PCIEF: a policy conflict identification and evaluation framework

Vimalathithan Subramanian*

Department of Integrated Computing,
University of Arkansas at Little Rock,
2801 S University Avenue, EIT-579,
Little Rock, AR-72204, USA
Fax: 501-569-8144
E-mail: sxvimalathit@ualr.edu
*Corresponding author

Remzi Seker

Department of Electrical, Computer, Software, and
Systems Engineering
Embry-Riddle Aeronautical University,
600 S Clyde Morris Blvd.,
Daytona Beach, FL 32114-3900, USA
Fax: 386-226-6678
E-mail: sekerr@erau.edu

Srini Ramaswamy

Industrial Software Systems,
ABB India Corporate Research Center,
Bangalore 560048, India
Fax: 501-569-8144
E-mail: srini@ieee.org

Rathinasamy B. Lenin

Department of Mathematics,
University of Central Arkansas,
201 Donaghey Avenue, Conway, AR 72035-0001, USA
Fax: 501-450-5662
E-mail: rblenin@uca.edu

Abstract: Information system security policies have grown in complexity and the emerging collaborative nature of business has created new challenges in creating and managing such policies. These policies address several domains ranging from access control to disaster recovery and depend not only on the business itself but on socio-political/legal requirements as well. Events like collaborative work or project-based organisational units result in the need to create a new information system security policy for the specific work/project, while maintaining status quo of existing policies. This requires identification

Copyright © 2012 Inderscience Enterprises Ltd.

and evaluation of existing policies to enable creating the new policy in line with the existing ones with acceptable deviations based on informed decisions. This paper provides a framework for capturing and converting security policies in terms of an XML format and further into alloy language format. Policies are converted to alloy format for performing further policy consistency analysis using Alloy Analyser.

Keywords: policy conflicts; policy inter-operation; computer security; information systems security; policy evaluation.

Reference to this paper should be made as follows: Subramanian, V., Seker, R., Ramaswamy, S. and Lenin, R.B. (2012) 'PCIEF: a policy conflict identification and evaluation framework', *Int. J. Information and Computer Security*, Vol. 5, No. 1, pp.48–67.

Biographical notes: Vimalathithan Subramanian is a Technical Information Security Officer in Citi at New Jersey, USA. He received his PhD in Integrated Computing Department in 2012 from University of Arkansas at Little Rock, MS in Applied Science in 2011 from University of Arkansas at Little Rock, MTech in Information Technology in 2004 from Punjabi University, India and MSc in Mathematics in 1991 from Madurai Kamaraj University, India. He is a Certified Information Systems Security Professional (CISSP), a Certified Ethical Hacker (CEH), a Certified ISO27001 Lead Auditor (ISO27001LA), and a Certified Information Technology Service Management (ITIL Foundation) professional. He has over 16 years of IT industry experience and of which five years are in the field of information security. His research interests are information security policy evaluation and management.

Remzi Seker is a Professor of Electrical, Computer, Software, and Systems Engineering at Embry-Riddle Aeronautical University, Daytona Beach, Florida. He received his PhD in Computer Engineering from the University of Alabama at Birmingham. His research interests are safety and security critical systems and computer forensics. His research is motivated by the trend in rapid penetration of computer-based technologies into the society. When computer-based technologies become commodities on which people rely, the vulnerability of the society to these technologies increase as well. He, as a researcher, focuses on ways to address the asymmetric threats that arise from rapid, yet necessary use of technology. Aside from professional activities, he participates in variety of venues for increasing public awareness on security.

Srini Ramaswamy earned his PhD degree in Computer Science from the Center for Advanced Computer Studies (CACS) at the University of Louisiana at Lafayette. He leads industrial software systems research group at ABBs India Corporate Research Center. He also serves as a Visiting Professor at the University of Arkansas at Little Rock and an Honorary Adjunct Professor at the Indian Institute of Information Technology – Bangalore. Earlier, he was in academia for 16 years, most recently as Tenured Professor and Chairperson of the Computer Science Department at UALR (2005–2010). His research interests include behaviour modelling, analysis and simulation, software stability and scalability – particularly in the design and development of better software systems, and intelligent and flexible control systems. He has held invited and visiting positions at INSA-Rouen (France), Vanderbilt and UT-Austin. He is a Senior Member of IEEE and ACM, an active member of IEEE SMCS TC on DIS and an Associate Editor for the IEEE Transactions on SMC.

Rathinasamy B. Lenin received his PhD in Applied Mathematics from the Indian Institute of Technology, Madras, India, in 1998. He is currently an Assistant Professor in the Department of Mathematics at the University of Central Arkansas, USA. Prior to joining UCA, he was working as a Postdoctoral Fellow at the University of Arkansas at Little Rock, USA. Earlier, he had also worked as a Postdoctoral Fellow at the University of Twente, The Netherlands, and at the University of Antwerp, Belgium, for about five years. His research interests are in mathematical modelling and simulation of discrete-event systems, stochastic models, optimisation techniques, performance analysis of computer and communication networks, rational approximation, information retrieval and network analysis. He has 49 papers published in international conferences and journals including *Journal of Applied Probability, Queueing Systems, Performance Evaluation*, *IEEE Transactions on Computers*, *Wireless Networks* and *IEEE Transactions on Microwave Theory and Techniques*.

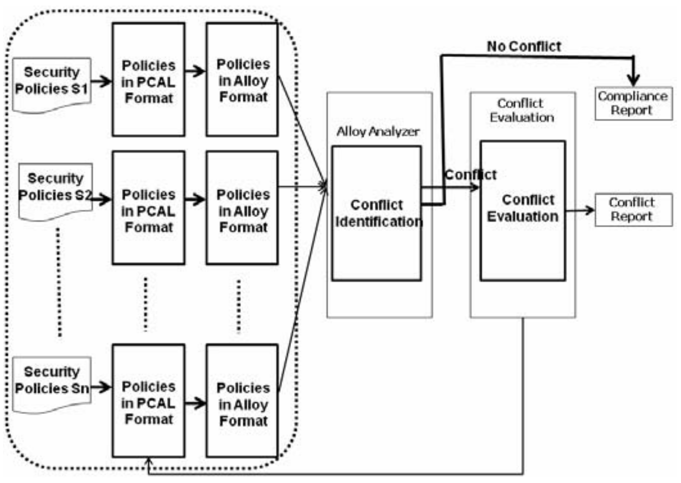
1 Introduction

In this era of converging industries, globalisations as well as competitive cooperation have become inescapable realities for organisations. To remain successful and to outsmart competitors, organisations need to quickly embrace latest technologies even before realising potential challenges associated with them. Organisations also need to utilise their fullest potential of infrastructure so as to eliminate border bounded access to their resources and adopt borderless environment by authorised users whenever they need and wherever they are. This causes network security to transcend from being in a border bounded environment into a borderless environment. Due to this growing complexity of information systems/networks and the need to share the information across geographical and functional boundaries, organisations need to strictly enforce information security policies to protect sensitive data. After all, information flows across any and every network.

With the expansion of an enterprise over diverse geographical locations, probability of policy conflict increases, more so with the increase in legal and regulatory requirements that also vary from region to region. Mergers and acquisitions are taking place in a rapid manner to enable organisations to be more competitive and to retain and potentially increase their market share. Mergers and acquisitions are faced with security issues and concerns as outlined by Skoudis (2007). Reconciliation of disparate sets of information security policies is a challenging daunting task. The inter-connection between dissimilar networks poses security and control issues pertaining to the famous security triad – confidentiality, integrity and availability (CIA) of the information transmitted across these networks some of which are highlighted by Grance et al. (2002). Failing to secure sensitive information may lead to significant legal risk, in addition to customer goodwill related issues that affect the bottom line of the organisation adversely.

To address these threats organisations develop and maintain organisational security policies that safe guard their assets. A security policy is often an overall general statement that outlines specific security requirements that must be met by each entity in the organisation. A number of security policy standards and guidelines are available today for arriving at a good information systems/information technology (IS/IT) security policy. However, given the diversity of operations and requirements from compliance to multiple audit mandates, identifying the optimum policy and to identify policy implementation mechanisms is extremely challenging. The main objective in any security policy is to ensure CIA. Organisational security policies aim to lay a solid foundation for the development and implementation of best practices within an organisation. Security policies can be written using natural languages but they cannot be directly processed by IS elements. Due to inherent ambiguities associated with natural languages, policy interpretation is likely to be subjective and interpreted by different people in different ways. Also, processing policies written in natural language will increase errors due to human intervention. Security policies written in formal logic using mathematical notations can be easily processed by the IS elements and involves less human intervention. But it is difficult to understand policy languages written in formal logic and it is cumbersome to design large complex policies using formal logic, given the fact that the policy needs to be understood by all stakeholders of the organisation.

Figure 1 Policy conflict identification and evaluation framework



There are many policy specification languages specified by various researchers some of them are listed in references section; we review some of them in Section 2. Policy specification languages provide the means to express the intent of the owner of an asset into a form that can be interpreted by the IS elements. These languages become effective when policies are specified using a language that is easy to understand by the administrators and processed in network systems. Hence to overcome these constraints we propose policy conflict analysis language (PCAL) with an ease of use and to give much leverage to security administrators for performing policy analysis. To achieve this, we capture security policies in terms of XML format and convert them into alloy language.

The converted policies are then fed into the Alloy Analyser to identify policy inconsistencies between security policies. This paper deals with a part of our policy conflict identification and evaluation framework (PCIEF) seen in the dotted region in Figure 1. PCIEF deals with the conversion of real world information security policies into alloy format and then identification of security policy conflicts with the help of Alloy Analyser. Policy conflicts are then evaluated based on their type. Each type of security policy conflict is based on their severity with respect to business impact.

2 Related work

Researchers have been seized with IS policy issues. For the purposes of this paper, handling policy specification languages are the most relevant and we review some of the related work within the constraints of available space. Damianou et al. (2001) specified a policy specification language Ponder, which is a declarative, object-oriented language for specifying security policies for both role-based access control (RBAC), and general-purpose management policies. Ponder specifies Meta policies for group of policies to express constraints which limit the permitted policies in the system. Using Ponder one can express authorisations, information filtering, obligations, delegation, and refrain policies. Ponder requires the security administrator to specify much details which limits its capability to provide a high level security policy specification. Also, Ribeiro et al. (2001) proposes security policy language (SPL) a policy-oriented constraint-based language. SPL consists of four basic blocks: entities, sets, rules, and policies. It expresses permission, obligation, and prohibition. SPL specifies only authorisation policies. Policy description language (PDL) (Lobo et al., 1999) developed at Bell Labs is an event-based policy language that specifies policies as declarative rules. Events in PDL can be compound event expressions and the actions can be local or remote method calls, complex workflows that trigger other events to form a hierarchical policy chain. PDL does not support access control policies.

Policy specification languages such as ASL (Jajodia et al., 1997), XACML (Godik and Moses 2003), and Tower (Michael and Vijay, 2001) specify access control policies. Authorization Specification Language (ASL) is a formal logic language. In ASL integrity rules are in the form of meta-policies to specify application-dependent rules that limit the range of acceptable access control policies. ASL provides support for RBAC, but ASL is not scalable to large systems as there is no way of grouping rules into structures for reusability. eXtensible Access Control Markup Language (XACML) is based on XML. XACML is intended to provide an application independent policy language to provide a query and response format for authorisation decision requests. It provides means for writing access control policies. However, the set of possible actions that can be defined in XACML is restricted (Agrawal et al., 2008). XACML leaves the policy analysis to policy analysers. Tower is a policy language to specify RBAC policies. The language is focused in object oriented systems. This language supports role hierarchy, separation of duties both static and dynamic, Chinese Wall policy, delegation and joint action-based access policies. To use Tower programming expertise is needed. The policy language IBM-ACPL (2005) is based on XML Schema from IBM in an attempt to support distributed IT system management. ACPL uses simple text format for writing policies. Policy management for autonomic computing (PMAC) (Agrawal et al., 2005) is a framework that is based on ACPL to simplify the management and automation of products and complex systems. Also, Enterprise Privacy Authorization Language (EPAL) (Schunter, 2003) is based on formal logic from IBM that specifies the authorisation decision in the form of abstract format and specifies attempted resource access in terms of attributes. The algorithm used in EPAL 'first applicable' algorithm makes it difficult to manage large enterprise-wide policies. The Language for Security Constraints on Objects (LaSCO) specified by Hoagland et al. (1998) used a graphical approach for specifying security constraints on objects. Policies defined in LaSCO are specified as logical expressions and by means of directed graphs. LaSCO does not specify confidentiality and integrity.

Application specific policy languages such as ISPS (Zhi et al. 2001), PAX (Nossik and Richardson, 1998), and PPL (Stone et al., 2001) are also worth mentioning here. IPSec Security Policy Specification (ISPS) apply policy constraints on entities such as security gateways along the path of a communication. ISPS provides means to express confidentiality and integrity rules. ISPS does not support authentication, audit, and delegation. Pattern Description Language PAX, a special purpose policy language used to define pattern-matching criteria in policy-based networking devices. In PAX simple patterns are combined to form more complex patterns. The use of field concatenation and field combination, and the ability to name patterns lead to a flexible and powerful language for describing patterns in data communication. Path-based policy language (PPL) is designed to support both the differentiated as well as the integrated services model proposed by IETF. PPL is based on the idea of providing better control over the traffic in a network by constraining the path the traffic must take. PPL does not cover the security functionalities other than access control. But these policy languages are application specific, the extension of these policy languages to other area would be cumbersome. Some of the languages discussed above do not deal with verification of policy inconsistencies.

Most of these policy languages are specific to a certain application or restricted to a certain security domain, for example, access control domain. If some policy specification languages cover more than one security domain then they require either expert knowledge of programming skills or formal logic. Hence, there is a strong need for developing a policy model that is simple and easy for security administrators to perform their jobs and covers all domains of security.

To address these issues, we have come up with our policy specification based on XML called PCAL. Also for performing policy analysis many researchers use alloy language (Jackson, 2006). Alloy is a full fledged declarative language with simpler semantics with more conventional syntax designed for automatic analysis. Alloy Analyser is a tool that can check consistency of policy specifications. It performs fully automatic analysis, and when an assertion is false, it generates a counterexample. Similar to Java, alloy follows an object oriented paradigm. In alloy a model is represented in terms of sets and relations. The structures in alloy model are built from atoms and relations. A relation in alloy is a structure that relates atoms. A more detailed overview of alloy language is provided in Jackson (2006). Park and Kwon (2005) model access control policies using UML and convert access control policies into alloy specification which is then analysed through the Alloy Analyser. Schaad and Moffett (2002) investigated how alloy can be used for verifying RBAC policies. Georg et al. (2001) analysed runtime configuration of distributed system with the help of alloy. RBAC schema is modelled by Zao et al. (2003) in alloy, with the automatic semantic analysis capability, they verified internal consistency of a RBAC schema. Mankai and Logrippo (2005) used Alloy Analyser to analyse conflicts and interactions among access control policies expressed in XACML. They also developed an XACML2 alloy tool. Hughes and Bultan (2004) translated XACML policies to their logical representation in the alloy language and verified them using Alloy Analyser. Dennis et al. (2004) exposed hidden flaws in the UML design of a radiation therapy machine using alloy. Some of these researchers such as Dennis manually converted to alloy language which may lead to human errors. Hence, we use automatic alloy conversion based on our PCAL policy model. Modelling policy specification in alloy language has an added advantage as policies can be verified by using Alloy Analyser. In our work we capture security policies in terms of XML format and automatically convert them into alloy language. We use Alloy Analyser to identify policy conflicts between security policies. Existing Policy conflict techniques are being analysed by the authors and presented in Subramanian et al. (2011).

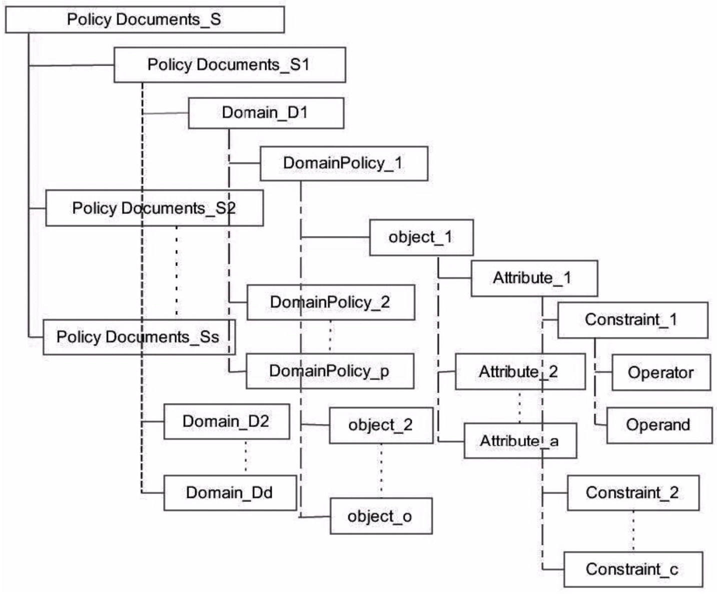
3 Policy model

Policy documents needs to be modelled in line with a policy model to perform policy analysis. This leads us to design a policy model for policy specifications. We analyse real world information security policies with our policy specification and come up with a policy model. Each organisation has a set of security policies, these policies are part of a security domain based on their security properties. According to International Information Systems Security Certification Consortium (ISC)^2 (Tipton, 2009), security policies are classified into ten domains and each security domain addresses security issues pertaining to a specific area of security. In our policy model we refer to a policy in a domain as 'DomainPolicy'. Each policy contains a collection of rules, each rule constrains on an object's use, and therefore each policy contains a set of objects. Object is an entity or a component that needs to be protected in an organisation. Each 'DomainPolicy' constrains certain objects based on their 'attributes'. Attribute of an object is a certain security property of the object. These attribute values are retrieved from objects to limit the object's use. Attributes are specific in type of information they hold. Each attribute has a 'Type' field to specify what kind of data the attribute represents. For example, an attribute 'password modification date' holds type of information as 'date' while attribute 'password length' holds an 'integer' type of information. Constraints on these attributes are specified as a set. Each attribute has a set of 'Constraints' and each constraint represents an expression. An expression is in the form of 'Operator' and 'Operand'. Each 'Constraint' interprets an 'Operator' which in turn contains 'Operand' for the condition. In summary the model is a collection of conditions over objects' use based on their attributes. In other words, a domain policy consists of 'objects' while objects have `attributes' which are limited by 'constraints'.

The resulting tree structure of the model is presented in Figure 2. We have expanded the tree structure for one element in each level for the sake of clear visibility. Organisation S_1 consists of security policy documents from all security domains. Domain D1 contains policies *DomainPolicy*_1, *DomainPolicy*_2, …, *DomainPolicy_p. DomainPolicy*_1 contains objects *Object*_1, *Object*_2, …, *Object_o. Object*_1 contains attributes *Attribute*_1, *Attribute*_2, …, *Attribute_a. Attribute*_1 contains *Constriant*_1, *Constraint*_2, …, *Constraint_c.* Each constraint contains constraint operator and constraint operand.

For performing fine grained comparative analysis of security policies a policy language needs to be able to specify a policy document. In general security policies are created by the information security group of an organisation with top management's support. The security policies may be represented in different form by different security groups. Representation of a single security policy by two different organisations may be in different forms. Comparison of policies in different forms is difficult and comparison of different complex policies in different forms will be even more difficult. One solution to address the problem is to have all security policy documents translated into a common format. We propose a policy specification language PCAL to provide a common format for all security policy documents. PCAL is an intermediate language based on XML structure to interface with Alloy Analyser that checks for policy consistency. We develop a policy capturing form (PCF) to capture security policies in PCAL format. A user can enter the policy document into PCF using a graphical user interface (GUI), the security policy document is then stored in PCAL database in XML structure. We use inter-operable XML structure to represent security policies in PCAL and to have a common language for expressing security policies. For example, every organisation comes up with a set of security policy documents (S_3) in PCAL. In PCAL Security policies are categorised according to their security properties. The categorised classes are called 'Domains', the 'Domains' are classified recursively into 'Domain Policies'. Each policy constrains its objects based on their attributes. Object is an entity or a component that needs to be protected in an organisation. Attribute of an object is a certain security property of the object. Therefore, a domain policy consists of 'objects' while objects have 'attributes' which are limited by 'constraint value'.

Figure 2 Policy model



Security policy conflicts can be categorised into four types. They are no conflict, weak conflict, strong conflict, and extreme conflict. They are defined as follows:

Definition 1 No conflict: Two Security policies of organisations *m* and *n* are said to have no conflicts policies if corresponding elements of both policies are equal.

Definition 2 Weak conflict: Two Security policies of organisations *m* and *n* are said to have weak conflicts if for some *i* and *j*, either corresponding operators have different values or corresponding operands have different values.

Definition 3 Strong conflict: Two Security policies of organisations *m* and *n* are said to have weak conflicts if for some *i* and *j*, there are no corresponding attributes in respective policies.

Definition 4 Extreme conflict: Two Security policies of organisations *m* and *n* are said to have conflicts of type 'extreme' if for some *i* and *j*, either there are no corresponding Objects in respective policies or there are no corresponding policies in respective domains.

With the structured representation of security policies we make the policy document machine process able. Policies are stored in PCAL format and automatically converted into alloy using web application. We are using alloy to represent security policies in terms of atoms and relations. Elements in each set in the mathematical notations will be represented as atoms in alloy. We automated the translation of security policies from PCAL to alloy. The definitions for each primitive in alloy are similarly used as class definitions in object oriented programming structure.

3.1 PCAL policy format

Figure 3 represent shows the policy specification in PCAL format. Policies are captured in PCAL policy language format using GUI-based PCF. Security policies can be easily represented using PCAL policy format and can maintain uniformity in security policy specification.

Figure 3 PCAL policy format

```
<PolicyDoc>
    <PolicyDocId>nn</PolicyDocId>
    <Domain>
        <DomainName> Domain Name </DomainName>
        <DomainId>nn</DomainId>
        <DomainPolicy>
            <Domain PolicyName>Policy Name <Domain Policy Name>
            <DomainPolicyID>nn</DomainPolicyID>
            <Object>
                <ObjectName>Object Name</ObjectName>
                <ObjectId>nn</ObjectId>
                <Object_attrib>
                    <Object_attrib_name >Attribute Name
                    </Object_attrib_name>
                    <Object_attrib_Id >nn </Object_attrib_Id>
                    <Object_attrib_Type >Attribute Type
                                        </Object_attrib_Type>
                    <Object_attrib_Constr>
                        <Constr_Operator>Operator
                                    </Constr_Operator>
                        <Constr_Operand> Operand
                                    </Constr_Operand>
                    </Object_attrib_Constr>
                </Object_attrib>
                ..................
                ...................
            </DomainPolicy>
        </Domain>
</PolicyDoc>
```

4 Alloy structure

An instance of a primitive is represented as an atom of corresponding signature. For example an object ‘password’ is written in alloy as:

one sig password extends Object{}

By specifying it, password inherits all the (set)properties of signature ‘Object’. Automatic generation of alloy code from PCAL is performed by simple mapping operation. Each of the primitive in PCAL is mapped to corresponding atoms in alloy. Policy document in human readable form is converted into alloy code and is ready to be used for further policy conflict analysis using Alloy Analyser. Architecture of the system for generating alloy code for a policy document and algorithm involved to convert from PCAL to alloy are discussed in the next section.

Security policy in alloy format is represented as below in Figure 4.

Figure 4 Alloy structure

```
abstract sig PolicyDoc{
    consists: set Domain
}
abstract sig Domain{
    ID: one Int,
    partof : set DomainPolicy
}
abstract sig DomainPolicy{
    ID: one Int,
    contains : set Object
}
abstract sig Object{
    ID: one Int,
    has : set Attribute
}
abstract sig Attribute{
    ID:one Int,
    type: one Int,
    rule: set Constraint
}
abstract sig Constraint{
    operator: one Int,
    operand: one Int
}
```

After a policy document in PCAL is converted to alloy language using the mapping table, conflict analysis can also be performed.

4.1 Procedure

As we discussed in earlier section, we capture policies using a GUI. The interface stores as well as retrieves policy documents from PCAL policy database. Policy database is a collection of policy documents in PCAL format. Each of the policy documents is then converted into alloy language. We developed a tool to automate the conversion procedure. The tool consists of following components.

1 Policy capture form (PCF): We implement the policy feeder as a web application so that users can access/create/modify/analyse policies from remote location. A user is allowed to feed necessary data for each policy of a policy document. PCF consists of four controls. The first control allows the user to enter a policy rule from the policy document. The second control allows the user to delete a policy rule if needed. Third control allows the user to retrieve the policy document from the database. The fourth control converts policy document into alloy language.

2 PCAL policy database (PPD): PPD is a collection of policy documents in PCAL format. A policy document is represented as per the XML schema as shown in Figure 2. The policies in the PPD are ready to be converted into alloy language.

3 PCAL to alloy converter: It consists of a mapping mechanism to automate policy conversion. Such converted policy documents are stored in alloy policy database. An algorithm to convert PCAL to alloy is presented in Figure 5.

4 Alloy policy database (APD): It is a repository that holds security policy documents in alloy format.

The above mentioned components constitute a full fledged tool which is used to perform necessary pre-processing of security policy document for conflict analysis and evaluation. We implemented this tool as a web application and performed several experiments on real world security policy documents. Some of them were presented in Section 6.

4.2 Policy facts and assertions

In alloy, fact is a formula that constrains the values of the sets and relations. Policy constraints are shown as facts in alloy. The Alloy Analyser enumerates all possible instances that conform to the policy specification. Facts can be named for documentation purposes.

For example, *fact Domainfact Domain.consists = Domain.consists + AccessControl.*

Domainfact is the name of the fact for the signature Domain. 'Consists' is the relation between Domain and its elements. Access control is an element of Domain. An assertion is a constraint that is intended to follow from the facts of the model.

Assertions can be checked by the Alloy Analyser. An assertion can be named so that it can be checked by Alloy Analyser.

For example, *assert weak_conflict {*

*all p: Policy || all d: p: consists || all dp: d: part of || all o: dp: contains || all a: o: has ||all c: a: rule || some p*1*: Policy – p || some d*1*: p*1*: consists || some dp*1*: d*1*: part of || some o*1*: dp*1*: contains || some a*1*: o*1*: has || some c*1*: a*1*: rule || d: ID = d*1*: ID&&dp: ID = dp*1*: ID&&o: ID = o*1*: ID&&a: ID = a*1*: ID&&c: operator = c*1*: operator&&c: operand = c*1*: operand }*

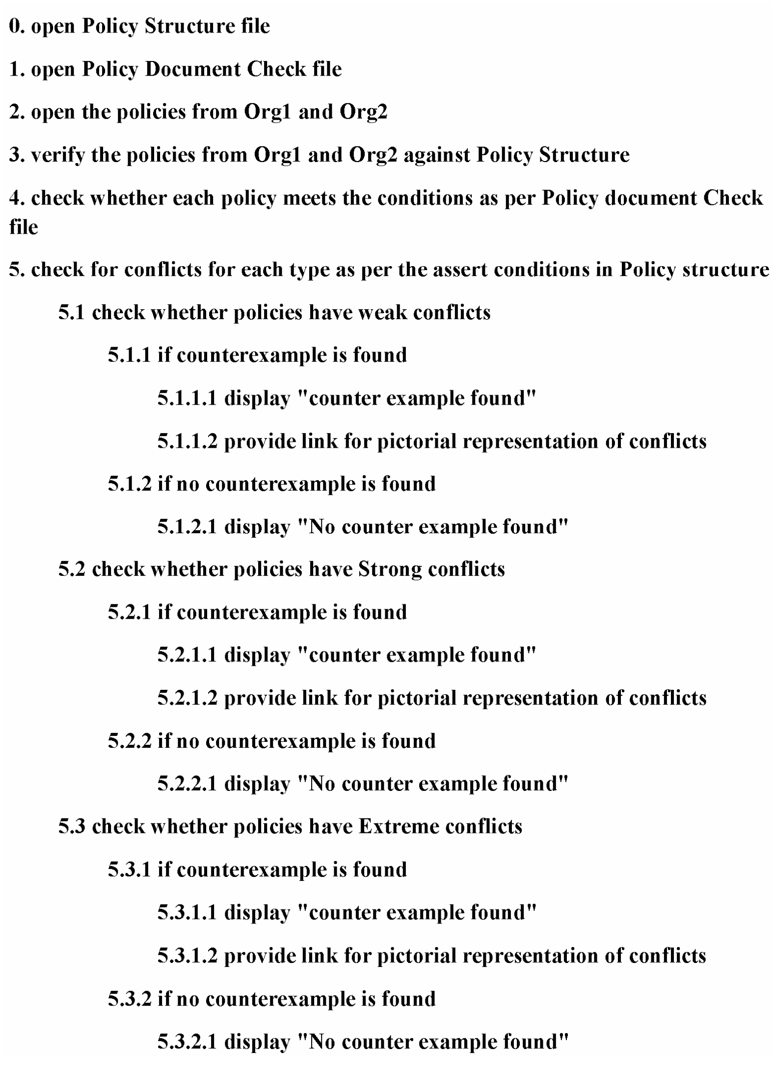
Figure 5 Algorithm for PCAL to alloy conversion

```
create_atom( PolicyDoc, Doc_n)
for each domain
      do
            create_atom( domain, cur_domain )
           add( cur_domain, Doc_n )

     for each DomainPolicy
           do
                 create_atom( DomainPolicy, cur_domainpolicy )
                 add( cur_domainpolicy, cur_domain )

           for each object
                 do
                       create_atom( object, cur_object )
                       add( cur_object, cur_domainpolicy )

                 for each attribute
                       do
                             create_atom( attribute, cur_attribute )
                             add( cur_attribute, cur_object )

                       for each constraint
                             do
                                   create_atom( Operator, cur_operator )
                                   create_atom( Operand, cur_operand )
                                   add( cur_operator, cur_attribute )
                                   add( cur_operand, cur_operator )
                       end
                 end
           end
     end
end

finish_conversion()
     module create_atom( signature_name, instance )
           {
                 print "one sig ". instance ." extends ". signature_name ." {} "
           }
     module add( instance, parent_instance )
           {
                 print "". parent_instance.consists ." = ". parent_instance.consists ." + {".
instance ."}"
           }
```

4.3 Policy comparison procedure

Alloy Analyser compares two given policies, prints conflict summary based on the conflict type. The conflict detection algorithm is presented in Figure 6. The Alloy Analyser first opens the following files: Policy Structure, Policy Document Check, and Policy Files from Org1 and Org2. Analyser then verifies the policy files against the policy structure to ensure the policies conform to specific alloy format. Alloy Analyser checks whether the policy documents meet the conditions laid down in policy document check. The policy document check file consists of facts that need to be satisfied by each policy document. The violations of an assertion for a particular type of conflict are then checked. If there are no conflicts between two policies, then Alloy Analyser displays a message, no counterexample found. Assertion may be valid.

Figure 6 Alloy comparison procedure



```
0. open Policy Structure file
1. open Policy Document Check file
2. open the policies from Org1 and Org2
3. verify the policies from Org1 and Org2 against Policy Structure
4. check whether each policy meets the conditions as per Policy document Check
file
5. check for conflicts for each type as per the assert conditions in Policy structure
    5.1 check whether policies have weak conflicts
        5.1.1 if counterexample is found
            5.1.1.1 display "counter example found"
            5.1.1.2 provide link for pictorial representation of conflicts
        5.1.2 if no counterexample is found
            5.1.2.1 display "No counter example found"
    5.2 check whether policies have Strong conflicts
        5.2.1 if counterexample is found
            5.2.1.1 display "counter example found"
            5.2.1.2 provide link for pictorial representation of conflicts
        5.2.2 if no counterexample is found
            5.2.2.1 display "No counter example found"
    5.3 check whether policies have Extreme conflicts
        5.3.1 if counterexample is found
            5.3.1.1 display "counter example found"
            5.3.1.2 provide link for pictorial representation of conflicts
        5.3.2 if no counterexample is found
            5.3.2.1 display "No counter example found"
```

If there are inconsistencies found, Alloy Analyser displays a message, ‘Counterexample found’. Assertion is invalid and provides a link to a visual diagram representing the conflict. Here, a counter example is a set of instances that satisfy the constraints defined by the model and doesn’t satisfy the analysed assertion.

5 Case study

We consider sample policy rules from real world security policies in different security domains. Alloy Analyser compares two given policies against policy facts and presents the report. Policy conflicts are identified by Alloy Analyser based on the assertions. Alloy Analyser shows output in different format such as graphical, XML format.

5.1 Example 1

Here, we consider sample password policy rules from Access Control Systems and Methodology domain from organisations A2Z Tech and Galaxy Inc. The password policy of organisation A2Z Tech contains 30 rules and password policy of Galaxy Tech contains 26 rules. Here, we have shown only the conflicting rules.

Domain name:	Access control systems and methodology
Policy name:	Password policy

Policy rules:

1 Password should contain at least eight characters.

2 Previously used ten passwords should not be used.

3 Accounts should be unlocked after two hours.

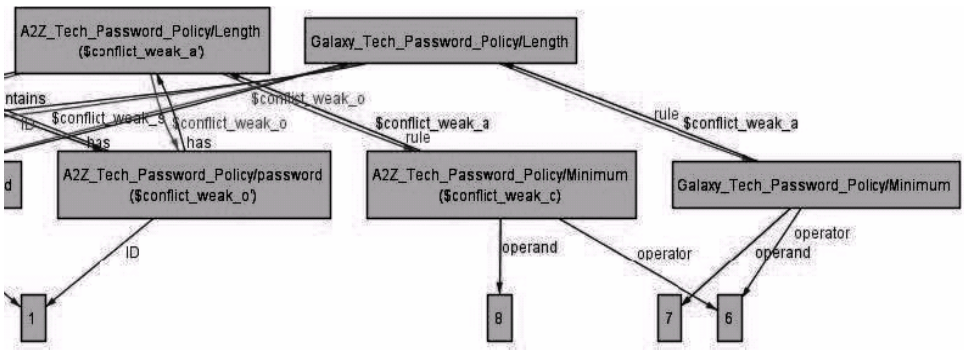
4 Do not use SSN in passwords.

5 Passwords should not be disclosed to any forum.

6 Passwords should not be transmitted through e-mail.

Figure 7 present simplified version of Alloy Analyser output. Minimum password length in A2Z Tech is eight while it is seven in the case of Galaxy Tech. The operand is highlighted with a directional arrow.

Figure 7 Weak conflict – an example


A2Z_Tech_Password_Policy/Length ($conflict_weak_a')
Galaxy_Tech_Password_Policy/Length
ntains
ID
$conflict_weak_s
has
$conflict_weak_o
$conflict_weak_o
has
$conflict_weak_a
rule
rule
$conflict_weak_a
d
A2Z_Tech_Password_Policy/password ($conflict_weak_o')
A2Z_Tech_Password_Policy/Minimum ($conflict_weak_c)
Galaxy_Tech_Password_Policy/Minimum
ID
operand
operator
operator
operand
1
8
7
6

Each policy conflict will be displayed in this fashion. For simplicity sake we have shown one of the weak conflicts using the graphical output. Entire Alloy Analyser output cannot be fit into one page. Hence, we have shown all the weak conflicting elements using the XML output. Constraint operand values which differ are shown below. The policy rules that are missing in Galaxy Tech are also highlighted in the XML output as shown in Figure 8.

Alloy Analyser's graphical output for the strong conflicts are shown in Figure 9. In Figure 9, the attribute 'Account Lockout' is missing from Galaxy Tech Password policy. For strong conflicts attribute section from the XML output is shown in Figure 10. The attributes 'Account Lockout' and 'Transmission' are missing from Galaxy Tech password policy. They are highlighted along with their constraints.

Figure 8 XML extract of weak conflicts

```
<field label="operand" ID="14" parentID="5">

  <tuple> <atom label="A2Z_Tech_Password_Policy_all/Minimum$0"/> <atom
label="8"/> </tuple>

  <tuple> <atom label="A2Z_Tech_Password_Policy_all/previously_used$0"/>
<atom label="10"/> </tuple>

  <tuple> <atom label="A2Z_Tech_Password_Policy_all/SSN$0"/> <atom
label="8"/> </tuple>

  <tuple> <atom label="A2Z_Tech_Password_Policy_all/Forum$0"/> <atom
label="7"/> </tuple>

  <tuple> <atom label="Galaxy_Tech_Password_Policy_all/Minimum$0"/> <atom
label="7"/> </tuple>

  <tuple> <atom label="Galaxy_Tech_Password_Policy_all/previously_used$0"/>
<atom label="6"/> </tuple>

  <types> <type ID="5"/> <type ID="1"/> </types>

</field>
```

5.2 *Example 2*

In this example, we present conflicting data classification policy rules from Security Management Practices domain in organisations A2Z Tech and Galaxy Tech. The data classification policy of organisation A2Z Tech contains 30 rules and data classification policy of Galaxy Tech contains 27 rules. Here we have shown only the conflicting rules.

Figure 9 Strong conflict example

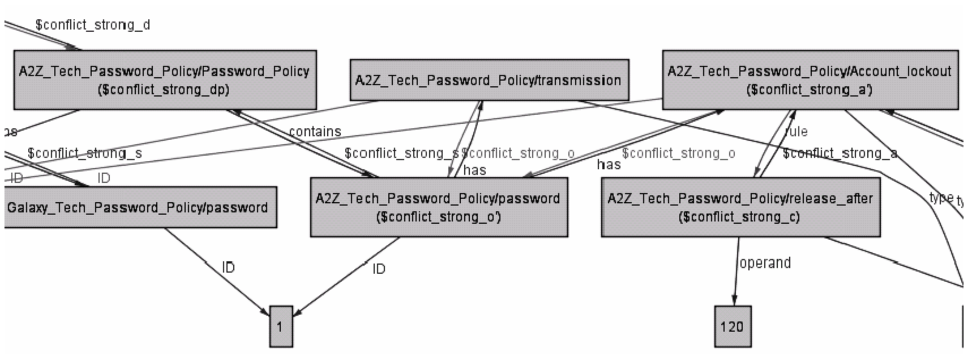

$conflict_strong_d
A2Z_Tech_Password_Policy/Password_Policy ($conflict_strong_dp)
A2Z_Tech_Password_Policy/transmission
A2Z_Tech_Password_Policy/Account_lockout ($conflict_strong_a')
contains
$conflict_strong_s
$conflict_strong_s
$conflict_strong_o
has
$conflict_strong_o
has
rule
$conflict_strong_a
ID
ID
Galaxy_Tech_Password_Policy/password
A2Z_Tech_Password_Policy/password ($conflict_strong_o')
A2Z_Tech_Password_Policy/release_after ($conflict_strong_c)
type
ID
ID
operand
1
120

Figure 10 Strong conflict XML extract

```
<skolem label="$conflict_strong_a" ID="38">

  <tuple> <atom
label="A2Z_Tech_Password_Policy_strong/release_after$0"/> <atom
label="A2Z_Tech_Password_Policy_strong/Account_lockout$0"/> </tuple>

  <tuple> <atom
label="A2Z_Tech_Password_Policy_strong/Thro_email$0"/> <atom
label="A2Z_Tech_Password_Policy_strong/transmission$0"/> </tuple>

  <types> <type ID="5"/> <type ID="10"/> </types>

</skolem>
```

Domain name: Security management practices

Policy name: Data classification policy

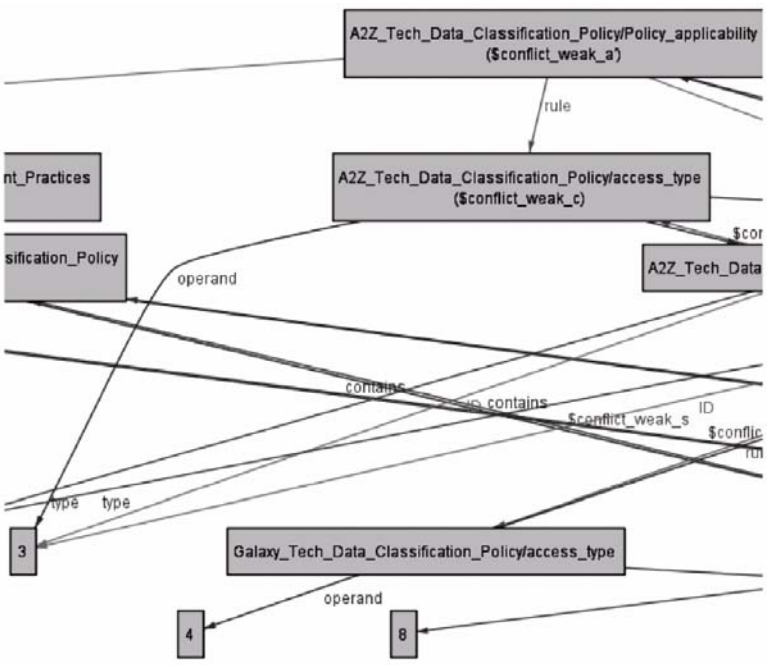
Policy rules:

1 Policy is applicable to information on paper.

2 Customer order is confidential and must be restricted to those with a legitimate business need for access.

3 Staff Directory is internal and meant for use by the company's staff.

Figure 11 Weak conflict example 2


A2Z_Tech_Data_Classification_Policy/Policy_applicability ($conflict_weak_a)
rule
nt_Practices
A2Z_Tech_Data_Classification_Policy/access_type ($conflict_weak_c)
$cor
sification_Policy
A2Z_Tech_Data
operand
contains
contains
ID
$conflict_weak_s
$conflic
type
type
3
Galaxy_Tech_Data_Classification_Policy/access_type
operand
4
8

Alloy Analyser analyses data classification policies from A2Z Tech and Galaxy Tech the output is shown in Figures 11 through 14. In A2Z Tech Data classification policy, the policy is applicable to all data stored in any format, including electronically, on paper and on visual displays, including screens and display boards. While in data classification policy of Galaxy Tech does not specifically mention about the information on paper. Also in A2Z Tech, the policy classifies customer order as confidential and in Galaxy Tech, customer order is classified as Internal. In A2Z Tech, the policy classifies Customer order as confidential (assigned ID 3) and in Galaxy Tech, customer order is classified as internal (assigned ID 4). Refer Figure 11. Figure 12 display the XML extract of the weak conflicting elements.

Figure 12 Weak conflict XML extract

```
<field label="operand" ID="8" parentID="5">

  <tuple> <atom
label="A2Z_Tech_Data_Classification_Policy/pertain$0"/> <atom
label="3"/> </tuple>

  <tuple> <atom
label="A2Z_Tech_Data_Classification_Policy/access_type$0"/>

<atom label="4"/> </tuple>

  <types> <type ID="5"/> <type ID="1"/> </types>

</field>
```

Figure 13 Extreme conflict

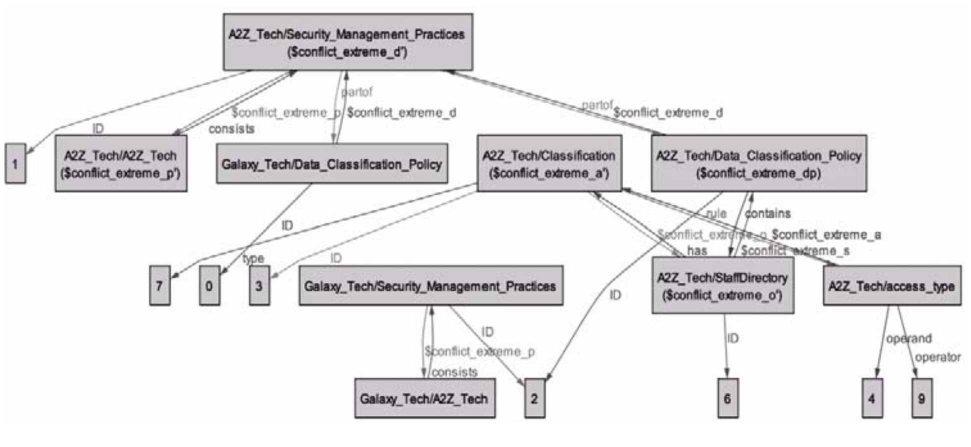
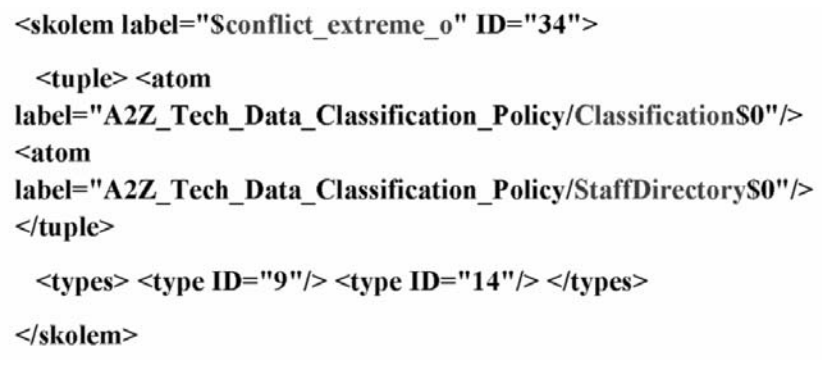

A2Z_Tech/Security_Management_Practices ($conflict_extreme_d')
partof
$conflict_extreme_p
$conflict_extreme_d
partof
$conflict_extreme_d
ID
consists
1
A2Z_Tech/A2Z_Tech ($conflict_extreme_p')
Galaxy_Tech/Data_Classification_Policy
A2Z_Tech/Classification ($conflict_extreme_a')
A2Z_Tech/Data_Classification_Policy ($conflict_extreme_dp)
rule
contains
ID
$conflict_extreme_o
$conflict_extreme_a
has
$conflict_extreme_s
type
ID
7
0
3
Galaxy_Tech/Security_Management_Practices
ID
A2Z_Tech/StaffDirectory ($conflict_extreme_o')
A2Z_Tech/access_type
ID
$conflict_extreme_p
consists
ID
operand
operator
Galaxy_Tech/A2Z_Tech
2
6
4
9

Figure 14 Extreme conflict XML extract



```
<skolem label="$conflict_extreme_o" ID="34">
  <tuple> <atom
label="A2Z_Tech_Data_Classification_Policy/Classification$0"/>
<atom
label="A2Z_Tech_Data_Classification_Policy/StaffDirectory$0"/>
</tuple>
  <types> <type ID="9"/> <type ID="14"/> </types>
</skolem>
```

In Figure 13, the object staff directory is classified as Internal in data classification policy of organisation A2Z Tech, in Galaxy Tech's policy did not mention about Staff directory. Figure 14 represent the XML extract of the extreme conflict.

6 Conclusions and future work

Information security policies address several areas of information security. In this paper, we proposed a PCIEF with a policy specification language PCAL as its backbone. PCAL is used to provide a common format for all security policy documents. We converted policy documents in the form of PCAL using a web-based GUI tool. The policy document in PCAL format is converted into Alloy Language format using a web application for performing further policy consistency analysis. Our policy model will help the user to easily convert human readable security policy documents into PCAL and then into alloy to identify policy conflicts without having to have in depth knowledge of Alloy Language. To use PCAL, the user need not be a programmer; understanding of information security is enough. Moreover if a user wants to use any other analyser to find policy conflicts, he/she can use policy document that are in PCAL format and easily convert them into any other language. Policies in alloy format are analysed by Alloy Analyser, policy inconsistencies are shown in the form of a visual diagram and XML extracts. There has been significant development made by various researchers in the field of policy consistency analysis. However there is a lack of policy analysis for conflicts arising from various compliances and regulations. These conflicts should also be analysed so as to provide a common framework that caters to multiple mandates.

References

Agrawal, D. et al. (2008) *Policy Technologies for Self-Managing Systems*, IBM Press, Indiana, USA.

Agrawal, D., Kang-Won, L. and Lobo, J. (2005) 'Policy-based management of networked computing systems', *IEEE Communications Magazine*, Vol. 43, No. 10, pp.69–75,.

Damianou, N. et al. (2001) 'The ponder policy specification language', *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, Springer-Verlag, Bristol, UK.

Dennis, G. et al. (2004) 'Automating commutativity analysis at the design level', *Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis*, ACM, Massachusetts, Boston, USA.

Georg, G., Bieman, J. and France, R.B. (2001) 'Using alloy and UML/OCL to specify run-timeconfiguration management: a case study', published in *Workshop of the pUML-Group held together with the «UML» 2001 on Practical UML-Based Rigorous Development Methods – Countering or Integrating the eXtremists*, pp.128–141, published by GI publications.

Godik, S. and Moses, T. (2003) 'eXtensible access control markup language (XACML) Version 1.0.', *OASIS Standard*, 18 February.

Grance, T., Hash, J., Peck, S., Smith, J. and Korow-Diks, K. (2002) 'Security guide for interconnecting information technology systems', Technical Report.

Hoagland, J.A. et al. (1998) 'Security policy specification using a graphical approach', Technical Report CSE-98-3, UC Davis Computer Science Department.

Hughes, G. and Bultan, T. (2004) 'Automated verification of access control policies', Technical Report, Department of Computer Science, University of California.

IBM-ACPL (2005) *ACPL: Autonomic Computing Policy Language*, available at http://dl.alphaworks.ibm.com/technologies/pmac/acpl.pdf (accessed on 18 September 2008).

Jackson, D. (2006) 'Alloy specification language and analyzer', available at http://www.Alloy.mit.edu (accessed on 25 May 2008).

Jajodia, S., Samarati, P. and Subrahmanian, V.S. (1997) 'A logical language for expressing authorizations', *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, IEEE Computer Society.

Lobo, J., Bhatia, R. and Naqvi, S. (1999) 'A policy description language', *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, American Association for Artificial Intelligence, Orlando, Florida, USA.

Mankai, M. and Logrippo, L. (2005) 'Access control policies: modeling and validation', *5th NOTERE Conference (Nouvelles Technologies de la R´epartition)*, pp.85–91, Gatineau, Canada.

Michael, H. and Vijay, V. (2001) 'Tower: a language for role based access control', *Policies for Distributed Systems and Networks*, Vol. 1995/2001, pp.88–106, available at http://dx.doi.org/10.1007/3-540-44569-2_6 (accessed on 18 September 2008).

Nossik, F.W.M. and Richardson, M. (1998) 'PAX PDL – a non-procedural packet description language', Technical Report.

Park, S. and Kwon, G. (2005) 'Verification of Uml-based security policy model', *International Conference on Computational Science and its Applications (ICCSA)*, pp.973–982.

Ribeiro, C. et al. (2001) 'SPL: an access control language for security policies with complex constraints', *Proc. Network and Distributed System Security Symposium (NDSS), 2001.*

Schaad, A. and Moffett, J.D. (2002) 'A lightweight approach to specification and analysis of role-based access control extensions', *Symposium on Access Control Models and Technologies (SACMAT)*, pp.13–22.

Schunter, M. (2003) *Enterprise Privacy Authorization Language (EPAL)*, available at http://www.zurich.ibm.com/security/enterprise-privacy/epal/ (accessed on 25 May 2010).

Skoudis, E. (2007) 'Mergers and acquisitions: building up security after an M&A', available at http://searchsecurity.techtarget.com/tip/0,289483,sid14_gci1261024_mem1,00.html (accessed on 18 September 2008).

Stone, G.N., Lundy, B. and Xie, G.G. (2001) 'Network policy languages: a survey and a new approach', *IEEE Network*, Vol. 15, No. 1, pp.10–21.

Subramanian, V., Seker, R., Bian, J. and Kanaskar, N. (2011) 'Collaborations, mergers, acquisitions, and security policy conflict analysis', *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, Oak Ridge, Tennessee, Article 36.

Tipton, H.F. (2009) *Official (ISC)2 Guide to CISSP CBK*, pp.12–15, Auerbach Publications, Florida, USA.

Zao, J. et al. (2003) 'RBAC schema verification using lightweight formal model and constraint analysis', *Proceedings of the eighth ACM symposium on Access Control Models and Technologies*.

Zhi, F. et al. (2001) 'IPSEC/VPN security policy: correctness, conflict detection, and resolution', *Policies for Distributed Systems and Networks*, Vol. 1995/2001, pp.39–56, available at http://dx.doi.org/10.1007/3-540-44569-2_3 (accessed on 25 May 2008).