RC24766 (W0903-067) March 18, 2009
Computer Science

IBM Research Report

A State Transition Model for Policy Specification

Dinesh Verma, Seraphin B. Calo
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Gregory Cirincione
Army Research Laboratories

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

A State Transition Model for Policy Specification

Dinesh C. Verma, Seraphin B. Calo
IBM T. J. Watson Research Center
dverma@us.ibm.com, scalo@us.ibm.com

Gregory Cirincione
Army Research Laboratories
cirincione@arl.army.mil

Abstract

The model of policies used in a variety of computer systems management and security domains has typically followed the paradigm of if-then-else statements. While the use of such statements has resulted in significant progress in the field of policy based systems, their use has also entailed significant effort in defining the policy rules and system models needed for employing policy based technologies. Better models of policy specification can ease some of the difficulties associated with building policy based systems. In this paper, we introduce a new model for specifying policies, based on a user defined state transition diagram describing system behavior, and demonstrate how this model can be employed advantageously in managing policies.

1. Introduction

Policy based management has been proposed and used successfully in many different research prototypes and commercial products. These systems use a variety of policy languages, some examples being Ponder proposed by researchers from Imperial College [1], PDL (Policy Description Language) proposed by researchers at Bell Labs [2], ACPL (Autonomic Computing Policy Language) proposed by researchers from IBM [3], and CIM-SPL (Simple Policy Language for CIM) from the standards association the Desktop Management Task Force [4]. Several other languages have been used in other research prototypes, commercial products and military pilots. Policies have been used in many contexts ranging from network management [5], security management [6], server data farms [7], storage systems [8], sensor networks [9] and privacy management [10], among others.

Despite the wide number of languages, the policy specifications invariably follow the model of an if-then-else expression, the format for a rule in predicate logic. The languages may differ in some nuances from a predicate logic rule, e.g., they may introduce the notion of an event as a qualifier in an if condition, or differentiate between obligation, authorization and delegation policies. However, all of the policy languages can be mapped relatively easily into rule-based semantics.

The use of predicate logic rules for specifying policies has several advantages as pointed out in the existing literature related to policy technology. The rules can be analyzed for conflicts, overlaps and inconsistencies. The rules simplify the task of managing systems and reduce the burden imposed on human administrators. However, many challenges still remain in the applications of policy based systems. Determining the right set of rules that need to be specified in any context is difficult and frequently non-intuitive. The specification of rules also needs to be accompanied with a complete or near complete specification of a model of the system to which the policies are to be applied. In infrastructure systems where the implementation of a standard model like CIM exists, such a specification may not be a significant issue. However, for most distributed applications, a full model of the system is time-consuming and difficult to develop.

In order to address the limitations of current policy based systems, we need to explore new models of policy which can eliminate the requirement to specify a full system model, and provide a more intuitive specification of administrator intent than the writing of a set of predicate rules. In this paper, we present such an approach which offers these advantages.

In section 2 of this paper, we introduce our model for specifying policies using a state transition diagram. In Section 3, we show how our model can be mapped to a set of predicate logic rules, and thus all of the existing policy infrastructure can be used in conjunction with our state transition model of policies. In Section 4, we discuss some algorithms that can be employed for validation of the policies when specified

in terms of a state transition diagram. In Section 5, we present some examples of applications of this model of policy to real systems. We discuss the usability aspects of state transition diagrams in Section 6, and draw our conclusions in Section 7.

2. The State Transition Model of Policies

Policies are a way to represent the intent of an administrator as to how a computer system ought to behave. We propose that the administrator intent be specified by means of a state transition diagram that describes the allowable states in which a system ought to be operating, and the conditions under which a system ought to transition from one state to another. The set of all states, the conditions under which the transitions occur, and the means by which that transition can be orchestrated describe the policy constraints under which the system should operate.

In order to specify policies using this model, an administrator would need to specify the following three pieces of information:

(i) ***A set of states***. Each state is defined in terms of constraints over some of the variables that are associated with the system. The set of states is mutually exclusive in that the system can exist in only one state at any time.

(ii) ***A set of transitions***. Each transition is an arc between two different states, and denotes a movement from one state to another. Each transition is defined by a procedure which can be used to move the system from the previous state to the next state

(iii) ***A set of transition conditions***. These define the events under which the administrator would like the state transitions to occur.

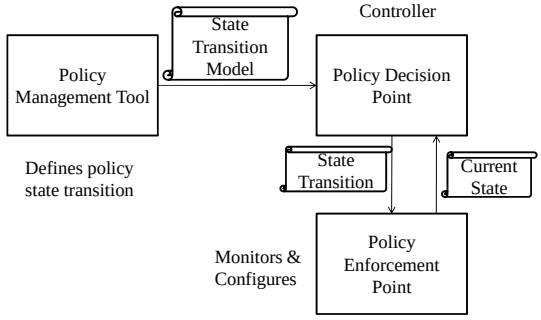

Figure 1. Policy Management Architecture

With this definition of policies, we still expect a policy-enabled system to follow the conventional policy management architecture that includes: a policy manager, policy decision points (PDP) and policy enforcement points (PEP), as shown in Figure 1. However, the function of each of these components is slightly different than that of their counterparts in conventional policy management systems. The Policy Management tool allows the administrator to define and draw the desired state transition diagram for the system. The PEP is responsible for monitoring the system and determining which of the states the system is currently in, based on the values of the system's state variables. The PEP invokes the PDP when one of the state transition conditions specified in the policy state transition diagram becomes true, and the PDP advises the PEP to invoke the transition function to change the state of the system. The PEP can be considered a monitor and the PDP a controller of the system in this case. We will use the term *resource* to identify the computer system to which a particular policy is applicable. In this case, the PEP is the resource monitor and the PDP is the resource controller.

An alternate implementation of the state transition model of policies would be to convert the state transition specification into a set of policy rule specifications, and then use the traditional policy architecture to implement the system.

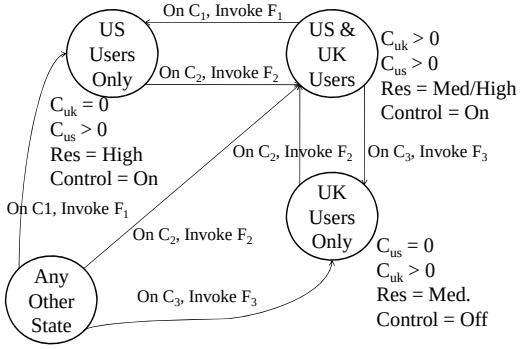

Figure 2. Example Policy

As an example, let us consider an unmanned airborne vehicle (UAV) which is being used to survey an area by a coalition of US and UK forces. The UAV is capable of delivering images at high resolution or at medium resolution. Let us make the simplifying assumption that the UAV only produces one information stream which could be in either high or medium resolution. We also assume a global control mode which indicates whether the user can control the

flight path of the UAV, or if it follows a preprogrammed path. A military planner may define three states for the use of the UAV: used by US forces only, used by UK forces only, and used by both US and UK forces. A state transition model that can be used to represent policy based control in this case is shown in the figure above.

The state transition model uses four state variables of the resource (UAV) to define the state, a count of U.S. users of the UAV (C_{us}), a count of UK users of the UAV (C_{uk}), the resolution of the image, and a status variable indicating whether the control is enabled.

The above example is just a toy example meant to illustrate the state transition model as a way to specify the policies. In any state, some of the variables are controllable (e.g., the resolution and the ability to control the UAV), while the others cannot be controlled by an automated system (e.g., the users in the system). However, when defining the policies in terms of the state, the military planner does not need to worry about these nuances of manipulating the state variables in the system. It is just that the constraints imposed upon the system by these variables needs to be satisfied.

The above toy example also illustrates an important difference between policies and mechanisms. The specification of the intended system states and their transitions is not the same as the actual state transitions that the system may experience on its own. The actual system state may consist of many variables, and not all of those variables need to be used in the definition of the policies by the administrator. Assuming that the same states that the administrator specified are used to track the status of the system, the mechanisms included in the operation of the system dictate how the state transitions happen without any constraints from the administrator. The policy state transition diagram dictates the conditions under which the system would need to make a state transition different than what it would do on its own. State transitions that happen naturally are ones due to mechanisms implemented in the system. State transitions that would not necessarily happen naturally and are imposed upon the system by an automated management process because of the policies (as indicated in the policy state transition diagram) specified by the administrator are the result of policy enforcement. In most real use cases, we would expect the administrator to specify the desired state transition diagram without worrying about whether those changes occur naturally within the system or are happening because they are forced by the PDP/PEP because of the policies that are in effect.

3. Relationship to Rule based Policies

A set of policies defined by a state transition model can be mapped to a set of policies that are represented by a system of predicate logic rules. One can also construct a state transition diagram model for policies that are defined by predicate logic rules. Thus, the state transition model is as expressive as a predicate-rule based model of policies. The state transition model of policies, however, provides additional information about the allowable states of the system under policy control.

In order to show that a state transition model can be converted to a set of rules, it suffices that we show that it can be done for a state transition model with two states only. Let the two states be s_1 and s_2 respectively. One can easily define predicate functions $p(s_1)$ and $p(s_2)$ which are true if and only if a system is in the corresponding state. One can now define another logical expression t_{12} which is true if the system moves from state s_1 to state s_2, and t_{21} which is true if the system moves from s_2 to s_1. In other words t_{12} is true if $p(s_1)$ is true at a time instance t, the condition c_{12} is true at time instant t, and $p(s_2)$ is true at the next time instance $t+1$. If the functions f_1 and f_2 are invoked on the transition of states, then one can write the two rules:

If t_{12} then invoke f_1
If t_{21} then invoke f_2

A similar collection of rules can be used to define other transitions in the state transition model. Thus, any policy statement expressed by means of a state transition model can be expressed as a collection of predicate logic rules.

In the reverse direction, a state transition diagram can be formulated to represent a set of rules. Let us consider a simple system which consists of two variables v_1 & v_2. Let us suppose actions a_1 and a_2 are defined. Now let us consider the set of predicate logic rules given by:

If v_1 then take a_1
If v_2 then take a_2

Necessarily, v_1 and v_2 are expressions over some set of variables. Similarly, a_1 and a_2 are operations on some set of variables. Take the collection of these variables as the state variables of a system. Define v_{12} as being true over the region in which both v_1 and v_2 are true, and false otherwise. Define v_1‘ to be equal to

v_1 except for the region over which v_{12} is true over which it would be false, and v_2' to be equal to v_2 except for the region over which v_{12} is true. Then v_1',v_2', and v_{12} are mutually exclusive.

Define any convenient state, s_0, over the collection of state variables. Define s_1 to be the state of the system after action a_1 has been executed, and s_2 to be the state of the system after action a_2 has been executed. Define s_3 to be the state of the system after both a_1 and a_2 have been executed (Note that if the order in which the two actions are taken matters, then there is an ambiguity in the subsequent state. However, that ambiguity exists in the original set of rules as well. This can be resolved in both systems by ordering the application of the actions. Alternatively, in the transition diagram, auxiliary state variables can be defined to separate the different outcomes.) With v_1',v_2', and v_{12} as conditions for invocation of actions a_1 , a_2 , and both a_1 and a_2 , we have a state transition diagram that models the original rules.

It follows that the state transition based policies are as expressive as the rule-based policies. Therefore, one can use the alternative representations interchangeably for many purposes. The state transition diagrams, however, capture the system evolution in a more convenient fashion.

Some of the analyses that can be performed on state transition policies are described in the next section.

4. Analysis Algorithms

Since a policy will be expressed as a desired state transition diagram, there are several analysis functions that can be performed in order to catch potential mistakes, conflicts and inconsistencies in the policies. The following is a list of some of the validation routines that can be run on the state transition model of policies.

State Description Errors: In the state transition models, states need to be defined to be mutually exclusive. Since each state is defined as a set of logical constraints on a set of different system variables, one can identify the conditions under which two states are not mutually exclusive. The errors in this specification can be caught by pair-wise comparison of any two state descriptions.

Policy Simplification: A state transition diagram can be simplified to eliminate redundant states, as well as to simplify transition conditions. A state is redundant if it is defined by a state condition that is false, or it can not be reached from the current state of the system. A transition occurring on a condition that will always be false can be eliminated. Multiple transitions occurring between two states can be combined into a single transition which is a disjunction of individual transition conditions, and the resulting logical expression can be simplified. If the resulting transition condition is always true (tautological), the two states can be merged.

Policy Composition: If there were a single administrator specifying a state transition diagram based policy for a system, there would not be any conflicts. However, when two or more organizations are involved in specifying policies for the operation of a system, they may express policies using different state transition diagrams.

If a system has to be operated within the constraints specified by two organizations, then the two state transition diagrams need to be composed together. In these cases, the composition algorithm would be as follows:

If there are N states in the first state machine and M states in the second state machine, then the composed state machine has NM states, each composed state consisting of a pair of states selected from the component state machines.

The state condition of the composed state is the Boolean addition of the state conditions defining each of the states in the individual state machines.

If the original state machine in a component machine has a transition t_1 happening on a condition c_1 from state S_1 to state S_2, then the composed state machine has a transition from the composed state (S_1,R) to the composed state (S_2,R) where R is a state in the other component state machine. If as a result of this definition, there are two parallel transitions in the composed state machine between the two set of states then that is reduced to a single transition on a condition which is the logical or of the two conditions under which the parallel transitions occur.

The above described process is a natural way to combine two state transition models.

Policy Consistency Checking: When policies from one or more organizations are applicable to a system, then the effective set of policies can be obtained by composing the two state transition diagrams. Conflicts

between policies can be identified by determining inconsistencies and conflicts that occur in the combined state machine.

In the specification of the state transition models, some conflicting constraints may exist. As an example, there may be two transitions out of a single state to two different states which occur on the same conditions, or occur on conditions that can both be simultaneously true. In order to be well-formed, the transitions happening out of a single state must all be happening on mutually exclusive criteria. There should be no state which automatically transitions to another state on a tautology. Also, there should not be a state description error in the specification.

Consistency checking in the state transition model is simply composing the two policies together and validating that the resulting state transition diagram does not have any errors.

Resource Aggregation: The policy state transition diagrams are specified on a per resource basis. However, one can combine state diagrams for multiple resources into a single transition diagram. This can be done by defining the new resource as the system containing all of the original resources, and defining states which have state conditions augmented to match the specific criteria for each of the individual resources.

5. Example Applications

In order to demonstrate the effectiveness of the state transition model for policies, let us compare the specification of policies for some application scenarios. We consider the specification of policies for a coalition sensor network as described in [9]

A coalition network environment consists of multiple military coalition members who are engaged in performing a joint operation. For this illustration, we assume that the coalition partners are the U.S. and UK armies, and they have come together to plan a joint operation. In the context of a joint operation, they have deployed a sensor network which provides information flows to networks that belong exclusively to either the U.S. or UK systems. One operational environment for the coalition is shown in the figure below which depicts a coalition camp-site connected to the US and UK sites.

The establishment and operation of such a campsite can be viewed as happening in three distinct stages: a mission planning stage, an operation planning stage and a operational stage. In the mission planning stage, the U.S. and UK planners discuss the types of assets they will each bring to the table for setting up the coalition camp-site. These discussions are governed by policies that dictate what asset information can be shared with other coalition partners. In the operation planning stage, members of the coalition team determine aspects such as the number and location of sensor assets that ought to be deployed at the coalition camp site, and the policies governing the information flow between the coalition camp site and the individual country sites. In the operational stage, the camp site is functioning; and policies determined in the operation planning stage need to be enforced on the operational network.

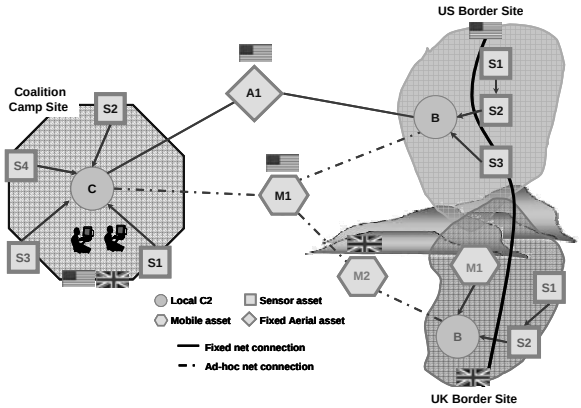

Figure 3. Coalition Base Camp Example

The following types of policies may need to be specified in the coalition sensor network scenario.

- *ISR Asset Characteristics Exchange*. Policies that state what information about sensors and other ISR assets can be exchanged between coalition partners. These will be an input in the mission planning stage.
- *Local Command & Control.* Policies that delineate the command structure, the concomitant roles, their authorizations, and their obligations, including who can develop and modify missions, taskings, and policies. These will be defined in the mission planning stage and enforced during the other two stages.
- *Platform Control*. Policies that define whom, with what authentication, and under what conditions platforms (e.g., UAV's, UGV's, robotic vehicles,

etc) can be controlled, configured, moved, and re-tasked. These are defined in the operational planning stage and enforced during the operational stage.

- *Sensor and Sensor System Control*. Policies that define whom, with what authentication, and under what conditions sensors and sensor systems can be controlled, configured, moved, and re-tasked. These are defined in the operational planning stage and enforced during the operational stage.
- *Sensor Information Access Control*. Policies that define whom (person, C2 element, data fusion element, etc), with what authentication, under what conditions, and in what form (i.e., raw, processed, fused) sensor information can be accessed. These are defined in the operational planning stage but can be modified during the operational stage.
- *Information Flow Protection*. Policies that define how information flows are to be secured and protected (confidentiality, integrity, etc.). These are defined in the operational planning stage and enforced during the operational stage.
- *Information Dissemination*. Policies that describe the conditions/events under which information must be sent and to whom; and, the conditions and to whom information can be provided when queried. These are defined in the operational planning stage but can be modified during the operational stage.

Let us examine how these policies can be expressed using the state machine transition model.

ISR Asset Characteristics Exchange: To specify these policies, a state transition diagrams is attached to each type of sensor asset. One can optionally attach state diagrams to a collection of sensor assets, or to a subset of attributes of a sensor. Let us use the term resource to refer to the unit to which the state transition model is attached.

Let us assume without loss of generality that we are specifying the policies for the resources owned by the U.S. Each state transition diagram consists of three states, one marking the resource being in a state where it is available only to the U.S., the other where it is available to U.S. and U.K., and the third where information about the asset is available publicly to any member. The conditions under which the specified resource can move from one state to another can be described as transitions. These conditions can include constraints on the resolution of data produced by the resource, its date of manufacture, or its capabilities.

Local Command and Control(C2): In this case, the resource is an individual and each state corresponds to a set of C2 authorizations to which an individual in that state would be entitled. The conditions under which an individual would be allowed to expand its roles and responsibilities to get additional authorizations is captured by the transition arcs. As an example, the state transition diagram below authorizes a U.S. member with the rank of a Major to define operational policies. However, if this person is the highest ranking individual in the U.S. team, he is also authorized to modify asset exchange policies.

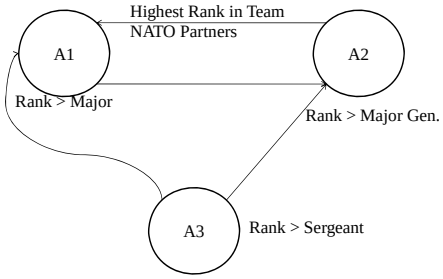

Figure 4. Example Policy for Coalition Operation

Other control policies, e.g., *platform control policies*, *sensor control policies*, and *sensor information access control policies* are described similarly. The states correspond to a set of permitted access levels, and the conditions under which that access is granted; and, the conditions under which a person can be provided more access (or less access) are outlined by means of transition arcs. The resources to which these policies are attached are different, platform control policies are attached to platforms, sensor control policies to sensors, and sensor information access control policies to the specific feeds from the sensors.

Information Flow Protection Policies are described by attaching policies to information flows, which are the resources in this case. To define protection policies, each of the different options for protecting a flow corresponds to a state in the policy specification diagram. The conditions under which a flow should be switched from one type of protection (e.g., being sent using a weak encryption algorithm) to another type of protection (e.g., a stronger encryption algorithm) describe the transition arcs.

Information Dissemination Policies are described by using a sensor as a resource. A state can be described by a set of distinct receivers that need to receive information from that sensor, along with an attribute that indicates whether information is enabled or disabled for those receivers. The conditions under which they are enabled or disabled are indicated on the transition arcs.

Thus, these policies can be expressed in the state transition model effectively and compactly.

6. Usability Considerations

Having introduced the state transition diagram model of policies, we now present the case why this model for policies is a more usable version that the rule-based policy language. The following are some of the areas where policies represented as state transition diagrams will result in a specification that will be easier for system administrators to specify.

Partial State Modeling: One of the key challenges associated with specifying policies as rule-based systems has been the need to specify the complete model of the system being controlled using policies. As an example, the use of the CIM-SPL policy language needs to be accompanied with a CIM model of the system, which is a complex model that may or may not be readily available in many systems. Other policy specification systems impose their own state modeling systems which is sometimes redundant with the available description of the system. As an example, the specification of differentiated services policies on computer servers requires specifying a policy model that requires the duplication of a set of network MIB specifications. The duplication is an additional work at best, and at its worst can result in an inconsistent specification. On the other hand, using a state transition diagram model only requires using a subset of the model that is actually used in the specification of policies. Identifying and using only the subset that is relevant to the policies being defined reduces the chore of policy specification significantly.

Visual Representation: The set of rules that are used for conventional policy specification are represented in textual or script based format that are easy to consume for computing systems but relatively complex for humans to understand intuitively. The program representation of policy represented in a language like CIM-SPL or Ponder are difficult to understand at a glance even for people familiar with the language. On the other hand, the state transition diagram model provides a visual representation of the indicated behavior of the system. The visual representation provides a much easier representation for humans as to the indicated effect and operation of the system. Furthermore, the state diagram can be rendered in a textual or XML representation to derive the benefits of fast computer interpretation and for administrators to easily cut and paste policies from one system to another.

Separation between Policy & Mechanism: The policy based system management approach has always been subject to criticism that the differences between policies and mechanisms of a system are not readily apparent. Policies are supposed to be high-level guidelines for system operation, while mechanisms are the natural behavior of the system when left to itself. However, in a rule-based representation of policies, they have frequently been mixed up with system mechanisms, and in various instances been confused with system configuration. The use of a state transition diagram model, in which one can readily identify differences between the natural state transition the system would undergo on its own (mechanism) and the ones that are additional (administrator's intent) are readily identified. Such identification is very difficult in a rule-based representation.

Other advantages of a state transition diagram approach include the use of a familiar paradigm for system administrators, and the grouping of different conditions into a cluster of states. The description of a small number of states provides for a simpler model that can characterize the system in a manner that will be more intuitive for the system administrators.

7. Conclusions

In this paper, we have proposed a new model for specifying policies using a state transition diagram paradigm. This state transition diagram paradigm is simple and flexible, and allows the specification of many different types of policies. The state transition diagram model does not require a full specification or modeling of a computer system or resource to which policies are attached. It provides a new way to represent policies that will be more effective and enable a more compact and usable implementation of policies.

8. Acknowledgements

The research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

9. References

[1] N. Damianou, N. Dulay, E. Lupu, M Sloman, "The Ponder Specification Language," in Proc. of Workshop on Policies for Distributed Systems and Networks (Policy2001), January 2001.

[2] J. Lobo, R. Bhatia and S. Naqvi, A policy description language. In Proceedings of AAAI, Orlando, FL, 1999

[3] I.B.M., Autonomic computing policy language. http://dl.alphaworks.ibm.com/ technologies/pmac/acpl.pdf, 2005.

[4] DMTF: CIM Simplified Policy Language (CIM-SPL). V 1.4.5, http://www.dmtf.org/apps/org/workgroup/policy/.

[5] D. Agrawal, K-W. Lee, and J. Lobo. Policy-based management of networked computing systems. IEEE Communications, 43(10):69–75, 2005.

[6] D. Clark and D. Wilson, A Comparison of Commercial and Military Computer Security Policies, IEEE Symposium on Security and Privacy, 1987.

[7] K. Appleby et. al. , Oceano-SLA based management of a computing utility, Proceedings of 2001 IEEE/IFIP International Symposium on Integrated Network Management, pp 855-868, Seattle, WA, 2001

[8] Dakshi Agrawal, James Giles, Kang-Won Lee, Kaladhar Voruganti, Khalid Filali-Adib: Policy-Based Validation of SAN Configuration. POLICY 2004: 77-86

[9] T. Pham, G. Cirincione, D. Verma and G. Peason, Intelligence Surveillance and Reconnaissance fusion for coalition operations, International Conference on Information Fusion, 2008.

[10] G. Jarjoth and M. Schunter, A privacy policy model for enterprises, Proceedings of Computer Security Foundations Workshop, 2002.