SCALABLE ACCESS POLICY ADMINISTRATION

Opinions and a Research Agenda

Arnon Rosenthal
The MITRE Corporation

Abstract: The emerging world of large, loosely coupled information systems requires major changes to the way we approach security research. For many years, we have proposed construct after construct to enhance the power and scope of policy languages. Unfortunately, this focus has led to models whose complexity is unmanageable, to reinventing technologies that other subdisciplines have done better, and to assumptions that large enterprises simply do not satisfy. We argue that it is time to emphasize a different challenge: radical scale-up. To achieve this, it will be crucial to emphasize simplicity, integration with (non-security) enterprise knowledge, and modularity for both models and administration. This position paper will illustrate the problems, and describe possible ways to achieve the desired capabilities.

Key words: Policy administration; access policy; scale; role based access control; semantic web; simplicity; security; privacy.

1. INTRODUCTION

We can no longer rely on inaccessibility to protect data integrity or confidentiality. In ancient times (i.e., the 1980s), only a few users could reach each data object, due to barriers of system connectivity, protocols, and data format standards. These barriers have diminished, due to cheaper storage and communication, standards, software advances (e.g., service oriented architectures, data integration environments), and new practices (e.g., cross-domain guards examine traffic to protect networks previously isolated for security). In today's interconnected world, we need appropriately selective access control policies.

There are daunting challenges in administering such access control policies at enterprise scale (including virtual enterprises, such as supply chains and military coalitions). A grand challenge for our field (IFIP 11.3's name, *Data and Application Security* seems appropriate) is to create a foundation for specifying, changing, reasoning about, and enforcing such policies.

The enterprise problem seems to have two fundamentally different sorts of challenges –specific policy needs, and scale.

For accommodating the many specific real world policy needs, our field has developed a rich base of "point" techniques. These include privilege limitation (e.g., negative policies, predicates, revocation), result filtering, separation of duty, inference control, temporal and spatial conditions, audit log mining, chains of trust, and so on – infinitely. The *scale* picture is less encouraging, due both to sheer volume (of enterprise knowledge and of policy specification), and to the problem of integrating the different techniques (from above) into commercially viable, extensible tools.

Our Thesis: **Our research community should treat scale as the *central* unmet requirement for an enterprise security system.**

Large scale data and application security is challenging both for enterprises and for the vendors who build tool suites to simplify administration. In our opinion, progress has been very slow, and one reason is the lack of research results that are worthwhile and ready to transfer. Many researchers' models contain good insights, but are neither simple enough nor robust enough to have attractive cost/benefit. Enterprises and tool vendors therefore choose to invest in other areas (such as GUIs to existing engines).

Most current research (in our unscientific sampling) concerns new specific capabilities. We pose the questions: Will we improve most by creating new features which will then be difficult to integrate? Or will techniques that promote scalability (both policy and integration) have a higher payoff?

1.1 Paper Goals and Roadmap

This is a very informal position paper, intended to stimulate discussion of the "big picture". We ask throughout whether conventional data security research (e.g., at ACM SACMAT, VLDB TDM, or IFIP 11.3 Data and Application Security conferences) is on track to produce a foundation for scalable security administration. Our goal is to provoke debate, and to provide useful questions and metaphors. To that end, we make broad claims about (painfully) vague notions like "simplicity". We hope our claims are 80% true; we welcome feedback and refinements. We leave it to the reader to determine whether our diagnoses apply to their own research.

Our aim is to encourage researchers to produce results that will be useful for enterprise problems. Elegant results are always good research; even if the original motivation is weak, they provide insight into other problems. Intricate techniques for minor challenges are not. Assemblages of incomplete and conceptually overlapping capabilities are suitable for initial exploration, but they add little clarity to techniques or architectures, and their immaturity makes them poor investments for implementers. We focus on being more *scalable* in proposed features, models and system designs. We hope the criteria remind researchers of interesting questions, and reviewers of under-emphasized evaluation criteria.

To be clear, we believe that a result can be valuable without being enterprise-ready. In fact, we point at a paper as problematic for scalability only if we felt it had other significant research contributions. But it is a serious problem for the field when perhaps 80% of results violate scalability tenets. An enterprise wants to build its initial systems on ideas that extend to large scale, not to require a wrenching transition and separate skill base.

Three important prerequisites for scalability seem absent from the majority of access control research papers:

- *Simplicity*: We should not needlessly sacrifice the power to capture enterprise complexities. Yet if our formalisms are too expressive or too complex, we will lose two critical capabilities: the leverage of automated reasoning-based assistants, and ability to widely share the administrative load (e.g., with domain experts and less technical security personnel). Section 2 proposes a principle – ruthless simplicity -- for avoiding both dangers.

- *Reaching out*: Access policies involve organization and domain knowledge, and express conditions on all sorts of data (e.g., Time and Space). Unfortunately, we often treat access control as a self-contained subsystem, both for formalisms (e.g., role based access controls, spatial data types) and for enterprise descriptions (e.g., the enterprise's organization chart as part of a role hierarchy). As a result, access control systems have limited access to external tools and to enterprise knowledge.
- *Modularity*. Few research papers componentize their models to export well-defined services. Fewer still define interfaces that reach out for services beyond the paper's core idea. Research ideas would be much easier to use in large systems if they fit with other peoples' partial solutions, instead of specifying how to handle the peripheral issues.

Section 1.2 reviews general requirements for access controls, and explains what we mean by Scalability. Sections 2 and 3 respectively discuss Simplicity and the need to reach out, rather than "do it yourself" just for the security system.

1.2 Requirements Discussion

The enterprise's tasks include integrating the tools from multiple vendors, training administrators, and especially administering the policies. Tasks involved in administering the policies include:

- *State policy from business viewpoint*. For example, a hospital upper management committee might state a governing policy, in English, over broad classes of information such as Patient Treatment, Patient Outcomes, Hospital Cost data, and so forth, for release to appropriate medical personnel, researchers, and regulators.
- *Based on the governing policies, derive a "concrete" policy on implemented objects.* Upper management does not specify a policy for every object in the automated system (e.g., table, message, or service). Instead, one must implement the intent of the business policy by deriving a suitable policy to be evaluated upon access to each concrete object, e.g., for Read privileges on a Toxin_Screen message type.
- *Translate the derived policy down to the physical enforcers.* These are likely to use a variety of languages (e.g., DBMS and middleware, with vendor variants.) The forces for heterogeneity (e.g., mergers, decades of legacy, decentralized acquisition decisions, distinct needs; virtual organizations that must interoperate with many constituent companies) are not disappearing. Diversity will continue.
- *Understand the net effect of all the policies that the system is actually enforcing.*
- *Change policy, and understand the consequences.* To help manage change, some models propose ways to edit policy sets, e.g., temporarily enabling or disabling. We are unsure if these belong in a security model or just in an editor. Automated reasoning is needed to help administrators understand how a change affects an object of concern to them.

Today, most of the above tasks are done manually. Many of the techniques scale poorly in terms of policy specification, administrator training and creating tool environments. Enterprises therefore compromise, to reduce specification labor and required skills. For policy, they can manage only coarse distinctions, e.g., based on what organization a person belongs to, or what data the network is on. The resulting policies are not really appropriate.

Tools are also inadequate. Vendors find that access control tool suites involve many parts, often more than one company can provide. Standards (e.g., XACML) are currently weakly supported. An enterprise faces a substantial consulting cost to integrate or migrate the various products used within it.

2. SIMPLICITY

Enterprises' policy management involves many people, and a huge number of policies. Tools need to provide numerous services (e.g., specification, analysis, enforcement) for a large number of constructs. To scale, the central foundations need to be *very* simple. A complex initial model when compounded by implementation needs, will render the whole confusing, opaque, and inextensible. In addition, formalism expressiveness (which enables richer input to reasoning and tools) needs to balance with the computational cost or undecidability of complex formalisms. Finally, administrators will reject features or environments that seem too difficult. We need models usable by administrators may who are unskilled or unmotivated – think Homer Simpson rather than CS graduate student.

We fear that in our data security research community, most proposed models are already too complicated, and thus, we believe, an unsuitable conceptual foundation. We speculate that researchers are motivated to seek a model that will meet all real requirements. The usual response to an unmet requirement is to add constructs to the model. Complexity is the consequence.

Many of the challenges faced by enterprise policies were previously faced by data management, and we find the parallels provocative and informative. For example, both data management and access control policies touch all the enterprise's structured data and many of its services. Both areas need to empower domain experts rather than rely on technical experts. Both have employed declarative formalisms (e.g., SQL, role hierarchies) that enable automated reasoning. Below, Section 2.1 gives a whirlwind tour of database history and its relevance to access control. Section 2.2 focuses on simplicity.

2.1 Parallels with the History of Data Management

Research has played a major role in creating the DBMS industry, which is large (according to Gartner, >$10B [PM]) and furnishes the base tier of enterprise architectures. Somewhat surprisingly (to this researcher), the contribution may not be primarily from innovation. The head of Microsoft's database research observes that developers (who outnumber researchers 60:1) are quite smart enough to innovate. Rather, the crucial and unique contribution of researchers was in clarification and formalization, i.e., *simplification*. [Lo]. This seems a key role for access control researchers, too.

Looking at databases in enterprises, we note that the main cost of database ownership is now salaries for administration, rather than hardware or software purchase. That is, increasingly, administration cost and responsiveness often determine if an application is cost effective. Today, access control activities at MITRE's sponsors (typically, large US government organizations) are mainly at an earlier stage. They are now deploying mechanisms that controls rely on (e.g., authentication, trust, policy engines, role and attribute based access control). There is also concern about efficient evaluation. Within a few years, however, we expect that policy administration will be recognized as a major roadblock, in cost, project delays, and guaranteeing adequate assurance.

Another lesson is that database tools were able to greatly reduce the skill requirements. As design methodologies were originally created, there was discussion of how to teach their ideas to the mass of business experts who were suddenly designing data schemas. This never happened. Instead, the subtleties (e.g., dependency theory) were built into the tools, and most administrators learned just enough to supply the required inputs. Where the inputs to a technique were too hard to supply (e.g., multi-valued dependencies), the technique withered. So rather than rely on better trained administrators (perhaps Lisa Simpson?), access control also should resolve subtleties *inside* the tools.

Having even an imperfectly-observed standard DBMS language was tremendously valuable, in an unanticipated way. The original goal – enabling enterprises to port from one product to another—was rarely cost effective. However, vendors of application development and end user query tools could (at bearable cost) make their products work with any of the ~5 major DBMS products. The result was the development of the "front end" application development industry, distinct from DBMS vendors. A similar split might be beneficial for security tools. In fact, the standard for Role Based Access Control appears to have been highly beneficial [NIST]. We can also learn from mistakes of DBMS standardization. As the SQL standard expanded, there was too little investment in abstraction to "factor" common portions out of the new pieces. Partly as a result, the detailed standard is huge, has minor inconsistencies in treatments of similar situations (e.g., access controls for procedures versus views), and is understandable only to a small priesthood. It is not too soon to seek a more elegant, more formal basis for XACML (e.g., to move difficult-to-scale conflict resolution to relevance predicates).

SQL security is not the principle means by which enterprises protect data; most protections are in middleware or in application code. Still, the number of systems using it is still very large. Extrapolating to all vendors the estimate that 30% of Oracle installations make nontrivial use of SQL security (W. Maimone, personal communication), we have 30% of >1 million installed systems. Oracle has offered both roles and delegation (grant option) since the 1980s, and recently Microsoft SQL Server implemented some negative privileges.

SQL *industrial* experiences deserve more attention than they get from security researchers – even if one (reasonably) believes that SQL's strengths (e.g., grant option) deserve less attention than the general predicates of XACML. For example, the very useful survey [BS] focuses only on the academic literature. Roles are indeed recent – in the research literature. Also, it describes time-sensitive Revoke semantics from early research papers, rather than the easier-to-administer time-insensitive operations from DBMS products and the SQL standard. Researchers could use SQL security as a source of experience with various constructs (e.g., how much is delegation used?) and as prior art when proposing new constructs. Perhaps more important, researchers could help abstract and extend SQL. Specific needs, especially those suitable for graduate student projects, are discussed in [RW]

In the 1980s, demands arose to add many new datatypes to DBMSs, e.g., for temporal, spatial, and textual data. SQL now includes low-end constructs such as a DATE datatype and simple text matches with wildcards. But the community rejected adding a *particular* form of each rich domain. Needs were simply too varied, e.g., should spatial locations use flat or spherical geometry, and the specialized expertise was outside the database community. Instead, the major vendors have provided general mechanisms for adding new data types (e.g., to specify language operators, storage, indexing, etc.), and then partnered with other organizations to meet the wide variety of specific needs.

Finally, we learned that to get administrators to supply accurate, current metadata, one must have the right incentives. Data files and queries are generated from database schemas, so these are current. In contrast, database conceptual models (e.g., entity relationship) almost invariably become obsolete shelfware after a system is deployed. Mandates to keep it up to date are ignored, or else errors creep in. As a result, quality problems soon make the conceptual models unusable. Similarly, we will find it difficult to keep policies or role hierarchies up to date if the burden is entirely on security administrators. Where possible, one wants to exploit information that users have other reasons to keep current.

2.2 Getting a Simpler Model

Access control models in the research literature have become quite complex. They involve a wide variety of features, often with unscalable assumptions (e.g., "all metadata is public") and restrictions on applicability (e.g., not applying to views or services). In the world of policy engines, some proposed languages (e.g., Ponder) are so powerful that they can be used only by professional programmers, and tools cannot effectively analyze their behavior. Even the simpler OASIS standard language XACML has a wide variety of features that seem difficult to mesh with policies that apply to different granules of the database (table, row-set, or "all data with personally identifiable information").

We have a dilemma: All these complications were motivated by real requirements, but the whole will not scale. To escape it, we propose a two-model (or perhaps n-model) approach. We would not design the same vehicle to act as both a bicycle and a fuel truck. Analogously, proposals [KK+] that we create a unified formalism for all our work may be suboptimal.

We base our thinking on database (and pre-database) history. In the early 1970s, many practitioners recognized the need to scale data sharing beyond the files managed by a single business application. Two lines of work ensued. Some researchers looked at the full spectrum of data management requirements, and observed the need for additional operators. They therefore added richer datatypes and persistence to COBOL, PL/1, and other popular general purpose programming languages. The result was to make the languages more complex and skill intensive. Also, since the languages were already Turing complete, there was only limited ability to provide analysis, synthesis, query optimization, and automated GUI generation tools.

At the same time, IBM researchers took a radical step – they divided business data processing into two portions. The new portion provided a clean, rigorous abstraction of manipulations of tables (relations). The "old" portion consisted of applications coded in general purpose programming languages, plus requests to the relational database. The researchers also took on the difficult task of interfacing the two portions.

The relational model they created was ultra-simple, insufficient on its own even for routine tasks such as generating hierarchically structured reports. Yet this is the approach that succeeded.

The relational researchers ruthlessly exiled all tasks they could not (at that time) handle cleanly. But within their chosen arena, they could support a simple abstraction with very powerful services. SQL was able to insulate users from details of iteration, access strategy, and file and index structures. Virtual tables were definable, and were useful for both convenience and security. The implementations were strengthened gradually, until they could handle huge loads. As discussed in Section 2.1, an industry of application development tools developed, evolving more quickly than the performance-optimized database engines. Later decades added additional capabilities, sometimes as separate models, without (excessively) compromising tractability. For the 1990s, the great extensions were triggers, distribution, and objects (including the ability to plug in numerous managers for temporal and spatial data). In our decade, DBMSs are supporting irregular and hierarchical data (i.e., XML). The (relatively) simple models have enabled great progress and became the foundation layer of enterprises' data architectures.

Of course, there are many aspects of database history that access control researchers and tool developers ought not to emulate. First, the interface to the residue capabilities is awkward. (M. Stonebraker memorably referred to the interface between databases and application code as "like gluing an apple to a pancake".) The conceptual and run-time costs of this interface must be included in assessing SQL cost/benefit. Also, databases lack a clean model of sequential execution, and this has complicated both the programming language interface and event-based features (triggers, handling constraint violations). Researchers need to continue extending the abstract foundation, rather than leave all extensions to ordinary programmers. Finally, as we will discuss in Section 3.2, we should keep closely aligned with abstract models of enterprise knowledge.

Much security research today resembles adding operators to create persistent COBOL. To change this, we make a radical proposal"

Proposal 1: Take "simplicity" as an absolute constraint on for the "foundation" model

That is, access control research should emulate the database field by pursuing a two-model (or n model) approach. The foundational models should *rigorously* exclude anything that is not tractable for knowledge capture, visualization, and reasoning.

This foundational model should only include treatments that are suitable for *all* uses. It might include services to create or delegate administrative rights. It should not hardwire choices that depend on semantics of the particular enterprise, e.g., whether Deny always overrides Allow. As an example of such a model, one might start with positive policies with predicates (e.g., provisions) and on object sets of different granularities. It may recognize that there are a variety of relationships among objects (e.g., IS-A, Part-Of, predicate-based restrictions as an alternative to negatives) and that each will have its own security semantics. (Section 3.2 suggests supporting just the relationships present in the OWL-DL language.)

One other line of research is badly needed – simplicity metrics. To achieve simplicity, then both researchers and acquisition contracts need to measure it, at least comparatively. Most new proposals make no attempt at the issue, and we know of no general approach. Even in our experience, we have too often conversed about model constructs, with each party claiming (to the other's puzzlement) to have a clearly simpler model. We applaud recent research that gives practitioners' insight into models' relative *logical* expressiveness [TL]. However, logical expressiveness gives little insight into scalability; it is central only to logicians. We hope the spirit of comparison might move on to formalizing and comparing criteria that *are* central -- simplicity and administrator workload.

3. THE TENDENCY TO “DO IT YOURSELF”)

Rich access control policies involve substantial knowledge about the enterprise and its environment, e.g., relationships among users, and natural groupings of privileges. This security-*relevant* knowledge is then captured in security-*specific* formalisms, to be managed using security-*specific* tools. We contend that this approach will not be suitable for the long term. There is a rich tradition, and growing industry, of knowledge management capabilities outside the security industry, and enterprises have many other uses for such structural knowledge. The redundancy increases cost and promotes inconsistency, with neither side getting full information. We need to reuse outside resources, not reinvent them in the security system.

We now look at three examples of reinventing rather than reusing. Section 3.1 discusses specialized datatypes, e.g., for Time and Space. Section 3.2 contains the bulk of the discussion. It looks at capturing organizational descriptions within the security systems, e.g., the organization of users, privileges, data, and operations.

3.1 Specialized Datatypes

Policies need to be sensitive to temporal or spatial conditions, so researchers have proposed corresponding extensions to access control models, e.g., [BJ+, AC]. This practice seems appropriate for prototypes, but if one takes it as a proposed model, we have two serious concerns:

- *Each proposal supports just one out of many viable treatments.* For example, should a spatial extension to access control models offer 2-D or 3-D, flat or spherical or ovoid geometry for the earth? Each is needed by a large community; each requires a complex implementation, including distance computations search indexes, and displays.
- *Lack of expertise.* We have expertise in security. The spatial and temporal research communities are far more expert in their domains. Do we intend for reviewers at security conferences to judge the novelty of a proposed spatial model?

It would be *far* better to allow applications and enterprises to reach out. Applications employ dozens of spatial and temporal facilities, to meet their own needs. Services are also available for text, imagery, audio, and video. We should not settle for one treatment in each domain, and then keep up with progress. An attempt to implement our own best try will leave us with a few obsolescent functions.

Since access controls do need to cope with data from these rich worlds, what should the responsibilities of security researchers? First, we should describe policy needs involving these multimedia types. Second, provide design guidance in being able to plug such datatypes into a security model. Is any change needed in security policies and tools when query language is extended with new datatypes? Will policies require very fast checks of certain predicates, and hence suitable indexes? There may also be architectural guidance, e.g., about whether the extensions are better made part of the query system or the security system, and whether there are special vulnerabilities (e.g., to resetting a workstation clock, or the need to evaluate security-sensitive predicates before transferring control to user-defined functions). Finally, researchers could identify *general security capabilities* one needs in order to plug in such predicates well. For example, a *general* (non-spatial) security feature that understands “contained in” might be useful when administrators grant access to large spatial regions, but each user requests ask for a tiny rectangle.

3.2 Describing the Organization

Access control decisions need to examine a myriad of considerations about the user, the request, the object accessed, and the world context. One major complication is that the policy-maker’s terminology probably differs from that used in the automated system’s requests, databases, and messages. For example, the policy may refer to “Product Planning”, “Customer-Related Information” and “Massachusetts customers” while the request comes from a Yield-Analyst, asking about Deliveries to Zipcode 02138.

Many organizations use role based access control (*RBAC*) to begin to organize such knowledge. The textbook [FMC] provides excellent descriptions of many RBAC approaches, which for our purposes includes both pure roles (NIST) and also role+group approaches. We believe that RBAC designers found a sweet spot for the 1980s and 90s, offering the following strengths:

- An easily-grasped means of organizing knowledge, with an intuitive visualization (acyclic digraph) and editing operations.
- A standard, which enables products to interoperate. (For example one vendor can provide a management interface, and another an enforcer).
- A formal and computationally efficient inference theory. (RBAC infers authorization, based on role activations and reachability).

Unfortunately, RBAC is painful to apply to large enterprises [KK+], for the reasons listed below. Section 3.2.1 will elaborate and discuss the limitations of access control researchers’ proposed ameliorations.

- The administration team needs to convert the enterprise’s complex structure to a single graph (or perhaps a role and a group graph). Having one graph for enterprise structure is analogous to defining an enterprise database using one table.
- RBAC formalisms are confined to the access control community. One cannot expect a rich base of tools, or of trained people, compared to the resources for knowledge management in general.
- Enterprise knowledge is split into roles versus ordinary data. Such barriers have very predictable effects, reducing the information available on each side. Yet role information may be useful for budgeting (e.g., how many people are in Medical roles) and ordinary management information (e.g., what customer is Joe assigned to today) is useful for security. Attempts to import data across the barrier tend to be expensive and incomplete.

A decade ago, general knowledge management formalisms could not compete with RBAC. For example, Datalog offered more representational and inference power, but lacks an interface standard, a convenient typing mechanism, and a graph visualization. Other formalisms had tiny user communities and inefficient inference (e.g., the AI community’s Knowledge Interface Format). Despite the above limitations, RBAC was the right choice then. But times are changing.

We believe that the “general knowledge” approach is on the verge of being industrially viable. These approaches provide a formal logic, plus constructs that help humans capture and organize knowledge. For example, for representing binary relationships, OWL (the web ontology language) is a standard endorsed by the World Wide Web Consortium (*W3C*). A significant experimenter and early adopter community uses OWL for knowledge capture and as a basis for data integration. Inference is complete and reasonably efficient at the first two compliance levels.

Other researchers propose more powerful languages, e.g., SWRL and Common Logic, but there has been less convergence. Following Section 2.2, we suggest splitting user specifications between one simple language (e.g., OWL) and one more powerful one. But our goal is not to advocate specific languages. The idea is to adopt suitable languages that are is gaining traction in the wider community.

Proposal 2: It is time to move access control research and tools to general knowledge base formalisms rather than by adding features to RBAC

It is time to apply the principle "Don't do it all alone! Access policies need to reflect the structure of the enterprise and its environment. We need to exploit both knowledge management expertise, and to share knowledge with the rest of the enterprise. RBAC was a fine idea for its time, and its core functionalities (aggregating users and privileges, activation for authorization) will always be needed. But it is time to reorganize to separate its security aspects (role activation as precursor to authorization, policies on which groups can exercise which roles, under what circumstances) from its enterprise knowledge.

- *Better expressive power and structuring, while (mostly) preserving simplicity.* GUIs can hide the complexity from administrators with simple tasks. The system architecture is simplified by removing the dichotomy and the extra formalism.
- *Larger user and tool communities.* The semantic web community is already beginning to provide skilled users and tools (including freeware) for OWL, including interfaces, reasoning engines, and import/export wrappers to exchange data to other formalisms (relations, XML, etc.)

3.2.1 RBAC and its proposed extensions

RBAC (especially the "only roles" form from NIST) makes enforcement easy (no doubt a factor in adoption by vendors), but is impoverished as a means of describing an enterprise. We now discuss limitations both of RBAC and of attempts at extensions.

RBAC offers just only a small, fixed set of types, essentially only Role and perhaps Group. (To a first approximation, User and Privilege are just singleton Groups and Roles). When administrators map their understanding of the enterprise to a role hierarchy, the mapping resides in human heads -- notoriously low capacity, unreliable storage devices. This mapping is not explicitly stored, hard to edit, not backed up, hard to share among the different heads, and inaccessible to display or reasoning tools.

A first step beyond NIST RBAC is to add a second fundamental node type – group. (This step has been well described by Sandhu and by Osborn [FKC]). Next, [KK+] proposes a third primary type, representing processes (i.e., operations). These fixed extensions help administration, but are not sufficient. For example, for grouping people in an organization, one aggregates along many dimensions – rank, organization, project they work on, skills, and so forth. Oracle Label Security offers an extensible grouping capability, but it is proprietary and limited to access control uses. It is best to let enterprises define their own descriptive categories. They need a general mechanism to define new types.

The need to express conditionals is well known, e.g., that Gold_Frequent_Fliers are Passengers with Mileage > 50000, or that certain privileges require mileage >75000. To augment User with attributes like Mileage, several researchers allow roles to have parameters [LMW, KK+]. However, we now have two differently-behaved categories, first class (roles, groups) and second class (parameters that cannot aggregate into graphs). We also need to support a parameter mechanism (which OWL natively supports as properties) as well as a predicate language.

Attribute Based Access Control (ABAC) seems simultaneously more elegant and more powerful for enterprise modeling. As a security model, it provides an authorization rule. The bulk of it, though, is concerned with defining attributes and their structuring operators. For example, [WWJ] provides a technically powerful and elegant logic for the KM aspect of security. It seems to have roughly the power of OWL, perhaps a bit more.

Yet while the research contribution is real, vendors are unlikely to flock to it. Despite its technical merits, this logic is supported only by its authors. Even if one looks at the underlying technology, Datalog, there is no standard software interface that would enable tool interoperability. More ambitiously, one wants a rich variety of services for reasoning, explanation, and exchanging data with outside. To obtain these, access control should plan to adopt whatever standard becomes widely used in the knowledge management community.

Clustering around a standard can even improve research quality. When each paper defines its own formalism, we reduce the opportunities for synergy -- among access control researchers, among our prototypes, with researchers in other subdisciplines (e.g., heterogeneous databases), and with industry. Also, with a self-defined model one may completely ignore many necessary features (e.g., metadata and views in SQL security). If instead one starts with a fuller model and explicitly defers some features, one will probably avoid unnecessary incompatibility, and junior researchers will see the gaps and fill them.

From the other side, semantic web researchers [IJITM, KF+, QA, AS] are interested in using OWL in access controls. Many of the semantic web researchers' insights apply to our problem, but there may be difficulties. First, some models seem complex; e.g., the override rules in [QA] look plausible but lack theoretical justification and will be difficult for administrators to apply. More fundamentally, they aim at data that is already in OWL, which includes little of today's data.

Our formulation is subtly different. We define access controls' target as all enterprise data (e.g., files and relational and XML objects), but do not require that the target data be modeled semantically in OWL. Semantically-aware access controls need enriched semantic descriptions (e.g., in OWL) only for structures that policies will exploit. Roughly speaking, this is structure that would be expressed in a role hierarchy. No OWL wrapping is needed for target data, even if that data is referenced in simple predicates. Creation of OWL models can proceed incrementally, only when applications (including access policies) need it.

To summarize, we believe security research would have more influence if, instead of exploiting obscure formalisms, it built on a more popular technology that gives an 80% solution. The researchers could then identify requirements and techniques for the extensions that security needs.

4. SUMMARY

As a researcher from a database rather than security background, we have observed that concern with *administrative* scalability have not been principal drivers in the security community. Many proposals seem, unnecessarily, to omit simplicity, reaching out, and modularity. We have described several important aspects of scalability, and given our impressions of which currents in data security research seem to promote it.

REFERENCES

[AC] V. Atluri, S. Chun, An Access Control Model for Geo-spatial Data, *IEEE Transactions on Dependable and Secure Systems*, October-December, 2004

[AS] S. Agrawal, B. Sprick Access Control for Semantic Web Services, *IEEE ICSW*, 2004

[BJ+] R. Bhatti, J. Joshi, E. Bertino, A. Ghafoor, X-GTRBAC Admin: A Decentralized Administration Model for Enterprise Wide Access Control, *ACM SACMAT Conference*, Yorktown Heights, 2004

[BS] E. Bertino, R. Sandhu, Database Security—Concepts, Approaches, and Challenges, *IEEE Transactions On Dependable And Secure Computing*, January-March 2005

[CCF] S. Castano, S. De Capitani di Vimercati, M.G. Fugini, Automated Derivation of Global Authorizations for Database Federations, *Journal of Computer Security*, 1997

[FKC] D Ferraiolo, R. Kuhn, R. Chandramouli, *Role Based Access Control,* Artech House, 2004

[GO] E Gudes and M. Olivier, Security Policies in Replicated and Autonomous Databases, IFIP11.3 Database Security 1998

[IJITM] Call For papers Special Issue on: Access Control and Inference Control for the Semantic Web, International Journal of Information Technology and Management (IJITM) http://www.inderscience.com/browse/callpaper.php?callID=189

[KF+] L. Kagal, T. Finin, M. Paolucci, N. Srinivasan, and K. Sycara, G. Denker, Authorization and Privacy for Semantic Web Services, *IEEE Intelligent Systems*, 2004

[KK+] A. Kern, M. Kuhlmann, R. Kuropka, A. Ruthert, A Meta Model for Authorisations in Application Security Systems and their Integration into RBAC Administration, *ACM SACMAT Conf.* Yorktown Heights, NY 2004.

[LMW] N. Li, J. Mitchell, W. Winsborough: Design of a Role-Based Trust-Management Framework. *IEEE Symposium on Security and Privacy 2002*

[Lo] D. Lomet, *A Role for Research in the Database Industry, ACM Computing Surveys 28(4es), Dec. 1996,*

[NIST] National Institute of Science and Technology (website) Role Based Access Control http://csrc.nist.gov/rbac/

[Oasis] Oasis Consortium, eXtensible Access Control Markup Language (XACML) and Security Application Markup Language (SAML), http://www.oasis-open.org/

[PM] T. Prickett-Morgan, Gartner Says Database Market Continued Its Recovery in 2004, UNIX Guardian, June 9, 2005

[QA] L. Qin, V. Atluri, Concept-level Access Control for the Semantic Web, *ACM Workshop on XML Security*, 2003

[RW] A. Rosenthal, M. Winslett Security of Shared Data in Large Systems: State of the Art and Research Directions, Tutorial, *ACM SIGMOD Conf*, 2004, and *VLDB*. 2004

[TL] M. Tripunitara, N. Li, Comparing the power of access control models, *ACM conference on Computer and communications security*, 2004

[WWJ] L. Wang, D. Wijesekera, S. Jajodia, A Logic-based Framework for Attribute based Access Control, *ACM FMSE* 2004

ACKNOWLEDGEMENTS

This work was supported by the technology program at the MITRE Corporation. The author benefited greatly from Frank Manola’s insights about the semantic web.