

A Security Model for Provenance

Uri Braun, Avi Shinnar

Harvard University

{uribraun,shinnar}@eecs.harvard.edu

Abstract

Most security models are designed to protect data. Some also deal with traditional metadata. Provenance metadata introduces additional complexity, as does the delicate interactions between provenance metadata and the data it describes.

We designed a security model for provenance metadata. Our requirements were derived from potential users. The security model consists of two non-interfering models. One protects the structure or work-flow — namely which ancestors and descendants are accessible to which users. A second model specifies which node attributes are accessible to which users. Our evaluation suggests that our security model meets the users' requirements.

1 Introduction

Ever since the Senate Watergate hearings, the questions “What did the President know and when did he know it?” have become synonymous with scandal. Senator Howard Baker’s famous words are more commonly known in their oft paraphrased form “What did they know and when did they know it?”¹

The question “What did they know” asks about the data. “When did they know it” relates to the provenance. Provenance is the history or lineage of an object. Who told the President? Where did they get their information? Provenance is the sequence of events that leads to the conclusion. It has multiple applications including: authentication, recreation, and accounting.

Scandals are not the only application of provenance (although they might guarantee its never-ending future). Scientist often run multi-staged experiments, tracing back their steps and veering in a new direction. Tracking the paths taken, parameters and other attributes should improve recreatability in science. The intelligence community is also keenly interested in the process used to reach a conclusion — e.g. there will be an attack on the New York Subway tomorrow. Should the veracity of a source come into question, it may be useful to trace back the provenance and reevaluate the conclusion.

Provenance is metadata, not data. Draft a document, submit it to a committee, discuss and vote; this describes the process of approving a paper — or going to war. While the two may be similar from a provenance standpoint, one would hope their differences would also be apparent. These differences are to a large part a matter of substance — or what data is being contemplated.

Dealing with data is in some sense well studied and understood. This paper investigates problems surrounding provenance.

Provenance requires its own security model. Sometimes the data needs to be protected more closely than the provenance. Credit card numbers are generally protected more closely than the process of applying for a credit card. In other cases, the provenance is more sensitive than the data. Even when intelligence documents are declassified, sources are usually closely protected. Anonymous sources in newspaper articles are another manifestation of protected provenance of public information.

For a security model to be useful, it must be usable. Users drive the effectiveness of a security model. If it is easy for them to use and fulfills their needs, then the model might work in practice. Theoretical results are not sufficient for they may fail for pragmatic reasons — such as being disabled due to the tedious bookkeeping they impose. We measure our model primarily by its acceptance by potential futures.

Our project begins by defining the requirements for the security model. This is done through user studies. Using those requirements, we construct a security model — our result. We then return to the users in order to evaluate our security model. The three phases are requirements, security model, and evaluation. Both requirements and evaluation consist of user studies followed by some analysis on our part. The middle security model phase uses synthesis and further analysis to construct our proposed security model. This allows users to specify the foundation upon which the security model is developed, and later judge the result.

We use user studies to drive the requirements for a provenance security model. In Section 2 we provide background on user studies and give an overview of how we use them. We then present the requirements for a security model and discuss the user studies which produced the requirements. Section 3 provides background on security models and describes the methodology we used to produce the security model we propose. Our model is presented in Section 4, followed by our evaluation of the model in Section 5. We suggest future work in Section 6 and finally conclude (Section 7).

2 User Studies

Lacking agreement on both the requirements and evaluation criteria for a provenance security model, we turn to users as the final decision makers. Having chosen to appoint our users to the jury, we must still decide who our users are and how to identify their requirements. To answer these questions, we consult existing work on how to gather information from users. We

¹<http://www.watergate.com/stories/watergate.asp>

then describe our plans for using these methods in the context of provenance security models.

This section begins with relevant background information on user studies in Section 2.1. Our usage of the technique follows in Section 2.2. Section 2.3 presents results we drew from these discussions. Our requirements are then described in Section 2.4.

2.1 Related Work

There are many methods for gathering data from users, such as surveys, questionnaires and user studies. User studies have been demonstrated to be an effective tool for soliciting user input in the early stages of the product life-cycle. They can also be useful in the evaluation of completed, or nearly completed designs as well as for working systems. As such, we use user studies to gather information from our users. Since there are multiple types of user studies, we begin with a brief description of the relevant ones.

High fidelity user studies are so called because they reproduce the interface being studied with a high degree of precision. Creating such a study requires a significant effort to construct a realistic, detailed mock up of the application to be studied. They do not contain or display real data, nor do they implement the full logic. However, the interfaces are presented in crisp and near final form. Data may be fixed ahead of time, but it appears as the user would expect. Overall, this approach yields a remarkably crisp model which may appear nearly complete.

Low fidelity user studies consist of studying users' interactions with a rudimentary prototype of the system under design. In contrast, high fidelity – or traditional – user studies rely on prototypes which are more sophisticated and in many ways resemble a complete product. Low fidelity user studies are so named due to their crude replication of a final crisp product. It has been demonstrated that low fidelity prototypes can lead to results which are significantly similar to the results of their more in-depth brethren — namely, high fidelity studies [10]. Further research has demonstrated that low fidelity user studies consisting of paper representations of various situations were sufficient to lead to many of the same results obtained through computer based low fidelity studies [8].

It is precisely their crude nature which makes low fidelity user studies especially applicable to early stages in the design cycle. High fidelity studies which provide a crisp interface give the user the feeling the system is mostly completed and perhaps even generally working. This can lead to comments about colors, fonts and other visual presentation effects. In general comments would tend to focus more on the details as the system is assumed to have already been designed. In contrast, low fidelity studies are crude which make it clear to the user that colors and other display features are not the focus of discussion. Discussion focuses on the key ideas and concepts with an understanding that interface issues will be considered later. Thus early in the design cycle low fidelity provides advantages not present in more elaborate mock ups.

2.2 Methodology

We use low fidelity user studies to define the requirements for our security model. These user studies were justified by the lack

of an existing set of requirements for provenance security models and the applicability of low fidelity user studies – as documented in the literature. We realized that many tasks involve provenance and thus users have security concerns they would like to have addressed. Our objective was to collect these concerns and use them to define the requirements for our model.

It might have been possible to analyze written accounts or regulations which describe security policies for provenance. While this may seem to be a good starting point, there are numerous reasons to take a different approach. Some fields such as some scientific disciplines do not have clearly documented security, privacy or other rules. Others have an overwhelming set of regulations – think medicine. In both cases, it would be difficult to understand from the written work how the rules impact users and what users would require. Since we wanted to touch on a few different fields, it would have been necessary to gather information from users anyhow. Since we needed users input, both to understand the rules and also to explain how the rules impact the users themselves, user studies seemed to be the clear approach.

Our low fidelity user studies made use of: cork boards, cut up colored index cards, rubber bands, pins, colored markers, scissors (see Figure 1). In addition we also brought a box of index cards to take notes. We chose these materials because they are easy to manipulate and modify quickly. It is important to realize that the choice of materials for a user study can affect the outcome. Our materials were intended to make it easy to capture directed acyclic graphs (DAGs). The notes and markers would then allow us to capture other information which we could not capture on the cork boards.

The use of cork boards with index cards, pins and rubber bands may limit expected results to graph structures. It is already known that provenance relations form a DAG. DAGs are used to represent data in Grid environments [11] and also more generally [9]. This representation is a result of the nature of ancestry data. Each transformation connects ancestors to descendants. Direction is relevant; ancestors and descendants must be differentiable. These characteristics suggest a DAG. Given these realizations, we consider the choice of materials appropriate.

In order to understand the needs of potential users across a variety of domains, we conduct sessions with users from: human resources and administration, medicine, and scientific research. These communities were considered due to: their accessibility, prior work showing utility of provenance in their work and their different uses of data. Medicine and administrative positions each deal with and guard data. The scientific community is more of a wild card since different disciplines may treat their provenance with vastly differing degrees sensitivity. It would have been nice to cover a larger number and variety of potential users, but this was not possible due to both time and access. It would be nice to know if our selection of users is more generally representative.

For each community we considered, we first constructed a sample use case. Figure 1 shows the examples we created. The human resources and administration sample describes the process used to write an employee review. Our scientific case depicted a genomic sequencing experiment. In our medical example, patient data is collected through an examination by a doctor

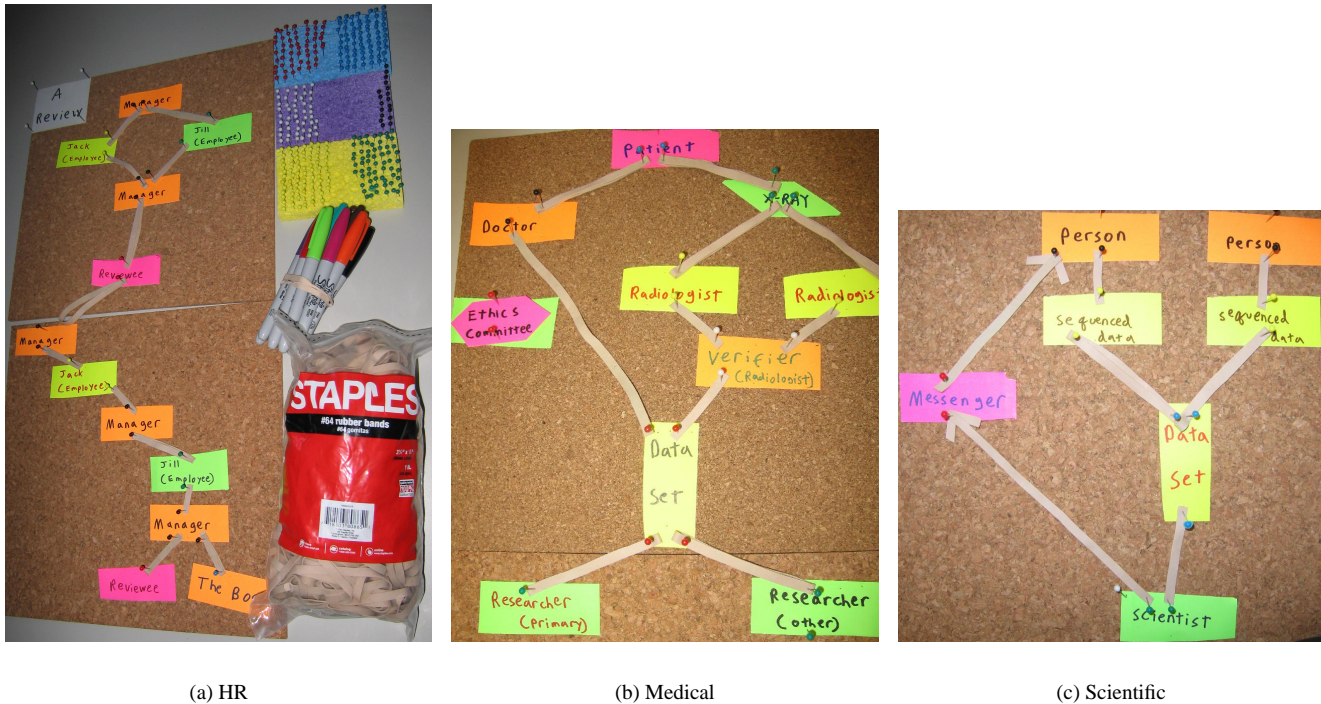


Figure 1: Initial domain specific use cases

as well as by reading the patients x-ray. Researchers analyze the collected patient data. These sample use cases are not meant to be accurate. Their role is to explain what information we are looking for.

With the sample use cases in hand, it is time to schedule time with subjects. To arrange user studies, we would contact potential subjects by email. We open by introducing ourselves and provide a brief description of our project. After making it clear that we value their input and their time, we ask for to meet with them for two sessions each for half an hour. Our schedule and a list of available times appears next. We then make it clear that they need no preparation

For each subject, we chose which of us would be a greeter/note taker and which would be the facilitator. The greeter’s job is to introduce us, describe the project, thank the subject and make it clear there is no way to fail. At that point the facilitator takes over.

The facilitator begins by describing the sample use case – encouraging the subject to ask questions if anything is unclear. As an example consider our sample case for administration and human resources – writing an employee review (Figure 1(a)). At the top of the figure, the manager asks Jack and Jill to comment on their coworker – the reviewee. After receiving Jack and Jill’s feedback, the manager writes a review and presents it to the reviewee. On the bottom cork board, we see that the reviewee responds with comments to the manager – arguing that it was Jack who did not contribute. The manager follows up with Jack, but Jack’s response does not clarify the situation, causing the manager to confer with Jill. Jill makes a convincing argument for the reviewee’s claims. The manager updates the review and

provides the updated version to “The Boss” and the reviewee.

Once the subject understands the use case, the facilitator asks the subject for a use case from their work. Most subjects are new they have no interesting use case to share. Fortunately, they tend to be wrong. Sometimes the subject might need some coaxing. We might ask, describe something you do. Another approach that worked was, what did you just finish doing before meeting with us? These would provide an entry point into some task the subject interacts with. Once the subject begins to explain, we find it critical to begin placing things on the cork board. As soon as the subject sees the cork-board filling up, they tend to relax and realize they have information we are interested in.

The facilitator and subject then begin to construct the model on the fly. Meanwhile the note-taker busily takes notes, making sure to capture anything not clearly presented on the cork-board. As the discussion progresses, the facilitator pesters the subject with questions. Does the manager always wait for Jack and Jill to respond? Does some of the data sometimes come from somewhere else? What complications sometimes arise? Does the process always begin the same way? Are there cases with different outcomes? Do other players sometimes come into the picture? Generally, these and similar question lead us to a deluge of information and we are busy trying to record the information as it keeps coming at us.

In most cases, we collect multiple use cases from one user session. The first case is often a learning experience, but it often highlights other cases to diagram. On other occasions, the subject realized the use case they had suggested was not all that interesting and really wanted to present another case which involved more participants or stickier issues. In general, subjects

often experienced an aha moment after which the information flowed quickly.

Throughout the session, we ask questions about security – however we would never use the *s* word. Questions might take the form, does the manager tell you what Jill said? During the middle of the session we try to keep the security questions from the subject’s perspective. Although, the subject often offers security data without asking – such as: the manager will not tell me, but may tell “the boss”. As the session progresses, we would focus on these security and privacy issues and start to consider the use case from the perspective of its other participants. Does Jill know what Jack told the manager? Does the manager tell Jack, that Jill has already been consulted? Generally, security issues are presented as either privacy issues or better yet as questions about who tells or has access to who’s information.

Finally, we both thank the subject. We tell them we might contact them with questions if something is unclear. Although our As the session wraps up, we focus on security and privacy issues. We avoid discussing security, but we ask questions about privacy issues – especially when the interviewee raised the issue.

2.3 Results

Figure 2(a) provides examples of use cases we constructed during sessions with users. Clearly the diagrams drawn in real time with subjects are not as aesthetically pleasing, yet they represent results from real people. Before considering each domain independently, we discuss some issues which generalize across the domains.

There was some deal of ambiguity in the data collected from our user studies. For example, when considering the flow between parties it became clear that edges represented several things at once. In Figure 2(a) the “Monthly Report” represents the document, the data, and the process of transferring data from Accounting to Edward. Additionally, it was not clear whether some interactions should be considered relevant from a provenance standpoint. We were unclear whether reminding somebody to send in a form or redirecting an inquiry was a relevant provenance operation for the use case.

In some cases, the subject specifically told us a process could begin in one of a few ways. A patient might come on their own, be referred by another doctor or be referred by a social worker. Such cases allow us to include multiple use cases in one diagram. Moreover, these intertwined cases make us question whether the varying cases are distinguished. Does the doctor know who the referrer was? Does it matter?

2.3.1 Science

Results from our interviews with scientists suggests that some users have no need for provenance security. One subject made this very clear by telling us repeatedly that their scientific research involved no interaction with other people until they had data to present. Interesting data would be published, everything else went to the dumpster. We cannot claim to envy that arrangement. Other scientists may need security for their provenance, but our subjects clearly did not.

2.3.2 Medical

Our interview with a doctor, highlighted several interesting issues. In the medical field, patients have near total control of their medical data. Under normal conditions, information cannot be shared among medical teams. Specifically, the doctor and psychiatrist could not share information without a signed consent form from the patient. Such clear rules, made the need for security considerations clear. Yet, other situations show that security within a team is often on another level. The nurses and doctors within a team tend to have equal access to the patients data. In fact, the doctor often did not even care if they received the patient’s record from the another doctor directly or via a nurse. However, in the latter cases the rules were not necessarily as black and white. Clearly the medical profession has given provenance information some thought, but our research suggests the rules vary significantly depending on the situation.

2.3.3 Administrative & HR

Administrative results were split. Erin presented a complex example which will consider in depth later because it highlights a good number of security issues. Edward is involved in multiple processes, but structurally the processes – at least those we were able to identify – are both simple and identical.

In all the use cases Edward presented, he uses reports with summarized data – but has access to detailed data. His interactions are always with managers and the dean. Managers have more detailed information about the data under their purview. The dean is made aware of the big picture. Both Edward and the dean have full access to the data their subordinates manage, but they rarely need to dig into it. From a security standpoint, their is a simple hierarchy of security.

Figure 3 describes the process for a new hire to get paid. This example highlighted several security issues. Examples of delegation show how in some cases the delegate has partial access but in other cases the delegate is almost indistinguishable from the party the delegate represents. We also see cases where different parties have access to different granularity of provenance information.

Erin described several cases where delegation occurred. When collecting information on funding source and job description, Erin often contacted the professor and the professor’s assistant. However, the answer always had to come from the professor or be able to be tracked back to the professor. In other cases the source of the provenance was less critical. Erin could begin the hiring process due to an inquiry from accounting or from the student.

Hiring of foreign nationals provides an example where an entire sub-process becomes relevant under a specific condition. In this case, the applicability of the sub-process may have its own security concerns. Hiring foreign nationals requires additional forms and the involvement of an additional organization – the foreign national information forum. While the professor, accounting and HR need to know about this additional process, others need not have access to this information. This provides a case where some people might only need to see the outcome of the process and not have access to the internal workings.

The role of payroll demonstrated another interesting charac-

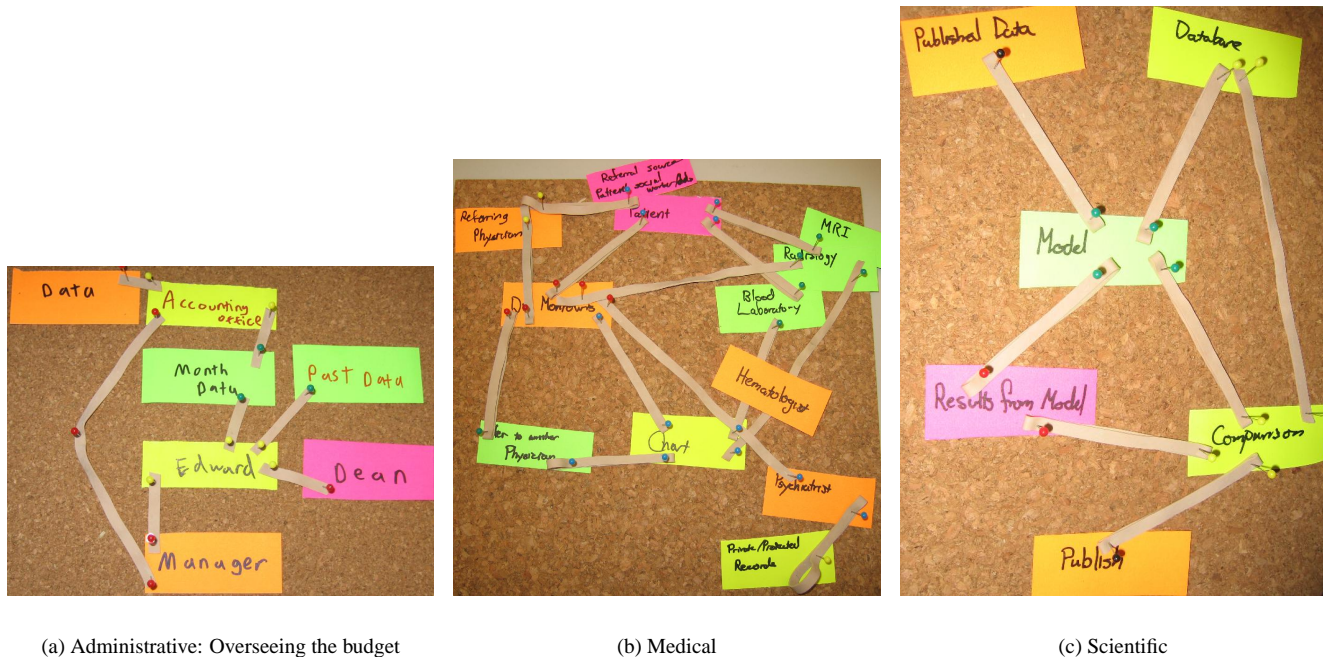


Figure 2: Use cases constructed in consultation with potential users

teristic. In some cases payroll is seen as a single entity, whereas in other cases there is a distinction between the payroll for FAS (Faculty of Arts and Sciences) and the department's payroll group. This distinction was relevant to some people but not to others. In this case, the difference applies to the level of detail as to what the payroll department is, which is somewhat different then the details of the procedure followed described immediately above.

2.4 Requirements

Our requirements stem from further analysis of the results described above. This list may not be complete, but it represents the requirements we were able to specify explicitly and state clearly. When we had questions we often had to return to the source data to see if it contained any example which might tell us if a specific feature was relevant to our users. Here we try to summarize the requirements we were able to identify.

Security needs depend upon the people and communities involved. Scientists, medical professionals and administrators have different needs. Security depends on who is viewing the data. Different people within use cases clearly had different permission to the data. Our medical example also shows that the security within a team is often different from the security between teams.

Sometimes the only relevant provenance is that there is provenance. Existence is often the issue. Erin cared that she received a referral – from a student or accounting – but often did not care which. Doctors sometimes do not care whether the patient is referred to them directly or via a referring agent – another doctor or perhaps a social worker.

In other cases, existence takes on a slightly different view as

the question is how many exist not just whether one exists. For example, if multiple doctors provide input on a patient's status, that may change the weight a scientist assigns to the results the doctors agreed on. Here each doctor's participation – existence in the process – is relevant even if the identity of the doctor is irrelevant. Often the questions: what was the original source, and how many sources were there – ask the critical information.

At other times the question is whether the source is the same or different. Researchers may only care that three cases were studied by the same doctor. They may not need to know the identity of the doctor. If that doctor tends to use a different rating system, they could then adjust their data appropriately. Medical use cases clearly demonstrated this issue of differentiating same different, as well as anonymizing the source. Thus differentiating actors is often sufficient without providing any more detail. This also means that the differentiation of actors is itself information that needs security permission.

Delegation is a required feature. Professors need the ability to delegate some access to their assistants, yet they maintain decision power over other information. Patients have to explicitly delegate access to their medical records. Charts are protected by psychiatrists and doctors by storing the information separately and granting access only when permission is delegated by the patient. Patients can rescind delegation privileges, disallowing access to future requests to access medical records.

Our discussion of delegation also suggested another issue, that of granularity of data. Erin sometimes had to be able to tell if the data originated at the professor or the professor's assistant. While this is a matter of delegation, it also gets at what the meaning of professor is in different contexts. In some cases, professor and the professor's assistant are part of one entity, yet

in other cases the differentiation is critical.

Through analysis of user study results, we are able to identify certain specific provenance security requirements. Different people and communities have different permissions. Existence information, number of inputs/outputs, and differentiation of sources require security mechanisms. Delegation is a needed feature and introduces several issues. Data may exist with multiple levels of detail. Some parties may have access to greater detail than other. These requirements are simply summaries – after analysis – of the results described earlier. They provide the starting point we need to drive the design of a security model.

3 Security Models

Here we present existing security models (Section 3.1), lay out goals (Section 3.2) for our security model and present our methodology (Section 3.3) for constructing it.

3.1 Related Work

Much research has been done on security models for data without provenance information. Related work in this area is intended to present ways of thinking about security and also to offer models that may be able to be extended to include provenance.

One of the most common models for data security is the access control matrix. The basic idea is from Harrison [4]. Rows in the matrix represent subjects, and columns represent objects. A matrix entry represents the access rights a subject has for an object. When discussing normal filesystem data, a common set of permissions such as read/write/execute is often assumed. The entries are then a subset of these possible permissions. In many systems, an access control matrices is encoded as access control lists (ACLs). Each object has an associated list that maps from users to their associated permissions on that object. This allows for a more efficient representation of a sparse matrix. Many such schemes, when dealing with hierarchical data, also allow for inheritance of permissions, further minimizing storage needs as well as simplifying policy, at the expense of simplicity. A good example is Windows NT[®], which uses such a scheme for filesystem data.

Another common way to model security system is with a capability based system. User's are given token's that enable them to perform actions. The token may grant the ability to perform a whole class of actions on a large group of people, or it may allow one time restricted access to a single resource. Possession of a token implies permission to perform the associated action. The user/program must present an appropriate token to the system when it attempts a restricted action. A good overview of capability based systems is given by Levy [5].

In many systems, dealing with specific users is too limiting. Rather than specify that a given person can access a resource, one would like to specify that any secretary can access the resource. Role-based systems address this scenario. Rather than assign permissions to users, role-based systems assign permissions to roles. Users may then assume roles to perform various tasks (assuming that they have permission to assume that role). Saunders discusses the relationships that role-based systems have with the access control matrix model [7]. Barkley

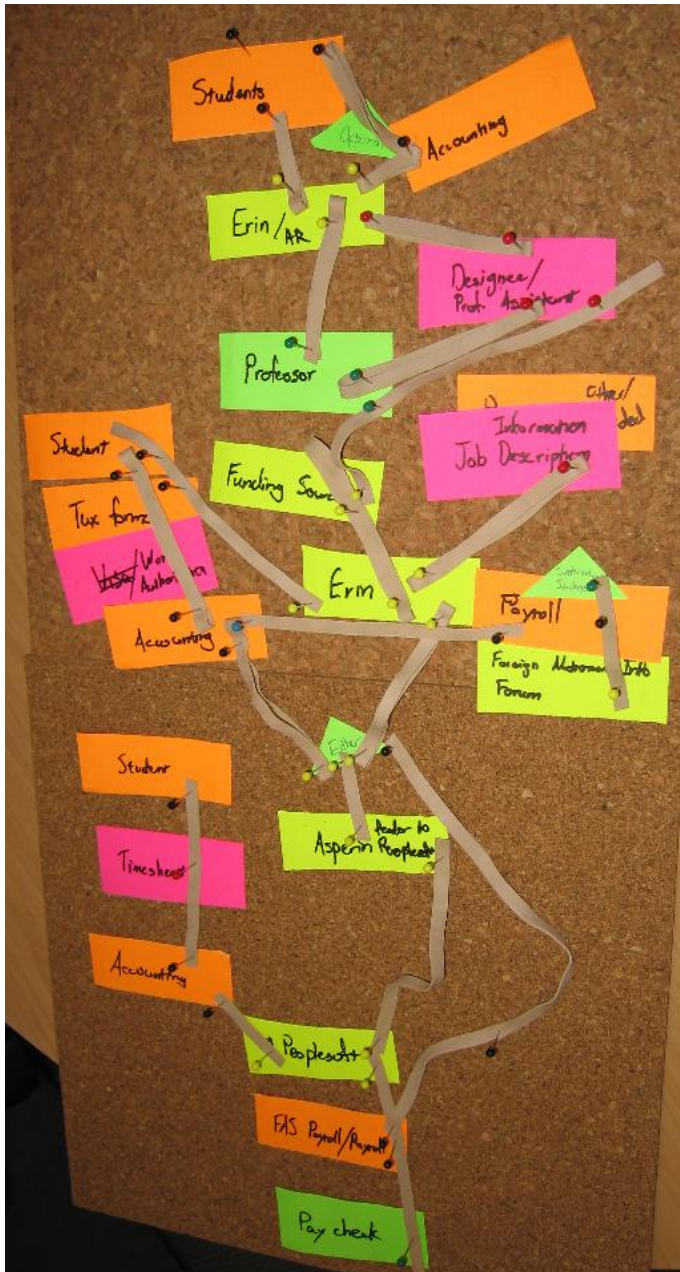


Figure 3: HR: Process for a new higher to get paid.

compares role-based access control systems to access control list based systems [1]. The role-based system essentially adds another level of indirection. This allows many other systems to be simulated. For example, Crampton discusses how a role-based access model can be integrated with permission inheritance, resulting in a type of multi-level security [2].

All of the models and systems discussed so far describe how permissions are specified and enforced. It is also interesting to discuss how the permissions are assigned. In a system based on discretionary access control, permissions are assigned by the users (generally with the help of some default). In a system based on mandatory access control, certain security policies (generally rule-bases) are enforced by the system. A typical example might include a rule that disallows anything to be read by someone with a lower security clearance than its creator. Mandatory access control is often used in military systems. It is also possible to have a system where the types of access control co-exist. Osborn and others [6] discuss the relationships between mandatory access control and other systems, such as role based systems. A commonly approximated formal model that supports mandatory access control is the Bell and LaPadula model.

Another security model, often used in large security conscious systems, is the ring model. Similar to a simple mandatory access control system, a system with based on a ring model enforces a strictly hierarchical set of permissions on multiple “rings” in a system. Components of the system can only access components of either the same ring or a lower level ring. Generally, the higher the ring level, the less permissions components of the ring has. Dijkstra is a strong advocate of these types of systems, pointing out its advantages for software engineering and reducing both complexity and bugs, in addition to its security advantages[3]. Windows NT[®] provides a modern example of such a system, where there are multiple kernel rings structured in such a fashion.

3.2 Goals

The goal of our security model is to support the operations described in the case studies. Secondly, the security model will be assessed based on its success at achieving our other desired properties. These secondary traits are:

- Self-consistency
- Security
- Flexibility
- Usability

Our security model is expected to be a consistent and secure system that is flexible enough to meet the needs of our users. If possible, we will attempt to extend existing well-used security models in a manner that is both flexible and intuitive to the user.

The degree to which we achieve these objectives will be assessed using user input. We believe the above traits are important because they are believed to facilitate a user’s understanding of the systems operation. This belief will be reinforced or challenged based on the information we gather from the user community.

3.3 Methodology

Constructing a security model consists of several analysis steps driven by the results of the user studies. We begin by resolving ambiguity in the initial results and then view the use case from the perspective of each participant in each model. This allows us to find common constructs and required capabilities. Our goal is then to develop mechanisms that can express these scenarios in the simplest possible manner. We used these mechanisms to develop a user level perspective of the system. Policy and defaults conclude the discussion and demonstrate usage of the aforementioned mechanisms.

Removing ambiguity in the use cases involves some relatively straightforward tediousness. Sometimes translating what the user said into a more formal representation requires considering the notes we took and combining them with the flow-diagrams. Disambiguation also requires realizing that some constructs represent several things at once. For example, in one use case a monthly data report is a document, a collection of data, and a process for transferring data between two parties. Results of this analysis identified the data, agents that interacted with the data and the sequence of operations which occurred.

We then created a view of each use case from the perspective of each of the agents that participated in it. Each participant has her or her own view of the rest of the system. This provides insight into what data that person could or could not access. Doing this for each use case and each participant provides multiple cases from which to consider the various situations the security model must address. Once we identified common constructs we could begin to collect a set of mechanisms that would be needed to satisfy the user’s needs.

One observation we make is that the data flow appears different from each participant’s viewpoint. This is used as part of the analysis presented above. It also leads to our decision to represent each of these views of the graph. Queries are then processed with respect to a particular user’s view of the world.

Another observation is that the structure of the flow diagram representing the provenance is fundamentally different from the attributes on nodes. As a result we partitioned the problem into creating two security models. One security model specifies what edges can be traversed thereby specifying which ancestors and descendants can be accessed. A second security model specifies which attributes of a node can be accessed by each party.

We were able to ascertain the need for several security features based on these models. Edges — reflecting dataflow — occur with several security levels:

- No knowledge of data source
- Data known to be from “higher up” but not where
- Data known to flow “lower” but not where
- Exact source/destination (as appropriate) known completely

Additionally, each attribute of a node is protected to differing extents for different users.

4 Results

The key innovation of our security model is that it is actually two non-interfering, easy to understand security models. We differentiate the structure of provenance metadata and deal with it separately. Viewing provenance as forming a graph based on the data, we have one separate security model for the edges (the existence of a relationship between nodes, or data) and another security model for provenance metadata stored in the actual nodes of this graph. This greatly reduces the complexity of the security system.

We first provide a formal description of the underlying security system. We then explain how these are presented to higher levels of the system. We model provenance security with a multi-levelled system. This allows us to have a nice simple level to think about formally while presenting something nicer to other components of the system. We then briefly provide some examples of policies that may be enforced above our system.

4.1 Edge Model

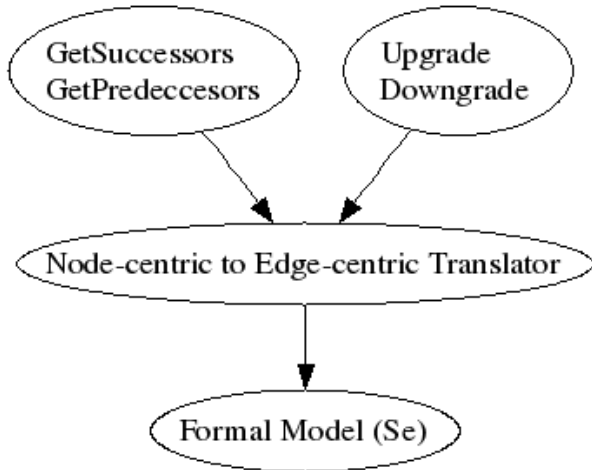


Figure 4: "The levels in the security model for edges"

We describe the model in terms of relations on the edges in a directed graph. We assume that P represents the set of possible personas that are relevant. Note that while we assume in discussion that these are people, there is nothing to restrict them from being roles or some other abstraction.

The levels in the edge security model are represented in Fig 4. On the bottom is the formal model, on the top is the interface other components in the system use.

4.1.1 Formal Model

We want to define a security model for provenance relationships. This means we want to allow the system to restrict the information available to a given person about an edge. Note, that as discussed later, at the user level, edges are not first class objects, only nodes are. However, for the underlying security model it is convenient to treat them as such, allowing us to clearly indicate how this model is independent of the node security model.

Head node	Tail node	Edge
Nil	Nil	Nil
Nil	Read Traverse	Tail
Read	Nil	Head
Read	Read Traverse	HeadTail
Traverse	Nil	Head
Traverse	Read	HeadTail
Traverse	Traverse	Traverse

Table 1: Mapping node-centric edge security to edge-centric edge security

For a given edge, different levels of knowledge are possible.

- Nil: The user may not be allowed to know that the edge exists
- Head: The user is allowed to know that the node at the head of the edge in fact has an edge coming from it, but is unaware of its tail. (So only this edge's head is knowable by the user).
- Tail: The user is allowed to know that the node at the tail of the edge in fact has an edge coming into it, but is unaware of its head. (So only this edge's tail is knowable by the user).
- Head || Tail: The user is allowed to know that the nodes at both ends have edges coming out of them (but don't know where either edge leads – so they don't know that a single edge joins them).
- Traverse: The user is allowed to know both nodes that are connected by this edge.

We would also like to allow users to delegate permission to other users. We represent this by explicitly specifying what permissions a user has as well as what permissions the user may delegate. Note that the delegation permissions may not be more permissive than the user's actual permissions.

If we let P be the set of people interacting with the system and let E be the set of edges in the system, then we can write the security relation as

$$S_E : E \times P \rightarrow T \times T$$

$$T : \text{Nil} \parallel \text{Head} \parallel \text{Tail} \parallel \text{HeadTail} \parallel \text{Traverse}$$

4.1.2 Translation

The formal model deals with edges. However, the fundamental abstraction for other components in the system is that of the node. Edges are represented implicitly.

Since we are now looking at everything from a node's perspective, we use three states (Nil, Read, Traverse) on each node's part of the edge. The translation layer translates between the two models, as described in Table 1.

4.1.3 Querying

The provenance security subsystem is called when another system component (or the user) tries to discover provenance information. We assume that the user obtains a starting node for a

Method	Affected Sets	Change	Check
Upgrade	see	$S_o = \max\{L, S_o\}$	$L \leq D_m$
Upgrade	see and delegable	$D_o = \max\{L, D_o\}$ $S_o = \max\{L, S_o\}$	$L \leq D_m$
Downgrade	see and delegable	$S_m = \min\{S_m, L\}$ $D_m = \min\{D_m, L\}$	
Downgrade	delegable	$D_m = \min\{D_m, L\}$	

Table 2: Modifying Security: L is the new permissions level, D is the delegable permission level for a node’s edge, S is the actual permission level for a node’s edge. The subscripts m and o mean my permission level and the other person’s permission level.

query from elsewhere. The provenance component exposes two methods, `GetSuccessors` and `GetPredecessors` which, when given a node, returns the appropriate set of nodes. Before being returned, this list is filtered by consulting with the provenance security subsystem. Each node and edge combination is passed through the translator to the underlying model, and is checked to see how much of it can be retained (the complete edge, knowledge of existence of an edge, or nothing).

4.1.4 Modifying Permissions

To change security permissions on edges, two methods are exposed, `Upgrade` and `Downgrade`. `Upgrade` allows a person to try and upgrade someone else’s permissions. `Downgrade` allows a person to downgrade their own permissions. They can upgrade or downgrade the set of permissions and the set of delegable permissions. The exact checks and actions are described in Table 2.

4.2 Node Model

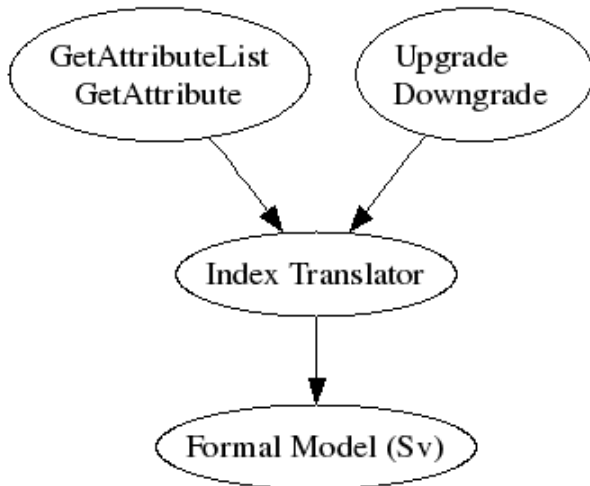


Figure 5: "The levels in the security model for nodes"

Provenance is more than just the relationship between data, it is also a description of that relationship. Note that this descrip-

tion is immutable: it is created along with the node and can’t be later modified. This data can be modeled by assigning to each node a set of attributes A which maps attributes (strings) to values. The security system allows the user to gain access to a specified subset of these attributes. There is one other interesting twist. It is useful to allow a user to only see a subset of an attribute. For example, we may want an attribute that indicates where the data came from. This may involve a person and their organization. We may only want some users to know the organization and other to know just the individual. A more general case may involve a file path attribute, where we want some users to only see a subset of the path. We model this situation by having each attribute actually map to a list of values. Each one has separate permissions (and delegation permissions).

As in the edge model, we present a multi-level model for nodes. The levels in the node security model are represented in Fig 5. On the bottom is the formal model, on the top is the interface other components in the system use.

4.2.1 Formal Model

If we let P be the set of people interacting with the system, let N be the set of nodes in the system, and let A be the set of possible attributes, then we can write the security relation as

$$S_v : N \times P \times A \rightarrow L \times L$$

$$L : \text{int list}$$

note that L represents a list of indices that the user has permission to read (or delegate), and as before, the first list is the list of readable indices, and the second list is that of delegable indices. The second list must be a subset of the first list. We also clearly require that the indices must not be out of bounds for the attribute list. A null L of readable indices indicates that the user can’t read (or even have confirmed the existence of) the attribute. The simple case is represented by a list with one item.

4.2.2 Translation

The formal model uses absolute indices into attribute lists. However, other components in the system use person-relative indices. For example, if a person can only view items 3, 5 in an attribute list, these are termed 0, 1 by other components. The translation layer translates between these systems.

4.2.3 Querying

Other system components may wish to query a given node to find out its provenance information. As with the edge security model, this query is filtered by the security subsystem. The provenance system provides two methods for querying, `GetAttributeList` and `GetAttribute`.

`GetAttributeList` returns a list of attributes available for a node. This list is filtered by the security subsystem. If the person does not have permission to see any elements in the attribute’s value list, the the attribute is removed from the returned list. If there is a desire to allow a person to see that a list exists without showing the user any contents this can be implemented by a higher level convention, such as having the first

Method	Affected Sets	Change	Check
Upgrade	see	$S_o = L \cup S_o$	$L \subseteq D_m$
Upgrade	see and delegable	$D_o = L \cup D_o$ $S_o = L \cup S_o$	$L \subseteq D_m$
Downgrade	see and delegable	$S_m = S_m - L$ $D_m = D_m - L$	
Downgrade	delegable	$D_m = D_m - L$	

Table 3: Modifying Security: L is the new set of indices, D is the set of delegable indices for an attribute, S is the set of visible indices for an attribute. The subscripts m and o mean my set and the other person’s set.

element in the list be a dummy value which is ignored. Allowing a person to see only this value would effectively inform them that the attribute exists while revealing no information as to its value.

`GetAttribute` returns the list of values for an attribute. This list is filtered by the security subsystem; only visible elements of the list are returned. As discussed above by `GetAttributeList`, not being able to see any elements is indistinguishable from the attribute not existing.

4.2.4 Modifying Permissions

Changing security permission on nodes has a similar interface to the analogous operations on edges. Two methods are exposed, `Upgrade` and `Downgrade`. `Upgrade` allows a person to try and upgrade someone else’s permissions. `Downgrade` allows a person to downgrade their own permissions. They can upgrade or downgrade the level of permission and the level of delegable permission. The exact checks and actions are described in Table 3. Note that the set of indices are of course person-relative, and are translated by the translation level into absolute indices. As a result invalid indices are simply out-of-range indices and are silently ignored. Higher level components are encouraged to check for this condition in advance, for improved error reporting.

5 Evaluation

Our evaluation is based on user input as well as formal analysis. Continuing our repeated theme, we consider the user the final determinant of the success of our model. That said, we try to structure the model so that it can be analyzed formally. Formal analysis is intended to argue that the model works, whereas user input argue for its applicability to real world situations. We present our evaluation based on user input in Section 5.1 and the formal analysis in Section 5.2.

5.1 User Studies

Since users decide whether the security model has any chance of being used, we begin our evaluation by considering their input on the security model. Lack of an implementation and the variety of samples studied limited the user base we could use during evaluation. As a result we used an iterative approach to evaluate our user study. This iterative approach consisted of a seconding interview with the users we had talked to earlier –

when developing our requirements. Based on their answers to various questions, we tried to determine whether our model can implement the users’ security needs.

Initially, we had wanted to evaluate our model by walking the users through use cases and presenting users with the systems response. This is not possible due to the limited size of our user base. We need users to determine the security *policies* the system would enforce. Policies might specify who can or cannot access what data. We would then need to translate those policy statements in our model. This part is possible, However, evaluating whether our model behaves correctly, would require walking through the use case – plus security – with another user. We did not have two users who were familiar with the exact same use case and would therefore know what the system should and should not provide the user access to. To be able to use this approach for evaluation, we would also need to repeat this run through with multiple users. While it would be nice to do this, we would need a much larger user base and a much more time.

Since we could not use our preferred approach, we used an iterative approach. Each subject provided the ability to iterate through the model and evaluate it. We used the first part of the evaluation scheme described above. The goal was to see if the policy statements users specified could be implemented in our model. Each time users specify a constraint, we check to see if we can implement it in our security model. If it can be we continue to the next constraint. Otherwise we modify the model to consider the case. The evaluation process consisted of multiple iterations of this process.

Most of the evaluation process has been presented as part of the user study section presented earlier. This was necessary because our model is the result of multiple iterations we described earlier. Had we tried to present the iterative process earlier, it would have been nearly impossible to describe our security model. The questions we asked to gather the information for each iterations were also covered earlier. We now present some of the particularly notable changes to our model and the responses which motivated them.

Granularity is a concept which came up late in the development of our model. Erin referred to payroll as both the department’s payroll and the larger payroll organization, highlights the need for a granular view — or lists — for specific attributes. This is repeated in Erin’s use of professor to mean the decision making body — possibly including the professor’s assistant — and the professor the person — excluding the assistant. Some decisions and notifications must originate from the professor whereas others could originate with the professor’s assistant.

Similarly, delegation was not originally part of our security model. This change required the creation of separating what a user can see from what they can delegate. These changes were motivated by the case where patients delegate access to their medical records. It also appears in Erin’s sending notification to the professor’s assistant on the professor’s behalf.

It is difficult to assess whether the evaluation is complete. However, over the past several iterations we have not had any need to modify our security model. This does not guarantee that the model needs no changes. In fact, we would argue that security models are dynamic changing entities not static constructs. Our models adaptability to the various issues described above,

suggest that we have used an approach that effectively represents the essence of the problem at hand.

5.2 Formal Analysis

Work on formal analysis of the security model has not yet lead to the proofs we would have liked. We simply have run out of time. However, we believe it should be possible to prove the model is consistent. as well as fully specified. Throughout development of the security model, we attempted to make the formal representation as simple as possible. We have also attempted to analyze the model more formally, but lack the time for a truly formal presentation of our analysis. Some of our constraints where designed to facilitate formal analysis. This analysis also lead to some subtle changes in semantics from what we had originally proposed. While having no visible effect on the usefulness on the system, they made analysis easier and removed minor inconsistencies and ambiguities.

6 Future Work

There are several directions for future work. One issue is assessing the applicability of this model to other areas. Another issue is determining reasonable default permissions for provenance structure, as well as defaults for permissions for provenance metadata. Policies for delegation face similar challenges. Finally, implementation presents several challenges.

Based on our user studies we would make the following suggestions for defaults — however their applicability more generally remains an open problem. Specifically it is not clear whether it is possible to significantly improve on the following very simplistic default scheme. Our user studies suggest that permissions on edges be set so that person P can traverse all edges neighboring nodes which represent data created by P . This would seem to make sense as edges neighboring nodes that hold P 's data represent items received or sent by P . P can probably know where the recieved the data from and where they sent it to. More intelligent defaults may be possible. Attributes default to allowing the creator full permissions, while initially granting everybody else none.

Specifying a delegation policy is even less clear. We are not clear how to collect delegation information from users. Based on the descriptions we have managed to accumulate, we would propose a policy where a user could by default delegate all data they have permission to. Restricting this further would require explicit action. This seems problematic, but sharing information occurs relatively frequently and we find it difficult to identify any rules which would be more selective.

An implementation would face several challenges. Notably performance in time and space may pose significant challenges. If each edge has a policy for even half of the vertices, the space could be seen to grow quite large. We would expect this space to be relatively sparse, so it would hopefully mean that only a small portion of the space would actually be realized. Additionally performing a query which crosses many edges and interrogates numerous nodes, would test the speed with which these security policies could be evaluated. Finally, implementation has a tendency to reveal flaws in the design or impossible practical issues which would otherwise be difficult to identify.

7 Conclusion

We have developed a security model for provenance. In developing our security model, we have designed a process to gather future users' requirements for a security model. Through our user studies, we have collected a set of requirements based on the input of users from a variety of communities. These requirements may or may not generalize to a wider user community but they provide a starting point for future work. We have also described an evaluation technique. Finally, we have identified directions for future work.

Our security model divides the task of protecting provenance information into protecting the flow of information — or structure — and protecting the attribute — or flat — provenance information. This division allows us to construct two simple security models. Each model operates independently. We argue that this division and its resultant simplification of the overall model makes the model simpler to reason about. Additionally, it simplifies the implementation and should also make the model easier for users to comprehend.

The process we used to develop a security model provides a blueprint which can be used to repeat our work. We provide users with the tools to construct a DAG and simultaneously collect notes. The construction materials make the task of capturing the structure quick and clearly visible. Notes capture any information not clearly captured by the construction on the corkboard. This process should generalize to any flow based workflow.

By analyzing the data collected from our users, we derived a set of requirements. We cannot claim these requirements are global. However, certain constructs repeat. These repeating constructs suggest that some requirements span disparate user communities. We identified: existence, number, uniqueness, delegation and granularity as common requirements.

Evaluating a system which lacks real users poses significant challenges. Our evaluation relies on user input as well as formal analysis. User input is used to determine to what extent our security model satisfies their needs. Formal analysis is used to detect internal flaws in the security model. Possible flaws we looked for included: underspecified states, unclear transitions and contradictory statements. We believe we have used an evaluation approach which is appropriate considering the lack of an implementation or a point of comparison.

We have presented a novel security model for an application which to our knowledge has no security model. User studies lend credibility to the applicability of our model. Our requirements and evaluation were derived from input provided by multiple users from a variety of fields. The continued acceptance of provenance as a needed tool, motivates further work in developing a security model for provenance.

8 Acknowledgements

We would like to thank all those who participated in the user studies. Dr. Shlomo Shinnar for providing a sample medical use case and providing contacts in the medical community. Professor Seltzer for her ideas, as well as contacts in human resources and administration. All the students in CS261, particularly the

anonymous reviewers, also deserve our appreciation for their numerous comments on how to improve this paper.

References

- [1] J. Barkley. Comparing simple role based access control models and access control lists. In *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, pages 127–132, New York, NY, USA, 1997. ACM Press.
- [2] J. Crampton. On permissions, inheritance and role hierarchies. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 85–92, New York, NY, USA, 2003. ACM Press.
- [3] E. W. Dijkstra. The structure of the “THE”-multiprogramming system. *Commun. ACM*, 11(5):341–346, 1968.
- [4] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, 1976.
- [5] H. M. Levy. *Capability-Based Computer Systems*. Butterworth-Heinemann, Newton, MA, USA, 1984.
- [6] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.*, 3(2):85–106, 2000.
- [7] G. Saunders, M. Hitchens, and V. Varadharajan. Role-based access control and the access control matrix. *SIGOPS Oper. Syst. Rev.*, 35(4):6–20, 2001.
- [8] R. Sefelin, M. Tscheligi, and V. Giller. Paper prototyping - what is it good for?: a comparison of paper- and computer-based low-fidelity prototyping. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 778–779, New York, NY, USA, 2003. ACM Press.
- [9] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, 2005.
- [10] R. A. Virzi, J. L. Sokolov, and D. Karis. Usability problem identification using both low- and high-fidelity prototypes. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 236–243, New York, NY, USA, 1996. ACM Press.
- [11] Y. Zhao, J. Dobson, I. Foster, L. Moreau, and M. Wilde. A notation and system for expressing and executing cleanly typed workflows on messy scientific data. *SIGMOD Rec.*, 34(3):37–43, 2005.