External Object Trust Zone Mapping for Information Clustering[!]

Yanjun Zuo[1], Brajendra Panda[1]

[1]Department of Computer Science and Computer Engineering
University of Arkansas, Fayetteville, AR 72701

Abstract. In a loosely-coupled system various objects may be imported from different sources and the integrity levels of these objects can vary widely. Like downloaded information from the World Wide Web, these imported objects should be carefully organized and disseminated to different trust zones, which meet the security requirements of different groups of internal applications. Assigning an object to a trust zone is called trust zone mapping, which is essentially a form of information clustering and is designed to guide internal applications when they are using objects from different zones. We developed methods to perform trust zone mapping based on objects' trust attribute values. The defined threshold selection operators allow internal applications to best express their major security concerns while tolerating unimportant issues to certain degrees. As two major trust attributes, the primary and secondary trust values are explained and we illustrate how to calculate each of them.

1 Introduction

Information assurance is a major concern for participating subjects in a loosely-coupled system such as a virtual organization, a federated system, or a dynamic coalition since various objects may be imported from different sources and the qualities of these external objects can vary widely. Conventional computer security and information assurance mechanisms, such as access control [8][9][10] and information flow models [11][12][13][14], have limitations when being applied to these semi-open systems since they are originally designed under a closed-world assumption and users must be known in advance. This assumption may not be valid for semi-open systems whose members dynamically join and leave the systems.

Clustering imported objects in a secured manner is important to facilitate information assurance and comply with the internal security polices of a computing system. One important aspect of information assurance is to disseminate data to different zones based on their security characteristics. All the members of a trust zone share the required trust features as defined for that trust zone.

Component-based approach provides a way to study an object's trust attributes. In [1], the authors developed a formal model to represent object component information and use this information to reason on an object's trustworthiness. Some

[!] This work was supported in part by US AFOSR under grant FA9550-04-1-0429.

objects are processed or produced from others by performing certain logical functions or methods. We call the former as compound objects and the latter as their components. Qualities of components directly determine the integrity of a compound object. This is similar to the case of logic reasoning that if some conditions are wrong, the conclusion is hardly correct. On the other hand, for a given compound object, if its components are highly trusted and the formula that was used to integrate the components is accurate, then it is very likely that the compound object is correct and trustworthy.

In our model, each object is associated with a data value and also a trust value for its owner. A compound object's data values (or trust value) are dependent on the data values (or trust values) of its components. This dependency relationship is transitive. For example, if C is a component of B (hence B is dependent on C) and B is a component of A (hence A is dependent on B), then A is indirectly dependent on C. We call C a sub-component of A. This component-based object dependency relationship can be visualized as a tree, where the root of the tree is a compound object, each vertex represents a component or a sub-component, and a directed edge is drawn from one vertex presenting a component object to another vertex representing one of its components. We call it as a component dependency tree for a given compound object. Depth of a sub-component in a dependency tree is the length of the directed path from the root to the sub-component.

In this paper, we developed methods to determine the trust features of an object by studying the trustworthiness of the owners of its components or a sub-component, the owner's reputation, contact history of the owners with the evaluating subject, the correctness of the formula to integrate the components, etc. Based on these trust related characteristics of imported objects, different objects are mapped to different trust zones. Each zone has unique trust features and all members of each zone satisfy these features. This form of information clustering for external objects can help reduce risks associated with uncertainties of imported information and protect internal resources.

The rest of this paper is organized as follows. Section 2 outlines some related work. Section 3 presents methods for calculating trust values, which are crucial trust attributes of an object. Section 4 discusses trust zone in detail and describes the cases for trust zone mapping. Section 5 concludes the paper.

2 Motivations and Related Work

Internet browsers classify downloaded external objects based on their origins. Internet Explorer, for instance, isolates the downloaded websites from native documents in local systems. It includes five predefined zones: Internet, Local Intranet, Trusted Sites, Restricted Sites, and My Computer [6]. Web sites in each zone have different levels of security. In our model, external objects can be mapped in a similar fashion based on their trust characteristics in a loosely-coupled system.

The concept of trust has been a subject of continuous interest in different research areas. Blaze et.al. [17] and Marsh [3] are among the first to formalize trust in computational models. Among the existing trust models, role-based trust management methods [2][15][16] assign each user, internal or external, to a role, which identifies a set of subjects that satisfy the security requirements of the role. It is designed based on trust delegation for external subject classifications. Our model, on the other hand, is developed to assign an object to trust zones and, hence, our trust management mechanism is focused at object level. We argue that trust management at object level is more appropriate for a decentralized system, where no single authority is assumed. Focusing on the security concerns at object level gives a higher level of information assurance since the ultimate goal of information assurance is to maintain the quality of objects. Making decisions solely based on subjects is not always reliable. Sometimes even honest people make unintentional mistakes. We believe trust management at object level is more attractive to ensure information quality within the context of a loosely-coupled system.

3 Computational Trust

An object has inherent attributes and trust attributes. The former describes the inherent characteristics of an object, such as its height, weight, etc. for a specific object. Trust attributes are defined by users to describe the object's trust-related features, such as its trust value for a user, its owner's trust value, its component trust information, etc. Reputation of a subject is also considered as a trust attribute of the subject. The difference between a subject's reputation and its trust value (for an evaluating subject) is that the former is a global consensus and the latter is an individual opinion towards the subject under evaluation (see [18] for an example of reputation models). Trust attributes are identified by system administrators as crucial factors to facilitate trust-based security management and help users evaluate the object's trustworthiness. We first define subject trust and object trust and introduce methods for their calculations. These are two essential attributes in trust zone mapping. Next, we give an algorithm for subject trust calculation.

3.1 Subject trust

Algorithm: All-pair shortest path discovery for subject trust computation

1. For every $e \in E(G)$, convert the "distance" as indicated by the weight of e from t to t' such that $t'=1-t$. This generates a new trust network, G', such that V(G) = V(G') and E(G) = E(G') but each $e \in E(G')$ has new weights as t';
2. Perform all pair shorted path algorithm on G' and generate a path P = $\{S_i, S_{i+1}, \ldots, S_j\}$ for every pair of nodes S_i and S_j.
3. For every pair of subjects (S_i, S_j), the trust value of S_j for S_i is calculated as

$$T_{ij} = Ts_{i},s_{i+1} * Ts_{i+1},s_{i+2}, \ldots, *Ts_{j-1}, s_j$$

   where $(S_k, S_{k+1}) \in E(G)$ and $(S_k, S_{k+1}) \in P$ for $j\text{-}1 \geq k \geq i$; Ts_k,s_{k+1} is the trust value of S_{k+1} for S_k as indicated in G, i.e., edge value of (S_k, S_{k+1}) and $j\text{-}1 \geq k \geq i$; R(k) is the reputation value of subject k and $j \geq k \geq i$.

Subject trust is a one-way relationship indicating how much a subject trusts another. This trust describes a general confidence of expectation that the trustor has on the trustee. Subject trust is reflexive, transitive (in a discounting manner) but not symmetric. A trust network represents only direct trust of a subject on its neighbors. But indirect trust can be calculated based on the principle of trust transitivity (see[4][5]). Based on the maximum aggregation principle, the "all pair shortest path" algorithm is applied to calculate the indirect trust values between any pair of subjects based on a trust network after some modifications on the value of each edge. The goal of this algorithm is to find a path between any pair of nodes with minimum accumulated *t'* values.

3.2 Object Trust

Object trust refers to the degree to which a subject evaluates the trustworthiness of an object within a certain context. Trust value of an object can be calculated in two ways. One is based on direct experiences obtained from a user's study of the object, its components, and combination functions. The trust value of the object obtained in this way is called the object's *primary trust value* for the user. The other method is related to a user's secondary experiences, which referred to indirect study of the object and its components. The trust value calculated in this way is called *secondary trust value* of the object. Primary trust and secondary trust values of an object can be used separately by a user or can be combined to an overall trust value of the object (for the user).

For an evaluating subject, S, the secondary trustworthiness of an object, O, can be calculated as the mathematical product of the trust level of the object for its owner, S', the trust level of S' for S, as well as a context adjusting parameter. The trust level of S' for S is context independent, which is similar to general or basic trust as defined in [3]. The context adjusting parameter reflects the degree of belief of S about the ability of S' evaluation of O in a specified context. For instance, the context adjusting parameter for the trust value of a mathematical function is the evaluator's belief of the domain knowledge of the subject, which provides the function.

In order to calculate the primary trust value of an object, an evaluating subject studies the component information of the object, i.e., how it has been integrated, which set of combination functions have been used to calculate the object, whether the owners of the components are trustworthy, what those owners' reputations are, etc. Let the set $\{C_1, C_2, \ldots, C_n\}$ be a set of components of the compound object O. $\{S_1, S_2, \ldots, S_n\}$ are owners of these components respectively. The trust value of O for an evaluating subject, S, is represented as $T_{s,o}$, which can be calculated by the following formula:

$$Ts,o = \Gamma s,o\ (F,\ Ts,s_1 * Ts_1,c_1,\ Ts,s_2 * Ts_2,c_2,\ \ldots,\ Ts,s_n * Ts_n,c_n)$$

where $\Gamma s,o$ is a trust function of *o'* based on its component trusts; F is the combination function used to form O from its components; Ts_i,c_i is the trust value of c_i for its owner S_i and Ts,s_i is the trust value of S_i for *S,* where $0<i\leq n$. A trust function answers such question as "given the trust values of all components and the combination function for an object, how much should the object be trusted?" Developing a general format for a trust function is domain and user dependent. We give a simple example to illustrate the idea (for more information, see [1]). For a weighted average data function $F = (w_1 * C_1)\ \theta\ (w_2 * C_2)\ \theta\ \ldots\ \theta\ (w_n * C_n)$, the corresponding trust function is

$$\Gamma s,o\ (F,\ Ts,s_1 * Ts_1,c_1,\ Ts,s_2 * Ts_2,c_2, \ldots,\ Ts,s_n * Ts_n,c_n) = w_1 * Ts,s_1 * Ts_1,c_1 + w2 * Ts,s_2 * Ts_2,c_2 + \ldots + w_n * Ts,s_n * Ts_n,c_n)$$

where $w_1, w_2, \ldots, w_n$ are real numbers in the range [0, 1] and they add up to 1; the symbol θ represents binary operators such as addition, subtraction, multiplication, division, union, join, intersection, etc. Intuitively, for the data formula with the format as weighted average, the trust weights assigned to each component is the same as the contribution of that component to the compound object's data value as expressed by that data function.

4 Trust Zones

In order to map external objects into different trust zones, a user's system defines testing conditions based on each individual trust attributes. Each condition is used to check if the value of an attribute for an object is satisfied with user's expectation for that trust attribute. We give some examples of trust attributes and their testing conditions here. *The secondary trust value* is a trust attribute for an object. The corresponding testing condition evaluates if a given object's secondary trust value is greater than a threshold. Consider another attribute called *Good object* that indicates if an object is publicly known as a good object. The corresponding testing condition checks if a given object is a member of the system-maintained good object list. Consider another trust attribute called *Security carrying proof*, which indicates if an object has been certified by some authorities to be free of common vulnerabilities. The testing condition for this trust attribute checks if such a valid proof can be presented and the integrity of the proof is not compromised.

Some trust attributes are considered as positive in the sense that users want to see higher values for them. For instance, a positive trust attribute could be the trust value of an object, or the reputation of the owner of an object. In contrast, some attributes are considered negative and users want objects to have lower values for those attributes. Those negative attributes represent unfavorable features of objects as viewed by users and they are identified in order for the system to define criteria to limit objects with these "negative" features to an acceptable degree. Based on these two types of attributes, the corresponding testing conditions are defined.

Trust attributes can be organized into dimensions. Each dimension consists of a set of trust attributes, which describes one aspect of the trust features of an object as viewed by users. For instance, a dimension of trust attributes for an object is based on the owners of the objects and their sub-components. Examples of trust attributes in this dimension include the trust value (for a user) of an object's owner and/or the owners of the object's sub-components, the reputation of the object's owner and/or the owners of the object's sub-components, the contact history of the object's owner with the evaluating user. Another dimension for the object is related to the trust features of the object itself as well as its sub-components, such as its secondary or primary trust values (for the user), the membership of the object and/or its sub-components to a public known good object list, trustworthiness of the combination function, etc. A third dimension is based on the object's security features such as security-proof-carrying code issued by an authority to demonstrate that the object program is free of malicious code, software security level is classified by some standard agents, etc.

It is usually too restrictive to require that an external object satisfy the entire trust attribute testing conditions. Rather, users may allow an external object to satisfy a subset of trust attribute conditions as defined for the corresponding trust dimension. In this sense, trust attributes within one dimension are “replaceable”, meaning as long as an external object satisfies a minimum number of test attribute testing conditions within that dimension, the user believes that the object satisfies the security requirement as specified by the trust attribute dimension. An analogous example is the curriculum developed for a graduate program, which specifies that a certain number of core courses must be successfully completed by a degree candidate in order for the student to be considered to satisfy the requirements of core knowledge. We have defined threshold operators to allow users to specify a subset of testing conditions that an object must satisfy in order to be selected.

Trust zone mapping policies are applied to assign an object to one or more trust zones according to its values for the pre-identified trust attribute testing conditions**.** A *trust zone* consists of a set of objects such that every object satisfies the logical combination of trust attribute testing conditions. A set of trust zone mapping polices can be represented in BNF format as shown in Figure 1.

```
trustZonePolicy ::= statement | statementSet
statementSet ::= statement statementSet | ε
statement ::= zoneName ← terms
zoneName ::= STRING
terms ::= term op1 terms | term | ε
term ::= op2(conditionList)
op1 ::= AND | OR
op2 ::= $\Theta_j$ | $\Omega_i$
conditionList ::= condition COMMA conditionList | condition | ε
condition ::= function(attribute values)
```

Fig. 1. Formal Representation for Trust Zone Mapping Policy Syntax

A trust zone is specifically designed to fit the needs of a set of internal applications with similar security focus. The logical combination operators used to connect trust attribute testing conditions include AND, OR, and threshold-selection operators.

Two types of threshold-selection operators, namely, lower-bound and upper-bound threshold-selection operators are used in trust zone mapping policies. These operators represent different forms of compositions of trust related information. The semantic meaning of a lower-bound threshold-selection operator, denoted as $\Theta_i(C_1, C_2, \ldots, C_m)$, is to select any object, which satisfies at least *i* out of *m* trust attribute testing conditions, $A_1, A_2, \ldots, A_m$, where $i \leq m$. In contrast, the semantic meaning of an upper-bound threshold-selection operator, denoted as $\mathbf{\Omega}_j(C_1, C_2, \ldots, C_n)$, is to select any object, which satisfies no more than *j* out of *n* test attribute testing conditions, $C_1, C_2, \ldots, C_n$, where $j \leq n$. Hence, if trust zone *t* is defined as the set of objects, which satisfy 4 of trust attribute testing conditions $C_1, C_2, \ldots, C_6$ and no more than 2 trust attribute testing conditions $C'_1, C'_2, \ldots, C'_7$, then the mapping policy for trust zone *t* can be expressed as

$$\textit{Trust Zone } t \leftarrow \Theta_4(C_1, C_2, \ldots, C_6) \text{ AND } \Omega_2 (C'_1, C'_2, \ldots, C'_7)$$

An AND-OR graph-like data structure, called trust zone mapping graph (see Figure 2), is used to visually represent how a trust zone is constructed based on a group of attribute-based sets. The root is labeled with the identifier of the given trust zone. Each of the leaf nodes represents an attribute-based set. Each node except for the root and the leaf is either an AND, OR, Θ_j, or Ω_i node.

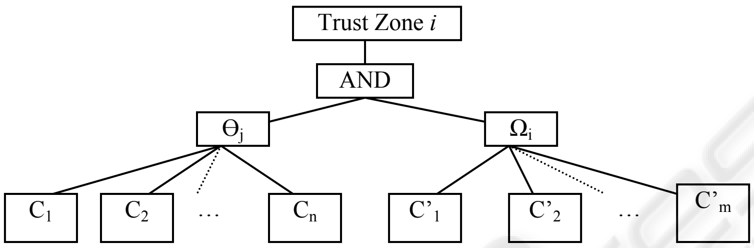

Fig. 2. A Trust Zone Mapping Graph

Specifying trust attribute dimension allows sub-set of attribute testing conditions within different trust dimensions to be satisfied by one external object in order to be mapped to a trust zone. Our defined lower-bound threshold operator provides implementation of this idea. By using both the upper bound and lower bound threshold operators together, users of some internal applications can use trust attributes with different weights according to their security needs. For instance, they can specify the critical trust attributes in certain dimensions as at least as good as some pre-defined conditions while requiring other less important trust negative attributes as no worse than some pre-defined conditions.

As we have mentioned before, an object is a generic term and it can refer to both passive data objects and active executable programs. Mapping external objects into trust zones serves two purposes: limit access rights of external executable programs to local resources and guide internal applications to use external objects such as documents, database items, or other passive knowledge. We discuss the two situations below.

Case 1: Information assurance example: Assume that we have a set of internal applications S_1, which need to use external objects for calculation and information processing. For the applications in S_1, the accuracy of the objects is very important. The users of S_1 focus on the trust features of the owners of external objects and their sub-components. Regarding the security features of those external objects, since they are not supposed to be run as executable programs, the users of applications in S_1 only specify some minimum conditions and as long as an external object does not have negative security features worse than those, it can satisfy the needs of the applications in S_1. Based on this specified situation, the following trust attribute testing conditions are defined:

C_1: all of the owners of an object and its sub-components (up to the depth of 3 in the version dependency tree) have trust values (for the users of S_1) greater than 0.8;

C_2: all of the owners of an object and its sub-components (up to the depth of 3 in the version dependency tree) have reputation values (for the users of S_1) greater than 0.7;

C_3: more than half of the owners of an object and its sub-components (up to the depth of 3 in the version dependency tree) had been in contact with the users of S_1 and the performances of those objects' owners were satisfactory;

C_4: the owner of the object is listed as a bad subject by some authority agencies;

C_1': an object and its sub-components (up to the depth of 3 in the version dependency tree) have trust values (for the users of S_1) greater than 0.8;

C_2': the combination function used to form an external object should be reasonable as viewed by at least two domain experts;

C_3': no sub-component of the given object is in the bad object list kept by the system;

C_1'': the given object under evaluation has no security-proof carrying code;

C_2'': sensitivity classification of the given object is high.

Based on the above testing conditions, trust zone 1 is defined as below:

$$\textit{Trust zone 1} \leftarrow \Theta_2(C_1, C_2, C_3) \text{ AND } \Theta_2(C_1', C_2', C_3') \text{ AND } \Omega_0(C_4) \text{ AND } \Omega_1(C_1'', C_2'')$$

Any object, which can be mapped to trust zone 1, must satisfy at least two conditions of C_1, C_2, and C_3 as well as 2 conditions of C'_1, C'_2, and C'_3. At the same time, the object mapped to trust zone 1 must not satisfy condition C_4 and, in a worse case, satisfy no more than 1 of conditions C''_1 and C''_2. We use the term $\Omega_0(C_4)$ to indicate that no external object can be selected if it is tested by condition C_4 as true. Besides, all external objects mapped to trust zone 1 can be used by internal applications in S_1, namely

$$\textit{Internal applications} \in S_1 \leftarrow \textit{Trust zone 1}$$

The term $\Theta_2(C_1, C_2, C_3)$ only specifies that an object should satisfy at least two of three testing conditions, C_1, C_2,and C_3. But it does not specify which particular conditions an object must satisfy in order to make the term as true. In order to specify that C_1 must be satisfied by an object within the first dimension and C_2'' must be satisfied within the second dimension, a restricted lower-bound threshold operator is defined as below:

$$\textit{Trust zone 1} \leftarrow$$
$$\Theta_{2,\,C1}(A_1, A_2, A_3) \text{ AND } \Theta_{2,\,C2''}(C_1', C_2', C_3') \text{ AND } \Omega_0(C_4) \text{ AND } \Omega_1(C_1'', C_2'')$$

In general, the restricted lower-bound threshold operator $\Theta_{j,\,Ck}(C_1, C_2, \ldots, C_n)$ indicates that an object satisfies this operator if it satisfies at least *j* out of *n* trust attribute testing conditions, $C_{1\text{-}n}$, and C_k must be one of the conditions satisfied.

Case 2: Access control example: By defining trust zones, we can limit access rights of external objects such as executable programs to get access to local resources. Formally, this form of access control is essentially a mapping between $R \times O \leftarrow Z$, where Z is a set of trust zones, R is a set of access rights, e.g., read, write, and O is a set of local protected resources. Each member of a trust zone can access local resources as defined for all the objects in the trust zone as a whole. If an object is a member of multiple trust zones, then its access rights is the union of the rights for all the trust zones. If two conflict rules exist, one may supersede another based on a pre-defined policy. Below, we provide a concrete example to illustrate the mapping of executable object programs to trust zones as well as trust zones to a set of access rights.

Suppose we have a set of internal applications, S_2, which would call external objects as sub-routines and these called external programs need to get access to local resources in order to be run correctly. In this case, the trust attributes along the security dimension is more important. The users of applications in S_2 identify the following testing conditions:

C_1: majority of the owners of an external object and its sub-components (up to the depth of 3 in the version dependency tree) are new to the users of S_2;

C_2: no more than one third of the owners of an object and its sub-components (up to the depth of 3 in the version dependency tree) have reputation values (for the users of internal applications in S_2) less than 0.4;

C_1'': the external object program successfully passed intrusion detection test monitored by a system authority agent;

C_2'': the external object carries security-proof code and the security assurance is verified by an internal security agent;

C_3'': the external object program is without any known software bugs;

C_4'': the external object program neither uses any routine to make network connections nor carries any sniper programs as verified by internal security agents;

C_5'': the external object program does not pass the malicious code detection.

Based on the above trust attribute testing conditions, trust zone 2 is defined as:

$$\textit{Trust zone 2} \leftarrow \Theta_2(C_1'', C_2'', C_3'', C_4'') \text{ AND } \Omega_0(C_5'') \text{ AND } \Omega_1(C_1, C_2)$$

As an example, consider that in order to grant minimum system resource requirements to external objects to be executed successfully, the internal applications in S_2 allow external objects, which satisfy the trust condition as defined for Trust zone 2, to access two public drives, D_1 and D_3, with full rights (Read and Write), access drive D_2 with only Read right, and no access rights on other storages. The mapping between Trust zone 2 and the specified access rights can be expressed by the following policy.

$$\{[D_1, R, W], [D_2, R], [D_3, R, W]\} \leftarrow \textit{Trust zone 2}$$

5 Conclusion

In this paper, we have presented a model for information assurance by mapping external objects to appropriate trust zones. This mapping serves two purposes: limit access rights of external executable programs to internal resources and guide internal applications to use trusted external information. We have defined two powerful threshold selection operators to check and verify if an external object satisfies the trust-based security conditions as specified by each trust zone. Formulas are provided to calculate primary and secondary trust values for an object in evaluation. We have also presented a simple algorithm to calculate indirect trust based on a given trust network by applying the well-known “all pair shortest path” algorithm.

Acknowledgment

We are thankful to Dr. Robert L. Herklotz for his support, which made this work possible.

References

1. Y. Zuo and B. Panda, “Component Based Trust Management in the Context of a Virtual Organization”, The 20th ACM Symposium on Applied Computing, NM, USA, March 2005
2. J. Bacon, K. Moody, W. Yan, “A Mode of OASIS Role-based Access Control and Its Support for Active Security”, ACM Transactions on Information and System Security, p. 492-540
3. S. P. Marsh, “Formalising Trust as a Computational Concept”, Ph.D. Dissertation, University of Stirling, 1994
4. M. Richardson, R. Agrawal, P. Domingos, “Trust Management for the Semantic Web”, the Second International Semantic Web Conference,. Sanibel Island, FL, USA, 2003.
5. R. Bellman, M. Giertz, “On the Analytic Formalism of the Theory of Fuzzy Sets”, Information Sciences, 5, p. 149-159, 1973
6. msdn.microsoft.com/library/default.asp?url=/workshop/security/szone/overview/overview.asp
7. E. Lupu and M. Sloman, “Reconciling Role Based Management and Role Based Access Control”, Second Role Based Control Workshop, Virginia, USA, 1997
8. L. Bauer, M. Schneider, and E. Felten, “A General and Flexible Access-Control System for the Web”, The 11th USENIX Security Symposium, p. 93-108, 2002
9. M. Abadi, M. Burrows, B. Lampson, and G. Plotkin, “A Calculus for Access Control in Distributed Systems”, ACM Transactions on Programming Languages and Systems, p.706-734, October, 1993
10. E. Bertino, B. Catania, E. Ferrari, and P. Perlasca, “A Logical Framework for Reasoning about Access Control Models”, ACM Transactions on Information and System Security, p.71-127, February 2003
11. D. F. Brewer and J. Nash, “The Chinese Wall Security Policy”, The IEEE Symposium on Security and Privacy, 1989
12. Dorothy E. Denning, “A lattice Model of Secure Information Flow”, Communications of the ACM, 19(5): p. 236-243, 1976
13. D. Bell and L. LaPadula, “The Bell-LaPadula Model”, Journal of Computer Security, p.303-339, 1997
14. Thomas A. Berson and Teresa F. Lunt, “Multilevel Security for Knowledge-Based Systems”, In proceedings of the 1987 IEEE Symposium on Privacy and Security, p. 235-242, 1987
15. Ninghui Li and John C. Mitchell, “*RT*: A Role-based Trust-management Framework”, T*he Third DARPA Information Survivability Conference and Exposition*, Washington, D.C., April 2003. IEEE Computer Society Press, Los Alamitos, CA, USA, p. 201-212
16. Ninghui Li, William H. Winsborough, and John C. Mitchell, “Distributed Credential Chain Discovery in Trust Management”, *Journal of Computer Security*, 11(1): p. 35-86, February 2003
17. M. Blaz, J. Feigenbaum and J. Lacy., “Decentralized Trust Management”, IEEE Conference, Anguilla, British West Inides, 1998

18. S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, “The Eigentrust Algorithm for Reputation Management in P2P Networks”, in Proceedings of the twelfth International Conference on World Wide Web, ACM Press, p. 640-651, 2003

SciTePress
Science and Technology Publications