Dependency Path Patterns as the Foundation of Access Control in Provenance-aware Systems

Dang Nguyen, Jaehong Park, and Ravi Sandhu
Institute for Cyber Security
University of Texas at San Antonio
dnguyen@cs.utsa.edu, jae.park@utsa.edu, ravi.sandhu@utsa.edu

Abstract

A unique characteristics of provenance data is that it forms a directed acyclic graph (DAG) in accordance with the underlying causality dependencies between entities (acting users, action processes and data objects) involved in transactions. Data provenance raises at least two distinct security-related issues. One is how to control access to provenance data which we call Provenance Access control (PAC). The other is Provenance-based Access Control (PBAC) which focuses on how to utilize provenance data to control access to data objects. Both PAC and PBAC are built on a common foundation that requires security architects to define application-specific dependency path patterns of provenance data. Assigning application-specific semantics to these path patterns provides the foundation for effective security policy specification and administration. This paper elaborates on this common foundation of PAC and PBAC and identifies some of the differences in how this common foundation is applied in these two contexts.

1 Introduction

In a provenance-aware system, transaction information is captured as provenance data. As commonly recognized in the literature, provenance data forms a DAG [9, 10]. Thereby provenance data captures dependency relationships between the graph nodes which represent the provenance entities of acting user, action process and objects. In Open Provenance Model (OPM) [10], provenance data captures causality dependencies (denoted as edges) between these three types of entities (denoted as nodes) involved in transactions. Provenance data can form a chain of dependencies from which useful information such as pedigree, usage and versioning information can be extracted by tracing through dependency chains. These dependencies are often expressible through regular path patterns constructed from graph edges which can then serve as the foundation upon which access control policies and mechanisms are built.

There are at least two access control-related concerns in provenance-aware system. One is Provenance Access Control (PAC) which focuses on how to control access to provenance data. This issue has been studied extensively in recent years [3, 8]. The other is Provenance-based Access Control (PBAC) which focuses on how to utilize provenance data to control access to underlying data that are referred by the provenance data. The latter has not received much attention in the literature but is an important benefit of provenance data.

In this paper, we first discuss foundational concepts for both PAC and PBAC by utilizing regular expression-based path patterns of dependency edges found in provenance data. To achieve simple and effective security policy specification and access control administration, we propose named abstractions of meaningful path patterns found in provenance data.

As shown in Figure 1, while both PAC and PBAC share the same foundation, as they are different in their goals they do present different issues to be addressed. Based on our recent study where we developed our initial models for PBAC, we identify essential issues of PBAC and PAC that need to be addressed for effective access controls [13]. We identify the issues related to PBAC and PAC and explore some potential solutions to address these. We believe our approach will provide a strong foundation for simple but effective access controls in provenance-aware system.

2 Common Provenance Foundation

In a provenance-aware system, we capture a system event of the form $< acting\ user, action, objectList >$ as a transaction where an objectList contains one or more objects. Storing all such transaction records provides the system with base provenance data.[1] From the raw provenance

[1]There is typically additional contextual information associated with an event such as timestamp, location, etc. For simplicity, we omit

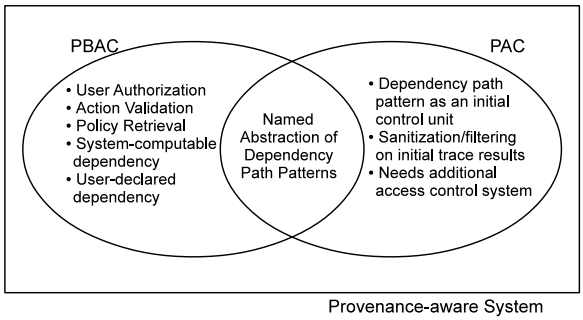

Figure 1: PBAC and PAC Comparison

data we can extract causality dependencies between the entities that form the transaction records. Such causality dependencies are captured and used as the basis of the popular Open Provenance Model [10].

The OPM model identifies direct dependencies as well as indirect dependencies between these entities. These indirect dependencies often carry semantics that reflect the application context of the system. Such semantics can also be extracted from paths of arbitrary lengths of dependencies. We believe capturing these semantics through named abstractions of the associated dependency path patterns provides a strong foundation for access control mechanisms that utilize provenance data. Identification and specification of these abstractions falls within the responsibility and capability of system and security architects. These will necessarily be application specific. Using these named abstractions of dependency paths provides simple and effective methods for retrieving provenance information. In our initial study on provenance-aware group collaboration environment [12], we showed a potential in how provenance data could be utilized for secure group collaboration.

Queries which utilize regular path expressions on DAGs have been addressed by languages such as SPARQL [7] or PQL [1]. These queries can and often do return results that comprise vertices of the underlying graph. Further computations or filterings on and evaluations of these results can be used in the access decision processes. Note that as provenance information changes incrementally over time, the same query can produce different result vertices each time it is invoked. This can significantly influence how the computing or filtering processes are carried out.

These dependency abstractions can also be used as a control unit in policy specifications. The interaction of information retrieval methods and policy specifications can enhance access evaluation both in effectiveness and expressiveness.

As shown in Figure 2, consider an online course management system where a sample set of three transaction records consists of: a student $s1$ submits

such contextual information in this paper.

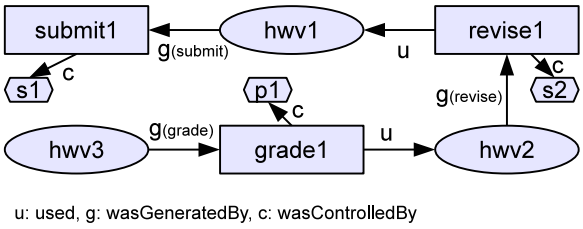

Figure 2: A Grading Example in OPM Graph

a homework document *hwv*1, later a team member *s*2 revises the submitted homework into *hwv*2, which becomes *hwv*3 after being graded by the professor *p*1. Here we can identify a causality dependency path between the graded homework *hwv*3 and the student who submits the original version. Such a dependency path can simply be captured in a regular path that can be assigned an abstraction name such as *isAuthoredBy*. A policy can then be constructed to reflect a rule that only allows a user *s* access to the homework object *hwv*3 if $s \in findVertices(hwv3, isAuthoredBy)$ where $findVertices$ is a function that returns resulting vertices from tracing provenance data using a path pattern defined for *isAuthoredBy* starting from the node *hwv*3. In other words, only the student submitter can access the graded homework object. If the homework document can be revised multiple times by different team member students, *isAuthoredBy* can be defined using the dependency path pattern $(wasGeneratedBy(role : grade).used)?.(wasGeneratedBy(role : revise).used) * .wasGeneratedBy(role : submit).wasControlledBy$, where "?" means 0 or 1 and "*" means 0 or more.

A query that seeks to retrieve original submitter can utilize the named dependency *isAuthoredBy*. At the same time, a policy can make use of this information as a control unit for access control.

3 Provenance-based Access Control (PBAC)

Traditional access control mechanisms often provide access protection on data objects through predefined constructs, e.g., roles in Role-based Access Control (RBAC) and clearances/classifications in Mandatory Access Control (MAC). PBAC aims to provide access control protection on data objects by utilizing the foundational construct of dependency path patterns found in provenance data as described in the previous section. Recently, we have developed our initial access control models for PBAC and identified a set of properties that are crucial for the fundamental access control functionalities [13].

While PBAC can support various access control capabilities that are beyond those available in traditional access controls, we recognize that in a realistic system de-

ployment, PBAC alone may not be sufficient. Combination with other access control mechanism such as RBAC would bring out more capable access control capabilities. Our grading example demonstrates this through the use of user roles (e.g., student, professor).

In PBAC, provenance data can be used for at least three purposes: **provenance-based user authorization, provenance-based action validation,** and **provenance-based policy retrieval**. In our recent study, the developed models incorporate provenance-based user authorization and action validation as part of access evaluation processes while provenance-based policy retrieval remains to be addressed.

By tracing provenance data, a system can evaluate whether a requesting user has or has not done certain activities on object(s) that are related to an access target object. Also, the system can evaluate whether the requested action can be performed on the target object or not. These evaluations are called *user authorization* and *action validation* respectively, and are crucial processes that should be evaluated for all access decisions. In the example from section 2, if only a user who submitted the homework can read graded homework, the system needs to check whether a requesting user has submitted the original homework of a graded version or not (user authorization). In the same example, if only a revised version can be graded, the system must check whether a target object has been revised before regardless of the fact who is requesting the action (action validation). These access evaluation processes provide enhanced capabilities of the system to support dynamic separation of duties or work-flow controls. While several policy languages have been proposed in the provenance-related literature [4, 11], they are not readily suitable for these two evaluation processes. We need a flexible policy language that can effectively specify and express these access control enhancements.

In addition, in case access control policy is not readily available to a system, the system may trace provenance data to retrieve necessary policies. For example, originator control requires to check policies of data originator. By tracing provenance data, the system can discover not only the very initial originator but also all the contributors who contributed to any previous or later versions of a target object from whom necessary access control policies can be retrieved. This provenance-based policy retrieval should be accommodated in provenance-based access control for richer access controls.

Provenance data naturally includes who performed actions on objects. Because of this, dependencies found in provenance data can be either dependency of a particular object or a user. We call these **object dependency** and **acting user dependency**, respectively. Depends on the dependency type, the query on provenance data chooses which path patterns to be used and where the starting node is anchored. It is also worth mentioning that if a query path pattern starts from an acting user and traverses through the associated object from the request, then the rule can be expressed in an equivalent way such that the path pattern starts from that object itself. In this case, there is nothing fundamentally beneficial in recognizing such type of acting user dependencies. On the other hand, if we simply look up all acting user's previous actions and requests history from the provenance data then it is essentially similar to the existing concepts of history-based access control [6]. Nonetheless we feel that the potential benefit of acting user dependencies should be further explored.

OPM identified three direct dependency types and two indirect dependency types and recognized that some indirect dependencies cannot be computed and have to be identified manually. However, it captures direct dependencies and both **system-computable** and **user-declared indirect dependencies** in a single provenance graph. System computable dependencies are derived from the raw provenance data. In addition users can declare a specific dependency between entities using a dependency name that is predefined by the system. In the perspective of access control, we believe these dependencies should be treated separately as they are different in their semantics and characteristics. The system computability of dependency is a critical notion upon which access control models can be built. The potential inclusion of user-declared dependencies poses a different set of concerns. The user-declared dependency may cause conflict with the dependencies with the same name that are computed using base provenance data. One approach to address this issue is by explicitly identifying the intentions of the declaration. For this, we believe there are at least three types of intentions: inclusive, exclusive and denying-intent dependencies. The inclusive-intent approves a user-declared named abstraction dependency such that the vertices reachable by this path is included in the result vertices set obtained by queries using the same name dependency. Similarly, the exclusive-intent approves the user-declared dependency and voids all other dependencies of the same type while the denying-intent voids the target path. Furthermore, PBAC also needs to resolve an authorization issue of who can declare what kind of dependencies and their intentions under which circumstance. We believe the issues identified in this section should be further studied as they are essential for secure provenance-aware systems.

4 Provenance Access Control (PAC)

Similar to PBAC, PAC needs some other access controls in play. Note that while this is true, and perhaps even PBAC can be used for PAC, this could be only a secondary concern of PAC. The main concern of PAC

includes unique issues that access to provenance data presents.

In PAC, one of the most significant differences is that provenance data that users want to access are likely to be captured by tracing the provenance graph using some meaningful dependency paths. Furthermore, users may want to access the information that can be derived from the vertices found as a result of the provenance graph tracing. In this respect, we believe there need multi-layer access control evaluations with different granularities.

At its core, PAC utilizes the dependency path patterns found in provenance data as a unit for access request as well as a control unit used in policy specification. We believe the initial and most essential control should be made through requiring access control to the dependency path patterns themselves. More precisely, access control is done on the semantical named-abstractions of these patterns that can be defined and assigned by system architects. Such assignments can be stored and utilized as a control unit for an access control solution enforced by the system. For example, a student is either allowed or disallowed from asking for the identity of other students who revised his paper. Here, a named abstraction such as *wasRevisedby* can be defined and used as a control unit.

Once the dependency path pattern is allowed, a finer grained control is necessary to determine how much of the resulting information (vertices) should be allowed. This requires vertex-level or type of vertex-level access control policies. For example, a student is allowed to know some revisers but not all or she is allowed to know departments information where the revisers belong but not individual revisers' names. Graph redaction or sanitization processes can be performed on provenance data to achieve this end [2, 5].

In addition, if the requester want to access information that are not available in provenance data but can be derived from the resulting provenance information, how to control access to dependency path patterns and how much of the resulting provenance data should be allowed for an access also have to be considered in PAC. We believe these issues are essential for PAC and necessary to be investigated in depth.

5 Conclusion

In this paper, we identified essential concepts and issues for access control models that utilize the intrinsic characteristic of provenance data that exhibit causality dependencies. To help achieve simple and effective security policy specification and access control administration, we proposed named abstractions of meaningful path patterns found in provenance data. We identified two access control-related concerns in provenance-aware system, PAC and PBAC, and explored some issues that are common to and unique to each. We believe our approach will provide a solid foundation for simple but highly effective access controls in provenance-aware system and the discussion made in this paper is critical for comprehensive and comparative understanding of PAC and PBAC.

References

[1] Pql-path query language. `http://www.eecs.harvard.edu/syrah/pql/`. Accessed: 03/31/2012.

[2] BLAUSTEIN, B., CHAPMAN, A., SELIGMAN, L., ALLEN, M. D., AND ROSENTHAL, A. Surrogate parenthood: protected and informative graphs. *Proc. VLDB Endow. 4*, 8 (May 2011), 518–525.

[3] BRAUN, U., SHINNAR, A., AND SELTZER, M. Securing provenance. In *The 3rd USENIX Workshop on Hot Topics in Security* (Berkeley, CA, USA, July 2008), USENIX HotSec, USENIX Association, pp. 1–5.

[4] CADENHEAD, T., KHADILKAR, V., KANTARCIOGLU, M., AND THURAISINGHAM, B. A language for provenance access control. In *Proceedings of the first ACM conference on Data and application security and privacy* (2011), ACM, pp. 133–144.

[5] CADENHEAD, T., KHADILKAR, V., KANTARCIOGLU, M., AND THURAISINGHAM, B. Transforming provenance using redaction. In *Proceedings of the 16th ACM symposium on Access control models and technologies* (2011), ACM, pp. 93–102.

[6] EDJLALI, G., ACHARYA, A., AND CHAUDHARY, V. History-based access control for mobile code. In *IN ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY* (1998), ACM Press, pp. 38–48.

[7] HARRIS, S., AND SEABORNE, A. Sparql 1.1 query language w3c working draft, jan 2012. `http://www.w3.org/TR/sparql11-query/`. Accessed: 03/31/2012.

[8] HASAN, R., SION, R., AND WINSLETT, M. Introducing secure provenance: problems and challenges. In *Proceedings of the 2007 ACM workshop on Storage security and survivability* (New York, NY, USA, 2007), StorageSS '07, ACM, pp. 13–18.

[9] HEINIS, T., AND ALONSO, G. Efficient lineage tracking for scientific workflows. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2008), SIGMOD '08, ACM, pp. 1007–1018.

[10] MOREAU, L., CLIFFORD, B., FREIRE, J., FUTRELLE, J., GIL, Y., GROTH, P., KWASNIKOWSKA, N., MILES, S., MISSIER, P., MYERS, J., PLALE, B., SIMMHAN, Y., STEPHAN, E., AND DEN BUSSCHE, J. V. The open provenance model core specification (v1.1). *Future Generation Computer Systems 27*, 6 (2011), 743 – 756.

[11] NI, Q., XU, S., BERTINO, E., SANDHU, R., AND HAN, W. An access control language for a general provenance model. In *Proceedings of the 6th VLDB Workshop on Secure Data Management* (Berlin, Heidelberg, 2009), SDM '09, Springer-Verlag, pp. 68–88.

[12] PARK, J., NGUYEN, D., AND SANDHU, R. On data provenance in group-centric secure collaboration. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 7th International Conference on* (oct. 2011), pp. 221 –230.

[13] PARK, J., NGUYEN, D., AND SANDHU, R. A provenance-based access control model. In *10th Annual Conference on Privacy, Security and Trust* (Jul. 2012), PST 2012, IEEE.