Future Generation Computer Systems 49 (2015) 8–27

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Access control and view generation for provenance graphs

Roxana Danger[a], Vasa Curcin[b,*], Paolo Missier[c], Jeremy Bryans[c]

[a] *Department of Computing, Imperial College London, South Kensington, London SW7 2AZ, UK*
[b] *Department of Primary Care and Public Health Sciences, King's College London, London SE1 3QD, UK*
[c] *School of Computing Science, Newcastle University, Newcastle NE1 7RU, UK*

ARTICLE INFO

Article history:
Received 8 September 2013
Received in revised form
15 December 2014
Accepted 22 January 2015
Available online 17 February 2015

Keywords:
Provenance
Semantic Web
Access Control Language

ABSTRACT

Data provenance refers to the knowledge about data sources and operations carried out to obtain some piece of data. A provenance-enabled system maintains record of the interoperation of processes across different modules, stages and authorities to capture the full lineage of the resulting data, and typically allows data-focused audits using semantic technologies, such as ontologies, that capture domain knowledge. However, regulating access to captured provenance data is a non-trivial problem, since execution records form complex, overlapping graphs with individual nodes possibly being subject to different access policies. Applying traditional access control to provenance queries can either hide from the user the entire graph with nodes that had access to them denied, reveal too much information, or return a semantically invalid graph. An alternative approach is to answer queries with a new graph that abstracts over the missing nodes and fragments. In this paper, we present TACLP, an access control language for provenance data that supports this approach, together with an algorithm that transforms graphs according to sets of access restrictions. The algorithm produces safe and valid provenance graphs that retain the maximum amount of information allowed by the security model. The approach is demonstrated on an example of restricting access to a clinical trial provenance trace.

1. Introduction

Data provenance refers to the knowledge about data sources and operations carried out to obtain a set of data. Making software systems provenance-aware enables users to investigate data sources and services that produced a particular output from the system, together with the individuals who instigated the requests and received those outputs. In such way, the software data outputs can be audited to establish their exact lineage and assess that correct procedures were followed.

Provenance information can be represented as directed acyclic graphs (DAG) establishing causal relationships between individual nodes. The Open Provenance Model [1] (OPM) is a community standard for provenance description. It defines three basic elements to be linked: artifacts, agents, and processes; and five basic causal dependencies: a process *used* an artifact, an artifact *was generated by* a process, a process *was controlled by* an agent, a process *was triggered by* another process, and an artifact *was derived from* another artifact. Following these basic elements, and others related to temporal dependencies, agents and process roles and accounts, OPM graphs can be drawn to express the causal dependencies amongst the elements interacting in a system. Digital representation of any provenance element is an essential requirement of OPM, to which goal it has adopted URIs as the element definition protocol, and is advocating the usage of RDF and OWL technologies for graph format storage and interpretation [2]. OPM has recently been superseded by the W3C PROV standard [3], which shares its basic structure.

A new set of security considerations arises for provenance data in relation to regulating access to records of a resource, rather than the resource itself, e.g. whether a user is allowed to access the provenance trail of a certain process, or of a certain data item. Even if the data item itself is not accessible to the user, this should not necessarily restrict the user from accessing some information about it—e.g. an auditor may not be allowed to see a patient's full electronic health record, but may see that the patient was entered into a clinical trial. To answer such questions, not only is a novel language needed to specify security constraints for provenance data, but also a new mechanism to allow access to aspects of the provenance graph that provide approved information to answer a query.

* Corresponding author.
E-mail address: vasa.curcin@kcl.ac.uk (V. Curcin).

http://dx.doi.org/10.1016/j.future.2015.01.014
0167-739X/© 2015 Elsevier B.V. All rights reserved.

Access control is a generic term for ensuring that a principal (person, process, etc.) can only access the services or data in a system that they are authorized to. It is typically implemented through security policies that try to enforce a certain protection goal such as to prevent unauthorized disclosure (secrecy) and intentional or accidental unauthorized changes (integrity). In general, authorizations for some resource can be positive or negative, giving rise to two classical approaches: closed policy and open policy. The former adheres to a deny-by-default policy, where access to a resource is only granted if a corresponding positive authorization policy exists. The latter allows all access unless a corresponding negative authorization policy exists. In practice a combined approach is used to support policy exceptions. If multiple policies apply to some resource (e.g. a specific one, and a more generic one) conflict resolution takes place depending on the approach adopted by the system. Typical conflict resolution approaches include denials-take-precedence, most-specific-takes-precedence, priority levels, and time-dependent access.

When the resources to be accessed are contained in a complex structure, i.e. in a tree or a graph, denied resources prevent browsing the whole system and answering certain queries, since alternative, non-restricted paths, are not computed. In order to compute such alternative paths, the initial complex structure needs to be transformed. In particular, transforming provenance graphs makes it possible to: (1) generate views depending on user roles and/or data analysis goals; (2) generate semantically valid views comprised of resources to which access is allowed; (3) obtain the whole set of query results that can be exposed to the user; (4) minimize the time–cost of access evaluation since transformation can be done only once per user, i.e. at connection time; and (5) allow visual graph user-interface, as view are user and task-oriented.

Our goal is to define a control access system for provenance graphs that guarantees maximum information to be retained from resources that the user is authorized to access. An overview of the proposed solution is given in Section 2. The motivating example of clinical trials is given in Section 3. Our formalism for provenance views is given in Section 4. Section 5 introduces the access control language. Sections 6 and 7 present the algorithms for query evaluation and graph transformation. Related work in the field is discussed in Section 8.

2. Provenance graph access control using graph transformations

Provenance graph access needs to utilize semantic aspects of provenance data. To that goal, we have extended a semantically-oriented access control language for provenance [4] to allow graph transformations according to user specifications and provenance inference rules. Thus, we introduce a new language, TACLP (Transformation-oriented Access Control Language for Provenance).

The access control language in [4] is itself an extension of a language defined by Qun Ni et al. [5] which includes a semantic description of subjects (user roles), resources to be accessed, conditions under which restrictions are applied, and four different types of access permissions. Cadenhead et al. [4] added regular expressions for resource and condition descriptions. Our extension allows users to define subgraphs to be transformed, with three different levels of abstractions (namely *hide, minimal and maximal*). Details of TACLP can be found in Section 5.

Using TACLP, a user can define an access policy for their data. This access policy is evaluated against a provenance graph (as described in Section 6), and abstract entities are introduced to replace the nodes to which access is restricted. The graph connections are then rearranged accordingly and the transformed view of the graph is returned for querying and browsing.

The provenance graph transformation is formalized in Section 4. Two different operations are used to transform the graphs: *removal* and *replacement*. The former removes a set of nodes, while the latter replaces them with a new *abstract node*. However applying these operations on an arbitrary set of nodes can introduce false dependencies in the new view, that is, dependencies that were not present in the original graph. To avoid this, the input set of nodes is partitioned in such a way that removal and replacement can be performed over the nodes in each partition without the risk of introducing false dependencies. Such a partition is called a *causality-preserving partition*, and can always be computed, as shown in Theorems 1 and 2 in Section 4. These theorems establish that a causality-preserving partition is such that each element of the partition contains a node whose external causes and effects with respect to the input set of nodes are shared by the rest of nodes in the partition element. External causes (effects) of a node w.r.t. to an input set of nodes are the nearest causes (effects) to the node, that are not in the given input set, but are linked to the node by a path of nodes in the input set.

Finally, an *optimal causality preserving partition* is one with the lowest cardinality. This feature is desirable since each partition element can be further partitioned, and, having this, a hierarchy of partitions could be computed and presented as alternative views to a user. In this paper, the partition hierarchy is not explored, but an optimized algorithm for computing optimal causality preserving partitions, including user-specified restrictions, is described at the end of Section 4. The algorithm is based on Theorem 3 which states the conditions under which a causality-preserving partition is considered optimal.

Although the same concepts can be applied to provenance graphs described using PROV ontology [3] (and to any directed acyclic graph in general), in this paper we use the OPM [2] formalism,[1] as the technology was developed as part of a project based on OPM. In future work, we plan to extend the solution to additional concepts introduced in PROV. Section 8 discusses the proposal in the context of other solutions in the literature.

3. Example: access to an electronic health record and clinical trial systems

This section presents a simple example of access control for the provenance data collected from an Electronic Health Record (EHR) and clinical trial systems. It will be used throughout the rest of the paper to illustrate the access control concepts our model uses and to clarify how our solution deals with them.

The rules governing the access to the clinical and provenance data of this healthcare system example are:

1. **Auditors**. Healthcare system auditors or law enforcement agencies can access the whole provenance graph during the auditing process.
2. **GPs and patients**. Electronic health records and their data provenance can only be accessed by patients during weekends, and by GPs during weekdays.
3. **Active GPs**. Active GPs have access to the provenance data associated with the clinical health records of their patients and its provenance; however,
4. **Clinical trial 1**. If some data comes from a clinical trial, the GP needs to be participant of the trial to see the subgraph associated with that trial.

[1] To be more precise, we define an extension of OPM which introduces an abstract provenance entity and an abstract causal dependence type, for reasons given in Section 4.