International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:6, 2016

A Proposal for Systematic Mapping Study of Software Security Testing, Verification and Validation

Francisco Nunes, Adriano Bessa Albuquerque

***Abstract*—**Software vulnerabilities are increasing and not only impact services and processes availability as well as information confidentiality, integrity and privacy, but also cause changes that interfere in the development process. Security test could be a solution to reduce vulnerabilities. However, the variety of test techniques with the lack of real case studies of applying tests focusing on software development life cycle compromise its effective use. This paper offers an overview of how a Systematic Mapping Study (MS) about security verification, validation and test (VVT) was performed, besides presenting general results about this study.

***Keywords*—**Software test, software security verification validation and test, security test institutionalization, systematic mapping study.

I. Introduction

COMMUNICATION and Information Security (CIS) has its foundation in the principles of Availability, Integrity and Confidentiality. It is understood that, in this way, areas or processes where security is mandatory must also ensure compliance with these principles to be considered secure, such as: Logical access control or Software development process. About development, a considerable lack of resources and control structures that facilitate or enable systems to meet the above principles is still not evident. Hence, it follows that problems and failures during software development cycle are generating increasingly insecure applications that weaken business processes and threaten corporate information assets.

The quest for software security should start with improving software development processes. Thus, software cannot be considered secure by having just implemented some security functionality, such as access control to protect confidentiality [1]. One of the consequences of misalignment between information security and development methodologies is the growth in the number of attacks against applications that generates rework to correct vulnerabilities and find root causes. For example, the statistic from CERT [2] highlights the increasing level of incidents of CIS. Moreover, the exploitation of vulnerabilities found in software products has increased causing changes in techniques and approaches in software development processes. Software insecurity is a reality that concerns those involved with development and maintenance projects, from managers to technicians, due to the perspective that incidents can impact negatively the availability of processes and services and, above all, the confidentiality, integrity and privacy of corporate information. This concern may be justified by the lack of knowledge, lack of maturity, or ineffective actions that attempt to solve software insecurity situations. The security test is, in many cases, the only alternative to reduce vulnerabilities in applications. However, the diversity and complexity of techniques and, in particular, the lack of reports about security test institutionalization focusing on the software development life cycle, complicate the broad and appropriate use of these test practices in the corporate environment. Thus, the approach of systematic study was conducted to characterize existing studies correlating structure and use of security VVT techniques as well as identify any influence in improving the software development process. This paper aims not only to describe the most relevant aspects of how MS occurred but also present the overall results achieved with the approach.

This paper is organized as follows: Section II summarizes the main points of verification, validation, and security testing; Section III presents a brief description of Systematic Literature Review; Section IV describes the approach used to perform the Systematic MS; Section V discusses the main findings of the study; and Section VI summarizes the main conclusions of this work.

II. Software Security VVT

Software security VVT practices aim to correct most of the vulnerabilities in order to provide quality and reliable software. Systems must undergo security tests which can be executed in various stages of the system development life cycle in order to increase the confidence that applied controls work effectively in the application [3].

Sommerville [4] states four complimentary approaches to verify a software protection condition:

- Validation based on experience: The system is analyzed in comparison with known types of attacks by validation team.
- Validation based on tools: Multiple security tools, such as password checkers, are used to analyze the system.
- Validation based on invasion: A team is formed whose objective is to break the protection system. The team simulates attacks to the system and uses its experience and knowledge to discover new ways to compromise system safeguards.
- Formal verification: A system can be checked against a formal specification of security.

Considering the activities of the validation phase of a development process, the following practices adjust better to

Francisco Nunes is with Banco do Nordeste do Brasil, Fortaleza, Brazil (corresponding author, phone: 00-55-85-32515522; e-mail: fcojbn@yahoo.com.br).

Adriano Albuquerque, Dr., is with University of Fortaleza, Fortaleza, Ceara, Brazil (e-mail: adrianoba@unifor.br).

security tests [4]:

- Requirement inspections: Find problems in requirements specification.
- Management of requirements: Track changes in requirements and coordinate impact of this change on development project.
- Design and code inspection: Discover defects before system security audits do.
- Static analysis: Perform automated analysis to find erroneous conditions.

III. Systematic Literature Review - SLR

Systematic Literature Review (SLR) assumes that it is feasible to obtain consistent evidences about the presence or lack of a predefined condition or phenomenon, based on preselected information (population). According to [5], evidence is defined as a synthesis of the best quality scientific studies on a specific topic or research question.

SLR is a means of evaluating and interpreting a research question, results, subject, or phenomenon of interest, relevant and available. Kitchenham [6] states that SLR is a coherent assessment of a research topic from a methodologically reliable, accurate, repeatable, and auditable approach. Therefore, the SLR is of benefit to Software Engineering, including its related processes, such as the testing process, since it helps to obtain some evidences originated from research in scientific basis [6]. These evidences can contribute with more experience and improvement of security verification, validation, and testing during development and maintenance of software.

Systematic literature reviews are referenced as a secondary study and studies that SLR analyzes are referred to as primary studies. According to [7], results originated from primary studies provide evidences which can be investigated by secondary studies. Kitchenham et al. [8] presents two different types of SLR:

- Conventional SLR: Combine results related to a specific research question, for example: “Would test approach 'A' be more effective in detecting defects than approach 'B'?”.
- Mapping or Systematic Study (MS): These studies are aimed at finding and classifying primary studies in specific knowledge area and can identify available literature before performing conventional SLR.

Kitchenham [6] divides the process of SLR into three main phases: Plan review, conduct the review, and Report result. During planning, a protocol is prepared that specifies the approach to be conducted to perform a systematic review as well as indicates the level of rigor and completeness. Such protocol is necessary to reduce deviations or inconsistencies in the results of scientific research. When the protocol is completed, it begins the search for primary studies.

In the protocol, it should specify the purpose (question) of research as well as procedures to be conducted to review the literature, including the query used in the search for specific content. The protocol also indicates the inclusion and exclusion criteria which select and indicate explicitly those primary studies that will participate or not in the study.

The aim of the conducted systematic study was to identify relevant material stating results and experience in institutionalizing software security ver VVT actions in a development process (life cycle), in such a way that it can be perceived as a process improvement initiative to produce more secure applications. To accomplish such objective, method and protocol proposed by [6] were followed as a model.

The next section presents a general description of how the MS was conducted, including the components of the defined protocol that make it possible to replicate the study.

IV. Application of a Systematic Mapping

The scope to apply this systematic mapping relates to experiences or initiatives in software security verification, validation, and test in an iterative and incremental process. In other words, the research question focuses on identifying the reflection of adaptation or improvement of the development process by institutionalizing practices of security verification, validation, and tests that are being used in the corporate environment.

The study was based on the classification of research approaches described by [9]. The contribution facet used was adapted from [10].

A. Research Questions

According to [11], one of the differences between a systematic literature review and a systematic MS is the fact that the research questions of the mapping are more general. Thus, the research questions that guided this study include:

- What are the most researched software security VVT practices? (According to the mapped scientific papers).
- What are the advantages and difficulties of performing software security VVT practices researched by mapped papers?

Based on the proposal presented by [6] and [12] of subdivision of the research question in separated categories, also known as PICOC approach (P - problem or population, I - intervention, C - comparison, control or comparator, O - outcomes, C - Context), Table I shows this division.

TABLE I
Subdivision of Research Question – PICOC Approach

Population	software life cycle OR software development OR software
Intervention	security test OR secure test OR security verification OR security validation
Comparison	assess OR understand OR know
Outcome	effect OR experience OR result OR adopt
Context	corporate

B. Research Process

The research took place from March to August, 2014. Initially, the scope of the questions was restricted to refer only to Unified Process or RUP [13]. However, this approach presented inadequate since no evidence directly related to the effect of applying software security VVT has been found. Therefore, the scope was widened so that evidence of experience in applying security VVT in any iterative and incremental software development life cycle would be

considered. Once the scope was agreed, it was defined the inclusion and exclusion criteria (Table II), and the search string (Table III).

TABLE II
INCLUSION AND EXCLUSION CRITERIA

Criteria	Description
Inclusion	• Paper demonstrating some experience or assessing the adoption of software security VVT practices in a development project. • Full paper accepted in scientific journals or conference proceedings.
Exclusion	• Paper without any experience description in applying software security VVT. • Paper not directly related to software security VVT practices or approaches in a software life cycle domain. • Oldest paper by the same authors. • Paper whose focus is only to propose security VVT tools. • Gray literature and technical reports. • Short or extended paper. • Monographs, dissertations and theses.

TABLE III
SEARCH STRING

("software life cycle" OR "software development" OR software) AND ("security test" OR "secure test" OR "security verification" OR "security validation") AND (assess OR understand OR know) AND (effect OR experience OR result OR adopt)

C. Data Collection and Analysis

In order to simplify the *modus operandi* of the MS, its implementation was divided in five phases:

- Phase 1 - The most common method of SLR described by [6] was used. That is, instead of a manual research process, considered restricted; a broad automated research was conducted [8], considering only the search string in Table III. Automatic research involved the following scientific and digital databases of papers: Web of Science (Thomson Reuters Scientific), Compendex (Engineering Village 2), SCOPUS (Elsevier), ScienceDirect (Elsevier), ACM Digital Library, and IEEE Xplore. Table IV shows the general criteria, specific criteria of each database, where applicable, as well as some peculiarity of research performed, and, finally, the amount of records found.
- Phase 2 - After identifying the quantity of items listed from the search string, manual search was performed by analyzing paper abstracts, introductions, and conclusions to separate papers related, directly or indirectly, with software security verification, validation, and test, following the inclusion and exclusion criteria from Table II. Search from Phase 1 returned many results with little relevance. Fig. 1 provides an overview of the results of this phase.
- Phase 3 – Based on the results of Phase 2, it was decided to organize the papers using a manual selection, according to the four questions listed as follows:

i. In which category (model based, source code, executable, etc.) is the security test being applied?
ii. Based on the stage of the life cycle, which security test techniques (injection, static analysis, dynamic analysis, etc.) are being used in each phase of the cycle?

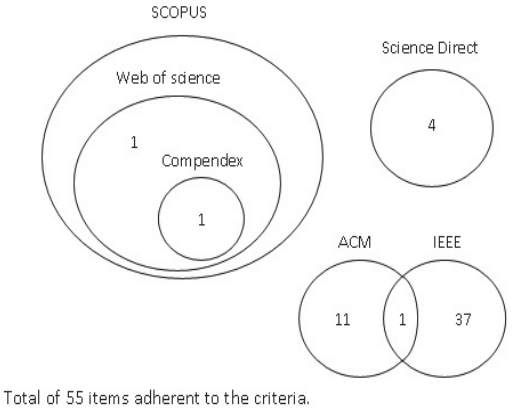

Fig. 1 Vision of results from Phase 2

TABLE IV
SEARCH CRITERIA OF DIGITAL SCIENTIFIC BASES

Scientific base	Criteria details	# Papers
--	• General criterion: "command search" was used (text box to insert all the search command). • There was no restriction in the search.	--
Web of Science	• An error appeared in the survey. So, it was used a variance of the search string, suggested by the scientific base: TS=(("software life cycle" OR "software development" OR software)) AND TS=(("security test" OR "secure test" OR "security verification" OR "security validation")) AND TS=((assess OR understand OR know)) AND TS=((effect OR experience OR result OR adopt)).	04
Compendex	• Search from the website www.engineeringvillage.com, with date starting at year 1980.	03
SCOPUS	• There was no restriction in the search.	21
ScienceDirect	• "Expert Search" was used. The search was refined for "Journals" and "Computer Science", with publication date starting from 1980. • The only restriction was the type of publication, with selection of "Journal", "Proceeding" and "Transaction".	166
ACM Digital Library	• The result was inconsistent and showed error when changing the list of paper. Thus, the search string was adapted to: (software AND ("security test" OR "secure test" OR "security verification" OR "security validation") AND (assess OR understand OR know) AND (effect OR experience OR result OR adopt)).	336
IEEE Xplore	• Option "full text and metadata" was chosen.	686

TABLE V
RESULTING ARTICLES FROM PHASE 4 (SNOWBALL APPROACH)

Paper	Backtrack
[19]	[70]
	Forward
	[44]
	[71]
	[72]

iii. A real case study (pilot project) or experimental / simulation validation was performed? (Real case study involves a pilot in a commercial / enterprise application, outside the academic environment, with the application of the pilot in development, ongoing maintenance or approval).

iv. Does the paper describe some result or experience to apply security verification, validation or test activity (approach, method, model, framework, methodology or technique) with focus on software development process (life cycle)?

The organization of papers among the previous questions contributed to restrict the analysis. So, it was decided to map only the papers related to real case and focusing on the process which resulted in a total of three papers.

- Phase 4 – The three papers from Phase 3 were refined by the snowball technique [14] using Google Scholar, considering criteria of Table II, as presented in Table V. The snowball identifies two variations:

a. *Backtrack*: After reading abstract and assessing its conformance, the references of each paper were analyzed, checking papers that apply. In this case, the applicable references were researched, at least, by reading the abstract, introduction, and conclusion.

b. *Forward*: Papers that referenced the paper under review were searched, identifying the applicable, ones. In this case, these references were analyzed, at least, by reading the abstract, introduction, and or conclusion.

- Phase 5 – With the result of the snowball, it could be possible to confirm or refute the existence of any primary study that can, in some way, answer questions from the protocol. As a complimentary objective, it was expected that the study showed, among software security VVT practices, which, in fact, have empirical evidence of their effectiveness. However, deciding on a level of effectiveness requires an associated metric that reflects the meaning of this effectiveness, such as: number of security vulnerabilities found in the software produced. Nonetheless, it will only be possible to assess the effectiveness if the MS indicates, among papers found, the effect of adoption as well as advantages and disadvantages of the practices.

V. Comparative Analysis and Results

TABLE VI
Complete Results from MS

Base	Phase 1	Phase 2	Phase 3	Phase 4
Web of Science	4	2	0	0
Compendex	3	1	0	0
SCOPUS	21	2	0	0
ScienceDirect	166	4	1	4
ACM Digital Library	336	12	0	0
IEEE Xplore	686	38	2	0

Table VI summarizes and compares the results from phases 1, 2, 3 and 4, presented in Section III of this paper.

Table VII includes general conclusions arising from Phase 3 and points to the respective graphical representation (Figs. 2 and 3).

The MS has offered evidences to confirm the existence of studies that answer questions of protocol. The main conclusions arising from the study, corresponding to Phase 5, include:

- It was noticed that different practices of security verification, validation, and test were used interchangeably. Nevertheless, the snowball confirmed that static and dynamic analyzes act as the most surveyed practices.
- Although it was realized that papers exalt the positive effects (benefits) of the proposed practices, compared to others, it was not possible to assess the veracity of this effect, since few articles report numbers or results which could prove the effectiveness of the technique investigated. It is therefore not possible to confirm which are the advantages and disadvantages of those most adopted practices.
- Although the practices (categories) based on model, source code or executable are more used, this does not imply that they have more or less benefits than others, since, for each paper reviewed, it was evident that the researcher highlights the benefits of the studied technique compared to the other(s). Nevertheless, the variety of types of systems tested, their criticality, and its context of use (operation) influence the effectiveness or ineffectiveness of a practice. It is also stated that there is a combined use of practices in order to add facilities and overcome difficulties or gaps in functionality.

The representation stated in Table VIII shows the distribution of the type of papers by contribution facet, for all the 55 papers included in our study. According their contributions, some papers were classified under more than one facet. For example, [18] made four contributions: (1) test based on real case, (2) paper referencing a test category of executable code, (3) paper referencing a test category of source code, and (4) paper referencing a test technique of static analysis.

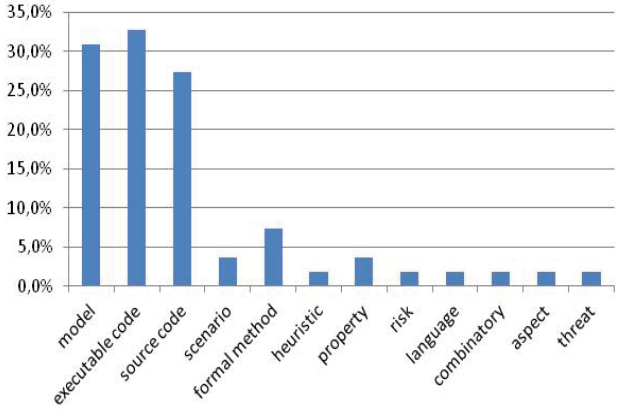

Fig. 2 Categories of verification, validation and security testing

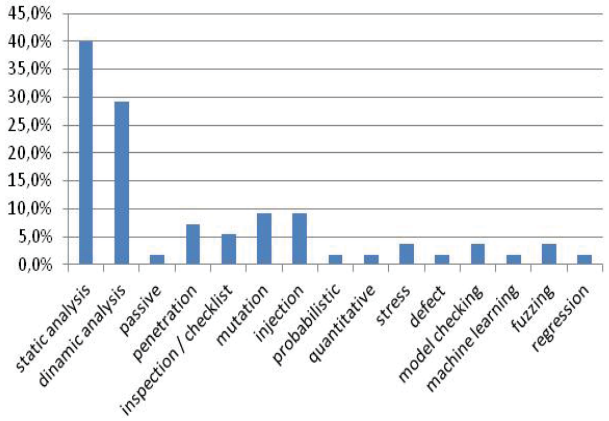

Fig. 3 Techniques of verification, validation and security testing

TABLE VII
CONCLUSIONS FROM PHASE 3

Question	Conclusion
i	The three main categories perceived include "executable code", "model" and "source code", respectively 32.7%, 30.9% and 27.3% (Fig. 2). This result can be explained by the fact that these categories are the most studied and known by many researchers and software testers.
ii	Although the most used techniques are "static analysis" (40%) and "dynamic analysis" (29,1%), the papers demonstrated that, in general, more than one technique is used in conjunction, complementarily (Fig. 3).
iii	Few articles involve corporate environment (real case), 18%, against 82% for experimental validation. This conclusion is attributed to the fact that the majority of practices are conducted in academic environment, more controlled and predictable, which facilitates the experiment.
iv	Few articles focus on software development process (life cycle), only 15%. Most of the papers deal, directly, with practices of software security verification, validation or test, explaining its evolution and applicability as well as restricting the context in the development process. One of the probable explanations may be the limitation of pages in a paper that prevents more details.

VI. FINAL REMARKS

TABLE VIII
CONTRIBUTION FACET

Based on real case	[18]-[20], [24], [25], [37], [42], [46], [49], [55], [65]
Based on process	[19], [34], [37]-[39], [42], [43], [49], [53]
Based on model	[15]-[17], [20], [21], [28], [29], [35], [39], [40], [52], [54], [58], [62], [65], [67], [68]
Executable code	[17], [18], [22], [23], [27], [33], [38], [41], [43], [45]-[48], [53], [55], [63], [64], [69]
Source code	[18], [24], [30], [31], [36], [41]-[43], [45], [53], [55], [60], [61], [63], [66]
Static analysis	[15], [16], [18], [24], [26], [29], [31], [32], [34], [37], [42]-[45], [52], [53], [55], [57], [59], [63], [65], [66], [68]
Dynamic analysis	[16], [20], [23], [28], [33], [38], [45]-[48], [50], [51], [56], [62], [63]

This paper was aimed at showing a simplified and straightforward way to perform a Systematic MS as well as presenting an overview of what security VVT practices are most applied in a context of software development life cycle. Thus, by the study, it was inferred that: (a) the most commonly used practices involve static and dynamic analysis, (b) there is no certainty about the effect of applying them, nor a direction about its advantages and disadvantages.

As a complementary intention of the work, it was presented the execution of a Systematic MS and an example of a research protocol, to disseminate knowledge about revisions, besides facilitate the development of other protocols for different research contexts. Therefore, it is seemed as appropriate to express the following lessons learned from MS:

- Knowledge about the subject to be revised contributes with a fast execution of the process and effectiveness of the result.
- Remains the feeling that there may still be a paper that was not found and could change any conclusion of the review.
- Restrict the scope of the systematic study is recommended, but it can be complex in some situations to be proven.
- It takes time to be done (at least, it should follow: i - search, read, and understand papers / ii - analyze, organize and report results).
- Abstract, Introduction and Conclusion of a paper cannot provide the information needed, requiring reading the entire content.
- Important information to reach a conclusion or find evidence may be omitted in a paper and be only available in more complete works such as theses and dissertations.

It is important to point that whether the question to be proven in the MS would focus on identifying the institutionalization of software security VVT actions in an iterative and incremental development cycle, contextualizing process improvement, there would be no paper attending this scenario.

Note that security verification, validation, and test represent just one important piece in the complex software security mechanism. One should persevere to institutionalize and integrate information security actions in all stages of the development process, beginning at initiation, and, specially, keeping objectives and requirements of the project aligned. In this context, practices of security VVT would also be applied from the beginning of the life cycle and act as a validation tool of security and quality of software products.

Due to time constraints further analysis of the results would be expected as a future work, such as interpreting and detailing in which way static and dynamic analysis influence or contribute with the results of the research project.

REFERENCES

[1] Gary McGraw, “Software security”, IEEE Security and Privacy, March/April 2004, pages 32-35.
[2] CERT.BR. Available: http://www.cert.br/stats/incidentes/.
[3] NIST (2010) Special Publication 800-53A, Revision 1, 2010 - Guide for Assessing the Security Controls in Federal Information Systems and Organizations - Building Effective Security Assessment Plans.
[4] Sommerville, I. (2010), Software Engineering, Addison Wesley, 9th edition.
[5] Kitchenham, B., Brereton P., Budgen, D., Turner M., Bailey J., Linkman, S. (2009) “Systematic literature reviews in software engineering - A systematic literature review”. Information and Software Technology Journal. Vol. 51. Issue 1. Pages 7 - 15. Elsevier. January 2009.

[6] Kitchenham, B. and Charters, S. (2007) “Guidelines for performing Systematic Literature Reviews in Software Engineering”. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.
[7] Mafra, S., Barcelos, R., Travassos, G. (2006) “Aplicando uma Metodologia Baseada em Evidência na Definição de Novas Tecnologias de Software”, v. 1, pages 239 – 254.
[8] Kitchenham et al. (2010) “Systematic literature reviews in software engineering – A tertiary study”. Information and Software Technology 52 (2010) 792–805. Elsevier.
[9] R. Wieringa, N.A.M. Maiden, N.R. Mead, C. Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Engineering, 11 (1) (2006), pp. 102–107.
[10] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008, pp. 71–80.
[11] Budgen, D., Turner, M., Brereton, P., Kitchenham, B. (2008) “Using Mapping Studies in Software Engineering”. Available: https://community.dur.ac.uk/ebse/biblio.php?id=86.
[12] Petticrew, Mark and Roberts, Helen. Systematic Reviews in the Social Sciences: A Practical Guide, Blackwell Publishing, 2005, ISBN 1405121106.
[13] RUP. (2013) IBM - Rational Unified Process ®. IBM Corporation. Copyright © 1987 – 2013.
[14] UBC. (2014) Snowballing technique. Available: http://hlwiki.slais.ubc.ca/index.php/Snowballing.
[15] Marback, Aaron, Do, Hyunsook, He, Ke, Kondamarri, Samuel and Xu, Dianxiang (2013) "A threat model-based approach to security testing". Software: Practice and Experience, v. 43, n. 2, p. 241-258, 2013.
[16] Gilliam, David P. et al. (2006) “Security verification techniques applied to patchlink COTS software”. In: Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE'06. 15th IEEE International Workshops on. IEEE, 2006. p. 319-325.
[17] Shahmehri, Nahid et al. (2012) “An advanced approach for modeling and detecting software vulnerabilities”. Information and Software Technology, v. 54, n. 9, p. 997-1013, 2012.
[18] Austin, Andrew; Holmgreen, Casper; Williams, Laurie. (2013) “A comparison of the efficiency and effectiveness of vulnerability discovery techniques”. Information and Software Technology, v. 55, n. 7, p. 1279-1288, 2013.
[19] Mouratidis, Haralambos; Giorgini, Paolo. (2007) “Security Attack Testing (SAT)—testing the security of information systems at design time”. Information systems, v. 32, n. 8, p. 1166-1183, 2007.
[20] Jürjens, Jan. (2208) “Model-based security testing using umlsec: A case study”. Electronic Notes in Theoretical Computer Science, v. 220, n. 1, p. 93-104, 2008.
[21] Xu, Dianxiang et al. (2012) “A model-based approach to automated testing of access control policies”. In: Proceedings of the 17th ACM symposium on Access Control Models and Technologies. ACM, 2012. p. 209-218.
[22] Wei, Tian et al. (2012) “Attack model based penetration test for SQL injection vulnerability”. In: Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual. IEEE, 2012. p. 589-594.
[23] Antunes, Nuno; Vieira, Marco. (2209) “Detecting SQL injection vulnerabilities in web services”. In: Dependable Computing, 2009. LADC'09. Fourth Latin-American Symposium on. IEEE, 2009. p. 17-24.
[24] Wassermann, Gary; Su, Zhendong. (2007) “Sound and precise analysis of web applications for injection vulnerabilities”. In: ACM Sigplan Notices. ACM, 2007. p. 32-41.
[25] Ciampa, Angelo; Visaggio, Corrado Aaron; Di Penta, Massimiliano. (2010) “A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications”. In: Proceedings of the 2010 ICSE Workshop on Soft. Eng. for Secure Systems. ACM, 2010. p. 43-49.
[26] Shaffer, Alan B. et al. (2008) “A security domain model to assess software for exploitable covert channels”. In: Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security. ACM, 2008. p. 45-56.
[27] Morais, Anderson; Cavalli, Ana; Martins, Eliane. (2011) “A model-based attack injection approach for security validation”. In: Proceedings of the 4th international conference on Security of information and networks. ACM, 2011. p. 103-110.
[28] Wang, Linzhang; Wong, Eric; Xu, Dianxiang. (2007) “A threat model driven approach for security testing”. In: Proceedings of the Third International Workshop on Software Engineering for Secure Systems. IEEE Computer Society, 2007. p. 10.
[29] Al-Azzani, Sarah; Bahsoon, Rami. (2010) “Using implied scenarios in security testing”. In: Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems. ACM, 2010. p. 15-21.
[30] Xu, Dianxiang. (2013) “Software security testing of an online banking system: a unique research experience for undergraduates and computer teachers”. In: Proceeding of the 44th ACM technical symposium on Computer science education. ACM, 2013. p. 705-710.
[31] Avancini, Andrea. (2012) “Security testing of web applications: A research plan”. In:Proceedings of the 2012 International Conference on Software Engineering. IEEE Press, 2012. p. 1491-1494.
[32] Huang, Song et al. (2010) “A Case Study of Software Security Test Based On Defects Threat Tree Modeling”. In: Multimedia Information Networking and Security (MINES), 2010 International Conference on. IEEE, 2010. p. 362-365.
[33] Smith, Ben; Williams, Laurie. (2012) “On the Effective Use of Security Test Patterns”. In: Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on. IEEE, 2012. p. 108-117.
[34] Gilliam, David P. et al. (2001) “Reducing software security risk through an integrated approach”. In: Software Engineering Workshop, 2001. Proceedings. 26th Annual NASA Goddard. IEEE, 2001. p. 36-42.
[35] Xu, Dianxiang et al. (2012) “Automated security test generation with formal threat models”. Dependable and Secure Computing, IEEE Transactions on, v. 9, n. 4, p. 526-540, 2012.
[36] Du, Wenliang; Mathur, Aditya P. (2002) “Testing for software vulnerability using environment perturbation”. Quality and Reliability Engineering International, v. 18, n. 3, p. 261-272, 2002.
[37] Murthy, K. Krishna; Thakkar, Kalpesh R.; Laxminarayan, Shirsh. (2009) “Leveraging Risk Based Testing in Enterprise Systems Security Validation”. In:Emerging Network Intelligence, 2009 First International Conference on. IEEE, 2009. p. 111-116.
[38] Smith, Ben. (2011) “Systematizing security test case planning using functional requirements phrases”. In: Proceedings of the 33rd International Conference on Software Engineering. ACM, 2011. p. 1136-1137.
[39] Xiong, Pulei; Peyton, Liam. (2010) “A model-driven penetration test framework for Web applications”. In: Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on. IEEE, 2010. p. 173-180.
[40] Ouchani, Samir; Jarraya, Yosr; Mohamed, Otmane Aït. (2011) “Model-based systems security quantification”. In: Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on. IEEE, 2011. p. 142-149.
[41] Fonseca, José; Vieira, Marco; Madeira, Henrique. (2013) “Evaluation of Web Security Mechanisms using Vulnerability and Attack Injection”. Dependable and Secure Computing, IEEE Transactions on, v. PP, Issue 99, p. 1, 2013.
[42] Carlsson, Bengt; Baca, Dejan. (2005) “Software security analysis-execution phase audit”. In: Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on. IEEE, 2005. p. 240-247.
[43] Ghindici, Dorina et al. (2006) “Integrated security verification and validation: Case study”. In: Local Computer Networks, Proceedings 2006 31st IEEE Conference on. IEEE, 2006. p. 1000-1007.
[44] He, Ke; Feng, Zhiyong; Li, Xiaohong. (2008) “An attack scenario based approach for software security testing at design stage”. In: Computer Science and Computational Technology, 2008. ISCSCT'08. International Symposium on. IEEE, 2008. p. 782-787.
[45] Savola, R. M. (2009) “Software security assurance of telecommunication systems”. In: Multimedia Computing and Systems, 2009. ICMCS '09. International Conference on Multimedia Computing and Systems.
[46] Mallouli, Wissam et al. (2008) “Modeling and Testing Secure Web-Based Systems: Application to an Industrial Case Study”. In: Signal Image Technology and Internet Based Systems, 2008. SITIS'08. IEEE International Conference on. IEEE, 2008. p. 128-136.
[47] Turpe, S. et al. (2008) “Supporting Security Testers in Discovering Injection Flaws”. In: Practice and Research Techniques, 2008. TAIC PART'08. Testing: Academic & Industrial Conference. IEEE, 2008. p. 64-68.
[48] Tappenden, Andrew et al. (2005) “Agile security testing of web-based systems via httpunit”. In: Agile Conference, 2005. Proceedings. IEEE, 2005. p. 29-38.

[49] Bessayah, Fayçal; Cavalli, Ana; Martins, Eliane. (2009) "A formal approach for specification and verification of fault injection process". In: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human. ACM, 2009. p. 883-890.
[50] Berbar, Ahmed; Ahmednacer, Mohamed. (2009) "Testing and fault tolerance approach for distributed software systems using nematode worms". In:Proceedings of the 4th International Conference on Queueing Theory and Network Applications. ACM, 2009. p. 7.
[51] Zech, Philipp et al. (2013) "A Concept for Language-Oriented Security Testing". In:Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on. IEEE, 2013. p. 53-62.
[52] Katkalov, Kuzman et al. (2012) "Model-Driven Testing of Security Protocols with SecureMDD". In: New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on. IEEE, 2012. p. 1-5.
[53] Hui, Zhanwei et al. (2010) "Software security testing based on typical SSD: A case study". In: Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on. IEEE, 2010. p. V2-312-V2-316.
[54] Jinhua, Li; Jing, Li. (2010) "Model Checking Security Vulnerabilities in Software Design". In: Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on. IEEE, 2010. p. 1-4.
[55] Bodeau, D. J.; Brusil, N. R.; Chang, I. N.; Reece, M. J. (1992) "Security test and evaluation for multilevel-mode accreditation: Lessons learned". In: Proceedings of eighth Annual Computer Security Applications Conference, 1992. p. 37-45.
[56] Wang, Wenhua et al. (2011) "A combinatorial approach to detecting buffer overflow vulnerabilities". In: Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on. IEEE, 2011. p. 269-278.
[57] Wang, Weiguang; Zeng, Qingkai; Mathur, Aditya P. (2012) "A Security Assurance Framework Combining Formal Verification and Security Functional Testing". In: Quality Software (QSIC), 2012 12th International Conference on. IEEE, 2012. p. 136-139.
[58] Schanes, Christian et al. (2013) "Generic Approach for Security Error Detection Based on Learned System Behavior Models for Automated Security Tests". In:Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on. IEEE, 2013. p. 453-460.
[59] Belblidia, Nadia et al. (2006) "AOP extension for security testing of programs". In:Electrical and Computer Engineering, 2006. CCECE'06. Canadian Conference on. IEEE, 2006. p. 647-650.
[60] Mouelhi, Tejeddine; Le Traon, Yves; Baudry, Benoit. (2007) "Mutation analysis for security tests qualification". In: Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION, 2007. TAICPART-MUTATION 2007. IEEE, 2007. p. 233-242.
[61] Hwang, JeeHyun et al. (2012) "Selection of regression system tests for security policy evolution". In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ACM, 2012. p. 266-269.
[62] Lebeau, Franck et al. (2013) "Model-Based Vulnerability Testing for Web Applications". In: Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on. IEEE, 2013. p. 445-452.
[63] Huang, Yao-Wen et al. (2004) "Securing web application code by static analysis and runtime protection". In: Proceedings of the 13th international conference on World Wide Web. ACM, 2004. p. 40-52.
[64] Li, Li et al. (2013) "The Application of Fuzzing in Web Software Security Vulnerabilities Test". In: Information Technology and Applications (ITA), 2013 International Conference on. IEEE, 2013. p. 130-133.
[65] Fourneret, Elizabeta et al. (2011) "Model-based security verification and testing for smart-cards". In: Availability, Reliability and Security (ARES), 2011 Sixth International Conference on. IEEE, 2011. p. 272-279.
[66] Jing-Nong, Du; Yan-Sheng, Lu. (2010) "An Effect Evaluation Model for Vulnerability Testing of Web Application". In: Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on. IEEE, 2010. p. 382-385.
[67] Ma, Jianli et al. (2010) "Information system security function validating using model checking". In: Computer Engineering and Technology (ICCET), 2010 2nd International Conference on. IEEE, 2010. p. V1-517-V1-521.
[68] Salas, Percy Antonio Pari; Krishnan, Padmanabhan; Ross, Kelvin J. (2007) "Model-based security vulnerability testing". In: Software Engineering Conference, 2007. ASWEC 2007. 18th Australian. IEEE, 2007. p. 284-296.
[69] Zhang, Xiao-Song; Shao, Lin; Zheng, Jiong. (2008) "A novel method of software vulnerability detection based on fuzzing technique". In: Apperceiving Computing and Intelligence Analysis, 2008. ICACIA 2008. Intl. Conf. on. IEEE, 2008. p. 270-273.
[70] Blackburn, Mark et al. (2001) "Model-based approach to security test automation". In: Proceedings of Quality Week 2001.
[71] Gupta, Daya; Chatterjee, Kakali; Jaiswal, Shruti. (2013) "A Framework for Security Testing". In: Computational Science and Its Applications–ICCSA, 2013. Springer Berlin Heidelberg, 2013. p. 187-198.
[72] Ouedraogo, Moussa et al. (2012) "Appraisal and reporting of security assurance at operational systems level". In: Journal of Systems and Software, v. 85, n. 1, 2012, p. 193-208.