Volume 5, No. 6, June 2014
Journal of Global Research in Computer Science
RESEARCH PAPER
Available Online at www.jgrcs.info
ISSN-2229-371X

MOTIVATION FOR SECURITY TESTING

Manju Khari[1], Chetna Bajaj[*2]

[1]Department of Computer Science & Engineering,
Ambedkar Institute of Advanced Communication Technologies and Research,
Delhi, India
manjukhari@yahoo.co.in[1]
*[2]Department of Computer Science & Engineering,
Ambedkar Institute of Advanced Communication Technologies and Research,
Delhi, India
chetnabajaj7@gmail.com[2]

Abstract: Security testing is used to build a secure system but it has been ignored for a long time. It is of immaculate importance these days. In today's world, privacy and security have been assigned foremost importance, therefore it is highly recommended to look forward for data and operations' security in software applications, which demands urgent attention but it is rather ignored. Therefore, our objective is to introduce developers with an esteemed importance of system's security, which can be induced by implementing security testing methodology in SDLC process to produce a secure software system. So, Security Testing has been defined from developer's point of view. It resembles methods that need to be incurred in SDLC process to incorporate security feature in software. Software Security Unified Knowledge Architecture not only describes Security testing's values and objectives but also provides some developer's guidelines to produce a secure software system.

Keyword: Security Testing, SDLC, Software Testing, Security Knowledge.

INTRODUCTION

Software Testing is an important process in SDLC. It provides assurance to both developers and users as well. Developers get assurance from system's extensive testing and can present it to users for acceptance whereas users consider a good amount of testing as an important parameter for accepting any software. But testing is not performed for this assurance only. It has major significance in day-to-day operation of software system. Software need to be tested thoroughly to enhance its capability to handle abnormal conditions. Software testing is performed by concentrating on points or situations where software may behave abnormally and can result in failure later on. System's failure may cause million dollars business loss all over the world which is not generally acceptable to any organization at any cost. As a result, testing process has been accrued a lot of importance and is given almost 40% time of the total time required in SDLC process. Testing reveals system's shortcomings and failure, which need to be fixed by developers properly without affecting other system's components. Although efforts involved in process make testing very costly, it is worth the benefits one gets.

The basic principle of software and security testing is same, to ensure system's security. Security testing is performed to incorporate security features i.e. authentication, confidentiality, integrity, availability etc. to the system. Security testing resembles a system's state where it can secure itself from unwanted actions and does not allow other entities/intruders to vanish system's integrity. The unwanted action may comprise system's unauthorized access to suspiciously alter file(s). So, Security testing is an act of making system defendable from attacks.

Software Security

Security testing helps in securing system/application. It is an advanced version of software testing. Software Testing mainly focuses on testing of software's functionality. Functions implemented in software are analyzed to ensure whether software system produces estimated response. Software testing resembles system's functional aspect. Security testing is more advanced than software testing as it considers security, a non-functional system's property. It depicts system's ability to make it secure. The system is made secure by implementing functions which prevent an unauthorized user to access system's valuable and confidential information (Gu Tian-yang et al., 2010) [1]. The developer needs to code security enforcing functions to protect system by preventing it from being exploited. System will be secured if it functions well, even in presence of vulnerable or malicious code or activity that can exploit system, and does not have any adverse effect on it.

Security is a vital task or property. Providing security to system is very complex in comparison to simple software testing process which involves black box and white box testing. For securing system, we need to check system's two important things: First, validity of implemented security measures that provide functionality and security to system. Security measures also include features like cryptography, strong authentication, and access control measures. And second, system's behavior when it gets attacked by attackers, resulting in destruction by accessing secured and confidential information.

Attackers can attack system with their most powerful and exotic skill set to create room for himself in it (Hoglund and McGraw, 2004) [2]. Developer and tester need to understand attacker's mindset so that they can restrict attacker's exploitable activities for hacking system.

Software Security Testing

Security testing is very much important for software application as it takes care of confidential data. It ensures that confidential data does not get overlooked by unauthorized entity. It works beyond functional (i.e. black box) and implementation (i.e. white box) testing. Security tester may use many techniques to locate system's vulnerabilities. Testing system's security checks the loopholes or vulnerabilities in system which may cause failure of security functions of system ultimately leading to great losses to organization. Therefore, security testing is employed to ensure that developed software is free from flaws and hence, the system is safe from unauthorized individual, be it an employee or an outsider. Security testing identifies threats and measures its impact on system. The impact is analyzed by developers or testers by playing the role of an attacker. They put their efforts to break the system or to get into it to find bugs. So, security testing is very essential in IT sector for data protection.

Security testing is related to risk based testing approach which analyze risk in each phase of SDLC. Proper measures are taken to eliminate risk to make system secure. So, Testers must incorporate a risk-based testing approach by keeping system's architectural reality and attacker's mindset into consideration for applying software security adequately (Erdogan and Stolen, 2012) [3]. In this approach, risk affected areas are identified for testing. Developers/Tester need to develop test cases to reveal problems if any. The approach provides high level of software security as compared to black-box testing.

Security testing deals with system's security. It observes system's behavior in presence of malicious attack. It tries to construct and execute test cases to make software work properly in attack phase as well.

In Section 2 of paper, various types of common techniques of testing would be discussed. In Section 3, various developer facing issues are described along with answer to 'who should do security testing and how' etc. would be highlighted. Software Security Unified Knowledge Architecture is elaborated in section 4 which focuses on three important knowledge catalogs. In Section 5, Security knowledge to secure SDLC is defined which includes various knowledge catalogues implemented along with SDLC to model it in a SSDLC. Section 6 discusses about integration of security process with SDLC phases, benefitted for a secure software system development. The same is also summarized in table. At the end, we conclude the paper with its future scope discussion in continuation of reference list.

RELATED WORK

In Software engineering practices, software systems are developed by a specific SDLC model among all. Each model consists of various common phases. Testing is one of them and is included at end to find software's operational response. Testing describes implemented software functionalities in practical way. Testing verifies the developed system with expected and unexpected inputs and observes its output thoroughly. Observations decide system's correctness. Following approaches are very common to test software:

1. Black Box Testing approach
2. White Box Testing approach
3. Gray Box Testing approach
4. Risk Based Testing approach

Black box Testing Approach

It is a very simple and efficient technique to test system as it just observes behavior of system under test (Aggarwal and Singh, 2005) [4]. It analyzes program's behavior with various input combinations to look for any abnormal behavior or wrong output. It does not perform code inspection checking. It executes program with valid and invalid inputs. The system is looked for its response and in case, an abnormal behavior is observed, it must be corrected (Khan and Khan, 2012) [5]. It is also called Functional testing. To check security, tester injects malicious input, resultant system behaves abnormally and developers fix it.

White box Testing Approach

White box testing approach is important software testing which actually looks into code to find system's flaws. It is also known as structural testing (Aggarwal and Singh, 2005) (Khan and Khan, 2012) [4] [5]. It requires tester to understand design and implementation knowledge of source code. It is very efficient in locating programming errors. Some testers use static analyzers and pattern matching to test programming errors in code, but it can provide false positive results (which show there is vulnerability but actually there is none).

Grey-box testing Approach

Grey-box testing (or gray-box testing) is defined as combination of black box and white box testing, and increases testing coverage of software testing (Khan and Khan, 2012) (Irena, 2008) [5][6]. It allows testers to test software with basic information about it. The basic information required for Grey Box testing includes knowledge of internal data structures and algorithms, used for designing test cases. The test cases are executed at exposed interfaces. Grey box testing is best suited for testing integration of two modules. The interface is checked or tested for modules' connectivity and data flow mechanism between them. It requires that tester must have knowledge about application's operation and functionality.

Risk based Testing Approach

Risk based testing technique refers risk associated with software system under test. In (Ould, 1999) [7], author defines risk as "any threat to the achievement of one or more of the cardinal aims of the project". Another definition of risk state that "A risk is a problem that has yet to occur, and a problem is a risk that has already materialized." (DeMarco and Lister 2003) [8]. The risk associated with software may cause great loss to organization, that's why risk based testing is considered of great importance. Risk based security testing also considers attacker's intentions and his abilities to perform attack. Developers identify risk associated with an attack and try to minimize it (Potter and McGraw, 2004) (Khan and Khan, 2013) [9] [10]. So, it provides very good methodology to improve software's quality. It also helps management personnel to make necessary decisions regarding software release in market.

SOFTWARE SECURITY TESTING ISSUES FOR DEVELOPERS

Testers need to understand how to provide a good and qualitative amount of testing time to system. It is required to thoroughly test the system to find out maximum errors. These errors must be removed to get software accepted by customer. Testers also need to learn to provide suitable time and efforts to locate non functionality security risk.

Developers perform exhaustive black box and white box testing to test software system. They look for maximum numbers of errors and flaws and upon finding, get them fixed to make software work well. But the process does not include security, an important non-functional feature. As a result, security testing process came into existence. Initially, when security testing was started as a new technology, people were uncomfortable with it. It was not easy for them to test system's security completely. Developers face many issues in security testing process during SDLC of a secure software system. Some of them are mentioned below:

1. Who should do the Security testing?
2. How Security testing can be done?

Who should do the Security testing?

Security testing is done by removing system's vulnerabilities. Vulnerability may occur from a misunderstood design flaw or can also occur from a fault in source code during implementation phase. Vulnerability injected during design phase become very complex. Hence, it becomes equally difficult to remove them. It can be removed by the expertise of an experience person having knowledge on similar kind of projects. Further, It is assumed that the person should be accomplished with knowledge or context's information, but there is shortage of good, experienced and knowledgeable testers (Thompson, 2003) [11].

Software industry requires a person who can use his expertise in every phase of SDLC to develop secure software. Now-a-days, organizations require a large number of security testers, to test application's security. Besides, we also need experts who can share their expertise (of developing secure applications) with other developers too.

Testers need to develop security test cases which can easily exploit and expose security related problems in system. They should observe it very carefully to identify software problems. But it is very difficult to model or design such test cases that can expose security related problems. Other problems constitute the deficiency of test cases. Sometimes, test cases developed are not sufficient to exploit software to identify actual problem. It has been said earlier also that it is very important to have experience for efficient security testing (Potter and McGraw, 2004) [12].

How Security testing can be done?

Security testing includes Black box, white box, Grey box and risk based testing approaches. White box testing includes testing of source code to find programming errors or bugs. Risk based security testing is implemented in SDLC to make a secure software product which includes following steps incorporated at each phase:

1. Creation of security abuse/misuse.
2. Listing of normative security requirements
3. Conducting architectural risk analysis
4. Constructing risk-based security test plans
5. Wielding the required static analysis tools
6. Conducting penetration testing in the final environment
7. Clearing the system from problems occurred due to security breaches.

Most important procedure in testing includes system's risk analysis, risk based security test planning and security testing which can formulate problems to risks faced by organization (Ould, 1999) [7]. Afterwards, these risks are ranked and prioritized by business authorities.

Software security can be measured by two important activities i.e. first, by testing functionality of implemented security mechanisms and second, by implementing risk based security testing to understand attack's simulation.

SOFTWARE SECURITY UNIFIED KNOWLEDGE ARCHITECTURE

Knowledge is defined as "information in context" i.e. the information can be used to perform some task. Security testing is not only adopting and implementing security features, it also assures about development of secure software by implementing various processes. So, Knowledge in context of security includes various processes and procedures used to develop secure software system. Hence, it is required to organize software security knowledge. Here, Software Security Unified Knowledge Architecture comes into picture with great importance given to security knowledge. It can provide best software security practices for developing secure software system. Software security knowledge can be applied in various phases of SDLC by using knowledge intensive practices. It also guides developers during designing and coding of software.

Software Security Unified Knowledge Architecture (Barnum, and McGraw, 2005) [13] is shown in figure 1. The architecture defines a structure which provides relation between different knowledge catalogues. These important seven knowledge catalogues are grouped in following three knowledge categories (Viega and McGraw, 2001) [14]:

1. Prescriptive Knowledge
2. Diagnostic Knowledge
3. Historical Knowledge

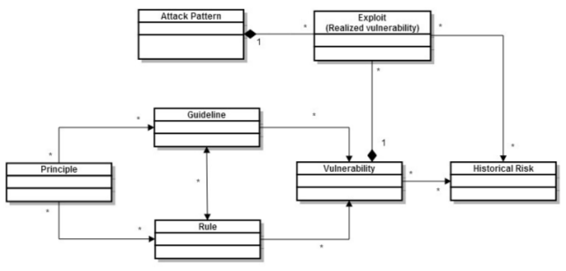

Figure 1. Knowledge objects of Software Security and their basic inter-relating architecture.

Prescriptive Knowledge

Prescriptive knowledge consist three knowledge catalogs: Principles, Guidelines, and Rules. It depicts the abstraction of high-level architectural principles at philosophical level to very specific and tactical code-level rules. The category suggests "what to do and what to avoid" during development of a secure software system.

Diagnostic Knowledge

Diagnostic knowledge consist three knowledge catalogs: vulnerabilities, exploits and Attack patterns. It helps developers to identify and tackle issues occurred from security attacks. Security analysts use it as a resource or component to be implemented in software development. Table 1 describes knowledge catalogs of diagnostic knowledge.

Historical Knowledge

Historical knowledge consist a knowledge catalogs called Historical Risks. It highlights system's issue and tries to analyze its impact on system from business perspective. The knowledge also provides lessons to minimize similar mistakes in future.

Table I. Knowledge Catalogs of Diagnostic Knowledge

Knowledge Catalogs	*Description*
Vulnerabilities Knowledge	It provides vulnerabilities experienced in systems.
Exploit	It depicts a scenario which tells sequence of operations carried out to compromise the system.
Attack Pattern	It includes a sequence of exploits which can attack a whole computer network.

SECURITY KNOWLEDGE FOR SECURE SDLC

Software Security requires expertise and experience of application programmer to maintain application's strength by code or control measures. Be it a security issue, requirement related constraints or any database constraints, Industry requires skilled practitioner who can input their exclusive skills to provide extensive facilities (i.e. not easy to implement) to applications. But due to lack of required knowledge, industry has very few skilled security practitioners. Moreover, Industry requires those skilled practitioners who can also share their expertise with others for better development of software (McGraw, 2006) [15]. Developers need to incorporate their software knowledge to provide excellent product. We can also use expertise of different software security practitioners to make use of it throughout the process. It can be named as security catalogs which define security constructs. Moreover, it can also be implemented in SDLC phases to define security perspective. Each knowledge catalog has its own phase, activities and outputs which are used successively in SDLC process and provide excellent results. Figure 2 shows SDLC phases with inclusion of knowledge catalogs. Table 2 describes different knowledge catalogs which are mentioned below:

1. Principles
2. Guidelines
3. Rules
4. Attack Patterns
5. Historical Risks
6. Vulnerability
7. Exploits

INTEGRATION OF SECURITY PROCESS WITH SDLC

We are well known with SDLC process used to develop software. Figure 3 shows implementation of security constructs in SDLC to develop secure software (Online Documentation, 2013) [16]. It shows how developers can develop a secure software product by SDLC process. Each phase has its own significance with some extra duties and responsibilities to be carried out to make a system secure. It is always agreed that if we detect an error or discrepancy in requirement phase then the cost involved to rectify it and to implement any change will be high during later stages of SDLC. That is, if we detect error in requirement phase and do not give adequate attention and postpone security testing after implementation or deployment, then that small error will become a security bug during later stages of SDLC. Consequently, the cost to solve problem i.e. corrective cost will get increased. So, it is equally necessary to involve security testing process in earlier phases of SDLC as well (McGraw, 1998) [17]. Figure 3 represents SDLC process incorporating implementation of security processes corresponding to each SDLC phase to produce a secure software system.

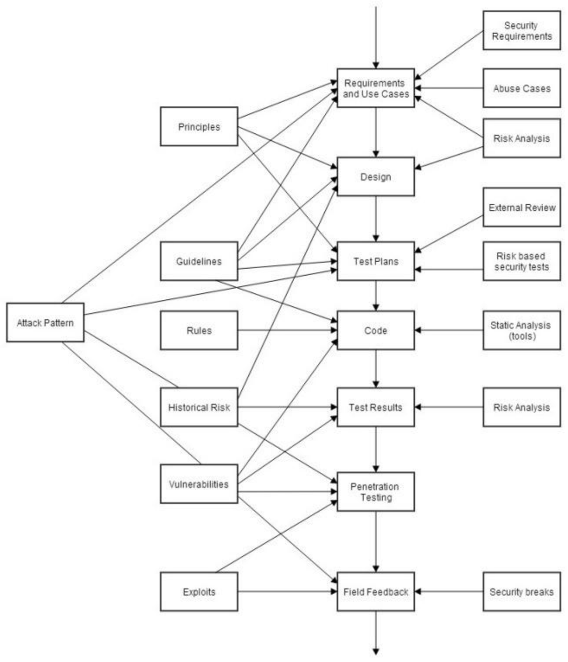

Figure 2. SDLC including various security testing practices to make a system secure

In Secured Software development process, security incorporating activities are included in each phase of SDLC process. Various phases are listed below and further explained in detail:

1. Requirement Phase
2. Design
3. Coding and Unit Testing
4. Integration and System Testing
5. Implementation
6. Support

Requirement Phase

In requirement phase of SDLC, developer gathers functional requirements required to develop software but forgets to collect security requirements. It's a developer mistake of not considering security requirements. However, it is important to collect security requirements along with function requirements. Developer can gather security requirements from users or from security guides. Security requirements must be documented and analyzed along with functional requirements. To understand the importance of security features, system's state must also be analyzed with or without security implementation. Feasibility study for the same must be performed before its implementation.

Design Phase

Developers must analyze each security requirement with respect to its design and implementation. They must understand system's design to propose efficient and effective security strategies, plans, designs and procedures. Special security practitioners can be hired who can focus on system's risk assessment. Design phase include creation of test plans that depicts how and when security testing will be performed to test system's security.

Table II. Knowledge Catalogs of Diagnostic Knowledge

Knowledge Catalogs	*Description*	*Sample High Level Schema*	*Artifact*
Principles	Developer involved in secure system development has learned many crucial points from their past experienced assignments and they do not want themselves to be in problem. So, they make best practices as principle to avoid hassles for smooth development of secure system. Principles play an important role in development and also find its place at philosophical level. It is used to find problem in architectural flaws or software. Principles are also used for practicing good security engineering.	1. Title 2. Related guidelines 3. Definition a. Description b. Examples c. References 4. Related rules	1. Security Requirements 2. Software System Architecture 3. Software Design
Guidelines	Guidelines refer to suggestions provided for secure system development. It includes directions to follow during development process. It specifies what needs to be done and what should not be included in process, along with its meaning to define semantic level of development process. Guidelines are specific and related to technical context i.e. it might be related to language like J2EE, .Net; or related to operating system like Linux Kernel modules. Guidelines help in finding architectural flaws and bugs during implementation phase.	1. Context Description a. Platform b. Operating system c. Language 2. Type 3. Title 4. Objective 5. Description 6. Development scenario 7. Related API 8. Related principles 9. Reference 10. Related rules 11. Security requirements 12. Software design	1. Security Requirements 2. Software Design 3. Code
Rules	Rules act as recommendations and guidance during SDLC process. It defines things to be followed or avoided at syntactical level of development process. These rules must be verified with development process by lexical scanner or by constructive software parsing tools used for source code and binary format. Rules are programming language specific. They are used to detect programming errors during implementation.	1. Context Description a. Platform b. Operating system c. Language 2. ID 3. Title 4. Attack category 5. Location 6. Description 7. Method of attack 8. Signature 9. Solution 10. Example 11. Reference 12. Related principles 13. Related guidelines	1. Code
Attack Patterns	Software gets exploited often. Keeping software exploit techniques in mind, attack patterns are designed. Attack patterns analyze the method that has challenged system's security by attacking and exploiting software system. Attack patterns help to indentify risk associated with each system's exploit. It plays an important role in designing misuse and abuse cases. It also facilitates in constructing test cases for specific security problem.	1. Context Description a. Platform b. Operating system c. Language 2. Title 3. Attack category 4. Description 5. Example 6. Reference 7. Related rules 8. Related guidelines	1. Abuse cases creation 2. Software design documents 3. Test plans for security 4. Tests Cases 5. Penetration tests
Historical Risks	Historical risk is identified during secure system development. Actually, it is a pair of condition and event, with its likelihood of occurrence and impact. Historical risks are very useful for early identification of potential issues in development process. It can also provide effective mitigation techniques and ideas for making risk management up to the mark to provide better security characteristics.	1. Type (business or technical) 2. Title 3. Subcategories(via taxonometric sorting) 4. Author 5. Owner 6. Project 7. Risk status 8. Likelihood 9. Impact 10. Severity 11. Risk context 12. Risk description 13. Estimated impact date 14. Potential cost	1. Architecture of Software 2. Software Design 3. Test plans 4. Fielded software

		15. Contingency plans and work around 16. Related risks 17. Related business goals 18. Related mitigation 19. Diagnostic methods	
Vulnerability	Vulnerability can be used to attack system. It may be due to programming error or software's defect which allows attacker to gain system's access.	1. Context Description a. Platform b. Operating system c. Language d. Application e. Version, and so on 2. Description 3. Title 4. Severity 5. Vulnerability type 6. Loss type 7. Reference	1. Code 2. Architecture of Software 3. Software design 4. Penetration tests 5. Fielded software
Exploits	Exploits is a particular act performed by the attacker.	1. Context Description a. Platform b. Operating system c. Language 2. Description 3. Title 4. Preconditions 5. Motivation 6. Exploit code 7. Exposure type 8. Related vulnerability 9. Blocking solution	1. Penetration tests 2. Fielded system.

Test Plan: Test plan must be developed with utmost care and should include:

1. Security related scenarios or test cases
2. Security testing related test data
3. Security testing test tools
4. Usage of different security tools to analyze various test outputs

Coding and Unit Testing

Developer must incorporate secure coding guidelines for implementing secure software. Developers must build in-depth knowledge of how vulnerability gets into software. They must also keep pace with learning to prevent system from sneaking into code programs and become able to differentiate design versus implementation vulnerabilities. Secure software developers and testers should attend and acquire proper and managed training sessions to be able to develop secure code by adhering secure coding standards. The development must be lined up with secure design, coding guidelines and standards. Testers should use secure coding standards and must develop test cases to verify with respect to the standard being followed to ensure system's security.

Security Test cases: Some sample test cases which can be used in security testing are:

1. Password used for verification should be kept in encrypted format in database.
2. Application or System should not allow access to invalid users.
3. Application's cookies and session time must be checked regularly.
4. Browser back button should be disabled while doing transactions on commercial websites.

Integration and System Testing

Black/white/gray box testing is performed in this phase. A virtual test environment is setup for performing extensive security testing. Testers must plan, track, and manage test environment setup activities and need to observe their performance. Tester is also responsible for installing hardware, software, and network resources on test environment. Afterwards, tester must integrate resources to obtain/refine test databases. At the end, tester develops environment setup scripts and test bed scripts.

Implementation

Functional requirements with security constructs are implemented in this phase. Along with implementation of functional requirements, uttermost care is given for implementing software's security. Security Testers develop security test scripts, and execute them to their refinement. They want to avoid false positives and/or false negatives by conducting evaluation activities. They document security problems by system reports, and support developers' understanding about software problems. They replicate issues to perform regression tests and to tackle problems closely.

Support

After developing a secure software using SSDLC, it is very important to put or execute a patch management process in place for managing vulnerabilities. In this phase, different internal and external vulnerabilities are identified, tracked and prioritized. Source code auditing and penetration testing is also accomplished in this phase so that a secure application environment can be maintained. The secure software development life cycle (SSDLC) and tasks associated with each phase are summarized in table 3.

CONCLUSION:

In this paper, software security is explained through various perspectives: how a user wants his system to work and how developers should use their creative minds towards creating a secured software system. The security is incorporated from the very first step of SDLC process to develop a secure system at end by incorporating necessary security features and measures. This analysis is performed to facilitate developers for developing secured software.

The developers, who are new to security world, will find it very beneficial. The study would help them to understand important security concepts for designing secure software system. If developers put their programming capabilities along with the above mentioned process, they would surely land up to a secure software system. It is also believed that it would be useful for general readers of security and for security enforcing team.

REFERENCES

[1] Gu Tian-yang, Shi Yin-sheng, and Fang You-yuan, "Research on Software Security Testing", World Academy of Science, Engineering and Technology, Vol. 70, p 647-651, September 2010.

[2] G. Hoglund and G. McGraw, Exploring Software: How to Break Code, Addison-Wesley, 2004.

[3] Gencer Erdogan, Ketil Stolen, "Risk-driven Security Testing versus Test-driven Security Risk Analysis", First Doctoral Symposium on Engineering Secure Software and Systems.

[4] K.K. Aggarwal, Yogesh Singh, "Software Engineering", (3rd ed.), Copyright © New Age International Publishers, 01-Jan-2005.

[5] Mohd. Ehmer Khan & Farmeena Khan, "A Comparative Study of White Box, Black Box and Grey Box Testing Techniques", International Journal of Advanced Computer Science and Applications, (IJACSA) Vol. 3, No.6, 2012.

[6] Jovanovic, Irena, "Software Testing Methods and Techniques".

[7] Ould, M. A. (1999). Managing software quality and business risk. Chichester: John Wiley & Sons.

[8] DeMarco, T. and T. Lister (2003). Waltzing with Bears: Managing Risk on Software Projects. New York: Dorset.

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

[10] Suhel Ahmad Khan, Raees Ahmad Khan "Software Security Testing Process", Proc. of the Intl. Conf. on Recent Trends In Computing and Communication Engineering-- RTCCE 2013, p39-42.

[11] Thompson, H.H., "Why security testing is hard", IEEE Security & Privacy, v 1, n 4, 83-6, July-Aug. 2003.

[12] B. Potter, G. Mcgraw, "Software Security Testing," IEEE Security & Privacy, v2, n5, 81-85, Sept.-Oct. 2004.

[13] S. Barnum, G. Mcgraw, "Knowledge for Software Security", IEEE Security & Privacy, v3, n2, 74-78, March-April 2005.

[14] J. Viega and G. McGraw, Building Secure Software: How to Avoid Security Problems the Right Way, Addison-Wesley, 2001.

[15] Gary McGraw, "Software Security: Building Security in", Addison-Wesley Professional, 2006.

[16] Online Documentation, August 2013. URL: http://www.guru99.com/what-is-security-testing.html.

[17] G. McGraw, "Testing for Security During Development: : Why We Should Scrap Penetratre-and-Patch," IEEE Aerospace and Electronic Systems, Vol. 13, no. 4, 1998, pp 13-15.

SHORT BIODATA OF ALL THE AUTHOR

Manju Khari (manjukhari@yahoo.co.in) is a research scholar with Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. She is an Assistant Professor in Ambedkar Institute of Advanced Communication Technology and Research, Guru Gobind Singh Indraprastha University, Delhi, India. She received her Masters degree in Information Security from Ambedkar Institute of Technology, Guru Gobind Singh Indraprastha University, Delhi, India. Her research interests are software testing, software quality, software metrics and artificial intelligence. She has published several papers in international journals and conferences.

Chetna Bajaj (chetnabajaj7@gmail.com) is a research scholar, pursuing Masters of Technology in Information Security from Ambedkar Institute of Advanced Communication Technologies and Research, Guru Gobind Singh Indraprastha University, Delhi, India. She has completed her B.Tech in Computer Science & Engineering from Guru Premsukh Memorial college of Engineering, Guru Gobind Singh Indraprastha University, Delhi, India. Her research interests are Near Field Communcation, software security testing and their tools, Mobile Ad-hoc Network, and Virus