A SURVEY REPORT ON SECURITY FOR TESTING PHASE OF SOFTWARE DEVELOPMENT PROCESS.

***G.ASHOK KUMAR, **PRASANTH YALLA**

**Department of CSE*

Krishnamurthy Institute of Technology and Engineering (Affiliated to JNTUH)

Ghatkesar, Hyd- Pin 501301 ashokkumar.wishes@gmail.com

*** Department of CSE*

K L University, (Deemed to be University)

Green Fields, Vaddeswaram, India- Pincode : 522 502.

prasanthyalla@kluniversity.in

ABSTRACT

Security testing is an integral part of risk management process and executives realize the benefits of an independent security test process. It is highly desirable to optimize the security test activities for a fast time-to-market while delivering a product that meets security expectation. Software security test process elaborates various testing activities and describes which activity is to be carried out when. Given the need and significance of phased approach of security testing, this article proposes a prescriptive framework elaborating security testing activities to be carried out while integrating it within the development life cycle.

Keywords— Security Testing, Software Security Testing.

INTRODUCTION

An increasing number of security incidents and growing awareness among business owners about invalidated applications due to security issues have moved security testing into the software tester‟s world. Security testing is all about finding out all the potential loopholes and weaknesses of the application, which might result into loss/theft of highly sensitive information or destruction of the application by an intruder or allow any intruder, enter into the system [1]. The conventional approaches of security testing fail to capture vulnerabilities and logical errors early. It also leads to the high cost for fixing issues identified at later stage. In case of conventional testing, developers get very less time for fixing the issues.

There are potential attacks at early phases of development life cycle. Design issues may be misuse of cryptography, broken or illogical access control and privileged block protection failure. Implementation issues may be buffer overflow, crosstie scripting, SQL injection and parameter manipulation. Development issues may be insecure configuration, brute force attacks and source code exposure. All of these different issues at various phases of development life cycle need to be

addressed and fixed there only.

One of the ways to do so is incorporating security testing in each and every phase of development life cycle. Therefore, there appears an urgent need of developing an integrated security testing life cycle which can be run in parallel with the development life cycle.

WHY SECURITY TESTING

Security testing is regarded as an important means to improve security of software. Software security testing validates the secure development and implementation of software product thus reducing the likelihood of security flaws being released and discovered by customers or malicious users. Ultimate objectives of security testing are to validate the robustness and to prevent security vulnerabilities from ever entering the software [1]. Software security testing has become an essential means of ensuring software security and trustiness. Software security testing verifies that the software‟s dependable operation continues even under hostile conditions, its anomaly, error and exception handling can recognize and safely handle all anticipated security-relevant exceptions and failures [2]. Security testing not only include conformance of resistance of the systems the organization uses, it also ensures that people in organization understand and obey security policies.

Security testing is the process of determining whether an information system protects data and maintains functionality as intended. It is the single most common approach for gaining assurance that a system operates within the constraints of given set of policies and mechanisms. Risk based security testing is performed motivated by understanding and simulating the attacker‟s approach. By identifying risks in the system and creating test driven by those risks, a software security tester can properly focus on areas of code in which an attack is likely to successes [3]. There are probably a thousand reasons to take security testing seriously. Some of the pertinent reasons are cited as follows:

- Security testing helps to find out loopholes that can cause loss of important information or allow intruders into system;
- Security testing helps to verify software trustworthiness in terms of inconsistently safe behaviors and state changes, and it’s lack of exploitable flaws and weaknesses;
- Security testing helps to improve the system, and finally helps it to work for longer time;
- An integration of security testing within the development life cycle can help in eliminating flaws early in design; as a result reducing cost and effort in blocking the potential

CURRENT SCENARIO

Software security testing is an important method of ensuring software security. Therefore, security testing research has become very important and practical, which is still in its infancy [4]. There are works cited in literatures on software security testing. Researchers and practitioners are advocating for traditional software security testing which plays more attention to security functions based on software requirement. Unfortunately, traditional security testing cannot ensure software security effectively [5]. Most of the testers are unfamiliar with application security testing, and generally executing penetration test at the end of the development life cycle. Vulnerabilities found latter requires a lot of rework for mitigation.

Today, many organizations are implementing security penetration testing at the end of the software development life cycle to cover all software security. Penetration testing is an authorized attempt to breach the security of a system using intruders and /or worth access technique. Implementation of penetration test at the end runs a significant risk of not only finding severe security bugs late, but also delivering a system that has an inadequate security design [6]. With growing concern about software security, research on security testing has made some progress in terms of developing generic tools such as monitoring tools and interfacing tools [7]. A critical look at efforts made in the direction of security testing reveals that there are some approaches proposed by researchers and practitioners. It includes Formal security testing, Model-based security testing, Fault-injection based security testing, Fuzzy testing, Vulnerability scanning testing, Property-based testing, White-box based security testing, and Risk-based security testing [8]. All of the approaches are developed and defended/validated by the research and practices. Every approach has its own advantage. But unfortunately, the contributions made in the direction lacks with the realistic problems and are not very efficiently implemented in real world. Some of the pertinent issues involved with most of the approaches are as follows:

- They do not help to test for the issues that commonly appear in software and are not easily fixed by using a more programmer-friendly platform;
- Most of the techniques fail to continue under the adverse conditions often encountered in security testing;
- Generally, the approaches available are not very robust and do not work well with incomplete information about the target of testing [4].
- Most of the technique do not provide guidelines to their users throughout the testing task;

- The result obtained on implementing approaches are not very useful under real-world conditions;
- Most of the approach do not provide useful abstractions;
- The current state-of-art fails to address the issue raised above and solutions to all the above research challenges. There is an urgent need to narrow the gap and create common ground for the integration of security issues into the development process.

REVIEW OF R & D IN PROPOSED AREA

CERT has been emphasizing the need for and the significance of addressing security [9]. A recent survey has revealed that the least secure software generally carries six times higher business risk than the most secure software product. The survey highlighted the fact that security quality of software product can vary drastically depending on its design and implementation [10]. Since decades, vulnerable software is invaded and modified making it cause damage to the healthy software. These damages are replicated across networks to cause dangers to the entire society. Increased network connectivity, complexity of code, attack sophistication and easier access to attack tools have led to an overwhelming and ever-growing number of security breaches-something that can hurt or severely injure most software development organizations. At the root of almost all acts of security violation, there is one or more software vulnerability that can be exploited by an attacker to perform actions. It appears feasible that a majority of software vulnerabilities to be traced back to a relatively small number of causes, if addressed security testing at development process. Finding and fixing the security related problems early will reduce the overall risk and cost of the software product. Security experts have clearly recommended the need for a concrete approach to address security testing. Framework and metrics are tools that may be used for evaluation of software security testing. It is evident from literature survey that considerable efforts are being made for security testing. The following section briefly describes some of the pertinent contributions made by the researchers and practitioners. Most of the work carried out in the area strengthens the need and importance of having security testing process in development life cycle.

Software Assurance Pocket Guide Report entitled Software Security Testing was presented in **May 2012** by SwA Community Resources and Information Clearinghouse (Web URL" http://buildsecurityin.us-cert.gov/swa). This report advocates that goal of software security testing is to find vulnerabilities, keep them from the build product and confirm that security of the software product is at an acceptance level. The software security testing process is key channel to validate the secure development & implementation of product from malicious users. The most effective security testing techniques like security requirement testing, risk analysis, automated static analysis, fuzz testing are discussed with their strength and weaknesses. The pocket guide is helpful to discuss the

detailed discussion of security testing and related issues and also when to apply them during software development life cycle [11].

Recently, In **2012,** Aaron Mar back et el proposed a threat model based security testing approach that automatically generates security test sequences from threat trees. The following approach is being executed through two steps. The first step involves automated test sequence generation from threat trees and second step is to transforms them into executable tests by considering valid and invalid inputs. An empirical study has been conducted using real world web applications project like demonstrate the feasibility of applied approaches. The result of the study showed that the testing approach is effective in exposing vulnerabilities that are presented in real world web applications. That data or information which is used in project that generates executable

Security test cases from threat trees is feasibly evaluated and tested to pursue the actual functionality [7].

In **2102** the most recent and relevant work identified is a Model Based Security Testing (MBST) proposed by Ina Scieferdecker, Juergen Grossmann, Martin Scheider. According to author the paper refers to active, dynamic testing, where behavior of system under test (SUT) is checked by applying intrusive tests that simulate the system and observe and evaluate the system reactions. Model-based security testing (MBST) is a relatively new field and especially dedicated to the systematic and efficient specification and documentation of security test objectives, security test cases and test suites, as well as to their automated or semi-automated generation. MBST includes e.g. security functional testing, model-based fuzzing, risk- and threat-oriented testing, and the usage of security test patterns. A project "European ITEA2-project DIAMONDS" is helpful to develop tools and methods based on MBST technique [12].

In **2011**, SaaS Security Testing: Guidelines and Evaluation Framework by Jayanti Vemulapati is presented by Infosys-View Point which introduces software as a service for business and economic model for software delivery. The work is very enlightened about SaaS Security challenges and result obtained through applied security testing. The paper also discusses the security standards as laid down by CSA and DMTF that should be followed by SaaS venders. On the basis of these guidelines and standards, any SaaS offering can be evaluated its security standard [13].]. In 2010, Huang Song et el present a modeling technique with threat tree. A threat tree traversal algorithm (Tri-T algorithm) based on depth first search is designed and used in an example to generate a test sequence based on typical security defect with respect to defect hypothesis [14].

In **2011** the work proposed by C. Warren Axelrod examines a functional security testing approach for secure and tough operation of application and relates to other forms of testing. Discussion regarding different categories of testing like functional requirement testing is carried out. In large

part, this is due to software assurance staff not testing fully for

"negative functionality," that is, ensuring that applications do not do what they are not supposed to. There are many reasons for this, including the relative enormity of the task, the pressure to implement quickly, and the lack of qualified testers. By examine the issues of FST, suggestions are made for improvement by ensuring that applications are prevented from functionally allowing damaging events [15].

In **2010** Mazen, by using extended finite automata, he represents a method to integrate interoperability security rules in functional model. A formal approach has been taken for interoperability testing of security rules. He took a detailed study to specify the transitions in which security rules have to be integrated. . The resulted functional model is used to perform testing of interoperability security rules. It is also Important to mention that this study could be used to integrate any security policies specified with the same modality as O2O. A case study has been produces along with result. The work finally produces interoperability test cases to validate the implemented security policy [16]. In **2009** Hao Chen and Jean-Pierre Corriveau proposed an approach for security testing based on the international security standards and policies which may help developers and testers to acquire a good understanding of security testing and develop a general compliance testing system focusing on client side security. The aim of study is suggesting to testers how to design security testing and identify potential vulnerabilities in current online banking system [17]. In **2008** Sven Turpe outlines a number of research problems to be addressed in security testing. Industry practices on security testing are limited. Working on these vulnerabilities and their causes may helpful to develop a better tools & techniques for real world applications. It discusses about security, vulnerability and used better tools and techniques [4].

SIGNIFICANT NEED

Security testing aims at finding security vulnerabilities prior to making them available to end users. One of the fundamental objectives of security testing is to identify whether the security features of the software implementation are consistent with design [8]. There are several objectives of implementing security testing during development; some of them are listed as follows [2]:

- To verify whether the behavior of software is predictable and secure;
- To check whether software exposes no vulnerabilities or weakness;
- To pinpoint if any of the system‟s intended functions are unintended or unauthorized;
- To verify software‟s survivability;

- To assess the software‟s trustworthiness;

INTEGRATED APPROACH

While addressing security testing, one of the major issues to consider is to discuss how the system under design deals with possible attacks at early design stage [18]. An effective security testing should test the entire phases, rather than just implementation. A test process is needed to ensure that the designed system can protect asset from attacks with the help of mitigation. Security testing with a structured approach throughout the entire development life cycle gives a good understanding of the software quality and protects from known security risks [4]. The tester should apply his expertise early in security review at the requirement elicitation phase. The required changes can be made even before the coders start their development. If security testing is not considered during development, the application can contain dangerous vulnerabilities and sufficient vulnerability propagation will occur throughout the development phases making enormous risks to the organization. In order to gain insight into the quality of software, an integrated approach of security testing is necessary to spot the vulnerabilities in each phase of development life cycle and toto avoid its propagation. An integration of security testing within the development process will reduce the cost of damages and risks associated. Security testing strategy for software product should be developed for each phase.

It has become mandatory and even challenging to investigate what is involved in security testing activities that make it so expensive and important. It is highly desirable to lookup what needs to be done at each activity, how to optimize the time and finally elucidate the areas where it could be used to save time. If, during testing process, optimal activities are identified and implemented, significant time could be saved with the adoption of innovating testing techniques.

In addition to this, an effective and prescriptive process of security testing specifying very clear prioritized activities may be advantageous in different perspectives as follows:

- An structured security testing process may lead to a dependable tested software;
- Implementing a correct security test process, scheduling in a systematic way and applying them smartly may result a significant reduction in time and effort with increased quality;
- Development with the prescriptive security test activities may lead to quality product with shortest span of time;
- Through efficient security testing activities, the power of automation could be exploited;
- An appropriate and accurate security test activity implemented during development may make

the software more profitable;

The advantages discussed strengthen the need to migrate from an immature, adhoc way of working to have a full-fledged security testing process. An effective security test process requires a wide verity of skills and activities, including the identification, collection and documentation and supporting materials. Within this context, test activities should be prioritized with the ultimate objective of delivering maximum benefits to the end-users in terms of security [19]. Therefore, there appears an urgent need of having a commonly accepted process of software security testing that can be integrated with each phase of software development life cycle to develop security software. A prescriptive framework is proposed in figure 1 in order to cater the need. The proposed process comprises of the seven phases including *Security Test Strategy and Security Test Plan, Design Security Test Cases, Execute Security Test Cases, Capture Security Test Results, Capture Security Test Metrics, Qualitative Assessment and Security Test Closure Report*.

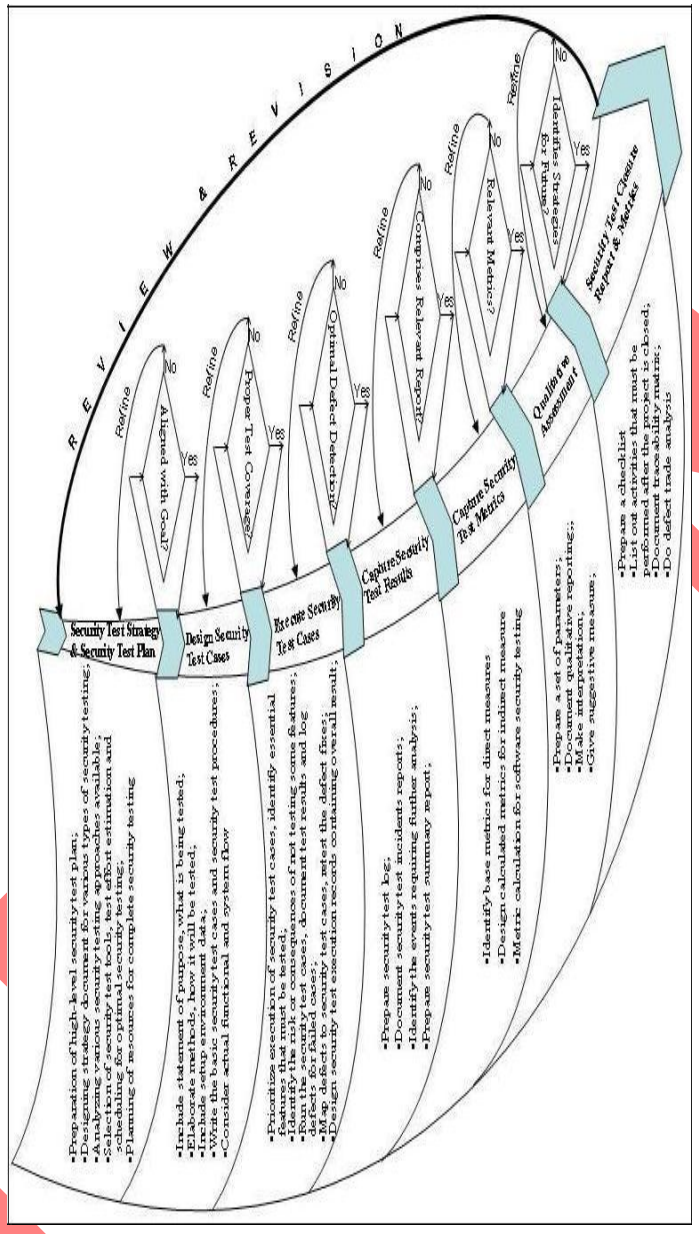

Figure 1: A Prescriptive Framework for Security Testing

CONCLUSION

It has become essential to integrate phased security testing process within the development life cycle with the intent of finding errors at each phase well in advance in order to reduce developmental cost, delivery time and rework efforts. A prescriptive framework consisting of seven phases is proposed with the objective of identifying defects early and preventing defect migration. The

proposed work may help testers to better understand and execute security test in an efficient and effective manner in order to deliver secure software.

REFERENCES

[1] G Kavitha Jayaraman, Incorporating Security in Software Testing Life Cycle, Cognizant Technology Solutions, 2009.

[2] Risk-Based Software Security Testing, Software Assurance Pocket Guide Series: Development, Volume III, Version 0.5, September 1, 2009.

[3] Bruce Potter & Gary McGraw, „Software Security Testing, IEEE Security & Privacy, 2004, pp. 32-36.

[4] Sven Turpe, „Security Testing: Turning Practice into Theory, IEEE International Conference on Software Testing, Verification and Validation Workshop (ICSTW08), IEEE Computer Society, 2008.

[5] Huang Song, Wang Liang, Zheng Changyou & YU Hong, „A Software

Security Testing Method Based on Typical Defects, 2010 International Conference on Computer Application and System Modelling (ICCASM 2010), IEEE Computer Society, 2010, pp. V5-150-153.

[6] SOA Test Methodology, Torry Harris Business Sollution, www.thbs.com/soa.

[7] Aaron Marback, Hyunsook Do, Ke He , Samuel Kondamarri and Dianxiang Xu, A Threat Model based Approach to Security Testing, Software Pract. Exper. (2012), Published Online in Wiley Online Library , DOI: 10.1002/spe.2111.

[8] Gu Tian-yang, Shi Yin-sheng & Fang You-uan, „Research on Software Security Testing, World Academy of Science, Engineering and Technology 69 2010, pp. 647-651.

[9] Available at :www.cert.org

[10] Dr. Sachar Paithis, The Future of Software Security, www.testingexperience.com, Page:61-62

[11] Software Security Testing, Software Assurance Pocket Guide Series: Development, Volume

III, Version 1.0, May 21, 2012.

[12] Ina Scieferdecker, Juergen Grossmann, Martin Scheider, Model based Security Testing, Workshop on Model Based Testing, MBT-2012, pp:1-12, doi:10.4204/EPTCS.80.1

[13] Jayanti Vemulapati, Neha Malhotra, Nitin Dangwal, Security Testing: Guidelines and Evaluation Framework, 11th Annual International Software Testing Conference, 2011, Infosys View-Point, August 2011, pp:1-7

[14] Huang Song, Wang Liang, Zheng Changyou, YU Hong, A Software Security Testing Method Based on Typical Defects, ICCASM 2010, 978-1-4244-7237-6/10, IEEE 2010,pp:V5-150-153

[15] C. Warren Axelrod, Ph.D.,The Need for Functional Security Testing, Delta Risk, Cross-Talk, March-April 2011, Rugged Software, pp: 17-21

[16] Mazen El Marrabani, Iksoon Hwang, Ana Cavalli,A formal Approach for Interoperability Testing of Security Rules, IEEE 2010, pp: 277-284,doi:10.1109/SITIS.2010.53

[17] Hao Chen and Jean-Pierre Corriveau, Security Testing and Compliance for Online Banking in Real-World, Proceeding of the International Multi Conference of Engineers and Computer Scientists, 2009, Vol. I, IMECS, March 18-20, Hong Kong, ISBN: 979-988-17012-2-0

[18] Ke He, Zhiyong Feng, Xiaohong Li, „An Attack Scenario Based

Approach for Software Security Testing at Design Stage, 2008 International Symposium on Computer Science and Computational Technology, IEEE Computer Society, 2008, pp. 782-787.

[19] Ljubomir Lazic, Nikos Mastorakis, Cost Effective Software Test Metrics, WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Volume 7, June 2008, pp. 599-619.

AUTHOR'S INFORMATION

Ashok Kumar Gottipalla is pursing PhD in Computer science and Engineering from Koneru LakshmaiahUniversity , Green Fields, Vaddeswaram, Guntur District, A.P., INDIA. He has been completed his MTech degree from Acharya Nagarjuna University, Guntur. Mr. Ashok kumar is young, energetic research fellow. He has more than 10 year of teaching & research experience. He is currently working in the area of Software Security and Security Testing. He has also published & presented

papers in refereed journals and conferences. He is a member of Computer Society of India.

Dr. Prasanth yalla has earned his doctoral degree from Nagarjuna University, Guntur, India and he is currently working as an Associate Professor in the Department of Computer science and Engineering, Koneru LakshmaiahUniversity Green Fields, Vaddeswaram, Guntur District, A.P., INDIA. His area of interest is Software Security, Web services,SOA, Software Quality and Software Engineering. He has published a number of National and International books, research papers, reviews and chapters on software quality and software testing.