IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 4, 2019, accepted September 23, 2019, date of publication October 8, 2019, date of current version October 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946315

Test Patterns for Cloud Applications

SIDRA SIDDIQUI[1] AND TAMIM AHMED KHAN[2]
[1]Department of Computer Sciences, Air University, Islamabad 44000, Pakistan
[2]Department of Software Engineering, Bahria University, Islamabad 44000, Pakistan

Corresponding author: Tamim Ahmed Khan (tamim@bahria.edu.pk)

ABSTRACT Software systems are becoming graphical user intensive. They involve web technologies organized in the cloud platform which supports translation of services to a wider community. Such cloud applications are more vulnerable to misuse. Consequently, system development needs to focus on system security features in a comprehensive manner. Therefore, techniques that are based on test-driven development will be a good choice to use for the quality maintenance of such systems. We need checklists and mechanisms that provide identification and knowledge of best practices to maintain consistency in performing testing activities. We propose a test patterns-based technique which supports identification of test cases on the bases of specification and domain analysis of system under test. We provide a set of test patterns that support Test Driven Development (TDD) as well. We link misuse cases and security requirement to testing and provide test patterns for testing cloud applications. We consider threats associated with cloud applications and make use of case studies to evaluate and present results.

INDEX TERMS Test pattern, TDD (test driven development), misuse case, test last development (TLD).

I. INTRODUCTION

Secure software is the one that is developed on foreseen security issues and then tested thoroughly at the end. Techniques such as test driven development (TDD) focus on meeting the quality needs in small cycles of development while Test Last development (TLD) enforces the detailed testing after development. We either use TDD (Test Driven Development) or TLD (Test Last Development) approaches, each new update in the development procedure of software introduces new issues. These issues need to be focused while still maintaining the quality of the system. Testers are continuously engaged in identification of recurring common issues and directing solutions across several softwares. Pattern based testing provides models to handle recurrent situations in testing such as, testing access control across different softwares. Whereas, patterns are packages of reusable knowledge that can be used to solve problem supporting reusability [1]. More specifically, patterns are useful to define something that is recurrent,and to describe repetitive behavior and their associated solution [2]. In terms of testing, patterns are strategies used to conduct testing which can be combined with existing patterns. They are test templates or test patterns that have context, intention, situation, action and some results in the form of test case development, or test case execution. Previously proposed approaches focus on effective implementation, patterns of structure, method and ways of creating systems as proposed in [1], [3] and [4]. All of these techniques help an experienced developer in improving their system. However, identification of loopholes that can cause system attacks is more important. Furthermore, standardization needs an approach which is experience independent, easy to use and provides more problem specific solutions. To model a standardized solution a predefined issue modeling and solution template is also required which will help to identify the issue in system under consideration. Considering all these aspects we propose an approach which provides detailed description of pattern structure and how it supports testing. In order to establish the discretion structure of our technique we have considered the concept of misuse case [5], [6]. Misuse case is a concept that is being extracted from use cases and inherits all descriptive methods of the use case concept [5]. Use cases provide the scenarios of positive use of systems, while misuse cases provide the opposite perspective. They give information related to situation of use which should be avoided [5]–[7]. The identification of use cases and their associated misuse case helps the developer to identify several threats at an earlier level of development else they may be overlooked [8]. A related concept known as security use-case is also common for identification of security requirements.

The associate editor coordinating the review of this manuscript and approving it for publication was Maurizio Tucci.

Misuse case and security use case deliver two different information that is, misuse case gives threat related information and security use case gives information related to mitigation. However both the information are interconnected with each other for system security [6]–[8]. Security requirements need identification and formulation of goal, risk and requirement. These three actions are strongly related to use cases and misuse cases for threat identification. On the other hand, the information of threats helps to foresee future issues. As we proposed in our test pattern technique one can train the system against future issues by applying threats as test cases. As misuse case rotates around threats identification and mitigation, they can support in testing too. As our proposed technique connects misuse cases, threats and testing, it provides support to identifying test case on the bases of misuse case. Test patterns, on the other hand, can also identify the misuse case requirements or security requirements for the system under test (SUT). This two directional application approach, as proposed, makes it compatible with conventional testing techniques and test driven development model(TDD). The details of concepts are given in further section of paper. In order to precise our research we have considered current cloud related threats. We have considered threats related to data, resource and interfaces [9], [10] and connected them to test patterns. Furthermore, social engineering aspect will be considered in the later research. This paper is the extension of our previously published work [11]. In the previous publication, we have provided the over view of the proposed pattern and its general template. This paper provides the detailed distribution of proposed pattern, revised general template of patter definition, associated test case, general formula to compute expected test case, detail case studies and their findings. Moreover, this paper provides the details of our finding and the outcome of our research. We have used the case study respective to demonstrate the usefulness of our approach

II. PURPOSE OF RESEARCH

The quality of software is identified based on testing strategies. Testing the software means analysis of the situation which can cause its failure. To get the knowledge about state of art about pattern-based technique, we conducted a survey. Currently, several researchers use pattern from designing, interface and defects to suggest pattern or strategies for testing. We also found out that the concept of test patterns has not been practically tested. Moreover, the description of actual pattern is missing in the literature. Moreover, a bi-direction solution is also needed.

As cloud software s are combination of data bases, platforms and internet, we believe that cloud pattern is a good source of identification of testing strategies in cloud software. Currently, none of the research focuses on cloud pattern base testing. Cloud patterns are a good source to analyze system reaction in the presence of vulnerability. Moreover, the vulnerability simulation on cloud environment acts as an executed test case for it. Each misuse case (threat) is hence connected to its associated testing. This will bridge the gap between situation analysis and testing prediction. Furthermore, it will also help in evolution of test patterns. Currently we shall focus on security related issues. Furthermore, we shall simulate each situation and describe how they can be used as a test case. Based on this information we will explain how cloud pattern can be used for predicting test pattern or testing strategies. Moreover, it will provide support for both *TDD* and *TLD*.

The main aim of this study is to identify the relationship between patterns of recurrent situations and testing strategies based on them which provide a bidirectional application. We are planning to identify the security situation which can help us in identification of condition of testing. We plan to connect misuse case to our approach. This connection will support early identification of required testing. Moreover, it will provide support to identify the associated threats of the systems. Designing a system while considering its associated threats will ensure secure development. This two-way connection of our technique will make it applicable for both test driven development and conventional test last developmental approach. We found that, security threat is a test case of security strength of system. On the other hand, it is the misused security requirement which need to be considered during development. We have used this to facilitate prediction and identification of testing pattern based on cloud pattern. Moreover, we have tried to bridge the gap between testing and appropriate solution identification. In order to establish research, we consider following research questions:

1) Are there existing patterns on the bases of which testing processes are conducted?
2) What type of relationship exists between (1) data security, (2) pattern in cloud and (3) test cases?
3) How does the threat documentation handle the misuse case?
4) How does cloud security act as a testing strategy to secure itself?
5) What test patterns can be identified from the study?

III. LITERATURE REVIEW

Patterns are explained as the packages of reusable knowledge [11]. This knowledge can also be described as common efforts to solve problems, which support reusability [1]. More specifically the pattern in software is used to define something that is recurrent, whether it is to describe some repetitive behavior, problem and their associated solution, trends of occurrence of related items or structural recurrence [2]. C. Alexander has described patterns as the combination of environment, problem and solution. In terms of testing, patterns are defined as strategies which are used to conduct testing. These strategies of testing are combined with existing pattern and defined as test templates or test patterns or pattern base testing. In all these techniques, the test is based on some pattern which have; context, intention, situation, action and some results in the form of test case development, or test case execution. From the literature, it has been found that such testing based on pattern, or test pattern can be described into 5 classes; Security, GUI, Design, Defects and Testing. The following paper is intended to describe them in detail

A. SECURITY

To model security situation an extension of use case named as UMLsec has been proposed in [12]. Observability of these security situations help in identification of expected misuse cases for the software. A general procedure to elicit security requirement from the misuse case is proposed in [5]. Similarly, in [13] authors have conducted a comparison between misuse case and BIBIS technique. They have claimed that the misuse case is an effective way to identify security requirements. Early identification of security requirement helps in identification of expected threats and helps in testing. To address security issues which arise recurrently, the developers use patterns. These patterns are packages of knowledge with reusability which help in dealing security situations [1]. These security or attacker patterns are described as the combination of; <Goal, precondition, action, post condition >. Goal is expressed as the reason for the attack. Precondition describes the situation which needs to be present for the attack to execute. Action is the sequence of event. Whereas, post condition describes the situation after the attack has been conducted [1], [14]. A classification survey of these pattern based on attributes, objectives and quality parameters are proposed in [15]. They have also described that the use of pattern at designing level is difficult because detailed information of attacker and vulnerabilities is limited. They have suggested that, there is a need to bridge this gap for effective selection of patterns. The security is compromised when the software faces an attack. Identification of these attacks at an early level can also bridge the gap of security pattern selection and helps in testing the software [14].

Similarly, like another pattern base approaches through exploitation of security or analyzing attacker patterns help in identification of expected testing. As in [14] authors have suggested a method of modeling attacker patterns and then using them for testing. They have supported the security modeling through UML state machine and identified attacker patterns from them. They have examined two attacker pattern XSS and SQL. These patterns are then used for testing. Similarly, in another paper [18] authors have suggested to use the XSS attack model based on vulnerability attacker patterns. The patterns are modeled in the form of UML. The basic focus of the technique is to define the test case. The authors have suggested that the test input is executed and when it matches the stored ones, the test is considered as positive. On the other hand, in the case of negative response on the test, the retest is conducted.

B. DESIGN

Designing patterns are supporting solutions for recurrent designing issues. Identification of appropriate patterns at the class diagram level is the researchers focus. Several techniques have been proposed to automate the process. A model-based technique to identify designing pattern have been proposed in [3]. They have used semantic query-enhanced web rule(SQWRL) language to identify properties and their relation to search the expected pattern with the help of extensible markup language(XML) base class diagram. Testing of the right implementation is important for overall software. A technique named as âĂIJpattern test case templateâĂİ (PTCT) is used for identification of structural test case for these patterns. (PTCT) is described as reusable test cases. Such technique has been discussed in [1]. Authors have described that, PTCT can be used for identification of implementation mistakes in designing patterns, which would lead to ultimate software quality. They have also described that, structural similarity among the software using a particular pattern could help in identification of recurrent errors. The situations which leads to the bug will be the test cases for ensuring the reliability of implementation.

On the other hand, the problem of anti-pattern arises when the designed pattern is wrongly implemented in the program. Anti-patterns are recurrent solutions for recurrent problems but wrongly implemented [16]. A GUI base unit testing of anti-patterns was proposed in the paper [17]. They have suggested that anti–pattern-based testing techniques reduce the testing cost.

C. GUI

Graphical interactions of software usually contain some recurrent interactions, known as UI patterns. These UI pattern share similar implementation and can be used to identify required testing requirements. UI Test patterns is defined as a testing strategy which has some specific set of testing. UI patterns are defined as a combination of goal, behavior, action, checklist, and pre-condition [18], [19]. The definition of these attributes, according to the paper, are expressed as Goal representing mostly test id, Behavior in terms of input pairs (i.e. valid or invalid), Action as the sequence of action been performed for testing, Check list as the reason of testing and Preconditions required to be fulfilled in order to perform the testing.

On the bases of these UI test pattern several researchers have suggested similar techniques for testing software under test. Such as, in [18] authors have explained that the similarity among UI patterns when used to conduct testing, the required testing requirements were identified by modeling in Paradigm DSL language. The models were then used to generate test cases. In another paper [4] authors have suggested the use of pattern base testing for mobile application. This technique focuses on identification of pattern in terms of behavior which it exhibits recurrently in android applications followed by testing strategies associated with the test pattern. Furthermore, reverse engineering is used. An *FSM* model is created, and depth first search is used. The actual results are then stored while exploring the behavior and is then compared with the pattern expected behavior with actual results. However, no practical application has been considered. On the other hand, in [19] authors have described the use of PBGT in android application and conducted a case study.

D. DEFECT

An understanding of the defects is the core requirement behind testing. Developers current focus is to predict pattern of defects in the project defect reports. In [20] authors have discussed this current focus. They have extracted defects from the black box testing using test reports. On the bases of 10 important defects they have defined 4 kind of frequent patterns in them. These patterns describe the defects, situation, severity level and relation among defects.

On the other hand, there is another research proposed in paper [21] which conducted an experiment and used machine learning and semi supervised learning to predict the defects. They have examined that how in the absence of historical data module of the software can be used to predict the defect of the overall software. They have also claimed that sampling with semi-supervised learning is better than the conventional machine learning technique.

E. TEST PATTERNS

Test suit parameterization is a concept used to select reusable testing strategies for testing. More specifically it is described as test pattern. A more formal description for test pattern is proposed in - [22] in which authors have described that like designing strategies, testing strategies can also be defined in the form of patterns. They have proposed a template of testing pattern which is the combination of 9 elements;

<Name, Type, Scenario, Problem, Solution, Implementation, Solution, Implementation, Effect, Example>

Each pattern is described by name, its category in terms of type, situation in terms of scenario, problem and its associated solution. It also gives importance to defects which could cause challenge during pattern implementation. Moreover, it also considers the advantage and disadvantage of each pattern. The pattern definition also describes the examples of pattern and the relation of one pattern to the another.

F. CLOUD PATTERNS

Cloud computing external attacks are termed as threats or vulnerabilities. These external attacks have affected the strength of cloud computing. A taxonomy of potential threats based on 4 layers; infrastructure layer, platform layer, application layer and administration have been proposed in [23]. Authors have described 23 threats related to these categories. These threats are affecting the security of cloud. In order to handle these threats several researchers have described patterns. These patterns describe the occurrence situation of vulnerability, their context and solution. Such as, in [9] patterns related to firewall are proposed. Authors have described that fire wall is the boundary for all data incoming and outgoing of cloud. They have proposed Cloud Web Application Fire wall pattern (CWAF) to handle SQL injection and another data vulnerabilities. In another research proposed in paper [24] researchers have proposed pattern of security virtual machine repository and cloud policy management point. In another research [25] authors have described countermeasure pattern and thread patterns in cloud. They have described countermeasure patterns for cloud provider to handle threats. Furthermore, they have described the relationship of pattern and counter pattern.

The term of misuse case is often used to describe negative handling of systems. The threats of clouds are considered as its misuse case. As in [26] authors have described cloud vulnerabilities in term of misuse case. They have described that how an authorized user could create misuses case. They have explained that an authorized user can also create a malicious image. When this image is accidently being used by another user it can create several misuse cases. They have claimed that understanding of these misuse case can facilitate testing and right implementation. However, among cloud vulnerabilities data security is the most common malicious handling of data in cloud as proposed in [9]. Among these cloud patterns the security attack pattern related to cross site scripting and SQL injection as described in [14], [27] have been used for testing which are described in detail under security.

G. TEST DRIVEN DEVELOPMENT AND TEST LAST DEVELOPMENT APPROACH

Test driven development is the way to start development in the backward direction. Writing the testing strategies first then develop code to full fill the criteria is the basis of Test Driven Development (TDD). TDD is expressed as Test First Development Approach(TFD)by authors in [28]. TDD supports the refactoring process of code development. It also provides automatic support for the regression testing. As codes are modified with each success or failed test, a cycle of test then develops, and property provides a short cycle of development [29]. On the other hand, the conventional auditing of software involves testing the software on completion. A generic term used to refer such method is Test Last Development(TLD). TDD and Test Last Development approach have some variations and similarities. Several researchers have studied TDD and compared it with traditional approaches(TLD) to enlighten these aspects. A comparison of TDD with traditional developmental procedures of SDLC which relies on TLD with respect to qualitative and quantitative characteristics have been proposed in [23]. They have expressed that TDD have major support of reusability. Whereas, flexibility, effectiveness, risk reduction is highly improved under TDD. Moreover, the complexity and rework cost have been reduced while using the TDD. The productivity and response to stakeholder need to have a better response in TDD as suggested in [9]. Moreover, they have stated that, TDD have long term benefits regarding defect density. On the other hand, in another paper [28], authors have compared TDD and TLD on the bases of branch coverage, mutation score, quality of test case and test code for the purpose of test quality. They have identified that these dimensions have shown that there is no relative difference between the two approaches. Moreover, they have additionally stated that

TDD have shown more biasedness towards positive test cases. However, the effectiveness of the testing is more influenced towards negative test cases. Since the TDD is a development approach so this deviation does not affect the actual motive of testing. Similarly, in [30] authors have compared TDD with TLD approach and identified that there is no enhanced difference, especially for external qualities there is no deviation.

TDD is an emerging field. It depicts that it has variation in its views about the two approaches. TDD helps building behavior based test that model system. On the other hand, test last implement test which is executed in the last. However, it is evident that both the approaches have benefits. The research which we have conducted provides benefits to both the TDD and last approaches. As user-oriented analysis of software help to identify user perspective base needs it also supports the understanding of the expected threats. The application of proposed techniques will support in establishing requirement, in turn they will help in identifying the use case and associated misuse case. As misuse case give information on how a system can be used wrongly, they also provide supported information on expected threats that can affect the system. These threats in turn are connected to test pattern thereby, identifying test cases and represents a forward approach. On the other hand, as we already have the test pattern with associated test case, these test patterns guides the required systems need and functionality while considering the threat. These functionalities will then be needed to implement software with strength against the associated test. Hence in this case it supports the Test First approach. Our approach has an additive advantage as it is considering the negative test cases which are the core for the testing procedure efficacy.

H. MISUSE CASE

Using the system with negative intentions or deliberately accessing the information or service that is out of the domain of user access is considered as misuse or abuse case of system usage. The understanding of the user accessions and rights support the developer to embed the required security measure. A more formal method to address the security measure is security requirements [31]. All of these concepts are interrelated because they support each other in their development. There are several methods proposed by researchers for defining use case, misuse case and the methodology of extracting security requirement through their help, as proposed in [5], [7]. In [32] authors have proposed an approach based on decomposition of misused case to support functional and security requirement incorporation. They have discussed the threat and mitigation relationship to support integration of requirements. In [6], [33] authors have used the term of abuse case to describe the threat based on use case (misuse case) for security requirement elicitation. They have discussed that presence of such abuse cases support designing, development and above all testing. Misuse case also supports the analysis of quality of the system. In [34] authors have proposed misuse case-oriented quality requirement engineering. Their proposed model uses the concept of misuse case to identify quality requirements along with the security requirements. They have explained that the documentation of threat in the form of misuse case not only supports identification of issues but also provides support to develop preventive measures. Misuse cases are executable in the absence of the safety measures, where one can only describe the mitigation measures if he or she is aware of risk/threats associated with the system. Similarly, misuse case provides support for risk management systems. As in [35] they have described the association of misuse case for security risk management. They have discussed the model for information system security risk management for the explanation of misuse case. The reference model as described in the paper shows the relationship of risk management, event, threat, mitigation etc. with each other. They have discussed that the integration of both concepts enhances the use. Majority of the associated misuse cases are similar so they can be used across projects. This similarity is because they depend on the threat associated with the system. Since threat for similar system remains the same, the reusability of misuse case is enhanced. Such e-commerce and m-commerce websites have threats which are common in each website application. They usually involves attacks for instance, flooded system, get privileges, steal critical information and propagation of malicious code [5]. As misuse case provides supports for threat information, they are a useful source for their testing purpose. One can test the misuse cases on to the system to check the system strength against the expected mishandling. To define testing based on misuse case it is needed to use misuse cases as executable testcases. A similar approach was proposed by [36]. Connecting misuse case with information such as attack pattern or attacker identification deliver results in two directions. As in our approach the execution of the threat as the test case supports the testing. On the other hand, as the use of misuse case rotates around the threat identification and threat mitigation, they can support testing. Since the test pattern approach as we proposed can be executed in the presence of threat information, one can identify the expected testing need for the system if he has knowledge of the misuse case. This will be beneficial for the developer and the tester as both can link the testing and the requirements on the bases of test patterns. This two-way connection will also provide support to TDD and TLD as discussed previously.

I. DISCUSSION AND COMPARISONS

The limitation of security pattern base technique is that, it does not consider other security vulnerabilities. Furthermore, it only considers those test cases which give positive results. Moreover, it does support testing by providing information of frequent attacks; however, little information is given about the test strategies and the model which they are using for automatic testing [14]. On the other hand, there are many designing patterns such as strategy, factory etc. which are not considered in the design pattern bases testing [1]. Moreover, this approach is based on PTCT which needs full understanding of code and structure of software.

Hence, the technique is more applicable for statistical analysis but not for dynamic analysis. Contrary to design bases testing GUI pattern base technique does not need this information. However, it also has some limitation. According to [18] most of the errors are related to searching and the relationship between GUI elements however, there is little information on the selection criteria for this output finding. In order to create general point of view independency is required; however, technique depends on paradigm base model for its results. The actual strength of technique is visible when it is experimented in the real world; unfortunately, the given research is limited to the students' practices. None of the above techniques have considered the defects which are the core part of testing. However, in paper [20] authors have identified defects and predicted testing. Unfortunately, the limitation with this technique is that; it uses the ODC-BD classification techniques for the classification however, it does not provide the causality relationship between the defects. Moreover, it also does not provide any information that how it can help in predicting the defects. Secondly, it depends on the historical data that is usually difficult to find.

The historical data limitation is being covered by technique proposed in [21] by moulding the current software under test module to predict defects of overall software. However, this technique acquires the data from the open source which challenges its validation for the industrial use. Furthermore, they have applied a random sampling for defects which require sampling of each individual with equal probability. This will further affect the importance of right selection of sample important for appropriate estimation.

Lastly the technique proposed in - [22], unfortunately does not describe its practical use. Moreover, to support the idea strong literature review is needed which is limited in the paper.

IV. OUR APPROACH

In this section, we present our approach of test pattern and describe its flow, resultant patterns and the associated test cases. Furthermore, this section provides details regarding approach meta-data. We shall also present the details of a general formula which we have developed to calculate expected number of test cases under proposed patterns. The details are discussed in the following subsections.

A. APPROACH FLOW

We consider cloud vulnerability as proposed in [9], [10] and suggest a general structure or test patterns. Our approach follows 4 phases; feature extraction, mapping, testing and verification. Each phase provides input to the next phase in the increment. When this method is applied to the TDL approach, the input is the implemented software and its specification. The approach on its execution will provide the enhanced software. On the other hand, when approach is applied as TDD it will provide enhanced specification and secure system. The overview process detail is shown in fig. 1.

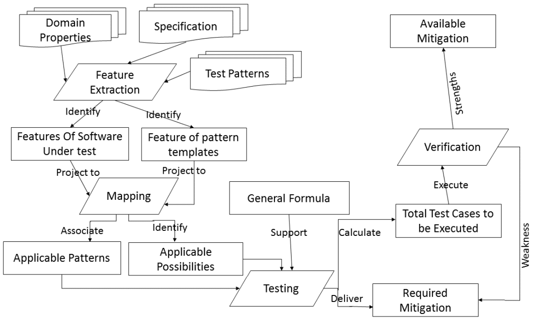

FIGURE 1. Approach flow showing stages and connection.

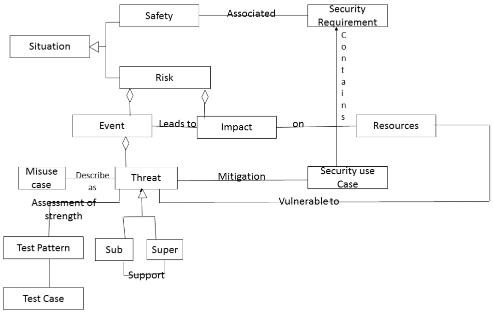

FIGURE 2. Test pattern meta model.

The approach can also be executed iteratively as it supports regression testing. The detail of the approach is explained in the following section:

1) FEATURE EXTRACTION

This phase takes the domain properties, software specification, implemented software specification and pattern information. These inputs are analyzed in detail and the feature of software under test and the pattern being considered are identified in fig. 1.

2) MAPPING

Identified features are then projected to mapping phase fig. 1. The mapping phase takes the unique feature of patterns and the software properties and then identifies applicability of pattern to the software under consideration. The possibilities in which a feature/resource can be affected is governed by the sub categories associated with test pattern fig. 2. The subcategories are also modeled in different ways and depend upon the type of features which are provided for pattern application.

3) TESTING

In this phase, we consider the general formula to calculate the expected number of test case and the output from the previous phase. The phase provides the total test cases and identifies the expected mitigation. The expected mitigation are the possibilities in which a system can be affected. The possibilities of attacks are the test case to be executed to ensure the strength of the system as shown in fig. 2. If the framework is applied on the TDD approach, then it will stop at this stage or else proceeds to the next phase as shown in fig. 1.

4) VERIFICATION

When the test case is executed the actual level of system the strength is identified. Strength level identifies the missing mitigation and required mitigation. At this stage a system which fails on each attack, expresses its limitation against the pattern. In case of already completed software, this phase supports feature enhancement and supports regression testing for next iteration

B. META MODEL OF TEST PATTERNS

Test pattern also has association with the concepts associated with threat cause and mitigation. These concepts provide support in emergence of the test pattern concept. In order to show how they are related with each other we have proposed the meta-model of test pattern. The detail of the model is explained in the following section:

Situation: Situation is the state of the system in which it is currently executing. It can be a safe state, or it can be risky. The situation of use defines the approach by which the system is addressed. In a safe state the system has or will associate the security requirements. On the other hand, the risky situation of use identifies the negative use of the system.

Security requirement: To demonstrate the required security parameter security requirements are associated with safe state. In security requirements, there are 5 basic steps; critical asset identification, security goals, threat identification of goal, risk and finally the requirements [5]. The security requirements as described in the security goals models the security use case. Security use case models the mitigation for the threat as shown in fig. 2.

Security use case: It is preventive measure of misuse case. Threats are prevented through security use cases which defines the preventions against misuse.

Risk: Risk is defined in terms of events and the impacts. Event which is executed have a positive or negative impact over the system resource. The resource is the system properties which it uses or provides.

Events: Events are the action of attacking the system. These actions when associated with threats are expressed as misuse case. On the other hand, misuse cases are used to describe the threats.

Impact: Each event when executed has some positive or negative impact. The event impact is on resources. The resources are the system properties used or provided.

Resources: The resources are the properties system delivers or the resources which it uses.

Threat: Threats are the attacks which affect the system. Each event when executed has associated threats which support the accomplishment of the risk. These threats are indicated through the misuse cases. These threats are further sub divided into sub and super threats. The division is previously explained in detail in the section of test pattern. Resources which have susceptible properties are vulnerable to these threats. Only those resources which are vulnerable have negative impact on them during malicious event execution.

Test pattern: Test pattern gets the input from the threats. Test pattern supports the threat identification. The threat identification in turn supports the misuse case definition and explanation. The bi direction information illustrates the *TDD* and *TLD*.

C. OVERVIEW OF PATTERNS

We study 37 threats [9], [10] for development of test patterns. In these patterns, we have taken only those which are easy to achieve in the lab and are related to data, resource and interface. The patterns which we have not considered are malicious insider, privacy breach, natural disaster, side channel attack, reliability of data calculation carried out, misuse of infrastructure, hardware theft, migration of virtual machine, breach of contractual rights, sanction due to political issues, unknown risk profile and other social engineering attacks. The attacks which we consider are those threats which possess describable properties, which express their occurrences or non-occurrences and have manageable cost of testing and visible outputs list given in Table. 3. These threats are further subdivided into main and sub categories of threats. Sub category threats are low level attacks which cause threats of associated patterns and they are also termed as low-level risks. These attacks include Denial of Service (DOS) attacks, unencrypted interfaces, session hijacking, resource contention, data interruption, target modification, access limit and trust level on shared VM environment, hypervisor compromises, insecure interfaces, shared technology issue, and discontinuity of external resources, incomplete transactions etc. These threats cover a broad spectrum and are associated with host, platform, application, infrastructure and administration. Threats remain the same; however, they have some variation in parameter of effect at different levels. Examples include (DOS) attacks at network level which will have a similar effect at application level and session hijacking at application have impact at the data level and it also depends on data level issues for its successful completion. Moreover, the subcategories may appear as sub categories in multiple patterns. The multiple association is due to their multi-dimensional usage across attack execution. We present our categories, subcategories, positive and negative situations of use in Table. 3.

Template for each pattern must contain the following attributes:

Name: It is used to identify test pattern.

Defect Type: It indicates types of defects; after effects of attacks are described as defects types.

Test Pattern Type: It includes type of testing and the relationship of test with the development level.

Goal: What testers intend to achieve from this type of testing.

Sub category: Sub category threats are the low level attacks which cause the threats of the associated patterns.

TABLE 1. Test case 1.

Test case	Example
Input 1	Run Select * data
Output: Data	Output: Data
Input 2	Run Malicious Select Query
Output:	System response Generate no alert
Output:	Data System Critical Data
Expected OutPut	Data
Actual Output	Data
Success scenario	Expected = Actual

TABLE 2. Access limit and trust level on shared VM environment.

Name	Access limit and trust level on shared VM environment
Defect types	User management, Access rights control, Access control
Test pattern	Non-functional level after the cloud application is fully functional.
GOAL	Test access limit for users. Test System management for user rights. Test the control mechanism.
Sub category	Resource usage limit functionality, system response to unauthorized user for critical data,
Situation	Authorized user can access unauthorized data and can modify it, Malicious user can manipulate critical shared data.
Target	Check the user rights division on shared critical data. Check the user access to critical and shared data. Check the authorization level for critical data modification.
Actions	• **Access user profile:** As different users and check response. • **Access critical data:** as different users and compare the two access authorization levels. • **Modify critical data:** as different users and compare the two modification levels rights.
Success criteria	

These attacks are the basic reason which lead to the pattern threat.

Situation: Actual issue to be tested (i.e. situation of use).

Target: Issues that are expected to be revealed, through targeting system part and system responses.

Action: Sequence of action needed in order to perform the test. These actions are the base for each subcategory in the form of ACTION 1 and ACTION 2. Otherwise if the actions for subcategory interact with each other in the joint fashion they are considered as one set of consolidated ACTION attribute.

Success criteria: Arrive at required attack.

Under each pattern we execute test case. Those test cases are governed by some input, output, actual output, success criteria. The result of success criteria will define the success of the test case. The general template consists of following elements:

Inputs: Number of inputs vary depending on the required field.

Expected Output: How the system should respond or Expect messages.

Actual Output: Required mitigation, required responds or Expected message.

Success Criteria Actual output = Expected Output

The number of test case for each pattern depends on number of possibilities in which its associated subcategories can be formulated. The detail about the testing under pattern is given in other section. To test our pattern, we run the test case. In the test case as shown in the Table. 1, we first access the system, select the required target field and then we choose the required data query. We first choose the valid select query to get data status and check the system response. Then we again select the required query. This time we have selected data non-valid query, executed the query, compare our expectation.

1) VOLUME LIMITATION

In the volume limitation attacks the system resource limitation is targeted.When resources and the user relationship when get unequal, system faces issues. Strength of system is identified by the limit of the numbers of users it can support over minimum resource availability. A user request which acquires all the resources of the system leads to unavailability of its services. In the case of volume limitation attacks DOS plays an important role to achieve such system failure hence we consider it as key lower level attack. As it populates bandwidth with request from one or more suspicious users. As a result, other legal user is unable to get the system resources. Such attack can affect the system if it lacks system ability to provide alert on reaching resource limitation. Moreover, Lack of control over number of request limit from one user could lead to attack.

2) TARGET MODIFICATION (INTERCEPTING AND, MODIFYING MESSAGE

In the following test pattern or target manipulation required service are temper leading to malicious services. Malicious user induces the unwanted services, alerts or other websites. In the worst case it induces the malicious codes which adversely affect the client users. XSS (cross site scripting) scripts provides a source to achieve such attacks. Malicious hypertext markup language strings support XSS (cross site scripting) and other similar vulnerabilities. All these attacks are inter connected to achieve the issue of target modification (intercepting and, modifying message) therefore they are considered as subcategory of this pattern. The outcome of the defect is visible in the form of defect types. The problem which leads to such defect is the proper user query analysis before executing it.

In this test pattern required services are tempered leading to malicious services. Unwanted services, alerts or other websites are induced. In the worst case the malicious codes which are being induced can adversely affects the client/user. XSS scripts provides a source to achieve such attacks. Malicious html strings support XSS and other similar vulnerabilities.

All these attacks are inter connected to achieve the issue of target modification (intercepting and, modifying message) therefore they are considered as subcategory of this pattern. The outcome of the defect is visible in the form of defect types. Such defects are possible because of improper user query analysis before its execution. The pattern is shown in the Table. 9.

3) SESSION HIJACKING

In the following test pattern of session hijacking, brute force profile login, access after session expiry, cookie data access are the low level attacks which lead to the required attack. Hence, they are subcategory. The outcome of the defect is visible in the form of defect types. Lack of system alert in the presence of cookies are the basic cause for such problem. If such alerts are there the user can deny the auto saving of passwords and other sensitive data. System refreshing mechanism after expiry of session could control this issue. Pattern is shown in Table. 10.

4) ACCESS LIMIT AND TRUST LEVEL ON SHARED VM ENVIRONMENT

In the following test pattern of access limit, resource usage limit functionality, system response to unauthorized user for critical data, are the low-level attack hence they are subcategory. The outcome of the defect is visible in the form of defect types. The system is being affected by the following threat because of lack of user level access rights implementation. This does not maintain any support for the limit of data been accessed by the users. Moreover, system needs a mechanism to define different classes of user. The pattern is shown in Table. 2.

5) HYPERVISOR COMPROMISES

The management of user and their profile are the key factors for managing authorization in SOA system. Increasing demands of the data integrity control enforces proper server and client rights and access control management. An improper and weak system can cause the propagation of data corruption over the network. A strong hypervisor is needed to avoid such issues. Lack of intrusion deduction and prevention are the source to compromise the hypervisor and could support attacks by virus or adverse data corruption. Rootkit similar other software provides support for such attacks. Therefore, lack of intrusion deduction, prevention and rootkit infection are the subcategory. Defect type in the pattern indicates the actual issue that can cause the attack on the system such as, improper intrusion detection as shown in Table. 5.

6) DISCONTINUITY OF EXTERNAL RESOURCES

The system response on the discontinuity of the supporting system is important for the system security in terms of its integrity of data and properties. The following threat can be achieved if connection is lost before saving information, data server connection lost during storage, incomplete transaction, hence these are the low-level attacks which combine to reach the attack therefore they are considered as subcategory. The lack of monitoring of amount of data been written or manipulated leads to threat associated defect types which come into existence due to above mentioned pattern success as shown in Table1 4.

7) INSECURE INTERFACES

Insecure interface is created because of different loop holes in the interface. These loop holes are considered as low-level threats for the interface security. Hence, they contribute to overall insecure interfaces. Visible passwords, lack of data flushing after each use, lack of control over API's use and data access limit, lack of logging mechanism for each usage are those low-level threats that lead to insecure interface hence they are considered in subcategory. Defect types illustrates the lack of proper implementation of interface security measures. Pattern is shown in Table. 12.

D. TESTING UNDER TEST PATTERN

Proposed approach covers the security aspect of the system. Subsequently, all testing conducted under the approach are non-function testing. Each pattern provides variety of test cases to administer the associated pattern. All the proposed general test cases are governed by categories, sub-categories and associated actions as proposed in each pattern. The test case is designed considering the possibilities in which a sub categories could be modeled as a test case. The number of test case of a *SUT* is identified by availing pattern and their associated availing possibilities. The details of possibilities are as under:

Data security pattern:

- Access basic information.
- Generating request for change of data.
- Not Notification on change of data or request of data.
- Lack of feedbacks.

Session Hijacking:

- Lack of data flushing after logout.
- Weak mechanism of refresh.
- Brute force access.

Insecure Interface:

- Visible passwords or other critical data that needs to be kept hidden.
- Lack of refresh mechanism.
- Lack of state refreshing.

Access Limit and trust level:

- Change user authentication level when not needed.
- Illegal rights manipulation to modify data.
- Illegal handling of data by unauthorized users.
- Modify data with wrong intention.

Volume Limitation:

- Misleading user request.
- Over usage of illegal access to system to reduce user access limit.

TABLE 3. Pattern mapping overview.

Category	Sub-Cat	Positive/Negative use
Data security testing, Data interruption	SQL injection, updating without back up	• Legal access of required data to avail system service. • Access data for illegal use by exploiting system services. • Modify data to reflect new update.
Volume limitation	DOS attacks, Connection flooding, Resource usage max limits.	• Multiple users accessing system. • Using unavoidable service by multiple users at the same time. • Generating multiple request to affect system integrity • Authorized user (un)able to access the system.
Target modification (intercepting and modifying message)	XSS Attack, Redirection, invalidated input with Html encoding disable.	• Participate in multi user activity • Redirecting system to affect integrity of system, trust of users, posting infected reply to infect associated users.
Session hijacking	Brute force profile login, Access after session expiry, Cookie data access	• Authentication mechanism. • Keeping session information, transaction tracking, keeping track of states during long interaction. • Stealing user information to hijack user account, to manipulate user resources and to generate illegal act on behavior of user.
Access limit and trust level on shared VM environment	Resource usage limit functionality, system response to unauthorized user for critical data	• Different user rights and access levels, sharing file, data and information on common resources and shared files, Multiple files and coordinated users and their access rights.Data access levels with different access controls. • User role enforcement management. • Authorized user with unauthorized data access for modifying, malicious user can manipulate shared data.
Discontinuity of external resources	Connection lost before saving information, Data server connection lost during storage, Incomplete transaction.	• Back up storage and long transactions. • Center data point. • A legal user is trying to store information but suddenly internet connection is lost. • A user has used the system success fully however, the data center fails to store the data.
Hypervisor compromises	Lack of intrusion detection and prevention, Rootkit, Data integrity verification	• Running plug-in on servers or server driven installation of software needing antivirus independent environment. • Multiple format files uploading and sever file corruption or virus attack infect the other guest user. • An admin access with a malware component, breaching the security and run malicious code on root system • Intrusion to manipulate all files on server and client system.

TABLE 3. *(Continued.)* Pattern mapping overview.

Category	Sub-Cat	Positive/Negative use
Insecure Interfaces	Visible passwords, Lack of data flushing after each use, Lack of control over API's use and data access limit, lack of logging mechanism for each usage.	• Login interface to access profiles. • Keeping track of information for specific time duration to improve user experience. • Third party interaction for payments or SOA architectures. • Unencrypted password can lead to access with unauthorized intention, Non-Flushing of data fields lead to unwanted access to system. Over provision of system control through the API's to the user, compromises the system security by allowing access to critical data. • Lack of logging mechanism, unwanted third party access
Discontinuity of external resources	Connection lost before saving information, Data server connection lost during storage, Incomplete transaction.	• Back up storage and long transactions. • Center data point. • A legal user is trying to store information but suddenly internet connection is lost. • A user has used the system successfully however, the data center fail to store the data.

TABLE 4. Discontinuity of external resource.

Name	Discontinuity of external resource
Name	Discontinuity of external resources.
Defect types	Media connection loss, Network connection lost while data updating, Data management server connection lost while updating information.
Test pattern	Non-functional level after the cloud application is fully functional.
GOAL	Check system response on connection lose age. Check system response on incomplete data transaction.
Sub category	Connection lost before saving information, Data server connection lost during storage, Incomplete transaction.
Situation	A legal user is trying to store an information but suddenly internet connection is lost, A user have used the system successfully however, the data center fail to store the data.
Target	Test system transaction logging and roll up procedures. Test system response to user on incomplete request. Test system response over data center failure.
Action	• Initiate the saving process and in the meanwhile shut down the system during saving. • Restart the system and resigning the user. Check if the system stores the data or not. • Check if the system on restart inform user about the incomplete data storage if occurred. • Use the system, make changes in it and then save it and log out. • Fail the data center between logging out and saving. • Check weather system informs about the user to the problem through message or email. • Check whether it automatically saves the data or not.
Success criteria	System lost the data updating from the user side, System does not inform the user about the problem and user assume the success of its process.

Hypervizor Compromises:

- Virus or corrupted data prorogation.

Discontinuity of external resource:

- Lack of retry mechanism.
- Lack of feedback mechanism on loss of data.
- Power supply discontinuation.
- No back up mechanism.
- No notification mechanism.

On the bases of this information we have proposed a formula which support calculated the total number of test

TABLE 5. Hypervisor compromises.

Name	Hypervisor Compromises.
Defect types	Intrusion detection management, Cloud Virtual environment management.
Test pattern	Non-functional level after the cloud application is fully functional.
GOAL	To test the cloud virtual environment management. To test the control of propagation of attack. To test the cloud effect of server attack on others.
Sub category	Lack of Intrusion detection and prevention, Rootkit, Data integrity verification is absent.
Situation	Server file contain malware code(Shellcode) file hence infect other systems.
Target	Check host system security response toward rootkit attack. Check host system response to ward intrusion.
Action	• **Access the system.** Log in as different groups of user profile. • **Access data:** Shared and unshared resources as different users. • **Place malware files:** At server and client system. • **Check system intrusion detection software response on**.
Success criteria	Virus or malware is propagated on all systems.

cases to be executed. The details are given in the following section.

1) GENERAL FORMULA FOR EXPECTED TEST CASE

In order to compute the associated test case for each pattern we propose the following formula. We propose that for each pattern test case there are two variables which determine its test case to be executed. The two variables are; availing sub category and associated possibility. Possibility means number of ways it can be achieved or demonstrated. Two test cases one from the negative class and one from the positive class are associated with each possibility. The summation of all associated possibility test cases is equal to total of the test cases in a sub category. Whereas, summation of all sub category for a pattern is the total number of test cases under that pattern. Sum of all the pattern test case describes the total test case executed on a system.

$$SC = \sum AP = 1^n AP * 2_(AP) \tag{1}$$

$$TSC = \sum SC = 1^n SC_(SC) \tag{2}$$

$$P = \sum AP = 1^n TSC_(AP) \tag{3}$$

V. EVALUATION

A. OVERVIEW OF IMPLEMENTATION

In order to demonstrate the applicability of our proposed technique, we conduct case studies of real time projects. We analyze system requirement specifications and extract features of each project after analysis. The features define requirements, management, abilities and core properties. These properties include attributes such as, authentication, managing multi users, multiple file, feedback system, back up storage and reboot mechanisms. We also perform test pattern analysis along with the feature analysis. Over all approach is shown in fig.1. The specific test cases are governed by associated possibility and general template for test pattern test case. We have applied our approach to two types of systems fully developed and underdeveloped.

1) EXPECTED MINIMAL TEST CASE

TABLE 6. General test case.

Pattern	Subcategory total number	Possibilities	Expected Test cases
Data security and interruption	3	5	10
Volume limitation	3	1	2
Target modification	2	1	2
Session hijacking	3	5	10
Access limit and trust on shared VM environment	3	4	8
Hypervisor compromise	3	3	6
Insecure interface	4	5	10
Discontinuity of external resource	3	3	6
Total	SC=23	AP =27	TC = 54

2) DOMAIN OF PROJECTS SELECTED

We have divided website usages into four categories. Categories are; social network, data storage, business management, information management and information dissemination. We have selected one project from each category represented in Table. 7.

TABLE 7. Domain and cases.

Domain	System
Information management	Moodle
Social network	Facebook
Business Management	Pocketdesk
Data storage	Dropbox
Information dissemination	www.cochraneventilation.com

Moodle, Facebook, Dropbox, www.cochraneventilation.com are fully functional systems. Where as, Pocket desk under developmental phase.

B. CASE STUDY 1

1) DOMAIN

Information management and the website we have considered is Moodle.

2) OVERVIEW

Moodle provides its user the information management. It manages multiple user and organize roles. Privileges are allocated on the bases of roles. Current version of Moodle which is been considered is a course management system. System has three types of users; teachers, students and the administrators. Teacher can avail the feature to create a course, upload documents, modify document, enroll student, generate email to all student. Teacher can manage multiple subjects. On the other hand, students can generate a request to enroll in a subject. Students can download the file. Each user manages its own account.

3) PROPERTIES / FEATURE/SERVICES PROVIDED

- Login
- Role management
- Access limit management
- Information dissemination Dashboard
- Uploading /downloading
- Email propagation

Feature: Login System provides individual login for each user. Teacher and administrator have privileges over the features. Teacher can manage multiple subjects under its profile. Student can access resources from the shared content through their profile login.

Feature: Role management System offers three major roles; teacher, student and administrator. Each role has different level of privileges. Privileges determine the access rights of the roles. Administrator is responsible to manipulate the role of teacher or student against any user.

Feature: Information dissemination Dashboard Admin have the access to all the data of the system related to his employee profiles. Admin can change any data from the dashboard or reset the user or passwords without any additional authentication.

Features: Resource management, Uploading /downloading System provides a resource management for its privileged users. Teacher can upload, download files. These file on later bases used by other resource such as students.

4) ASSOCIATED THREATS

- Abusive use of Role: Role can be reset by admin.
- Abusive use of Resource: Can upload malicious file or can access critical data especially in case of teacher role can access all students ids.
- Unencrypted password: As usual admin provided password are designed on the bases of organization id's and if remain unchanged by user can be easily accessed.
- Insecure Interface: Open access to user profile without additional authentication.
- Data security issue: Password or user name reset does not generate any notification to the user. Multiple time access to the user profile does not generate any notification. No notification of system been login through different system.
- Brute force access: As passwords and Username are easy to predict.
- DOS Attack: As server gets hang up on multiple users during exams as many student or teachers are access resources.
- Unawareness of critical data manipulation: In term of admin all information is accessible and in terms of teacher all student information including picture or phone number is access able.
- Malicious content propagation: no filter on data been uploaded. Malicious file from teacher or student can affect multiple systems.
- Backdoor installation or uploading possible: As no file filter is there one can upload a hidden malware.
- Malicious content uploading. As no file filter is there one can upload a hidden malware in simple word or jpg file.
- Malicious content propagation: As files can be upload or access if delivered as share data
- Unawareness of change: No notification or email system for data modification or network connection lost feedback is there.

5) ASSOCIATED PATTERN

- Insecure interface
- Data security issue or data interruption
- Access limit and trust level on shared VM environment
- Volume limitation
- Discontinuity of external resource
- Hypervisor compromising
- Session hijacking

6) ANALYSIS

We have applied 18 test cases and conclude as following:

- Data security issue and Target Modification issue: Vulnerability does exist as feedback mechanism is missing so updating is not notified. Anyone with knowledge of user id and previous password can change the profile. No notification mechanism; user is unaware of changes in his profile. Vulnerability exists. User passwords are not managed properly hence and easily be predict and users profiles can be accessed.
- Insecure Interface:Security of password encryption is maintained.However, user profile give full data on login lack of two-factor authentication vulnerability is there.
- Session hijacking:Session are properly refresh after 2 min of inactive interaction.
- Hypervisor compromising: Weak share machine and data scanning.
- Access limit and trust level on shared VM environment: vulnerability is there. Admin user can edit profile without the information of user. Users have less control over their own data as compare to admin. Lack of notification can results in to loss of important contacts in the list. Vulnerability is present. Lack of communication of data updating can cause data inconsistency across different users. Vulnerability is there role management should be model with detail authentication procedure. Rights of different users should be mange to maintain security of critical data.
- Volume limitation: Vulnerability is there system is unable to handle too many quires.

TABLE 8. Volume limitation.

Defect types	Volume limitation
Test pattern type	Non-functional level after the cloud application is fully functional.
GOAL	Test the system failure limit on bandwidth volume. Resource management. To check the limitation of amount of user that can be functional at one time.
Sub category	DOS Attacks, Connection flooding, Resource usage max limit.
Situation	Large number access to SOA system can lead to over flow of the user limit, leads to problem that authorized users are unable to access their data. Long processing time also effectse SOA system response. Redirecting to wrong webpage can lead to several issues.
Target	Subject the system to either ping attack to prevent DOS attacks in the future or to maximum volume to test its failure limit. Test by applying means to change system response. .
Action	• Access the url. • **Ping the website:** To get IP Address • **Get the response:** In this case IP Address. • **Choose the request type.** • **Chooses the number of request.** • **Ping the website againwith:** Use IP Address and concatenating space, -t (i.e. command for ping the system until stops), -l (i.e. command for sending buffer size), space, number of packet data. • **Get the response:** measure performance and efficiency/ response time/through put after the attack. • **Compare expected and actual response:**pre attack response and after attack system response
Success criteria	Response time is too delayed . Get a pop up message that website not available.

C. CASE STUDY 2

1) DOMAIN

Social network and the website under consideration is ‘‘Facebook’’.

2) OVERVIEW

This online website is the network of users. It provides user connectivity and manages their social network. It provides facility of personal profiles for each independent user. It offers social group and pages facility to support forums of multiple discussion.

3) PROPERTIES / FEATURES/SERVICES IT PROVIDES

- Login.
- Manage multiple users.
- Manipulating responsibilities.
- Shared forums.
- Initiate multi user discussion.
- Upload/ Download file.
- Upload /Download file on shared forums.
- Provide server storage.
- Edit personal profile data.

Feature: Login

Website provides independent login facility for each user. Users under one profile can manage one personal profile and multiple shard groups, forums and pages. The user can manage multiple contacts under login. If user is the owner of the page, group or forum he manages the user access limits. Moreover, user can administer the authentication of other user for its owner ship. A website interface is provided for login facility to avail the feature. Each individual profile supports updating of profile associated information.

Feature: upload / download file, upload /download file on shared forums

Website supports the file sharing facility to its users. These files are shared on the individual base and local group bases or publicly. These file specifically the picture and video support the profile information unit.

Feature: Manage multiple users, manipulating responsibilities, shared forums, and Initiate multi user discussion.

Website manages multiple users with independent profiles. In shared forums the holder of the forum can manage the roles of the user. Holder have the right to apply restriction to the client of its forum. The holder manages the content sharing properties. His/her friend list editing is in his or her hand. Moreover, each user of the profile manages information regarding its profile. Every user has the right to manipulate its profile information and can restricts its dissemination.

Feature: Edit personal profile data

User manages his or her profile. User have the right of its information dissemination selection from the provided property level of security in user hand.

4) ASSOCIATED THREATS

- Unauthorized access.
- Profile hijacking.
- Brute force access.
- Insecure interface.
- Unencrypted password.
- Unawareness of critical data manipulation.
- Access limitation and trust level on shared VM environment.
- Malicious content propagation.
- Unaware of changes.
- Manipulation of critical data by unauthorized access.

5) ANALYSIS

Feature and the identify threats are mapped to the Table. 3 to identify the applicable test pattern. The identified pattern is shown in the following section. On the base of the available feature we have mapped the test pattern and executed the test case. We have used SqlMap a penetrating testing tool to support our testing for data security issue or interruption pattern. We have run 18 test cases and found that it incorporated defense against most basic problems, as Passwords were encrypted, session are managed, user on share data are managed properly and notification is properly in place. However, there are few issues such as, a malicious link through notification can infect users. Virus can easily propagate on share communities. User has to reload in case of uploading of file issues.

D. CASE STUDY 3

1) DOMAIN

Data storage and the website under consideration is Dropbox.

2) OVERVIEW

The online web site provides the data storage facility to its users. User can manage their file of different types in cloud storage. Users can share files by providing privileges to other users. Website also provide and offline storage drive facility to its users.

3) PROPERTIES / FEATURES/SERVICES PROVIDE

- Login
- Manage multiple users with access rights.
- Resource management for shared resources.
- Upload / Download file.
- Provides Server storage.
- Edit user profile data.
- Notification system
- Offline backup storage.

Feature: Login

Website provides independent login facility for each user. User can store file of different types in one storage area. Stored files can have private access or shared. Access to a particular file is allotted on the bases of user access rights defined by owner. The access has different level of manipulation rights. User can use the allocated space for group discussion and work depending upon the access it allowed to other users.

Feature: Resource management Website provides resource management to each user. User can store multiple files of different types in multiple organizations. User can download or upload any file. The resource can be edited by the allowed users. The owner can manage the level of manipulation on to the document.

Feature: Manage multiple users, managing multiple access level rights Website manages multiple users. Website offers multiple types of access level for different users. User can define the access level to its private folder. User can customize the level of manipulation on its shared data. Website offers the facility to share data between user of the website and even non user. Owner of the content decides the access level and sharing level of the resource. The user which can edit the file have the right to manipulate the member access except the owner of the resource. In case of team work admin have the right to sign in to the team member account.

Feature: Upload/ download file, Provide server storage, upload /download file in shared resource

Website offence the facility to upload / download data from the server. However, during upload disconnection leads to loss of fie previous uploading information. One has to initiate the uploading transaction from the beginning. However, website offers a good data feedback for the information of failed and uploaded file.

Feature: Shared resource.

Resource owned by a user can be shared among different users. The owner of the resource is the ultimate administrator of the resource. The set the properties of access to the share folder. Owner can also hand over the owner ship to other users.

Feature: Offline backup storage

Website offence support of offline back up storage. User can download the drive and can upload the file in it. Website automatically configure the changes from the offline storage. Similarly, backup storage automatously configures the changes made online.

Feature: Feedback

Website comprises of good feedback mechanism. Website delivers feedback to the user on access to any critical data that user has managed.

Feature: Login

Website provides independent login facility for each user. User can store file of different types in one storage area. Stored file can have private access or shared. Access to a particular file is allotted on the bases of user access rights defined by owner. The access has different level of manipulation rights. User can use the allocated space for group discussion and work depends upon the access it allowed to other users.

Feature: Resource management

Website provides resource management to each user. User can store multiple files of different types in multiple organizations. User can download or upload any file. The resource can be edited by the allowed users. The owner can manage the level of manipulation onto the document.

Feature: Manage multiple users,and managing multiple access level rights

Website manages multiple users. Website offers multiple types of access level for different users. User can define the access levels to its private folder. User can customize the level of manipulation on its shared data. Website offers the facility to share data between user of the website and even non user. Owner of the content decides the access level and sharing level of the resource. The user which can edit the file have the right to manipulate the member access except the owner of the resource. In case of team work admin have the right to sign in to the team member account.

Feature: Upload/download file, Provide Server storage, upload /download file in shared resource

Website offence the facility to upload / download data from the server. However, during upload disconnection leads to loss of the previous uploading information. One has to initiate the uploading transaction from the beginning. However, website offers a good data feedback for the information of failed and uploaded file.

Feature: Shared resource.

Resource owned by a user can be shared among different users. The owner of the resource is the ultimate administrator of the resource. The set the properties of access to the share folder. Owner can also hand over the owner ship to other users.

Feature: Offline backup storage

Website offence support of offline back up storage. User can download the drive and can upload the file in it. Website automatically configure the changes from the offline storage. Similarly, backup storage automatically configures the changes made online.

Feature: Feedback

Website comprises of good feedback mechanism. Website delivers feedback to the user on access to any critical data that user has managed.

4) ASSOCIATED EXPECTED THREATS

- Unawareness of share data manipulation: No notification or email system for data modification by author
- Brute force access.
- DOS Attack
- Malicious content propagation: As anyone can upload or access share data
- Unaware of changes on shared data: If no notification is generated

5) ASSOCIATED PATTERN

- Target modification (Intercepting and, modifying message):Folder modification by author to any access able share data without notification other audience
- Volume limitation
- Access limitation and trust level on shared VM environment.
- Target modification: As many users can have access to share data.
- hypervisor compromising: Files on share folders can affect the users.
- Session hijacking
- Discontinuity of external resource: If offline mode is not update along with online data inconsistency can arise,

6) ANALYSIS

On the basis of features identified, we mapped patterns and executed the associated test case. The detail of case study is shown in the table. We have used Sqlmap a penetrating testing tool to support our testing. We have used SqlMap a penetrating testing tool to support our testing for data security issue or interruption pattern. We have in total executed 19 test cases. We have found that Dropbox have in placed all necessary requirement to make the system secure. However, test cases related to file share, prorogation of malicious file, notification and offline and online data consistency have shown some gray area.

TABLE 9. Target modification.

Name	Target modification (intercepting and Modifying message)
Defect types	Data Mishandling for malicious manipulation and misdirection.
Test pattern	Non-functional level after the cloud application is fully functional.
GOAL	To check the system response on a data security attack. To check system response on input validation. To check system response on HTML encoding. To check the system response on scripts input. To check the system response on an attempt to modify results in order to get access to data.
Sub category	XSS Attack, Redirection, invalidated input with Html encoding disable.
Situation	Malicious user can manipulate the website and can induce unwanted alerts. Service redirection can lead to reduction of trust. Information could be extracted by scripting a cookie that well deliver the user information to the malicious user.
Target	Subject the system to HTML with tags. Subject the system with HTML tag without validation of input. Subject the system with java scripts.
Action	• Access the system. • **Choose the Field**. • **Submit data with malicious format**: Data with HTML tags /java scripts. • **Check system response.** • **Disable**: Input validation. • **Repeat steps from** 3 **to** 4. • **Disable**: Html encoding. • **Repeat steps from** 3 **to** 4. • **Compare expected and actual response.**
Success criteria	Pop up message. Website redirection. Receive information in cookies.

- Target modification (Intercepting and, modifying message): Owner can change the data without notification to other sub users chances of data inconsistency.
- Access limitation and trust level on shared VM environment: Multiple role with privacy in place for all no not vulnerable. Sub users need to access permission on accessing own file not vulnerable.
- hypervisor compromising: Files on share folders can support transfer virus to others.
- Session hijacking: Refresh session after specific duration not vulnerable.
- Discontinuity of external resource: If offline mode and online need constant refreshing of system.

E. CASE STUDY 4

1) DOMAIN

Domain is business management and the application considered is "Pocket desk".

2) OVERVIEW

Organization employ task management system. System manages employ task assignment on the bases of skill set and past records. Moreover, it supports employ task tracking through location intelligence and millstone completion records.

System also provides support for leave management. Employees can share data with each other and can perform live chat for issue resolutions. System is been backup by data base which is update online.

3) PROPERTIES/FEATURES/SERVICES PROVIDED

- Login
- Role management
- Information dissemination Dashboard
- Uploading /downloading
- Live Information exchange
- Notification system
- Location tracking

Features: Information forms.

Delivers information regarding services is provided by the organization and associated event schedules. The overall website is designed to deliver information which is non query based.

Features: Login:

System provides login facilities for users. During Login user can select two type of user access Admin or Employee. User can log in with any email address of his choice.

Features: Role management:

User with Admin role can access to employee location, and personal profile information like name, phone number, picture etc. On the other hand, the employee does not have the access to other employee data. However, if the admin selects an employee as team lead then he or she have the access to the data of his/her subordinates.

Features: Information dissemination Dashboard:

Admin have the access to all the data of the system related to his employee profiles, current tasks and leave of other employees. Admin can change any data from the dashboard.

Features: Uploading /downloading:

Admin can assign task and can upload file for the task. Moreover, he can update the user profile for the task assignment. On the other hand, employ can upload the leave form.

Features: Live Information exchange:

Employees can exchange view through live chat, they can also share files with each other.

Features: Notification system:

Whenever a new task is assigned notification is generated in the notification center for each employ.

Features: Location tracking:

Location of each employ can be tracks by the admin.

4) ASSOCIATED EXPECTED THREATS

- Abusive use of Role: As user can set his or her role on login
- Abusive use of Resource: As user can access data of other employ
- Abusive use of Privilege: As user can set his or her role on login
- Unencrypted password.: As password is shown

TABLE 10. Session hijacking.

Name	**Session hijacking**
Defect types	Invalid access to authorized user profile by taking session id, accessing critical data by using cookies between server client response, unauthorized user masking as an authorize user.
Test pattern	Non-functional level after the cloud application is fully functional.
GOAL	To check the cookie security, system response on Time out, system response on unauthorized access to user profile, system response on brute force URL logging after session expiry, limit of data been expose to our application user and system response strength to sniffing of user data cookies.
Sub category	Brute force profile login, Access after session expiry, Cookie data access.
Situation	Sniffing information of user through cookie. Session alive even after the user log out could lead to unauthorized access important data. Brute force prediction of profile address. Without user knowledge saving user credential could lead to data leakage in future.
Target	Check restriction on cookies and on client side scripts to access information. Checking response of cookies flags
Action	• **Login the profile**: Giving credential / URL of the profile been copied from last login. • **Check system response**: Login or Access not granted. • **If access is granted by either of the options from step 2**: Follow next steps or repeat from 1 to 3. • **Submit a data retrieval query and check system response:** It uses the cookie to keep track of results of a user. • **Submit a suggestion for product request**. • **Check the cookie field.** • **Check Http table:**Table is not check and http is not appended in the cookie header. • **Check to access the cookie information**. • **Logout the system.**
Success criteria	Access profile after logout using the URLs. Access user information from cookie even after session expiry.

- Insecure Interface: Open access to critical data. Missing priority of type of data that can be seen
- Data security issue: Data is accessible easily no notification or email system for data modification feedback is there
- Unawareness of critical data manipulation: No notification or email system for data modification feedback is there
- Backdoor installation or uploading possible: As no file filter is there one can upload a hidden malware in simple word or jpg file.
- Malicious content uploading.: As no file filter is there one can upload a hidden malware in simple word or jpg file.
- Malicious content propagation: As anyone can upload or access share data
- Un awareness of change: No notification or email system for data modification or network connection lost feedback is there.

5) ASSOCIATED PATTERNS

- Insecure Interface: As the Dashboard for profile and information available

TABLE 11. Test case example.

Data	Detail
Input	Set login user as admin
Input	Access Employer profile
Expected Output	Only name, skill and work performance available
Actual output	Whole profile can been access and can be edit
Success criteria	Actual output=Expected
Pass / fail Criteria	Failed (Vulnerability exists: access roles can be modified to access critical data)
Argument	Vulnerability exist access roles can be modify to access critical data

- Data security issue or Data interruption: As Data base update, modification is available through interface.
- Target modification (Intercepting and, modifying message): As Multiple user have access to critical non-shared and shared data.
- Access limit and trust level: As different users with different roles and needs are present the system.
- Discontinuity of external resource: As Network, GPRS and server hard disk connection are important for system execution.
- Hypervisor compromising: As Server manages the data exchange between different users and types of application.
- Session hijacking: Multiple users

Example test case from the case study is shown in Table. 11.

6) ANALYSIS

We have run in total 16 test case across different test patterns and it was found that system have defects related to:

- Insecure interfaces: As password are visible.
- Access limit and trust level on shared VM environment: Vulnerability exist access roles can be modified to access critical data. Rights of new user needs to be managed to avoid unlawful updating.
- Data security issue or Data interruption: Vulnerability exist any one can modify without information.
- Session hijacking: Vulnerabilities of lack of session refreshing.
- Hypervisor compromising: Vulnerability is there no file update or access notification, or version control exist. Vulnerability exist no file filter exist
- Discontinuity of external resources: Vulnerability exist employ have no information of connection lost. Vulnerability exist employ have no information of connection lost.

F. CASE STUDY 5

1) DOMAIN

Information dissemination and the website we have considered is ‘‘www.cochraneventilation.com’’.

TABLE 12. Insecure interface.

Name	Insecure Interfaces
Defect types	Interface security in terms of data encryption at interface level. Input interface data flushing is important for security. Loose interface control and management of data flushing could lead to accidental and malicious attempts.
Test pattern	Non-functional level after the cloud application is fully functional.
GOAL	To check the management, monitoring and controlling level/mechanism of interfaces and API's. Understand their security trust level. To Check the data flushing and error logging mechanism. Check the third party usage of service.
Sub category	Visible passwords, lack of data flushing after each use, lack of control over API's use and data access limit, lack of logging mechanism for each usage.
Situation	Unencrypted password steal by user can lead to authorized access with unauthorized intention, Non-Flushing of data fields lead to unwanted access. Over provision of system control through the API's to the user, compromises the system security by allowing access to critical data. Lack of logging mechanism, Unwanted third party access.
Target	Test system data encryption, flushing mechanism. Test access type, limit and involvement of third party. Test logging mechanism.
Action	• **Check data encryption mechanism:** Checking that critical characters are visible while typing or not. • **Check system flushing and refresh mechanism:** Log out the system and click the signing without credentials. • **Check interface or API level of authorization and access control to user: (1)** Check the Information provided to the use of the system through the API. • **Check system log keeping mechanism:** Check the system log file after every usage. • Check the policy agreement and access rignts of third party used by the system.
Success criteria	Lack of encrypted, system allows logging by just clicking signing with no data in fields after first logout. Third part have access to critical data, agreement shows a full authority over data hence limiting data integrity and user trust level.

2) OVERVIEW

Website is used for information dissemination purposes. Website provide information regarding services offered by the firm. Website gives information regarding event and help which it provides. Website only deliver information.

3) PROPERTIES/FEATURES/SERVICES PROVIDED

- Information forms.
- Links to other resources.

Features: Information forms.

Delivers information regarding services provided by the organization and associated event schedules. The overall website is designed to deliver information which is non query based. **Features:** Links to other resources

4) ASSOCIATED THREATS

- Data security: As website maintains information
- Data tampering: As data changes can affect others
- Volume limitation: As many users can reach website one time

5) ASSOCIATED PATTERNS

- Data Security issue testing or data Interruption
- Volume limitation

VI. DISCUSSION: TECHNIQUE ASSOCIATIONS

As it can be seen from the case studies that test patterns provide two directional outcome in-terms of pre establishment of software requirement and test case for validation. Such as, Pocket Desk and Moodle. In Pocket Desk it has supported the developers to understand the level of system security they have in place in their requirements and services during development. This have supported them to improve their system and incorporate missing facilities, hence resulting in improvement of system services than its previous versions. Applying our approach on Moodle, facilitated the organization to identify issues in their existing system. This shows its applicability for software improvement and development techniques such as TDD. Furthermore, as it facilitates the recognition of misuse cases it supports the reduction of likely hood of exploitation of system by guiding the user to incorporate the required security requirements. In addition, it correspondingly offers help by examining risk and recommitting suggestions to avoid it. In the following section we explain in detail the relation between our proposed technique and TDD, misuse case and risk.

A. TEST DRIVEN DEVELOPMENT

Model base development adheres the system according to established models for the domains. Test pattern provides the general templates for the testing. The test pattern general template and the specific templates governs the development of the system in the model driven development manner. System when designed considering the pre specified threat, they are designed with care. The development is then governed by measures which prevent the identified threats. The identified threats are defined in term of the misuse case. However, the mitigation of these misuse cases is achieved through the security use case governed by security requirements. As defined in our approach threats are associated with test patterns. These pattern and test case model the threat. Threat information supports to define the functionality of the system which is strong enough to sustain the attack. This supports provision of use cases and misuse cases. This backward development of the system from test pattern are define as Test First Developmental approach (TFD) or Test Driven Development (TDD). Test pattern is a TDD approach as it provides information through testing data. TDD supports the refactoring of the code. Each iteration of testing in TDD increases the maturity of the code been developed. The iterative nature of TDD supports the regression testing. TDD helps construction of test built on the behaviors to model the system. Analyzing the system and applying the testing to check the system adherence to requirements is expressed as Test Last Development (TLD) approach. Test pattern along with TDD supports TLD. To ensure the system strength test pattern provides the assessment criteria. Test pattern and established system properties are matched to identify the applicable test pattern. Execution of associated test case assess the level of system security requirement implementation. A negative or positive response to the negative intention test case depicts the presence or absence of pattern. The success of pattern implementation delivers the information regarding the weakness present in current functionalities. The connection of the misuse case the test pattern provides support to identify the applicable test at early level of development. Test pattern is the bidirectional approach. It provides benefit of both TDD and TLD. The overall cycle of test pattern relation with TDD and TLD is shown in fig. 3 and fig. 4. The connection of the misuse case and the test patterns support this additional benefit. With the information of misuse case one can identify the required testing. On the other hand, with the information of test patterns and associated test case one can identify the expected misuse cases. A meta model fig. 2 shows that how different element regarding test pattern, threats, event, impact, situation, etc., are associated with each other the detail of the model is expressed in the next section.

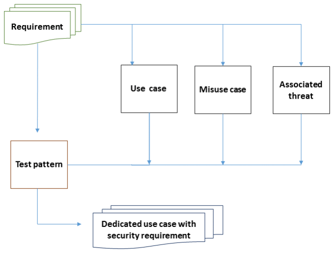

FIGURE 3. Test pattern cycle relating TDD.

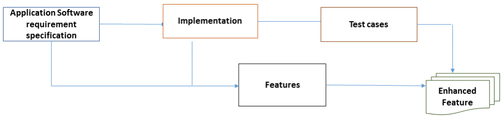

FIGURE 4. Test pattern cycle relating TLD.

B. TECHNIQUE AND ASSOCIATED RISK MANAGEMENT

1) RISK AND IMPAIRMENT

When system is properly tested over its vulnerabilities, the chance of being attacked are reduced. On the other hand, if the system evolution is not being performed chances of system attack increases. In the presence of possibilities that can lead to system failure we express them as risks of system. Risk of system are usually defined along with the event of mapping of that risk and the impact it generates [35], [37]. Event that effects the system usually uses or exploits the resources. The resources are the properties which software system uses or provides. Every effect of the system propagates through property exploitation. An event when executed have associated threats. Threats are the attack which are launched on to the resources. These attacks can only be launched if the resource are vulnerable. Non vulnerable properties cannot be manipulated by attacks or un secure event. Threats are further sub divided into sub and super attack to achieve the final goal of impairment as shown in the diagram fig. 2. Super attacks are supported by low level sub attacks. These attacks exist for each system according to their properties. Similar domain system has similar attacks. The attacks are modeled in the test patterns which when executed identifies the system strengths. Misuse case as threats are executed as test cases against the stated test patterns.

2) RISK IDENTIFICATION

Utilization of system properties to make negative progress are communicated as misuse case. Misuse cases exploit the system properties with negative intensions. Misuse cases are articulated to deliver the information regarding the threats. Misuse case are also articulated as threats to express system vulnerabilities as in [26], [38]. Threats and impact joint together to achieve risks as shown in fig. 2. Test patterns are used to distinguish the related danger which a system can encounter. These patterns help in distinguishing proof of the hazard which can influence the properties of the system. Mapping of the test patterns and their related experiments support to convey necessities which are concentrated on the reduction of hazard properties. This technique supports the test first philosophy with the assistance of risk identification, threats and misuse case alliance to propose appropriate test case to generate secure system requirements.

3) RISK AVOIDANCE

Situation of safe state is associated with security requirements. Security requirements contain the security use case. Use case defines the threats and these threats are handled through these security use cases. Security properties as define by security requirements are needed to establish for secure systems. The model supports risk avoidance by identifying the associated risk at early level of development. The model of approach supports the TDD as it is giving the test base model as test pattern to identify the associated threat factor of the system properties. Early identification helps in designing the measures for the avoidance of the risk situations. Pre establishment of security requirements for the vulnerable property strengthens the system vulnerable properties.

VII. CONCLUSION

We study previously identified cloud application threats and provided a mapping of categories, sub-categories, positive and negative situations of use. This helps us in identifying test patterns which can be applied to individual situations to achieve quality in a smooth and repeatable manner.

We provide a generic template which is helpful in finding out test cases required for testing a system under test. We map feature associated threats and identify patterns for each case study. We then use pattern information to identify test cases. We then make use of comparison table to illustrate association between these elements. Patterns provide a two-way connection with misuse cases and threats. It supports identification of misuse case in the presence of test cases and test cases in the presence of misuse case. This two-way property results its support for test driven development and test last development. We evaluate our approach through case studies. Our proposed patterns facilitate identification of threats and expected test cases in a systematic manner.

REFERENCES

[1] N. Soundarajan, J. O. Hallstrom, G. Shu, and A. Delibas, "Patterns: From system design to software testing," *Innov. Syst. Softw. Eng.*, vol. 4, no. 1, pp. 71–85, 2008.

[2] N. Yoshioka, H. Washizaki, and K. Maruyama, "A survey on security patterns," *Prog. Inform.*, vol. 5, no. 5, pp. 35–47, 2008.

[3] M. Thongrak and W. Vatanawood, "Detection of design pattern in class diagram using ontology," in *Proc. Int. Comput. Sci. Eng. Conf. (ICSEC)*, Jul./Aug. 2014, pp. 97–102.

[4] I. C. Morgado, A. C. R. Paiva, and J. P. Faria, "Automated pattern-based testing of mobile applications," in *Proc. 9th Int. Conf. Qual. Inf. Commun. Technol. (QUATIC)*, Sep. 2014, pp. 294–299.

[5] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Eng.*, vol. 10, no. 1, pp. 34–44, 2005.

[6] P. Hope, G. McGraw, and A. I. Antón, "Misuse and abuse cases: Getting past the positive," *IEEE Security Privacy*, vol. 2, no. 3, pp. 90–92, May/Jun. 2004.

[7] G. Sindre and A. L. Opdahl, "Templates for misuse case description," in *Proc. 7th Int. Workshop Requirements Eng., Found. Softw. Qual. (REFSQ)*, Interlaken, Switzerland, 2001, pp. 1–13.

[8] G. Firesmith, "Security use cases," *J. Object Technol.*, vol. 2, no. 3, pp. 53–64, 2003.

[9] S. M. Hashemi and M. R. M. Ardakani, "Taxonomy of the security aspects of cloud computing systems—A survey," *Networks*, vol. 4, no. 1, pp. 21–28, 2012.

[10] M. M. Alani, "Securing the cloud: Threats, attacks and mitigation techniques," *J. Adv. Comput. Sci. Technol.*, vol. 3, no. 2, pp. 202–213, 2014.

[11] S. Siddiqui and T. A. Khan, "On test patterns for cloud applications," in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Islamabad, Pakistan, 2016, pp. 57–62. doi: 10.1109/FIT.2016.019.

[12] J. Jürjens, "UMLsec: Extending UML for secure systems development," in *Proc. Int. Conf. Unified Modeling Lang.* Springer, 2002, pp. 412–425.

[13] N. Ikram, S. Siddiqui, and N. F. Khan, "Security requirement elicitation techniques: The comparison of misuse cases and issue based information systems," in *Proc. IEEE 4th Int. Workshop Empirical Requirements Eng. (EmpiRE)*, Aug. 2014, pp. 36–43.

[14] J. Bozic and F. Wotawa, "Security testing based on attack patterns," in *Proc. IEEE 7th Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Mar./Apr. 2014, pp. 4–11.

[15] A. K. Alvi and M. Zulkernine, "A comparative study of software security pattern classifications," in *Proc. 7th Int. Conf. Availability, Rel. Secur. (ARES)*, Aug. 2012, pp. 582–589.

[16] A. L. Correa, C. M. Werner, and G. Zaverucha, "Object oriented design expertise reuse: An approach based on heuristics, design patterns and anti-patterns," in *Proc. Int. Conf. Softw. Reuse*. Berlin, Germany: Springer, 2000, pp. 336–352.

[17] H. Kaur and P. J. Kaur, "A GUI based unit testing technique for antipattern identification," in *Proc. 5th Int. Conf.-Confluence Next Gener. Inf. Technol. Summit (Confluence)*, Sep. 2014, pp. 779–782.

[18] R. M. L. M. Moreira, A. C. R. Paiva, and A. Memon, "A pattern-based approach for GUI modeling and testing," in *Proc. IEEE 24th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2013, pp. 288–297.

[19] P. Costa, A. C. R. Paiva, and M. Nabuco, "Pattern based GUI testing for mobile applications," in *Proc. 9th Int. Conf. Qual. Inf. Commun. Technol. (QUATIC)*, Sep. 2014, pp. 66–74.

[20] N. Li, Z. Li, and L. Zhang, "Mining frequent patterns from software defect repositories for black-box testing," in *Proc. 2nd Int. Workshop Intell. Syst. Appl. (ISA)*, May 2010, pp. 1–4.

[21] M. Li, H. Zhang, R. Wu, and Z.-H. Zhou, "Sample-based software defect prediction with active and semi-supervised learning," *Automated Softw. Eng.*, vol. 19, no. 2, pp. 201–230, 2012.

[22] W. Yi-Chen and W. Yi-Kun, "The research on software test pattern," in *Proc. Int. Conf. Future Comput. Sci. Appl. (ICFCSA)*, Jun. 2011, pp. 109–113.

[23] S. Kumar and S. Bansal, "Comparative study of test driven development with traditional techniques," *Int. J. Soft Comput. Eng.*, vol. 3, no. 1, pp. 2231–2307, 2013.

[24] E. B. Fernandez, N. Yoshioka, and H. Washizaki, "Patterns for cloud firewalls," in *Proc. AsianPLoP*, Tokyo, Japan, 2014, pp. 1–11.

[25] T. Okubo, Y. Wataguchi, and N. Kanaya, "Threat and countermeasure patterns for cloud computing," in *Proc. IEEE 4th Int. Workshop Requirements Patterns (RePa)*, Aug. 2014, pp. 43–46.

[26] E. B. Fernandez, R. Monge, and K. Hashizume, "Two patterns for cloud computing: Secure virtual machine image repository and cloud policy management point," in *Proc. 20th Conf. Pattern Lang. Programs*, Oct. 2013, p. 15.

[27] J. Bozic and F. Wotawa, "XSS pattern for attack modeling in testing," in *Proc. 8th Int. Workshop Autom. Softw. Test*, May 2013, pp. 71–74.

[28] A. Cauevic, S. Punnekkat, and D. Sundmark, "Quality of testing in test driven development," in *Proc. 8th Int. Conf. Qual. Inf. Commun. Technol. (QUATIC)*, Sep. 2012, pp. 266–271.

[29] K. Bajaj, H. Patel, and J. Patel, "Evolutionary software development using test driven approach," in *Proc. Int. Conf. Workshop Comput. Commun. (IEMCON)*, Oct. 2015, pp. 1–6.

[30] D. Fucci and B. Turhan, "A replicated experiment on the effectiveness of test-first development," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Oct. 2013, pp. 103–112.

[31] A. Souag, R. Mazo, C. Salinesi, and I. Comyn-Wattiau, "Using the AMAN-DA method to generate security requirements: A case study in the maritime domain," *Requirements Eng.*, vol. 23, no. 4, pp. 557–580, 2018.

[32] J. Pauli and D. Xu, "Integrating functional and security requirements with use case decomposition," in *Proc. 11th IEEE Int. Conf. Eng. Complex Comput. Syst. (ICECCS)*, Aug. 2006, p. 10.

[33] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," in *Proc. 15th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Dec. 1999, pp. 55–64.

[34] A. Herrmann and B. Paech, "MOQARE: Misuse-oriented quality requirements engineering," *Requirements Eng.*, vol. 13, no. 1, pp. 73–86, 2008.

[35] R. Matulevicius, N. Mayer, and P. Heymans, "Alignment of misuse cases with security risk management," in *Proc. 3rd Int. Conf. Availability, Rel. Secur. (ARES)*, Mar. 2008, pp. 1397–1404.

[36] J. Whittle, D. Wijesekera, and M. Hartong, "Executable misuse cases for modeling security concerns," in *Proc. ACM/IEEE 30th Int. Conf. Softw. Eng.*, May 2008, pp. 121–130.

[37] N. Mayer, P. Heymans, and R. Matulevicius, "Design of a modelling language for information system security risk management," in *Proc. RCIS*, 2007, pp. 121–132.

[38] K. Hashizume, N. Yoshioka, and E. B. Fernandez, "Misuse patterns for cloud computing," in *Proc. 2nd Asian Conf. Pattern Lang. Programs*, 2011, Art. no. 12.

SIDRA SIDDIQUI received the B.S.(CS) degree from FAST-NUCES and the M.S.(SE) degree from Bahria University. She is currently a Lecturer with Air University, Islamabad, Pakistan. Her research interest includes secure software developing through reverse engineering and testing.

TAMIM AHMED KHAN received the B.E. degree (Hons.) in software engineering from the University of Sheffield, U.K., in 1995, the M.B.A. degree in finance and accounting from Presston University, Islamabad, Pakistan, in 1997, the M.S. degree in computer engineering from CASE, Texila University, Pakistan, in 2006, and the Ph.D. degree in software engineering from Leicester University, U.K., in 2012. He is currently serving as a Professor with the Department of Software Engineering, Bahria University, Islamabad. His research interests include service-oriented architectures, E-learning, and software quality assurance.