Formal Specification and Validation of Authentication and Authorization

Shafiq Hussain[1*], Muhammad Jamil[2], Rana Abu Bakar[3], Muhammad Farhan[4], Muhammad Amin Abid[5], Hafiz Tariq Masood[6]

Abstract: **Authentication and authorization are the major stakeholders of a computer system in terms of security. The collapse of any of these mechanisms may cause the failure of the entire system. There are numerous techniques of authentication that are being implemented in several secure software systems. These procedures include passwords, pass cards, fingerprints, retinal scans, etc. similarly, a large number of practices are available that implement authorization such as access control lists, role-based access, etc. In this paper, a novel approach is presented to model authentication and authorization by using formal methods. Models of software systems are specified in Z notation. The resulting models were verified and validated by using the Z/EVES theorem prover. The adopted approach provides a mechanism to develop secure software systems by using formal methods.**

Keywords: **Authentication, Authorization, Security, Formal Methods, Specification, Validation, Z, Z/EVES**.

INTRODUCTION

Software security is an identical aspect of modern computer systems, especially in the context of web-based applications. For software systems, authentication and authorization are the primary security properties. Other security properties such as confidentiality, integrity, non-repudiation, and availability are also very important and dependent upon strong controls of authentication and authorization. Authentication is the process under which unique identification of a user or entity is established. A person must have proper credentials in order to authenticate into the system. An authenticated user can get access to the system to be able to use the system resources under certain access rights for the required resources. Without authenticating credentials like passwords, keys, access cards, tokens, badges, ID, etc., a user cannot be recognized as an authenticated user and must be refused to gain access to the system. There are three main methods of authentication. The first method consists of keys, badges, IDs, pass cards, and tokens. These are physical objects and a user must show these objects in order to gain access to the system. The second method consists of properties like DNA, fingerprints, voice match, the cadence of typing, retinal scan, vein patterns, etc. The third method comprises passwords, passphrases, etc. Authorization is a process by which an authenticated user is granted access to the system resources. In authorization access policy for a particular user or a group of users is defined. Depending upon the access policy already in the system for a particular user, access is granted to that user. The access is denied if the user has no policy defined in the system to access a particular resource. Various authentication and authorization schemes and mechanisms have been defined and implemented in academia and industry. Very little work has been done for defining these mechanisms of authentication and authorizing using formal methods. Formal methods are best suited for the specification and verification of authentication and authorization. In this paper, a formal model for authentication and authorization has been presented. The Z notation, a formal specification, and modeling language have been used for the specification of the models and the Z/EVES theorem prover has been used for verification and proof of the models. These models are proposed to for the formal specification and validation of authentication and authorization. It is a novel approach that will help to develop secure and more reliable software systems. The structure of paper consists on 7 major sections. Section 1 introduces the paper while section 2 describes the related work. The research methodology is presented in section 3. where we discussed formal methods, and formal specifications overview. In section 4, research implementation is discussed where the formal model is described using Z. In section 6, formal verification and validation are done by using the Z/EVES theorem prover. We also provided future work opportunities in a sub-section called recommendations. In the end, the conclusion is presented in section 7.

RELATED WORK

For any successful secure system, authentication and authorization are the key components and are being used in every system ranging from critical systems to commercial systems [1-3]. Authentication refers to the verification of an individual or entity's claimed identity. On the other hand, authorization pertains to the granting of access or permission for an individual or entity to perform certain actions or possess certain assets [4]. Authentication allows a user to be recognized in the system as a valid user of the system [5] and

[1-5-6] University of Sahiwal
[2-4] COMSATS University, Sahiwal Campus
[3]Chulalongkorn University Thailand
Country: Pakistan, Thailand
Email: drshafiq@uosahiwal.edu.pk,

[6]. Authorization allows an authenticated user to use a particular system resource [7-9]. Strong authentication and authorization controls provide a more secure environment for transactions on the system [10]. Formal methods are techniques based on mathematical logic for the specification, analysis, and design of software systems and thus enable us to reason about different properties of the whole range of data [11-12]. Formal specification helps us to design computer systems in such a way that enables us to analyze systems at the design level before implementation [13]. On the other hand, conventional methods do not distinguish the code of component computations from their coordination, which creates challenges in terms of debugging and maintenance [14]. Z is a formal specification and modeling language used in a number of critical systems and is one of the more widely used formal methods [15]. Z is also being used for modeling systems in the domain of operating systems, databases, and software engineering [16-17]. The use of formal methods for authentication and authorization enables us to reason about the system with powerful tools and thus producing more reliable and secure software systems [18-19]. Formal methods have been used in many other security systems such as SSAI and Correctness by Construction [20]. The ZMSec model, an extension of the MARTE model, has been developed to aid in embedded software modeling and verification, particularly in the context of semi-formal and formal methods frameworks. The ZMSec model, along with a proposed security property checking algorithm, is presented as a valuable tool for enhancing the security of embedded software systems. [21]. To improve the security and quality of the software system, the integration between the formal and UML specification is very important. It will help to reduce errors and uncertainty in software requirements. By keeping in view this idea, a new approach in library system management is proposed by formalization using Z language and UML use case. The formal approach has a special emphasis on UML-to-Z schema diagrams consistency. Then it is validated using the Z / EVEs tool [22]. In the traditional programming language, the integrity of the program is verified at run time. However, formal specification statements are generally not applicable in this traditional way. It is also true that verifying the formalization rules consistency is very difficult. A method to prove the Object-Z norms theorem is constructed. It will build confidence by checking formal rules consistency. The analysis and theorem verification are done using prover Z / EVES [23]. Formal software specifications are useful if and only if they are consistent or do not overlap. However, checking the accuracy or consistency of formal specifications is a difficult task. A method for proving consistency or truth by generating the corresponding proof of the theorem is proposed in this study. Verifying the accuracy and consistency of the formal Object-Z specifications is the primary goal that can secure the determinants. Since Object-Z has features of inheritance, this article looks at it from various aspects and focuses on reusing the theorem's proof. Finally, the Z / EVES proof theorem is used to automatically analyze and verify the proof of the Object-Z (semi) theorem [24]. Smart contracts are gaining increasing attention due to their ability to expand the reach of blockchain applications. However, contract security is critical to its wide application. This study proposed a multi-level smart contract modeling solution for analyzing contract security. This model improved the logic rules for the program's byte-code and applied Hoare conditions to create a colored Petri network (CPN) model. The pattern detection method provided by the CPN tool helps us to analyze the security of the contract from various angles by displaying the full state field and incorrect execution paths. In addition, a highly automated modeling method is designed, adding a custom call library and a route derivation algorithm based on trackback, which increases the efficiency and precision of the dynamic simulation of the CPN model [25].

RESEARCH METHODOLOGY

The purpose of this research is to present a novel approach for modeling authentication and authorization using formal methods. We explained the core research terms briefly and discussed our research goals by highlighting our research goals in following subsections.

Formal Methods

The design of software and hardware systems is very complex, particularly the safety-critical systems. The informal and semi-formal approaches for designing good computing systems are not sufficient. The design developed in these approaches cannot be executed by using automated tools. Hence such designs cannot be verified, resulting in the production of low-quality systems. If these designs can be developed by using formal methods which are approaches based on mathematics, then the resulting designs can be verified by using theorem proving and model checking techniques. Formal methods also help to validate requirements against a software design. A wide range of formal methods is available such as Z, Object-Z, VDM-SL, VDM++, Alloy, Z/EVES, Isabelle, Coq, SPIN, etc. Z is a specification and modeling language based on set theory. Z is a model-based language. A model in Z consists of a set of state variables and a collection of constraints collectively called the state of the system and a set of operations that can change the state of the system. Models in Z can be analyzed and verified by using automatic tools such as Z/EVES. A model is written in Z and then it is passed to Z/EVES which can show that model is consistent and correct. Z/EVES is a theorem prover used for analyzing and verifying Z specifications. The major functions of Z/EVES are parsing, type checking, domain checking, schema expansion, precondition calculation, refinement proofs, and theorem proving. Z/EVES is based on the EVES system and uses EVES prover to carry out its proof steps. The language for EVES is Verdi.

Security Properties

Software systems need to be self-resilient in order to avoid attacks. This can be achieved by identifying vulnerabilities in the system by using techniques such as threat modeling and incorporating security properties into the design of systems. Security properties if incorporated properly into the system act as mitigation measures to protect systems assets and resources. There are many security properties, the most important of which are authentication and authorization. These security properties determine the access control of the system. In this paper, only authentication and authorization will be formally specified, verified, and validated. Other security properties are out of the scope of this paper.

Formal Specification

This formal model for authentication and authorization is developed in Z, a model-based formal specification language. The formal specification of authentication and authorization consists of a set of basic types, a collection of global types, a static model, and a dynamic model. The static model is a set of state variables, collectively called the state of the system and a set of invariants, imposing constraints on the state variables. These invariants must always be satisfied for the system to work securely. The operations of the system are defined in the dynamic model. These operations can change the state of the system. The operations in the dynamic model also include pre and post conditions. The pre conditions represent the state before the operation and the post conditions represent the state after the operation has been carried out.

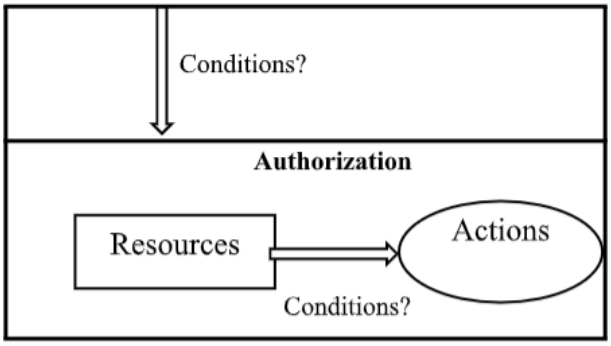

Figure 1. Flow of Research Methodology

IMPLEMENTATION

Basic Types

There are four basic types in the system. The first type "USER" represents all the real-world users or subjects acting on behalf of users in this system. The second type "CREDENTIAL" represents the credentials of users. These credentials may be passwords, keys, pass cards, fingerprints, retinal scans, etc. The type "RESOURCE" represents all the resources the system manages. These are those resources that a user may access to perform his or her actions in this system. The type "ACTION" represents all the actions that a user may perform on system resources.

$$[USER, CREDENTIAL, RESOURCE, ACTION]$$

Figure 2. Basic Types

Global Types

The global types are those types that can be accessed and used in all the schemas of a Z specification. There are two global types in this system. The first global type is Status which represents the status of a user or set of users by informing whether a user is authenticated, authorized, or a current user. A current user is always the one who is authenticated user and authorized user. Also, an authorized user is always an authenticated user.

$$Status ::= AUTHENTICATED \mid AUTHORIZED \mid CURRENT$$

Figure 3. Global Types

The second global type is which represents the status of action of a user by telling whether an action is allowed or not allowed to a user.

$$Action\_Status ::= ALLOWED \mid NOTALLOWED$$

Figure 4. Global Types

Static Model

Access Control System: this schema represents the state of the system. In this schema, the sets users, resources and actions represent all the users, resources and user actions respectively. The function *registered_users* represents those users who have valid credentials and are recognized in the system. The function represents those users who have been allocated a set of resources. The function represents a set of actions that are allowed on a particular resource.

AccessControlSystem

$$\begin{aligned}
&users: \mathbb{P}\ USER\\
&resources: \mathbb{P}\ RESOURCE\\
&actions: \mathbb{P}\ ACTION\\
&registered_users: USER \rightarrow \mathbb{P}\ CREDENTIAL\\
&user_resources: USER \rightarrow \mathbb{P}\ RESOURCE\\
&resource_actions: RESOURCE \rightarrow \mathbb{P}\ ACTION\\
&user_actions: USER \rightarrow \mathbb{P}\ ACTION\\
&authenticated_users: USER \rightarrow \mathbb{P}\ CREDENTIAL\\
&authorized_users: USER \rightarrow \mathbb{P}\ CREDENTIAL\\
&current_users: USER \rightarrow \mathbb{P}\ CREDENTIAL\\
\hline
&\mathrm{dom}\ registered_users \subset users\\
&\mathrm{dom}\ user_resources \subset users\\
&\forall u: USER \mid u \in \mathrm{dom}\ user_resources \bullet user_resources\ u \subset resources\\
&\mathrm{dom}\ user_actions \subset users\\
&\forall u: USER \mid u \in \mathrm{dom}\ user_actions \bullet user_actions\ u \subset actions\\
&\mathrm{dom}\ resource_actions \subset resources\\
&\forall r: RESOURCE \mid r \in \mathrm{dom}\ resource_actions \bullet resource_actions\ r \subset actions\\
&\mathrm{dom}\ authenticated_users \subset \mathrm{dom}\ registered_users\\
&\mathrm{dom}\ authorized_users \subset \mathrm{dom}\ authenticated_users\\
&\mathrm{dom}\ current_users \subset \mathrm{dom}\ authorized_users
\end{aligned}$$

Figure 5. Static Model

The function represents the set of actions that are allowed to a particular user in the system. The function represents the authenticated users in the system. The function authorized_ users represents those users who are authorized in the system. The function current_users represents the current users of the system.

Invariants: (i) All the registered users must be a subset of users. (ii) The users who want to access the system resources must be a subset of users and the resources for which access is granted must be a subset of resources. (iii) The users who want to perform an action on a resource must be a subset of users and all the associated actions must be a subset of actions. (iv) All the resources on which certain actions are defined must be a subset of resources and associated actions must be a subset of actions. (v) All the authenticated users must be a subset of registered users. (vi) All the authorized users must be a subset of authenticated users. (vii) All the current users must be a subset of authorized users.

Initialization Schema

InitAccess Control System: this is the initialization schema. The initialization schema is a very important schema for the validation of models developed in any formal specification language. In this initialization schema, all the state variables are empty initially. This means that at least one state of the system exists. Although obvious, we will prove this property in the next section of this paper.

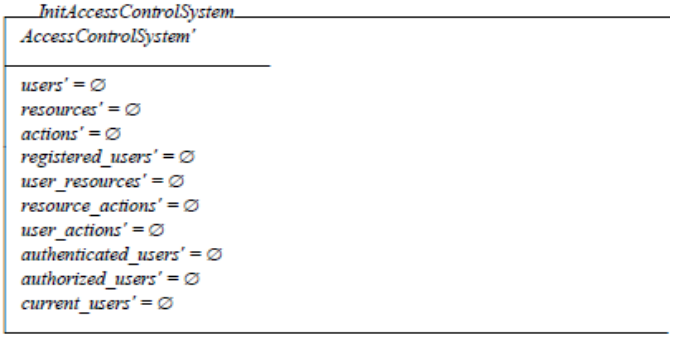

Figure 6. Initialization Schema

Dynamic Model

The operations in the system are defined in the dynamic model. In the dynamic model of the system, the state of the system may change as a result of the operations performed. The primed variables show the state after the operation has been performed. The unprimed variables show the state before the execution of the operation.

Add_Users: this operation adds new users into the system. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells us that the system state can change as a result of this operation. The pre-condition is: that the users must not already be present in the set users. Otherwise, there are no constraints on the new users. In this operation, the set users represent all those users who are recognized in the system and can access very general information from the web application. If they want to access further information then they must be registered users before accessing any particular data. The users in set users need not have credentials to access general information.

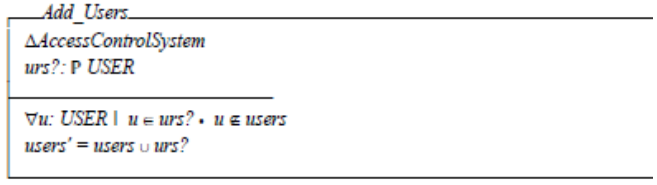

Figure 7. Adding Users to the System

Add_Resources: this operation adds new resources to the system. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre-condition is: that the resources must not already be added to the system. Any resource from the universe cannot become the resource in the system. In order for a resource to be accessed and operated upon, it must be recognized and added in the system. This operation is doing the same thing.

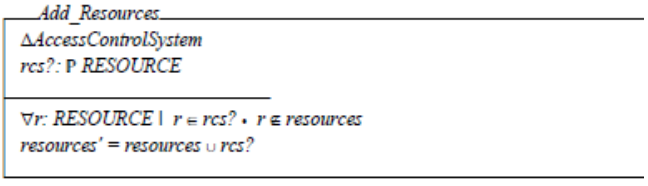

Figure 8. Adding Resources in the System

Add_Actions: this operation adds actions to the system. These actions may be the read action, write action, append action and execute the action, or some other action. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre-condition is: that the actions must not already be present in the set actions. Users cannot perform every action on the system resources. Only those actions can be performed which are explicitly added to the system resources. This operation does the same thing.

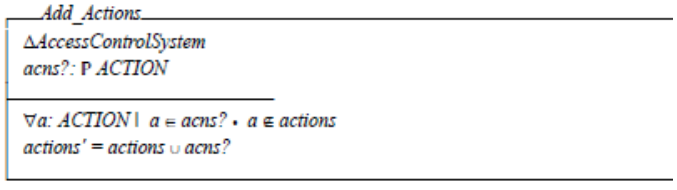

Figure 9. Adding Actions in the System

Add_Registered_User: this operation registers the users who have a set of credentials for unique identification. The first

line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre conditions are: (i) the user must be present in the set users. (ii) the user must not already be a registered user.

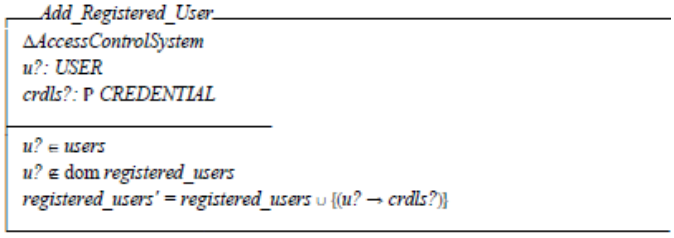

Figure 10. Registering Users in the System

Add_Actions_to_User: this operation adds actions to the registered users. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre conditions are: (i) the user must be a registered user. (ii) the actions must be present in the set actions.

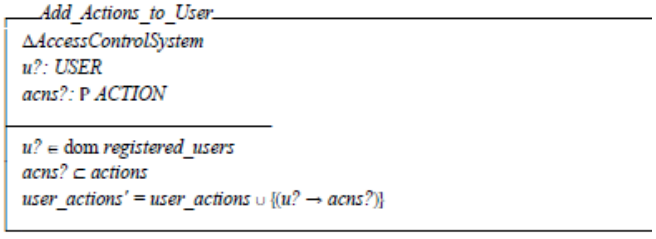

Figure 11. Adding Actions to Users in the System

Add_Actions_to_Resource: this operation adds actions to the resources. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre conditions are: (i) the resource must be present in the set resources. (ii) the actions must belong to the set actions.

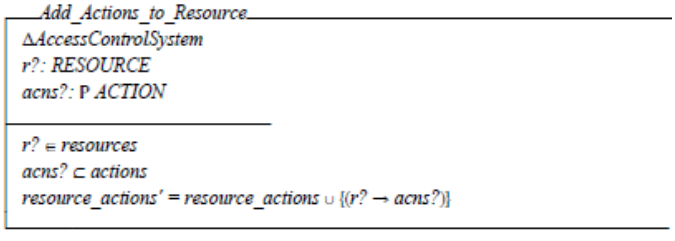

Figure 12. Adding Actions to Resources in the System

Add_Resources_to_User: this operation adds resources to users. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre conditions are: (i) the user must be a registered user (ii) the resources must be present in the set resources.

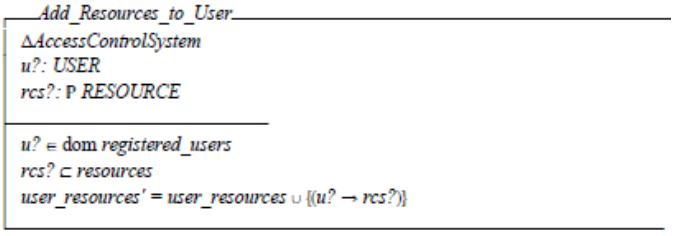

Figure 13. Resources to Users in the System

Remove_Users: this operation removes users. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre-condition is: that the users must be in the set users.

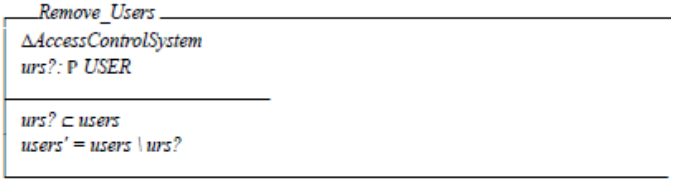

Figure 14. Removing Users from the System

Remove_Actions: this operation removes actions from the set actions. The first line shows inheritance. All the state variables from static model are accessible in this operation. It also tells that system state will change as the result of this operation. The pre condition is: the actions must belong to the set actions.

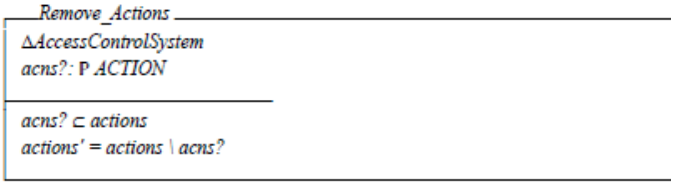

Figure 15. Removing Actions from the System

Remove_Resources: this operation removes resources from the system. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre-condition is: that the resources must already be present in the system.

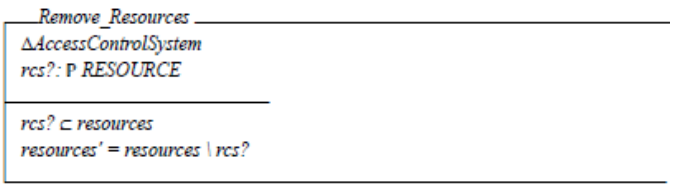

Figure 16. Removing Resources from the System

Authentication: this operation authenticates a user into the system. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre-condition is: that the user must be a registered user.

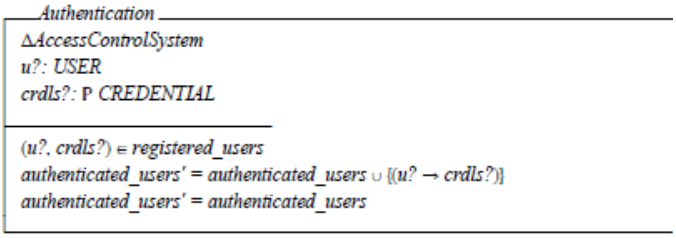

Figure 17. Authenticating Users in the System

Authorization: this operation authorizes a user to access resources. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre conditions are: (i) the user must be an authenticated user. (ii) the user must be associated with a set of required actions. (iv) the resource must have the required set of actions the user wants to perform.

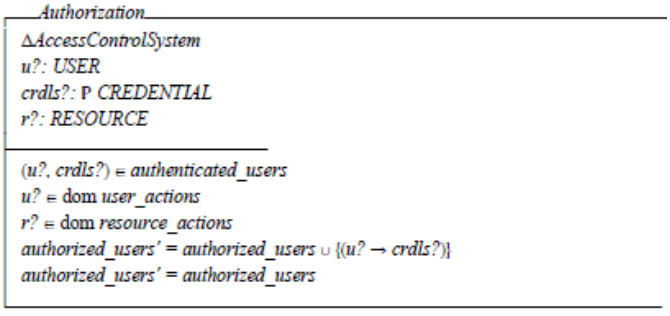

Figure 18. Authorizing Users in the System

Add_Current_Users: this operation adds current users to the system. The first line shows inheritance. All the state variables from the static model are accessible in this operation. It also tells that the system state will change as a result of this operation. The pre conditions are: (i) the user must be an authenticated user. (ii) the user must be an authorized user.

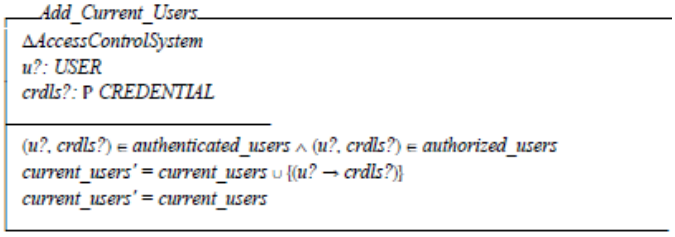

Figure 19. Adding Current Users in the System

Check_Status: this operation checks the status of a user to access resources. The first line shows inheritance. All the state variables from the static model are accessible in this operation. If the user is an authenticated user and not an authorized user then the status will be authenticated. If the user is an authorized user and not a current user then the status will be authorized. If the user is a current user then the status will be current.

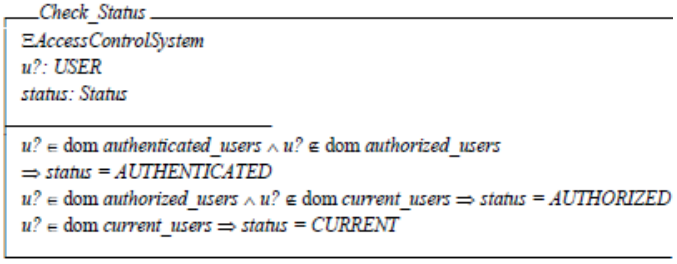

Figure 20. Checking Status of Users in the System

Check_Action: this operation checks the status of an action to be performed by a user. The first line shows inheritance. All the state variables from the static model are accessible in this operation. The pre conditions are: (i) the user must be an authenticated user. (ii) the user must be an authorized user. If the action supplied by the user is present in the set of actions granted to that user then the user is allowed to perform that action. If the action supplied by the user is not present in the set of actions granted to that user then the user is not allowed to perform that action under the system resources.

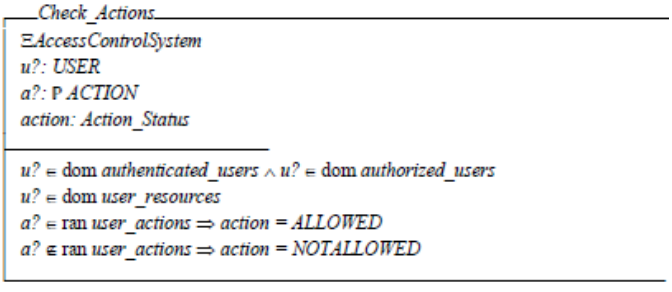

Figure 20. Checking Actions in the System

Formal Validation **:** In conventional computer software systems, testing is a useful technique for validation. But testing has limits due to the infinite volume of input data. It is

impossible to test every input within due time. Also, there are some systems that are not feasible for exhaustive testing. For example, security systems are the case where exhaustive testing is not possible because security properties cannot be validated by exhaustive testing techniques. A solution to this problem is the formal validation of models of security systems. Formal validation can be done with the help of tools like model checkers and theorem provers. By using these tools, software systems can be analyzed and proved for the whole range of data, and important properties can be checked and validated. There are three main approaches for formal validation: theorem proving, model checking, and static analysis. In theorem proving formal mathematical proofs are generated by using theorem provers or proof assistants. Some theorem provers are automatic and generate formal proofs automatically without human user intervention. While other theorem provers are semi-automatic and require guidance from human users. Model-checking tools exhaustively explore the finite models of computer systems for all possible states of the systems and generate counter-examples if some model fails to satisfy a certain property. But model checking has the problem of state explosion. In that case, only some parts of the system are checked by model checkers and it is assumed that the system will perform as in the case of this small part of checking. Static analysis automates the abstraction of program execution. Also, software systems can be validated manually by solving theorems using mathematics.

In this paper, formal models of authentication and authorization specified in the previous section have been validated automatically by using the Z/EVES theorem prover. Z/EVES is an automatic theorem prover and generates formal proofs automatically. The formal validation of authentication and authorization in this research consists of syntax checking, type checking, proving the initial state theorem, and proving properties.

Syntax and Type Checking: the formal specifications of authentication and authorization were written by using a mini text editor provided by the Z/EVES theorem prover. The syntax and types of these specifications were checked automatically by using the command "Check all paragraphs" provided by Z/EVES. In this way, we checked that all the variables in the specification have right syntax and have right type.

Proving Initial State Theorem: the purpose of the initial state theorem is to show that at least one state of authentication and authorization exists in the system.

theorem *TheInitAccessControlSystem*
$\exists AccessControlSystem' \bullet InitAccessControlSystem$

proof of *TheInitAccessControlSystem*
prove by reduce

Figure 21. Proving Initial State Theorem

This theorem has been proved by using prove by reducing the command of the Z/EVES theorem prover.

RESULTS

Proving Properties: properties are used to assert that the system behaves as expected. In this paper, three properties: checkAuthenticated, checkAuthorized, and checkAction have been proved by using theorem proving commands of Z/EVES as shown below. This theorem checks that an authorized user is an authenticated user. This theorem was proved by using prove by reducing the command of Z/EVES.

theorem *checkAuthenticated*
$\forall AccessControlSystem; u: USER \mid u \in \text{dom } registered\_users \bullet u \in \text{dom } authorized\_users \Rightarrow u \in \text{dom } authenticated\_users$

proof of *checkAuthenticated*
prove by reduce

Figure 22. Proving Theorem for Authenticated Users

This theorem checks that a current user is an authorized user. This theorem was proved by using prove by reducing the command of Z/EVES.

theorem *checkAuthorized*
$\forall AccessControlSystem; u: USER \mid u \in \text{dom } authenticated\_users \bullet u \in \text{dom } current\_users \Rightarrow u \in \text{dom } authorized\_users$

proof of *checkAuthorized*
prove by reduce

Figure 23. Proving Theorem for Authorized Users

This theorem checks whether a user who submits an action is allowed or disallowed to perform an action on the system resources. This theorem was proved by using prove by reducing the command of Z/EVES.

theorem *checkAction*
$\forall AccessControlSystem; u: USER; a: ACTION; action: Actions \mid u \in \text{dom } authorized\_users \land u \in \text{dom } user\_resources \land a \in actions \bullet a \in user\_actions\ u \Rightarrow action = ALLOWED \land a \in user\_actions\ u \Rightarrow action = NOTALLOWED$

proof of *checkAction*
prove by reduce

Figure 24. Proving Theorem for Checking Users' Actions Recommendations

CONCLUSION

In the realm of computer software, security is of utmost

importance, especially due to the increasing prevalence of distributed systems and the internet. Authentication and Authorization is the main concern, to develop secure software products. Formal methods are very suitable to analyze and validate security properties. This paper presented the formal specification of authentication and authorization with the help of practical implementation. The described models are formally specified using Z specification and the validation is done using the Z/EVES theorem prover. The paper covers a fine review of past studies. Security properties including confidentiality, integrity, availability, and non-repudiation have been elaborated as well. The proposed models of authentication and authorization have been formally specified and validated in the paper.

RECOMMENDATIONS

In this paper, the models for authentication and authorization using formal methods are discussed with the help of Z notation specifications. The models are also verified and validated using Z/EVES theorem prover. The adopted approach addressed two major properties of security (authentication and authorization) with the help of formal methods. In future, the formal specification and validation of other security properties and the relationship between these properties can be investigated.

REFERENCES

[1] W. Jie, et al., "A review of grid authentication and authorization technologies and support for federated access control." ACM Computing Surveys (CSUR) 43(2), 2011.

[2] V. Welch, et al., "Security for grid services." Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing, IEEE, 2003

.

[3] GJ. Ahn, and S. Ravi, "Role-based authorization constraints specification." ACM Transactions on Information and System Security (TISSEC) 3(4), pp. 207-226, (2000).

[4] De Almeida, M. G., & Canedo, E. D. (2022). Authentication and authorization in microservices architecture: A systematic literature review. Applied Sciences, 12(6), 3023.

[5] X. You, and Z. Lingfeng , "Improved Authentication Model Based on Kerberos Protocol." Advances in Multimedia, Software Engineering and Computing Vol. 1. Springer, Berlin, Heidelberg, pp. 593-599, 2011.

[6] G. López, et al. "A network access control approach based on the AAA architecture and authorization attributes."Journal of Network and Computer Applications 30(3), pp. 900-919, 2007.

[7] V. Kolovski, J. Hendler, and B. Parsia, "Analyzing web access control policies." Proceedings of the 16th international conference on World Wide Web. ACM, 2007.

[8] J.B. Joshi, E. Bertino, et al., "A generalized temporal role-based access control model." IEEE Transactions on Knowledge and Data Engineering 17(1), pp.4-23, 2005.

[9] H. Hu and A. Gail-Joon, "Multiparty authorization framework for data sharing in online social networks." IFIP Annual Conference on Data and Applications Security and Privacy. Springer, Berlin, Heidelberg, 2011.

[10] D. F. Ferraiolo, et al., "Proposed NIST standard for role-based access control." ACM Transactions on Information and System Security (TISSEC) 4(3), pp.224-274, 2001).

[11] J.X. Xia, and H.W. Tang, "Formal method for requirement analysis using Z notation." Sci. Technol. Eng, pp.2245-2246, 2008.

[12] R.M. Sidek, and N. Ahmad, "Deriving formal specification using Z notation." International Conference on Computer Technology and Development, 2009. ICCTD'09, Vol. 1, IEEE, 2009.

[13] N.A. Zafar and A. Fahad , "Transformation of class diagrams into formal specification." International Journal Computer Science and Network Security 11(5), pp. 289-295, 2011.

[14] Le Khanh, T. (2023). Design of Correct-by-Construction Self-adaptive Cloud Applications using Formal Methods (Doctoral dissertation, Université de Lille).

[15] J.B, Almeida, et al., "An overview of formal methods tools and techniques." Rigorous Software Development. Springer London, pp. 15-44, 2011.

[16] V. Dimitrov, ""Relationship" specification in Z-notation."Physics of Particles and Nuclei Letters 8(4), pp.391-394, 2011.

[17] J. Woodcock, et al., "Formal methods: Practice and experience." ACM computing surveys (CSUR) 41(4), p.1-36, 2009.

[18] S. Hussain, E. Harry and D. Peter "Threat modeling using formal methods: A new approach to develop

secure web applications." 7th International Conference on Emerging Technologies (ICET), IEEE, 2011.

[19] J. Jacky, The way of Z: practical programming with formal methods. Cambridge University Press, 1997.

[20] A. Coronato, and D. Giuseppe, "Formal specification and verification of ubiquitous and pervasive systems." ACM Transactions on Autonomous and Adaptive Systems (TAAS) 6(1), p.1-9, 2011.

[21] X. Hu, Y. Zhuang, and F. Zhang, "A security modeling and verification method of embedded software based on Z and MARTE," Computers & Security, vol. 88, p. 101615, 2020.

[22] Z. H. Muhamad, D. A. Abdulmonim, and B. Alathari, "An integration of uml use case diagram and activity diagram with Z language for formalization of library management system.," International Journal of Electrical & Computer Engineering (2088-8708), vol. 9, 2019.

[23] Q. Cheng and Z. Wen, "Checking Object-Z Formal Specification with Z/EVES automatically," in IOP Conference Series: Materials Science and Engineering, 2019, vol. 490, no. 4, p. 04 2016.

[24] W. Wan, Y. Yu, Q. Zeng, and Z. Wen, "Checking the consistency of Object-Z formal specification based on theorem proof," Journal of Computational Methods in Sciences and Engineering, no. Preprint, pp. 1–10, 2019.

[25] W. Duo, H. Xin, and M. Xiaofeng, "Formal Analysis of Smart Contract Based on Colored Petri Nets," IEEE Intelligent Systems, 2020.