面向间接依赖的数据起源过滤方法

孙连山 欧阳晓通 徐艳艳 王艺星
(陕西科技大学电气与信息工程学院 西安 710021)

摘 要 起源过滤是改造起源图,隐藏起源图中所蕴含的敏感信息的新兴技术。然而,现有的起源过滤研究大多关注节点过滤问题,很少关注边过滤问题,尚未关注并解决间接依赖过滤问题。首先,结合实例阐明过滤间接依赖的动机以及保持溯源效用的挑战,并形式地定义起源间接依赖过滤的目标和约束。其次,扩展针对边的"删除+修复"过滤机制,提出一种面向间接依赖的过滤方法。该方法采用最小代价决策法和贪婪算法设计删除策略,断开与间接依赖对应的所有连通路径,通过在被破坏的非敏感间接依赖端点之间引入非确定依赖关系来修复过滤视图的效用。最后,采用在线开放起源数据集开展模拟实验。实验结果表明,所提方法能在过滤敏感间接依赖的同时保持过滤视图的效用。

关键词 PROV数据模型,数据起源,信息安全,起源过滤,间接依赖
中图法分类号 TP309.2 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.03.025

Novel Sanitization Approach for Indirect Dependencies in Provenance Graph

SUN Lian-shan OUYANG Xiao-tong XU Yan-yan WANG Yi-xing
(College of Electrical & Information Engineering, Shaanxi University of Science & Technology, Xi'an 710021, China)

Abstract Provenance sanitization is a new technology that aims at producing secure provenance views by hiding or redacting sensitive nodes, edges or even indirect dependencies in a provenance graph. However, existing research works mostly focus on sanitizing nodes, rarely on sanitizing edges, not on sanitizing indirect dependencies. To this end, this paper first exemplified the motivations and analyzed the challenges of sanitizing indirect dependencies while keeping utility of provenance views, and formally defined goals and constraints of sanitizing indirect dependencies. Second, this paper proposed a novel mechanism for sanitizing indirect dependencies on the basis of the "Delete+Repair" mechanism for direct dependency in literature. The proposed mechanism includes both deletion rules and repairing rules. Deletion rules specify what edges can be deleted for breaking all connected paths among two end nodes of a sensitive indirect dependency while minimizing the sanitization cost. Repairing rules specify what uncertain dependencies can be added for improving the utility of the sanitized provenance views harmed by applying deletion rules. Finally, a comprehensive sanitization algorithm for sanitizing indirect dependency was implemented and experiments was conducted upon an online open dataset. The experiments results show that the proposed approach can effectively sanitize indirect dependencies while preserving utility of the sanitized provenance view.

Keywords PROV-DM, Data provenance, Information security, Provenance sanitization, Indirect dependency

1 引言

数据起源记录数据在整个生命周期中涉及的数据实体、处理过程以及相关人员和组织等信息[1],可用于数据可信性验证[2]、数据核查和追责[3]以及数据历史版本管理[4]等,在不引起歧义的情况下,下文将其简称为起源。

起源可能包含各种敏感信息,如客户身份信息、银行卡以及支付密码等,敏感信息一旦泄露将造成不可预知的后果,因此,在公开或共享起源供第三方使用之前,必须隐藏其中的敏感信息,以保证起源的安全。然而,起源具有区别于普通数据的特殊性。一方面,起源是记录数据演变历史的元数据[5-6],具有语义不可变性;另一方面,起源常常呈现为有向无环图(简称起源图),数据溯源依赖于起源图的连通路径,具有语义非孤立性。传统的信息安全及隐私保护技术无法满足起源的安全需求[7-8]。

起源过滤又称起源校订[9],是解决起源安全挑战的新兴技术。起源过滤通过改造原始起源图隐藏的敏感信息,来生成安全的过滤视图[10],其面临的主要挑战是在保持起源安全的同时尽可能地满足用户的溯源需求。起源过滤视图通常须满足机密性、语法有效性和语义可用性[11]。机密性表示恶意用户无法根据过滤视图发现或推断被隐藏的敏感信息;语法有效性表示过滤视图应满足起源模型规定的各种结构约束,

到稿日期:2018-04-26 返修日期:2018-06-21 本文受国家自然科学青年基金资助项目(61202019),陕西省教育厅自然科学专项(17JK0087)资助。

孙连山(1977-),男,博士,副教授,CCF会员,主要研究方向为软件安全工程和数据安全与隐私,E-mail:sunlianshan@sust.edu.cn(通信作者);**欧阳晓通**(1993-),男,硕士生,主要研究方向为起源数据安全;**徐艳艳**(1995-),女,硕士生,主要研究方向为起源数据安全;**王艺星**(1993-),女,硕士生,主要研究方向为起源数据安全。

是实现跨组织共享数据起源的前提；语义可用性即溯源效用，表示相对于原始起源图，过滤视图保留溯源语义的程度。

起源图中的节点、节点间的直接依赖（边）乃至间接依赖均可能蕴含敏感信息[7]。现有的起源过滤研究大都关注节点过滤问题，很少关注边过滤问题，尚未关注和解决间接依赖过滤问题。Dey 等提出的起源过滤框架 PROPUB[12]，允许用户声明待过滤的敏感节点，进而选择匿名、删除、抽象和保留等机制过滤敏感信息。Missier 等提出的起源抽象方法 ProvAbs，用一个抽象的实体或活动节点代替一组敏感节点[13]。Hussein 等针对敏感节点制定了一系列图转换规则[14]，满足语法有效性，但也未讨论如何过滤蕴含敏感信息的边或间接依赖。Nagy 等也关注敏感节点过滤问题，提出 ProvS 过滤方法，综合利用匿名和抽象两种过滤机制，提高了过滤视图的安全与效用[15]。与上述方法不同，王艺星等提出了一种高效用起源过滤机制[16]，除过滤敏感节点外，还能过滤敏感边并保留边的端节点，但该方法也未关注和解决间接依赖过滤问题。

总之，现有的起源过滤研究尚未关注和解决间接依赖过滤问题。为此，本文结合实例阐明过滤间接依赖的动机以及所面临的挑战，形式地定义起源间接依赖过滤的目标和约束，在直接依赖过滤机制[16]的基础上，提出一种面向间接依赖的数据起源过滤方法。实验结果表明，该方法能够在有效过滤敏感间接依赖的同时，较好地保持过滤视图的溯源效用。

2　预备知识及问题说明

本节首先介绍标准数据起源模型 PROV 作为研究起源过滤的预备知识，其次阐明过滤间接依赖的动机和挑战，并形式地定义间接依赖过滤的目标和约束。

2.1　起源模型

W3C 于 2013 年发布了包括数据起源模型（PROV-DM）在内的一系列起源相关标准[17]，支持基于万维网的起源发布和共享。根据 PROV-DM 及其前身开放起源模型（Open Provenance Model，OPM）[18]，起源往往呈现为如图 1 所示的有向无环图。起源图通常包含实体（Entity）、活动（Activity）和代理（Agent）3 类节点，分别表示影响数据终态的中间制品、处理过程以及控制和影响处理过程的人或组织。起源图中不同节点之间可能存在 7 种依赖关系，如实体与活动之间的产生关系（Generation）、活动与实体之间的使用关系（Usage）、活动之间的通信关系（Communication）、代理与活动之间的负责关系（Attribution）等。

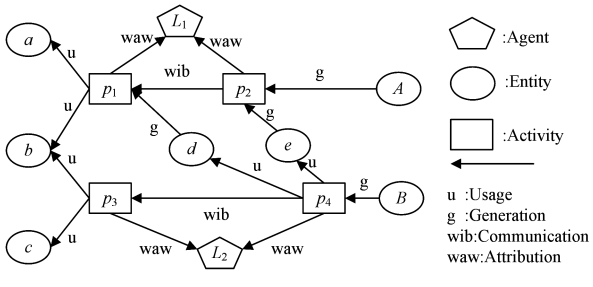

图 1　药品配方起源图

Fig. 1　Provenance graph of drug formulas

图 1 为研究所 L_1 和 L_2 模拟分析药品 A 及 B 的配方起源图。其中，药材 a，b，c、中间产品 d 和 e 以及药品配方 A 和 B 为实体，加工工艺 p_1-p_4 为活动，研究所 L_1 和 L_2 为代理，图中的有向边表示不同节点之间的依赖关系。药品 A 由 L_1 采用加工工艺 p_1 和 p_2 处理药材 a 和 b 得到；药品 B 由 L_2 采用加工工艺 p_3 和 p_4 处理药材 b 和 c 得到。假设研究所 L_1 改进加工工艺，使得 p_1 和 p_2 除生产药品 A 之外，还生产部分副产品用于药品 B 的生产，如中间产品 d 和 e，提升药品 B 的药效。

2.2　间接依赖过滤

起源图的节点、节点间的直接依赖（边）乃至间接依赖（连通路径）均可能蕴含敏感信息。如图 1 所示，为保护知识产权或遵守合作约定，研究所 L_1 可能希望对 p_1 与 p_2 进行匿名处理，隐藏其部分具体细节；L_1 还可能希望隐藏其在研发药品 A 的过程中同时生成 d 和 e 的事实，如直接删除 d 和 e 或隐藏 d，e 分别与 p_1，p_2 之间的依赖关系；L_2 可能期望公布药品 B 对原材料 a，b，c 的间接依赖，但同时期望隐藏 B 对中间产品 d 和 e 以及研究所 L_1 的间接依赖。

现有研究大都关注节点过滤问题，较少研究边过滤问题和间接依赖过滤问题。事实上，间接依赖过滤面临特殊挑战，既要保留间接依赖的两个端点，又要打断二者之间的所有连通路径，同时还要尽可能保留非敏感的间接依赖关系，即尽可能保持过滤视图的溯源效用[16]。

下面形式化地定义起源图及起源间接依赖过滤问题，为探索可能的间接依赖过滤机制奠定基础。

定义 1（起源图 $PG=(V,E,PS)$）　$V=\{v_1,v_2,\cdots,v_n\}$ 为节点集，表示起源图 PG 的 n 个节点；$E=\{e_i \mid e_i=\langle u,v\rangle, u,v\in V, i=1,2,\cdots,m\}$ 为边集，表示 PG 中的 m 条有向边；边 $\langle u,v\rangle$ 表示节点 u 对 v 的直接依赖，称 v 为 u 的直接前因，u 为 v 的直接后果；$PS=\bigcup P(u,v), u,v\in V$，表示起源图中所有间接依赖路径集合。其中 $P(u,v)$ 表示节点 u，v 间所有间接依赖路径的集合，$p_i(u,v)=\{\langle v_{k-1},v_k\rangle\in E, v_0=u, v_j=v, j\geqslant k\geqslant 1\}$ 表示从 u 到 v 的一条连通路径。

定义 2（过滤视图 $PV=f(PG,C)$）　PG 为原起源图；C 为 PG 中待过滤敏感信息集合，可能包含敏感节点、边或者间接依赖；f 表示所采用的过滤操作，则过滤视图 PV 为采用过滤操作 f 过滤 PG 中的敏感信息 C 得到的过滤视图，记为 $PV=f(PG,C)$。

定义 3（溯源效用 $U(PG,PV)$）　起源图中节点、边和连通路径均蕴含着具体的溯源信息，执行过滤操作后，称 PG 中的溯源语义在 PV 中的保留度为 PV 的溯源效用，可记为：

$$U(PG,PV)=g(Sem(PV),Sem(PG)) \tag{1}$$

其中，$Sem(PG)$ 和 $Sem(PV)$ 分别表示原起源图以及过滤视图中所蕴含溯源语义信息的基本统计度量；g 表示融合原起源图及过滤视图的统计度量的过滤视图溯源的效用度量函数。起源图溯源语义的度量方法及溯源效用度量函数多样，例如，Blaustein 等采用原起源图中节点和边的数量等统计度量表示溯源语义，而溯源效用度量函数则定义为原起源图中的节点或边在过滤视图中的保留率[19]。

基于以上对起源图的形式定义，本文形式地定义间接依赖过滤问题如下。

定义 4（间接依赖过滤）　PG 为原起源图，$P(u,v)$ 表示节点 u，v 间的敏感间接依赖路径集合，f 表示可选的过滤操作，

所有可能的过滤操作构成的集合记为 Γ，V' 和 PS' 分别表示过滤视图的节点集合和所有间接依赖路径集合，则间接依赖过滤问题可形式地表示为一个带约束的目标优化问题：

$$\max_{f\in\Gamma} U(PG, f(PG, P(u,v)))$$
$$\text{s.t.} \begin{cases} u,v\in V\cap V' \\ P(u,v)\subseteq PS \\ P(u,v)\cap PS'=\emptyset \end{cases} \tag{2}$$

其中，目标函数表明在过滤敏感间接依赖的同时，应尽可能使过滤视图溯源效用最大化；约束条件表明过滤敏感间接依赖时，应保留其端点元素并断开所有相关连通路径。

现有的典型节点过滤机制（如匿名、删除和抽象等）不能直接实现间接依赖过滤。首先，对连通路径上的节点或边进行匿名处理，不能打断间接依赖路径。其次，简单地删除连通路径上的节点或边，虽能打断间接依赖路径，但可能会破坏其他非敏感间接依赖。图 1 中，删除〈d，p_1〉和〈e，p_2〉能隐藏 B 与 L_1 之间的间接依赖，但同时将破坏 B 与 a 之间的间接依赖。接着，若使用抽象机制将间接依赖路径抽象为一个匿名节点，不但无法满足保留间接依赖端点的需求，而且容易引起过度过滤，造成溯源效用低下。图 1 中，使用抽象方法过滤药品 B 与 L_1 之间的间接依赖，将导致 B，p_4，e，p_2，L_1 等被抽象为一个匿名节点。总之，匿名机制不能打断间接依赖路径，删除和抽象机制均可能破坏非敏感的间接依赖，降低过滤视图的溯源效用，甚至无法满足用户需求。

本文将基于课题组的前期研究工作，扩展针对直接依赖的过滤算法，定义针对间接依赖的过滤规则，实现间接依赖过滤。

3 基本的直接依赖过滤

文献[16]提出了一种“删除＋修复”的直接依赖过滤机制，该机制能大幅提高过滤视图的效用。本节将简要介绍该机制，为间接依赖过滤机制的提出奠定基础。

起源图的基本用途是数据溯源，即沿着连通路径的方向查找可能影响节点当前状态的所有历史节点。这些历史节点的集合称为溯源结果。本文对溯源结果的形式化定义如下。

定义 5（溯源结果 $\Delta_{PG}(u)$） 起源图 $PG=(V,E,PS)$，$u\in V$ 为 PG 中的任意节点，若存在节点 $v\in V$，使得 $p(u,v)\in PS$ 或〈u，v〉$\in E$，则称 v 为 u 的一个可能前因节点，节点 u 的所有可能前因节点构成的集合为其溯源结果，记为 $\Delta_{PG}(u)=\{v\mid p(u,v)\in PS$ 或〈u，v〉$\in E\}$。如图 1 所示，$\Delta_{PG}(A)=\{p_2, p_1, L_1, a, b\}$。

一般地，过滤视图与原起源图的溯源结果之间的差异越小，过滤视图的溯源效用越高[16]，而过滤往往会破坏原起源图中的部分连通路径，使过滤视图的溯源效用降低。在不引入假依赖的前提下，文献[16]采用添加不确定的依赖关系的方法修复连通节点对之间的路径，提高过滤视图溯源效用。所谓不确定的依赖关系是路径长度大于 1 的连通节点对之间可能存在的某种因果依赖关系，其类型由端节点的类型决定。

定义 6（不确定的依赖关系） 原起源图为 $PG=(V,E,PS)$，若删除敏感信息后所得过滤视图为 $PV=(V',E',PS')$，对于任意 $v\in V\cap V'$，满足条件：

$$\begin{cases} u\in\Delta_{PG}(v) \text{且} \langle u,v\rangle\notin E \\ u\notin\Delta_{PV}(v) \end{cases}$$

则称 u 与 v 之间存在某种不确定的依赖关系。

在过滤视图中引入不确定的依赖关系，在本质上是混合不同抽象层的起源记录[18]，在语法层面上修复被破坏的连通路径，从而提高过滤视图的溯源效用。

直接依赖的过滤使用了一种“删除＋修复”的高效用起源过滤机制，即依据过滤规则首先删除敏感边以保证起源安全，然后添加不确定的依赖关系修复被破坏的非敏感连通路径。过滤规则与边的端节点有关，包括删除规则和修复规则。删除规则要求删除敏感边并保留边的端节点，而修复规则要求依据敏感边的类型确定不确定的依赖关系及其类型，例如在图 1 中，L_2 通过隐藏加工工艺 p_1 与其副产品 d 的依赖关系来保护其知识产权，根据生成边的过滤规则，首先删除边〈d，p_1〉，然后在 p_4 与 p_1 之间添加一条不确定的通信边，过滤视图如图 2 所示。

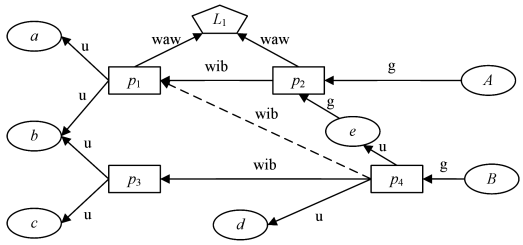

图 2 直接依赖过滤示例

Fig. 2 Example of sanitizing direct dependency

4 间接依赖过滤方法

间接依赖是起源的重要组成部分，可用于追踪影响数据对象当前状态的其他数据、处理过程以及人员组织等。过滤敏感间接依赖是一种重要的起源安全需求，其目标是隐藏敏感间接依赖并使过滤视图溯源效用最大化。本节借鉴直接依赖过滤的思想，扩展“删除＋修复”机制，设计间接依赖过滤方法，包括设计间接依赖过滤规则并实现相应算法。

与文献[16]提出的直接依赖过滤机制类似，本文提出的间接依赖过滤方法也包括两个阶段。首先，在删除阶段，采用基于最小代价决策法和贪婪算法的删除策略，选择并删除敏感连通路径上的某一条边，断开两节点之间的连通路径，达到隐藏间接依赖关系并尽可能地减少溯源效用损失的目的；其次，在修复阶段，在过滤视图中引入不确定依赖关系，修复因删除边而被破坏的非敏感连通路径，进一步提高溯源效用，从而使 $U(PG,PV)$ 达到最大值。

4.1 删除

本文采用删除边的方式断开敏感间接依赖路径 $p(u,v)$，隐藏节点 u 和 v 之间的间接依赖。事实上，$p(u,v)$ 上存在多个节点和多条边，理论上可通过删除 $p(u,v)$ 上除节点 u 和 v 以外的任意节点或边来断开 $p(u,v)$。但根据起源图结构约束，节点被删除后必须删除与其邻接的边。与删除边相比，删除节点往往会造成更大的溯源效用损失，因此本文采用删除边的方式断开 $p(u,v)$。更进一步，路径 $p(u,v)$ 上的边可分为两部分，一部分边为其所独有，另一部分边被多条连通路径共有。这意味着，删除不同边时，原起源图被破坏的连通路径数不同，溯源效用损失也不同。本文期望筛选出溯源效用损失最小的边作为待删除边以断开 $p(u,v)$。为此，本文将过滤

视图中连通路径的减少率定义为过滤代价，用于衡量删除边带来的溯源效用损失，称其为边的删除代价。边的删除代价与过滤视图的溯源效用负相关，即边的删除代价越小，所得过滤视图的溯源效用就越大。设 PG 中的连通路径数为 P，PV 中的连通路径数为 P_S，则删除代价的计算如式(3)所示：

$$Cost=\frac{P-P_S}{P} \tag{3}$$

为尽可能地减少因断开敏感连通路径而引起的过滤视图溯源效用损失，本文提出了一种基于最小代价决策法和贪婪算法的删除策略。在起源图 PG 中，若敏感间接依赖为 $P(u,v)$，则删除策略如下：首先，计算 $p_i(u,v)\in P(u,v)$ 上每条边的删除代价，选取删除代价最小的边作为 $p_i(u,v)$ 上的待删除边；其次，根据贪婪算法，将所有待删除边组成的集合作为“删除”阶段的最优可行解；最后，从 PG 中删除所有待删除边，隐藏敏感间接依赖 $P(u,v)$。

4.2　修复

为了弥补因断开敏感间接依赖路径造成的溯源效用损失，使间接依赖过滤目标函数 $U(PG,f(PG,P(u,v)))$ 取得最大值，本文引入不确定的依赖关系修复被多过滤的非敏感连通路径，包括除过滤目标以外被破坏的非敏感连通路径。记节点 u 和 v 之间的敏感间接依赖路径集为 $P(u,v)$，$p(u,v)\in P(u,v)$，$\langle s,t\rangle$ 为 $p(u,v)$ 上的待删除边，为简化修复操作，本文只对以下两种情况进行修复：

1)打断路径 $p(u,v)$ 将破坏 u 的直接后果节点与 v 之间以及 u 与 v 的直接前因节点之间路径的连通性，根据节点 u，v 的类型以及 u 的直接后果节点和 v 的直接前因节点的类型，选择并应用适当的修复规则进行修复。

2)若存在 s 的直接后果节点 ss 和 t 的直接前因节点 pt 均不在 $P(u,v)$ 上的情况，则删除 $\langle s,t\rangle$ 会破坏 s 与 pt 之间的依赖路径以及 ss 与 t 之间的依赖路径的连通性。这时应根据节点 s 和节点 t 的类型以及节点 pt 和 ss 的类型，选择并应用适当的修复规则进行修复。

具体的修复规则如表 1 所列。

表 1　间接依赖修复规则

Table 1　Repairing rules of indirect dependency

$T(u)$	$T(v)$	情况类型	修复规则
Act	En	$T(pv)=Act$	$addEdge(u,pv)$
		$T(pv)=En$	$addEdge(u,pav)$
En	Act	$T(su)=Act$	$addEdge(su,v)$
		$T(su)=En$	$addEdge(sau,v)$
En	En	$T(su)=Act$ $T(pv)=Act$	$addEdge(su,pv)$
		$T(su)=Act$ $T(pv)=En$	$addEdge(u,pv)$
		$T(su)=En$ $T(pv)=Act$	$addEdge(su,v)$
		$T(su)=En$ $T(pv)=En$	$addEdge(u,pv)$ 或 $addEdge(su,v)$
Act	Act	$T(su)=Act$ $T(pv)=Act$	$addEdge(su,v)$ $addEdge(u,pv)$
		$T(su)=En$ $T(pv)=En$	$addEdge(u,pav)$ 或 $addEdge(sau,v)$

设节点 u,v 之间存在从 u 到 v 的依赖关系，其中映射 $T:V\rightarrow\{En,Act\}$ 将节点 n 映射为其类型，若 $n\in V$，则 $T(n)\in\{En,Act\}$。pv 和 pav 分别属于节点 v 的直接前因节点集合和节点 v 所在连通路径上距其最近的前因活动节点集合。su，sau 分别属于节点 u 的直接后果节点集合和节点 u 所在连通路径上距其最近的后果活动节点集合。注意，表中未出现的情况不用修复或无法修复。

如图 3 所示，设原起源图 PG 中 a_2 与 e_1 之间存在敏感间接依赖，若通过删除连通路径 $p(a_2,e_1)$ 上的边 $\langle e_3,a_1\rangle$ 实现敏感间接依赖的过滤，得到过滤视图 PV，则需：

1)根据 $T(u)=T(a_2)=Act$，$T(v)=T(e_1)=En$，$T(su)=T(e_4)=En$，$T(pv)=T(a_0)=Act$，由表 1 可知，可通过添加一条类型为 Communication 的不确定边 $\langle a_2,a_0\rangle$ 修复节点 a_2 与 a_0 之间的连通性，而节点 e_1 与 e_4 之间的连通性则无法修复。

2)根据 $T(u)=T(e_3)=En$，$T(v)=T(a_1)=Act$，$T(su)=T(a_3)=Act$，$T(pv)=T(e_2)=En$，由表 1 可知，可通过添加一条类型为 Communication 的不确定边 $\langle a_3,a_1\rangle$ 修复节点 a_3 与 a_1 之间的连通性，而节点 e_3 与 e_2 之间的连通性则无法修复。

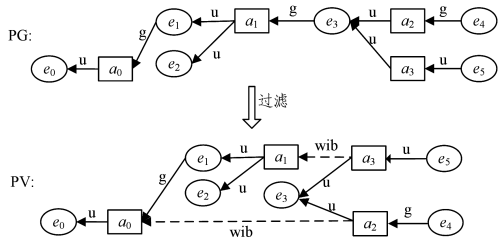

图 3　间接依赖修复示例

Fig. 3　Example of repairing indirect dependency

4.3　整体算法

设原起源图 PG、过滤视图 PV、节点 u 和 v 之间存在敏感间接依赖。依据间接依赖过滤规则设计间接依赖过滤算法(Sanitizing Indirect Dependency，SID)，如算法 1 所示。

算法 1　间接依赖过滤

输入：PG，u，v

输出：PV

```
1. SID(PG,u,v)
2. PV=PG //初始化过滤视图
3. P=sumPath(PG)//统计 PV 中的连通路径数
4. P(u,v)=getPathSet(PG,u,v)
5. subN(PV,u)//获取 u 的直接后果节点集
6. preN(PV,v)//获取 v 的直接前因节点集
7. for each p_i(u,v) ∈P(u,v) &&P(u,v) ≠Ø do
8.    temp=PG
9.     minCost=MAX //初始化最小代价
10.    for each ⟨s_j,t_j⟩∈ p_i(u,v) do
11.      delEdge(temp,s_j,t_j)//删除边⟨s_j,t_j⟩
12.      P_S=sumPath(temp)
13.      Cost_i'=(P−P_S) /P
14.      if minCost > Cost_i  then
15.      minCost=Cost_i
16.      ⟨s,t⟩=⟨s_j,t_j⟩ //代价最小的边
17.      end if
18.    end for
19.    subN(PV,s) //获取 s 的直接后果节点集
20.    preN(PV,t) //获取 t 的直接前因节点集
21.    delEdge(PV,s,t) //从 PV 中删除边⟨s,t⟩
```

```
22.   for each pt∈ preN(t),ss ∈subN(s) do
23.       if pt,ss 不在 P(u,v)上 do
24.       repairIndDep(PV,s,t,pt,ss)
25.       end if
26.   end for
27. end for
28. for each pv∈preN(v),su∈subN(u) do
29.   repairIndDep(PV,u,v,pv,su)
30. end for
31. return PV
32. end SID
```

算法 1 中,第 2－3 行初始化过滤视图并统计 PG 中的连通路径数;第 4 行获取 u 和 v 之间的敏感间接依赖路径集;第 5－6 行获取节点 u 的直接后果节点集和节点 v 的直接前因节点集;第 7－27 行筛选出各连通路径上删除代价最小的边,将其删除并修复,其中第 10－18 行筛选出连通路径 $p_j(u,v)$ 上删除代价最小的边$\langle s,t\rangle$,第 21 行在 PV 中删除边$\langle s,t\rangle$,第 22－26 行根据修复规则修复因删除边$\langle s,t\rangle$而被破坏的非敏感连通路径;第 28－30 行根据修复规则修复因断开节点 u 和 v 之间敏感间接依赖而被破坏的非敏感连通路径。

算法 1 中,第 22－26 行的修复操作耗时最大。设原起源图 PG 中有 n 个节点,$preN(t)$和 $subN(s)$两个集合最多形成 n^2 种情况,则匹配修复规则的时间复杂度为 $O(n^2)$;起源图中待过滤敏感间接依赖路径条数通常远小于节点数 n,因此第 7 行外层循环时间复杂度可视为常数;第 23 行判断节点在敏感间接依赖路径上的时间复杂度是否为 $O(n)$,而由于第 24 行修复操作的时间复杂度通常仅与规则集的大小相关,因此算法 1 的平均渐进时间复杂度为 $O(n^3)$。

5 实验与结果分析

本节通过模拟实验,对比分析本文提出的间接依赖过滤方法 SID 与传统的抽象过滤方法 ProvAbs[13] 之间的差异。首先介绍了实验所用的数据集、实验环境以及参数设置,然后进行实验并分析实验结果。

5.1 实验设置

本文的实验数据来源于美国印第安纳大学模拟产生的 Animation,Ncfs 和 Scoop 等工作流起源数据[20]。数据仅包含实体和活动两类节点。本实验选取其中的 5 类工作流起源数据,相应地设置了 5 组实验,分别使用 SID 方法和 ProvAbs 方法过滤相应起源图中的敏感间接依赖。实验环境为 Lenovo G480 笔记本电脑,Intel Core i5-3230M 2.60 GHz CPU,4 GB 内存,64 位 Windows 10 操作系统。算法在开源图处理工具包 JGraphT[1)] 的基础上,用 Java 语言实现。本文使用文献[16]中的溯源效用评估模型计算过滤视图溯源效用,参数设置为 $\alpha=\beta=0.9$,$\gamma=0.2$,$\theta=\mu=0.2$,$\omega_1=\omega_2=\omega_3$。其中,$\alpha$ 和 β 分别为匿名节点和抽象节点的溯源效用损失率,γ 表示不确定边的溯源效用损失率,θ 和 μ 分别表示匿名路径和不确定路径的溯源效用损失率,ω_1,ω_2,ω_3 分别为节点溯源效用率、边溯源效用率和路径溯源效用率的权重,且 $\omega_1+\omega_2+\omega_3=100\%$。

1) http://jgrapht.org/

5.2 实验与结果分析

首先,在每一组起源图中模拟 4 对敏感间接依赖,包含实体到实体、实体到活动、活动到实体以及活动到活动 4 种类型;其次,分别使用 SID 和 ProvAbs 方法过滤起源图中的敏感间接依赖,重复执行 1000 次,记录运行时间和过滤视图溯源效用;最后,依据实验结果,从溯源效用和安全性两个方面对比分析 SID 与 ProvAbs 之间的差异。

图 4 为工作流 Nam-wrf 的部分起源图,其中 P_1-P_6 表示活动节点,F_1-F_4 表示实体节点。设 $P(F_2,P_6)$为敏感间接依赖,则过滤 $P(F_2,P_6)$的步骤如下:首先,查找出节点 F_2 与节点 P_6 之间的连通路径 $P(F_2,P_6)=\{\{\langle F_2,P_4\rangle,\langle P_4,P_6\rangle\},\{\langle F_2,F_4\rangle,\langle F_4,P_6\rangle\},\{\langle F_2,F_4\rangle,\langle F_4,P_4\rangle,\langle P_4,P_6\rangle\}\}$;其次,在每一条连通路径上选出删除代价最小的边并将其删除;最后,修复被破坏的非敏感连通路径。以连通路径$\{\langle F_2,P_4\rangle,\langle P_4,P_6\rangle\}$为例,删除边$\langle F_2,P_4\rangle$后,除修复 P_2 与 P_5 之间的连通性外,还要修复节点 P_2 与节点 P_4、节点 F_1 之间的连通性,以提高过滤视图的溯源效用。

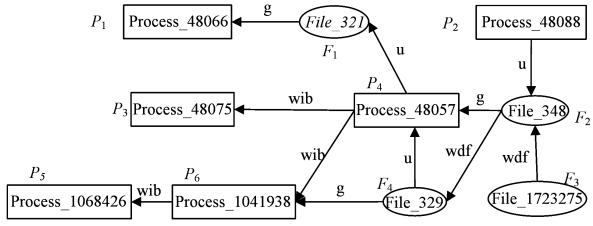

图 4 Nam-wrf 起源图

Fig. 4 Provenance graph of Nam-wrf

针对 Animation,Ncfs,Gene2life,Scoop,Nam-wrf 等实验数据进行相同的实验。

5.2.1 过滤视图溯源效用分析

各组实验的过滤视图溯源效用对比结果如表 2 所列。

表 2 过滤视图溯源效用对比

Table 2 Utility comparison of sanitization views

(单位:%)

Case	U_1	U_2	η
Animation	84.6	35.3	139.7
Ncfs	86.5	63.3	36.8
Gene2life	89.6	65.4	37.5
Scoop	88.2	72.5	21.6
Nam-wrf	88.7	68.9	28.7

表 2 中,U_1 表示使用 SID 过滤敏感间接依赖所得过滤视图的溯源效用;U_2 表示使用 ProvAbs 过滤敏感间接依赖所得过滤视图的溯源效用;η 表示 SID 相对于 ProvAbs 效用提升的百分比。在使用 ProvAbs 处理数据集 Animation 时,过滤视图的溯源效用只有 35.3%,这是因为为了避免违反 PROV 的语法和结构约束,ProvAbs 通常同时抽象敏感间接依赖路径上的所有节点和边以及其周围的非敏感节点和边,造成过度过滤。另外,敏感间接依赖的端点元素也被 ProvAbs 抽象为空节点,不能保留间接依赖的端节点。由 η 的值可以看出,使用本文方法过滤间接依赖所得过滤视图的效用比 ProvAbs 高了 30%左右,模拟实验结果表明,所提方法能够在有效过滤敏感间接依赖的同时,保持过滤视图的溯源效用。

5.2.2　过滤视图安全性分析

对比原起源图和过滤视图可知，采用本文提出的 SID 方法所得的过滤视图不包含敏感的间接依赖关系，具有基本的安全性。与 ProvAbs 相比，采用 SID 方法得到的过滤视图中保留的溯源信息更丰富，但也面临着更高的推理攻击风险，一些攻击者可能利用领域知识推理并恢复被过滤掉的敏感间接依赖关系。本文关注于在保证过滤视图基本安全的情况下，引入不确定的依赖关系，提升过滤视图的溯源效用。未来的研究工作将形式地建立过滤视图的安全威胁模型，设计安全与溯源效用权衡的起源过滤技术。事实上，一种思路是在引入不确定依赖关系的基础上，对相关节点进行匿名化处理，以达到提高过滤视图安全性的目的，但相关问题和解决方案的深入研究和探讨超出了本文的范围。

5.2.3　SID 和 ProvAbs 性能分析

图 5 给出了 SID 与 ProvAbs 的性能对比结果，其中纵轴表示对 Animation，Ncfs，Gene2life，Scoop，Nam-wrf 5 组数据集分别使用 SID 和 ProvAbs 重复运行 1000 次的耗时。实验结果表明，在性能上，SID 与典型的 ProvAbs 接近，证明本文提出的方法在实际中是可行的。

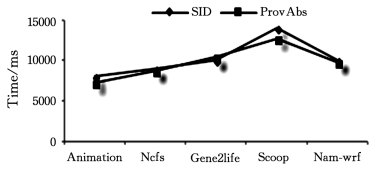

图 5　SID 与 ProvAbs 的性能对比结果

Fig. 5　Performance comparison between SID and ProvAbs

结束语　为解决现有研究无法过滤敏感起源间接依赖的问题，本文扩展基于“删除＋修复”思想的直接依赖过滤机制，提出一种面向间接依赖的数据起源过滤方法。实验结果表明，该方法能够在有效地过滤敏感间接依赖的同时，维持较高的溯源效用，保证了过滤视图的机密性和有效性。未来工作包括对间接依赖过滤算法进行优化、建立过滤视图安全评估模型、研究安全与效用权衡的综合过滤方法。

参 考 文 献

[1] MING H，ZHANG Y，FU X H. Survey of Data Provenance [J]. Journal of Chinese Computer Systems，2012，33(9)：1917-1923. (in Chinese)
明华，张勇，符小辉. 数据溯源技术综述[J]. 小型微型计算机系统，2012，33(9)：1917-1923.

[2] GURJAR K. Comparative Study of Evaluating Trustworthiness of Data Based on Data Provenance[J]. Journal of Information Processing Systems，2016，12(2)：234-248.

[3] TAN A Y S，KO R K L，HOLMES G，et al. Provenance for cloud data accountability[M]. The Cloud Security Ecosystem. 2015：171 -185.

[4] KOOP D. Versioning Version Trees：The Provenance of Actions that Affect Multiple Versions[C]// International Provenance and Annotation Workshop (IPAW). Berlin：Springer International Publishing，2016：109-121.

[5] SUN L S，QI Z B，HOU T. A UML model-based analysis approach for provenance-aware access control policies[J]. Computer Engineering& Science，2015，37(6)：1114-1126. (in Chinese)
孙连山，祁志斌，侯涛. 一种基于 UML 模型的起源感知访问控制策略分析方法[J]. 计算机工程与科学，2015，37(6)：1114-1126.

[6] BRAUN U，SHINNAR A，SELTZER M. Securing provenance [C]// Proc of the 3rd USENIX Workshop on Hot Topics in Security. California：USENIX Association，2008：21-25.

[7] DAVIDSON S B，ROY S. Provenance：Privacy and Security [M]. Encyclopedia of Database Systems. Berlin：Springer，2017.

[8] TORRA V，NAVARRO-ARRIBAS G，SANCHEZ-CHARLES D，et al. Provenance and Privacy[C]// Modeling Decisions for Artificial Intelligence. Cham：Springer，2017：3-11.

[9] CADENHEAD T，KHADILKAR V，KANTARCIOGLU M，et al. Transforming provenance using redaction[C]// ACM Symposium on Access Control MODELS and Technologies. Innsbruck：ACM，2011：93-102.

[10] SHI L B，SUN L S，WANG Y X. Survey of data provenance security [J]. Application Research of Computers，2017，34(1)：1-7. (in Chinese)
石丽波，孙连山，王艺星. 数据起源安全研究综述[J]. 计算机应用研究，2017，34(1)：1-7.

[11] HASAN R，SION R，WINSLETT M. Introducing secure provenance：problems and challenges[C]// ACM Workshop on Storage Security and Survivability. Alexandria：ACM，2007：13-18.

[12] DEY S C，ZINN D. PROPUB：towards a declarative approach for publishing customized，policy-aware provenance[C]// International Conference on Scientific and Statistical Database Management. Portland：Springer，2011：225-243.

[13] MISSIER P，BRYANS J，GAMBLE C，et al. ProvAbs：Model，policy，and tooling for abstracting PROV graphs[C]// Proc of the 5th International Provenance and Annotation Workshop (IPAW) on Provenance and Annotation of Data and Processes. Cologne：Springer，2014：3-15.

[14] HUSSEIN J，MOREAU L，SASSONE V. Obscuring Provenance Confidential Information via Graph Transformation[C]// IFIP International Federation for Information Processing，IFIPTM 2015，IFIP AICT 454. 2015：109-125.

[15] NAGY N，MOKHTAR H M O，EL-SHARKAWI M E. A Comprehensive Sanitization Approach for Workflow Provenance Graphs[C]// International Workshop on Privacy and Anonymity in the Information Society. Bordeaus：CEUR，2016：9-16.

[16] WANG Y X，SUN L S，SHI L B. A Provenance Sanitization Mechanism for Highly Utility[J]. Computer Engineering，2018，44(3)：144-150. (in Chinese)
王艺星，孙连山，石丽波. 一种高效用数据起源过滤机制[J]. 计算机工程，2018，44(3)：144-150.

[17] MISSIER P，BELHAJJAME K，CHENEY J. The W3C PROV family of specifications for modelling provenance metadata[C]// Proc of the 16th International Conference on Extending Database Technology. Genoa：ACM，2013：773-776.

[18] KWASNIKOWSKA N，MOREAU L，BUSSCHE J V D. A Formal Account of the Open Provenance Model[J]. ACM Transactions on the Web，2015，9(2)：1-44.

[19] BLAUSTEIN B，CHAPMAN A，SELIGMAN L，et al. Surrogate parenthood：protected and informative graphs[J]. Proceedings of the Vldb Endowment，2011，4(8)：518-525.

[20] CHEAH Y W，PLALE B，KENDALL-MORWICK J，et al. A Noisy 10GB Provenance Database[M]. Business Process Management Workshops. Berlin：Springer，2012：370-381.