Research Journal of Applied Sciences, Engineering and Technology 5(19): 4664-4670, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: September 22, 2012 Accepted: November 23, 2012 Published: May 10, 2013

Formal Specification of Security Properties using Z Notation

[1]Shafiq Hussain, [1]Peter Dunne and [2]Ghulam Rasool
[1]Department of Computing, Engineering and Technology, University of Sunderland, UK
[2]Department of Computer Science, COMSATS Institute of IT, Lahore, Pakistan

Abstract: Software security is a challenging issue for distributed and open systems. Despite the importance of external protections of software systems, internal security has significant impact on the overall security of the software systems. In this study, internal security issues of software systems are addressed. Internal security of software systems is defined in terms of security properties: authentication, authorization, confidentiality, integrity, non-repudiation and resource availability. Internal security of software systems largely depends on the integration of these security properties into the software systems. A precise and unambiguous representation of these security properties is crucial for any successful secure system. Majority of the existing models are based on informal or semi-formal approaches to model these security properties. But no model is based on formal methods. Therefore, in this study, a formal specification of these security properties is presented in Z notation because formal methods are the only way to specify system properties unambiguously, completely and precisely. The resulting models are then analyzed by using Z/EVES theorem prover. The formal specifications of these security properties are analyzed only for syntax checking, type checking and automatic proofs of models.

Keywords: Availability, authentication, authorization, confidentiality, formal methods, integrity, non-repudiation

INTRODUCTION

Software security is a challenging and most demanding aspect of current software systems, particularly for web based systems. The emergence of sophisticated and complex distributed systems has made software security as an integral component of secure systems. Software security depends not only on the external protective measures but it also depends on the internal security aspects of software systems themselves. The internal security issues of software systems are more challenging and complicated as compared to external protections. This research is focused on the internal security properties of software systems. For internal security of software systems, security properties: confidentiality, integrity, non-repudiation, availability, authentication and authorization must be integrated into the systems. In authentication unique identification of a user or entity is established. Any user who wants to get access the system must have credentials to access the system by system must have authentication policy predefined for that user. Authorization security property enables an authenticated user to access a particular resource in the system. Users, who want access, must have access rights for that resource. Confidentiality is the property which allows an authorized user to read the contents of a confidential resource. Confidentiality defines the read access policy of the system. Only those users can read confidential data who has rights to read that data. Integrity security policy allows an authorized user to write on a resource. Integrity policy defines write access for users of the system. Only those users, who have write permissions, can write data on a resource that has permissions for writing. Non-repudiation security property ensures auditing and logging mechanisms of software systems. Non-repudiation requires that every action by all the users of the system must be properly recorded and logged in the system. Availability security property enforces the availability of system resources to legitimate users. Every user of the system must get sufficient resources available in the system to complete his/her tasks. These security properties have been specified formally by using Z notation, a formal specification and modeling language and Z/EVES theorem prover has been used for analyzing the resulting models.

LITRATURE REVIEW

With the speedy developments in the area of distributed computing and web based systems, the needs to resolve security issues become critical for smooth and successful operations of software systems (Subashini and Kavitha, 2011). For web based distributed systems, there is a need of secure communication mechanism because all communication is dependent on message passing in these systems (Savola and Abie, 2010). Security properties are the mechanisms used to stop an attack on software systems

Corresponding Author: Shafiq Hussain, Department of Computing, Engineering and Technology, University of Sunderland, UK, Tel.: 0044 7466638765

and are built into the software systems (Bishop, 2003a). The security properties are required to be specified precisely at the requirements engineering phase of software development (Wolter *et al*., 2009). The security properties such as authentication, authorization, confidentiality, integrity, non-repudiation and availability are the main security properties and must be built into the software systems (Bertino *et al*., 2009; Menzel and Meinel, 2009). Authentication and authorization are the basic controls for software systems and all other security mechanisms such as confidentiality are dependent on these two properties (Jie *et al*., 2011; Welch *et al*., 2003; Thompson and Montague, 2011; Sandhu and Samarati, 1994). Authentication property makes it ensure for a user to be identified the system as a legitimate user of the system (López *et al*., 2007). Authorization is built on the authentication property and determines the access or denial of a user to access particular resources in the system (Joshi *et al*., 2005; Hu and Ahn, 2011; Ahn and Sandhu, 2000). Strict authentication and authorization mechanisms enable an enhanced and more secure environment for execution of business on the systems particularly electronic commerce and banking systems (Ferraiolo *et al*., 2001; Kolovski *et al*., 2007). Confidentiality property determines the read access to data in the system and grants read access to only those users who have read rights for that data (Bishop, 2003b). Integrity property determines the write access to data in the software system and grants write access to users who have write access for that data (Zhou *et al*., 2010; Xu and Liu, 2009; Thompson and Montague, 2011). Non-repudiation property ensures the auditing and logging mechanism in the software systems. It makes it mandatory to record any action performed by authorized users in the system (Feng *et al*., 2010; Onieva *et al*., 2009). Availability property manages the readiness of system resources for legitimate users. It ensures system resource must be available to legal users when demanded and no deadlock occurs (Milanovic and Milic, 2011; Bishop, 2003a; Arci *et al*., 2003). Formal methods are mathematics based techniques for developing mathematical models of software systems and enables to analyze software designs to remove flaws before implementation. Hence, reducing huge effort spent at the testing phase (Jacky, 1996; Xia and Tang, 2008; Zafar and Alhumaidan, 2011; Sidek and Ahmad, 2009). Formal specification helps us to specify, analyze and design computer systems ranging from commercial systems such as banking systems to critical systems such as air traffic control systems (Zafar, 2009; Coronato and De Pietro, 2011; Hussain *et al*., 2011). Software security is important area for the application of formal methods because security properties are impossible for exhaustive testing. Formal methods have been used successfully in many security systems such as SSAI and Correctness by Construction (Gilliam *et al*., 2001; Dimitrov, 2011; Woodcock *et al*., 2009). The use of formal methods for software systems helps to analyze system properties by using powerful tools to produce formal proofs and making possible the production of more reliable and secure software systems (Almeida *et al*., 2011).

FORMAL METHODS

Formal methods are mathematics based tools and techniques used for developing mathematical models and designs of software systems. Formal methods can be applied at all the stages of software development life cycle from specification to implementation. Formal methods help to write formal specifications from the requirement documents in a precise way. In this way, common specification errors such as incompleteness and ambiguities can be removed. These specification errors are occurred due to the ambiguous nature of design methodologies being used for developing software applications. These design methodologies are based on informal and semi-formal techniques. The solution lies with the use of formal methods. By using formal methods, software systems are designed without ambiguities in formal specification language such as Z. The resulting models are then analyzed by using formal method tools such as theorem provers and model checkers. These models can be verified and validated by automatic formal methods tools. Formal proofs of desired properties can also be generated by using theorem provers such as Z/EVES and Isabelle. There are some areas of computer science such as security where exhaustive testing is not possible because security properties are very hard to prove before implementation. Therefore, formal methods can be used to validate security properties before implementation. Theorem provers are normally used to generate formal proofs of software systems while model checkers are used to validate models and remove faults such as deadlocks. The use of formal method specification languages, model checkers and theorem provers depends on the nature of the application being developed. Formal methods can be categorized broadly into three types: formal specification languages such as Z, VDM-SL, VDM++, Object-Z, B-Method, Event-B and Alloy etc., model checkers such as SPIN, Alloy and Pro-B, theorem provers such as Z/EVES, Isabelle, Coq and RODIN. In this study, security properties have been specified in Z notation. Z/EVES theorem prover has been used for analysis of models produced in Z. Z is a model based language and built on set theory. Models in Z are a collection of state variables, invariants on state variables, set of operations and a set of pre and post conditions.

INFORMAL MODEL

In this study, six security properties have been considered. These security properties include

authentication, authorization, confidentiality, integrity, non-repudiation and availability. These security properties are defined informally as follows:

Authentication: Authentication is the mechanism in which unique identification of a user of the software system is determined. Authentication process ensures a login policy stored in the software system. Any user who want to login into the system must provide unique credentials such passwords defined in the login policy for that user.

Authorization: Authorization property grants or denies access to system resources for a particular user. If the user has access rights for intended resources, he or she is granted access to that resource otherwise, no access is granted. Access is granted to user depending upon the policy already defined for the users according to the roles of the users.

Confidentiality: Confidentiality property determines the read access to data in the system and grants read access to only those users who have read rights for that data. Confidential data can only be read by the users who are authorized to read that data. Some additional credentials are required to read confidential data.

Integrity: Integrity property determines the write access to data in the software system and grants write access to users who have write access for that data. Data can only be written by the users who are authorized for it. There must be a strict writing policy in the system to ensure integrity of data. The common issue of data integrity such as redundancy, inconsistency etc. must be considered.

Non-repudiation: Non-repudiation property ensures the auditing and logging mechanism in the software systems. It makes it mandatory to record any action performed by authorized users in the system. This policy sets limits on the legitimate users of the system and ensures that every action or activity of these users must be recorded. It helps for auditing at the later stages.

Availability of resources: Availability property manages the readiness of system resources for legitimate users. It ensures that system resource must be available to legal users when demanded and no deadlock occurs. The resource allocation algorithm should be defined in such as way that at least sufficient system resources remain available to legitimate users all the time. System resources must not be consumed by existing users of the system.

FORMAL MODEL

The formal models of security properties have been developed by using Z notation, a formal specification and modeling language. This formal model consists of static model and dynamic model. In static model, state variables are defined formally along with invariants on these variables. The invariants impose constraints on data hold by these variables. Furthermore, these invariants should remain true for all the time for smooth functioning of the system. In dynamic model, operations on the system are defined along with a set of pre and post conditions.

Static model: The static model of security properties starts with the definition of basic types for users, credentials, resources and actions on those resources. In the following specification *USER* represents set of users, *CREDENTIAL* represents set of credentials the user may have, *RESOURCE* represents set of resources in the system and *ACTION* represents the set of action allowed on system resources in the system:

[*USER*, *CREDENTLAL*, *RESOURCE*, *ACTION*]

The following specification describes global types in the system. The global types are accessible in all the schemas on the system. In this model of security properties, only one global type *OPERATIONS* has been defined. This global type is used in the definition of confidentiality and integrity properties:

OPERTIONS::=*READ*|*WRITE*|*APPEND*|*EXECUTE*

This schema describes the state of the system having state variables and invariants on them.

Access Control System
Users: ℙ user
Resources: ℙ resource
Action: ℙ action
registered_ users: User→ ℙ CREDENTIAL
user_resources: USER→ ℙ RESOURCE
resource_actions: RESOURCE→ ℙ ACTION
user_actions: User → ℙ ACTION
authenticated_ users: User → ℙ CREDENTIAL
authorized_ users: User → ℙ CREDENTIAL
log: USER → ℙ ACTION
owner: USER → ℙ RESOURCE
dom registered_users ⊆ users
dom users_ resources ⊆ users
∀u: USER| u ∈ dom users_resources. User_resources u ⊆ resources
dom user_action ⊆ users
∀u: USER| u ∈ dom user_action . user_actions u ⊆actions
dom resource-action ⊆ resources
∀r: RESOURCE| r ∈ dom resource_action. resource_action r ⊆ actions
dom authenticated_users ⊆ users
dom authorized_users ⊆ users
dom log ⊆ users
∀u: USERS| u ∈ dom log. log u ⊆ actions

dom owner ⊆ users
∀u: USER| u ∈ dom owner. owner u ⊆ resources

This is the initialization schema and which ensures that at least one state of the system exists and system will work for at least one time.

init = ø
Access Control System
users = ø
resources = ø
actions
dom registered_users = ø
dom users_resources = ø
dom resource_ action = ø
dom authenticated- users = ø
dom authorized_users = ø
dom log = ø
dom owner = ø

Dynamic model: The following is the dynamic model of the system in which basic operations for the system have been defined as follows:

Basic operations: This operation adds users into the system. The pre-condition says that users to be added must not already be added into the system.

Add_ Users
ΔAccess Control System
urs? : ℙ USER
∀u: USER|u ∈ urs:. U ∉ users
users` = users ∪ urs?

This operation adds resources into the system. This operation requires that resources to be added must not already be added into the system.

Add_Resources
ΔAccess ControlSystem
rcs?: ℙ RESOURCE
∀r: R ESOURCE|r ∈ rcs?. r ∉ resources
resources` = resources ∪ rcs?

This operation adds actions into the system and ensures only allowed actions on the system resources. The pre-condition says that actions to be added must not already be added in the system.

Add_Actions
ΔAccess Control System
Acns?: ℙ ACTION
∀a: ACTION| a ∈ acns: . a ∉ actions
Actions' = actions ∪ acns?

This operation adds registered users in the system. Only those users who have proper credentials can be registered in the system. They must be recognized in the system and must not already be registered in the system.

Add_Registered_ User
ΔAccess Control System
us?: USER
crdls?? ℙ CEDENTIAL
u? ∈ users
u? ∉ dom registered_users
registered_users' = rsgistered_users ∪ {(u? ↔crdls?)

This operation adds actions to a user. Only those actions are added to a user for which he or she is authorized in the system. The user must be a registered user and actions must be recognized in the system.

Add_Actions_to_User
ΔAccess Control System
u?: USER
acns:" ℙ ACTION
u? ∈ dom registered_users
acns? ⊆ Actions
user_actions' = user_actions ∪ {(u? ↔acns?)}

This operation adds actions to system resources. Both actions and resource must be recognized in the system.

Add_Actions_to_Resource
ΔAccess Control System
r?: RESOURCE
acns?: ℙ ACTION
r? ∈ Resources
acns? ⊆ actions
resource_actions`=resource_actions ∪ {(u? ↔acns?)}

This operation adds resources to users depending upon the requirements. The user must be a registered user and resource must be recognized in the system.

Add_ Resource_to_User
ΔAccess Control System
U?: USER
acs?: ℙ RESOURCE
u? ∈ dom resistered_users
acs? ⊆ resource
user_resource' = user_resource ∪ {(u? ↔acs?)}

Formal model of security properties: Formal model of security properties developed in Z notation is presented in this subsection. All the six security properties: authentication, authorization, confidentiality, integrity, non-repudiation and availability of resources are specified in the following paragraphs.

The first schema below represents authentication security property. A user who wants to login into the system, enters user id and credentials. If the user supplied user id and credential are matched with those already stored in the system, the user is allowed to login into the system.

Authentication

ΔAccess Control System
r?: USER
crdls?: ℙ CREDENTIAL
if u? ∈ dom resistered_users ∧ crdls? = registered_users u?
then aauthenticated_users' = authenticated_users ∪ {(u? crdls?)}
els authenticated_users' = authenticated_users

This schema represents the authorization security property. This schema determines the access to system resources for a particular user. A user enters user id, credentials, actions he or she wants to perform, resource on which actions are to be performed to the system. This schema checks whether the user is registered or not. If the user is a registered user then it checks whether the required actions are allowed on the requested resource and required actions are allowed to that user. If yes the user is allowed to access that resource.

Authorization
ΔAccess Control System
u?: USER
crdls?: ℙ CREDENTIAL
acs?: ℙ ACTION
r?: RESOURCE
u? ∈ dom authenticated_users
if acs? ⊆ Resource_action r? ∧ acs? ⊆User_actions u?
then authorized_users' = authorized_users ∪ {(u? ↔ crdls)}
else authorized_users'= authorized_users

This schema describes the formal model of confidentiality. Confidentiality refers to read access to system resources. A user, who wants to read confidential data, enters user id and system checks whether the user is an authenticated user and an authorized user. If the user is authenticated and authorized then system checks the operations defined in the system for that user. If the operation, defined for that user is read operation then read access is granted to that user.

Confidentiality
ΔAccess Control System
u?: USER
op?: OPERTIONS
u? ∈ dom authenticated_users
? ∈ dom authorized_users
if op = READ
then user_actions' = user_action ∪ {(u? ↔ action)}
els user_action' = user_actions

This schema describes the formal model of integrity. Integrity refers to write access to system resources. A user, who wants to write new data, append data to an existing data or modify existing data, enters user id and system checks whether the user is an authenticated user and an authorized user. If the user is authenticated and authorized then system checks the operations defined in the system for that user. If the operation, defined for that user is write operation then write access is granted to that user.

Integrity
ΔAccess Control System
u?: USER
op?: OPERTIONS
u? ∈ dom authenticated_users
u? ∈ dom authorized_users
if op = WRITE ∨ op = APPENDREAD
then user_actions' = user_action ∪ {(u? ↔ action)}
els user_action' = user_actions

This schema describes non-repudiation security property. Non-repudiation ensures the logging of any event or activity performed by legitimate users. For actions to be recorded, a user must be an authenticated user and an authorized user.

Non Repudiation
ΔAccess Control System
u?: USER
acns!: ℙ ACTION
u? ∈ dom authenticated_users
u? ∈ dom authorized_users
acns! = user_action u?
if user_action u? ≠ ø then log' = log ∪ {(u? ↔ acns!)}els log' = log

This schema defines the availability of resources security property. There are predefined and recognized set of resources in the system. Any authenticated and authorized user is allocated a set of resources to perform his or her actions. The free set represents resources which are not currently occupied. To ensure availability, the free set must never be empty.

Availability
Access Control System
Free: ℙ RESOURCE
u?: USER
u? ∈ dom authenticated_users
u? ∈ dom authorized_users
owner u? ≠ ø
free = resources / owner u?
free ∪ owner u? = resources
free ∩ owner u? = ø
free≠ ø

FORMAL ANALYSIS

In this study, formal analysis of models of security properties is done by using Z/EVES theorem prover. This analysis was done for syntax checking, type checking and automatic formal proofs checking of Z

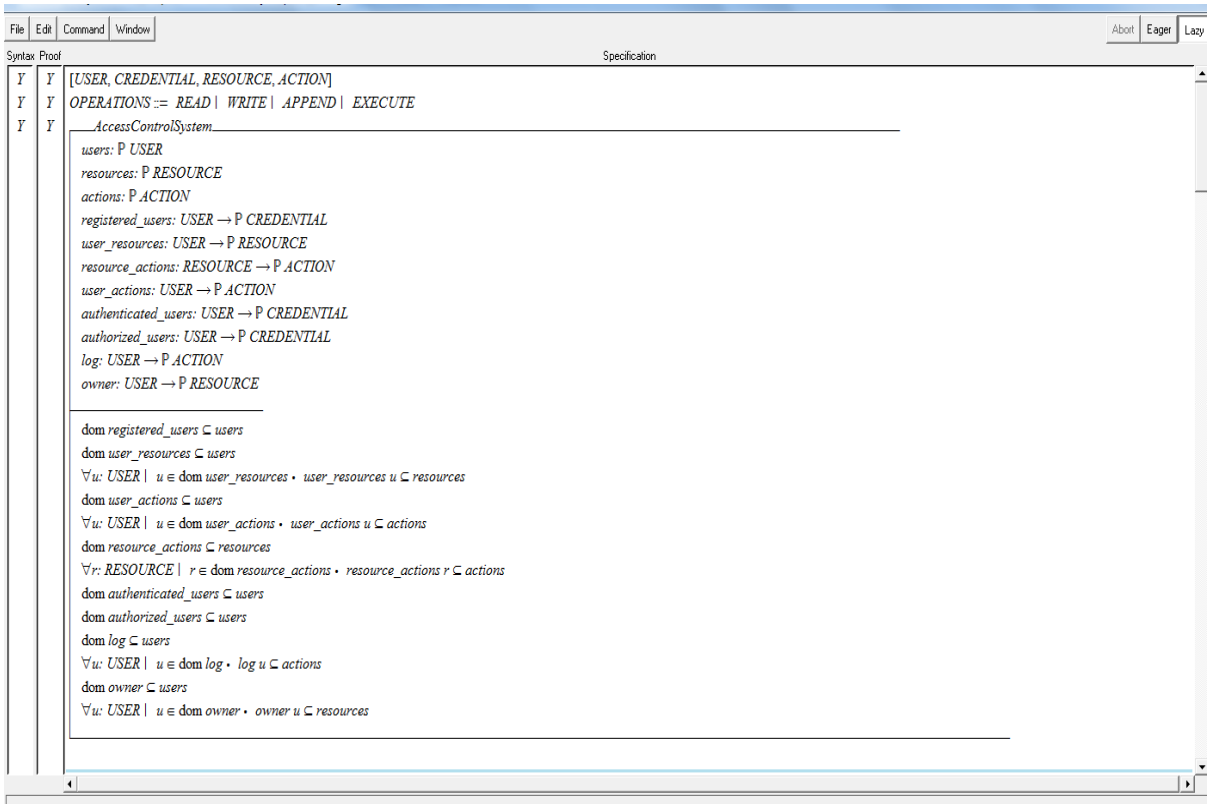

Fig. 1: Formal analysis of security properties

paragraphs. Other types of analysis such as invariants preservations, domain checking, validation of certain properties and formal proofs are not in the scope of this study. For brevity, only one snapshot taken from Z/EVES theorem prover has been shown in Fig. 1. In this snapshot, we have three columns: syntax, proof and main window containing actual specification of properties. Here, "Y" in syntax column shows that the specification in the main window has correct syntax and correct type. The "Y" in proof column shows that the specified paragraphs have no pending proof and all the attached invariants, pre-conditions and post-conditions have been satisfied.

CONCLUSION AND FUTURE WORK

In this study, software security is discussed as external protections and internal mechanisms. Software security is mainly concerned with the internal security of the software system. For internal security, security properties such as authentication, authorization, confidentiality, integrity, non-repudiation and availability of resources has been defined and discussed. A review of relevant literature has been presented. Formal methods have been described very briefly as mechanisms to formally specify security properties. An informal description of security properties is also presented in this study. Then the formal model is developed along with very brief commentary on the formal schemas. At the end, brief analysis of formal models has been presented. In the future, these formal models will be refined to add more details in the models. Analysis of properties such as domain checking, invariants preservation and precondition calculation is also a future task. In the future, an investigation will be performed to generate formal proofs of these security properties to validate the models.

REFERENCES

Ahn, G.J. and R. Sandhu, 2000. Role-based authorization constraints specification. ACM Trans. Inform. Syst. Security, 3(4): 207-226.

Almeida, J.B., M.J. Frade, J.S. Pinto and S.M. de Sousa, 2011. An Overview of Formal Methods Tools and Techniques. Rigorous Software Development, Undergraduate Topics in Computer Science, pp: 15-44.

Arci, D., G. Maier, A. Pattavina, D. Petecchi and M. Tornatore, 2003. Availability models for protection techniques in WDM networks. Proceeding of 4th International Workshop on Design of Reliable Communication Networks (DRCN), pp: 158-166.

Bertino, E., M. Lorenzo, P. Federica and S. Anna, 2009. Security for Web Services and Service-Oriented Architectures. Springer, Heidelberg, pp: 208, ISBN: 354087741X.

Bishop, M., 2003a.Computer Security: Art and Science. Addison-Wesley, Boston, pp: 1084, ISBN: 0201440997.

Bishop, M., 2003b. What is computer security? IEEE Secur. Priv., 1(1): 67-69.

Coronato, A. and G. De Pietro, 2011. Formal specification and verification of ubiquitous and pervasive systems. ACM Trans. Auton. Adapt. Syst., 6(1).

Dimitrov, V., 2011. Relationship specification in Z-notation. Phys. Part. Nucl. Lett., 8(4): 391-394.

Feng, J., C. Yu, K. Wei-Shinn and L. Pu, 2010. Analysis of integrity vulnerabilities and a non-repudiation protocol for cloud data storage platforms. Proceedings of the 39th International Conference on Parallel Processing Workshops. IEEE Computer Society Washington, DC, USA, pp: 251-258.

Ferraiolo, D.F., R. Sandhu, S. Gavrila and R. Kuhn, 2001. Proposed NIST standard for role-based access control. ACM Trans. Inform. Syst. Secur., 4(3): 224-274.

Gilliam, D.P., J.C. Kelly, J.D. Powell and M. Bishop, 2001. Development of a software security assessment instrument to reduce software security risk. Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. IEEE Computer Society Washington, DC, pp: 144-149.

Hu, H. and G.J. Ahn, 2011. Multiparty authorization framework for data sharing in online social networks. Data Appl. Secur. Priv., 25: 29-43.

Hussain, S., H. Erwin and P. Dunne, 2011. Threat modeling using formal methods: A new approach to develop secure web applications. Proceeding of 7th International Conference on Emerging Technologies (ICET), pp: 1-5.

Jacky, J., 1996. The Way of Z: Practical Programming with Formal Methods. Cambridge University Press, Cambridge, pp: 372, ISBN: 0521559766.

Jie, W., J. Arshad, R. Sinnott, P. Townend and Z. Lei, 2011. A review of grid authentication and authorization technologies and support for federated access control. ACM Comp. Surv., 43(2): 1-26.

Joshi, J.B.D., E. Bertino, U. Latif and A. Ghafoor, 2005. A generalized temporal role-based access control model. IEEE Trans. Knowl. Data Eng., 17(1): 4-23.

Kolovski, V., J. Hendler and B. Parsia, 2007. Analyzing web access control policies. Proceedings of the 16th International Conference on World Wide Web (WWW '07). ACM, New York, NY, USA, pp: 677-686.

López, G., A.F. Gomez, R. Marin and O. Canovas, 2007. A network access control approach based on the AAA architecture and authorization attributes. J. Network Comp. Appl., 30(3): 900-919.

Menzel, M. and C. Meinel, 2009. A security meta-model for service-oriented architectures. Proceeding of IEEE International Conference on Services Computing (SCC). Bangalore, India, pp: 251-259.

Milanovic, N. and B. Milic, 2011.Automatic generation of service availability models. IEEE T. Serv. Comp., 4(1): 56-69.

Onieva, J.A., J. Lopez and J. Zhou, 2009. Fundamentals of Non-repudiation. Secure Multi-party Non-repudiation Protocols and Applications: Advances in Information Security. Springer, US, pp: 1-15.

Sandhu, R.S. and P. Samarati, 1994. Access control: Principle and practice. IEEE Commun. Mag., 32(9): 40-48.

Savola, R.M. and H. Abie, 2010. Development of measurable security for a distributed messaging system. Int. J. Adv. Secur., 2(4): 358-380.

Sidek, R.M. and N. Ahmad, 2009. Deriving formal specification using Z notation. Proceeding of IEEE International Conference on Computer Technology and Development (ICCTD), pp: 225-229.

Subashini, S. and V. Kavitha, 2011. A survey on security issues in service delivery models of cloud computing. J. Network Comp. Appl., 34(1): 1-11.

Thompson, J. and P. Montague, 2011. Review of Data Integrity Models in Multi-level Security Environments. Addendum, Retrieved from: http://www.dtic.mil/cgibin/GetTRDoc?AD=ADA542134.

Welch, V., F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski *et al*., 2003. Security for grid services. Proceeding of 12th IEEE International Symposium on High Performance Distributed Computing, pp: 48-57.

Wolter, C., M. Menzel, A. Schaad, P. Miseldine and C. Meinel, 2009. Model-driven business process security requirement specification. J. Syst. Architect., 55(4): 211-223.

Woodcock, J., P.G. Larsen, J. Bicarregui and J. Fitzgerald, 2009. Formal methods: Practice and experience. ACM Comp. Surv., 41(4): 1-36.

Xia, J. and H. Tang, 2008. Formal method for requirement analysis using Z notation. Sci. Technol. Eng., 8(8): 2245-2246.

Xu, Q. and G. Liu, 2009. Configuring clark-wilson integrity model to enforce flexible protection. Proceeding of International Conference on Computational Intelligence and Security (CIS), pp: 15-20.

Zafar, N., 2009. Formal specification and validation of railway network components using Z notation. Software IET, 3(4): 312-320.

Zafar, N.A. and F. Alhumaidan, 2011. Transformation of class diagrams into formal specification. IJCSNS, 11(5): 289-295.

Zhou, Z.Y., Y.P. He and H.L. Liang, 2010. Hybrid mandatory integrity model composed of Biba and Clark-Wilson policy. J. Software, 21(1): 98-106.