Using Trust and Risk in Role-Based Access Control Policies

Nathan Dimmock   András Belokosztolszki   David Eyers
Jean Bacon   Ken Moody
University of Cambridge Computer Laboratory
JJ Thomson Avenue, Cambridge CB3 0FD, UK
Firstname.Lastname@cl.cam.ac.uk

ABSTRACT

Emerging trust and risk management systems provide a framework for principals to determine whether they will exchange resources, without requiring a complete definition of their credentials and intentions. Most distributed access control architectures have far more rigid policy rules, yet in many respects aim to solve a similar problem. This paper elucidates the similarities between trust management and distributed access control systems by demonstrating how the OASIS access control system and its rôle-based policy language can be extended to make decisions on the basis of trust and risk analyses rather than on the basis of credentials alone. We apply our new model to the prototypical example of a file storage and publication service for the Grid, and test it using our Prolog-based OASIS implementation.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.4.6 [**Operating Systems**]: Security and Protection—*access controls*

General Terms

Security

Keywords

Trust, Risk, Access Control, OASIS, SECURE

1. INTRODUCTION

Computing entities will continue to become ever more ubiquitous over the next decade. These entities increasingly 'talk' with each other in a peer-to-peer fashion, avoiding the need for large-scale hierarchical communication infrastructures. It is highly desirable that these communities of inter-communicating entities provide a means for sharing or trading resources. In the past such communities would tend to be manually configured in a static manner, however as devices are now often portable, we would prefer to have that configuration occur as automatically as possible.

Acceptance of foreign devices is only prudent if adequate levels of trustworthiness can be ascertained. The term *trust management system* was introduced by Blaze et al. in [6], but the solution it proposes involves an unduly static notion of trust — application programmers choose where to insert code to evaluate their notion of trust, for example at the starting point of a given execution session. Such an approach does not adequately model the fact that these trust values may be continually changing. To take an example from the non-computing world, a customer interacting with a particular business will at first tend to be cautious as to the quality of the service provided, but over time will increase their trust in the quality and reliability of the services provided by that organisation. A good overview of trust management systems is given in [9].

In this paper, we explore access control systems which extend the current practice in which privilege decision outcomes are determined through the credentials presented by a principal. Instead we make it possible to incorporate notions of trust into the checks performed by an access control system during its rule inferencing process. We use the OASIS system, our existing Rôle-Based Access Control (RBAC) framework described in section 3, to validate our ideas and elaborate on potential applications in sections 5 and 6.

Most of the past research combining access control with trust concepts focuses on a trust-management approach in which trust values flow in a manually-defined way through access control policy — for example, using explicit incremental negotiation to establish mutual trust [14, 17]. Our approach instead relies on a computational-trust scheme; trust computation itself is orthogonal to our trust-aware access control architecture and can be included or excluded as required, giving maximum flexibility to the policy specifier.

2. THE SECURE FRAMEWORK

The SECURE[1] project is working towards a trust-based generic decision-making framework for use in Global Computing. One of the target application-areas is Trust-Based Access Control (TBAC), extending our existing work on rôle-based access control (described in section 3) to give the authorisation manager finer-grained control over who they trust.

In SECURE, the access control manager grants or denies permission for *principals* to execute *actions*, as shown in figure 1. A *decision* is a parameterised boolean value — the parameters allow the AC manager to indicate its reasons for denying a request or

To appear in the Ninth ACM Symposium on Access Control Models and Technologies(SACMAT), Yorktown Heights, New York, June 2004.
Please do not distribute.

[1]Secure Environments for Collaboration among Ubiquitous Roaming Entities, European Union project IST-2001-32486.

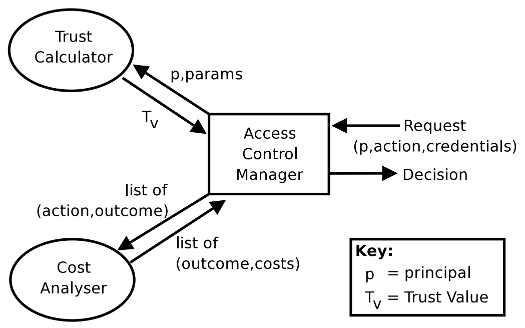

Figure 1: A simplified overview of the SECURE framework

constrain a positive decision.

For every decision the SECURE framework considers the trust it has in the requesting principal p and the risk of granting the request. In our previous work [8], we observed that *risk* is the combination of the costs and likelihoods of all the possible outcomes and we described a model for combining trust and cost information to give a risk metric. The problem with this approach is that the risk metrics were insufficiently expressive to capture all the subtleties conveyed by the trust value. Information is lost since decisions can only be made based on simplistic metrics such as expected benefit and standard deviation. In this new model, we still use an outcome based approach, but allow the policy author to reason about and compare the raw trust and cost information on a per-outcome basis, thereby giving them full-control over the level of uncertainty they wish to permit.

2.1 The SECURE Trust Model

A request by principal p to perform an *action* is submitted to the *access control manager*. The principal may also supply a list of *credentials* which may include signed trust-assertions (recommendations) from other principals, and/or a list of referees whom the trust calculator may wish to contact for recommendations. The AC manager looks up the relevant contexts for the requested action, and queries the trust calculator for a trust-value, T_v about p. The notion of *context* is important in the SECURE trust model. We observe that trust is a multi-dimensional quantity — by analogy, a person who is trusted to drive a car may not be trusted to fly a plane. However, in the absence of directly relevant information, we may infer a trust-value in one dimension from trust-information in related dimensions, so it may be possible to infer some information about a server's trustworthiness to relay e-mail (and not spam) from how much they are trusted to serve webpages. We call these different dimensions, *trust-contexts*.

Trust in a principal is computed by examining *evidence* relevant to the current context. Evidence consists of *observations* of previous interactions we have had with this principal and *recommendations* from other principals, suitably discounted depending on our trust in them [12].

The output of the trust calculator is T_v, a list of (t, c) pairs, where t is the trust-value assigned to p for trust-context c. The domain of a trust-value t_i in trust-context c_i is the lattice T_c over which there are two orderings defined, trust (indicating increasing/decreasing trustworthiness) and information (how much evidence was used to calculate the trust-value) [7]. For each possible action, the policy author specifies which trust-contexts are relevant and the trust calculator forms a projection of the relevant contexts from all the trust-information known about p.

Since principals may be identified by any suitable mechanism, for example, public-keys, biometrics, passwords or similar, and some mechanisms are more capable and/or secure than others, T_v also includes t_{id}, our trust in the mechanism used to authenticate or recognise p [15].

2.2 Making Trust-based Decisions

Intuitively, a high-risk action requires greater trust in its participants and the lower the risk, the less worthwhile it is expending resources in establishing a high level of trust. The majority of computational trust systems, such as [1] and [2], concentrate on aspects relating to assigning a trust-value to a principal; they do not consider policy-driven decision-making using trust.

[18] and [19] make use of thresholding in their policy languages — the former checks that the trust-value is greater than a scalar, while in the latter there must be at least a certain number of evidence statements of at least a predefined level of reliability. However, all of these thresholds are statically determined by the policy author and there is no run-time evaluation of risk.

In SECURE, an explicit cost-benefit analysis is used to determine how much trust is required to offset the risk. While the trust framework is calculating a trust-value for p, the AC manager looks up the outcome costs for the action and checks any specified environmental constraints (for example, time of day), then evaluates a series of predicates which compare trust-values to costs.

3. THE OPEN ARCHITECTURE FOR SECURE INTERWORKING SERVICES

Rôle-Based Access Control (RBAC) introduces a level of indirection between users (principals) and permissions (privileges) and in doing so is intended to simplify access control policy management compared with previous mandatory (MAC) and discretionary (DAC) access control models. In RBAC, users are assigned to rôles, and rôles are assigned privileges. Access control (authorisation) policy is therefore expressed simply in terms of rôles, and user administration merely involves principal to rôle assignment. Most RBAC research involves models more complex than this basic situation. One common extension is the support of *rôle hierarchies*, wherein rôles inherit the privileges of their subordinate rôles. Often this allows the access control rôles to reflect the hierarchy of organisational rôles. Another useful extension is to allow *constraints* to be associated with rôle activation and resource access. Constraints may involve time, database queries and, for rôle activation, may require a principal to already be active in some set of prerequisite rôles. RBAC models supporting constraints thus have a dynamic notion of rôle activation compared to the static principal-rôle assignment functions in simpler RBAC models.

The *Open Architecture for Secure Interworking Services (OASIS)* [4, 5] is a rôle-based access control implementation which includes some novel extensions. It has a formal, logic-based model, which is reflected in the Horn-clause style of its policy language.

OASIS policies define the conditions for activating rôles and those for acquiring privileges. All OASIS policy elements can be parameterised, thus allowing request-dependent information to be included in the access control decisions. OASIS access control policy is specified via two types of rules. An authorisation rule specifies the rôles a principal must hold and environmental constraints that must be satisfied for a privilege to be exercised. A rôle activation rule specifies the prerequisite conditions and constraints that a principal must satisfy in order to enter a rôle.

Rôle credentials are short-lived; that is, since rôles are activated within the frame of a given user session the rôle credentials only persist for the duration of that session. This allows users to limit the rôles in which they are active to those which provide a path to currently necessary privileges. Persistent credentials, such as employment categories and group memberships are supported through appointments, which may also appear in rôle activation rules as prerequisites. Delegation of privilege is also supported, indirectly, through the appointment mechanism – the delegator may issue an appointment to the delegatee which can be used as a credential in the appropriate rôle activation.

Information about the current environment may be assigned to rôle and privilege parameters. OASIS allows environmental conditions to be evaluated through environmental predicates. These predicates may provide two-way communication to components outside the policy-enforcing environment. OASIS parameters are strongly typed and valid binding behaviour is indicated by distinguishing between *in* and *out* parameters. Below we include some example policy rules and introduce the OASIS policy rule notation. Note that in the notation we use in this document, *out* parameters are marked by a question mark postfix.

Authorisation rules map rôles to privileges, and follow the structure:

$$r, e_1, ..., e_{n_e} \vdash p$$

According to such rules the r rôle is assigned the p privilege, if all of the n_e number of e_k environmental predicates are satisfied.

Activation rules have the form:

$$r_1, r_2, ..., r_{n_r}, ac_1, ..., ac_{n_{ac}}, e_1, ..., e_{n_e} \vdash r$$

They may contain any number of prerequisite rôles r_i, appointment certificates ac_j, and environmental predicates e_k. If a principal holds all the prerequisite rôles and appointment certificates prescribed in such an activation rule, and all of the environmental predicates are satisfied, they may enter the target rôle r.

In activation rules certain prerequisites can be marked as membership conditions. Such prerequisites must remain true for the principal to remain active in the rôle, as opposed to only needing to be true at rôle activation time. If the membership condition becomes false (the principal is no longer active in the marked prerequisite rôles, does not hold the marked appointment certificates, or the environmental predicates become false) the target rôle is automatically revoked from the principal.

4. A MODEL FOR TRUST-BASED ACCESS CONTROL

4.1 Trust-reasoning in the OASIS Policy Language

This section presents the first main step towards unifying trust-reasoning into access control, namely integrating trust and cost information into our existing OASIS policy language.

Both OASIS and SECURE support the concept of principals making requests to obtain permission to carry out an action. OASIS models this via its authorisation rules which determine whether a principal has the right to do an action. To fully support the rich range of applications we envisage for SECURE (see sections 5 and 6), we use OASIS's type system to create an action type with which to parameterise privileges and subtypes to distinguish between different sorts of action, as shown in figure 2. Actions are in turn parameterised with request-specific information, such as a filename. Depending on the action type, OASIS will use different functions to access these parameters. In this paper we use capitalised sub-words to denote types (for example, FileAction) and lower case words separated with underscores in parameter names (for example, file_action).

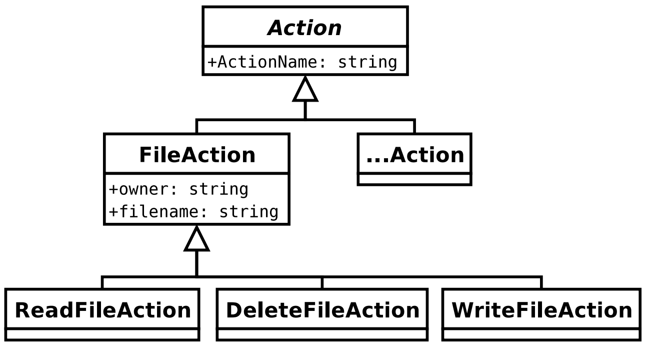

Figure 2: The Action type and its sub-types.

The following functions, encoded in policy rules through the use of environmental predicates, can be used to access parameters of the subtypes shown in figure 2:

$$getActionName(act : Action, action\_name? : String)$$
$$getFileActionOwner(act : FileAction, owner? : String)$$
$$getFileActionFilename(act : FileAction, file\_name? : String)$$

4.2 OASIS Environmental Predicate interfaces to SECURE

Due to the openness of the OASIS environmental predicate call-out mechanism, it is straightforward to define the interfaces required to allow trust and cost/benefit computations to be included within OASIS policy rules. We describe the three main predicates we require:

Trust retrieval: The first predicate uses the contexts associated with the action to retrieve relevant trust values into rule parameters for a named principal:

$$trust(principal, context, value?)$$

Cost retrieval: The second predicate is one which retrieves the set of outcome costs into rule parameters for a named action:[2]

$$cost(action, outcome, cost?)$$

Risk thresholding: This third type of predicate fails if the trust is too low for the outcome's cost. This failure is in the same sense as any other OASIS rule predicate returning false — if using OASIS semantics in which all routes are explored to reach the potential target rôle or privilege, such failure will cause backtracking to attempt other potential activation or authorisation rules. Further details of these *risk* predicates are given below.

4.3 Risk Thresholding Predicates

Although the OASIS policy language is not sufficiently expressive to encode all the policies we require in SECURE, we use the ability to call external services via environmental predicates to extend the OASIS model to provide the required functionality. Our initial investigations have suggested that many policies will be

[2]Unfortunately this formulation means that outcomes cannot be updated dynamically — the structure of the policy rules must also be changed.

fairly simple comparisons of trust and cost metrics which can be expressed in the following basic grammar, described in standard BNF notation:

```
PREDICATE ::= CONDITION
            | if CONDITION then PREDICATE
                             else PREDICATE endif
CONDITION ::= CONDITION CONDOP COND2
COND2     ::= EXPR | COND2 CONDOP2 EXPR
EXPR      ::= true | false | (CONDITION)
            | ARITHEX EXPROP ARITHEX
ARITHEX   ::= ARITHEX2
            | ARITHEX ARITHOP ARITHEX2
ARITHEX2  ::= ARITHEX3
            | ARITHEX2 ARITHOP2 ARITHEX3
ARITHEX3  ::= VALUE | exp(ARITHEX) | (ARITHEX)
VALUE     ::= tval | cost | id | NUM | -VALUE

CONDOP    ::= ||
CONDOP2   ::= &&
EXPROP    ::= == | != | > | < | =< | >=
ARITHOP   ::= + | -
ARITHOP2  ::= * | /
NUM       ::= int | float
```

The standard definitions of the numeric datatypes int and float are assumed. `id` is an identifier (variable name) and `exp(ARITHEX)` is the exponential function, e^x where x may be another arithmetic expression. Operator binding and precendence is based on ANSI C [10].

5. EXAMPLE APPLICATION: THE GRID

While recent years have seen tremendous performance advances in micro-computers, there are still some science, engineering and business applications that require specialised resources that remain beyond the means of an infrequent user. One solution has been to attempt to create something akin to a "power grid" of computational resources — the user "plugging-in" to this grid may access a whole range of services not available at their current location, possibly in return for a fee [3].

As the Internet has shown, exposing computers to the entire world makes for some tricky problems — how does a user know which server to trust with their computation, or which server has the correct version of the document they want to retrieve. Likewise, the server must decide whether this user is a genuine customer if they are likely to pay for the work carried out or if their program will use more resources than the server is willing to allocate. Linking the user to a credit card may mitigate some of these problems, but this is inappropriate for small transactions or pro-bono agreements (for example, where there exist reciprocal agreements between organisations to allow them to use each others' computing resources).

5.1 A file-storage and publication service for the Grid

To illustrate our model, we will apply it to a Grid file-storage and publication service. There are a number of possible use-cases:

- Alice is going on a long research trip, and wishes to ensure some data she needs is widely available, not only for herself, but to potential collaborators she may meet along the way.
- Bob has a very large data set he needs processed by a grid computation server. To save network cost, he wants to host the data as close to the compute server as he can.
- Charlie has a very controversial paper he is expecting a lot of people will want to download. His local sysadmin has told him that their web-server is unlikely to take the strain, and Charlie is also worried that people who object to his paper may try to block its distribution.

Each server offering this service has different properties — apart from their fee, they differ in bandwidth available, uptime guarantee, storage integrity and the strictness with which they check access control credentials. These are all represented in our trust-framework via contexts. Correspondingly, the user must specify the importance of the file in each of these categories. Alice will wish to use several servers that are geographically distributed with average availability and integrity, but are known to be very trustworthy when it comes to not allowing unauthorised people to access her files. In contrast, Bob needs only one server (so can pay a little more for it), but he needs very good availability and storage integrity. Finally, Charlie wants to publish his paper widely on a large number of cheap servers which are trusted for their high level of integrity.

The Grid hosts also use SECURE to make decisions about which clients they can trust. Since there is a certain amount of anonymity on the Internet, there is always the danger that a service vendor will not receive payment, or that they will publish a file that results in the server being overwhelmed or dropped into hot water with the authorities or lawyers! Of course, there are also services which may want very popular files (if they have a fat pipe and an appropriate charging scheme in place) and even ones that actively look to host information that has been censored in other legal jurisdictions (for example, `http://cryptome.org`).

The advantage of SECURE here is that while a Grid host may have never personally dealt with a potential client before, our trust-framework allows us to also consider the reported experiences of its peers. This nicely models human communities in which persons who repeatedly act in an anti-social manner are eventually rejected by the community, thus giving principals an incentive to behave well.

5.1.1 File Retrieval and Deletion

Trust-based reasoning may also be used for determining who should be able to retrieve or delete the shared files. During her travels, Alice meets David at a conference and wants to give him access to a copy of her slides. Rather than having to log into all the servers where the files are stored to update the access control policy, she gives him a signed recommendation that any Grid host containing her files should permit him to download the file, `slides.pdf`.

David returns home from the conference and contacts a server that Alice told him contains the file. Retrieving files does not require payment, so he submits a request to read Alice's file, along with the recommendation she gave him. The server weighs up the strength of Alice's recommendation and its confidence in David's authenticity against the importance Alice has indicated she places in the confidentiality of the file. For a public presentation, only a low confidence in David's authentication as being the David referred to in the recommendation is needed, but for a presentation intended only for project partners, the server may decide the file is sufficiently important to expend resources on further authentication, such as a challenge-response exchange.

A similar process is used when a server is asked to delete a file, but in addition to checking the authorisation and identity of the requester, it also examines their *carefulness* trust-context to check how much the user themselves can be trusted when issuing commands of serious consequence, such as delete! This models the concept of trust in oneself.

Action	Outcomes	Trust Contexts	Trust-value
FileAction (client publishes)	(un)available ignore AC policy too strict on authorisation file is corrupted or deleted	availability confidentiality integrity identification	% uptime, no. of days measured no. of incorrect grants, no. of incorrect denials, total requests no. of corrupted/deleted files, no. of files checked % confidence
WriteFileAction (server)	client doesn't pay popular file (overloaded) controversial file (legal difficulties)	honesty popularity controversialness identification	(belief,disbelief) avg. bandwidth consumption, no. of files evaluated no. of controversial files, no. of files hosted
ReadFileAction	(un)authorised	authorisation identification	(belief,disbelief)
WriteFileAction	(un)authorised (un)intended	authorisation carefulness identification	 no. of mistakes, no. of operations

Table 1: Outcomes for a Grid File Storage and Publication Service

Full details of the possible actions, their outcomes, relevant trust-contexts and format of trust-values are shown in table 1. Most of the trust-values used are self-explanatory, but for a few we use (*belief*, *disbelief*) pairs which summarise the weight of evidence for and against a particular trust-assignment, with *belief* + *disbelief* ≤ 1. This can be compared to Jøsang's logic of uncertain probabilities, based on the Dempster-Shafer theory of evidence [11]. A further complication of the *honesty* trust-context is that it was observed that a person's ability and willingness to pay might be dependent on the amount of money involved. For example, a user might be known to be willing to pay small amounts but tend to renege on the deal if the amount is large, or a credit-card company might recommend that someone is good for up to £100, but no more. To model this we divide the space of possible payments into intervals and compute a (*belief*, *disbelief*) pair for the relevant one. For example, in our prototype implementation (see section 5.2) we use the intervals [0, 10), [10, 30), [30, 60), [60, ∞).

5.1.2 A Publish-File Policy in OASIS

For each server which offers a publication service within her budget, Alice submits the request:

$$request(server\_id,\ file\_action)$$

The $file\_action$ object (of type FileAction) is parameterised with the name of the file to be published, and Alice's name. Alice has specified in the file's meta-data that high file availability is important to her, but she intends to achieve that via replication, so she only attributes medium importance to the individual servers uptime and integrity. The file does not contain highly confidential data so Alice is able to assign a low importance to the server giving the file away indiscriminately. Further, she needs to be able to easily retrieve the file and so she is anxious not to host it with servers which are over-zealous in enforcing access control policy therefore assigning low importance to the enforcement aspect. All of this information is made available to the AC manager via the SECURE cost analyser.

The OASIS policy is then used to determine whether this server is a suitable host for her files as shown in figure 3, where the costs are computed from the information supplied by Alice as to the importance of her file.

$$\boxed{\begin{aligned}&trust(server\_id,\ \text{'availability'},\ t_1),\\ &trust(server\_id,\ \text{'confidentiality'},\ t_2),\\ &trust(server\_id,\ \text{'integrity'},\ t_3),\\ &cost(file\_action,\ \text{'unavailable'},\ cost_1),\\ &cost(file\_action,\ \text{'ignore\_AC'},\ cost_2),\\ &cost(file\_action,\ \text{'too\_strict'},\ cost_3),\\ &cost(file\_action,\ \text{'lost'},\ cost_4),\\ &risk\_unavailable(t_1,\ cost_1),\\ &risk\_ignoreAC(t_2,\ cost_2),\\ &risk\_tooStrict(t_2,\ cost_3),\\ &risk\_lost(t_3,\ cost_4)\ \ \vdash\ \ request(server\_id,\ file\_action)\end{aligned}}$$

Figure 3: The OASIS policy used to determine whether $server\_id$ **is a suitable server for hosting Alice's files.**

5.1.3 A Retrieve-File Policy in OASIS

To retrieve the file Alice has given him permission to download, David submits the following request where the $read\_file\_action$ is paramaterised with (Alice, slides.pdf). The OASIS policy the server uses to determine whether he may read the file is shown in figures 4 and 5.

$$request(David,\ read\_file\_action)$$

The costs Alice assigned to her file are given in section 5.1.2, from which we can deduce that provided the server is over 60% confident that is it "David" that it is interacting with, since $cost_1$ is "low", the server must just have more belief than disbelief that David is authorised to read the file to grant him access to it. Since David has supplied a recommendation from Alice strongly recommending that he be able to read the file, unless the server has further information to the contrary (perhaps another recommendation from Alice with a later time-stamp revoking that authorisation) it will therefore allow him to read the file.

5.1.4 More Complex Policies

The expressiveness gained by using the OASIS infrastructure is clearly demonstrated if we consider that Alice may also make recommendations about rôles, as well as individuals. For instance, she might recommend that all members of the University of Cambridge Computer Laboratory can retrieve her presentation, but members of the University of Oxford may not read any of her files.

$$\begin{array}{l} trust(principal, \text{`authorised'}, t_1), \\ trust(principal, \text{`identification'}, t_{id}), \\ getFileActionName(read\_file\_action, file\_name), \\ cost(file\_name, \text{`authorised'}, cost_1), \\ cost(file\_name, \text{`unauthorised'}, cost_2), \\ read\_file\_risk(t_1, t_{id}, cost_1, cost_2) \\ \qquad\qquad \vdash request(principal, read\_file\_action) \end{array}$$

Figure 4: The OASIS policy used by the server to determine whether a principal may access a file.

```
if trust_id < 0.6 then
  false
else
  if cost1 == low then
    t1.belief - t1.disbelief > 0
  else
    if cost1 == high then
      if cost2 == high then
         t1.belief - t1.disbelief > 0.6
      else
         t1.belief - t1.disbelief > 0.7
      endif
    else
       if cost2 == high then
          t1.belief - t1.disbelief > 0.5
       else
         t1.belief - t1.disbelief > 0.6
       endif
    endif
  endif
endif
```

Figure 5: Definition of the $read\_file\_risk$ predicate.

5.2 Implementation

To check the viability of the ideas presented in this paper, we integrated the SECURE trust predicate interface into our Prolog OASIS implementation. For non-prototype deployment, we would expect to rework the necessary environmental predicates into our more heavy-weight J2EE OASIS implementation.

For example, translating OASIS XML policy to check whether a user is willing to store a given file on a particular server, we get the Prolog OASIS shown in figure 6. Note that this is a particular instance of a FileAction, with a set parameter structure (server ID followed by the filename). This is a consequence of the Prolog translation; in fact our XML policy format allows more flexibility than this when specifying the binding of parameters within rules.

For our tests, the SECURE properties for trust and cost relating to users, servers and objects are encoded statically with facts of the following form — that is, the SECURE trust and cost properties are static and locally available.

```
trustProperty(server(1),availability=[99,1]).
% ...
costProperty(file(testFile),authorised=high).
costProperty(file(testFile),
             unauthorised=high).
% ...
```

As a consequence of this encoding, we could have implemented

```
privilegeRequest(fileAction_store,
                 [ServerID,FileName]):-
        % check policy parameter modes
        nonvar(ServerID),nonvar(FileName),
        % check prerequisites
        trust(ServerID,availability,T_1),
        trust(ServerID,confidentiality,T_2),
        trust(ServerID,integrity,T_3),
        trust(ServerID,reliability,T_4),
        cost(FileName,unavailable,Cost_1),
        cost(FileName,ignore_AC,Cost_2),
        cost(FileName,too_strict,Cost_3),
        cost(FileName,lost,Cost_4),
        risk(unavailable,T_1,Cost_1),
        risk(ignoreAC,T_2,Cost_2),
        risk(tooStrict,T_3,Cost_3),
        risk(lost,T_4,Cost_4).
```

Figure 6: The Prolog OASIS form of the policy used to determine whether to store a file on a server.

trust and cost predicates within the OASIS XML policy. Our choice not to do so emphasises the OASIS/SECURE interface which would be necessary to support interaction with an external SECURE engine. The interfaces to retrieve SECURE properties within OASIS policy were thus able to be implemented as follows:

```
trust(User,Property,Value):-
          trustProperty(User,Property=Value).
% ...
cost(Object,Property,Cost):-
            costProperty(Object,Property=Cost).
```

The `risk` predicates are encoded into Prolog from our risk expression syntax given in section 4.3. This translation is done using the Definite Clause Grammar (DCG) extensions of SWI-Prolog (DCGs are not part of ISO Prolog, but commonly appear nonetheless). A simpler example of a SECURE-influenced OASIS Policy rule is given below, in which a principal requests to read a particular file. For the FileName parameter to make sense, this rule would need to be evaluated on the server protecting the file in question.

```
privilegeRequest(fileAction_read,
                 [Principal,FileName]):-
        % check policy parameter modes
        nonvar(Principal),nonvar(FileName),
        % check prerequisites
        trust(Principal, authorised,T_1),
        trust(Principal, identification, T_id),
        cost(FileName,authorised, Cost_1),
        cost(FileName,unauthorised,Cost_2),
        readFileRisk(T_1,T_id,Cost_1,Cost_2).
```

A number of manual privilege requests were issued to check that our framework was producing the expected results. Integration with the under-development SECURE engine to allow us to advance beyond this proof of concept level is in progress.

6. OTHER APPLICATIONS

Given the similarity of the goals of Grid computing and peer-to-peer systems, namely the pooling and coordination of the use of distributed resources, our system is likely to be applicable in peer-to-peer systems, and would provide considerably greater expressiveness than existing trust models designed for peer-to-peer applications [16, 2].

The large-scale, widely-distributed and unpredictable nature of pervasive computing systems along with their limited resources, also makes it difficult to use existing access control models. A set of requirements for a trust-based policy language for use in a pervasive environment are described in [13] and we believe our model meets these.

Other potential applications include distributed spam detection, more powerful reputation systems and online auctions — all currently being investigated by the SECURE project.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have demonstrated how to extend existing access control architectures to incorporate trust-based evaluation and reasoning, and the resulting advantages — namely a more dynamic form of policy that can reason with uncertainty and explicitly manage risk. We have further demonstrated how this model can be applied to a file-storage and publication service for the Grid and explored its application to the areas of peer-to-peer and pervasive computing.

We have validated our ideas through our prototype Grid application implementation, and work is on-going with our project partners to further integrate this work with the SECURE engine, and develop a spam filtering application, among others.

Future work includes exploring automated policy evolution, by allowing risk to evolve in a similar manner to trust. We foresee the possibility of the outcomes and costs being updated based on evidence already gathered by the SECURE framework for the purposes of updating trust beliefs. We also believe we can improve the efficiency of our engine by making use of OASIS sessions and caching potentially expensive trust-calculator lookups where the risk from stale information is acceptably low.

8. REFERENCES

[1] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *Hawaii International Conference on System Sciences 33*, pages 1769–1777, 2000.

[2] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the Tenth International Conference on Information and Knowledge Management(CIKM01)*, pages 310–317. ACM Press, November 2001.

[3] Ashton Applewhite. Getting the Grid. *IEEE Distributed Systems Online*, May 2002.

[4] Jean Bacon, Ken Moody, and Walt Yao. Access control and trust in the use of widely distributed services. In *Middleware 2001*, volume 2218, pages 300–315, November 2001.

[5] Jean Bacon, Ken Moody, and Walt Yao. A model of OASIS role-based access control and its support for active security. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):492–540, November 2002.

[6] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proc. IEEE Conference on Security and Privacy*. AT&T, May 1996.

[7] Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A formal model for trust in dynamic networks. Research Series RS-03-04, BRICS, Department of Computer Science, University of Aarhus, January 2003. EU Project SECURE IST-2001-32486 Deliverable 1.1.

[8] Nathan Dimmock. How much is 'enough'? Risk in trust-based access control. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises — Enterprise Security*, pages 281–282, June 2003.

[9] Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Society, Surveys and Tutorials*, 3(4), 2000.

[10] ISO/IEC JTC1/SC22 Working Group. *ISO/IEC 9899 - Programming languages - C*, 1999.

[11] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(9), June 2001.

[12] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Analysing topologies of transitive trust. In *Proceedings of the Workshop of Formal Aspects of Security and Trust (FAST)*, September 2003.

[13] Lalana Kagal, Tim Finin, and Anupam Joshi. Trust-based security in pervasive computing environments. *IEEE Computer*, pages 154–157, DEC 2001.

[14] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *2002 IEEE Symposium on Security and Privacy*, pages 114–131. IEEE, May 2002.

[15] Jean-Marc Seigneur, Stephen Farrell, Christian Damsgaard Jensen, Elizabeth Gray, and Chen Yong. End-to-end trust in pervasive computing starts with recognition. In *Proceedings of the First International Conference on Security in Pervasive Computing*, 2003.

[16] Li Xiong and Ling Liu. Building trust in decentralized peer-to-peer electronic communities. In *The 5th International Conference on Electronic Commerce Research*, October 2002.

[17] Walt Teh-Ming Yao. Fidelis: A policy-driven trust management framework. In *Proc. of the 1st Intern'l Conf. on Trust Management*, number 2692 in LNCS. Springer-Verlag, May 2003.

[18] Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *1st Intern'l Joint Conf. on Autonomous Agents and MultiAgent Systems*. ACM, July 2002.

[19] Yuhui Zhong and Bharat Bhargava. Authorization based on evidence and trust. In Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa, editors, *Data Warehousing and Knowledge Discovery(DaWaK 2002), 4th International Conference*, volume 2454 of *Lecture Notes in Computer Science*, pages 94–103. Springer, 2002.