Securing Provenance

Uri Braun, Avraham Shinnar*, Margo Seltzer
Harvard School of Engineering and Applied Sciences

Abstract

Provenance describes how an object came to be in its present state. Intelligence dossiers, medical records and corporate financial reports capture provenance information. Many of these applications call for security, but existing security models are not up to the task.

Provenance is a causality graph with annotations. The causality graph connects the various participating objects describing the process that produced an object's present state. Each node represents an object and each edge represents a relationship between two objects. This graph is an immutable directed acyclic graph (DAG). Existing security models do not apply to DAGs nor do they easily extend to DAGs. Any model to control access to the structure of the graph must integrate with existing security models for the objects. We need to develop an access control model tailored to provenance and study how it interacts with existing access control models. This paper frames the problem and identifies issues requiring further research.

1 Introduction

Provenance is meta-data that represents the ancestry of an object. In theory, provenance begins at the Big Bang and describes all operations that brought an object to its current state. In practice, we lack full provenance from the beginning of time, so we speak about provenance in terms of a specified starting state and a collection of transitions through intermediary states that eventually bring it to the state being discussed.

Provenance exists in many applications today and is fundamental to many of them. For example, the accounting transactions reconciled to form a financial report, a lab notebook, the sources supporting an intelligence dossier and patient logs from a medical trial are all comprised of provenance. Typically, each of these examples supports a process whose end result is a concise report. Provenance tracking systems capture the processes leading to these concise reports. The benefits are numerous. Provenance allows scientists to express the processes used to produce results completely and accurately. Executives gain a more transparent view of their organization's financial controls, and investigators are better able to audit those records if they can track the process.

Provenance is often sensitive. Many of the applications described above operate on sensitive data. Companies maintain protections on much of their financial accounting process. Double blind medical trials prevent the doctor from knowing whether a participant is receiving the investigational treatment or is part of the control group. The intelligence community often guards its sources and methods even more securely than the data they produce.

Provenance differs from data and most other meta-data in that it forms a directed acyclic graph (DAG). It is a graph where each node represents an entity and each edge represents a causal relationship between two entities. Examples of an entity include a file on a file-system, a process, a doctor or an experiment. Relationships capture information flow from inputs to outputs. Since time always moves forward, cycles are nonsensical. We may not know whether the chicken preceded the egg but clearly one came first. Provenance is not a tree because an entity may have multiple inputs.

Unlike most data, provenance is immutable. It records history, and history does not change. While provenance describes data — which is presumably changing — the provenance itself is immutable. Commentary on history is often valuable. In addition to the causal graph we have attributes that describe the nodes and edges. Such entities usually have names and other properties. Relationships may also have attributes. Attributes are key value pairs that describe a node or an edge. Annotations may be added but the relationships should not change. This jux-

*This material is based upon work supported under a National Science Foundation Graduate Research Fellowship

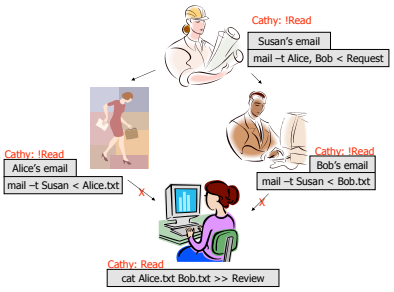

Figure 1: Cathy's performance review. Susan emails Alice and Bob requesting feedback for Cathy's review. Alice and Bob's input are combined to form the review.

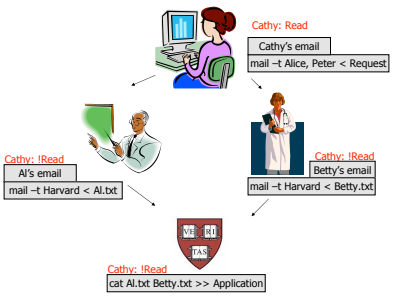

Figure 2: Cathy's college application. Cathy emails Al and Betty requesting a letter of recommendation from each of them. They send their letters, which are combined to form Cathy's application.

taposition of mutable data with an immutable description leads to interesting challenges in designing an appropriate security model.

Provenance is valuable because it allows us to track a result back to its sources. Regardless of how this ancestry is recorded, each relationship reveals information about both parties in the relationship. As we show later, data ancestry can be more or less sensitive than the data itself. Thus provenance security cannot be trivially subsumed by existing security systems.

The remainder of this paper is organized as follows. Section 2 explains why provenance needs its own security model. We then describe the components of a provenance security model in Section 3 and the challenges of interactions between provenance security and data security are discussed in Section 4. Section 5 presents related work followed by our conclusions in Section 6.

2 Provenance needs its own security model

Provenance has particular characteristics that differentiate it from data typically considered by secure systems. We use the term *traditional data* to mean data traditionally considered by secure systems. This includes data stored in file-systems and databases. The sensitivity of provenance and the data it describes may be different. Specifically, it is possible for the data to be more sensitive than the provenance or vice versa. The following examples illustrate this.

An employee's performance review is an example where the provenance is more sensitive than the data (see Figure 1). Such a performance review provides feedback on an employee's work over the previous year. The data in this case is the performance review document. Generally employees are permitted — and usually encouraged — to read their performance review. Thus they are able to read the data. However, the employee is not told who had input in writing the review. Thus the employee can see the data but not all of the provenance of that data.

Letters of recommendation — such as those provided in applications to universities — are an example where the data is more sensitive than the provenance (see Figure 2). Students often select who will write letters on their behalf. In some cases the student may even receive the sealed envelope and send it. Thus the student has access to their application's provenance, including who wrote the letter, when it was sent and where it was sent. Students generally cannot read the contents of the letter. In this case, the data is more sensitive than the provenance of that data.

Clearly we need different security settings for the data and the provenance. Our initial idea was to treat provenance just like traditional data. As we shall see, that does not work. Our next thought was to use two instances of traditional security models: one for the data and one for the provenance. This too appears to be problematic.

Provenance is poorly served by traditional data security models, because they focus on individual data items whereas provenance focuses on the relationships between those items. We need to secure the edges in the graph. These relationships form a graph and it is the structure of that graph that is our focus.

Since provenance forms a DAG, our model does not need to deal with cyclic graphs. However, models that are limited to trees are probably not expressive enough. Provenance queries include path traversal both up and down the graph. One node can have multiple ancestors and multiple descendants. To represent this in a tree, we would need one tree to store the ancestors and another for the descendants. Maintaining duplicate copies of all the data and keeping permissions consistent across both copies is fraught with difficulty. As a result there is no

obvious way to adapt existing tree based models for use with provenance.

One option would be to translate the DAG into a tree by copying the subgraphs rather than having junctions (see Figure 3 and Figure 4). Thus if Alice and Bob provide data to Cathy, make two copies of Cathy, one as a child of Alice and one as a child of Bob. This would introduce exponential growth, which is incompatible with the desire for a long running recording of history. The security implications are also problematic as the security system would need to consider the various copies as one object.

Since provenance is not a tree, a rooted path is not a unique identifier for a node. Our experience with provenance has shown that paths are important. While paths are a collection of individual edges, it is not clear if they can be secured as such collections or if more complex capabilities are needed.

Another distinguishing feature of provenance is that it is immutable. Additions are likely. In the medical records context, a patient we have not seen in a long time may undergo additional treatment. Annotations can also be added later. These additions are valid. It is not valid to modify the treatments a patient underwent after the fact. We should be able to take advantage of the immutable nature of provenance when designing the security model.

Provenance is often created as a side-effect whereas most models assume that the data is created explicitly by the user. While it is not clear what impact, if any, this has on the model, it likely means that default permissions become important. We will likely want to have a rich method for specifying the initial permissions for new nodes and edges.

As we have described, provenance is about relationships and while there has been much work on securing data items, there is significantly less on securing relationships between multiple items. While it is tempting to think of a provenance security model as additional controls that interact with a traditional security model, it is not clear how to effect such a composition. Nonetheless, we will explore the idea of multiple coexisting models in the next section.

3 Data, ancestry and attributes

Provenance consists of an annotated ancestry graph that records the history of some data. A system to secure provenance conceptually consists of three subsystems. Figure 5 represents this graphically.

We assume the data is already protected using an existing access control system. Such a system may be either mandatory or discretionary. Options include role-based access control (RBAC) [4] and the various database security models. Another option is to use the UNIX file-system security model, as is done in the PASS project (Section 5). While there may be repercussions to choosing a specific option, our focus is on how provenance security interacts with data security.

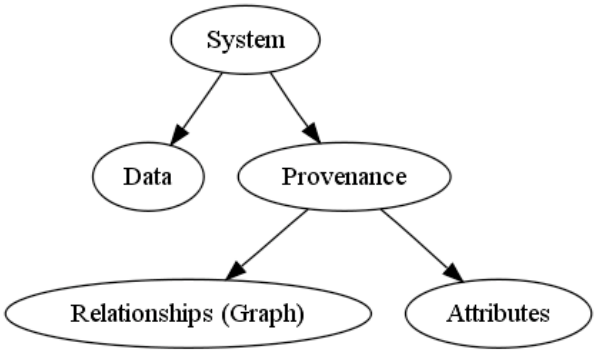

Figure 5: The system consists of "traditional data" and provenance. The provenance is further broken down into attributes and relationships. Relationships are the edges in the causal graph and therefore capture the structure.

Attributes that describe the provenance entities can be protected using traditional means. The access rights to the attributes will likely differ from those of the data. We view attributes as key-value pairs. So while they may need their own set of policies, attributes can be protected by traditional data security models. Thus the models described in the previous paragraph ought to apply here.

It is less clear how we can go about securing the graph itself. Going back to the employee performance review, the manager should know which employees had input into the review but the employee being reviewed should not. Human resources and higher level managers may or may not be granted access to this information. It is not clear how to express these limitations or even what primitives are necessary.

4 Putting it back together

In addition to securing the provenance causality graph, there are other challenges we need to address. In describing the provenance security model it was convenient to divide the system into three parts. It is less clear how to put the pieces back together again. So far we are unable to fully express these interactions. Instead we have a few examples to demonstrate that interactions do, in fact, exist.

4.1 Hiding participation

Often we want to hide the participation of an entity. In the performance review we might not want Cathy to know that Alice participated in her review. Clearly the

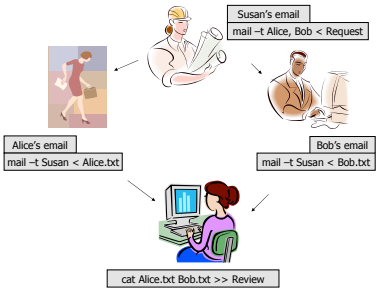

Figure 3: Cathy's performance review as a DAG.

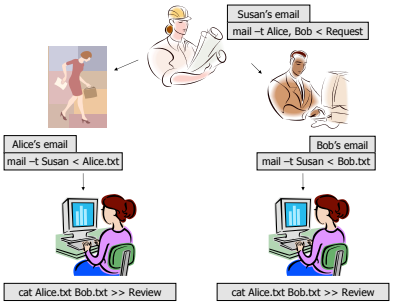

Figure 4: Cathy's performance review transformed into a tree.

data should not contain Alice's name or other identifying information. We may believe that it is acceptable for Cathy to know that four people were involved in her review. Of course, if Cathy has only four coworkers, even revealing this number is problematic. Perhaps instead, we only tell Cathy that at least three people participated in her review. Constraints of this form are not typical in "traditional data" but seem essential for provenance. It may seem that revelation of some attributes is obviously acceptable, but when those attributes interact with relationships, the situation becomes confusing. For example, it seems that we should be able to tell Cathy when her review was generated, that it is her review, and that her manager produced it. However such revelation of dates could reveal information about which of her coworkers could have been involved in the review process. These sorts of problems may involve composing multiple attributes and therefore this is reminiscent of the work on composition. As in those cases a piece of confidential information may be inferred it from other data. Clearly we must account for interactions among the models.

4.2 Identity

Each of the models have a different notion of identity. From a data model perspective, there are multiple different notions of the identity of a file including its contents and its identifier. To further complicate matters, the identifier could be either the absolute path or the volume and inode. Commonly used models, such as the UNIX filesystem model, have multiple notions of identity. Hard links allow different absolute paths to refer to objects with same identifier. If two objects have different contents they are clearly different. However, if they have the same contents but different identifiers are they the same object?

In provenance identity is similarly ambiguous. Objects could be considered the same if their provenance starts with the same inputs and follows the same set of steps. Alternatively, the identity of an object could be its specific instance in the provenance graph. Recreation would lead to an object with a new identity.

Since each model expresses identity differently we have a problem. If we have access to the file we would want to be able to access the provenance entity that describes the file. It would also be useful to go in the other direction, from a provenance entity to the file it describes. Note that since provenance is immutable this may necessitate a data store that supports versions. Problems arise when we want to allow access to the data but not the provenance or vice versa. If we want to control access to the data when the provenance is accessible we could make the data identifier simply an attribute of the node and then limit access to this attribute. In the other case, we may have access to the file but want to restrict access to its provenance.

5 Related Work

This work is an extension of work done on PASS (Provenance Aware Storage Systems) [9]. PASS is a modified Linux kernel that automatically and transparently captures provenance by intercepting system calls in real time. It tracks what files a process read and wrote and records this information together with the data in the same file system. We also draw on user studies we conducted to determine what security capabilities users expect [2].

There are several projects that capture and represent provenance data. The first provenance challenge brought together a collection of teams working on this topic [7]. A related website provides information on the approaches used by the participating teams [10]. Most teams do not yet consider security. One noteworthy exception is PASOA [5]. While most of their discussion focuses on authentication and non-repudiation, they do

highlight the need for access controls and mention that it would likely be useful to operate in terms of groups of records [11].

The authors have been unable to find any related work on securing directed acyclic graphs. Security models for XML such as XACML seem most closely related but they are of limited applicability since they are limited by XML's tree structure and therefore do not appear to apply to general graphs [8]. There is work on combining several existing security models [3] although it does not seem to apply.

Databases generally provide security on a per cell basis. While this is useful and can be used to implement security controls for relationships it does not instruct our study as to how those controls should be set or how they interact with attribute and data security.

Hippocratic databases use query rewriting to enforce policies [1]. This approach is clearly different than that used by traditional database systems. While this approach is appealing, it is still not clear how to represent policies on relationships nor how those relationships interact with the attributes and "traditional data". The question changes from how to represent relationships to how to represent query rewrite expressions on relationships. This may be progress but it is still not clear how to limit access to certain relationships.

Security for logs seems to be a related area. This area is in desperate need of further investigation. The National Institute of Standards and Technology (NIST) only recently published a guide to log management [6]. This guide is remarkable in that it took so long before such a document was produced, and because it only considers entire log files. There is little discussion about what log files reveal about their source. Such a discussion would presumably provide partial access to log files.

6 Conclusion

Provenance is a class of meta-data with security needs that differ from those of "traditional data". Since provenance captures history it is immutable. The graph that describes the provenance is directed and acyclic. There are applications where this information needs to be secured. We have argued that the security of the provenance is different from that of the data it describes. How should provenance be secured?

Provenance can be modeled as an annotated causality graph. Our discussion splits a security solution into subsystems for the data, attributes and causality graph. Each of these need access controls. Unlike the data and attributes, it is not obvious how to represent security permissions on the causality graph. Even with a security model for the causality graph, there are interactions among the three security subsystems. We have presented some examples of interactions, but it is not clear how the three subsystems should interact in general. Our challenge is to construct a security model for causal graphs and study how that model interacts with security model for the attributes and data.

References

[1] AGRAWAL, R., KIERNAN, J., SRIKANT, R., AND XU, Y. Hippocratic databases. In *28th Int'l Conf. on Very Large Databases (VLDB), Hong Kong* (2002).

[2] BRAUN, U., AND SHINNAR, A. A Security Model for Provenance. Technical Report TR-04-06, Harvard University, Jan. 2006.

[3] DING, G., CHEN, J., LAX, R. F., AND CHEN, P. P. Graph-theoretic method for merging security system specifications. *Inf. Sci. 177*, 10 (2007), 2152–2166.

[4] FERRAIOLO, D., AND KUHN, R. Role-based access controls. In *15th NIST-NCSC National Computer Security Conference* (1992), pp. 554–563.

[5] GROTH, P., JIANG, S., MILES, S., MUNROE, S., TAN, V., TSASAKOU, S., AND MOREAU, L. D3.1.1: An architecture for provenance systems. Tech. rep., University of Southampton, Feb. 2006.

[6] KENT, K., AND SOUPPAYA, M. Guide to computer security log management. *Recommendations of the National Institute of Standards and Technology* (April 2006).

[7] MOREAU, L., ET AL. The First Provenance Challenge. *Concurrency and Computation: Practice and Experience*. Published online. DOI 10.1002/cpe.1233, April 2008.

[8] MOSES, T. eXtensible Access Control Markup Language (XACML) version 1.0. Tech. rep., OASIS, February 2003.

[9] MUNISWAMY-REDDY, K.-K., HOLLAND, D. A., BRAUN, U., AND SELTZER, M. Provenance-aware storage systems. In *Proceedings of the 2006 USENIX Annual Technical Conference* (June 2006).

[10] Provenance Challenge participating teams. `http://twiki.ipaw.info/bin/view/Challenge/ParticipatingTeams`.

[11] TAN, V., GROTH, P., MILES, S., JIANG, S., MUNROE, S., TSASAKOU, S., AND MOREAU, L. Security issues in a soa-based provenance system. In *Third International Provenance and Annotation Workshop* (May 2006), Springer.