Test Case Design of Security Threats through State Machine OCL

Asif Raza  
Riphah International University  
Islamabad, Pakistan  
Asifraza0765@yahoo.com

Rizwan Bin Faiz  
Riphah International University  
Islamabad, Pakistan  
rizwan.faiz@riphah.edu.pk

Hafiz Syed Muhammad Muslim  
COMSATS University  
Islamabad, Pakistan  
muhammad.muslim@comsats.edu.pk

Abstract— Software testing is one activity for Software Quality Assurance (SQA). One way to test is through Black box test cases at design level. Since UML is a de-facto design language and is one of the most used design language now a day. Therefore, there is need to design black box test cases using UML behavioral diagrams. Object Constraint Language (OCL) is a formal language to write constraints on UML diagrams as it is unambiguous. Although test cases against robustness have been designed through OCL expressions but no test cases as yet have been designed against security specific authentication, access control and availability. Specification and testing of security specific authentication, access control and availability through OCL. An experiment is performed with a hypothesis ‘if Security Specific Authentication, Access Control and Availability is specified through OCL in state diagram then Test cases can be designed’. We have specified security in OCL then same OCL expressions are used to model security in UML state diagram. Furthermore, mutation testing is performed on OCL expressions. Then ECP is done using same extracted mutants. Security specific authentication, access control and availability is specified in OCL and then test cases are designed using a black box testing technique Equivalence Class Partitioning. Hypothesis results shows that one can design test cases through OCL. This thesis shows that Specification and Testing of Security specific authentication, access control and availability through OCL.

Index Terms— OCL, state machine, Black Box test case, security threat mitigation, Equivalence Class Partitioning.

1. INTRODUCTION

Software testing is one fundamental way to control software quality. In order to perform testing, we have to design test cases and generate test data against specific requirements may be FR or NFRs. Software testing increase the quality and efficiency and ensure that the software does not contain any bug which can become a reason of software failures. A test case that is appropriately designed, detect the bugs in a software, and also support to decrease the cost pertaining to software testing. Test cases designed at designing phase reduce overall cost of testing and chance of project failures [25]. UML offers powerful mechanism for describing software and it is widely used in Software Design. There are 13 different diagrams offered by UML in which only 4 diagrams show the dynamic behavior of system. State diagram is one of the UML diagram which shows the dynamic behavior. UML state chart diagram describes the changing aspects of a model element as it changes its internal state as the response of receiving some external trigger [12]. We can use a State Diagram to model software systems. States of state diagrams are main element in which system can be in, transitions have inputs that trigger states. According to latest superstructure of the UML (version 2.5.1) [OMG, 2017], diagrams given by the UML do not address important NFRs related semantics. To overcome this drawback, Researchers proposed different techniques to Extend UML Diagrams so that one can add NFRs [1, 5, 6, 9, and 11]. To extend state diagram, one technique is to append OCL expression in it. OCL are used to specify constraints on transitions, same constraints can be used for individual software requirements. Applications of OCL constraints in the UML diagrams are state invariants / guard conditions in state chart diagram, class invariants in class diagrams, and pre- and post-conditions of use case. According to latest version of OCL (Version 2.4) [OMG, 2014], OCL expressions have no side effect on model as it cannot alter anything in the model. This means that the state of the system will never change because of an OCL expression, even though an OCL expression can be used to specify a state change. Whenever an OCL expression is assessed, it simply delivers a value. OCL Expressions can be specified against mitigation of threats. Threats on a system can be derived by threat modeling. Threat modeling is a technique for improving security by detecting threats, and then defining mitigations to remove the effects of threats to the system [24]. Using OCL Expressions one can perform mutation testing [23]. Mutation Testing is a technique which is used to test specifications of software. In mutation testing, all
II. LITERATURE REVIEW

The literature shows that test cases against FRs through OCL in state diagram are designed by using guard condition on transitions between two states but NFRs i.e. security through OCL in state diagram still require attention. Shaukat Ali et al. [15], specified robustness by appending OCL Expressions on it. They stated that They proposed RobUstness Modeling Methodology (RUMM) to model robustness behavior of software system. The limitation of this paper, as mentioned by authors is that they only considered robustness behavior of system, they have not investigated other nonfunctional requirements such as Security. Shaukat Ali et al. [16], performed testing using OCL constraints. OCL constraints are used for testing are same, which was appended by them in 2011 on state diagram to model robustness behavior of systems. They used a Search-Based Heuristics technique to design test cases. Latterly [17] improved heuristics to design test cases. Shaukat Ali et al. [18], performed testing using OCL constraints. They extended their previous work [16, 17] by rewriting the OCL Expressions. Further, they developed an OCL solver named as ESOCL. They generated boundary values from OCL Expressions and then test cases are designed. Similarly, one can design test cases using other black box testing techniques i.e. equivalence class partitioning. Md Azharuddin Ali [7], designed test cases through OCL using state diagram. The proposed approach is to transform a given state diagram into a finite state Machine (FSM). They Build OCL Expressions using pre-condition and post condition of Use-Cases and then appended same OCL Expressions on transition as guard condition of state diagram. Finally, they design the test cases using same diagram. This papers advocates that, test cases against FRs were designed using OCL Expression, likewise, one can design test cases against security using OCL.

A. Research Gap

Research Gap extracted from research papers is as follows:

- It is evident from existing literature that for unambiguous modelling of security threats and their corresponding mitigation in UML models we need to specify formal constraints in OCL [26]. However, there is no evidence of such modelling upon state machine. Therefore, there is a need to specify security threat and their corresponding mitigation and then design test cases through such model.

III. RESEARCH METHODOLOGY

- **Experiment**

  An experiment is performed to do this research work with a hypothesis i.e. if security is modeled through OCL in State diagram then Test cases can be designed and an alternative hypothesis i.e. if
security is modeled through OCL in State diagram then Test cases cannot be designed.

- **Factors and Treatments**
  We have tested our hypothesis with One factor i.e. Security and three treatments i.e. authentication, access control and availability.

- **Variables**
  Dependent variable of this experiment is Test Cases and security is Independent Variable. Control variable in this experiment is threats which are discussed.

- **Data Collection**
  Security specific requirements used in this thesis i.e. authentication, access control and availability are extracted from Hospital Information and Management System’s case study.

A. **Security Specification in OCL**

In this section, possible attacks / threats on security i.e. Brute Force and Dictionary attack on Security specific authentication, SQL Injection / URL manipulation on Security specific access control and Denial of Services (DOS) attack on Security specific availability. Moreover, mitigation of these threats in natural language and in object constraint language (OCL) are also described. All OCL Expressions are also validated through tool i.e. Papyrus. In Security Authentication threats can be Brute Force, Dictionary Attack and heuristic attacks are used to apply different combinations of username and password. Using these attacks, one can enter into system if mitigation is not implemented correctly and its corresponding mitigation is to block user after number of wrong attempts which can be formally specifies in OCL as:

\[
\{\text{OCL}\} \text{Self. Attempts} = \text{attempts} + 1 
\]

\[
\{\text{OCL}\} \text{Self. username} = \text{true} 
\]

\[
\{\text{OCL}\} \text{Self. Username} = \text{false} 
\]

\[
\{\text{OCL}\} \text{Self. Password} = \text{true} 
\]

\[
\{\text{OCL}\} \text{Self. Password} = \text{false} 
\]

\[
\{\text{OCL}\} \text{Self. Attempts} < 4 
\]

\[
\{\text{OCL}\} \text{Self. Attempts} \geq 4 
\]

Whereas in access control one common threat is SQL Injection is an attack in which user can get access of unauthorized information by altering query [21]. This can be mitigated through blocking user after no of attempts and verify that the type of data queried matches the type expected and can be formally specified in OCL as:

\[
\{\text{OCL}\} \text{hims. username} = \text{true} 
\]

\[
\{\text{OCL}\} \text{hims. password} = \text{true} 
\]

\[
\{\text{OCL}\} \text{hims. role} = '\text{Admin}' 
\]

\[
\{\text{OCL}\} \text{attempts} < 3 
\]

\[
\{\text{OCL}\} \text{attempts} \geq 2 
\]

In Mitigation, system always checks role stored in system at time of login, not from query entered to access the page. Besides in availability one common threat is Denial of Services is an attack from malicious user in which intentions of mal user is to make system unavailable for other legal users. This can be mitigated through adding restriction in number of requests from same IP. This can be mitigated by formal specification in OCL as:

\[
\{\text{OCL}\} \text{self. hims. get IP()}
\]

\[
\{\text{OCL}\} \text{if IPaddress = Ipsaver then hims. attempts = hims. attempts + 1; else 'nothing'; endif}
\]

\[
\{\text{OCL}\} \text{hims. Interval} \geq 5 \text{ and hims. attempts} \leq 2
\]

\[
\{\text{OCL}\} \text{hims. Interval} < 5 \text{ and hims. attempts} > 2
\]

\[
\{\text{OCL}\} \text{IP Blocked}
\]
IV. SECURITY MODELING

A description of a system using concepts and guidelines is called Model. In UML, there are 13 different diagrams to model a system. Only four diagrams show dynamic behavior of system which are Activity Diagram, State Diagram, Sequence Diagram and Collaboration Diagram. In UML State Diagram, States of state chart diagram are main element, states are types, do not have any attributes. Without attributes and properties, it is impossible to write OCL Expressions on it. To resolve this anomaly, a state diagram should be defined as nested type of a Class diagram, so that it can acquire properties of specific class and it can be done by using a tool called Papyrus (Eclipse_documentation) [2]. Same tool is also used to validate OCL Expressions.

A. Class Diagram

Keeping above in view, an abstract class diagram against Software Requirement Specification (SRS) of Hospital Information Management System (HIMS) is created and its Classes having objects are used to create state diagrams. Figure 1 shows a class diagram of Hospital Information and Management System in which distinct roles such as Admin, Department User, Store Keeper, OIC Store, Doctor, Patient and Dispenser are
highlighted. Roles use the attributes of Users class which are Username and Password to access the HIMs.

B. State Diagram

Models of UML state diagram is designed for different processes, same are extended by appending OCL Expressions which are extracted by mitigation statements written against threats in table 2. Same OCL Expressions are appended as guard conditions on transitions. In Figure 2, mitigation against Brute Force and Dictionary Attacks were implemented. When a user accesses a login page then a variable i.e. attempts increase by 1, first time from 0 to 1, when mal user enters correct username but invalid password then the attempts against usernames are increased. This process repeats until wrong numbers of attempts are less than 4. After 4 invalid attempts, the process is ended and Login Page is not displayed. On the other hand, when a user enters a valid username and password Home Page pertaining to username is displayed. In Figure 3, mitigation against SQL Injection / URL manipulation Attacks were implemented. When a legal user / role login to system and try to access an unauthorized page, he becomes a mal user. In this diagram only Admin have rights to access the accounts page. When an Admin try to access accounts page the account information will displayed. When role other than admin tries to access the accounts page variable attempts against role is increased by one. It repeats until no of attempts are less than three. After three attempts account of role is locked. In figure 4 DOS attack is applied to HIMs system. A Mal user can initiate attack using web browser to make system unavailable. When any user tries to access HIMs system, server gets IP address of user and save it in a file named IP Saver and increase number of attempts against IP by one. While giving access to HIMs, system checks two variables i.e. number of attempts from unique IP Address within time interval of five seconds. If time interval is less than five seconds and no of attempts are more than two (One can adjust these variable as per requirements) then IP blocked state achieved, otherwise system give access to user. After security specific authentication, access control and availability is specified and OCL Expressions are validated through tool, mutants against OCL are extracted by performing mutation testing.

![Fig. 2. State Diagram for Access Control](image)
V. MUTATION TESTING

Mutation Testing is a technique which is used to test specifications of software. Vadim Okun 2004, used mutation testing technique to test requirement specifications [19]. Similarly, in 2005, Percy Antonio Puri Salas and Bernhard K. Aichernig used this technique to test OCL Specifications [20]. Hence, we are using this technique to generate mutants of OCL Expressions which are used to mitigate threats, appended on UML State Diagram.

A. Mutation Testing for Security Specific Authentication

In table three, mutation testing of variables which were used to implement mitigation for security specific authentication threats i.e. Brute Force and Dictionary Attack, is applied. Possibilities / mutants of usernames are true and false in case of attack username should be false. Similarly, Possibilities / mutants of password are true and false. Mutants against No of attempts are less than four, equal to four and greater than four. A Mal user can only have rights to make attempt to login the system,
if no of attempts are less than four, username is true and password is true.

### TABLE II
**MUTATION TESTING FOR AUTHENTICATION**

<table>
<thead>
<tr>
<th>Ser</th>
<th>OCL Expression</th>
<th>Attributes</th>
<th>Authentication Mutants</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{OCL}Self.Username = Valid</td>
<td>Username</td>
<td>Invalid</td>
</tr>
<tr>
<td>2</td>
<td>{OCL}Self.Password = Valid</td>
<td>Password</td>
<td>Invalid</td>
</tr>
<tr>
<td>3</td>
<td>{OCL}Self.attempts &lt; 4</td>
<td>Attempt</td>
<td>&gt;=4</td>
</tr>
</tbody>
</table>

### B. Mutation Testing for Security Specific Authentication

In this table mutation testing of variables which were used to implement mitigation against threat i.e. SQL Injection / URL manipulation, is applied. Possibilities of usernames is true or false in case of attack username could be false. Similarly, Possibilities of password are true and false. Here, only Admin role have permission to access the accounts page as described in figure 3 all other roles shown in mutation column are the possibilities that can try to access the accounts page. Possibilities against No of attempts are less than 3, equal to 3 and greater than 3. If a role other than admin attempts more than 2 times then the role account will be locked.

### TABLE III
**MUTATION TESTING FOR ACCESS CONTROL**

<table>
<thead>
<tr>
<th>Ser</th>
<th>OCL Expression</th>
<th>Attributes</th>
<th>Mutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{OCL}Self.Username = Valid</td>
<td>Username</td>
<td>Invalid</td>
</tr>
<tr>
<td>2</td>
<td>{OCL}Self.Password = Valid</td>
<td>Password</td>
<td>Invalid</td>
</tr>
<tr>
<td>3</td>
<td>{OCL}Self.attempts &lt; 3</td>
<td>Attempt</td>
<td>&gt;=2</td>
</tr>
<tr>
<td>4</td>
<td>{OCL}Self.role = Admin</td>
<td>Role</td>
<td>Department User, Store Keeper, OIC Store Keeper, Patient, Doctor, Dispenser</td>
</tr>
</tbody>
</table>

### C. Mutation Testing for Security Specific Access Control

In table four, mutation testing of variables which were used to implement mitigation for Security Specific Availability threat i.e. Denial of Services (DOS), is applied. Possibilities of IP-Address is there should be an IP address. Attempts from same IP address in time Interval greater than five seconds are acceptable, if time interval is less than five seconds then number of attempts from IP address increased by one, it repeats until no of attempts are less than three. If no of attempts are greater than two than IP will be blocked.

### TABLE IV
**MUTATION TESTING FOR AVAILABILITY**

<table>
<thead>
<tr>
<th>Ser</th>
<th>OCL Expression</th>
<th>Attributes</th>
<th>Mutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hims.get_IP()</td>
<td>IP</td>
<td>Null Address</td>
</tr>
<tr>
<td>2</td>
<td>{OCL}Self.Password = Valid</td>
<td>Password</td>
<td>&lt;= 5</td>
</tr>
<tr>
<td>3</td>
<td>{OCL}Self.attempts &lt; 3</td>
<td>Attempt</td>
<td>&lt;= 2</td>
</tr>
</tbody>
</table>

In this section test cases are designed and test data is generated. There are two main types of testing used to generate test data of software, which are Black Box Testing and White Box Testing. White Box testing is used to test code of software module. While Black Box testing is used to test specification of software systems as code of system is not available [21]. Test Cases against authentication are: Table 5 shows that; mal user tries to login into system by applying different combination of usernames & passwords through brute force and dictionary attack. In pre-state user credentials are required, when user apply attacks with different combination of username and password system check number of attempts with respect to username. If no of attempts are greater than three then the expected post state is ’User Blocked’ otherwise system increase no of attempts by one and expected post state is ’Login Page’. Test cases against access control that only Admin can update and view account information’s are as follow: Table 6 shows test data of invalid ECP for access control. Pre-state shows home page of user that is logged in. Inputs check the role of user and attempts from user to access a page. Here an unauthorized user e.g. Patient try to access the accounts page using external query which is not authorized to patient then the second input that is attempts increased by one and expected post state is ‘Role.Home Page’ with a message of ’Access Denied’. If the same user attempts more than two times then expected post state is ‘Lock Account’. Test cases against availability that the system should be available 24 hours and 7 days a week for its valid users are as follows. Table 7 shows test data of invalid ECP for availability. In test data of test case id TC021, a valid IP address generate five requests to access HIMS in time interval of three seconds. This activity from IP address count as malicious activity and IP will blocked. In test case ids TC022 and TC023 invalid IP addresses tries to access the system.
TABLE V
TEST DATA AGAINST INVALID ECP FOR AUTHENTICATION

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>Pre-State</th>
<th>Input</th>
<th>Expected Post-State</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC005</td>
<td>Get_Credentials</td>
<td><a href="mailto:admin@gmail.com">admin@gmail.com</a></td>
<td>12345 1</td>
<td>Login Page</td>
</tr>
<tr>
<td>TC006</td>
<td>Get_Credentials</td>
<td>labengineer Admin</td>
<td>2</td>
<td>Login Page</td>
</tr>
<tr>
<td>TC007</td>
<td>Get_Credentials</td>
<td>admin Pakistan</td>
<td>3</td>
<td>Login Page</td>
</tr>
<tr>
<td>TC008</td>
<td>Get_Credentials</td>
<td>malikhabibahm <a href="mailto:ad101@gmail.com">ad101@gmail.com</a></td>
<td>4</td>
<td>Blocked State</td>
</tr>
<tr>
<td>TC009</td>
<td>Get_Credentials</td>
<td>admin Pakistan</td>
<td>5</td>
<td>Blocked State</td>
</tr>
</tbody>
</table>

TABLE VI
TEST DATA AGAINST INVALID ECP FOR ACCESS CONTROL

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>Pre-State</th>
<th>Input</th>
<th>Expected Post-State</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC013</td>
<td>Role Home Page</td>
<td>Patient</td>
<td>1</td>
<td>Role Home Page</td>
</tr>
<tr>
<td>TC015</td>
<td>Role Home Page</td>
<td>Patient</td>
<td>2</td>
<td>Role Home Page</td>
</tr>
<tr>
<td>TC016</td>
<td>Role Home Page</td>
<td>Patient</td>
<td>3</td>
<td>Locked Account</td>
</tr>
</tbody>
</table>

TABLE VII TEST DATA AGAINST INVALID ECP FOR AVAILABILITY

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>Pre-State</th>
<th>Input</th>
<th>Expected Post-State</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC021</td>
<td>Web Browser</td>
<td>192.168.1.12</td>
<td>3</td>
<td>IP Blocked</td>
</tr>
<tr>
<td>TC022</td>
<td>Web Browser</td>
<td>194.168.1.13</td>
<td>4</td>
<td>IP Blocked</td>
</tr>
<tr>
<td>TC023</td>
<td>Web Browser</td>
<td>192.178.1.12</td>
<td>1</td>
<td>IP Blocked</td>
</tr>
</tbody>
</table>

Current paper not only formally specifies and validate authentication, access control and availability specific threats and their corresponding mitigation in Object Constraint Language but it also designs black box test cases through OCL expressions.

VI. CONCLUSION

Our research aim was to specify authentication, access control and availability specific threats and their corresponding mitigation in OCL and then to design black box test cases through OCL specification. Current research specifies authentication e.g. brute force and dictionary attacks, access control e.g. SQL injection/ URL manipulation attack and availability threats e.g. denial of services attack and their corresponding mitigation in OCL. Such formal specification in OCL will lead to unambiguous constraint specification. Using a plugin ‘Papyrus’ of tool Eclips, OCL expressions are appended on transitions as guard conditions of state diagrams. All OCL expressions were validated and we then applied mutation testing upon validated OCL.
expressions in same tool. Our second contribution is apply mutation testing to design test cases from all validated OCL expressions. We used Equivalence Class Partitioning for identification of valid and invalid classes against each variable in OCL expressions. Then using same classes test cases were designed and test data was generated.

VII. FUTURE WORK

One possible future work is that, one can automate same models to design test cases automatically. Besides one can also specify other security aspect like integrity and confidentiality through OCL and design test cases against same.

REFERENCES

[4] Braunstein, Cécile; Hartxhausen, Anne Elisabeth; Huang, Wen-ling; Hübner, Felix; Peleska, Jan; Schulze, Uwe; Vu, Linh Hong. (2014). Complete Model-Based Equivalence Class Testing for the ETCS Ceiling Speed Monitor, Formal Methods and Software Engineering. DOI: 10.1007/978-3-319-11737-9_25
[23] Saqib Iqbal a, Issam Al-Azzoni , Test case prioritization for model transformations Journal of King Saud University - Computer and Information Sciences Volume 34, Issue 8, Part B, September 2022, Pages 6324-6338