Web Applications Security Testing Evaluation

Aristides Dasso, Ana Funes

Universidad Nacional de San Luis, Ejército de los Andes 950, D5700 BPB
San Luis, Argentina
{arisdas, afunes}@unsl.edu.ar

Abstract. An important part of a good security software development program is the ability to determine how things are going. Therefore, it is important to track the results of testing and also apply metrics to this aim. A model to help in evaluating security testing in web applications is presented in this work. This model is based on the OWASP Web Security Testing Guide (WSTG) and the Logic Score of Preference (LSP) method. Using the LSP method we are able to construct a model that can be of help in assessing compliance respect to the tenets of the Check List of the WSTG, during and after testing in an application development project. Since LSP is a multicriteria and multiattribute decision method that allows the creation of models that can give different relevance to the various features under evaluation, in this case –items in the WSTG Check List– it can help in deciding which item or items should be given more importance according to the needs of the project considered.

Keywords. multicriteria decision methods, LSP method, software security

1 Introduction

Once an organisation has chosen a development software process where security methodologies are applied, the demand for having a model that make possible the evaluation of the security methodology applied within the process is mandatory. Therefore, this model should allow for getting the knowledge and the control about the degree of implementation of the considered security directives, tools employed, the reliability build in the final product as well as the advance in the implementation of the security measures during development.

To construct a model for the evaluation of complex systems, among them security systems, is a must for organisations, and it is not a simple task. Several features must be considered having in mind not only physical facets, such as installations, and its access policies but also software security rules, firewalls, permissions, coding, etc,

Therefore, any institution worried about its computing installation security, especially those on the web, needs to have standards as well as tools ready to evaluate how well those standards are implemented.

We propose here the adoption of a model to assess how much a set of rules for the implementation of security measures during web application development are observed. The model is based on the Web Security Testing Guide (WSTG) of the Open Web Application Security Project (OWASP) [17], particularly on its Check List[1], whose tests are considered as the main features to be evaluated by the model implemented by the Logic Scoring of Preference (LSP) method [13].

Different ways of confronting the issue of security evaluation can be found in the literature and on the web. There is an interesting review of the literature on the subject in [2]. For instance, the authors in [6] use an interactive system based on graphs that was developed on the context of secure coding standards to handle vulnerabilities. With the help of this graph, they expect to overcome the possible human errors when applying those standards.

In The Open Source Security Testing Methodology (OSSTMM), Khairul Anwar Sedek et al. [15] adopt the Top Ten Critical Vulnerabilities defined by OWASP to create an additive model to evaluate the performance based on the OSSTMM.

Jun Zhu et al. [14], employing an interactive static analysis within IDEs (Integrated Development Environment), provide in situ secure programming so system developers can prevent vulnerabilities during code development.

Sajjad Rafique et al. [21] explore the diverse security vulnerabilities so as to ensure the security of the web application layer, the approaches and techniques used in the development process, as well as the phases in the software development where those techniques are employed together with the tools and mechanisms employed to detect vulnerabilities.

In helping to reduce the risks associated in the development of web applications in a given environment, S. Vargas et al. [20] developed a strategy based on the analysis of the security policies used in similar institutions. They also analyzed the rules and regulations and the state of the art in the security of web applications.

As it was said before, many other proposals on the subject can be found in the web, e.g. see [3], [11], [22].

The rest of the work is organised as follows. Section 2 gives a brief description of the OWASP WSTG, used for the construction of the proposed model. Section 3 describes the method adopted for the development of the security testing assessment model. Section 4 shows parts of a possible model and Section 5 closes the work with some discussion and conclusions.

2. Web Security Testing Guide

The Open Web Application Security Project (OWASP) is a non profit foundation that centres its activity in promoting the security of software in all its applications. It does so by developing Tools and Resources, providing Community and Networking,

[1] https://github.com/OWASP/wstg/blob/master/checklist/Testing_Checklist.md

and Education and Training. Lots of enterprises follow their recommendations, which cover big areas of software development, applying them to different areas of software industry, and use as well their support tools, manuals, etc. The OWASP Web Security Testing Guide (WSTG) provides a reliable cybersecurity testing resource for web application developers and security professionals. This testing guide shows how to verify the security of running –or in production– software applications, making it a highly recommend guide for application of security initiatives.

The WSTG should be used by different actors involved in a software development project. Not only developers to assure that they are producing secure code, or testers and SQA team members to enhance their set of testing cases, but also by security specialists and project managers.

In the introduction of the WSTG there is a quote by DeMarco: "You can't control what you can't measure." [5], and it is also said there that "Security testing is no different. Unfortunately, measuring security is a notoriously difficult process." [16]. The purpose guiding us is to provide a model to assess the implementation of the tenets implied in the Check List of the WSTG coupled with the LSP method so as to be able to measure the degree of compliance of the Check List during the development of the project.

Finally, it is important to note that the WSTG comprises a set of techniques that can be used to find different types of security flaws, nevertheless not all these techniques are equally important. The information must be tailored according to the organization's technologies, processes, and organizational structure. Therefore, the same observation applies also to the LSP assessment model to be created.

3. The LSP Method

In the development of a model for the compliance evaluation of the OWASP Web Security Testing Guide (WSTG) Check List[1], we propose the adoption of the Logic Scoring of Preference (LSP) method [13]. This model should be of help in assessing the observance to the main guidelines of the WSTG, and also those embodied into its Check List.

LSP is a method for the realization of complex criterion functions and their application in the evaluation, optimization, comparison and selection of general complex systems. As its creator establishes "The methodology draws on work in soft computing, fuzzy systems, multicriteria and multi-attribute decision making." [13, pp xvii]. It has been used in several domains to evaluate and measure diverse systems in wide areas of different industries –not only in the computing industry– as can be seen in [1], [4], [7], [10], [9], [8], [12].

Since it is a general evaluation method, it can be applied to different domains, in particular, for the evaluation processes involved in the implementation of a security system, in this case to assess the compliance of items in the WSTG Check List by personnel in charge of the web security implementation. As J. Dujmovic expresses in [13, pp 368], it is not easy to find multicriteria decision problems where all individual

criteria have exactly the same importance; therefore, the LSP method is especially useful when complex and/or decisions are present in the assessment, and simple additive scoring methods are not enough.

As with many other methods, the first steps in LSP involve to make clear what the user requirements are. Requirements come in the form of a set of elementary attributes, which contains all quantifiable inputs that affect the overall suitability of the evaluated object. They must be clustered into cohesive categories and subcategories, all of them organized into a tree; the resulting tree is referred as the Requirement Tree (RT). The leaves of the RT are the attributes of the system to be measured. These attributes are called performance variables. As the schema in Fig. 1 shows, each one of these performance variables is mapped into an elementary preference by applying, during evaluation, the corresponding elementary criterion, each of which has been previously defined during model development.

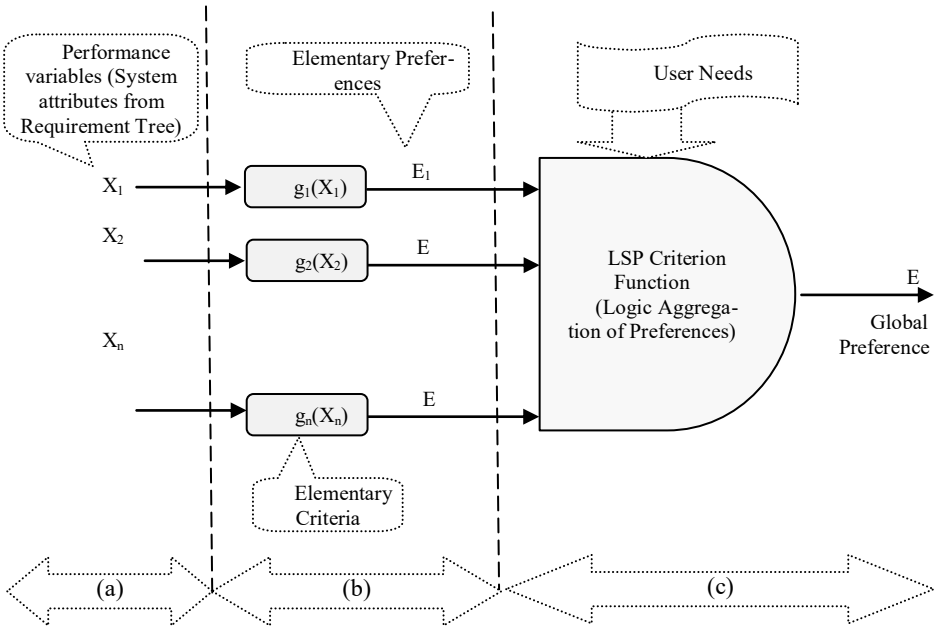

Fig. 1 An overview of the LSP method.

Fig. 1 presents three different parts –denoted as (a), (b) and (c). They show the main artifacts that form part of a LSP model; they are created during the phases prescribed by the LSP method and used for the evaluation of a given system. Part (a) shows the performance variables, which corresponds to the identified system´s attributes and are the leaves of the RT. During evaluation, they are instantiated with the observed values of the system under evaluation.

Part (b) of Fig. 1 shows the elementary criteria and the corresponding elementary preferences. An elementary criterion is a function that maps a value assigned to a

performance variable of the system into a value in the closed interval [0..100], namely a performance variable value into an elementary preference. Each elementary preference represents the degree of fulfilment of a given requirement (performance variable) in the requirement tree, where 0 means that the requirement has not been fulfilled at all and 100 that it has been completely satisfied.

These elementary preferences, obtained from the transformation of each performance variable via its corresponding elementary criteria, are aggregated under the form of an Aggregation Structure or LSP Criterion Function, as it is shown in the part (c) of Fig. 1. The process of aggregation is the most difficult method's phase and requires of several attempts until the final aggregation structure is properly calibrated. It is built by aggregating elementary preferences by means of a set of operators. Aggregating preferences means to replace a group of input preferences by a single output preference, which denotes the degree of satisfaction of the evaluator with respect to the whole group of input preferences. The output preferences must be aggregated again until a single global preference can be obtained. Therefore, the LSP Criterion Function is the result of the combination of preferences, taking into account both the relative importance of each preference and the necessary logic relationship between them. During evaluation, the LSP Criterion Function yields a single global indicator of the degree of fulfilment of the whole system requirements, as well as a set of partial indicators (one for each of the aggregated categories).

The most complex phase in the whole modelling process is the calibration of the LSP Criterion Function. In this phase, it is necessary to pay special attention not only to the needs of end users but also to the relationship between the aggregated preferences. Once the calibration of the LSP Criterion Function has finished, the LSP model is ready and the evaluation can proceed; in this case, the assessment process for the compliance of the selected items in the WSTG Check List can start. It means to collect the values corresponding to each of the performance variables in the RT and provide them as input to the elementary criteria and then to the LSP Criterion Function, to finally obtain, not only a global performance indicator of the compliance degree with respect to all the selected items in the WSTG Check List, but also a set of partial indicators for each of the identified categories and subcategories in the RT. These partial indicators are also very important since they show clearly which of the identified categories and subcategories present security flaws.

To aggregate n elementary preferences $E_1, \ldots, E_n$ in a single preference E_0, the resulting preference E_0 –interpreted as the percentage of satisfaction of the n requirements– is expressed by a function having the following properties:

$$min(E_1, \ldots, E_n) \leq E \leq max(E_1, \ldots, E_n) . \quad (1)$$

The relative importance of each elementary preference E_i ($i = 1...n$) is expressed by a weight Wi ,

A set of Graded Conjuction Disjuction (GCD) functions can be obtained from the instantiation of the weighted power means:

$$E(r) = (W_1 E_1^r + W_2 E_2^r + \ldots + W_n E_n^r)^{1/r} , \quad (2)$$

where

$$0 < W_i < 100,\ 0 \le E_i \le 100,\ i = 1, \ldots, n,\ W_1 + \ldots + W_n = 1,\ -\infty \le r \le +\infty \qquad (3)$$

The choice of r determines the location of $E(r)$ between the minimum value $E_{min} = min(E_1,...,E_n)$ and the maximum value $E_{max} = max(E_1, \ldots, E_n)$. For $r = -\infty$ the weighted power mean reduces to the pure conjunction (the minimum function) and for $r = +\infty$ to the pure disjunction (the maximum function), giving place to a Continuous Preference Logic (CPL). The range between pure conjunction and pure disjunction is usually covered by a sequence of equidistantly located CPL operators named: C, C++, C+, C+–, CA, C–+, C–, C– –, A, D– –, D–, D–+, DA, D+–, D+, D++, D, which are the Graded Conjuction Disjuction (GCD) functions. For a more detailed description of the technique see [13].

The weights (W_i) associated to each elementary preference are assigned by the user according to the importance that each elementary preference has in the model being constructed. The same goes when choosing the different CPL operators.

All the produced artifacts that are part of the final LSP model are further explained in section 4; all of them are illustrated through examples, where several aggregation structures built using the first level categories of the WSTG Check List and also subcategories of one of the first level categories are shown; an example of the definition of an elementary criteria is also given and the RT is partially presented.

4 A Model for the Compliance Assessment of the WSTG Check List

In this section, parts of the model to gauge adherence to the WSTG Check List is presented. Examples of different portions of the model are shown so as to give the general idea of the model.

The following subsections are presented in the same order as the different model artifacts are developed. As prescribed by the LSP method, we start with the creation of the Requirement Tree, portion of which is given in subsection 4.1. It follows a subsection that shows an example of one of the elementary preferences defined for one of the performance variables of the RT, and finally, in subsection 4.3, some parts of the final aggregation structure are also shown.

4.1 Requirement Tree

In first place, the relevant attributes for evaluating web application security testing were organized on a hierarchical structure, namely the Requirement Tree (RT). The attributes in this case, have been taken from the items in the WSTG Check List, which are conveniently organized in a hierarchical way too. For this model we can adopt as RT the complete WSTG Check List or only those categories and items that are con-

sidered relevant for the particular case. The first level of the RT shown in Table 1 contains the eleven different categories present in the WSTG Check List..

Each first level item involves a set of sub categories or sub items; e.g. item 7 “Input Validation Testing” in Table 1 is shown expanded in Fig. 2. It must be remarked that item 7 has eighteen sub items as it can be seen in Fig. 2. These sub items are labelled in the WSTG Check List as WSTG-INPV-01 to WSTG-INPV-18. Also it is important to mention that each of those sub items is considered to have their own sub items, a set of related tests (and the tools to be employed in each case). For details on the WSTG Check List see the OWASP Web Security Testing Guide (WSTG) [17].

Table 1. The first level of the WSTG Check List.

Groups	**Items**	
I	1	Information Gathering (INFO)
	2	Configuration and Deploy Management Testing (CONF)
II	3	Identity Management Testing (IDNT)
	4	Authentication Testing (ATHN)
	5	Authorization Testing (ATHZ)
	6	Session Management Testing (SESS)
III	7	Input Validation Testing (INPV)
	8	Testing for Error Handling (ERRH)
	9	Testing for Weak Cryptography (CRYP)
	10	Business Logic Testing (BUSL)
	11	Client Side Testing (CLNT)

The last level items in the WSTG Check List are the leaves of the RT shown, where each corresponds to a performance variable. Note that leaves can be found at different levels in the tree.

Given the magnitude of the resulting RT, we show in Fig. 2 just part of it, where only two of its branches are shown expanded (coloured in grey). They have been chosen to illustrate the concepts in the two next subsections. The full RT can be obtained from the OWASP Web Security Testing Guide (WSTG) [17].

1. Information Gathering (INFO)
2. Configuration and Deploy Management Testing (CONF)
3. Identity Management Testing (IDNT)
4. Authentication Testing (ATHN)

5.	Authorization Testing (ATHZ)		
6.	Session Management Testing (SESS)	6.1. Testing for Bypassing Session Management Schema	6.1.1. SessionID analysis prediction 6.1.2. Unencrypted cookie transport 6.1.3. Brute-force
		6.2. Testing for Cookies attributes 6.3. Testing for Session Fixation 6.4. Testing for Exposed Session Variables 6.5. Testing for Cross Site Request Forgery 6.6. Testing for logout functionality 6.7. Test Session Timeout 6.8. Testing for Session puzzling	
7.	Input Validation Testing (INPV)	7.1 Testing for Reflected Cross Site Scripting 7.2. Testing for Stored Cross Site Scripting 7.3. Testing for HTTP Verb Tampering 7.4. Testing for HTTP Parameter Pollution 7.5. Testing for SQL Injection 7.6. Testing for LDAP Injection 7.7. Testing for XML Injection 7.8. Testing for SSI Injection 7.9, Testing for XPath Injection 7.10. Testing for IMAP SMTP Injection 7.11. Testing for Code Injection 7.12. Testing for Command Injection 7.13. Testing for Buffer Overflow 7.14. Testing for Incubated Vulnerability 7.15. Testing for HTTP Splitting Smuggling 7.16. Testing for HTTP Incoming Requests 7.17. Testing for Host Header Injection 7.18. Testing for Server Side Template Injection	
8.	Testing for Error Handling (ERRH)		

9. Testing for Weak Cryptography (CRYP)
10. Business Logic Testing (BUSL)
11. Client Side Testing (CLNT)

Fig. 2. The Requirement Tree (RT).

4.2 Elementary Criteria for the Evaluation of the Performance Variables

The purpose of an elementary criterion is to transform a performance variable X_i to an elementary preference e_i on a normalized scale of [0..100]. Each e_i expresses the degree of satisfaction or fulfilment of the parameter X_i. Therefore, the metrics that allow obtaining the value for each attribute of the requirements tree must be defined.

These performance variable values can be obtained by different methods according to the organization rules and the kind of attribute to be measured or calculated. They can be assigned by a team, by only one person, by an algorithm, etc. A measurement process allows assigning values to the performance variables in a direct way, while a calculation process using a metric allows also assigning values although in an indirect way. Therefore, some performance variables would not have the need of an elementary criteria definition, namely when the attribute values can be obtained from a direct measurement in the range [0..100].

For example, considering that SQL Injection is recognized as one of the main ten major errors [18] we have chosen to show in Table 2 the definition of an elementary criterion for the attribute "Testing for SQL Injection" of the "Input Validation Testing" category in the RT. As it can be observed, this elementary criterion has been defined as the percentage of input fields free of SQL injections, penalizing the presence of SQL injections.

Table 2. Example of definition of an elementary criterion

Elementary criterion %SQL: Percentage of passed test of SQL injection detection
Type of metric: Indirect Objective: Determine the percentage of input fields free of SQL injections. The tester has to make a list of all input fields whose values could be used in crafting a SQL query, including the hidden fields of POST requests and then test them separately, trying to interfere with the query and to generate an error. In any case, it is very important to test each field separately: only one variable must vary while all the other remain constant, in order to precisely understand which parameters are vulnerable and which are not.

Related metrics:

1) $\#if$: is the number of identified input fields, considering also HTTP headers and cookies.
2) $\#SQLif$: is the number of input field with detected SQL injections.

Calculation method:

$$\#if > 0 \rightarrow \%SQL = 100 - \frac{\#SQLif}{\#if} * 100$$

$$\#if = 0 \rightarrow \%SQL = 100$$

4.3 The LSP Aggregation Structure from the WSTG Check List

A possible aggregation structure considering the eleven top level categories given in Table 1 is shown in Fig. 3. It has been created by aggregating three groups of the top level categories in the following way: group I includes top level categories 1 and 2, in group II categories 3 to 6 and group III categories7 to 11. Although cohesion of each grouping has been considered, another reason for this is that –in practice– most of the tools supporting the LSP method accept up to five parameters as input to the CPL operators provided by the method;. Of course, it should be taken into consideration whether another grouping might be more convenient for a given organization and so that could also be done. This flexibility is one of the main advantages of the method, allowing to calibrate the model according to the actual organization needs.

In this top level aggregation structure, the three groups are given a nearly equal weight, only Group II has a slighter greater weight, since it has been considered to be a little more important than the other two. The C+ CPL operator has been chosen to aggregate the three groups of items. C+ operator corresponds to a strong conjunction, meaning that a high degree of simultaneity is expected in the mandatory aggregated items. If there were present at least one completely unsatisfied input then the entire criterion (E_0) would be completely unsatisfied; therefore, all inputs to E_0 must be at least partially satisfied, otherwise the whole preference goes to zero. Also, let us note that when this operator is applied, the presence of low input values affects the output much more than the presence of high input values.

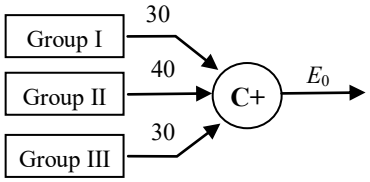

Fig. 3. Aggregation structure for three identified groups in the top level categories shown in Table 1.

Fig. 4 shows a portion of the LSP criterion function that corresponds to the aggregation of the items that are part of the Group III in Table 1. Rather strong CPL opera-

tors have been also used here to aggregate these items. The reason is that all have been considered important and should be fulfilled during evaluation, i.e. they are mandatory items. However, a note should be done at this point. It could easily being said that all the items in the Check List are important, i.e. evenly important, all of them mandatory. Although the shown model would seem of little use, given that it is highly conjunctive and returns a global output E_0 equal to zero when any of its inputs is not satisfied, it should not be forgotten that, in addition to the global indicator E_0, the model provides with a set of partial indicators, which allow to quickly identify those sub items where the implementation of the security of the system is failing, enabling to focus on those areas to correct these faults in next stages of debugging and testing.

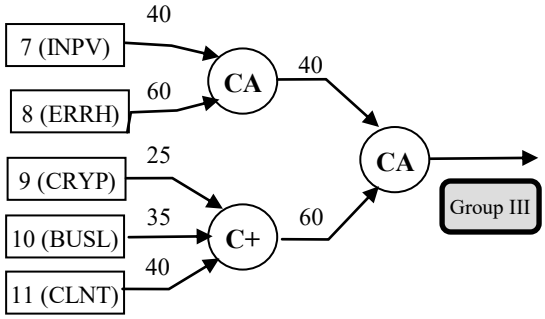

Fig. 4. Aggregation structure for the items in Group III of Table 1.

5 Discussion and Concluding Remarks

During system development, gauging that the security measures specified are correctly implemented, particularly those having to do with testing, is a difficult task, so the existence of an assessment model should be of help in determining how much and how well they are being fulfilled. We have shown here how a model to measure the level of compliance with respect to the WSTG Check List can be developed. As it has been shown through examples, the model can be built up and calibrated using a method based on a graded logic, the LSP method, which is a very flexible method that can be adjusted to the needs and requirements of the end users, and powerful enough to solve not only multicriteria problems but also problems where and/or decision are necessary and a simple additive method is not sufficient.

The proposed model is flexible enough to be updated to specific needs and requirements by adapting its artefacts, making them as comprehensive as needed; so, different categories of or even the full WSTG Check List could be taken into consideration as required, according to the organizational structure, its technologies and processes involved. As a result, the WSTG Check List can be extremely useful as a roadmap during testing since it gives a comprehensible list of the steps to be taken, and it has embedded an extensive knowledge about security flaws in applications. In addition, the integration of the Check List with the LSP method can give together a more precise tool to aid in the evaluation of the different stages of security testing,

allowing to control the advances in the discovery of faults during the software development project since the various sub structures of the complete model permit to gauge the level of compliance with different categories of security testing.

As an ending note, it must be noted that the application of the model to diverse environments –where testing for security purposes is necessary– is an open area for improvement.

References

1. Bai, H. Evaluation and comparison of search engines using the LSP Method. San Francisco State University, Computer Science Dept., Report SFSU-CS-CE-07.12, 2007.
2. Bala Musa Shuaibu, Norita Md Norwawi; Mohd Hasan Selamat; Abdulkareem Al-Alwani. "Systematic review of web application security development model". Artificial Intelligence Review February 2015 https://doi.org/10.1007/s10462-012-9375-6
3. Cody Arsenault- 11 Web Application Security Best Practices. Updated on March 4, 2019. https://www.keycdn.com/blog/web-application-security-best-practices
4. Dasso, Aristides y Funes, Ana. "A Model for Choosing the Right ERP System", SII 2013, 42 JAIIO, 16 al 20 de septiembre de 2013, Córdoba, Argentina.
5. DeMarco, Tom. Controlling Software Projects: Management, Measurement, and Estimates. Prentice Hall. June 1986 ISBN-13: 9780131717114. ISBN-10: 0131717111
6. Divya Rishi Sahu & Deepak Singh Tomar. "Analysis of Web Application Code Vulnerabilities using Secure Coding Standards". Computer Engineering and Computer Science. Arabian Journal for Science and Engineering volume 42, pages 885–895 (2017)
7. Dragićević, S., J. Dujmović, R. Minardi. Modeling urban land-use suitability with soft computing: the GIS-LSP method. In: Thill, J-C, Dragićević, S. (Eds.) GeoComputational analysis of regional systems. Springer, 2018, pp. 257–275.
8. Dujmović, J., G. De Tré, and N. Van de Weghe. Suitability maps based on the LSP method. Proceedings of the 5th MDAI conference (Modeling Decisions for Artificial Intelligence), Sabadell (Barcelona), Catalonia, Spain, October 30–31, 2008. Lecture Notes in Computer Science, Vol. 5285, pp. 15–25, Springer-Verlag, Heidelberg, Germany, 2009.
9. Dujmović, J., G. De Tré, and S. Dragićević. Comparison of multicriteria methods for land-use suitability assessment. Proceeding of the 13th IFSA World Congress and the 6th EUSFLAT Conference, July 20–24, 2009, in Lisbon, Portugal, pp. 1404–1409, ISBN: 978-989-95079-6-8, 2009.
10. Dujmović, J., J. W. Ralph, and L. J. Dorfman. Evaluation of disease severity and patient disability using the LSP method. In L. Magdalena, M. Ojeda-Aciego, J. L. Verdegay (Eds.), Proceedings of the 12th Information Processing and Management of Uncertainty international conference (IPMU 2008), pp. 1398–1405, Torremolinos (Malaga), June 2008.
11. guru99. Web Application Testing: 8 Step Guide to Website Testing. https://www.guru99.com/web-application-testing.html
12. Hatch K., Dragićević S., Dujmović J. (2014) Logic Scoring of Preference and Spatial Multicriteria Evaluation for Urban Residential Land Use Analysis. In: Duckham M., Pebesma E., Stewart K., Frank A.U. (eds) Geographic Information Science. GIScience 2014. Lecture Notes in Computer Science, vol 8728. Springer, Cham
13. Jozo Dujmović. "Soft Computing Evaluation Logic. The LSP Decision Method and Its Applications". Wiley, IEEE Press. Hoboken, NJ : John Wiley & Sons, 2018
14. Jun Zhu, Jing Xie, Heather Richter Lipford, Bill Chu; "Supporting secure programming in web applications through interactive static analysis". Cairo University. Journal of Ad-

vanced Research. 2013 Production and hosting by Elsevier B.V. on behalf of Cairo University. http://dx.doi.org/10.1016/j.jare.2013.11.006
15. Khairul Anwar Sedek, Norlis Osman, Mohd Nizam Osman, Hj. Kamaruzaman Jusoff, "Developing a Secure Web Application Using OWASP Guidelines". Vol. 2, No. 4 (2009), Computer and Information Science.
16. OWASP Web Security Testing Guide (WSTG) – Latest version (not stable); Introduction. https://owasp.org/www-project-web-security-testing-guide/latest/2-Introduction/ (Retrieved 5/29/2020
17. OWASP Web Security Testing Guide (WSTG). https://owasp.org/www-project-web-security-testing-guide/
18. OWASP. Top 10 Web Application Security Risks. https://owasp.org/www-project-top-ten/ Retrieved May 5, 2020.
19. Premal B. Nirpal; K. V. Kale, A Brief Overview Of Software Testing Metrics. International Journal on Computer Science and Engineering (IJCSE). ISSN : 0975-3397 Vol. 3 No. 1 Jan 2011
20. S. Vargas, M. Vera and J. Rodriguez. "Security strategy for vulnerabilities prevention in the development of web applications". Published under license by IOP Publishing Ltd. Journal of Physics: Conference Series, Volume 1414, V International Conference Days of Applied Mathematics 15–17 May 2019, Barranquilla, Colombia.
21. Sajjad Rafique, Mamoona Humayun, Zartasha Gul, Ansar Abbas, Hasan Javed. "Systematic Review of Web Application Security Vulnerabilities Detection Methods". Journal of Computer and Communications, 2015, 3, 28-40. Published Online September 2015 in SciRes. http://www.scirp.org/journal/jcc. http://dx.doi.org/10.4236/jcc.2015.39004.
22. Top 30+ Web Application Testing Tools In 2020 (Comprehensive List). https://www.softwaretestinghelp.com/most-popular-web-application-testing-tools/ Last Updated: March 19, 2020.