Fast Semantic Attribute-Role-Based Access Control (ARBAC) in a Collaborative Environment

Leo Obrst[a], Dru McCandless[b], David Ferrell[a]
The MITRE Corporation
[a]McLean, VA
[b]Colorado Springs, CO
{lobrst, mccandless, ferrell}@mitre.org

***Abstract*—This paper is an early report of our continuing effort to provide a platform-independent framework so that information originators and security administrators can specify access rights to information consistently and completely, and that this specification is then rigorously enforced. To accomplish this objective it is necessary to link a security policy model to a policy language with sufficient expressive power to ensure logical consistency. For the purposes of this research we are using a modified Attribute-Role-Based Access Control (ARBAC) security model and the Web Ontology Language (OWL) with additional rules in a logic programming framework to express access policy, going beyond the limitations of previous attempts in this vein. In addition we are developing a mechanism using knowledge compilation techniques that allows access policy constraint checking to be implemented in real-time, via a bit-vector encoding that can be used for rapid run-time reasoning.**

***Index Terms*—access control policy, attribute-based, role-based, Semantic Web, logic programming, knowledge compilation, social network, ontology, rule-based reasoning**

I. Introduction

This paper is an early report of our continuing effort to provide a platform-independent framework so that information originators and security administrators can specify access rights to information consistently and completely, and that this specification can then be rigorously enforced.

Information sharing cannot be totally open for numerous sensitivity, privacy, and proprietary concerns. In an open and dynamic environment , it is difficult to specify and enforce a coherent and consistent security policy that spans multiple organizations, diverse computing environments and data systems, multiple dissemination platforms, and potentially thousands of people with complex and overlapping roles and responsibilities, and multiple group memberships.

Furthermore, it is difficult to enforce access rights consistently and coherently: the originator must be able to state the access rights clearly and unambiguously; a security administrator must implement the correct matching rules for a given computing environment; and the system must enforce the access rules correctly and efficiently.

Access control of information protecting privacy, security, and also enabling a complex range of policy respecting those requirements, in a collaborative social environment is difficult.

To accomplish these objectives it is necessary to link a security policy model to a policy language with sufficient expressive power to ensure logical consistency. For the purposes of this research we are using a modified Attribute-Role-Based Access Control (ARBAC) security model and the Web Ontology Language (OWL) with additional rules in a logic programming framework to express access policy, going beyond the limitations of previous attempts in this vein. In addition we are developing a mechanism using knowledge compilation techniques that allows access policy constraint checking to be implemented in real-time, via a bit-vector encoding that can be used for rapid run-time reasoning.

We have focused on three aspects: expressivity, adaptability, and efficiency. To this point, we have developed an architecture and concept of operations document, an overall design, a semantic policy model expressed in OWL and its instantiation, and are currently working to transform the policy model instance into a logic programming execution environment that includes rules. The latter itself is embedded in a Java program that interfaces with external services, e.g., obtaining identity and access tokens (and their specific attribute information) from the authentication service, and passing that information to the underlying logic programming execution. We expect soon to address optimization issues and integration of the access policy enforcer into the larger emerging collaborative environment and its processes and services. We intend to develop a mechanism that allows access policy constraint checking to be implemented in real-time via a bit-vector encoding that can be used for rapid run-time reasoning.

Our access control regime is one component of a larger effort (dubbed the MITRE Partnership Network (MPN)) to develop a large-scale collaborative environment for group-based (social network) information sharing among disparate governmental, commercial, academic, and other communities.

II. Proposed System Architecture

The proposed system architecture of the semantic ARBAC system is represented in Figure 1. It consists of three processes which flow from left to right. The three processes are: 1) the *Development* time process; 2) the *Transformation* time process; and 3) the *Execution* (runtime) process.

The *Development* time process involves: 1) the creation (or update) of the ARBAC policy ontology, represented in OWL and RDF, i.e., the semantic policy model SPM; and 2) the instantiation of the specific ARBAC policy (policies) to be transformed and deployed, i.e., the semantic policy instance (SPI).

COLLABORATECOM 2012, October 14-17, Pittsburgh, United States

DOI 10.4108/icst.collaboratecom.2012.250750

The *Transformation* time process involves developing and/or generating in Prolog and Java: 1) the transformer interpreter that will take the SPI and generate the runtime semantic policy instance (RSPI), which is the bit-vector representation of the policy + rules; the attribute signature assignment engine (ASAE) which generates and updates the resource access registry (RAR); and the RAR, which captures the attributes of the resources in bit-vector representation, indexed by resource URI; 2) the runtime user access routine (RUAR); 3) the runtime inference engine (RTIE) which will execute the RSPI using the RUAR. The Transformation time process can thus be considered a knowledge compilation process, where source semantic models and their interpreting engines get transformed to efficient Execution time process objects.

The *Execution* time process thus includes the RAR, ASAE, RTIE, and the RUAR, in addition to access to the Development and Transformation models and data.

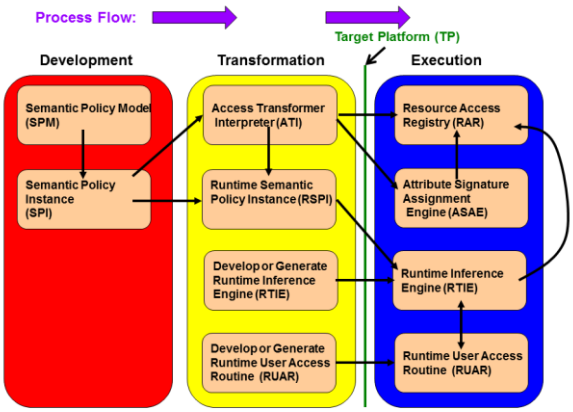

Fig. 1. Proposed Fast Semantic ARBAC System Architecture Proposed Fast Semantic ARBAC System Architecture

III. Fast Semantic ARBAC Runtime System Components

This section briefly describes the proposed runtime system components of the Fast Semantic ARBAC system, as depicted in Figure 2. The runtime system components view represents most components of the system architecture modules displayed in Figure 1, but focuses on their relationships at runtime only.

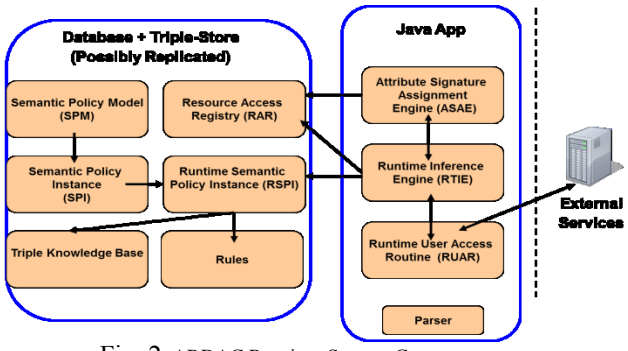

Fig. 2. ARBAC Runtime System Components

A. Semantic Policy Module (SPM)

The SPM consists of the OWL ontology classes, object properties, and data properties. The major classes will consist of: Subject (the person, organization, software that requests specific access to a resource), Action (the kind of access requested, e.g., read, write, create, delete, execute, etc.), Resource (the object needing to be accessed by a subject: executable, graphic, text, sound, video, hardware, etc.), Environment (salient aspects of the space or session's environment, e.g., risk or alert level, initial access vector (e.g., entry network domain), Role (traditional roles such as administrator, expert, end user, developer, etc., that are also related to groups), and related notions: Authentication, Security, ClassificationLevel, Identity, Time, etc.

In addition, rules are a very important component of the SPM. Rules exist outside of the OWL ontology per se, but are based on the classes and properties specified in the ontology. Rules are expressed in a logic programming language such as Prolog. Rules are potentially recursive and express logical constraints among and across class and property values (instances). Some examples are given below.

The SPM represents a set of generic semantic components for ARBAC policy, and thus constitutes a family of potential specific ARBAC instantiations.

B. Semantic Policy Instance (SPI)

The SPI is a specific instantiation of the SPM. It consists of those classes, object properties, and data properties deemed needed for a specific ARBAC policy, populated with instances of those, e.g., specific Resources (specific executables, files, hardware, etc.), specific Subjects (specific persons, organizations, registered consumer executables, etc.), specific Groups (specific communities of interests + their inter-linkings or hierarchies, user-defined friendships groups, topic subscribers, etc.), and specific Environment objects and constraints (specific authenticity or identity standards, network entry points, etc.), and specific rules to be in effect.

Note that there could be multiple SPIs created, e.g., one for normal-risk/normal-alert global situations, one for high-risk/high-alert global situations, etc. In addition, it is probable that the high-level ARBAC controller (the Java application that controls the runtime components in Figure 2, above) could detect or be instructed to change to a high-risk/high-alert situation and swap the SPI in effect, by switching to a different runtime semantic policy instance (RSPI).

C. Runtime Semantic Policy Instance (RSPI)

The RSPI is the bit-vector-based optimized runtime model generated from the SPI. Because there are potentially multiple SPIs, there are respectively multiple RSPI. Each RSPI is generated by the access transformer interpreter (ATI), consisting of Prolog and Java code, into a bit-vector representation of the specific portions (classes, properties, values) of the OWL and RDF model, along with the specific instances and rules, that constitutes the SPI.

The RSPI is the optimized model executed by the Runtime Inference Engine (RTIE), the latter of which also uses the Resource Access Registry (RAR), which in turn has bit-vector representation (instances of resources and their access attributes). An important point is: the RSPI, RTIE, and the RAR all use the *same, consistent bit-vector-based representation*, because they were all generated or transformed by the same optimization scheme in the Transformation time process.

D. Resource Access Registry (RAR)

The RAR is the runtime data structure for resources, with individual resources indexed by URI/IRI and each resource having a bit-representation of its access attributes. Conceptually (notionally), the RAR is depicted in Figure 3. The primary elements are the URI (which can be considered a unique identifier, and possibly also the address of the resource), a short description in English of the resource, and the bit-vector encoding of access attributes. The latter are simply notionally given at this point, since the actual encoding scheme is not yet worked out. It is possible that the bit-encoding could be segmented, with distinct segments for class, group, environmental, and risk attributes, etc.

Still to be determined are issues such as how the RAR is implemented, e.g., whether a hash scheme for the identifier is used, and the separation of human-usable and machine-usable elements.

Resource URI	Resource Access Attribute Bit-Vector	Resource Description
http://www.mitre.org/resource/printer/MWPS4/3H114-HP#	10110 00111 01100 00000 00001	HP Universal Printing PS (v5.2) Printer in 3H114
http://www.mitre.org/resource/jpsmith/document/ARBAC-14#	10010 10110 11010 01000 01101 00000 00001	Jane P Smith document ARBAC-14 conops, 01/02/12
TBD	TBD	TBD
TBD	TBD	TBD

Fig. 3. Resource Access Registry (RAR)

E. Attribute Signature Assignment Engine (ASAE)

The ASAE is the module that assigns the attributes to the resources and is responsible for maintaining the bit-representation of the attributes. It must do so in a fashion consistent with the runtime inference engine (RTIE), the RAR, and typically is at least partially generated by the Transformation time processes that provide the consistent bit-encoding of all runtime bit-representations.

F. Runtime Inference Engine (RIE)

The RTIE is the main knowledge component of the ARBAC system: it enforces the prospectively multiple, bit-represented RSPIs, which are the actual runtime policy instances in effect, provides efficient rule-based reasoning over the bit-represented RAR, and interacts with the RUAR, by which users interact with the runtime policy instances, resources, and the inference engine. Typically, the RTIE will also interact with the ASAE to update the RAR. The RTIE may invoke the SPI directly and other components.

G. Runtime User Access Routine (RUAR)

The RUAR is the primary component responsible for interaction at runtime with the user. It is invoked by the runtime controller, which is the encapsulating java application that also controls the external service interface. The RUAR is the internal API to the ARBAC.

H. Parser

The parser consists of the parser of OWL/RDF and related formats, the set of translation routines (XSLT or otherwise, from OWL/RDF to logic programming, XML, and service syntaxes such as SOAP, WSDL, etc.)

I. Encapsulating Java Application and External Service Interface

The encapsulating Java application (with external service interface) controls the external interaction with the ARBAC system, providing an API for external services. This component also invokes the parser and related services required for transformation time and execution time processes.

IV. Access Decision Process flow

The following depicts the access decision process flow involved.

- Initially, the Policy/Rules KB is read and loaded (including any general rules that apply to all circumstances) by the inference engine.
- Then a request comes in containing the Subject, Resource, Action, and Environment.
- The Subject's Group membership is looked up and formed.
- An initial Resource/Group/Access may be performed.
- For some common accesses these may be cached, or may require no further processing if a quick decision can be made.
- Otherwise, the appropriate rule set is generated and populated with: any referenced access rule (pre-filtered to keep the KB small and fast), all facts about the Subject, Resource, Groups, and Environment, and General (generally applicable) rules.
- The rule set is passed to a runtime inference engine which evaluates the truth of the permission statement (something along the lines of allow(Subject, Access, Resource)).
- The Inference Engine passes back the permission decision.

V. Walkthrough

The Semantic Policy Model (SPM) is the holder of much of the underlying knowledge. Its contents include:

- Data Model and Ontology
- Access Rules
- Group Membership Rules
- General Rules

The Access Rules ultimately determine whether an action can be performed on a resource – we will retain the use of the term 'Privilege' to denote the pairing of actions and resources; each rule can be thought of as consisting of 3 parts:

1. The head, or consequence, which is always a privilege (e.g., hasPrivilege(subject22, read,medicalRecord66)). This leaves the body of the rule which for convenience is broken into 2 parts:
2. The Group membership required to obtain the privilege, and
3. Any additional requirements, expressed in terms of environment variables.

Example:

hasPrivilege(Subject, Action, Resource)

← agent(Subject), member(Subject, Group), environmentalConstraints(Group, Action, Resource, Environment), groupWithPrivilege(Group, Action, Resource, Environment).

Premises:

- All access decisions can be expressed as a *privilege* ← *requirements* rule.
- All role or subject attribute requirements for access can ultimately be expressed as group membership.
- Group membership is both dynamic and contextual.
- Resources and their attributes are known a priori (otherwise, we couldn't reason about them except in the most general sense); therefore it is possible to extensionalize much of the access constraints about them.
    - Definition of *Extensionalize*: bind all rule variables to concrete values (instances) in advance.
    - Note that if resources and attributes can change arbitrarily dynamically, then we will have to factor this in, and it will contribute to lesser performance.
    - In general, any variable at run-time (i.e., evaluated dynamically) will decrease performance.

Knowledge of four things is used to resolve a permission question:

1. The Subject (the entity requesting the permission)
2. The Resource that the Subject is requesting permission about
3. The Action that the Subject wishes to perform
4. The Environment, which is a set of facts/assertions that the rules may take into account in order to make a permission determination.

The result will be either a yes or no answer as to whether permission is granted (i.e., the privilege is allowed to the Subject).

The overall process for determining whether to grant the permission is as follows:

- The Subject's Group Membership is determined and generated (possibly using knowledge about the Resource and the Environment).
- All of the Access Rules applicable to the privilege are identified.
- The Group Membership and Environment assertions are combined with the Access Rules and passed to the Inference Engine, which is then queried for the privilege.
- If the query is satisfied, then permission is granted.

Because group membership is both dynamic and independent of other decisions in this process, it is possible to isolate it as a separate function. By doing this we can build a more efficient and scalable ARBAC system.

The access rules can have fairly complicated group membership conditions (e.g., a doctor who is an associate of a patient's primary care physician can have read access to that patient's medical record). Therefore, determining group membership may rely on a number of General Rules to help resolve the inferences (e.g., a doctor may be a member of a group; if another doctor is also a member of that group, then that doctor is an associate of the first doctor, etc.). By making group membership dynamic we can keep the access rules general (i.e., the extensionalized access rule would only be fired if the dynamic assertion of group membership was generated by the Policy Instance and passed to the Inference Engine – which would only happen if the Subject were found to be an associate doctor based on the rules). As will be seen, this allows Subject/Resource relations to be extensionalized, but prevents having to extensionalize every possible combination of Subject and Privilege.

A. Example

To illustrate all of this, suppose that Dr. Nick Riviera (an associate of Dr. Hibbert) wants to read Homer Simpson's medical record (id: medicalRecord66); Homer's primary care physician is Dr. Hibbert. The request would come in to the RUAR. The RUAR would pass the Resource ID to the RTIE that would access the RAR to obtain all of the attributes (the attribute signature), which would then, along with the Subject's ID (for Dr. Nick, his ID is person002) be returned to the RTIE which would compute all of the Subject's group memberships based on the RSPI. The RTIE would infer that Dr. Nick is an associate of Dr. Hibbert, and then invoke the RAR to check to see which access rules apply to the resource and the requested action, and pass them back to the RTIE. The RTIE would (if required) modify the RSPI to update the runtime rules containing the requested privilege (read,medicalRecord66), and the Subject's group membership, plus any environment assertions, and return them to the RTIE. It would then query the RTIE whether the privilege (read,medicalRecord66) was satisfiable by the runtime engine, and upon receiving a positive answer, would grant permission.

In more detail, the SPM contains the following access rules:

Rule 1: A doctor has Write access to a medical record if they are the patient's primary care physician

Rule 2: A doctor who is an associate of a patient's primary care physician has Read access to that patient's medical record

Rule 3: A person may access any printer in the Springfield Medical Center if they are a member of the medical staff, and they are accessing the printer from inside the center (i.e., no external print jobs allowed).

It also contains the following facts:

TABLE I. SEMANTIC POLICY MODEL FACTS

person001	hasName	Homer Simpson
person002	hasName	Julius Hibbert
person003	hasName	Nick Riviera
medicalRecord66	type	Medical record
printer23	type	Printer
printer23	location	facility21
facility21	hasName	Springfield Medical Center
medicalRecord66	about	person001
medicalRecord66	primaryCarePhysician	person002
person002	occupation	Physician
person002	hasAssociate	person003
person002	employedBy	facility21
person003	employedBy	facility21

The SPM also contains the following inference rules, which associate a Subject with a Resource (for the purpose of Group Membership):

isPrimaryPhysician(X,Y) ← type(X,Medical record) & primaryCarePhysician(X,Y).

hasAssociatePhysicianGroupMember(X,Y) ← type(X,Medical record) & occupation(Y,Physician) & primaryCarePhysician(X,Z) & hasAssociate(Z,Y).

isMedicalStaffGroupMember(X,Y) ← type(X,Printer) & location(X,Z) & employedBy(Y,Z).

Using these General Rules many of the Subject/Resource relations can be extensionalized and stored in a bit vector. Each predicate, whether from a relational property in the SPI or the head of a rule, is captured in a table that assigns a bit position to it:

TABLE II. ASSIGNMENT OF BIT REPRESENTATION

predicateID	**predicate**
1	hasName
2	type
3	location
4	about
5	primaryCarePhysician
6	occupation
7	hasAssociate
8	employedBy
9	hasAssociatePhysician
10	isMedicalStaffGroupMember

The predicateID corresponds to the bit position in the next table, which captures (some) of the Subject/Resource Relations:

TABLE III. PREDICATE-ID BIT POSITION OF SUBJECT/RESOURCE RELATIONS

Resource	Subject	Vector
medicalRecord66	person001	0001000000
medicalRecord66	person002	0000100000
medicalRecord66	person003	0000000010
printer23	Facility21	0010000000

So the third table row contains the fact that person003 (Dr. Nick Riviera) and medicalRecord66 (Homer's medical record) are in fact associated by predicate 9 (hasAssociatePhysician). This relationship would be passed to the RTIE.

The RAR has the following bit vectors mapping Resources to Access Rules:

medicalRecord66 1 1 0
printer23 0 0 1

This means that privileges associated with the resource 'medicalRecord66' can be obtained (but aren't necessarily guaranteed to be granted) by calling Access Rules 1 and 2. Similarly, privileges associated with the resource 'printer23' can be obtained by calling Access Rule 3. These bit vectors are built a-priori by the system – the information needed to build them is contained in the model (e.g., printer23 is of type Printer, and Access Rule 3 only applies to resources of type Printer, etc.).

The RAR also has the following bit vectors mapping Actions to Access Rules:

Read 1 1 0
Write 1 0 0
Print 0 0 1

This means that the Read privilege is granted by rules 1 and 2 (Write subsumes Read, so Read is automatically granted if Write is granted – this would be a General Rule), Write is granted by rule 1, and Print is granted by rule 3.

The complete set of associations between privileges and rules can then be built by comparing each row vector for each Resource and Action, and for each Rule (position) where both values are 1, then that corresponding rule would get passed to the RTIE. So if the requested privilege were (Read, medicalRecord66), the RTIE would get passed rules 1 and 2. If the requested privilege were (Write, medicalRecord66) the RTIE would get passed rule 1. If the requested privilege were (Print, printer23) the RTIE would get passed rule 3. All other privilege combinations would result in an empty set of access rules (meaning it would be impossible to obtain permission for the requested privilege).

The RTIE already contains the relation hasAssociatePhysicianGroupMember(medicalRecord66,person003) based on the inference from the General Rules described above. by calling the rules to compute Group Membership for the Subject. The Therefore, the RTIE has assembled the following knowledge base:

Privilege(Read,medicalRecord66) ← isPrimaryPhysician(medicalRecord66,Y).
Privilege(Read,medicalRecord66) ← hasAssociatePhysicianGroupMember(medicalRecord66,Y).
hasAssociatePhysicianGroupMember(medicalRecord66,person003).

The knowledge base would be passed to the runtime portion of the RTIE, which would then be queried whether the assertion Privilege(Read,medicalRecord66) was satisfiable. Since it is, the permission for Dr. Riviera to read Homer's medical file would be granted.

Since all of the consequents in the knowledge base are ground (no variables), the resolution of the query should be extremely fast. Also, the privileges themselves should be formed very quickly since they are the product of comparing bit matrices. Group membership is the potential bottleneck, since a number of inferences (and potential calls to outside sources) must be made before all of the group determinations can be made.

Extending the example, suppose Dr. Nick wanted to now print something on printer23, but he is working from his office across town (and not in the Springfield Medical Facility). The requested privilege would be (Print,printer23), so the RAR would only pass rule 3 to the RSPI. Dr. Nick's group membership for printer23 would be computed as isMedicalStaffGroupMember(printer23,person003) from the inference rules, but the RTIE would get passed the following knowledge base:

Privilege(Print,printer23) ← isMedicalStaffGroupMember(printer23,X) & accessType(X,local).

isMedicalStaffGroupMember(printer23,person003).
accessType(person23,remote).

The query whether Privilege(Print,printer23) is satisfiable would return false, since the accessType is remote rather than local. Here the predicate 'accessType' represents an environment condition, which must be preserved in the Access Rule directly. Its value would be computed by the ASAE, and passed (along with the other model information) to the RTIE.

VI. Related Work

There is much previous related research across multiple dimensions (access control regimes, policy languages and approaches, specialized languages (and logics) vs. ontology approaches, knowledge compilation issues, bit-vector and other optimization approaches, social network approaches, privacy vs. security issues and approaches, etc.) that have influenced our current and impending work.

In order to accomplish our objectives it was necessary to link a security policy model to a policy language with sufficient expressive power to ensure logical consistency. We extend the NIST Role-Based Access Control (RBAC) security model [22] and related approaches [25-26], as have many other researchers to include attributes, and extend the Web Ontology Language (OWL) with additional rules to express access policy using logic programming, and beyond the limitations of [29]. Unfortunately, given our own space limitations here, we cannot do an extensive comparison of our approach across the multiples dimensions with other approaches, nor justly describe those other approaches.

In addition, there is extensive research in more general policy-based approaches that could be employed also for access control [6-7, 28-29].

There are other Semantic Web-based approaches (including [29]), some of which address more specifically social network types of applications [30-42].

For implementation in real-time, via a bit-vector or other efficient encodings that can be used for rapid run-time reasoning, we've looked at [1-5, 8-12, 24]. Much of the research for efficient taxonomic and type encodings that can be applied to ontologies and RDF graph structures go back many years, to the mid-19980s at the minimum. For bit-vector representation to support RDF triples, we investigated [14-20].

Our own previous work addressed issues in translating OWL/RDF ontologies and Semantic Web Rule Language Rules (SWRL) [43], now generally superseded by the Rule Interchange Formal (RIF) standard [44], into logic programming for efficient runtime reasoning, and employing knowledge compilation techniques [45-49], which we also generalized to address services using first-order logic theorem provers and for ontology alignment [50-51].

VII. Impending Work

Our effort shortly will be to address the following tasks.

A. *Optimization of the Reasoning Engine*

Although we will shortly have some optimizations done – e.g., extensionalization (solving rules in advance as much as possible and tabling the results, which is much like relational database tabling done by deductive database engines), and delayed rule evaluation (partially instantiating rules, then delaying their further instantiation until the requisite information is received) – we will not have made substantial implementation of the second-level of optimization we intend, i.e., the bit-representation execution at runtime.

B. *Dynamically Swapping Out Policy Models*

We need the ability to swap out policy models, their instantiations, and the reasoning engines that enforce those policies on the fly based on changing environmental constraints. The paradigmatic environmental situation will be if there is a global change of threat level, e.g., we go to high-threat from normal-threat. In a case like this, the policy must be able to change on the fly, based on communication of the increased threat level to the engine from elsewhere. This will allow a spectrum of more conservative policies to be enforced, enabling limited, defined access by specific groups or individuals, rather than just throttling shut access to the collaborative space (which could always be done if needed at a higher level).

C. *Integration*

We need to integrate the fast semantic ARBAC component with the other components of the emerging MITRE Partnership Network (MPN) portal. Currently we are working towards integration by building in expected quasi-service/data access to other MPN components, primarily authentication, but real integration must still be done. In addition, we desire to explore closer collaboration with non-MITRE, external researchers, implementers, and standards groups, who are working in the access control policy area.

ACKNOWLEDGMENT

REFERENCES

[1] Abadi, Daniel J. , Adam Marcus, Samuel Madden, Katherine J. Hollenbach. 2007. Scalable Semantic Web Data Management Using Vertical Partitioning. In Proceedings of VLDB, pages 411~422, September 2007.

[2] Ait-Kaci, Hassan. 1984. A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially-Ordered Type Structures. Ph.D thesis, Computer and Information Science Dept., Univ. of Pennsylvania, Philadelphia, PA.

[3] Ait-Kaci, Hassan; Robert Boyer; Patrick Lincoln; Roger Nasr. 1989. Efficient Implementation of Lattice Operations. TOPLAS 11-1-1989. http://www.csl.sri.com/users/lincoln/papers/toplas89.ps.gz.

[4] Blandford, D. K., Blelloch, G. E., and Kash, I. A. 2003. Compact representations of separable graphs. In Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, Maryland, January 12 - 14, 2003). Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, 679-688. http://portal.acm.org/citation.cfm?id=644219#.

[5] Blandford, D. K., Blelloch, G. E., and Kash, I. A. 2004. An Experimental Analysis of a Compact Graph Representation. In Proceedings of ALENEX04. http://people.seas.harvard.edu/~kash/papers/BBK04.pdf.

[6] Bradshaw, J. M., Beautement, P., Breedy, M. R., Bunch, L., Drakunov, S. V., Feltovich, P., Hoffman, R. R., Jeffers, R., Johnson, M., Kulkarni, S., Raj, A. K., Suri, N., & Uszok, A. (2003). Making agents acceptable to people. In N. Zhong & J. Liu (Ed.), Handbook of Intelligent Information Technology. IOS Press.

[7] Bradshaw, J. M., Dutfield, S., Benoit, P., & Woolley, J. D. (1997). KAoS: Toward an industrial-strength generic agent architecture. In J. M. Bradshaw (Ed.), Software Agents. (pp. 375-418). Cambridge, MA: AAAI Press/The MIT Press.

[8] Caseau, Y. ; M. Habib; L. Nourine; O. Raynaud. 1999. Encoding of multiple inheritance hierarchies and partial orders. Computational Intelligence 15 (1) (1999) 50-62.

[9] Dershowitz, Nachum. 2008. Bit Inference. Workshop on Practical Aspects of Automated Reasoning, August, 2008, Sydney, Australia, pp. 26-35. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.143.341.

[10] Fall, A. 1995. Heterogeneous Encoding. In Proceedings of International KRUSE Symposium: Knowledge Retrieval, Use, and Storage for Efficiency, Gerard Ellis, Robert Levinson, Andrew Fall, Veronica Dahl, eds., Santa Cruz, CA, Aug. 11-13, pp. 134-146 (1995).

[11] Krall, A., Vitek, J., Horspool, N.1997. Near optimal hierarchical encoding of types. 11th European Conference on Object Oriented Programming (ECOOP'97). Springer (1997).

[12] Li, N., B. N. Grosof, and J. Feigenbaum. 2000. A practically implementable and tractable delegation logic. In Proc. of IEEE Symp. on Security and Privacy, Oakland, CA, USA, May 2000.

[13] Li, N., J. Mitchell, and W. Winsborough. 2002. Design of a role-based trust-management framework. Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium, pages 114–130.

[14] McGlothlin, James P., Latifur Khan, Bhavani Thuraisingham. 2011. RDFKB: A Semantic Web Knowledge Base. Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI) 2011.

[15] McGlothlin, James P., Latifur Khan. 2008. RDFVector: A Scalable Data Model for Efficient Querying of RDF Datasets. http:// www.utdallas.edu/~jpm083000/ssDBM.pdf.

[16] McGlothlin, James P.; Latifur Khan. 2009. RDFKB: efficient support for RDF inference queries and knowledge management. In Proceedings of IDEAS, pages 259-266, September 2009.

[17] McGlothlin, James P.; Latifur Khan. 2010b. Efficient RDF data management including provenance and uncertainty. In Proceedings of IDEAS, pages 193-198, August 2010.

[18] McGlothlin, James P.; Latifur Khan. 2010c. A Semantic Web Repository for Managing and Querying Aligned Knowledge. In Proceedings of ISWC, November 2010.

[19] McGlothlin, James P.; Latifur R. Khan. 2010a. Materializing Inferred and Uncertain Knowledge in RDF Datasets. In Proceedings of AAAI, pages 1405-1412, July 2010.

[20] McGlothlin, James. 2010. RDFVector: An Efficient and Scalable Schema for Semantic Web Knowledge Bases. PhD Symposium, 7th Extended Semantic Web Conference (ESWC 2010), Heraklion, Greece. May 30 – June 3, 2010. http:// www.utdallas.edu/~jpm083000/eswc.pdf.

[21] Moses, T. et al. 2005. eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, 200502. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[22] http://csrc.nist.gov/groups/SNS/rbac/.

[23] Neumann, Thomas, Gerhard Weikum. 2009. RDF-3X: a RISC-style engine for RDF. In Proc. of VLDB, pages 647-659, September 2009.

[24] Preuveneers, D., Berbers, Y., 2006. Prime numbers considered useful: Ontology encoding for efficient subsumption testing, Tech. Rep. CW464. http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW464. Department of Computer Science, Katholieke Universiteit Leuven, Belgium (October 2006).

[25] Sandhu, R. 1998. Role-based access control. In M. Zerkowitz, editor, Advances in Computers, volume 48. Academic Press.

[26] Sandhu, R., E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. 1996. IEEE Computer, 29(2):38–47, February 1996.

[27] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham. 2008. ROWLBAC: representing role based access control in OWL. In Proceedings of the 13th ACM symposium on Access control models and technologies (SACMAT '08). ACM, New York, NY, USA, 73-82. DOI=10.1145/1377836.1377849 http://doi.acm.org/10.1145/1377836.1377849.

[28] Tonti, G., J. M. Bradshaw, R. Jeffers, R. Montanar, N. Suri, and A. Uszok. 2003. Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder. In Proceedings of the 2nd International Semantic Web Conference (ISWC2003). Springer-Verlag.

[29] Uszok, Andrzej, Jeffrey M. Bradshaw, James Lott, Maggie Breedy, Larry Bunch, Paul Feltovich, Matthew Johnson, Hyuckchul Jung. 2008. New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS, Policies for Distributed Systems and Networks, IEEE International Workshop on, pp. 145-152, 2008 IEEE Workshop on Policies for Distributed Systems and Networks, 2008.

[30] H. C. Choi, "D-FOAF: Distributed identity management with access rights delegation," in Proc. 1st Asian Semantic Web Conference. Springer, 2006, pp. 140–154.

[31] B. Carminati, E. Ferrari, and A. Perego, "Rule-based access control for social networks," in Proc. OTM 2006 Workshops, ser. LNCS, vol. 4278. Springer, Oct 2006, pp. 1734–1744.

[32] W. Villegas, B. Ali, and M. Maheswaran, "An access control scheme for protecting personal data," in Proc. 6th Annual Conference on Privacy, Security and Trust, 2008, pp. 24–35.

[33] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "A semantic web based framework for social network access control," in Proc. 14th ACM Symposium on Access Control Models and Technologies. ACM, 2009, pp. 177–186.

[34] T. Ryutov, T. Kichkaylo, and R. Neches, "Access control policies for semantic networks," July 2009, pp. 150–157.

[35] E. Barka and R. S. Sandhu, "Framework for role-based delegation models," in Proc. 16th Annual Computer Security Applications Conference. IEEE Computer Society, Dec 2000, pp. 168–176.

[36] M. Shehab, A. Squicciarini, and G.-J. Ahn, "Beyond user-to-user access control for online social networks," in ICICS '08: Proceedings of the 10th International Conference on Information and Communications Security. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 174–189.

[37] P. Reddivari, T. Finin, and A. Joshi, "Policy-based access control for an RDF store," in Workshop on Policy Management for the Web, 2005, pp. 78–81.

[38] S. Dietzold and S. Auer, "Access control on RDF triple stores from a semantic wiki perspective," in Scripting for the Semantic Web Workshop at 3rd European Semantic Web Conference (ESWC), June 2006.

[39] A. Dersingh, R. Liscano, A. Jost, J. Finnson, and R. Senthilnathan, "Utilizing semantic knowledge for access control in pervasive and ubiquitous systems," Mobile Networks and Applications.

[40] M. Liu, D. Xie, P. Li, X. Zhang, and C. Tang, "Semantic access control for web services," vol. 2, April 2009, pp. 55–58.

[41] A. Jain and C. Farkas, "Secure resource description framework: an access control model," in SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies. New York, NY, USA: ACM, 2006, pp. 121–129.

[42] Masoumzadeh, Amirreza; James Joshi. 2010. OSNAC: An Ontology-Based Access Control Model for Social Networking Systems. : Social Computing (SocialCom), 2010 IEEE Second International Conference on Social Computing, 20-22 Aug. 2010, Minneapolis, MN, pp. 751 – 759.

[43] Horrocks Ian. Patel-Schneider, Peter F. Boley, Harold. Tabet, Said. Grosof, Benjamin. Dean, Mike. 2004. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.w3.org/Submission/SWRL/ .

[44] Rule Interchange Format (RIF). http://www.w3.org/2005/rules/wiki/RIF_Working_Group.

[45] Samuel, Ken; Leo Obrst; Suzette Stoutenberg; Karen Fox; Paul Franklin; Adrian Johnson; Ken Laskey; Deborah Nichols; Steve Lopez; and Jason Peterson. 2008. A*pplying Prolog to Semantic Web Ontologies & Rules: Moving Toward Description Logic Programs.* The Journal of the Theory and Practice of Logic Programming (TPLP), Massimo Marchiori, ed., Cambridge University Press, Volume 8, Issue 03, May 2008, pp. 301-322.

[46] Ken Samuel; Leo Obrst. 2007. Answer Set Programming: Final Report on a Comparison Between ASP and Prolog for Semantic Web Ontology and Rule Reasoning. October, 2007. MITRE Technical Report MTR090069.

[47] Obrst, L; Stoutenburg, S; D. McCandless; D. Nichols; P. Franklin; M. Prausa; R. Sward. Chapter 5: Ontologies for Rapid Integration of Heterogeneous Data for Command, Control, & Intelligence. Chapter in: Obrst, Leo; Terry Janssen; Werner Ceusters, eds. 2010 Ontologies and Semantic Technologies for the Intelligence Community. Amsterdam, The Netherlands: IOS Press. IOS Press book series: Volume 213 Frontiers in Artificial Intelligence and Applications. August, 2010.

[48] Obrst, Leo; Dru McCandless; Suzette Stoutenburg; Karen Fox; Deborah Nichols; Mike Prausa; Rick Sward. 2007. Evolving Use of Distributed Semantics to Achieve Net-centricity. Regarding the "Intelligence" in Distributed Intelligent Systems, AAAI Fall Symposium, Arlington VA, Nov. 8-11, 2007.

[49] Stoutenburg, Suzette; Leo Obrst; Deborah Nichols; Paul Franklin; Ken Samuel; Michael Prausa. 2007. Ontologies and Rules for Rapid Enterprise Integration and Event Aggregation. Vocabularies, Ontologies and Rules for the Enterprise (VORTE 07), EDOC 2007, Annapolis, MD, Oct. 15-19, 2007.

[50] McCandless, Dru; Leo Obrst. 2009. Dynamic Web Service Chaining using OWL and a Theorem Prover. Third IEEE International Conference on Semantic Computing, Berkeley, CA, USA - September 14-16, 2009.

[51] McCandless, Dru; Leo Obrst. 2009. Using Ontology Alignment to Dynamically Chain Web Services. Ontology Matching Workshop, poster, International Semantic Web Conference (ISWC) 2009, Oct. 25-29, 2009, Chantilly, VA.