Access Control for OPM Provenance Graphs

Roxana Danger, Robin Campbell Joy, John Darlington, and Vasa Curcin

Department of Computing, Imperial College London, London SW7 2AZ

The field of provenance in computer science arose from the need to capture the lineage of software data outputs in an automated manner that is semantically consistent across various applications that participated in producing the said outputs. This vision being outside the capabilities of simple text logs, a series of Provenance Challenges investigated the suitability of different approaches, in the process giving rise to the Open Provenance Model (OPM) [3], currently being reworked into PROV, a W3C standard.

Provenance data brings with it a new set of security considerations, since the access permissions to data and to its full lineage may not always coincide. Clinical trial auditor may not be allowed to see a patient's full electronic health record used to gather data for the trial, but may see the trial results, while the patient owns their health record, but may not necessarily access the full trial information. Given that provenance information is commonly represented, and therefore browsed, as causal graphs, access to any individual node in the graph is potentially affected by other nodes connected to it.

Motivation for our work is to provide access policy language and query evaluation method that will offer to the user the maximum permissible amount of information. To that goal, we define ACLP, an extension to XACML, to support such policy definitions, and introduce graph transformations that hide the restricted graph items from the user.

ACLP[1] takes as its starting point the language defined in [1], which is itself an extension of XACML with regular expressions to represent terms of random depth in provenance graphs. Our extension retains their work, and introduces the *transform* construct to support the new query evaluation strategy. SPARQL 1.1 property paths are used for describing complex graph patterns expressions.

A policy in ACLP (Figure 1) is described by a *target* and an optional *condition*, the *scope*, *effect* and *transform* descriptors associated to the target. Full description of these is available in [1], and we only focus on novel elements.

The *effect* element specifies the intended outcome if the applicable rule matches some part of the provenance graph. It can take four values: *Absolute permit* guarantees access to the graph regardless of other policies' outcomes, *Deny* guarantees that certain parts of the graph will never be accessed by users in the subject element, *Necessary permit* describes parts of the graph that need explicit permission to be accessed, and*Permit* describes those graph segments that can be accessed if there are no other policies denying access.

[1] The full ACLP XSD is available at:
`http://www.doc.ic.ac.uk/~rdanger/aclschema_new.xsd.`

P. Groth and J. Frew (Eds.): IPAW 2012, LNCS 7525, pp. 233–235, 2012.

```
<?xml version="1.0" encoding="UTF-8"?>
<policies>
    <policy>
        <target>
            <subject>ocld:Patient</subject>
            <record>ocld:DiagRecommProcess</record>
            <restriction>ocld:Patient.patId <> ocld:User.id</restriction>
            <restriction>rdf:property*/ocld:Diagnosis/opm:wasGeneratedBy*</restriction>
        </target>
        <scope>transferable</scope>
        <effect>deny</effect>
        <transform>
            <type>subgraph</type>
            <tranformation_scope>ocld:clinicalEvidence</tranformation_scope>
        </transform>
    </policy>
</policies>
```

Fig. 1. Example of access control definition for an EHR system

The *transform* element specifies if and how the graph is transformed to allow access to a subgraph when either *deny* or *necessary permit* policies apply to some nodes within the scope. There are three possible values: *None* denotes that transformations are not allowed and no part of the graph can be returned, *Single* means that the graph may be transformed and modified version returned, and *Subgraph* which also allows graph transformation, and also transfers the access restriction to its children nodes. The scope of this transfer depends on the value of the *transformation_scope* element, which can be either a set of resources, defined through a path query, or 'all' for all reachable nodes.

The example of a fictional EHR system in Figure 1 shows a patient access policy: the patient has no access to any EHRs other than their own, neither to any information associated to a diagnosis that was generated by using an automatic diagnosis recommendation process (*ocld:DiagRecommProcess*) and to the subgraph connecting it with the clinical evidences.

Our query evaluation strategy aims to transform the response graphs so that they conform to the query requestor's authorisation level. In rule conflicts, the strategy takes a *wider-allowed-access-takes-precedence* approach [2], i.e. the algorithm guarantees access to all resources that are not a target of a specific deny rule. The evaluation pseudo-code is shown in Algorithm 1.

To construct the transformed graph, we distinguish between *causality-preserving* and *non-causality-preserving* transformations. The former maintain some causal links between remaining nodes (through previous inferred relations), while the latter change the semantics by removing all connections between some remaining nodes. The algorithm removes all excluded nodes while the overall transformation remains causality-preserving, and when this is not the case, it replaces deleted nodes with the minimal set of *fictitious* artifacts and processes) that act as place-holders for one or more deleted nodes, and maintain the causal dependencies of remaining nodes. This is shown in Algorithm 2.

In this paper, we introduced a novel query evaluation algorithm on provenance data that returns graphs transformed based on user's authorisation levels, and the corresponding extension to XACL to support this in policy definition. The system is currently being implemented.

Algorithm 1. Access policy evaluation

Require: g: OPM graph to access.
$targets$: Applicable targets filtered by user and validity.
Ensure: $accessibleGraph$: subgraph of g to which the access is granted, or null if the access to the graph is denied.

{*Step One: evaluate 'absolute permit'*}
for $target \in targets$ **do**
  if $target.effect =$ '$abs.permit$' **and** $eval(target.cond)$ **then**
    return g
{*Step Two: evaluate 'deny' or 'necessary permit'*}
$accessibleGraph = g$
$excludedNodes = \{\}$
for $target \in targets$ **do**
  if $(target.effect =$ '$deny$' **and** $eval(target.cond)$) **or**
    ($target.effect =$ '$nec.permit$' **and** not $eval(target.cond)$) **then**
    if $target.transform =$ 'no' **then**
      return null
    else
      $excludedNodes = excludedNodes \cup getConflictNodes(target, g)$
$accessibleGraph = transformGraph(g, excludedNodes)$
{*Step Three: evaluate 'permit'*}
for $target \in targets$ **do**
  if $target.effect =$ '$permit$' **and** $eval(target)$ **then**
    return $accesibleGraph$
return null

Algorithm 2. Provenance graph transformation

Require: g: OPM graph to access.
$removedNodes$: set of nodes to be removed
Ensure: g': graph equivalent to g in which all minimal subgraphs associated to the nodes in $removedNodes$ have been transformed.

{Step One: Selection of 'retain' graph nodes}
$g' = g$
for $n \in removedNodes$ **do**
  if $(\exists n_c, n_e, cause(n, n_c), effect(n, n_e) \wedge$
    $\exists n', n' \notin removedNodes, (cause(n, n') \vee effect(n, n'))$ **then**
    $mark(n,$ '$retain$'$)$
{Step Two: Transform}
Delete from g' all $n \in removedNodes, causalityPreserving(n)$ without 'retain'
Replace all consecutive graph nodes without 'retain' in g' with a fictitious artifact if they are all artifacts, or with a fictitious process otherwise
Replace all consecutive artifacts with 'retain' in g' with a fictitious artifact
Replace all consecutive processes with 'retain' in g' with a fictitious process

return g'.

References

1. Cadenhead, T., Khadilkar, V., Kantarcioglu, M., Thuraisingham, B.: A language for provenance access control. In: Proceedings of the First ACM Conference on Data and Application Security and Privacy, CODASPY 2011, pp. 133–144. ACM, New York (2011)
2. di Vimercati, S.D.C., Foresti, S., Samarati, P., Jajodia, S.: Access control policies and languages. International Journal of Computational Science and Engineering 3(2), 94–102 (2007)
3. Moreau, L., Freire, J., Futrelle, J., McGrath, R.E., Myers, J., Paulson, P.: The Open Provenance Model: An Overview. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, vol. 5272, pp. 323–326. Springer, Heidelberg (2008)