The Nucleus **55**, No. 2 (2018) 74-84

www.thenucleuspak.org.pk

The Nucleus
ISSN 0029-5698 (Print)
ISSN 2306-6539 (Online)

Modeling of Access Control System in Event-B

S. Hussain[1*], S. Farid[2], M. Alam[3], S. Iqbal[2] and S. Ahmad[4]

[1]Department of Computer Science, University of Sahiwal, Sahiwal, Pakistan
[2]Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan
[3]ICCC, Informatics Complex, H-8, Islamabad, Pakistan
[4]Department of Computer Science, Government College of Commerce, Multan, Pakistan

ARTICLE INFO

Article history :
Received : *08 February, 2018*
Accepted : *13 September, 2018*
Published : *12 October, 2018*

Keywords:
Formal methods,
Event-B,
RODIN,
Atelier-B provers,
Access control list

ABSTRACT

Computer security is a major challenge in the current era of ubiquitous computing and the Internet. The external security measures are good but not enough to secure software systems. That is why the internal security of software systems is of much importance and more emphasis needs to be given to describe internal design of software systems. Access control system is a mechanism to ensure the internal security of software systems. There are various access control systems which are claimed to provide a secure way to access the resources but in reality these systems have many loopholes and drawbacks. Authentication and authorization are the major key elements of access control systems. Authentication is a mechanism to verify unique identification of a user in the system, and authorization is to grant access to a user to system resources. In this paper, a new generic and simplified model of access control system is proposed which is based on formal methods. The formal method used in this access control system is Event-B; Which is a formal specification and modeling language based on set theory and first order logic. Authentication process ensures that which type of users are allowed to access the system. In authorization mechanism it is ensured that a user is granted access to a system resource only if he/she has access rights for that particular resource. The resulting formal models are analyzed and verified by using RODIN tools.

1. Introduction

Now a day, more than half a billion people are using Internet all over the world which has raised various security issues. For example, attack from anywhere, sharing of information, automated infection, hostile hosts or codes, are few common issues. Such problems may cause damage to software systems. Moreover someone may mount an attack on an individual or on organization to corrupt the data or steal the information [1]. To protect the data and valuable information in the system, high security mechanisms are required. Computer security generally means software security which can be divided into two categories, i.e., external and internal securities [2]. The external security of software systems consists of security measures such as firewalls. The internal security measures ensure the secure design and security mechanisms integrated into the system design [3, 4]. Various studies reveal that security issues have increased with the passage of time which has raised a big question how to secure computer systems? Another problem is complication of software systems and it is almost impossible to develop breach free computer systems. This is because security needs to set up user accounts, passwords, access control of resources and building a trust among the stack holders. The software security was not given much importance in the past. There are two major reasons for ignoring the security importance. For software developers, security is compromised because of timely delivery of software to market. For users and administrators, security is compromised because of work to be done in a convenient way. On the other hand, security setup takes time without contributing anything to useful outcome.

Access control systems provide a secure way to access system resources by the legitimate users of the system. Access control system (ACS) is a mechanism comprises of set of policies to restrict the access of users to computer resources. There are many access control systems which are based on rigorous and strong controls; these are described elsewhere [5]. The bases of an access control are authentication and authorization. Authentication means unique identification of user of a system while authorization ensures which resource is allowed to a legitimate user of a system. There is another component of ACS called reference monitor which acts as guard to access the resources. Gollmann [6] has described computer security as a measure to keep computer resources safe from unauthorized access. These measures include detection, protection and reaction to illegal access. Computer security can be classified in terms of confidentiality, integrity and availability. Confidentiality is the prevention of unauthorized discloser of data, integrity is the unauthorized modification of data and availability is the prevention of unauthorized withholding of data. There are different algorithms to implement access control which include access control matrices, capabilities and access control lists. An access control matrix is a table in which subjects are

[*]Corresponding author : shafiqhussain@bzu.edu.pk

listed along the column and objects are listed along rows. Here subjects represent the users and objects represent the resource. The intersection of each row and column holds the access rights for this subject and object. In capabilities access rights are attached to the subject. These are stored as part of subjects. Access control lists store the access rights to an object; these are stored as part of object. Although there exist various access control models in theory but these are not very successful due to their limitations and drawbacks. For example, Role Based Access Control (RBAC) model is most widely used; however, its current version is ambiguous and has various interpretations. Some other improved versions of access control models [7, 8] are not fully verified and are lacking of mathematical foundation and proof.

A new generic and simplified formal model of access control in Event-B has been developed in this research. There are two main components of access control: authentication and authorization. Formal specification is described using contexts and machines available in Event-B. Contexts are used to describe static part of the model and machines are used to specify its dynamic part. Invariants define constraints on the system. Guards define security properties. Pre and post conditions are also defined by guards. Stepwise refinement is used to develop the formal models of the system gradually. Events are used to define actual behavior of the system in terms of operations by adding pre and post conditions.

The Event-B is a formal specification and modeling language based on set theory and first order predicate logic. The models developed in Event-B have been analyzed and verified by using Atelier B provers which are available in RODIN (Rigorous Open Development Environment for Complex Systems) tools. The RODIN tool is used for the specification and analysis of formal models developed in Event-B [9, 10]. It is to be noted that light weight formal methods are used because of usefulness and increasing understandability of the model.

There exists a lot of work on modeling of access control system but only the most relevant are critically analyzed here. Harrisson, Ruzzo and Ullman (HRU) [11] have defined a security policy model that applies to subjects, objects and actions. These entities are required to be introduced and updated on regular basis. Eventually the security policy needs recording the permissions granted which increases the complexity of the model. The Role Based Access Control (RBAC) model is most widely used model. However, its current form has various limitations [12]. In this model, the security policy specification needs to be clear in terms of structure of permission. The concept of role hierarchy is ambiguous and has various interpretations. Access control is needed in situations where different subjects require different level of access for the same resource. This is done by attaching access control list with each subject in role-based-access control [13, 14]. A subject is assigned different roles and access to resources is granted based on the role of subject or user. In mandatory access control [15, 16], different levels of subjects and resources are created and access is granted to that subject who has level greater than the level of a resource.

Authentication and authorization are major components of an access control system as described earlier. Bishop [17] has described authentication mechanism in which unique identification of a user of the system is determined. Belapurkar et al. [18] have described authorization as a mechanism to determine the access for users in software systems. The drawback of this model is to provide redundant information that increases the complexity of the model. Human resources and physical security is refined to implement the policies at a lower level [19]. The social domain of IT security is described and information is processed in a digital domain such that the refinement is done at the higher level policies [20]. The mistakes which may occur during refinement of higher level policies by external parties and insiders are addressed by Probst [21]. The informal description of policy alignment is presented elsewhere [22, 23]. The higher level policies are stated and specified with refinement of the system into components [24, 25]. A formal theoretical approach is provided for policy alignment with substantial practical implications in terms of connecting existing methods for security analysis [26]. There exist some authorization systems, for example, for multi-agent systems [27], GRID structures [28], P2P systems [29], federated scenarios [30] and cloud computing [31] for distributed environments in general. Authentication and authorization is usually done uniformly in distributed systems, for example, in the Internet. Mostly, operating systems, e.g., UNIX and Windows perform authentication and authorization at a local level. Such operating systems have a local database for user authentication in terms of passwords and a local database of authorization in the form of access control list. On the other side, a distributed system may involve systems that belong to different organizations. It is needed to have a uniform treatment of sharing the data and making decision to allow or deny the access opposite to local access controls. Some of the authorization systems, mentioned above, are based on prototyping and the others and do not provide any mechanism to describe hidden semantics under the textual models. Such procedures may cause inconsistencies between the semantics of the authorization underlying the information. An authorization model presented by Calero et al. [32] is based on semantic web-based technologies represents the underlying information focusing in the domain of information systems. Some guidelines are provided for biometrics-based client-server architecture for authentication in e-learning environments for continuous user [33]. It is argued that security issues need to be addressed at all levels of the system development to ensure a secure and reliable system implementation [34, 35]. In another interesting work, it is focused on the security issues caused by unintended flows of information in embedded systems [36]. In this work, investigation is made on the logical flows both for implicit and explicit flows which lay a solid foundation to information flow security but this work is more related to hardware security.

2. Problem Formulation

A secure system may be characterized with three titles. The first one is specification or policy: what is it supposed to do? The second one is implementation or mechanism: how does it do? The last one is correctness or assurance: does it really work? Computer users can describe information security policy under the headings: secrecy, integrity, availability and accountability [37]. Secrecy means to control who can get the information. Integrity is to control how information changes or use of resources. Availability provides prompt access to information or resources. Accountability is to keep record of users who have access to information or resources.

Organizations protect their sensitive data by means of implementing security policies which can be defined at different levels of abstraction. Higher level policies provide the assets of the organization. Physical security and IT infrastructure transforms policies into implementable lower level policies [38]. The security policy needs to be defined accurately. The implementation of security is usually divided into two parts, i.e., code and configuration. The code is a program and the configuration is a mechanism in terms of operations that controls the program which can access control lists, folder structure, passwords, encryption keys, etc. Bad programs, agents and tapping communications are three main kinds of vulnerabilities where the task of a security implementation is to defend against such vulnerabilities. All these vulnerabilities are existed due to poor design of software systems.

There are three types of access controls systems: discretionary access control system, role based access control systems and mandatory access control system. In discretionary access control, access is granted to a user based on the privileges and access level for the same resource. It allows users to access only those components of the resource for which access is granted. Its implementation is an access control list (ACL), and to implement this type of access, ACL is attached with each user. In role based access control, roles are defined for users of the system. Access to the system resources is granted to these roles, not to users directly. To define role based access control, a list of roles along with a list of mappings from roles to users is defined. In mandatory access control, access level of users and resources is defined. Access to system resources is granted based on these levels. If the access level of a user of the system is higher as compared to the access level of system resources then access is granted. There are various application-based access control systems ranging from information systems to high security domains. The proposed generic model for access control is shown in Fig. 1, which provides a framework for implementing this model. In contrast to other access control systems such as role-based access control system, discretionary access control system and mandatory access control system, this access control system is based on the use of formal methods at the design level. Formal methods are advanced rigorous techniques for constructing correct software system. Many critical software systems have been developed by using formal methods. In this access control system formal models of security properties: authentication and authorization are specified using Event-B. The resulting formal models are then verified by using RODIN tools. These verified models are correct and more secure as compared to other models as cited above. In this system, guards control access of requests from user to use resources with security. The information is usually encapsulated in objects. Based on the predefined mechanism, the guards decide whether the source is allowed to perform any operation on the object or not. The decision is made based on two kinds of information, namely, authentication and authorization.

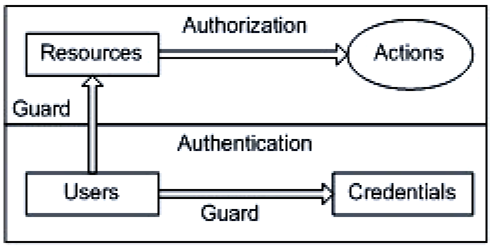


Fig. 1: Proposed access control model.

The authentication information is needed for the principal who mades the request. The authorization information is used to verify who is allowed to perform operations on the object. Authentication and authorization can be defined by the following questions:

- Authentication is done by answering the question "who has requested access to system?" The requesting people are called principals may be a person, groups, machines or programs.
- Authorization can be provided after answering the question "who is authorized to perform which kind of operations on a particular object?"

Access control mechanism determines the access of a subject who has proper access rights to a resource. The subject may be a user, process acting on behalf of user or some other entity. The resource may be any document, a file, an entity or a hardware component. The access may be read, write, append, execute or delete rights.

In this article, formal specification and analysis of generic model independent of any system for authentication and authorization is provided by using formal specification language Event-B and RODIN tools [39]. Authentication process ensures that a user is allowed to access the system. During authentication process the credentials provided by the user are matched against the credentials already stored in the system. If the credentials match, the user is authenticated otherwise refused and no access is provided to the system resources. A user is granted access to a system resource only if he has access rights for that particular resource. If a user does not have access rights for a particular resource, access to that resource is denied. Access is granted to a user depending upon the policy already defined for the users.

3. Formal Methods

Clarke and Wilson [1] and Bowen et al. [40] have defined formal methods as set of techniques and tools based on mathematical logic, symbols, notations, and formulas to specify design and analyse software as well as hardware systems. Formal methods help developers to design software systems in a precise way without ambiguities by using formal specification languages [41]. The resulting models can be analysed by using theorem provers and model checkers [42]. Further, these models of software systems can be verified and validated at the design level before implementation. In this way, formal methods help to check and analyse the desired behaviour of software systems without implementing these in a programming language [43]. Many real world software systems have been successfully designed, verified and validated [44]. Formal methods have been used successfully in complex systems ranging from medical systems, air traffic control systems [45, 46], space shuttle systems, weapon control systems, railways systems to computer hardware systems. Formal methods are very useful in the areas where exhaustive testing of properties is not possible. The formal models produced in this way are analysed at the design level without implementing these in a programming language.

Event-B is a formal modeling and specification language based on set theory developed by Abrial [9]. It is a state based language which is built on B method. There are two types of components in Event-B models: contexts and machines.

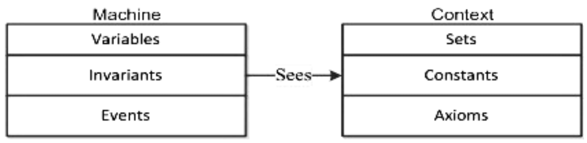


Fig. 2: Machines and Contexts [9].

Contexts define the static properties and machines define dynamic properties of its models. A context consists of carrier sets, constants, axioms and extended contexts. Carrier sets are used to define new data types. These are independent with in a context and are non-empty. Constants are declared in constants section of a context and their type is defined by axioms in the axiom section of a context. A machine can see a context to access data from the context as shown in Fig. 2. An axiom is a predicate and used to define types of carrier sets and constants. It is also used to define additional constraints on carrier sets. Axioms define theorems in the system. A context can extend another context. All the carrier sets, constants and axioms of other contexts are accessible in the extending context. Machines define dynamic properties of a system in Event-B. A machine can see contexts. All the sets, constants and axioms declared and defined in a context are accessible in machine. A machine consists of three elements: variables, invariants and events. Variables represent the state of the system and are declared in the carrier sets of a machine. The invariants are used to define the type of variables, constraints on the variables and other properties. Events define the actual behavior of the system in terms of operations. The state of the system changes due to execution of events in the machine.

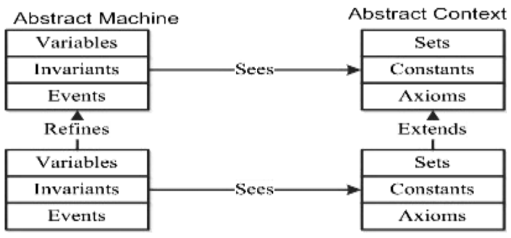


Fig. 3: Refinement in Event-B [9].

The variables in the machine represent the state of the machine. The elements of events include names, parameters, guards and actions. Names represent the names of events. Parameters are used to declare local variables to be used in the event. Guards are the conditions that must always be true. Actions are the results of event execution. Actions determine the values of variables of a machine. There is special event in every machine of Event-B model. This is called the initialization event. There are no parameters, guards and actions in this event. This event is only used to initialize state variables of a machine. The most significant feature of Event-B is concept of refinement as shown in Fig. 3. For a successful model in Event-B, a machine must be consistent which means all the invariants defined in a machine must be preserved all the time and if a machine refines another machine, it must be consistent with that machine.

4. Formal Specification of Access Control System

Formal model of access control system is developed using Event-B in this section and then refined by adding the authentication and authorization properties.

4.1 Initial Model

The model consists of static and dynamic aspects of the system. The static model is defined in data context which has four carrier sets, i.e., USER, CREDENTIAL, RESOURCE and ACTION. The USER denotes the set of all possible users. The CREDENTIAL specifies the set of all possible credentials for the users. The RESOURCE represents the set of all possible resources. Finally, the ACTION denotes the set of all possible actions. In Event-B, axioms define the constraints on the carrier sets and constants defined in the contexts. There are four axioms defined on the carrier sets in the context data as given below.

```
Carrier Sets: USER,
CREDENTIAL,
RESOURCE, ACTION
```

```
axioms:
finite (USER)
finite (CREDENTIAL)
finite (RESOURCE)
finite (ACTION)
```

These axioms ensure that all the carrier sets, USER, CREDENTIAL, RESOURCE and ACTION, are finite because Event-B handles only finite sets. In dynamic model, machines define the behavior of the system. So, two machines have been developed to model the dynamic properties of the system. The machine **InitialModel** defines the basic operations of the system. The other machine **SecurityPropertiesModel** consists of authentication and authorization. The machine InitialModel sees the context data which means that all carrier sets, constants and axioms defined in the context data are accessible by this machine.

This machine InitialMode defines variables for the users of the system, resources defined in the system, actions that are allowed on the system resources, the users that are registered in the system and the access control list. The values of these variables can change when events defined on these variables are executed. Invariants define constraints on the variables of a machine InitialModel. The types of the variables and other constraints are defined by invariants.

```
Variables:
users, resources, actions,
allowable_actions,
registered_users,
access_control_list
```

```
Invariants:
users ∈ℙ(USER)
resources ∈ℙ(RESOURCE)
actions ∈ℙ(ACTION)
allowable_actions ∈RESOURCE⇸ℙ(ACTION)
registered_users ∈USER⤔ℙ(CREDENTIAL)
access_control_list ∈USER⇸ (RESOURCE⇸ℙ(ACTION))
```

```
Sees:  InitialModel sees Data
```

In the initialization event, variables are assigned initial values. All the variables are initialized to empty sets which ensure that at least one system state exists. The events Add_User, Add_Resource, Add_Action and Add_Allowable_Actions of InitialModel add a new user, new resource, new action and assign allowable actions to system resources, respectively. The safety guards define type and ensure other constraints on these values in the system.

```
Event: Initialisation
begin
users ≔∅
resources ≔∅
actions ≔∅
allowable_actions ≔∅
registered_users ≔∅
access_control_list ≔∅
end
```

```
Event: Add_User
any u
where
  grd1: u∈USER
  grd2: u∉ users
then
  act1: users ≔ users ∪ {u}
end
```

```
Event: Add_Resource
any r
where
  grd1: r∈RESOURCE
  grd2: r∉ resources
then
  act1: resources ≔ resources ∪ {r}
end
```

```
Event: Add_Action
any a
where
  grd1: a∈ ACTION
  grd2: a∉ actions
then
  act1: actions ≔ actions ∪ {a}
end
```

```
Event: Add_Allowable_Actions
any a r
where
  grd1: r∈ resources
  grd2: a∈ℙ(ACTION)
  a_is_in_actions: a ⊆ actions
  r_not_in_domain: r∉ dom(allowable_actions)
  a_not_in_range: a∉ ran(allowable_actions)
then
  act1: allowable_actions ≔ allowable_actions ∪ {r↦a}
end
```

The events Add_Registered_User and Add_Access_Control_List, register users in the system that are already recognized by the system and grant access to these registered users to resources along with actions assigned to these resources. The safety guards ensure types and other constraints on these users, resources and actions defined.

```
Event: Add_Registered_User
any u crdls
where
  grd1: u∈ users
  grd2: crdls∈ℙ(CREDENTIAL)
  grd3: u∉ dom(registered_users)
  grd4: crdls∉
        ran(registered_users)
then
  act1:registered_users≔
  registered_users ∪ {u↦crdls}
end
```

```
Event: Add_Access_Control_List
any u a r
where
  grd1: u∈ users
  grd2: u∈ dom(registered_users)
  grd3: r∈ resources
  grd4: r∈ dom(allowable_actions)
  grd5: a∈ ℙ(ACTION)
  grd6: a⊆ allowable_actions(r)
  grd7: r∈ dom(allowable_actions)
  grd8: u∉ dom(access_control_list)
then
  act1: access_control_list ≔ access_control_list ∪ {u↦ {r↦a}}
end
```

The events Remove_Access_Control_List, Remove_Registered_User and Remove-User, remove users from the access control list, registered users and users from the system, respectively.

```
Event: Remove_Access_Control_List
any u a r
where
  grd1: u∈ users
  grd2: u∈ dom(registered_users)
  grd3: r∈ resources
  grd4: r∈ dom(allowable_actions)
  grd5: a∈ℙ(ACTION)
  grd6: a⊆ allowable_actions(r)
  grd7: r∈ dom(allowable_actions)
  grd8: u∈ dom(access_control_list)

then
  act1: access_control_list ≔
access_control_list \ {u↦ {r↦a}}
end
```

```
Event: Remove_Registered_User
any u
where
  grd1: u∈ users
  grd2: u∈ dom(registered_users)
  grd3: u∉ dom(access_control_list)
then
  act1: registered_users ≔ registered_users \{u↦ registered_users(u)}
end
```

```
Event: Remove_User
any u
where
  grd1: u∈ users
  grd2: u∉ dom(registered_users)
  grd3: u∉ dom(access_control_list)
then
  act1: users ≔ users \ {u}
end
```

The events Remove_Allowable_Actions and Remove_Resource, remove allowable actions assigned to a resource and resources from the system.

```
Event: Remove_Allowable_Actions
any r a
where
   grd1: r∈ resources
   grd2: a⊆ actions
   grd3: (r↦a) ∈ allowable_actions
   grd4: {r↦a}  ∉ ran(access_control_list)
then
   act1: allowable_actions ≔
         allowable_actions \ {r↦a}
end
```

```
Event: Remove_Resource
any r
where
   grd1: r∈ resources
   r_is_in_domain: r∈
         dom(allowable_actions)
   r_not_in_ACL: {r↦
         allowable_actions(r)} ∉
         ran(access_control_list)
then
   act1: resources ≔ resources \ {r}
   act2: allowable_actions ≔
         allowable_actions \ {r↦
         allowable_actions(r)}
```

The event Remove_Actions removes actions from the system. The safety guards ensure constraints on the values of actions and resources.

```
Event: Remove_Actions
any a r
where
  grd1: r∈ resources
  grd3: a⊆ actions
  r_not_in_domain: a∉ ran(allowable_actions)
  a_not_in_ACL:  {r↦a}  ∉ ran(access_control_list)
then
    act1: actions ≔ actions \a
end
```

The Initial Model has been completed now. All the basic features of this access control system have been completed. All the operations such as addition of a user, addition of resource and addition of actions for resources have been completed. Also, the operations such as removal of users, removal of resource and removal of action have been completed.

4.2 Model Refinement

Every machine in Event-B can refine other abstract machines. All the data of the machines being refined is accessible in the refining machine. The refining machine can add more details in the model. In this way models are developed gradually in layers. In the Initial Model machine, the basic functionality and operations of access control system have been described. The machine Security Properties Model refines Initial Model machine and adds authentication and authorization properties through refinement. This machine sees the context Data which means all the carrier sets, constants and axioms are accessible in this machine. This machine defines new variables for authenticated users and authorized users. The variables represent state of the system and are initialized to as empty sets and to ensure that at least one system state exists. Invariants define constraints on the system and properties that must remain true during the operations. These invariants define the types of the variables declared in this machine.

```
Variables:
Authenticated_users,
authorized_users,
```

```
Invariants:
authenticated_users ∈USER⤔ℙ(CREDENTIAL)
authorized_users ∈USER⤔ℙ(CREDENTIAL)
```

```
Refines:
SecurityPropertiesModel refines InitialModel
```

```
Sees:
SecurityPropertieslModel sees Data
```

```
Event: Initialisation
begin
authenticated_users ≔∅
authorized_users ≔∅
end
```

The event Adding_Authenticated_User, authenticates the users who are registered users in the system. The safety guards ensure constraints on these values in the system. The event Adding_Authorized_Users, authorize a user who is already authenticated in system and assign access to various resources available in the system.

```
Event: Adding_Authenticated_User
any u crdls
where
  grd1: u∈ users
  grd2: crdls∈ℙ(CREDENTIAL)
  grd3: u∈ dom(registered_users)
  grd4: u∉ dom(authenticated_users)
  grd5: crdls∉ran(authenticated_users)
then
  act1: authenticated_users ≔ authenticated_users ∪ {u↦crdls}
end
```

```
Event: Adding_Authorised_Users
any u r a
where
  grd1: u∈ users
  grd2: u∈ dom(registered_users)
  grd3: u∈ dom(authenticated_users)
  grd5: r∈ resources
  grd6: a∈ℙ(ACTION)
  grd7: (r↦a) ∈ allowable_actions
  grd8: (u↦ {r↦a})∈access_control_list
  grd9: u∉ dom(authorized_users)
then
  act1: authorized_users ≔ authorized_users ∪ {u↦r}
end
```

The event Remove_Authenticated_User, removes authenticated users from the system. Safety guards ensure that the user to be removed is registered and authenticated user and is not an authorized user.

```
Event: Remove_Authenticated_User
any u
where
  grd1: u∈ users
  grd2: u∈ dom(registered_users)
  grd3: u∈ dom(authenticated_users)
  grd4: u∉ dom(authorized_users)
then
  act1: authenticated_users ≔
  authenticated_users \ {u↦
  authenticated_users(u)}
end
```

```
Event: Remove_Authorised_User
any u r a
where
  grd1: u∈ users
  grd2: u∈ dom(registered_users)
  grd3: u∈ dom(authenticated_users)
  grd5: r∈ resources
  grd6: a∈ℙ(ACTION)
  grd7: (r↦a) ∈ allowable_actions
  grd8: (u↦{r↦a})∈ access_control_list
  grd9: (u↦{r↦a})∈ authorized_users
then
  act1: authorized_users ≔
  authorized_users
     \ {u↦ {r↦a}}
end
```

The machine Security Properties Model, defines authentication property and authorization property through refinement. These properties are the essence of this proposed access control system.

5. Formal Verification

Describing even specification using formal languages does not provide confidence about complete correctness of a system. It requires validation and verification of the system for surety. Formal analysis of the access control in this regard is provided by using various available facilities of RODIN.

5.1 RODIN Tool

RODIN is a tool used for formal modeling and developing software systems in Event-B. The graphical user interface of RODIN platform consists of two parts: modeling perspective and proving perspective. The purpose of modeling perspective is writing and analyzing Event-B specifications. There are three main sections in it: Event-B explorer, editor and problem view. The explorer shows all the projects, contexts and machines in the projects. The editor is used to write formal specifications of models and problem view, displays errors in the specification. The proving perspective generates formal proofs and discharge proof obligations of the models. It comprise of five main sections. The first one, proof tree section, represents all the steps carried out to complete the proof. The second section shows the invariant or event for which proof obligation is generated. The third section describes the goal to be proved. The fourth one proof control section highlights different provers available to be applied for generating proofs and discharging proof obligations. These provers are the Atelier B provers. Event-B explorer expresses all the machines and contexts. The RODIN tool consists of three components: static checker, proof obligation generator and proof obligation manager. The static checker is responsible for checking syntax and typing errors in the models. The proof obligation generator is responsible for generating proof obligations. The proof obligation manager is responsible for the management of proof obligations and related proofs. The tasks such as proof status, generation of proof rules and construction of proof trees are performed by the proof obligation manager. For the generation of proof rules, the proof obligations manager makes use of reasoners. RODIN is an open source platform and is extensible. Many plugins are available for RODIN platform such as ProB, ProR, Brama, AnimB, UML-B and Camile editor.

6.2 Verification Mechanism

Formal models of access control system, designed here, consist of one context and two machines. The context of the model includes data and the machines including InitialModel and SecurityPropertiesModel. In this section, verification of these models is presented. If all of the proof obligations are discharged then the resulting models are verified as shown in Fig. 4. In RODIN, provers run automatically when a specification in Event-B is saved and all the associated proof obligations are discharged. But some of the proof obligations are not discharged automatically. To discharge these proof obligations and verify the formal models, provers are run by the user and different tactics are applied.

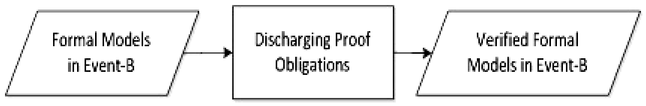
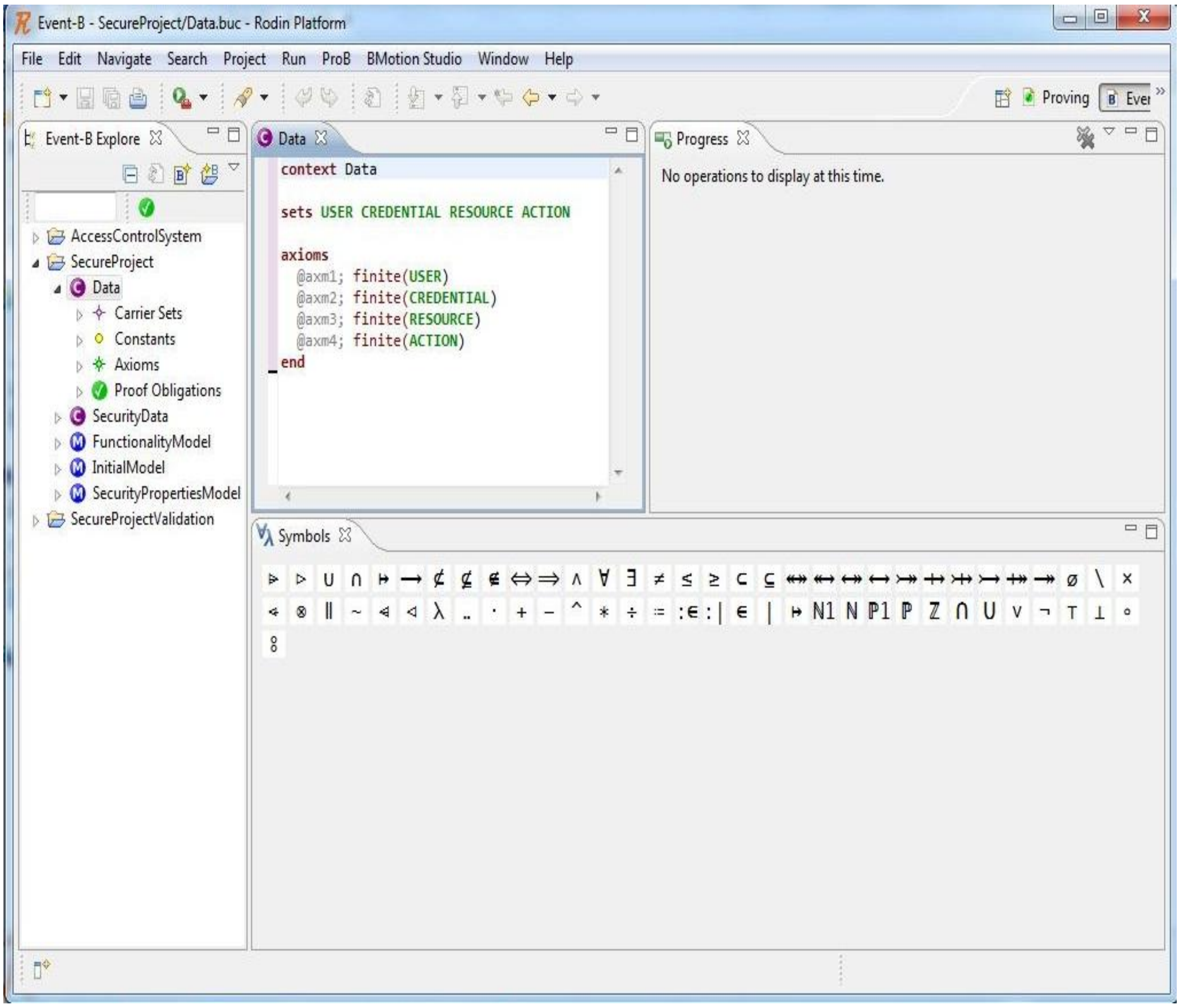

Formal Models in Event-B
Discharging Proof Obligations
Verified Formal Models in Event-B

Fig. 4: Verification of formal models.

Fig. 5 shows snapshot of the verification of the static models of the system. In Event-B explorer window, proof obligations of the context are turned green which means that the model of this context is consistent and there is no voilation of any axiom defined in it.



```
context Data

sets USER CREDENTIAL RESOURCE ACTION

axioms
  @axm1; finite(USER)
  @axm2; finite(CREDENTIAL)
  @axm3; finite(RESOURCE)
  @axm4; finite(ACTION)
end
```

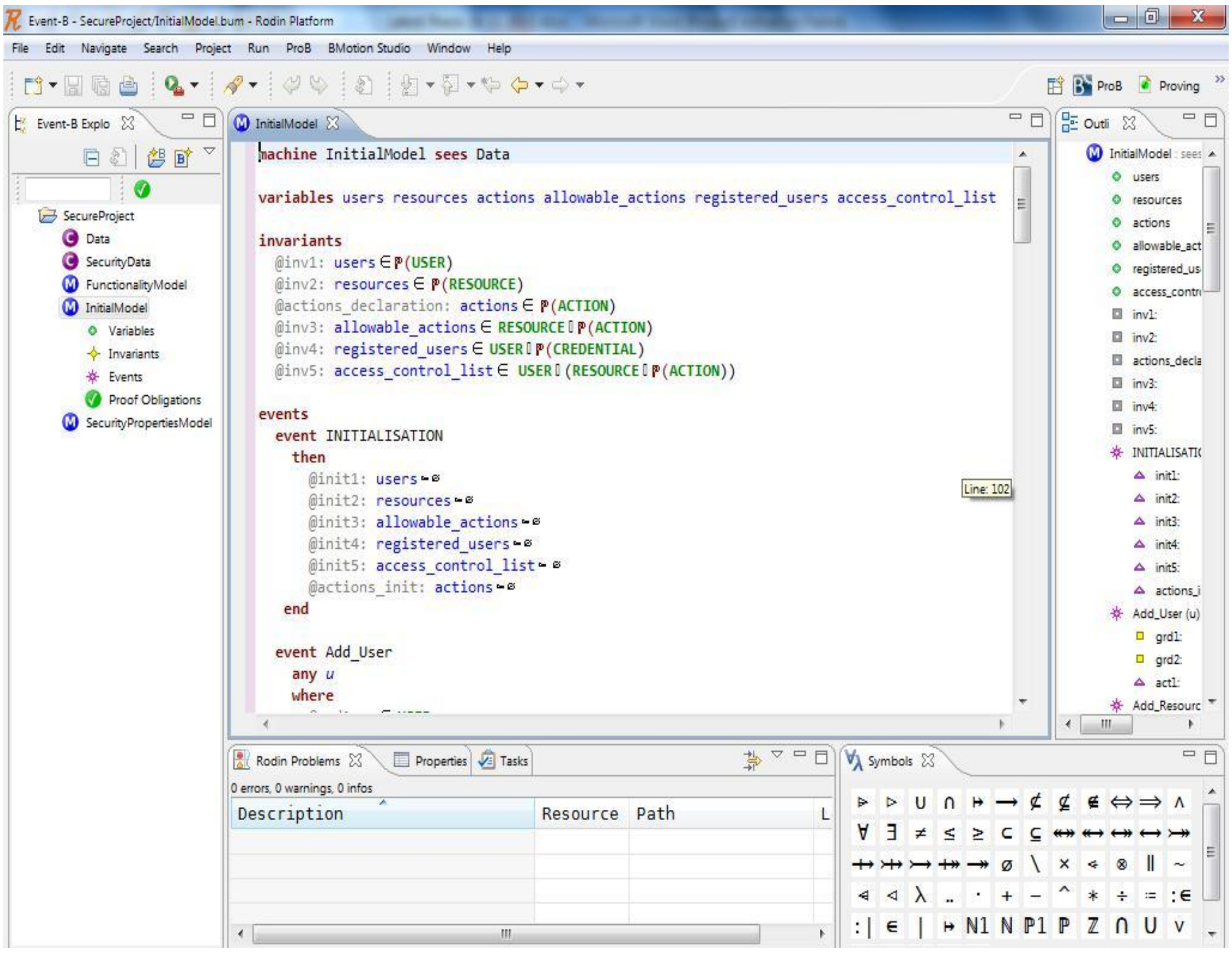
No operations to display at this time.

Fig. 5: Formal verification of static model.

The dynamic model of the access control system consists of machines: InitialModel and SecurityProperties Model. Each of these machines contains invariants, events and guards within events. Verification of these constructs means that there is no proof obligation associated with these constructs. Fig. 6 shows snapshop of the verification of the initial model. In this figure, the proof obligations associated with the invariants, events and guards are green which means all the proof obligations are discharged successfully and the model is consistent and verified. In verification of SecurityPropertiesModel, the associated proof are done and there is no proof obligation remains undischarged and the model of this machine has been found consistent.

7. Conclusion

A new generic simplified formal model of access control independent of any system has been presented. Event-B is a formal specification and modeling language used for the modeling of this security critical system underhand. The layer-wise approach has been used to develop the model by refinement. In the first layer, static model of the access control system is developed. Then dynamic model of the system is described based on the static model. This dynamic model is refined by integrating authentication and authorization security properties. The invariants and constraints on the critical information are used to define the security properties in static part of the model. Further the security is ensured by defining pre and post conditions on the operations describing behavior of the system. These conditions are implemented through guards by predefined procedures which decide whether the source is allowed to perform any operation on the object or not. Finally, the model is verified using various facilities of RODIN tools where Atelier B Provers has been utilized as plugin. The results verify the both static and dynamic models successfully. The model can be applied to real scenarios for its evaluation in future.



```
machine InitialModel sees Data

variables users resources actions allowable_actions registered_users access_control_list

invariants
  @inv1: users ∈ ℙ(USER)
  @inv2: resources ∈ ℙ(RESOURCE)
  @actions_declaration: actions ∈ ℙ(ACTION)
  @inv3: allowable_actions ∈ RESOURCE ▯ ℙ(ACTION)
  @inv4: registered_users ∈ USER ▯ ℙ(CREDENTIAL)
  @inv5: access_control_list ∈  USER ▯ (RESOURCE ▯ ℙ(ACTION))

events
  event INITIALISATION
    then
      @init1: users ≔ ∅
      @init2: resources ≔ ∅
      @init3: allowable_actions ≔ ∅
      @init4: registered_users ≔ ∅
      @init5: access_control_list ≔ ∅
      @actions_init: actions ≔ ∅
   end

  event Add_User
    any u
    where
```

Fig. 6: Formal Verification of Initial Model

References

[1] D.D. Clark and D.R. Wilson, "A comparison of commercial and military computer security policies", IEEE Symposium on Security and Privacy, pp. 184–184, 1987.

[2] S. Hussain, P. Dunne and G. Rasool, "Formal Specification of Security Properties using Z Notation", Research Journal of Applied Sciences, Engineering and Technology, vol. 5, no. 19, pp. 4664–4670, 2013.

[3] S. Hussain, G. Rasool, M. Atef and A.K. Shahid, "A review of approaches to model security into software systems", Journal of Basic and Applied Scientific Research, vol. 3, no. 4, pp. 642–647, 2013.

[4] S. Hussain, G. Rasool, M. Atef and A.K. Shahid, "FDMSWAP: Formal Development Methodology for Secure Web Applications", Journal of Basic and Applied Scientific Research, vol. 3, no. 4, pp. 1123-1128, 2013.

[5] T.S. Hoang, D. Basin and J.-R. Abrial, "Specifying access control in Event-B", Technical Report, vol. 624, 2009.

[6] D. Gollmann, "Computer Security", Wiley Interdisciplinary Reviews: Computational Statistics, vol. 2, no.5, pp. 544-554, 2010.

[7] J. Barkley, K. Beznosov and J. Uppal, "Supporting relationships in access control using role based access control", Proceedings of the fourth ACM workshop on Role-based access control, pp. 55–65, 1999.

[8] E.C. Cheng, "An object-oriented organizational model to support dynamic role-based access control in electronic commerce", Decision Support Systems, vol. 29, no. 4, pp. 357–369, 2000.

[9] J.R. Abrial, Modeling in Event-B: System and Software Development, Cambridge University Press, 2010.

[10] J.R. Abrial, "Formal methods: Theory becoming practice", J. UCS, vol. 13, no. 5, pp. 619–628, 2007.

[11] M.A. Harrison, W.L. Ruzzo and J.D. Ullman, "Protection in operating systems", Communications of the ACM, vol. 19 no. 8, pp. 461-471, 1976.

[12] S.I. Gavrila and J.F Barkley, "Formal specification for role based access control user/role and role/role relationship management", Proceedings of the third ACM workshop on Role-based access control, October 1998, pp. 81-90.

[13] E. Bertino, P.A Bonatti and E. Ferrari, "TRBAC: A temporal role-based access control model", ACM Transactions on Information and System Security (TISSEC), vol. 4, no. 3, pp.191-233, 2001.

[14] R. Sandhu, D. Ferraiolo and R. Kuhn, "The NIST model for role-based access control: Towards a unified standard", ACM Workshop on Role-based Access Control, pp. 1-11, July 2000.

[15] D.E. Bell and L.J. LaPadula, "Secure computer systems: Mathematical foundations", Mitre Corp Bedford MA, vol. 1, no. MTR-2547, 1973.

[16] K.J. Biba, "Integrity considerations for secure computer systems", Mitre Corp Bedford MA, vol. 1, no. MTR-3153, 1977.

[17] M.A. Bishop, "The art and science of computer security, Addison-Wesley Longman Publishing Co., Inc., 2002.

[18] A. Belapurkar, A. Chakrabarti, H. Ponnapalli, N. Varadarajan, S. Padmanabhuni and S. Sundarrajan, "Distributed systems security: issues, processes and solutions", John Wiley & Sons, 2009.

[19] R. Baskerville, and M. Siponen, "An information security meta-policy for emergent organizations", Logistics Information Management, vol. 15, no. 5/6, pp. 337-346, 2002.

[20] J. Rees, S. Bandyopadhyay and E.H. Spafford, "PFIRES: A policy framework for information security", Communications of the ACM, vol. 46, no 7, pp.101-106, 2003.

[21] C.W. Probst, R.R. Hansen and F. Nielson, "Where can an insider attack?", Proc. of International Workshop on Formal Aspects in Security and Trust, Heidelberg, Berlin: Springer, , pp.127-142, 2004.

[22] M. Abrams, and D. Bailey, "Abstraction and refinement of layered security policy", Information Security: An Integrated Collection of Essays, pp.126-136, 19995.

[23] I.M. Olson and M.D. Abrams, "Information security policy", Information Security: An Integrated Collection of Essays, 1995.

[24] D. Pavlovic and D.R. Smith, 2003. "Software development by refinement", Formal Methods at the Crossroads from Panacea to Foundational Support, Heidelberg, Berlin: Springer, 2003, pp. 267-286.

[25] D. Sannella and A. Tarlecki, "Foundations of algebraic specification and formal software development", Springer Science & Business Media, 2012.

[26] W. Pieters, T. Dimkov and D. Pavlovic, "Security policy alignment: A formal approach", IEEE Systems Journal, vol. 7, no. 2, pp.275-287, 2013.

[27] S. Fugkeaw, P. Manpanpanich and S. Juntapremjitt, "A hybrid multi-application authentication and authorization model using multi-agent system and PKI", Proc. of the Fourth IASTED Asian Conference on Communication Systems and Networks, ACTA Press, pp. 96-101, March 2007.

[28] J. Li, and D. Cordes, "A scalable authorization approach for the Globus grid system", Future Generation Computer Systems, vol. 21, no. 2, pp. 291-301, 2005.

[29] E. Palomar, J.M. Estevez-Tapiador, J.C. Hernandez-Castro and A. Ribagorda, "Certificate-based access control in pure p2p networks", Sixth IEEE International Conference on Peer-to-Peer Computing, pp. 177-184, 2006.

[30] O. Cánovas, A.F. Gómez-Skarmeta, G. López and M. Sánchez, "Deploying authorization mechanisms for federated services in eduroam (DAMe)", Internet Research, vol. 17, no. 5, pp.479-494, 2007.

[31] A. Gbadegesin, R. Batoukov and D.R. Reed, "Flexible scalable application authorization for cloud computing environments", U.S. Patent 8,418,222, Microsoft Corporation, 2013.

[32] J.A Calero, G.M. Perez and A.G. Skarmeta, "Towards an authorisation model for distributed systems based on the Semantic Web", IET information security, vol. 4, no. 4, pp.411-421, 2010.

[33] A. Moini, and A.M. Madni, "Leveraging biometrics for user authentication in online learning: A systems perspective", IEEE Systems Journal, vol. 3 no, 4, pp. 469-476, 2009.

[34] D. Arora, S. Ravi, A. Raghunathan and N.K. Jha, "Hardware-assisted run-time monitoring for secure program execution on embedded processors", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, no. 12, pp. 1295-1308, 2006.

[35] D.D. Hwang, P. Schaumont, K. Tiri and I. Verbauwhede, "Securing embedded systems", IEEE Security & Privacy, vol. 4, no. 2, pp.40-49, 2006.

[36] D. Mu, W. Hu, B. Mao and B. Ma, "A bottom-up approach to verifiable embedded system information flow security", IET Information Security, vol. 8, no. 1, pp. 12-17, 2014.

[37] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols", IEEE transactions on Software Engineering, no. 1, pp. 6–15, 1996.

[38] P. England, B. Lampson, J. Manferdelli and B. Willman, "A trusted open platform", Computer, vol. 36, no. 7, pp. 55-62, 2003.

[39] J.R. Abrial and S. Hallerstede, "Refinement, decomposition and instantiation of discrete models: Application to Event-B", Fundamenta Informaticae, vol. 77, no, 1-2, pp. 1-28, 2007.

[40] J.P. Bowen, K. Bogdanov, J.A. Clark, M. Harman, R.M. Hierons and P. Krause, 2002, August. "FORTEST: Formal methods and testing", in Proc. of IEEE 26th Annual International of Computer Software and Applications Conference, pp. 91-101, August 2002.

[41] J.R. Abrial, M. Butler, S. Hallerstede, T.S. Hoang, F. Mehta and L. Voisin, "Rodin: An open toolset for modelling and reasoning in Event-B", Int. Journal on Software Tools for Technology Transfer, vol. 12, no. 6, pp. 447-466, 2010.

[42] P. Bjesse, "What is formal verification?", ACM SIGDA Newsletter, vol. 35, no. 24, p. 1, 2005.

[43] L. Ma and J. J. Tsai, "Formal modeling and analysis of a secure mobile-agent system", IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, vol. 38, no. 1, pp. 180–196, 2008.

[44] J. Bendisposto, M. Jastram, M. Leuschel, C. Lochert, B. Scheuermann and I. Weigelt, "Validating Wireless Congestion Control and Realiability Protocols using ProB and Rodin", Workshop on Formal Methods for Wireless Systems, August 2008.

[45] N.A. Zafar, "Safety control management at airport taxiing to take-off procedure", Arabian Journal for Science and Engineering, vol. 39, no. 8, pp. 6137-6148, 2014.

[46] N.A. Zafar, "Formal Model of Aircrafts Safety Separation", International Journal of Innovative Computing, Information and Control, vol. 10, no. 4, pp. 1401-1412, 2014.