Policy Development Software for Security Policies

Anjali Shah, Lalana Kagal, Tim Finin, and Anupam Joshi
Department of Computer Science and Electrical Engineering, UMBC, Baltimore, MD 21250
{anjali1, lkagal1, finin, joshi}@cs.umbc.edu

Security policies define rules for access control, authentication, or authorization of entities in a system. With the increase in interest in web based e-commerce, the amount of business that is transacted on-line and the explosion in the amount of services available, the ability to handle security and privacy is a must. Also, as computationally enabled devices (laptops, phones, PDAs, and even household appliances) become more commonplace and short range wireless connectivity improves; there is an increased need for more automated security in the resulting pervasive environments. Policy-based security is often used in such environments to provide access control to resources from a large number of requesting entities that may be unknown to the former, provide security without necessarily authenticating requesters completely, provide flexibility in specifying security requirements and give every entity a certain amount of autonomy in making their own security decisions. Also, it makes possible to modify how different entities act without modifying their internal mechanism. We have put this idea of using policies to handle security and privacy into practice by using security policies expressed in a higher level policy language to provide a secure infrastructure for mobile devices [7]. We are also making use of policy based approaches for enhancing World Wide Web Consortium's Platform for Privacy Preferences (P3P) privacy architecture [8].

This growing importance of policy-based security underlines the need for usable interface for creating policies and providing support for policy debugging, policy validation and policy engineering. Motivated by this need, we are developing an Integrated Development Environment (IDE) for security policies by extending the plug-in architecture of IBM's Eclipse Platform [3]. The policies are specified in Rei [1], a policy language with general specifications for policies as well as mechanisms for policy verification. It includes few constructs based on deontic logic that allow policies to be described in terms of rights, obligations, dispensations, and prohibitions. It has a semantic interface for describing policies in semantic languages. The use of a semantic language enhances their interoperability and extensibility. Associated with the language is a policy engine that can be used within the application domain to interpret and reason over policies, help resolve in case of conflicts between policies, answer queries related to policy making and aid security and privacy governance by means of policy enforcement.

Editing facilities for Policy Creation

The IDE consists of a N3 Editor. N3 [2] is the XML syntax for RDF [5] and makes policies easier to understand and more readable. Policies written in N3 can be easily translated into semantic web languages like RDF-S and OWL [6]. The editor provides such features as Policy Templates, Content Assistance, Syntax Highlighting, and display of appropriate Context Information to facilitate policy creation. N3 uses the concept of namespaces. To make the process of including namespaces easier we have the concept of policy templates that include all the standard namespaces Rei policy developers would need. They can also add more namespaces to those existing in the templates and create new policy templates. In the editor window, based on the position of the cursor and the grammar of N3, content assistance is provided to the user. For example, on a blank position with no ':' before it, the user will get a list of namespaces to choose from. After selecting a namespace, prefix will be put in the cursor position with a ':' following it. At this point, based on whether the last delimiter was a ';' or a '.', the user is either shown a list of classes/instances in the chosen namespace (if '.') or a list of attributes of the object currently being described (if ';'). Also, based on the information in the namespaces about the classes/instances used, appropriate context information is displayed about that particular class/instance. For example, the properties of those classes, their domains and ranges etc. Apart from this text interface, we also plan to provide a graphical interface for creating simple policies through wizard extensions. This would give policy developers a choice between using a friendlier graphical interface for simple policies or the text interface for creating complex policies.

Policy Debugging and Validation

Once created, policies are parsed using the Jena API [4] to check for syntax errors and suggest corrections, if required. The IDE provides an interface to the Rei policy engine for developers to verify their policies and execute queries over them. We plan to offer provision for developing use-cases that include certain facts and queries. Policies along with the use-cases can be fed to the policy engine and answers to the use-cases can be used to verify if they are consistent with the intended purpose of the policies.

Policy Engineering

This is the support we plan to offer for consistency maintenance between different policies within a given domain or policies across domains. A policy domain could have several policies defined on the entities it contains. If creation of a policy or modification of an existing policy leaves any of the other policies in the domain inconsistent, then the user should either be warned or not allowed to do so. For example, a university domain may consist of some general policies that hold for all the departments within the university. In addition, each department may have policies defined specifically for the entities in that department. A user could make a change to an existing policy or create a policy that is inconsistent with the general policies. This also holds true for policies across domains. Consistency maintenance for policies across domains is required when several domain ontologies are merged together and policies are being created on entities within these domains.

References

[1] Kagal L., Finin T., Joshi A., A Policy Based Approach to Security for the Semantic Web, InProceedings, *2nd International Semantic Web Conference (ISWC2003)*, September 2003.
[2] Berners-Lee, T., Primer: Getting into RDF and Semantic Web using N3, http://www.w3.org/2000/10/swap/Primer (2003).
[3] Eclipse on line help [http://dev.eclipse.org/help21/index.jsp]
[4] McBride B., An Introduction to RDF and the Jena RDF API, http://jena.sourceforge.net/tutorial/RDF_API/
[5] Brickley, D. and R. Guha, Resource Description Framework (RDF) Schema Specification 1.0 - W3C Recommendation, http://www.w3.org/TR/2000/CR-rdfschema-20000327 (2000).
[6] Bechhofer, S., F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider and L. A. Stein, OWL Web Ontology Language, http://www.w3.org/TR/owl-ref (2003).
[7] Patwardhan A., Korolev V., Kagal L., Joshi A., Enforcing Policies in Pervasive Environments, TechReport, University of Maryland, Baltimore County, March 2004.
[8] Kolari P., Trust, Policy and the Semantic Web, Enabling Intelligent User Agents for Web Privacy, http://www.csee.umbc.edu/~kolari1/semanticp3p.html