条流规则往往并不违反网络系统安全策略, 但多条流规则合成一条虚拟流规则可能会与现有的流规则及安全策略相冲突。扩展评估 1 中的场景, 设仍存在流规则 r_1: *->B, 丢弃, 但允许外网中的可信主机 A′以向 B 以隧道的方式通信; 优先级取值范围为 0~999, 其中 0 优先级最高, 至 999 优先级顺序递减, 局域网内部授权的流规则优先级为 0~99, 未授权的流规则优先级为 100~999。具体设置如下四条流规则:

表 4 流规则示例 1

Table 4 Flow rule example 1

流规则	动作	优先级
r_1: *->B	丢弃	99
r_2: A->B	设置 A->A′, 写入流表	98
r_3: A'->B	设置 A′->B′, 写入流表	98
r_4: B'->B	转发	98

r_2-r_4 三条流规则合成 A->B 的虚拟流规则, 优先级为 98 高于 r_1 的优先级。该合成流规则表示与内网主机 B 通信, 需经过外网可信主机 A′和内网主机 B′的同时认证。

而假设局域网内部授权生成如下流规则 r_1-r_3, 某恶意应用试图利用流规则 r_2 和 r_3, 其在未授权的情况下写入流规则 r_4, 该 r_2-r_4 三条流规则合成 A->B 的虚拟流规则(A->A′, A′->C, C 重写为 B, A′->B), 通过控制主机发包(A->C)到达内部主机 B。但合成流规则优先级取 r_2-r_4 三条流规则的优先级最低值 100, 低于 r_1 的优先级, 被拒绝执行。

表 5 流规则示例 2

Table 5 Flow rule example 2

流规则	动作	优先级
r_1: *->B	丢弃	99
r_2: A->C	设置 A->A′, 写入流表	98
r_3: A′->C	设置 C->B, 写入流表	98
r_4: A′->B	转发	100

评估 4: 流规则更新。在 SDN 知识平面中, 流规则更新可有两类方式, (1)调整流规则优先级或动作, 使其被执行或不再被执行; (2)新增或删除流规则, 使其生效或失效。对于第(1)种方式, 安全策略可通过其被采用或取消, 如评估 3 中, 第 1 组流规则 r_1-r_4 可通过取消 r_2, r_3, r_4 中的授权而使 A->B 的合成流规则优先级处于 100~999 间, 或使得其转发动作为丢弃包, 从而取消流隧道; 第 2 组流规则 r_1- r_4 可通过授权流规则 r_4 使得其优先级处于 0~99 间, 合成流规则被执行, 增加外网至内部主机 B 的流隧道。对于第(2)种方式, 也可以达到与第(1)种方式同样的效果, 设共有 n 条流规则, 其查询并删除、合并或新增节点的时间效率为 $\log(n)$。如本文第四部分所述, 当新增流规则与现有搜索树 T 中的节点同源-目的地址时, 则选择合并节点; 同时, 合并节点继承新增流规则和合并前节点与流知识图谱 FG 间的关联关系; 当删除节点中部分流规则(即节点中还有未被删除的规则)时, 仅删除合并前该部分流规则与流知识图谱 FG 间的关联关系。

评估 5: 基于流规则的推荐。在本文第五部分算法 5 所述, 当新加入某单元 u, 可通过对一组单元集合 U 的决策图 RGu 合并学习生成流规则决策图 RGU, 但会删除部分单元 u 中所不具备的安全属性节点, 其含义为不处理具备该安全属性标记的流, 当作正常流转发; 如网络流量异常, 但单元 u 并无有效措施处理, 则单元 u 并不生成具备相应安全策略的流规则, 而按照正常流进行规则转发。在此之外, 可针对单元 u 的缺项(即其所不具备的安全属性节点), (1)推荐新增流规则导流至其他具备相应安全策略的单元 u', (2)推荐单元 u 新增网络节点部署。对于第(1)种方法, 针对单元 u 并不具备的缺项, 可以单元 u 为起点广度遍历流知识图谱 FG, 找到可处理缺项、跳数最近或依据网络划分距离最近的单元 u', 生成流规则建议。第(2)种方法可针对某类安全属性标记的流频发且并无较近可处理相关流量的单元, 建议单元 u 新增具备相应安全策略的流处理设施。

评估 6: 安全事件成因分析。结合流附加的安全属性标记以及流知识图谱 FG 中的知识要素信息, 可对安全事件的成因展开进一步的分析及学习。如表 6 所示, 路由震荡是指在一段时间内流表总是频繁变化, 可结合搜索树 T 和流知识图谱 FG 找出在该段时间内所更新的知识连接, 依据相关更新归纳推断其产生原因是否集中到某单元 u, 或集中到某一个或几个应用上; 如集中到单元 u 可进一步依据知识要素推理判断是否因为 u 中某些链路的不稳定导致, 集中到某一个应用上则可推断是否为应用更新过于频繁导致。同样, 节点拥堵可通过流量统计以及流规则集中更新发现定位到拥堵单元 u, 结合流安全属性标记以及流知识图谱 FG 要素信息等推理其成因是否为主机应用服务端的客户端连接数在某一时段过多导致。节点遭受攻击, 也可就流安全属性标记、攻击来源及目的地址及在 FG 中相关联的知识要素信息, 辅助推断其具体的脆弱点及攻击目标信息资产。

表 6 安全事件示例

Table 6 Security event example

现象	成因
路由震荡	链路不稳定/应用更新频繁等
节点拥堵	某一时段上层主机应用的客户端连接数过多等
节点受攻击	节点为某类信息资产的媒介且存在脆弱性等问题

在性能评估上, 本文主要集中在流知识图谱上的关键算法性能衡量上, 即算法 2-算法 4 等对处理延迟时间敏感的算法。我们在开源 SDN 项目 ONOS[22] 上新增知识平面原型并开展实验, 所采用的服务器为 PowerEdge R430 1U 机架式服务主机(Intel Xeon E5-2600v4/32G 内存/4TB SAS 硬盘), 安装 ubuntu 14.10 系统; 由于处理延迟时间随机器配置变化而不同, 本文将统一用相对延迟来描述实验结果, 如相对于某一基础参数设置的处理延迟倍数等。

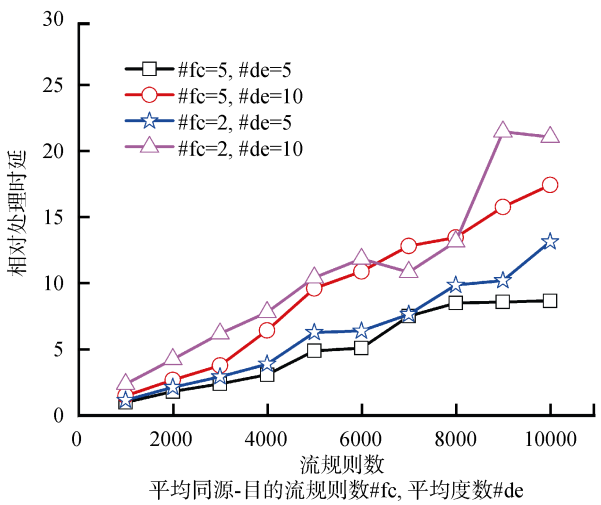


图 7 流规则合成及查找处理时延评估

Figure 7 Flow rule synthesis and lookup processing delay evaluation

针对算法 2 的流规则合成及查找, 参数变量设置为: 分别构建包含 1000 条至 10000 条流规则的搜索树图 T, 具备相同源-目的地址的流规则(#fc)包含平均 2 条、5 条两种情况, 以及搜索树图 T 与流知识图谱 *FG* 建立边连接关系(#de)为平均 5 度、10 度两类。实验以 10 次重复测试平均, 将#fc=5, #de=10 参数设置下处理 1000 条流规则的时延设为 1(实际处理时延约为 0.63 秒)。结果如图 7 所示, 可见(1)处理时延随流规则数增长近似线性增长, 可简单通过多处理并行解决可能的性能瓶颈; (2)#fc 越大表示构建的搜索树图 T 节点数越少, 只需合并现有节点, 处理时延越短; (3)#de 越大表示搜索树图 T 节点需与流知识图谱 *FG* 建立边连接关系越多, 处理时延越长; (4)参数值#de-#de′=5 相比#fc-#fc′=3, 平均对时延的增加程度约为 3.4 倍, 即当流知识图谱 *FG* 规模逐渐增大时, 构建搜索树图 T 的时间相比建立边连接关系的时延比逐渐减小。

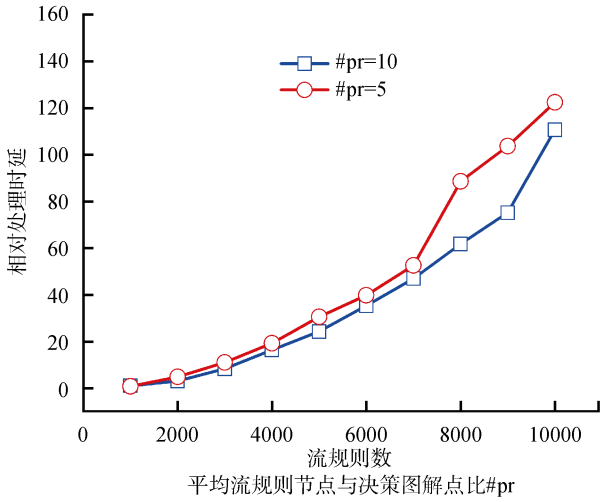


图 8 流决策图构建时延评估

Figure 8 Flow decision graph construction delay evaluation

针对算法 3 的决策图构建, 参数变量设置为: 分别从已构建的具备 1000 条至 10000 条流规则的搜索树图 T 创建决策图, 分为平均 5 条、10 条流规则创建一个决策图节点(#pr)两种情况。实验以 10 次重复测试平均, 将#pr=10 参数设置下处理 1000 条流规则的时延设为 1(实际处理时延约为 0.02 秒)。结果如图 8 所示, 可见(1)决策图生成时延随流规则数增长程度与线性增长接近, 当从 10000 条流规则构建决策图时, 总时间为秒级; (2)#pr 越大表示所决策图节点数越少, 相对处理时延也较短; (3)本文通过比对#pr=5, 10, 以及更多的#pr 参数取值, 发现#pr 的取值对处理时延影响较小, 如#pr-#pr′=5 对时延影响远小于在 9000 条流规则新增 1000 条流规则的影响。

针对算法 4 的流规则决策过程, 参数变量设置为: 分别从已构建的具备 100 个至 1000 个安全属性节点的决策图决策, 一条流规则分别均有 1 条、2 条流安全属性标记(#fl), 是否允许正常转发(#al)。实验以 10 次重复测试平均, 将#fl=1, #al=0 参数设置下处理具备 100 节点决策图的时延设为 1(实际处理时延约为 0.03 毫秒左右)。在#al=1 时, 设流表规则数为 10000 条, 结果如图 9 所示, 可见(1)在决策图规模为千级左右时, 节点数增长对处理时延的影响可忽略不计; (2)流规则标记数增长, 如#fl=1→#fl=2, 处理时延仅有微小增长; (3)当允许流规则正常转发时(#al=1), 基于决策图和安全属性标记的流规则生成对处理时延的影响远小于查询流表规则的时延, 因此在优化决策过程时仍需重点考虑流表查询优化。

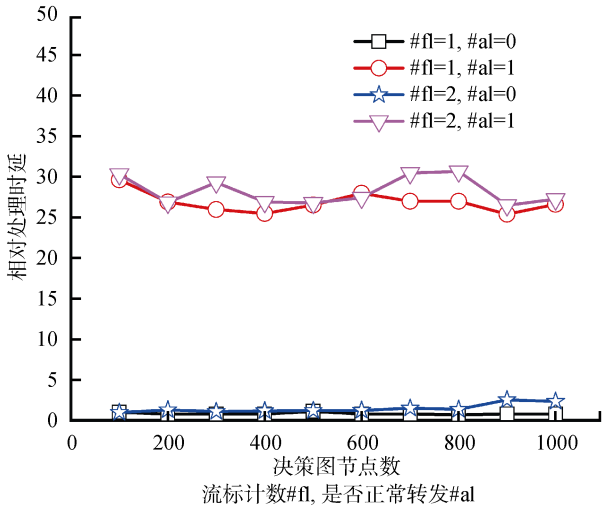


图 9 流规则决策时延评估

Figure 9 Flow rule decision delay evaluation

7 相关工作

本文将知识平面的概念用于 SDN 上, 涉及流知识图谱的构建及其上流规则选择与学习, 该部分将分“基于知识平面的网络”和“基于知识要素的流规则处理及应用”两部分叙述相关工作。

“基于知识平面的网络”: 知识平面的概念在[2]一文中被首先提出, 其观点在于知识平面有助于极大促进网络操作智能化的潜在价值, 如自动化(认知-行动)或推荐(认知-解释-建议)。随后, 多项研究工作也引入了数据知识的理念, 如[23]深入研究了数据驱动网络的潜力, [24]提出针对网络控制及管理的 4D 方法-数据、发现、决策和传播, [25]提出针对网络分布服务的知识平面。SDN[26, 27]也是基于分层平面化的理

念提出, 其从数据平面中分离出控制功能, 形成逻辑上统一的控制平面。近期, 有研究同样基于这一理念, 直接提出知识定义网络的概念[28]。同时, 网络知识平面的构建已具备充分的研究和实践基础, 如可实时获取到包和流粒度上的知识[29], 实时集中网络状态及配置到统一的分析平台[30], 可借鉴日志分析领域的丰富工作[31], 以及可用于网络的学习方法[32, 33]。

“基于知识要素的流规则处理及应用”: 可分为(1)基于知识要素对流规则进行分析和(2)依据流规则知识动态决策两大类。第(1)类研究点较分散, 如可对网络节点建模开展流规则的可达性分析[34, 35], 应用决策图对防火墙流规则进行配置[36], 流规则相关测试集构建[37, 38], 检查流规则更新的一致性[39], 用约束求解和静态知识推理找到 BGP 流规则配置缺陷[40]等。第(2)类研究集中在合规、冲突及缺陷检查上, 如将流表编码至二叉树决策图并用模型检测验证安全属性[41], 将流规则切分成等价类并验证某些网络不变属性是否被满足[42], 通过检测以及解决流规则冲突构建防火墙[43]等。

8 总结及展望

流表及相应流规则构建是 SDN 网络环境下实现安全策略全局协同及动态映射的关键。针对流表构建知识要素分散、难以通过预设策略满足的现状, 本文将知识平面的概念用于 SDN, 以网络流行为与流表安全要素集合的知识图谱相结合, 在流知识本体及相应实体构建的基础上决策生成流规则。在流规则选择决策方面, 合成以同源-目的地址为节点的搜索树图, 并关联扩展流知识图谱。在流规则学习生成方面, 依据流单元划分学习生成给定单元的流规则决策图, 并依据决策图和流安全属性标记学习生成流规则。在评估部分, 从流规则知识、选择及学习等三方面例举评价了流知识平面的实用性以及相关算法性能。

在未来工作中, 将以本文构建的流知识平面作为基础, 深入到具体的应用场景, 实践场景下流知识采集、规整化以及学习的过程, 如流安全属性的实时标记、零知识启动的流规则学习及演进等。

致 谢 在此向对本文工作提出指导的各位同门以及提出建议的评审专家表示感谢。

参考文献

[1] Rossow C, Dietrich C J, and Bos H, “Sandnet: Network traffic analysis of malicious software,”*Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security. ACM*, pp.78-88, 2011.

[2] Clark D D, Partridge C, and Ramming J C, “A knowledge plane for the internet,”*Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications. ACM*, pp.3-10, 2003.

[3] Richard Cyganiak, David Wood, and Markus Lanthaler., “Resource Description Framework (RDF) 1.1 Concepts and Abstract Syntax. Recommendation,” World Wide Web Consortium (W3C), Feb. 2014.

[4] Ankolekar A, Burstein M, and Hobbs J, “DAML-S: Web service description for the semantic web,” The Semantic Web—ISWC 2002, pp.348-363, 2002.

[5] OWL Working Group and others. W, “OWL 2 Web Ontology Language: Document Overview,” W3C Recommendation (27 October 2009). 2012.

[6] Bordes A, Usunier N, and Garcia-Duran A, “Translating embeddings for modeling multi-relational data,”*Advances in neural information processing systems*, pp. 2787-2795, 2013.

[7] Xiao H, Huang M, and Hao Y, “TransG: A Generative Mixture Model for Knowledge Graph Embedding,”*arXiv preprint arXiv*, 2015.

[8] Nickel M, Tresp V, and Kriegel H P, “A three-way model for collective learning on multi-relational data,”*Proceedings of the 28th international conference on machine learning* (ICML-11), pp. 809-816, 2011.

[9] Nickel M, Jiang X, and Tresp V, “Reducing the rank in relational factorization models by including observable patterns,”*Advances in Neural Information Processing Systems*, pp.1179-1187, 2014.

[10] Lao N, Mitchell T, and Cohen W W, “Random walk inference and learning in a large scale knowledge base,”*Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics*, pp.529-539, 2011.

[11] Neelakantan A, Roth B, and McCallum A, “Compositional vector space models for knowledge base inference,”*2015 aaai spring symposium series*, 2015.

[12] Nickel M, Murphy K, and Tresp V, “A review of relational machine learning for knowledge graphs,”*Proceedings of the IEEE, 2016, 104(1)*, pp.11-33, 2016.

[13] Murthy S K, Kasif S, and Salzberg S, “A system for induction of oblique decision trees,” *Journal of artificial intelligence research*, pp.1-32, 1994.

[14] Raileanu L E, “Stoffel K. Theoretical comparison between the gini index and information gain criteria,” *Annals of Mathematics and Artificial Intelligence, 2004, 41(1)*, pp. 77-93, 2004.

[15] Han W S, Lee J, and Lee J H, “Turbo iso: towards ultrafast and robust subgraph isomorphism search in large graph databases,”*Proceedings of the 2013 ACM SIGMOD International*

Conference on Management of Data. ACM, pp.337-348, 2013.

[16] Lai L, Qin L, and Lin X, "Scalable distributed subgraph enumeration," *Proceedings of the VLDB Endowment, 2016, 10(3)*, pp.217-228, 2016.

[17] Chen Y N, and Lin H T, "Feature-aware label space dimension reduction for multi-label classification,"*Advances in Neural Information Processing Systems*, pp.1529-1537, 2012.

[18] Yu K, Yu S, and Tresp V, "Multi-label informed latent semantic indexing,"*Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp.258-265, 2005.

[19] Izenman A J, "Modern multivariate statistical techniques," New York: Springer, 2008.

[20] Porras P A, Cheung S, and Fong M W, "Securing the Software Defined Network Control Layer,"*NDSS 2015*, 2015.

[21] FloodLight, "Open SDN controller," http://floodlight.openflowhub.org/.

[22] Berde P, Gerola M, Hart J, Higuchi Y, Kobayashi M, Koide T, Lantz B, O'Connor B, Radoslavov P, Snow W, and Parulkar G., "ONOS: Towards an open, distributed SDN OS," *In: Proc. of the 3rd Workshop on Hot Topics in Software Defined Networking. Chicago, ACM*, pp.1-6, 2014.

[23] Jiang J, Sun S, and Sekar V, " Pytheas: Enabling Data-Driven Quality of Experience Optimization Using Group-Based Exploration-Exploitation,"*NSDI. 2017*,pp.393-406, 2017.

[24] Greenberg A, Hjalmtysson G, and Maltz D A, "A clean slate 4D approach to network control and management,"*ACM SIGCOMM Computer Communication Review, 2005, 35(5)*,pp.41-54,2005.

[25] Madhyastha H V, Isdal T, and Piatek M, " iPlane: An information plane for distributed services,"*Proceedings of the 7th symposium on Operating systems design and implementation. USENIX Association*, pp.367-380, 2006.

[26] McKeown N, Anderson T, and Balakrishnan H, "OpenFlow: enabling innovation in campus networks,". *ACM SIGCOMM Computer Communication Review, 2008, 38(2)*, pp.69-74, 2008.

[27] "Clean Slate Program," http://cleanslate.stanford.edu/, 2007.

[28] Mestres A, Rodriguez-Natal A, and Carner J, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review, 2017, 47(3)*, pp.2-10, 2017.

[29] Kim C, Sivaraman A, and Katta N, "In-band network telemetry via programmable dataplanes,"*ACM SIGCOMM*., 2015.

[30] Clemm A, Chandramouli M, and Krishnamurthy S, "DNA: An SDN framework for distributed network analytics,"*Integrated Network Management (IM 2015), IFIP/IEEE International Symposium on. IEEE*, pp.9-17, 2015.

[31] Oliner A, Ganapathi A, and Xu W, "Advances and challenges in log analysis," *Communications of the ACM, 2012, 55(2)*, pp. 55-61, 2012.

[32] Bruna J, Zaremba W, and Szlam A, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv*, 2013.

[33] Duvenaud D K, Maclaurin D, and Iparraguirre J, "Convolutional networks on graphs for learning molecular fingerprints,"*Advances in neural information processing systems*. pp.2224 - 2232 , 2015.

[34] Al-Shaer E, Marrero W, and El-Atawy A, " Network configuration in a box: Towards end-to-end verification of network reachability and security,"*Network Protocols, 2009 (ICNP 2009), 17th IEEE International Conference on. IEEE*, pp.123-132, 2009.

[35] Xie G G, Zhan J, and Maltz D A, " On static reachability analysis of IP networks," *24th Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings (INFOCOM 2005.),* pp.2170-2183, 2005.

[36] Liu A X, "Formal verification of firewall policies," *Communications, 2008. ICC'08. IEEE International Conference on. IEEE(ICC'08)*, pp.1494-1498, 2008.

[37] Senn D, Basin D, and Caronni G., "Firewall conformance testing,"*IFIP International Conference on Testing of Communicating Systems. Springer, Berlin, Heidelberg*, pp.226-241, 2005.

[38] El-Atawy A, Samak T, and Wali Z, "An automated framework for validating firewall policy enforcement,"*Policies for Distributed Systems and Networks, 2007,Eighth IEEE International Workshop on. IEEE(POLICY'07)*, pp.151-160, 2007.

[39] Reitblatt M, Foster N, and Rexford J, "Consistent updates for software-defined networks: Change you can believe in!,"*Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, 2011.

[40] Feamster N, and Balakrishnan H, "Detecting BGP configuration faults with static analysis," *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2. USENIX Association*, pp.43-56, 2005.

[41] Al-Shaer E, and Al-Haj S, "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures," *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*, pp.37-44, 2010.

[42] Khurshid A, Zhou W, and Caesar M, "Veriflow: Verifying network-wide invariants in real time," *ACM SIGCOMM Computer Communication Review, 2012, 42(4)*, pp.467-472, 2012.

[43] Hu H, Han W, and Ahn G J, "FLOWGUARD: building robust firewalls for software-defined networks,"*Proceedings of the third workshop on Hot topics in software defined networking, ACM*, pp.97-102, 2014.

游瑞邦 于 2012 年在中国科学院计算技术研究所计算机系统结构专业获得硕士学位。现任中国科学院信息工程研究所助理研究员, 同时在职攻读计算机系统结构专业博士学位。研究领域为计算机系统安全。研究兴趣包括: 系统安全、云数据中心网络安全。Email: yourui-bang@iie.ac.cn

涂碧波 于 2009 年在中国科学院计算技术研究所计算机系统结构专业获得博士学位。现任中国科学院信息工程研究所研究员、博士生导师、中国科学院大学网络空间安全学院岗位教授。研究领域为数据中心安全, 包括操作系统安全、软件定义边界和信息保护等。Email: tubibo@iie.ac.cn

袁子牧 于 2015 年在中国科学院大学计算机系统结构专业获得博士学位。现任中国科学院信息工程研究所副研究员。研究领域为软件数据分析、知识图谱构建。研究兴趣包括: 软件脆弱性、漏洞挖掘、知识图谱构建等。Email: yuanzimu@iie.ac.cn

孟 丹 于 1995 年在哈尔滨工业大学获得计算机系统结构专业博士学位。现任中国科学院信息工程研究所研究员, 所长。研究领域包括高性能计算机体系结构, 高效通信协议, 分布式文件系统与存储服务器, 计算机系统安全, 大数据与云计算安全等。研究兴趣包括: 计算机系统安全、大数据安全、云计算安全等。Email: mengdan@iie.ac.cn