ARPN Journal of Science and Technology

http://www.ejournalofscience.org

Adapting Negative Requirement Model for Security Testing in Cloud Computing Environment

[1] G.Jagadeesh, [2] Sakineti Chaitanya Varma, [3] Dr.Anirban Basu

[1] Asst Professor(S.G), School of Information Techonolgy and Engineering,VIT University,Vellore 632014,INDIA.

[2] School of Information Techonolgy and Engineering,VIT University,Vellore 632014,INDIA.

[3] CEO,PQR software,Banglore,INDIA.

ABSTRACT

Providing security for a software system has always been a difficult task. The success of a software system always depends on how good the system is tested and the level of assurance for error prone system. All the positive requirements are considered for the testing which will not may enough for providing assurance. Hence negative requirements are also incorporated using risk analysis. Considering the implementation in Cloud computing environment where a huge number of test cases are also required due to its openness.

Keywords: *Error prone, Negative requirements, risk analysis, cloud computing*

1. INTRODUCTION

Cloud computing was initially explained by John McCarthy [1] in early 1960's predicting its future that it can be widely accepted by the public.

The well known service provider for the cloud services, Amazon has initiated its development by launching Elastic Compute Cloud[2] which led to the transformation from mainframe to the service side of client /server which led to development in such a way that companies can minimize time consumption for the completion of the project and thereby minimizing the budget allocated for the project development.

Functional and non-functional requirements have to be maintained correctly for launching any system.. So when considered about the cloud platform a more precise observation has to be maintained as it involves in security flaws such as unauthorized access ,hijacking and loss of sensitive data etc.. [3][4]. These risks can be eliminated by proper implementation of testing. But the correctness of functional requirements solely does not assure the security as we can consider a example of strcpy() function in C language which will lead to the overflow of the buffer. Though these risks can also be eradicated by proper use of mitigation but these mitigations also cannot guarantee the eradication of risks.So implementation of these mitigations also cannot give the confidence of eradication of risks and hence a proper methodology is required.

Implementation of security testing in cloud computing is done in three layers namely SERVICE, INFRASTRUCTURE and PLATFORM layer. Security testing has to be provided to both cloud providers providing Software-as-a-service(SaaS) and also to the cloud consumers which makes a use of Platform-(PaaS) and Infrastructure -(Iaas).

Fedler[5] proposed a approach which concentrates only on the positive security requirements but does not mention about false security requirements.This risk oriented security testing in the cloud computing can be approached by considering a model driven approach whose results vary depending on the implementation of the Thereby risk analysis would produce a risk model by performing Cloud Under Test(CUT)from which negative(false) requirements are identified. These negative(false) requirements help in identifying the false(misuse) cases by analyzing based on CUT. This can be analyzed as positive requirements can be identified by failure of false (misuse) cases. Conducting of this risk analysis is appropriate as so many number of test cases have to be verified for maintaining security of the system.. The results obtained from this risk analysis are ranked so that it will be helpful in starting the testing process there by increasing the efficiency of the system. Thus it can be justified that design flaws are responsible for most of the possible risks which can be know under probable circumstances. Thus a model based approach can be helpful in identifying risks and developing a risk free software. The most commonly used modeling language UML[6] is considered for explanation of the risks identified.

2. RESEARCH:

A new approach of research has been analyzed to identify the risks. The test related information is separated from the cloud system model in a specific test model which can be observed in the following diagram. This separation is found to be advantageous as the system model can be independently developed an also the relation between the system and test model can also be identified which will help in deriving detailed tests.

ARPN Journal of Science and Technology

http://www.ejournalofscience.org

The Cloud Under Test can be pictorially represented by interfaces as available services, classes as application data types and operations as service implementations.

In cloud computing environment applications SaaS, IaaS, PaaS are treated as services in which IaaS and PaaS can be accessed from the outside. Even the classes and operations of the interface services should also be analyzed as they may contain the design problems which are also the security risks.

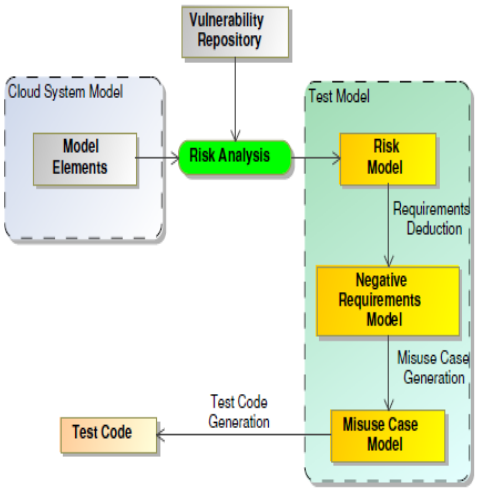

Fig1: The test model

The Test Model:

The test model has its own sub models like

1. Risk Model(RM)
2. Negative Requirements Model(NRM) and
3. Misuse Case Model

are generated automatically from this Model to model(M2M) approach which are different from the system model in which all transformations are done manually. This model based approach contains a vulnerability repository which has information regarding the past experiences which will be helpful in comparing the identified risks. Risk Model is considered to be a difficult task which is overcome by this vulnerability repository. These past experiences in the repository cannot be updated regularly; hence there is a chance of editing the Risk Model and adding the required risks for the model. The Risk Model explains the level of a particular risk and explanation of damage caused by a particular risk.

Identifying the false (negative) requirements from the Risk Model does not hugely involve in the process of analyzing the risks. So a M2M transformation is implemented by transforming to NRM from RM. The NRM contains the negative requirements which are identified from the possible risks.

These identified false requirements are in the form of text and explains unauthorized activities for the system and from there it is transformed to MCM. The use of NRM in the middle of RM and MCM is that the identified false requirements are analyzed at NRM rather than directly forwarding from RM to MCM . This will increase the possibility of analyzing more on the identified negative requirements.

Change–Driven Test Model Evolution:

In any model change management is considered as very important tasks as some wrong changes may corrupt the entire system. So by having the availability of editing the changes is always been a very useful task in any environment under consideration. In our system we intend to implement the changes to the identified design problems or changing requirements. So the required changes can be analyzed and implemented manually.

In our system these intended changes can be done in two ways, the first one is to implement the modifications that are identified in the cloud under test (CUT) and the second one is implementation of the test cases. Changes in the system would bring a change in the RM which will lead to a change in NRM there by modification in MCM.

The failure of these test cases would ultimately result in the positive requirements which are in turn useful for the system. This it-self not a completed task of change management but a state machine has to be incorporated each and every model as per the methodology.

The state machine is divided in to four states

1. New
2. Modified
3. Incorporate changes and
4. Clean

The first state of this model is considered to be the New state which is latently created state. The states modified and incorporate changes can be divided. The transition from either modified or incorporate changes state requires a manual modification of model where as transition from new state can be done by an external event to the incorporate changes . If the current state of the model is in incorporate changes then it can change to modified state by applying changes. After adopting all these changes the model changes ti state clean signifying that changes have been finished and cleared. As soon as the model changes from the state of incorporate to modified then it can be identified that the expected changes can be cleared. This entire process can be explained using the following diagram.

ARPN Journal of Science and Technology

http://www.ejournalofscience.org

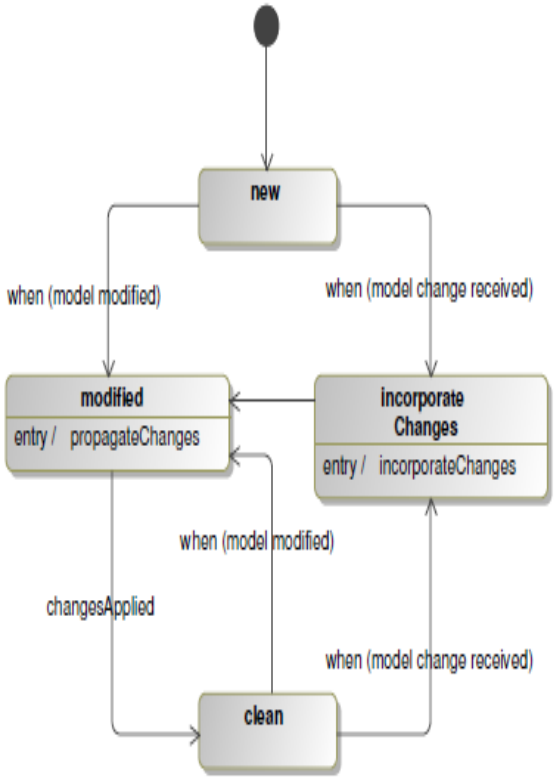

Fig2: The state machine

3. TEST CODE GENERATION

A research paper Telling Test Stories [5] explains that the UML can be converted to executable code which is very efficient in working

Evaluation of the Approach:

Analyzing the success at the early stages of development is considered to be a success for any project and this process of identification are considered to be as a milestones for the project

1. Model based analysis of risk proved that automatically obtained results are far better than the manually obtained results
2. An intermediate models such as Negative requirement model(NRM) and Misuse case model (MCM) are always beneficial for better analysis of risks and maintaining security.
3. If proper testing is assured then implementation of this method using a tool minimize the risks occur during the development which creates secured cloud platform.

4. CONCLUSION AND SUMMARY

A brief study of this paper will help in knowing the importance of identifying negative requirements and performing testing on them to maintain a secure system. With the help of UML an automated risk analysis can be performed there by identifying the misuse cases which will provide secure cloud environment. So this can be considered as better approach in providing security.

REFERENCES

[1] S. L. Garfinkel, Architects of the Information Society: Thirty- Five Years of the Laboratory for Computer Science at MIT, H. Abelson, Ed. The MIT Press, April 1999.

[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” SIGCOMM Comput. Commun. Rev., pp. 50–55, 2009.

[3] Cloud Security Alliance, Cloud Security Alliance (CSA)- Security Guidance for Critical Areas of Focus in Cloud Computing, V2.1, 2009. [Online]. Available: http: //www.cloudsecurityalliance.org/csaguide.pdf

[4] Cloud Security Alliance (CSA) - Top Threats to Cloud

[5] M. Felderer, B. Agreiter, and R. Breu, “Security Testing by Telling TestStories,” in Modellierung 2010. Gesellschaft fuer Informatik, 2010, pp. 195–202.

[6] OMG, OMG Unified Modeling Language (OMG UML),