International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:6, 2013

Comparative Analysis and Evaluation of Software Vulnerabilities Testing Techniques

Khalid Alnafjan, Tazar Hussain, Hanif Ullah, and Zia ul haq Paracha

***Abstract*—**Software and applications are subjected to serious and damaging security threats, these threats are increasing as a result of increased number of potential vulnerabilities. Security testing is an indispensable process to validate software security requirements and to identify security related vulnerabilities. In this paper we analyze and compare different available vulnerabilities testing techniques based on a pre defined criteria using analytical hierarchy process (AHP). We have selected five testing techniques which includes Source code analysis, Fault code injection, Robustness, Stress and Penetration testing techniques. These testing techniques have been evaluated against five criteria which include cost, thoroughness, Ease of use, effectiveness and efficiency. The outcome of the study is helpful for researchers, testers and developers to understand effectiveness of each technique in its respective domain. Also the study helps to compare the inner working of testing techniques against a selected criterion to achieve optimum testing results.

***Keywords*—**Software Security, Security Testing, Testing techniques, vulnerability, AHP.

I. Introduction

SOFTWARE testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item [6], [7]. In the process of testing software item is passed under specified conditions to observe it for particular aspects. There are two main goals of software testing one objective is that to probe the software for bugs so that these can be removed, the second objective is to ensure that the software works according to specifications. Software errors and defect give rise to vulnerabilities, which is the main cause of software failure. Software assurance is defined by department of defense DOD as "The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software" [22]. Most of the software contains flaws and errors that are often exploited to compromise the functions and security of the software. Software security assurance is an evolving subject and is much less mature than software quality assurance and software safety assurance. Software security assurance objective is to ensure the confidentiality, integrity and availability of software system by following different techniques and mechanism throughout the software development life cycle SDLC. Security testing activities are performed to validate security requirements and identify potential vulnerabilities. Standard software processes identifies all types of related to software quality attribute and software functional aspects but security vulnerabilities can also be discovered through standard testing process. The objective of the security testing is to assess security properties and behavior of the software as it interact with the external or internal entities interact regardless of the functionality that software implements. We choose five types of functional testing techniques which include both black box and white box approaches, these includes Source code analysis, Fault code injection testing, Robustness testing, Stress testing, and Penetration testing techniques. These techniques are first analyzed to understand how they work and how these can be used to identify security related vulnerabilities and bugs in software systems. Than these techniques have been compared based on a criteria which we think will help the software security testers and researches to select the optimum tool in particular scenario. Multi criteria decision support system MCDM based on analytical hierarchy analysis AHP has been used to evaluate the selected testing techniques. AHP is a structured based on mathematics and intuitive developed by Thomas L. Saaty [14] in 1970s and has been extensively used in fields such as government, business, industry, healthcare, and education. AHP enables the evaluation of inconsistency of the decision-maker known as consistency check, inconsistencies below 10% are accepted for matrices of the range n≥5 (5% for n=3 and 9% for n=4). Otherwise, the judgments made must be revised or the matrix discarded [3]. The study also helps to know the relationships of known vulnerabilities and how particular testing techniques deal with it.

Khalid Alnafjan, Tazar Hussain, and Hanif Ullah are with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: Kalnafjan@ksu.edu.sa, tazhussain@ksu.edu.sa, hanif@ksu.edu.sa).

Zia ul haq Paracha is with the Department of Management Information System, college of Business Administration King Saud University, P.O. Box 51178 Riyadh 11543 Saudi Arabia (e-mail: zparacha@ksu.edu.sa).

II. Software Security and Testing

The presence of Software errors during software development life cycle (SDLC) that leads to software vulnerabilities is very common and inevitable. Discovering vulnerabilities is a favorite activity of attackers who want to use the software systems for their own benefits. In 2008, 6058 vulnerabilities were catalogued by CERT [1] NIST national vulnerability database [5] and common vulnerability management [11] contain data about software flaws and errors. These statistics indicate the fact how software can be used to compromise the system if an attacker attacks with evil

intents. The objective of the software testing is to cause failures in order to make fault visible [7] so that these faults can be removed. Security testing emphasizes what the software should do in relation to confidentiality, integrity and availability but the emphasis on "what the software should not do" is much more unlike traditional testing. Security testing must ensure to consider all the security requirements and these cannot be dropped unlike traditional testing requirements. Software testing helps contributes towards developing secure software by testing insecure programming practices and testing can also identify flaws which are not visible at architecture level. Therefore traditional software testing can be used with security in mind, based on the knowledge about software internals the software testing is of two major types that is black box and white box testing.

A. Black Box Testing

In this type of testing the software code is considered as "black box" and the tester has normally very little or no knowledge of system under test or when the source code and internal mechanisms of the system are not available. Black box security tests are performed on executable software and used a variety of inputs to simulate the behavior of attackers and other misusers. In this regard black box testing plays a very vital role to ensure input validation and checking it also identifies some serious security vulnerabilities e.g. Sql injection, buffer overflows and cross site scripting etc.

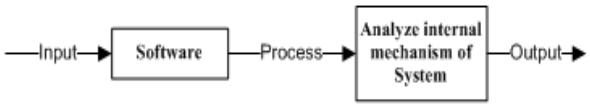

Fig. 1 Black Box testing process

Black box testing activities are carried out throughout the SDLC and help to test security in following areas. Black box tests are performed to evaluate the behavior of COTS, executables packages, it also examine the interaction of the software with the environment such as attackers and external entities. This type of test is not possible with white box mechanism. Uncover security issues that arise as a result of missing modules, packages and files. Discover potential security issues resulting from boundary conditions.

B. White Box Testing

The type of testing that takes the internal mechanism of the system into account and is performed when the source is available. Because the white box testing has access to the source code in internal mechanism it has capabilities to identify coding errors, data flow, and error handling etc to evaluate software for security requirements. Static and dynamic source code analysis is the core activities performed as part of white box testing. To perform white box security testing one must have the knowledge about how to develop secure and avoid insecure systems, how to think like an attacker.

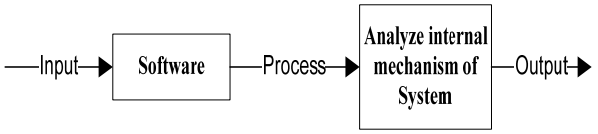

Fig. 2 White box testing process

White box testing helps to ensure software security and identify some common and serious errors.

III. Security Testing Techniques

We have selected the five major types of testing techniques that performs crucial role in security enhancements, both white box and black box techniques are included in the selected techniques. Although some of the techniques behaves as hybrid, the following section analyze the selected techniques and terms of its working mechanism, performance, types of security flaws identified, and effectiveness.

A. Source Code Analysis

Source code analysis is the process of analyzing the source code, before compilation (static analysis) or analyzing the both source code and executable (dynamic analysis), for coding errors, insecure practices and vulnerable code. In manual code analysis the tester inspect the source code for vulnerable code such as finding strcpy () functions without the use of a tool. But modern security testing analyzers are much more sophisticated in term of identifying bugs, it also reduces false alarms. In dynamic source analysis the compiled executable is run and feed as input for testing the program variables in order to detect code behavior. Depending on the type of testing tool some errors and discrepancies are identified but some are harder to be identified.

Source code analysis tools [9], [10] has the ability to examine calls in the argument to insecure library functions, e.g. the C/C++ testing tools have the ability to preprocess the source code which enable the analyzing tool to see the same code as seen by compiler.

Bound detection and checking error functionality enable these tools to detect vulnerabilities due to integer overflow, integer truncation and unsigned underflow etc.

To detect vulnerabilities associated with incorrectly implemented sequences of operations, security analyzers often look for specific library function calls and print a warning about potential security problems associated with those functions.

Pointer aliasing is a static analysis that tries to solve the problem when two pointers pointing point to the same data as explained in [9], [10]

B. Fault Code Injection

In this type of testing the bugs are intentionally injected into the code, the code is then compiled and executed so the tester can determine how software reacts when it is forced in anomalous states. Fault code injection increases the robustness and reliability by identifying incorrect use of pointers and arrays, the presence of dangerous calls and race condition.

This type of testing is used in situation where high assurance is required against well known serious vulnerabilities but is a complex process because every scenario cannot be simulated. Fault propagation analysis it is not only observed that how code behaves with injected faults but it is also the propagation of the fault (in the source code) is analyzed through fault trees. This enables the tester to determine the impact of a fault on a module, and system as a whole. Interface propagation analysis enables the tester to determine how a fault in one component affects other component of the system.

C. Stress Testing

Also known as load or performance testing, in stress testing the system is passed through stressful states to expose vulnerabilities arises as a result of when software are exposed to maximum design load and beyond it.

D.Software Penetration Testing

Penetration techniques have long been used in network security but this testing technique has also made it place to penetrate software systems for faults and bugs. Software penetration testing is the type of black box which focuses on vulnerabilities having external access. The idea of penetration testing is more like ethical hacking that is "attempt to compromise the security of the systems under test". Penetration testing helps to expose complex vulnerabilities e.g. vulnerabilities arises as a result of inter and intra component communication or communication of software to its resources and environment. In software security one of the vital activities is to increase the test coverage and penetration tests can be more extensive in its coverage. Penetration testing currently faces two major challenges that is a push towards automation and minimizing the cost in term of labor time associated with test cases. Although in penetration testing the systems is seen as an outside attacker might see it and is therefore consider as black box mechanism but it can also be used in white box fashion.

E. Vulnerability Scanning

In this type of testing the software is scanned for well known vulnerabilities based on repository of "signatures" to observe software's behavior associated with attack pattern. Host based scanners sophisticatedly analyzed the internal of the system such as the insecure configuration, while network based scanners are good to analyzed attack carried out from outside remotely. Vulnerabilities scanners exercise vulnerabilities on the target system, it has the ability to probe every network service and applies all available "signatures". Scanners observe the application for vulnerabilities like buffer overrun, cookie manipulation, Sql injection, and cross site scripting etc. vulnerability scanner works in black box manner and can be used only against small set of attack pattern.

IV. Background Study

Testing is an essential process to evaluate the quality of software, software community has discussed the topic from different perspectives including the cost of testing, testing methodologies, and limitation of testing process. But less research work has been carried out in the field of security testing to ensure software assurance and reliability. Reijo Savola and Kaarina Karppinen [15] have used security testing for telecommunication systems and argued that security requirements are within the focus of the information security testing process. Besides this security testing has been used in literature in various domains [12], [17], [16]. But the following section of related study summarized points related to our work, this data has been used for evaluation of the techniques we have identified earlier.

A. Data Collected for Analysis

Literature review and research explains various testing techniques in terms of effectiveness, coverage, efficiency, security testing capabilities, pros and cons and cost etc. the following section summarized the major observations about testing techniques gained from literature, experience and research.

- White box techniques (source code analysis, fault code injection) have been proved better in term of detecting vulnerabilities (sql injection, buffer overflow) [22]. However black box techniques such as penetration testing and vulnerability scanning are better in term of cost (time and resources consumption).
- Fault injection (white box) techniques can be used to increase the coverage of hard to reach parts of the program [4], [20].
- The black box techniques (penetration tests) have been proved better to identify interfaces errors, faulty functions, data structure errors with less cost and specialized skills [8], [19].
- Penetration testing has no direct access to source code so therefore have limited in term of coverage analysis [21].
- The experimental study in [2] indicated that the coverage analysis for source code analysis is higher than penetration techniques however penetration techniques have less false positives.
- Insecure coding, or coding errors are the main source of software exploitation but source code analysis and fault injection mechanism can be used to quickly identify coding errors. Also penetration testing can be modified in white box manner to reduce time consumption.
- Dynamic and static analysis and fault injection techniques required more time consumption, required specialize skills.
- Stress testing and penetration techniques concentrate on checking and validation, SQL insertion attacks, injection flaws, cross-site scripting attacks, buffer overflow vulnerabilities [20].

V. Methodology and Implementation

Multi-criteria evaluation is a fundamental step of the rational decision-making process in order to gain reliable information on strengths, weaknesses and overall utility of each option. The purpose of our study is to identify and

analyze the strength and weaknesses of security testing techniques in particular direction. The process is several steps including selecting a goal, list criteria/subcriteria, determining the alternatives, assignment of priorities, calculation of weights, results and discussions. These steps have been explained in the following sections.

A. Selecting Goal and Objectives

The goal of this work is to analyze the role security testing techniques to base on criteria/subcriteria to help the testers in applying these techniques according to requirements efficiently.

B. Criteria/Subcriteria to Evaluate Testing

Software tester and professionals have different option available to test software at different level of abstraction; depending on the security requirements of the system testers normally prioritize the security tasks. We have selected five key criteria (standard) to evaluate software testing mechanisms against them, the following section briefly introduce those criteria and why they are important.

1. Cost

To use a particular technique it is vital to understand its cost in term of skills required, labor time to develop and execute test cases, tool and utility support and integration. We have two subcriteria 1. Skills required 2. Testing time, the subcriteria contribute to main criteria.

2. Thoroughness

Thorough check that every segment of software has been tested is required to secure it, it also encompasses that every possible interaction during runtime has been covered. White box technique offers the opportunity to be more through as it can see inside the code. We have divided this criteria into two subcriteria; 1. Coverage 2. Completeness. Coverage or code coverage analysis is an important measurement of the effectiveness of a testing tool. Code coverage determines the degree of covered paths, flow and statement during a test process. Completeness means that the entire code or modules have been covered through test cases and is closely relevant to code coverage.

3. Ease of Use

Particular support or facilities provided by testing technique and its tools to ease the process testing. We have to have subcriteria integration means that how tightly a testing technique is integrated to the application under test. Platform and tool support is another relevant sub criterion to denote how well particular technique support is available for different platforms and also the degree of interoperability with other testing techniques.

4. Effectiveness

In our scenario security testing effectiveness means how well the security bugs have been identified by particular testing technique or the number faults identified by the technique. According to Weyuker [18] “effectiveness of a test technique is only possible to measure if you can compare two techniques for the same set (i.e. software), but the result is not general”

5. Efficiency

Denotes the testing consumed resources [13] such as time, testing resources, the amount of code required.

Assigning Priorities

The priorities are assigned to criteria subcriteria and alternative on the basis of the Table I. Priorities are the numbers assigned to criteria, subcriteria associated with an alternative.

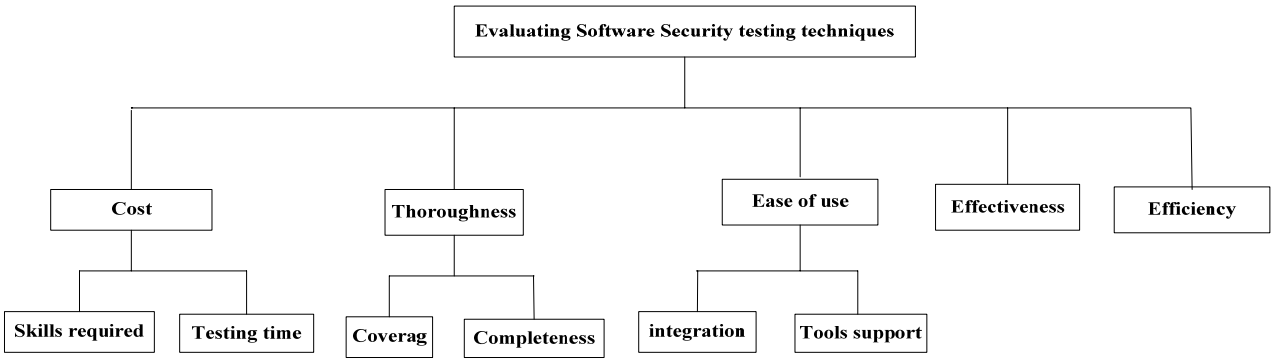

Fig. 3 Hierarchal Block diagram for criteria and subcriteria

Each alternative has been evaluated against criteria and subcriteria and the priorities have been assigned in the form of weights. We have assigned priorities to criteria/subcriteria against each alternative by using the previous studies results and the results obtained from the outcome of the testing tools of selected techniques. The basis of assigning priorities has been discussed in the background study of this paper. Calculated Local and global weights for the main criteria and subcriteria in the through MCDM tool has been shown in Table II.

TABLE I
PRIORITIES WITH THEIR IMPORTANCE

Intensity	Importance	Intensity	Importance
1	Equal Importance	6	Strong Importance plus
2	Weak Importance	7	Very Strong Importance
3	Moderate Importance	8	Very Strong Importance plus
4	Moderate Importance plus	9	Extreme Importance
5	Strong Importance		

TABLE II
LOCAL AND GLOBAL WEIGHTS FOR CRITERIA AND SUBCRITERIA

Cost		Thoroughness		Ease of Use		Effectiveness	Efficiency
L=11.4% G==11.4%		L=34.6% G=34.6%		L= 8.2% G= 8.2%		L= 42.1%	L= 3.8%
Skills Required	Testing time	Coverage	Completeness	Integration	Tool support	G= 42.1%	G= 3.8%
L=25.0%	L=75%	L= 80.0%	L= 20%	L=75%	L= 25%		
G=2.9%	G=8.6%	G= 27.7%	G= 6.9%	G=6.1%	G= 2.0%		

TABLE III
EVALUATION IN CONTEXT OF: EVALUATING AND COMPARING SOFTWARE SECURITY TESTING TECHNIQUES

Thoroughness vs. Efficiency	Effectiveness vs. Thoroughness	Cost vs. Ease of Use	Effectiveness vs. Cost	Effectiveness vs. Ease of Use
4 : 1	2 : 1	3 : 1	4 : 1	5 : 1
Cost vs. Efficiency	Ease of Use vs. Efficiency	Thoroughness vs. Cost	Effectiveness vs. Efficiency	Thoroughness vs. Ease of Use
3 : 1	3 : 1	4 : 1	4 : 1	5 : 1

VI. RESULTS AND DISCUSSION

This section presents the results obtained using MCDM systems, fig. 4 shows the alternative ranking against criteria and by observing the chart given in the figure some useful information could be obtained. For security vulnerability effectiveness and thoroughness source code analysis has topped the list, but the relative cost of the source code analysis is also relatively high. Vulnerability scanning the type of testing which has been proved as a less effective but also it consumes less resource. In term if effectiveness and coverage analysis Penetration testing and fault code injection methods are at number two and three respectively but the cost of the fault injection method is relatively high.

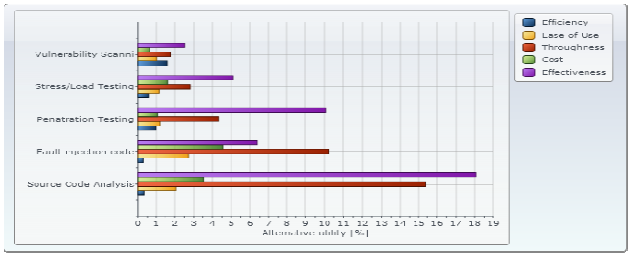

Fig. 4 Alternative ranking against criteria

Fig. 5 shows the alternative comparison in a diagonal graph, the figure highlight the fact that source code analysis being more twisted towards effectiveness and the cost of fault code injection is relatively on higher degree.

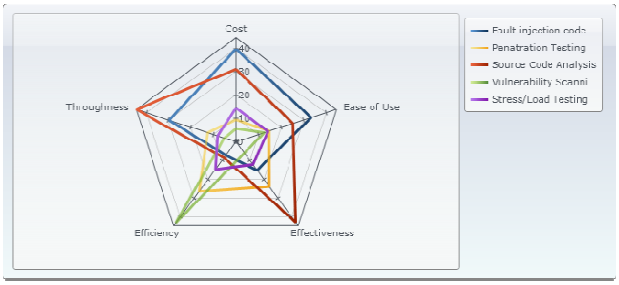

Fig. 5 Alternative comparison

Fig. 6 illustrates the pair wise comparison of our main criteria in percentage, using pair wise comparison the relative importance or preference of one criterion over another has been expressed. Because it is an important in testing to measure the number of defects per test case, therefore effectiveness has comparatively more important than other criteria.

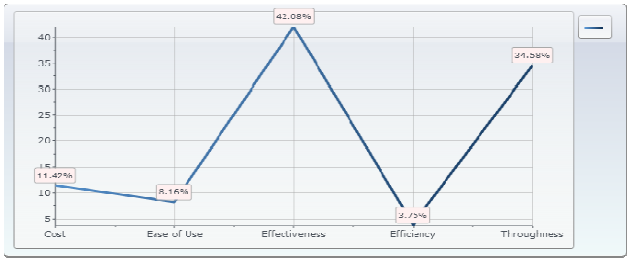

Fig. 6 Criteria weight preferences and percentage

Fig. 7 depicts the graph for effectiveness, static and dynamic source code analysis is thirty degree higher than

vulnerability scanning. Penetration testing and fault code injection methods score is also good for effectiveness.

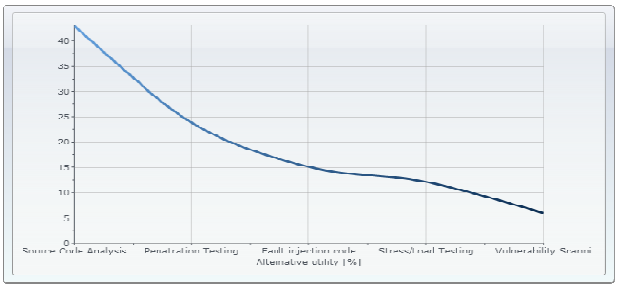

Fig. 7 Alternative ranking for Effectiveness

VII. Conclusion

Software testing plays crucial role to ensure software quality assurance but in this paper we have analyzed five testing techniques to check its application to software security. Each testing technique identifies various types of vulnerabilities in software according to its own way and logic. The study carried out in this paper use MCDM method and tool to evaluate each alternative and the study results show the strength and weaknesses of each technique. According to this study the most effective technique to identify and cover more code for vulnerabilities and bugs is both source code analyses. Penetration testing and fault code injection also effective but fault code injection is more time consuming and needs special skills. Vulnerability is the weakest one in term of effectiveness but require less skills, time and resources. The results of this study can be used by testers before developing test cases to optimize testing and reduce the security risk with optimum resources and time at different phases of SDLC. In future we intend to extend this study by comparing testing techniques against specific type of threats such as sql injection and buffer overflow in real world scenario.

Acknowledgment

This work was supported by the Research Centre of College of Computer and Information Sciences, King Saud University. The authors are grateful for this support.

References

[1] (CERT/CC), C. C. C. "Cataloged vulnerabilities." from http://www.cert.org/stats/

[2] Antunes, N. and M. Vieira (2009). Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services. Dependable Computing, 2009. PRDC '09. 15th IEEE Pacific Rim International Symposium on.

[3] Aznar Bellver Jerónimo, C. R., Roberto, Romero Civera, Agustín (2011). "New Spanish Banking Conglomerates.Application of the Analytic Hierarchy Process (AHP) to their Market Value "International Research Journal of Finance and Economics (78).

[4] Bieman, J. M., D. Dreilinger, et al. (1996). Using fault injection to increase software test coverage. Seventh International Symposium on Software Reliability Engineering, 1996. Proceedings.

[5] Database, N. V. (2012). "National Vulnerability Database Version 2.2." Retrieved 10/08/2012, 2012, from http://nvd.nist.gov/.

[6] IEEE (1986). ANSI/IEEE Standard 1008-1987, IEEE Standard for Software Unit Testing.

[7] IEEE (1990). IEEE Standards Collection: Glossary of Software Engineering Terminology, IEEE Standard 610.12-1990.

[8] Khan, M. A. and M. Sadiq (2011). Analysis of black box software testing techniques: A case study. Current Trends in Information Technology (CTIT), 2011 International Conference and Workshop on.

[9] Lavenhar, C. M. a. S. R. "Code Analysis Tools - Overview." Retrieved 02/06/2012, from at https://buildsecurityin.us-cert.gov/bsi/articles/tools/code/263-BSI.html.

[10] Lavenhar, S. (2006). "code Analysis." Retrieved 01/06/2012, from https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/code/214-BSI.html.

[11] Mitre. (2012). "Common vulnerability managment database." Retrieved 10/08/2012.

[12] Romero-Mariona, J., H. Ziv, et al. (2010). Increasing Trustworthiness through Security Testing Support. Social Computing (SocialCom), 2010 IEEE Second International Conference on.

[13] Rothermel, G. and M. J. Harrold (1996). "Analyzing regression test selection techniques." Software Engineering, IEEE Transactions on 22(8): 529-551.

[14] Saaty, T. (1980). The Analytic Hierarchy Process. New York, McGraw Hill.

[15] Savola, R. and K. Karppinen (2007). Practical Security Testing of Telecommunications Software--A Case Study. Telecommunications, 2007. AICT 2007. The Third Advanced International Conference on.

[16] Shahriar, H. and M. Zulkernine (2009). Automatic Testing of Program Security Vulnerabilities. Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International.

[17] Thomas, L., X. Weifeng, et al. (2011). Mutation Analysis of Magento for Evaluating Threat Model-Based Security Testing. Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual.

[18] Weyuker, E. J. (1993). Can we measure software testing effectiveness? Software Metrics Symposium, 1993. Proceedings., First International.

[19] Will Radosevich , C. C. M. (2009). "Black Box Security Testing Tools." Retrieved 31/05/2012, 2012, from https://buildsecurityin.us-cert.gov/bsi/articles/tools/black-box/261-BSI.html.

[20] Wyk, G. J. a. K. v. (2009). "White Box Testing." Retrieved 31/05/2012, 2012, from https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/white-box/259-BSI.html.

[21] Wyk, K. R. v. (2007). "Adapting Penetration Testing for Software Development Purposes." Retrieved 03/06/2012, 2012, from https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/penetration/655-BSI.html.

[22] Wysopal, C. (2009). White Box Better Than Black Box Retrieved 31/05/2012, 2012, from http://www.veracode.com/blog/2009/10/white-box-better-than-black-box/.