International Journal of Engineering &Technology, 7 (3.12) (2018) 773 -778

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Analysis of Vulnerability Detection Tool for Web Services

Senthamil Preethi K[1], Murugan A[2]

[2]Assistant Professor (SG),
[1,2]Dept. of Computer Science & Engineering, SRM IST, Chennai, Tamil Nadu, India
**Corresponding Author Email: [1]senthamilpreethi@gmail.com, [2]murugan.abap@gmail.com*

Abstract

The demand of the web services requirement is increasing day by day, because of this the security of the web services was under risk. To prevent from distinct types of attacks the developer needs to select the vulnerability detection tools, since many tools are available in the market the major challenging task for the developer to find the best tool which suitable for his application requirements. The recent study shows that many vulnerability detection tools provide a low coverage as far as vulnerability detection and higher false positive rate. In this paper, proposed a benchmarking method to accessing and comparing the efficiency of vulnerability detection tools in the web service environment. This method was used to illustrate the two benchmarks for SQL injection and cross site scripting. The first one is depending on predefined set of web services and next one permits user to identify the workload (User defined web services). Proposed system used the open source and commercial tools to test the application with benchmarking standards. Result shows that the benchmarks perfectly depict the efficiency of vulnerability detection tools.

Keywords: *Web services; vulnerability identification; Benchmarking;*

1. Introduction

Web services are used to enable the communication among various application to exchange data and incorporate the framework by using open standards such as XML, SOAP, and REST and so forth [1].
In general, security of web application is exceptionally feeble [2]. Often codes are deployed with vulnerability in the web services.
Web services are generally insecure so that any security threat will most possibly be undetected and misused by attacker.
A vulnerability can be defined as a "Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source."
OWASP identify top 10 web application security vulnerabilities [4] are Injection such as SQL injection, LDAP injection, and CRLF injection and Cross-Site Scripting etc.
Injection flaws, for example, SQL injection, LDAP injection, and CRLF injection arise while untrusted information is directed towards the mediator as a component of the command or query. The attacker's aggressive information could trap a translator into run the inadvertent commands or getting the information with lacking legitimate approval.
Cross site scripting flaws happen when the attacker infuse malicious script to the application which contains untrusted information and sent to new browser with improper inspection or else update the existing web page with user specified data [6]. So, it allows attackers to run or modify the contents in the objective's program which can commandeer client sessions, damage websites or divert a user to mischievous activity.

There are three types of cross site scripting: -

- Stored XSS Attacks
- Reflected XSS Attacks
- Dom-Based XSS Attacks

Stored XSS attacks is directly injected a script is permanently stored in the target servers such as database, visitor log and comment field etc.., Later the victim is tried to access the affected webpage in a web browser, in the background the malicious script is executed and retrieve the stored user information [6]. It is also called as persistent or type-I XSS.
Reflected XSS attacks are in the form of email messages, social media and web links which routed to the phishing sites and inject the vulnerable code in the victim browsers, it is also called as non-persistent or type-II XSS.
Dom-Based XSS Attacks, it is based on the client-side attack and web server doesn't know the client is affected by this type of attack.
The attacker can inject the payload which will be stored as a part of the DOM and executed when the data is read from the DOM.
Vulnerability detection tools. To prevent vulnerabilities, the developer has to do the list of actions such as apply coding best practices [7], utilize static code analyzers, perform security code audit and run the penetration tests etc.
Vulnerability detection tools are most normally utilized by the engineer in the web service environments to help mechanized security inspection and contain some of the critical tools for ensure programming improvement. There are distinctive techniques proposed for vulnerabilities detection such as penetration testing and static code analysis. These two methods were broadly utilized.
Because of time points of confinement or asset limitations, designer need to choose the particular tool from the list of tools accessible on the web without knowing the subtle elements of tools [7] whether this tool will distinguish every one of the

vulnerabilities or not. They depend on the tool to distinguish all the security issues on the code which was created by the engineer.

2. Related Works

Vulnerability Detection: Penetration testing and static code analysis both are surely understood methods regularly utilized through designers towards discover security weaknesses in web-services. It is finished utilizing powerful implementation of a program, detection consists in examination of the results, which restrict perceptibility on the interior activities. Static code analysis is depending upon investigation of the source code, which permits characterizing particular code patterns designs inclined to security vulnerabilities. In any case, it does not have a dynamic perspective of the application conduct within the sight of a realistic workload. Distinctive between these two techniques is previous does not expect access to the bytecode and the last need to get to byte code.
Penetration testing tools used to test the application for vulnerabilities by an automatic way with the given input esteems. Past study demonstrates an adequacy of the tools in web-services is extremely pitiable. e.g., in [5] demonstrates a few disadvantages, in particular: the significant contrast in the distinctive tool is low scope (For 2 scanners, there is under 20%), as well as high quantity of false positives. The contemplate displayed in [8], [9] are additionally affirmed the constraints.
Static code analyzers give a viable programmed approach to feature the conceivable coding mistakes without the run the application [8]. In [12] creators broke down 3 tools and contrasted the viability and the code audit. The tools attained higher productivity than the surveys in identifying software defeat (security issues not considered in the investigation) in five modern Java-based applications, however every one of the tool reachable false positive proportions is more noteworthy than 40 percentage. Those issue is additionally affirmed on [8].
Runtime anomaly detection used to find deviances in a chronicled outline of legitimate commands. In [13] author proposed a method that merge pen. testing with anomaly detection to revealing SQL Injection. Additional examination demonstrates an AMNESIA [14], a tool used to identify and stay away from SQL injection attacks and that tool merge both runtime monitoring and static analysis. The drawback for that approach depends on the learning stage with the goal that it's hard to ensure the culmination and it prompt false positives and undetected vulnerabilities.
Benchmarks are standard tools that permit assessing and comparing systems or components according to with particular qualities such as dependability, performance, and so on [3]. The investigation on performance benchmarking has been begun long back. The benchmark range can be easy to extremely complex benchmarks centering complex framework (e.g., DBMS, working system). This benchmark used to enhance next ages of frameworks. More works has been led for dependability benchmarks by various individuals and subsequent diverse methodologies [15]. At long last, inquire about a security benchmark is most recent theme with numerous undeveloped inquiries [16], [17].
A few researches, demonstrate the significance of assessing testing methods utilizing controlled experimentations. In [11] the creators specified the troubles behind making controlled situations.
Further works tried to evaluate the efficiency of vulnerability detection tools (e.x. [9], [12]). Indeed, past works think about dissimilar tools under circumstances, which can't be generalized or effortlessly reproduced, in this manner results about are of restricted utilize.
Another work is introduced in [9]. It suggests a strategy to assess WVS using software fault injection techniques. Programming issues were infused with code, demonstrating its qualities and shortcoming about scope of VD and FP.
HP WebInspect is a tool that "performs web application security testing and assessment for today's complex web applications, built on emerging Web 2.0 technologies. HP WebInspect delivers fast scanning capabilities, broad security assessment coverage and accurate web application security scanning results" [18].
Acunetix Web Vulnerability Scanner "is an automated web application security testing tool that audits a web application by checking for exploitable hacking vulnerabilities" [19].

3. Benchmarking Approach

The proposed system to develop a benchmark for vulnerability detection tools based on measurement-based techniques. The basic concept to working with this type of this tools under the benchmarking using a web services code with and without vulnerabilities and the results shows that the small set of measures represent the tools detection capabilities. A benchmark must be particularly focused to a specific domain [3].

- Metrics. Portray the viability of the tools which identifying the benchmarking in vulnerabilities that continue in the workload services. The metrics needs to be easy to understand and must permit comparison of tools.
- Workload. The method to represents the tool which perform the Benchmark execution and it present the list of services are utilized during the Benchmarking Process. Based on the requirement workload can be predefined or info provided by the consumer.
- Procedure. Interpret the process and guidelines have been extracts the report after executing the benchmark.

3.1 Metrics Computation

The benchmark measurements ought to be processed from the data gathered at the time of benchmark execution and should take after the entrenched measuring the viewpoint commonly utilized as a part of performance benchmarking [3]. Comparative measures that can be utilized for correlation or for development and tuning.
A key trouble identified with the benchmark measurements is dissimilar vulnerability detection tools report vulnerability in various manner. For instance, for each vulnerable input corresponding vulnerabilities are reported in the penetration-testing tools. In static analysis tools for each vulnerable line in the code the vulnerabilities are reported. Our proposition is to portray the vulnerability detection tools utilizing the F-Measure proposed by van Rijsbergen [10]. It can be characterized as

- Precision. The ratio of correctly detected vulnerabilities to the number of all detected vulnerabilities:

$$\text{Precision} = TP / (TP+FP) \quad (1)$$

- Recall. The ratio of correctly detected vulnerabilities to the number of known vulnerabilities:

$$\text{Recall} = (TP/ TV) \quad (2)$$

Where:
- True positives (TP) is the no. of true vulnerabilities detected;
- Fault positives (FP) is the no. of vulnerabilities detected but it doesn't present;
- True vulnerabilities (TV) is the total no. of vulnerabilities could be present in the code.

$$\text{F-Measure} = (2*\text{precision}*\text{recall}) / (\text{precision} + \text{recall}) \quad (3)$$

3.2 Workload

The workload characterizes the work that must be finished by the vulnerability detection tools in the time of benchmark run. It is predominantly influenced by three factors such as type of web services, kinds of vulnerabilities and detection methods.
Two choices are accessible with respect to the meaning of the workload:

- Predefined workload. The benchmark contains a predefined collection of web services with vulnerabilities.

- User-defined workload. The responsibility of the user to select the target collection of services.

3.3 Procedure

1. **Planning**. Planning stage comprising:
a. Workload selection and characterization. Needed while user responsibility to select target collection of services.
b. Tools identification. There are lot of tools are present both in open source and commercials. Determine the tools to be benchmarked.
2. **Execution**. Run the tools for different benchmarking and generate the report for further analysis.
3. **Analysis**. Characterize the tools benchmarked using the report generated in the execution phase.
a. Metrics computation. Analyze the vulnerabilities revealed by the tools and calculate the metrics.
b. Ranking and selection. Prioritize the tools and select the most effective tool (or tools) using the preferred ranking.
Step 1.a isn't required in case of benchmarks depends on a predefined workload. Then Step 1.a is extremely significant for user-defined benchmarks, as it incredibly impacts the benchmark outcomes.

Table 1: Vulnerability Found in the Workload

Benchmark	Service Name	Vuln. Inputs	Vuln. Queries	LOC	Avg.C
tpc-app	New products	14	1	102	4.6
	Product detail	0	0	122	5.6
	Change payment method	1	2	98	5.1
	New customer	2	3	204	5.7
tpc-w	Create new customer	1	1	44	3
	Creates hopping cart	1	1	208	2.68
	Do authors each	0	0	162	2.9
	Admin update	10	4	85	5.2
	Get customer	2	1	48	4.1
	Get password	1	2	41	2
	Do subjects each	3	1	47	3.1
	Get username	0	0	45	2.3
	Get most recent order	2	3	125	5
tpc-c	payment	2	2	328	24
	Stock level	3	1	90	5
	Order status	6	12	208	12
	delivery	4	5	228	22
	New order	2	4	328	33
total		**54**	**43**	**2513**	_

4. Benchmark for Predefined Workload (VDBMWS-PRE)

- **Type of web services**. SOAP web services implemented in Java, which are nowadays commonly used for data exchange and systems integration [21].
- **Kind of vulnerabilities.** SQL Injection. Vulnerabilities allowing SQL Injection are especially significant in web services [7].
- **Detection methods.** Penetration testing, static code analysis, [22], [23].

Goal to characterize a delegate workload we have chosen to adjust code from three standard benchmarks created by the Transactions Processing Performance Council, to be specific: TPC-App, TPCC, and TPC-W (for details http://www.tpc.org).
Table I shows the collection of web services discovered by the security professionals, LOC indicate number of lines of the code and Avg.C indicate average Cyclometric complexity of the code. The outcomes display 54 vulnerable inputs and 43 vulnerable SQL queries in the collection of services. Clearly, this was an unsafe decision as there was some likelihood of getting code without vulnerabilities. However, as expected, the final code incorporates a few SQL Injection vulnerabilities (see Table 2), which is illustrative of the present circumstance in real web services development (as appeared in [5], [18]).
The workload services are presently implemented in Java. To rise a benchmark domain, we need to improve the workload in more languages.

Table 2: Tools under Benchmarking

Code	Provider	Tool	Technique
VT1	HP	WebInspect	Penetration testing
VT2	Kali linux	Kali	
VT3	Acunetix	Web Vulnerability Scanner	
SAT 1	Univ. Maryland	FindBugs	Static code analysis
SAT 2	Veracode,Inc	Veracode	
SAT 3	IBM	Rational software analyzer	

4.1 Experimental Evaluation

Three penetration-testing tools have been benchmarked, including three well-known commercial tools, in particular: Kali linux, HP WebInspect, and Acunetix scanner. Sequence of the tools changed in evaluation.

A. Benchmarking Results

To exhibit benchmarks, Tools chosen in the Stage1 (planning) are executed on the workload code (Stage2. execution). A vulnerability detailed are physically affirmed, differentiated and then recognized on a Stage1 to compute a benchmark measurement and prioritize the tools (Stage3.analysis). Tools properly identified the vulnerability are calculated as true positives. Tools not properly identified the vulnerability with the previous ones are physically affirmed and then only it is stated as false positives. In spite of the fact that this isn't a stage incorporated into the benchmarking method, it was very helpful to endorse the outcome of the code audits directed by the security experts.
Table 4 introduces a potential rank for the tools for VDBMWS-pre and PEBMWS-USD. Fragmented the ranking into inputs or q
ueries for the predefined benchmarking (VDBMWS-PRE).
Table 3 shows some whole benchmarking outcomes (F-Measure, precision, and recall). Moreover, 2 of the static code analysis tools (SAT2 and SAT3) show preferred outcomes over the penetration-testing tools. SAT1 and VT3 shows exceptionally poor F-Measure

outcomes. About recall, SAT3 have the best outcomes, next SAT2 is best. VT3 is lower for the both F-Measure and recall, regarding precision is good and announced no false positives but it recognized just 3 vulnerabilities. Static analysis is better than penetration testing except SAT1.

Table 3: Benchmarking Results for both benchmark

Benchmarking	Injection	Tool	F-Measure	Precision	Recall
UserDefined	SQL	VT1	0.0554	1	0.0285
		VT2	0.3984	0.4361	0.3666
		VT3	0.2963	0.336	0.265
	XSS	VT1	0.4	0.3119	0.3505
		VT2	1	0.0588	0.1111
		VT3	0.4173	0.2924	0.3439
PreDefined	Inputs(SQL)	VT1	0.2626	0.3693	0.2037
		VT2	0.1894	0.2731	0.145
		VT3	0.0226	1	0.0114
	Queries(SQL)	SAT1	0.168	0.1561	0.1618
		SAT2	0.8719	0.6324	0.7331
		SAT3	0.7293	1.1236	0.8845

Table 4: Benchmarking Ranking for both benchmark

Benchmarking	Injection	Criteria	1st	2nd	3rd
UserDefined	SQL	F-Measure	VT2	VT3	VT1
		Precision	VT1	VT2	VT3
		Recall	VT2	VT3	VT1
	XSS	F-Measure	VT1	VT3	VT2
		Precision	VT2	VT1	VT3
		Recall	VT1	VT3	VT2
PreDefined	Inputs	F-Measure	VT2	VT1	VT3
		Precision	VT3	VT2	VT1
		Recall	VT2	VT1	VT3
	Queries	F-Measure	SAT3	SAT2	SAT1
		Precision	SAT2	SAT3	SAT1
		Recall	SAT3	SAT2	SAT1

Figure 1 demonstrates a vulnerability revealed via tools (bar chart exhibits the quantity of liabilities identified through PT and SAT). A main perception is that every one of the tools distinguished under 34% of the vulnerabilities. SAT3 distinguished greater number of vulnerability with 100% of TP (outstanding outcome), however it recognized ≈37% false positives. Indeed, an issue shared by SAT1, which stated more than ≈68% of false positives. The reason is that these tools identify certain examples that generally point out the vulnerabilities, but many times they identify vulnerabilities that do not exist, because of natural restrictions of the static outline of the script.

Tool	TP	FP
VT1	23.05%	0.61%
VT2	32.39%	55.32%
VT3	1.82%	0.00%
SAT1	13.89%	68.80%
SAT2	56.28%	8.27%
SAT3	100%	37.12%

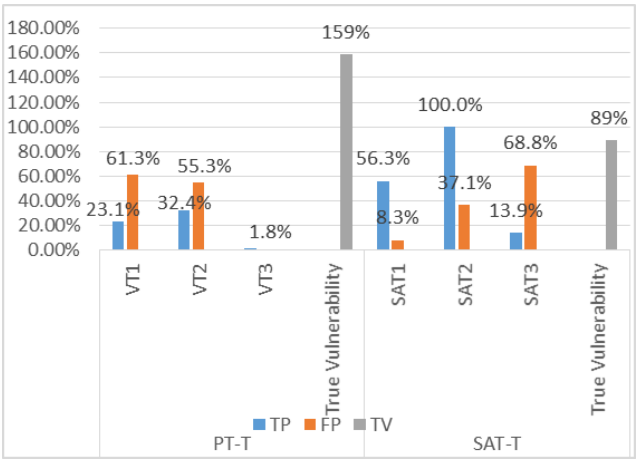

Fig. 1: VDBMWS-PRE-results for both PT and SAT

5. Benchmark for User Defined Workload (PEBMWS-USD)

- The type of web services - SOAP web services [21],
- Kind of vulnerabilities- SQL Injection [2] and XSS [6]
- Detection methods. Penetration testing [22].

5.1 Workload Definition and Characterization

We propose a programmed method to distinguishing the collection of liabilities beached on the utilization of a tool that consolidates attack signatures and interface monitoring to recognize SQL Injection vulnerabilities in web services [20].
A vital feature is proposed benchmark could be simply stretched out to various kinds of infusion vulnerabilities. The main imperative is that the benchmark client needs to characterize a workload comprising different sorts of vulnerability and afterward physically portray these vulnerabilities.

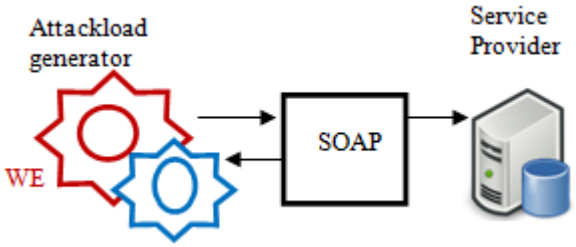

Fig. 2: Simple representation of ws-sign detection tool

5.1.1 Vulnerabilities Identification

A benchmark incorporates the Sign-WS tool, which actualizes the method proposed in [20]. By using attack signatures and interface monitoring method (ASIM) tends to the confinements of

penetration testing in the detection of injection vulnerabilities in web services.
A Workload Emulator component evaluates a web service portrayal also produces collection of legitimate solicitations, those are later altered by the attack injector unit. For the period of these procedure, the interfaces have been checked toward identify the signatures that denote vulnerability.

5.2 Experimental Evaluation

To exhibit benchmarks, I have taken a collection of web services incorporated into the proposed benchmark. To describe a workload (Stage1. planning) we utilized the ASIM approach. Benchmark over the penetration testing tools (exhibited on table 1) running on the workload code (Stage2. execution). A vulnerability detailed are physically affirmed, differentiated and then recognized on a Stage1 to compute a benchmark measurement and prioritize the tools (Stage3.analysis).

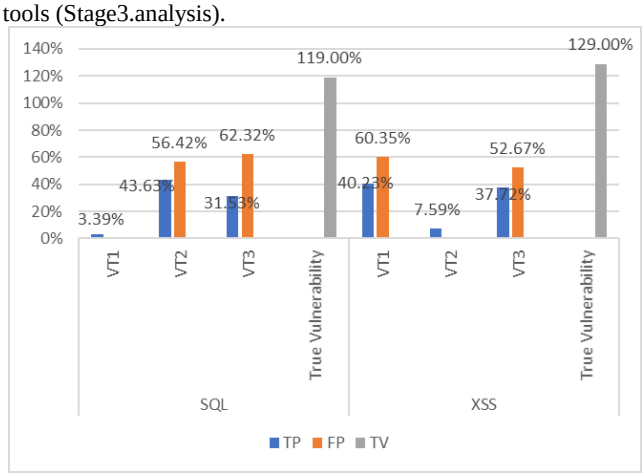

Fig. 3: PEBMWS-USD results for the PT

Tools	TP(SQL)	FP(SQL)	TP(XSS)	FP(XSS)
VT1	3.39%	0.00%	40.23%	60.35%
VT2	43.63%	56.42%	7.59%	0.00%
VT3	31.53%	62.32%	37.72%	52.67%

5.2.1 Characterization of the Workload

A Sign-WS tool identified vulnerability has been physically affirmed to ensure a nonexistence of a false positives (asserted on [20]). The tool stated zero false positives, however a scope was just 73.9% (119 tp obtainable of 161 TV). Later we will demonstrate, in spite of the fact that some of the true vulnerabilities are considered while count of some measurements, sign-WS stated ones were sufficient for a decent estimation of the tools efficiency.

5.2.2 Benchmarking Results

Table 4 displays a benchmark measurement for each tool, Sign-WS revealed a vulnerability upon consider a base collection of 119 vulnerabilities. As should be obvious, highest F-Measure is VT2 and then VT3. VT1 shows exceptionally poor F-Measure outcomes. About precision, VT1 is best and announced no false positives, and VT2 shows some best outcomes. At last, as far as recall, VT2 have the best outcomes, next VT3 best. A recall of VT is low as it recognized just 3 vulnerabilities.
Fig. 3 demonstrates a vulnerability revealed via tools (bar chart exhibits the quantity of liabilities identified through Sign-WS tool). A main perception is that every one of the tools distinguished under 44% of the vulnerabilities revealed through Sign-WS.
For cross site scripting, we have taken collection of 129 vulnerabilities. VT1 is best for F-measure and recall. About precision VT2 is best but it produces lower outcomes for F-measure and recall. Some tool identify SQL injection is very well but it's fail to detect XSS.

5.2.3 Contrast with the Predefined Benchmark

To analyze some consequences for a current benchmark through a benchmark of a predefined workload. In spite of the fact that we are thinking about the similar collection of web services, in user defined workload consider just subcategory of current vulnerability. This is clearly additionally a path for authenticating the workload description, measurements estimate methods proposed to help a benchmark.
Table 3 depicts some measurements acquired for together benchmarks. Of course, the measurements contrast somewhat in light of the fact that the base collection of TV is unique. The F-Measure points are reliably lesser in VDBMWS-PRE. Because of the greater values for recall in PEBMWS-USD. At long last, precision is the equal in the two benchmarks, except for the instance of VT2. This is because of the Sign-WS tool not reported a vulnerability identified for VT2 also is excluded in the base collection of true vulnerabilities. But the coverage is high in Sign-ws. Indeed, it doesn't influence the relative outcomes and the tools positioning is exactly the equal for the two benchmarks.

6. Conclusions

This research work produced an approach to describe benchmarks for vulnerability detection tools in web services. This approach has been utilized to characterize two concrete benchmarks focusing on tools capable to identify SQL Injection vulnerabilities and cross site scripting. The first one is in view of predefined workload, while the second one is user responsible for describing a workload. Some of the tools has been benchmarked, comprising both commercial and open-source tools.
The outcomes demonstrate a proposed benchmark effortlessly utilized to evaluate and contrast different experts. Indeed, some benchmark measurements gave a simple method to prioritize the tools in benchmarking, prompting comparable rankings in the two scenarios.
A benchmark property was authenticated also examined then recommend benchmarking method useful to determine benchmarks for vulnerability detection tools focusing on various domains.
Furthermore, work can be expanding benchmark to different sorts of vulnerability and utilizing a benchmarking method to deal with characterize benchmarks for various kinds of web services. The job of collecting and portraying the workload might expensive, however a few situations it surely might be justified regardless of the effort as it enables to comprehend the efficiency for some various tools accessible to identify vulnerability. Finally, we also have an idea to automate the verification of outcomes then the support of a benchmark.

References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, Web Services: Concepts, Architectures and Applications. first ed., Springer, 2010.
[2] S. Christey and R.A. Martin, "Vulnerability Type Distributions in CVE," The MITRE Corporation. V1, 1 2007.
[3] J. Gray, "The Benchmark Handbook: For Database and Transaction Processing Systems". Morgan Kaufmann Publishers Inc, 1993
[4] https://www.checkmarx.com/2017/12/03/closer-look-owasp-top-10-application-security-risks/
[5] H. Madeira, M. Vieira, N. Antunes, "Using Web Security Scanners to Detect Vulnerabilities in Web Services," International Conference on Dependable Systems and Networks, Lisbon, Portugal, July 2009
[6] S. Fogie, J. Grossman, R. Hansen, A. Rager, and P.D. Petkov, XSS Attacks: Cross Site Scripting Exploits and Defense, Syngress Publishing, 2007.

[7] D. Stuttard and M. Pinto, The Web Application Hacker's Handbook.
[8] M. Vieira and N. Antunes, "Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services," Proc. 15th IEEE Pacific Rim Int'l Symp. Dependable Computing (PRDC '09), pp. 301-306, 2009.
[9] J. Fonseca, H. Madeira, and M. Vieira, "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks," Proc. Presented at the 13th Pacific Rim Int'l Symposium on Dependable Computing (PRDC '07), pp. 365-372, 2007.
[10] Van Rijsbergen C.J., Information Retrieval. Buttersworth, 1979.
[11] G. Rothermel and H. Do, S. Elbaum, "Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact," Empir. Softw. Eng., vol. 10, pp. 405–435, Oct. 2005.
[12] P. Trischberger, S. Wagner, C. Koller, and J. J€urjens, "Comparing Bug Finding Tools with Reviews and Tests," Proc. 17th Int'l Conf. Testing of Communi. Systems, pp. 40-55, 2005.
[13] N. Antunes, N. Laranjeiro, M. Vieira, and H. Madeira, "Effective Detection of SQL/XPath Injection Vulnerabilities in Web Services," Proc. IEEE Int'l Conf. Services Computing (SCC '09), pp. 260-267, 2009.
[14] A. Orso and W.G.J. Halfond, "Preventing SQL Injection Attacks Using AMNESIA," Proc. 28th Int'l Conf. Software Eng., pp. 795-798, 2006.
[15] L. Spainhower and K. Kanoun, "Dependability Benchmarking for Computer Systems. John Wiley & Sons-IEEE CS Press", 2008.
[16] H. Madeira and M. Vieira, "Towards a Security Benchmark for Database Management Systems," Proc. Int'l Conf. DSN '05, pp. 592-601, 2005.
[17] A.C. d. Ara_ujo Neto and M. Vieira, "Selecting Secure Web Applications Using Trustworthiness Benchmarking," Int'l J. Dependable and Trustworthy Information Systems, vol. 2, no. 2, pp. 1-16, 2011.
[18] HP WebInspect, 2008, http://www.hp.com
[19] https://www.acunetix.com/vulnerability-scanner/
[20] M. Vieira and N. Antunes, "Enhancing Penetration Testing with Attack Signatures and Interface Monitoring for the Detection of Injection Vulnerabilities in Web Services," Proc. IEEE Int'l Conf. Services Computing (SCC), pp. 104-111, 2011.
[21] W. Nagy, F. Curbera and N. Mukhi "Unraveling the Web services Web: An Introduction to SOAP, WSDL, and UDDI," IEEE Internet Computing, vol. 6, no. 2, pp. 86-93, Mar./Apr. 2002.
[22] G. McGraw and S. Stender "Software Penetration Testing," IEEE Security & Privacy, pp. 84-87, Jan./Feb. 2005.
[23] J.D. Morgenthaler and N. Ayewah, "Using Static Analysis to Find Bugs," IEEE Software, vol. 25, pp. 22-29, Sept./Oct. 2008.