The Virtuous Circle of Expressing Authorization Policies

David Chadwick[1], Angela Sasse[2]

[1] Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK
[2] Computer Science, University College London, Gower St, London, WC1E 6BT, UK

Abstract. This short paper reports on a current project to conduct a detailed investigation into non-security professionals' vocabulary and understanding of e-infrastructure and assets, with the longer term aim of building an ontology and controlled natural language interface that will allow them to build security policies, incorporating complex concepts such as delegation of authority, separation of duties (SoD), obligations and conditions. The interface is designed around the principle of the virtuous circle, whereby the user's controlled natural language input is converted into machine processable XML, and then converted back again into natural language, so that the user can compare the computer's understanding of his policy with his own. The user can then iteratively alter his policy until the input and output are semantically the same. To date, two GUI interfaces have been constructed that aid users in the construction of authorization policies, and produce natural language output. This will serve as a benchmark for measuring the ease of use and effectiveness of the controlled natural language interface. Work has started on the controlled natural language interface, and the first results are reported.

Keywords: Authorization, Policies, Controlled natural language, Virtuous Circle, XML.

1. Introduction

If web services and Grids are to become widely used, they need to be accessible to their target research communities, be secured well enough to be available as needed and function reliably. A key element in realising this ambition is that the owners and donors of web services need to retain control of their resources, and ensure their availability and integrity. To do this, resource owners need to express their policies for who can use their resources, and how. This is termed authorization. Saltzer and Schroeder define authorization as "grant(ing) a principal access to certain information" [1]. Several things are needed to ensure that the authorization policy that the owner intended to be enacted, is the policy that will finally be implemented by the resource's PDP (policy decision point). Firstly resource owners need to be able to state their security requirements correctly and efficiently. In the world of work, this is done through written security policies. In today's computer systems, this is typically done via command line or graphical user interfaces (GUIs) which use specialised security terminology. However, as [3] points out, many computer resources owners fail to even approach this task because they cannot translate their knowledge of resources and access into the computer security terminology used in the GUIs. Secondly, the policy's author needs to be assured that the policy recipients have received the policy and have interpreted it correctly, and behave as intended. In the world of work, policies are circulated to all employees and contractors of a business, in the expectation that they will read and obey them. In the case of computing resources, the policy generated by the interface is translated into a machine processable format, and transferred to the resource PDP for it to enforce. Thirdly, the policy owner needs to periodically check that the policy is indeed being enforced. In the world of work, this might be through periodic reports, audits or spot checks. For computer resources, the PDP will typically write its access control decisions to an audit log that can be periodically inspected by the IT staff. In this way, it is possible to belatedly check, after the fact, that the actual users who eventually gained access to the resource were exactly equal to those that the resource owner intended them to be. A fourth – and key – aspect in the policy specification and enactment process is that the resource owner did not make a mistake in specifying the policy in the first place. By "mistake" we mean that unintended consequences arise from enforcement of the policy (granting access to those who should not have it, or denying access to those who should). These mistakes are caused by misconceptions (in human error terminology) as opposed to errors in executing an intended policy incorrectly, e.g. through typing errors or confusing resource names ("slips" or "lapses" in human error terminology) [4]. When policies are written in controlled natural language, the scope for misconceptions is much reduced, and slips or lapses are more easily detected. Misconceptions can be due to the complexity of the policy, and the likelihood of specifying ambiguities or mutually exclusive clause. The audit log is currently the only (post-facto) way of determining if mistakes were made in the policy specification, as well as in its enforcement. Something better is needed, namely a pre-facto way of determining if the policy specification is correct before enforcement starts.

2. The Virtuous Circle

When specifying web services and grid authorization policies, ideally we want the policy tool to support the resource owner in the entire process of correct policy specification, and improve his/her understanding of access policies as a result of repeated interaction and feedback. In figure 1 below we show a 'virtuous circle' in which the computer system itself helps the user:

- to specify a correct policy,
- ensure that this is the policy that the user intended to specify, and then
- to confirm to the user that this is the policy that will finally be implemented by the PDP.

In figure 1, the user starts with a mental concept of the policy that he intends to enact, and the first step is to transcribe this into the written word in natural language. The language and vocabulary used to describe the policy are underpinned by an ontology that we are currently developing. The natural language policy is then parsed and processed by the computer and converted into a machine understandable policy, written in XML. We have chosen to use XML, since several policy decision points (PDPs) already exist that can read in XML policies and enforce them e.g. XACML [7] and PERMIS [6]. The XML is then processed with an XSL stylesheet, converted back into natural language and displayed to the user. The display not only shows the machine's understanding of the policy, but also is capable of printing out diagnostic error and warning messages to show the user where his policy is wrong, inconsistent or contains superfluous elements. This allows the user to compare the machine's understanding of his policy with his own, and also to correct the errors in his policy.

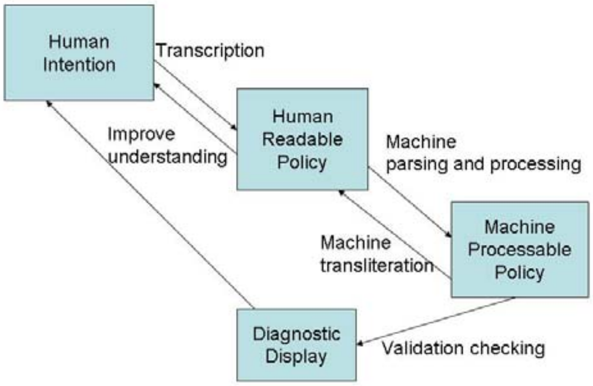
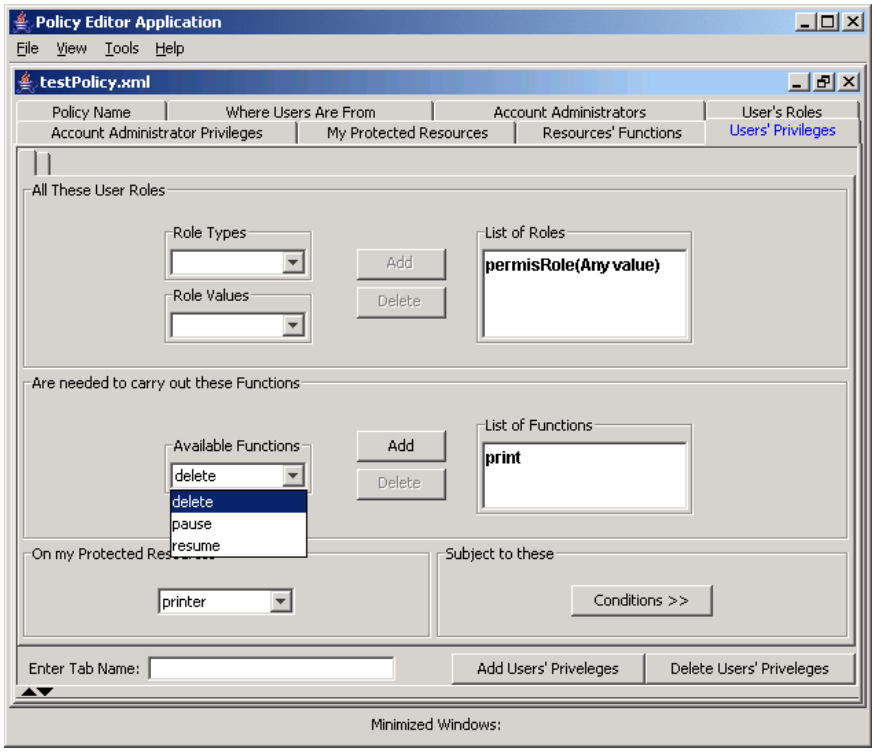


Figure 1. The Virtuous Circle of Policy Specification

3. Progress to Date

To date, we have captured a basic security ontology from interactions with the user community, and built a graphical user interface (GUI), the Policy Editor (see Figure 2) that allows a user to specify a basic authorization policy using this vocabulary. A fuller description of this can be found in [2]. We also have a Policy Wizard variant (see Figure 3) that takes the user step by step through the process of creating a policy, using individual windows from the Policy Editor. The Wizard allows the user to easily create several flavours of a basic authorization policy, but editing an existing policy or adding additional features to a basic policy created by the Wizard, is achieved via the main Policy Editor.

Both GUI tools have screens which display the final policy in either natural language (Figure 4) or XML. The natural language display enables the user to validate, in terms of his own understanding, what the policy actually means to the computer system. This forms the second half of the virtuous circle shown in Figure 1.



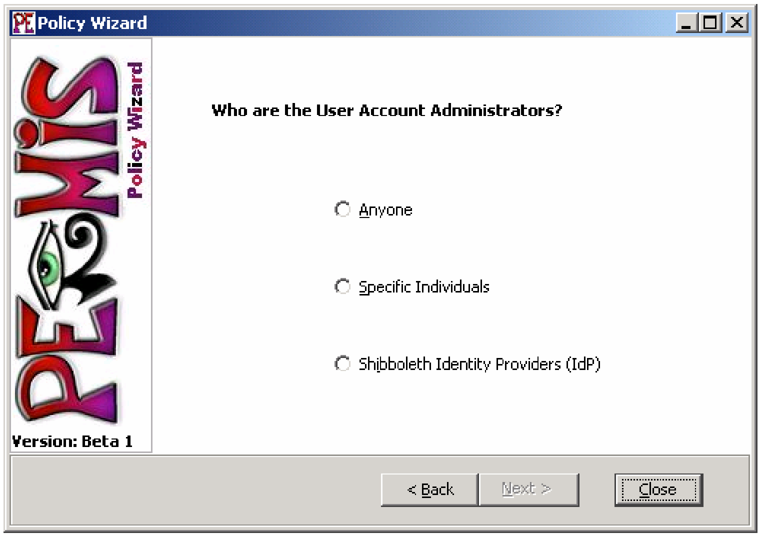
List of Roles: permisRole(Any value)

Available Functions: delete

List of Functions: print

On my Protected Resources: printer

Figure 2. The GUI Policy Tool



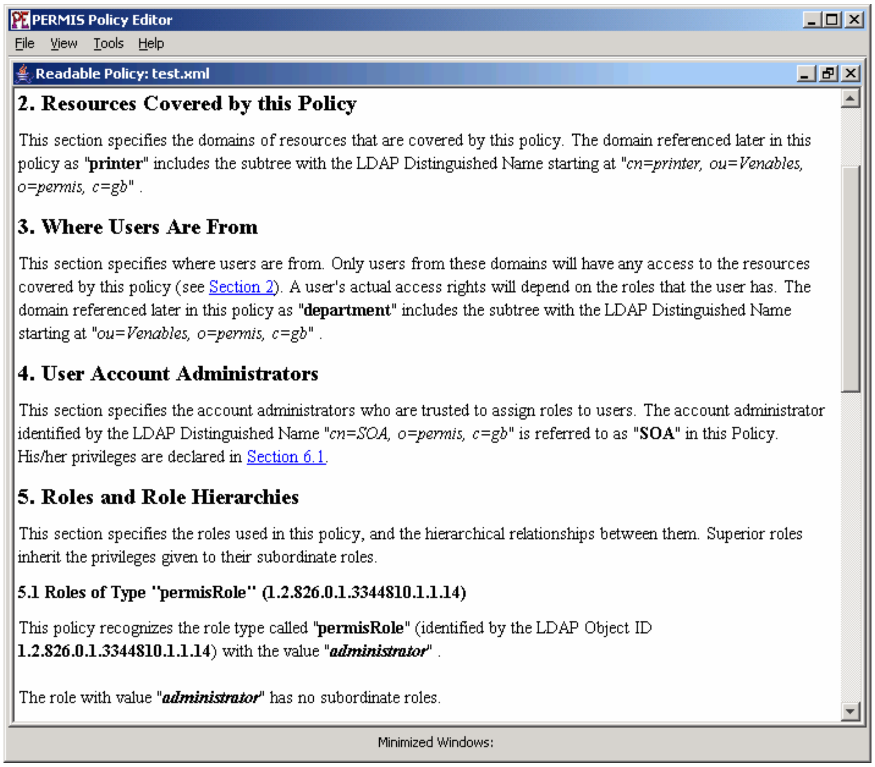
Who are the User Account Administrators?

○ Anyone

○ Specific Individuals

○ Shibboleth Identity Providers (IdP)

Figure3. The Policy Wizard



2. Resources Covered by this Policy

This section specifies the domains of resources that are covered by this policy. The domain referenced later in this policy as "**printer**" includes the subtree with the LDAP Distinguished Name starting at "*cn=printer, ou=Venables, o=permis, c=gb*" .

3. Where Users Are From

This section specifies where users are from. Only users from these domains will have any access to the resources covered by this policy (see Section 2). A user's actual access rights will depend on the roles that the user has. The domain referenced later in this policy as "**department**" includes the subtree with the LDAP Distinguished Name starting at "*ou=Venables, o=permis, c=gb*" .

4. User Account Administrators

This section specifies the account administrators who are trusted to assign roles to users. The account administrator identified by the LDAP Distinguished Name "*cn=SOA, o=permis, c=gb*" is referred to as "**SOA**" in this Policy. His/her privileges are declared in Section 6.1.

5. Roles and Role Hierarchies

This section specifies the roles used in this policy, and the hierarchical relationships between them. Superior roles inherit the privileges given to their subordinate roles.

5.1 Roles of Type "permisRole" (1.2.826.0.1.3344810.1.1.14)

This policy recognizes the role type called "**permisRole**" (identified by the LDAP Object ID **1.2.826.0.1.3344810.1.1.14**) with the value "***administrator***" .

The role with value "***administrator***" has no subordinate roles.

Figure 4. The Policy Specified in Natural Language

Note that the policy is displayed to the user in full natural language using an XSL style sheet. In the next stage of the project the user will be allowed to create a new authorisation policy using controlled natural language.

4. Controlled Natural Language

Natural language processing (NLP) is very hard due to the ambiguities and complex structure of natural language. Machine translation has been continuously refined for decades. Major industry leaders are still performing research into machine translation, paraphrasing and information extraction [8, 9]. However, they provide no free tools to academia. A number of universities in the UK are developing NLP and Information Extraction technologies. However, they are mostly directed at annotating scientific texts and analysing vast sources of information in natural language to spot pieces of text that are of interest to a scientist [12, 13].

In controlled natural language either the vocabulary and/or grammar that can be used are limited, being a subset of natural language. This makes machine processing much easier and more tractable than using free form natural language. The GATE project (http://gate.ac.uk/), lead by the University of Sheffield, has produced a natural language processing kit [5]. It includes a set of tools and grammars that allow English and other texts to be analysed. In the project SEKT (Semantic Knowledge Technology – http://www.sekt-project.com/), the GATE team has investigated the application of controlled languages [10] to the provision of natural language interfaces for tasks such as web service protocol description or ontology construction. Sheffield has developed a Controlled Language Information Extraction [11] tool (CLIE) to aid users in their task. The number of sentence structures allowed by CLIE is very limited, which means that it is very easy to learn to use, much easier than for example OWL or RDF or tools such as Prodigy. However, the vocabulary for classes and instances is unlimited, which means that complex ontologies can still be created.

We are experimenting with using CLIE to construct ontologies for authorization policies. CLIE supports three sentence constructs for creating class and instance hierarchies. “There are <class>”, “<subclass> is a type of <superclass>” and “<Instance> is a <class>”. CLIE supports one sentence construct for defining relationships between classes “<class> (can) have <class>”, and a similar one for adding properties to classes “<class> (can) have textual <property name>”. CLIE supports two sentence constructs for setting property values for instances: “<instance> has <instance>” and “<instance> has property <property> with value <instance>”. This limited language allows us to reproduce nearly all the functionality

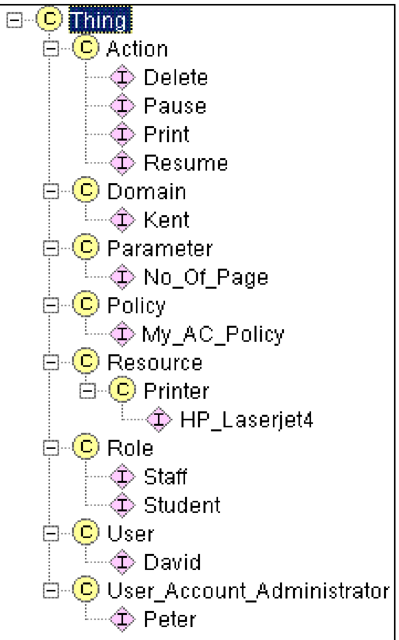
There are policies. “My AC policy” is a policy. There are resources and users. David is a user. Printer is a type of resource. “HP Laserjet4” is a printer. There are domains. Kent is a domain. There are “User Account Administrators”. Peter is a User Account Administrator. There are actions and parameters. Print is an action. Delete is an action. Pause and resume are actions. “No of pages” is a parameter. Actions have parameters. Print has action with value “No of pages”. There are roles. Student is a role. Staff is a role. Resources have actions. “HP Laserjet4” has action with value print. “HP Laserjet4”has action with value delete. “HP Laserjet4” has action with value pause. “HP Laserjet4” has action with value resume.	 Thing Action Delete Pause Print Resume Domain Kent Parameter No_Of_Page Policy My_AC_Policy Resource Printer HP_Laserjet4 Role Staff Student User David User_Account_Administrator Peter

Table 1. An Example Authorisation Policy Ontology using CLIE

of 6 of the 8 tabs in our Policy Editor. (The two tabs that currently cannot be specified are the Account Administrator Privileges and the Users' Privileges.) The left hand column of Table 1 shows the policy sentences typed in by the user, and the right hand column shows the resulting class-instance hierarchy ontology created by CLIE (note that properties and instance property value assignments are not shown in the right hand column).

5. Future Work

CLIE is a good start but still needs some enhancements. A current limitation is that it only allows the "(can) have" relationship between classes and instances. We have a requirement to specify new types of relationship, such as the superior/subordinate relationships in role hierarchies, and the "can assign" relationship between administrators and roles. Sometimes simpler sentence constructs would be more user friendly, for example, making "with value" optional when setting property values. More complex sentence constructs are also needed such as "users with these properties can access resources with these properties providing these conditions are met". This will require us to tailor the GATE software to use the specific grammar of these authorisation sentences using the base ontology provided by CLIE. GATE creates an in-memory semantic representation of the user's newly created sentences using the ontology. Any unrecognised or erroneous words will be highlighted, prompting the user to take some clarifying action. Plugin tools will be developed to extract the user's intended semantics from unrecognised words, from partially specified conditions (e.g. If later than 5), ambiguous phrases (e.g. double negatives) or conflicting semantics (e.g. employees can print but managers cannot, when managers are superior to employees). Finally, the in-memory semantic representation of the policy will be compiled into two XML authorisation policy languages (XACML and PERMIS) so that they can then be displayed in natural language via style sheets, and subsequently read into their respective PDPs for access control decision making.

Turning to the GUI tools, we are currently increasing the ontology used in them and incorporating more complex security concepts such as separation of duties, mutually exclusive roles, obligations and constraints. This is being done by analysing transcript's from interviews with researchers recorded during previous e-Science research projects, collected over the past few years. The extracted ontology will be validated by testing it with potential e-Science users. The users will be asked to specify polices in natural language for a number of scenarios set in their own research environments, and to interpret a number of semi-structured policies and queries.

The final evaluation trial is planned to be conducted with e-Science researchers from a variety of projects – medicine and bioinformatics, sciences, social science, data grids and computational grids. They will be asked to carry out a set of standardised policy specification tasks using the natural language tool and the existing GUI tools. With participants' permission, the interactions will be recorded (both on the system and with a video camera), and participants will be asked to *think aloud* while carrying out the tasks and when interpreting feedback received from the tools.

This will be followed by a brief questionnaire to assess user satisfaction and perceived user effort. The ease of use and effectiveness of the natural language interface will be compared and contrasted with that of the GUI and Wizard interfaces.

References

1 Saltzer, J.H., and Schroeder, M.D. "The Protection of Information in Computer Systems," Proceedings of IEEE, 63(9), 1278-1308, 1975.

2 Brostoff, S., Sasse, A., Chadwick, D., Cunningham, J., Mbanaso, U., Otenko, S. "*"R-What?"* Development of a Role-Based Access Control (RBAC) Policy-Writing Tool for e-Scientists" to appear in Software Practice and Experience, 2005

3 Barman, S. "Writing Information Security Policies". New Riders 2001

4 Reason, J. "Human Error". Wiley 1990.

5 H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia, July 2002.

6 D.W.Chadwick, A. Otenko, E.Ball. "Role-based access control with X.509 attribute certificates", IEEE Internet Computing, March-April 2003, pp. 62-69.

7 OASIS "eXtensible Access Control Markup Language (XACML) Version 2.0" OASIS Standard, 1 Feb 2005

8 Quirk, C., Brockett, C., and Dolan, W.B. "Monolingual Machine Translation for Paraphrase Generation", In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 25-26 July 2004, Barcelona Spain, pp. 142-149.

9 Jennifer Chu-Carroll, Krzysztof Czuba, John Prager, and Abraham Ittycheriah, "In Question Answering, Two Heads are Better Than One", Human Language Technology Conference (HLT/NAACL), 2003

10 S. Pulman. Controlled Language for Knowledge Representation. In *CLAW96: Proceedings of the First International Workshop on Controlled Language Applications*, pages 233–242, Leuven, Belgium, 1996.

11 Valentin Tablan, Tamara Polajnar, Hamish Cunningham, Kalina Bontcheva. "User-friendly ontology authoring using a controlled language". Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), Genoa, Italy, May 2006

12 Briscoe, E. J. and J. Carroll. "Robust accurate statistical annotation of general text". In *Proceedings of the 3rd International Conference on Language Resources and Evaluation* (LREC 2002).

13 Alex Morgan, Lynette Hirschman, Alexander Yeh, and Marc Colosimo. "Gene name extraction using FlyBase resources". In Sophia Ananiadou and Junichi Tsujii (eds.), *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*.