J Supercomput (2018) 74:1061–1089
https://doi.org/10.1007/s11227-016-1770-3

An algorithm to find relationships between web vulnerabilities

Fernando Román Muñoz[1] · Luis Javier García Villalba[1]

Published online: 13 June 2016

Abstract Over the past years, there has been a high increase in web sites using cloud computing. Like usual web site, those web applications can have most of the common web vulnerabilities, like SQL injection or cross-site scripting. Therefore, cloud computing has become more attractive to cyber criminals. Besides, in many cases it is necessary to comply with regulations like PCI DSS or standards like ISO/IEC 27001. To face those threats and requirements it is a common task to analyze web applications to detect and correct their vulnerabilities. The most used tools to analyze web applications are automatic scanners. But it is difficult to comparatively decide which scanner is best or at least is best suited to detect a particular vulnerability. To evaluate scanner capabilities some evaluation criteria have been defined. Often a web vulnerability classification is also used to evaluate scanners, but current web vulnerability classifications do not usually include all vulnerabilities. To face evaluation criteria which are not up-to-date and to have the fullest possible classification, in this paper a new method to map web vulnerability classifications is proposed. The result will be the vulnerabilities an automatic scanner has to detect. As classifications change over time, this new method could be executed when the existing classifications change or when new classifications are developed. The vulnerabilities described this way can

✉ Luis Javier García Villalba
javiergv@fdi.ucm.es
http://gass.ucm.es

Fernando Román Muñoz
froman@pdi.ucm.es
http://gass.ucm.es

1 Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), Faculty of Information Technology and Computer Science, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases, 9, Ciudad Universitaria, 28040 Madrid, Spain

also be seen as a web vulnerability classification that includes all vulnerabilities in the classifications taken into account.

Keywords Cloud computing vulnerabilities · Cyber-security · Vulnerability classifications · Web vulnerabilities

1 Introduction

Using automatic scanners like Acunetix [1], HP Webinspect [2], IBM AppScan [3] or OWASP Zed Attack Proxy [4] is a common task to asses dynamically the security of web application, a.k.a dynamic application security testing (DAST), reachable through networks like the internet. This is even more true and necessary to web applications running on cloud computing environments because often other kind of analyses is not possible, like static application security testing (SAST), and because cloud computing presents an added level of risk [32].

Security professionals have two main requirements [44] about automatic scanners: which should be their capabilities and if their applications are protected against a particular vulnerability or vulnerabilities, from a particular classification. To evaluate scanners capabilities some evaluation criteria have been defined and sometimes vulnerability classifications have been used. One of the main challenges of evaluation criteria is that they are not up-to-date. To know if an application has a particular vulnerability from a particular classification, mappings between classifications have been developed. The drawbacks of the mappings are that they are not full and up-to-date.

1.1 Background

An approach to evaluate scanners capabilities is the web application security scanner evaluation criteria (WASSEC) [68], where a guideline to enable anyone to evaluate web application security scanners is provided. It covers all stages to assess web application security, including crawling, testing and reporting. It was developed in 2009 and it has not been updated, so it does not include new features that scanners need and in fact they used to provide. Although the guideline is accurate and useful, there is little information about the results of applying it. Only a few papers, like [30] or [61], use WASSEC to compare and show the differences between various web application security scanners.

Another interesting approach to evaluate web application scanners on their ability to effectively test web applications and identify vulnerabilities is the NIST Special Publication 500-269 [49]. Unlike WASSEC, it just defines the minimum functionality requirements for web application security scanner. It was published in 2008 and it has not been updated since then. At least to the author's knowledge, there are no subsequent scanner evaluations that use it.

As a part of the NIST Software Assurance Metrics and Tool Evaluation (SAMATE) project, the definitions and functions of a web application scanner are presented in [25]. It includes the definitions of web application and web application scanner, identifying a list of vulnerabilities that a web application scanner should detect. The proposed list is not extensive, it just includes some common vulnerabilities, but it provides references to other web security issues classifications like WASC or CWE.

To test software security tools, the National Institute of Standards and Technology creates the SAMATE Reference Dataset (SRD) described in [12]. It is a compilation of test cases design to test vulnerability scanners and other related tools. It was initially developed as a set of C source code tests and later it has included other programming languages like PHP and other kind of code like binaries. It includes a lot of security flaws that should be used by end users, developers, students or security professionals to test their scanners. To the author's knowledge there are no developed works about using the test cases to evaluate security scanners.

Fuzzy metrics for classification offer a special case of research about scanner assessment. The authors of [41] define some fuzzy metrics to grade both, vulnerabilities and capabilities of Web application scanners. Fuzzy metrics in this paper are based on the difficulty of detection of the vulnerability, turn based on the true-positive, true-negative, false-positive, and false-negative values. The target is to grade every scanner for each vulnerability using vulnerable test web sites, but it is left for further work. The proposed method is potentially effective to evaluate scanner capabilities, but mainly due to the continuous updating of scanners, it can involve too much effort. Besides it does not make clear which vulnerable tests web site should be used to grade vulnerabilities and scanners.

Because there are no standards of the capabilities of security scanners, in [42] the common vulnerabilities and exposures (CVE) are used to define types of weakness, idiosyncrasies, faults and flaws (WIFFs). The result is the common weakness enumeration (CWE). The target is to obtain an enumeration of software security issues that can serve to measure scanners capabilities, in accordance with key industry players. Although it is frequently updated it is not indeed used as a standard of security scanner capabilities, at least regarding web applications scanners. Their relations with other security issues classifications are also not keep updated.

There are not only methods to evaluate dynamic web vulnerability scanners, but methods have also been defined for evaluating static source code analysis tools. In [43] a prototype benchmark for C source code analyzers is described. Their goal is to include in the benchmarking all vulnerability types but it did not indicate whether they do or not.

Scanners requirements can be customized to the web application that is going to be analyzed. In [7] is propose a guide to the elaboration of scanner requirements. The goal is to map the security requirements, from the analysis of requirements of the application being developed, with the items in the OWASP testing guide. Later the scanner that better matches the security requirements is selected to scan the web application. The benefit of this approach is that it includes the processing of vulnerabilities at the initial stages of the life cycle of system development. The main drawback is that it can leave untreated critical vulnerabilities if they have not been taken into account in the requirement analysis stage.

The OWASP Top 10 [57] vulnerability classification has been used in several papers to test the analysis capabilities of web vulnerability scanners. Some of them, like

[23,24] or [45], developed vulnerable web applications with vulnerabilities from the OWASP Top 10, others, like [9], also use vulnerable versions of established web applications. Vulnerable web applications are analyzed to test a few scanners. The conclusions are the challenges that scanners need to overcome. Two of the major challenges are crawling web applications properly when the application has multi-step forms and to execute client-side code like JavaScript or Flash.

Some papers take into account just a few vulnerabilities. For example, in [26] just "cross side scripting" and "SQL injection" detection capabilities are evaluated. In [27] also just "cross side scripting" and "SQL injection" detection capabilities are evaluated after injecting those vulnerabilities in web applications. Reference [67] tests just one "cross-site scripting" scanner. In [22] the detection capabilities of the scanners are compared in testing six known vulnerabilities.

Other papers are focused on a specific environment. In [38], the web-based user interface of a SCADA application is assessed. The work [21] classifies web services vulnerabilities. In [6,8] the security vulnerabilities of open-source health care systems are analyzed.

There are also papers that focus on open-source or low-cost scanners. For example, [46,55] recommends a combination of open-source or low-cost scanners and [52] provides guidelines in selecting free scanners.

A few papers analyzed the detection capabilities of scanners over a broad set of vulnerabilities. Reference [15] reports features and detection capabilities of 64 scanners in detecting 33 vulnerabilities. Reference [64] analyzed the accuracy of seven scanners using the own vendor's test sites.

To face the two major challenges of scanners, as seen before, in [59] the authors propose a method of analyzing web page, simulating the behavior of a browser and getting values from external sources like [28] where necessary, attempting to obtain a set of correct values to fill web form fields. The authors tested the method on a virtual store and on a set of web pages. In the first case, valid form field values were obtained to create new accounts. In the second case, the field values found lead to new web page.

Previous paper [58] compared five of these studies focused on web application vulnerability scanners. The results showed the scanners, vulnerabilities, and vulnerable web applications that are evaluated in every paper. Regarding vulnerabilities tested, overall 27 vulnerabilities are tested, but just two of the most popular vulnerabilities are tested in all evaluations: "cross side scripting" and "SQL injection". Any of the analyzed studies gave information about web vulnerability scanner tools accuracy in detecting vulnerabilities in best known classifications like OWASP or WASC. Therefore, it was concluded that it is necessary to define a vulnerability classification to test both web application vulnerabilities and the accuracy of web vulnerability scanners.

There are many useful surveys which give broad coverage of creating ontologies or taxonomies focused on cyber-security-related concepts area [13,33]. They have not been included in this section because this paper focuses on finding vulnerability mappings and defines scanners capabilities. This paper does not focus on defining any ontology or taxonomy.

1.2 Contributions

In this paper, the authors describe a new method to map security issues of various classifications. The targets are: (1) keep the list of vulnerabilities updated that vulnerability scanners have to detect and (2) know the relations between vulnerabilities in different classifications, to help decide if my web application is protected against vulnerabilities of a particular classification. The new method uses the information about web security issues mappings provided by web vulnerability classification developers, other organizations and specially designed search engine queries.

Although the relationships provided by classification developers are more reliable, there is also another piece of information on the internet that links issues from some classification providing additional relationships. This information is mostly provided by security professionals talking about a particular issue.

During the process described in this paper, all security issues are reduced to a small and unique set of keywords that can identify it. The results of the process are the mapping between security issues classifications and a new classification with all security issues included in the classifications taken into account. This classification can be used as automatic scanner requirements.

The structure of the paper is as follows: Sect. 2 describes web vulnerability classifications and the mappings between them. Section 3 explains our method to establish relationships between security issues. Section 4 describes the results of applying the method with several web security issues classifications. Finally, Sect. 5 concludes the paper and describes future work.

2 Vulnerability classifications and relations

2.1 Vulnerability classifications

A number of organizations categorized cyber-security issues [47]. This section characterizes the most common classifications related to application cyber-security issues. It covers both web applications and applications of any kind.

The WASC Threat Classification (WASC TC) [69] provided by the Web Application Security Consortium (WASC) classifies web application threats with the aim of obtaining a comprehensive list of web vulnerabilities. This list classifies web threats according to weaknesses or attacks, and indicates the source of the threat: design, implementation, or deployment.

The list contains understandable descriptions and examples of 49 threats, and the current version 2.0 was released in 2010.

OWASP Top 10 is provided by the Open Web Application Security Project (OWASP) and contains the top 10 most critical web application security risks. They obtain the data from consulting companies and scanner tools vendors. The current version was released in 2013. This list includes not only descriptions of the vulnerabilities accompanied by examples and methods to detect them, but also some methods to mitigate the impact of the vulnerabilities.

OWASP also develops the OWASP Testing Guides (OWASP TG) that describe a set of vulnerability tests to check web applications. In these guides, web vulnerability tests are grouped into several sets. OWASP TG v3 [53] contains 66 items and v4 [54] contains 88. Version v3 was released in 2008 and v4 in 2014. These guides include the descriptions of the vulnerabilities, examples, and methods to detect the weaknesses; it does not include any methods to mitigate the impact of the vulnerabilities, but does include references to other OWASP information about mitigation.

NIST Special Publication 500-269 was elaborated by NIST SAMATE project in 2007, and describes the tasks that a web applications scanner must be able to accomplish. Annex A to that document includes a list of 14 vulnerabilities that a web application security scanner must be able to identify. Vulnerabilities in this list have been included based on the likelihood of their exploits. Annex B suggests some brief mitigation methods.

WASSEC was elaborated on by WASC in 2009, and includes a list of security problems that a web application scanner should detect. The list of problems was mostly extracted from WASC TC, and includes 55 items grouped into several categories:

– Authentication
– Authorization
– Client-side attacks
– Command execution
– Information disclosure.

SecToolMarket [15] compares the features of web vulnerability scanners and includes a web vulnerability classification developed by Shay Chen that lists 33 web vulnerability scanner audit features. This list is usually updated annually and includes web vulnerability descriptions and references to WASC TC v2.0 and OWASP TG v3.

The Common Weakness Enumeration (CWE) [17] is a set of software weaknesses developed by The MITRE Corporation that classifies not only web application vulnerabilities, but also all kinds of application vulnerabilities. CWE includes 1003 software weaknesses. The current version 2.8 was released in 2015, and includes examples, detection methods, potential mitigations, and references to other vulnerability classifications.

SANS and MITRE developed CWE/SANS TOP 25 Most Dangerous Software Errors (SANS CWE/25) [62], which were last updated in 2011. It is a subset of CWE and, like CWE, classifies all kinds of software vulnerabilities. This classification includes examples and detection methods.

Another classification that includes all kinds of software vulnerabilities is the Common Attack Pattern Enumeration and Classification (CAPEC) [66]. This list contains attack patterns and is updated by The MITRE Corporation. The current version 2.1 was released in 2013. CAPEC includes examples and attack execution flow descriptions.

Tables 1 and 2 summarize the features of analyzed vulnerability classifications. Table 1 includes web vulnerability classifications, and Table 2 all kinds of vulnerability classifications. As can be seen and according to [71,72] classifications of software cyber-security issues are usually out-of-date.

Table 1 Features of web vulnerability classifications

Classification	TC	OWASP Top 10	OWASP v3 testing guide	OWASP v4 testing guide	NIST Special Publication 500-269	WASSEC	SecTool Market
Developer	WASC	OWASP	OWASP	OWASP	NIST	WASC	Shay Chen
Concept	Threat	Risk	Vulnerability test	Vulnerability test	Vulnerability	Problem of security	Auditing feature
Number	49	10	66	88	14	55	33
Last version	2.0 (2010)	2013	2008	2014	1.0 (2007)	1.0 (2009)	2012
Description	Yes	Yes	Yes	Yes	Yes-briefly	No	Yes-briefly
Examples	Yes	Yes	Yes	Yes	No	No	No
Detection	No	Yes	Yes	Yes	No	No	No
Mitigation	No	Yes	No	No	Yes-briefly	No	No
References	Yes	Yes	Yes	Yes	No	No	Yes

Table 2 Features of all kinds of software vulnerability classifications

Classification	SANS CWE/25	CWE	CAPEC
Provider	SANS-MITRE	MITRE	MITRE
Concept	Weakness	Weakness	Attack pattern
Number	25	1003	463
Current version	3.0 (2011)	2.8 (2015)	2.6 (2014)
Description	Yes	Yes	Yes
Example	Yes	Yes	Yes
Detection	Yes	Yes	Yes
Mitigation	No	Yes	Yes
References	Yes	Yes	Yes

2.2 Cloud computing security issues

A categorization of security issues for cloud computing is presented in [32]. It is focused in the three cloud computing service models (SaaS, PaaS and IaaS), identifying and summarizing the main vulnerabilities and threats in cloud computing systems. Furthermore, it describes the relationship between those vulnerabilities and threats, how these vulnerabilities can be exploited and also some countermeasures. They get information about security issues in cloud computing from sources like the ACM digital library or the IEEE digital library. It includes all kinds of vulnerabilities that may affect cloud computing environments, for example, vulnerabilities in virtual machines, networks, operating systems, data bases, web servers or web applications. Just because of trying to reach every cloud computing security issue, it does not provide detailed information on any of them, like web application vulnerabilities.

The aim of [63] is to make a survey of the major security threats and vulnerabilities affecting cloud computing and the possible solutions available. From the top nine cloud computing threats identified by the Cloud Security Alliance (CSA) in 2013 and a criterion that can be met by a vulnerability to be cloud specific, seven major vulnerabilities of cloud computing have been identified. The main solution proposed to these security issues is the service level agreement (SLA) between the vendor and the customer, including a good degree of encryption standards. The vulnerability list does not include common vulnerabilities that may also impact the service, just cloud-specific ones.

In [19] it is given an complete and useful description of cloud computing including its main characteristics and services models. It also highlights and categorizes many of security issues introduced by the cloud computing and makes some recommendations to mitigate the risks related to the use of these services. As in other cases, a survey about security issues in cloud computing is shown. Unlike other papers, some real-world examples of vulnerability in cloud computer are included.

Otherwise, tools like Acunetix and IBM App Scan, offer online vulnerability scanner services that take advantage of its cloud computing infrastructure. Its main advantage is that there are no specialized system requirements on clients.

2.3 Cyber-security issues

Although these classifications classify by different cyber-security concepts, they can be mapped. For example, considering two of the most well-known vulnerabilities, Cross Side Scripting and SQL Injection, they are attacks in WASC TC v2, weaknesses in CWE and risks in OWASP Top 10.

Different classifications classify different issues according to vulnerability, threats, weakness, risks, controls, audit features, or attack patterns. Although different terms are used, according to risk analysis methods, all of them are related. In [11], risk is defined as the potential harm caused if a particular threat exploits a particular vulnerability to cause damage to an asset. Reference [10] includes the risk analysis structure: identify the assets in the system, find possible threats against the identified assets, quantify the risks and to develop countermeasures to mitigate the identified risks. Prashanth and Sambasiva [56] defines vulnerability as flaws in software or hardware that may provide an attacker the open door to get into a computer or network and have unauthorized access to resources; threat as any potential action or situation that may exploit a vulnerability; risk as the result of a successful exploitation and safeguard as control or countermeasure to a vulnerability.

Also papers about risk mitigation highlight the relations between cyber-security terms. For [40] risks are closely associated with system vulnerabilities and vulnerabilities refer to any weaknesses that can be exploited. Reference [20] calculates the risk of a threat over an asset by multiplying the probability of occurrence of the threat over the asset by the relevance of the asset and dividing the result by the effectiveness of the security controls implemented. In [21] vulnerabilities are flaws or weaknesses in a system that could be exploited, exploits are ways known to take advantage of specific software vulnerabilities and threats have the potential to violate security.

In addition, there are papers which are not about risk analysis or mitigation that reveals these relations. In [31], where the effect of security requirements on the functional requirement is studied, an asset is something inside the system, an attack is what a system must be protected and a vulnerability is a weakness of a system that an attack tries to exploit. Reference [51], which speak of intrusion detection systems, defines a vulnerability as a flaw or weakness in a system which could be exploited and an attack as any malicious attempt to exploit vulnerability.

NIST Special Publication 800-30 [50] and International Standard ISO/IEC 27001 [34] glossaries also establish the relationships between security concepts such as risk, threat, vulnerability or security control.

Figure 1 shows a summary of the relations between these concepts. Weaknesses are mistakes in software that in proper conditions could contribute to the introduction of vulnerabilities. Vulnerabilities can be exploited by threats, and can lead to the loss, damage or destruction of an asset. The risk is the probability of a threat of exploiting vulnerabilities. Testing web applications is to check if the application includes some controls or safeguards to avoid risks. Web vulnerability scanner has audit features to test web applications and attack patterns are methods for exploiting applications.

As can be seen, although all these concepts are different, all of them are related to each other. Therefore, when speaking about vulnerabilities, in this paper it will actually refer to the corresponding cyber-security issue, according to their classification.

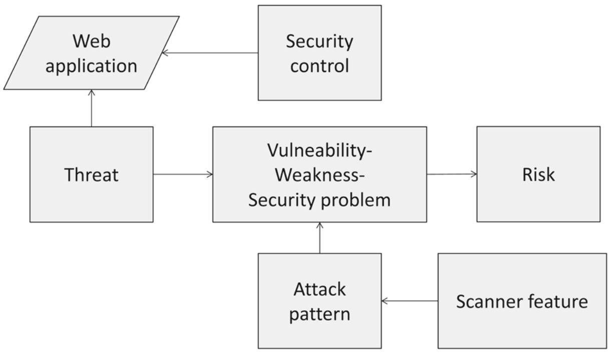


Fig. 1 Relationships between vulnerability cyber-security concepts

2.4 Mappings between classifications

Web vulnerability classification mappings are made either by a web classification developer or by an independent entity. They include all vulnerabilities of one or more classifications and try to map vulnerabilities of other classifications. Below is a brief description of the main mappings between the web vulnerability classifications.

Mapping by Denim Group (Denimgroup) [16], written in 2010, correlates web vulnerabilities on WASC TC v1.0, SANS CWE/25, and OWASP Top 10 2004 and 2007.

WASC Threat Classification to OWASP Top Ten RC1 Mapping (Jeremiahgrossman) [29] is a mapping created by Jeremiah Grossman, which was updated in 2009. It correlates web vulnerabilities on WASC TC v2.0 and OWASP Top 10 2010.

Threat Classification Taxonomy Cross Reference View (Webappsec) [70] is a view of WASC, updated in 2013. It maps WASC TC v2.0 with CWE, CAPEC, OWASP Top Ten 2004, 2007, and 2010 and SANS CWE/25. This mapping uses the information provided by the Denim Group and the Jeremiah Grossman mappings.

Previously mentioned NIST Special Publication 500-269 annex A includes a list of security problems and a mapping between some web vulnerability classifications. This publication correlates its own list of vulnerabilities with CWE, OWASP Top 10 2007, and Common Vulnerabilities and Exposures (CVE) [18].

Also Common Weakness Enumeration (CWE) [17] includes in some of their software weaknesses references to threats in the WASC TC [69] and to OWASP Top 10 [57].

Securing Telligent Evolution (Telligent) [65] is a document created by Telligent in 2012. It describes the threats that are tested in the Telligent Evolution platform and how they map to OWASP and WASC threats.

In 2013, Critical Watch company researched [37]. They optimized Webappsec and added new relationships to obtain a mapping between OWASP Top 10 (2013), WASC TC v2.0, SANS CWE/25 and CWE.

Last, as mentioned above, SecToolMarket includes a web vulnerability mapping between the WASC TC v2.0 and the OWASP TG v3.

Table 3 shows the current vulnerability classification mappings that map the classifications described previously. Also the source of the mapping is shown. Primary key is the classification that has all its vulnerabilities in the mapping.

The information about web vulnerability classification mappings can be summarized to establish which classifications are more closely related to other classifications.

The WASC TC v2.0 is mapped with eight classifications, SANS CWE/25 is mapped with five classifications, OWASP Top 10 2007, OWASP Top 10 2010 and WASC TC v1.0 with three classifications, and OWASP Top 10 2005 with two. SecToolMarket, the OWASP TG, CWE, CVE, and CAPEC are related to just one classification.

3 Searching for web vulnerability relationships

In this section, a method to develop an up-to-date web vulnerability classification is shown. The classifications obtained will have web vulnerabilities from current classifications, and available mappings between them. The key point is to reduce each vulnerability name to a unique short summary of keywords, or tags, that describe the vulnerability, and then use search engine queries.

3.1 General steps

The first step (1) is to select the current web vulnerability classifications. This task can be achieved by searching in organizations like OWASC, WASC or MITRE, and extracting their classifications.

The second step (2) is to find the mappings with other classifications that each organization supplies. Mappings developed by organizations can be supplied in two ways: through a single document with the relationships or through a body of documents each one associated with a vulnerability.

The third step (3) is to search for mappings provided by other organizations or previous studies. There are organizations and previous papers that do not develop their own classifications, but do provide mappings between others.

The last step (4) is to look for relationships between vulnerabilities not included in previous mappings. This step is achieved by searching on the web sites of the organizations that develop classifications. To search on web sites of organizations for vulnerability relationships several tests were performed. The conclusion was that the best way is to reduce each vulnerability name to a unique summary of three keywords. Then queries composed by the three keywords summary and the vulnerability code suffix of each classification are searched on the web sites of organizations.

Figure 2 schematically describes those four steps. The tasks that are performed in the step number four (4) are described in the next subsection.

3.2 Searching internet for vulnerability relationships

The goal of the fourth step introduced in the previous paragraph is to find web vulnerability relationships which are not included in classification mappings. New

Table 3 Vulnerability classification mappings

Mapeo	Classifications	Primary key	External source	Last update
Denimgroup	OWASP Top 10 (2004), OWASP Top 10 (2007), SANS CWE/25, WASC TC (v1)	All	No	2010
Webappsec	WASC TC (v2), CWE, CAPEC, SANS CWE/25, OWASP Top 10 (2004), OWASP Top 10 (2007), OWASP Top 10 (2010)	WASC TC (v2)	Denimgroup and Jeremiahgrossman	2013
NIST	NIST, CWE, OWASP Top 10 (2007) [5]	NIST	No	2008
Telligent	OWASP TG v3, WASC TC (v2)	None	No	2011
Jeremiahgrossman	OWASP Top 10 (2010), WASC TC (v2)	None	No	2010
Critical Watch	OWASP Top 10 (2013), WASC TC (v2), SANS CWE/25, CWE	OWASP Top 10 (2013)	Webappsec	2013
Sectoolmarket	Sectoolmarket, OWASP TG (v3), WASC TC (v2)	Sectoolmarket	No	2012

relationships mean relationships not supplied by sources like classification developer mappings or other organization mappings. These new relationships are found in organization web site documents about some vulnerability. The relationships found in this way are obtained from sources that are not focused on a whole classification, but focused on just one vulnerability or a few of them.

3.2.1 Extracting relations from web pages

To extract relations from web pages a process like extracting security entities from text is going to be used. Extracting security entities can be accomplished using regular expressions [36] or using named entity recognition tools [48]. Because named entity recognition tools often fail to identify many cyber-security domain concepts [36], in this paper, regular expressions have been used to extract vulnerability relations from text in web pages.

To look for such relationships between vulnerabilities the queries structure will be the vulnerability name and a classification code prefix, like "OWASP-" or "WASC-". Then the vulnerability code will be extracted from the response. The challenge at this point is to decide what the vulnerability name is, in other words, which keywords unequivocally describe the vulnerability.

3.2.2 Keyword extraction

Topic model is a type of analysis for discovering the words that are included, more or less frequently, in documents or pieces of text. There are two main sorts of topic modeling methods [39]. One is focus on discovering common topics, for example, [14] and the other on distinct topics, as in [35]. Common topics are the more representative

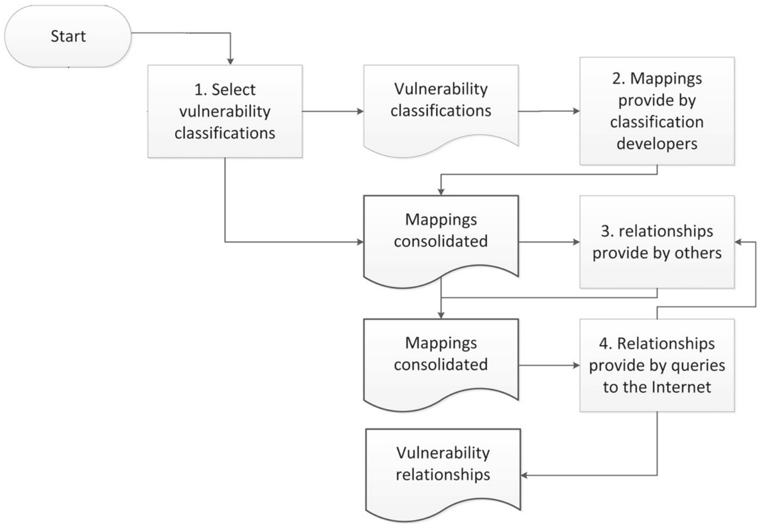

Start
1. Select vulnerability classifications
Vulnerability classifications
2. Mappings provide by classification developers
Mappings consolidated
3. relationships provide by others
Mappings consolidated
4. Relationships provide by queries to the Internet
Vulnerability relationships

Fig. 2 Method to get web vulnerability classification mappings

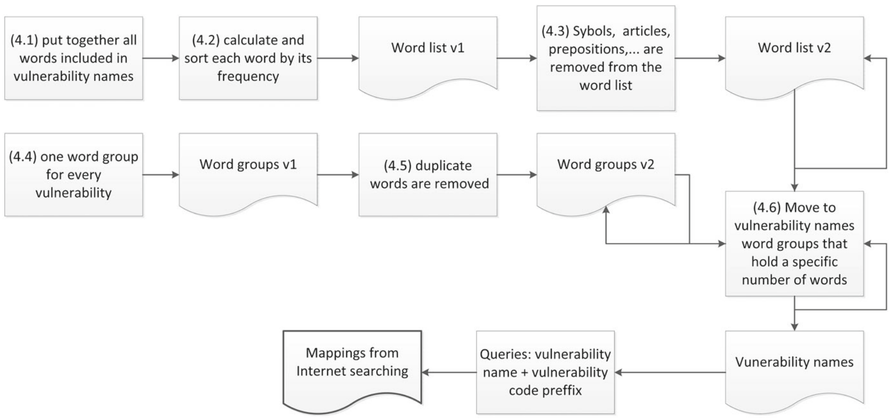

(4.1) put together all words included in vulnerability names
(4.2) calculate and sort each word by its frequency
Word list v1
(4.3) Sybols, articles, prepositions,... are removed from the word list
Word list v2
(4.4) one word group for every vulnerability
Word groups v1
(4.5) duplicate words are removed
Word groups v2
(4.6) Move to vulnerability names word groups that hold a specific number of words
Vunerability names
Queries: vulnerability name + vulnerability code preffix
Mappings from Internet searching

Fig. 3 Mappings from web search queries: task of the process of finding a set of keywords that describe each web vulnerability

words of the text, the words that tell what the subject of the text is. Otherwise, distinct topics are those that can differentiate a piece of text or document in a set of piece of texts or documents.

In this paper, distinct keywords are needed. The most frequent words in vulnerability descriptions are the less important words. The keywords will be the words that could be used to distinguish vulnerabilities from one another. The steps of the process to reduce vulnerability names to a few keywords that stand for the vulnerabilities are as follows, as shown in Fig. 3:

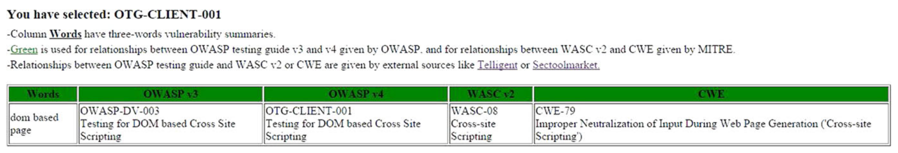
4.1 Put together all the words that are included in the vulnerability names of all classifications.
4.2 Calculate and sort each word by its frequency to create a word list.
4.3 Characters like quotes, commas or brackets, and articles and prepositions are removed from the word list.
4.4 Vulnerability names in the classifications for the same vulnerability are put together in a word group, one word group for each vulnerability.
4.5 Duplicated words inside each word group are removed.
4.6 Word groups that hold a specific number of words are moved to the final vulnerability name list, and the first word in the word list (the word with a higher frequency) is removed from the word groups and then from the word list.

Then steps (4.6) are repeated until every word group has the specific number of words. In the next section, the process described in this section has been implemented to unify four vulnerability classifications and get their mappings.

4 Experiment and discussion

4.1 Implementation

The first step of the algorithm is to select the vulnerability classifications. The selected classifications to tests the method described in the Sect. 3 are OWASP TG v3, OWASP



You have selected: OTG-CLIENT-001

-Column **Words** have three-words vulnerability summaries.

-Green is used for relationships between OWASP testing guide v3 and v4 given by OWASP, and for relationships between WASC v2 and CWE given by MITRE.

-Relationships between OWASP testing guide and WASC v2 or CWE are given by external sources like Telligent or Sectoolmarket.

Words	OWASP v3	OWASP v4	WASC v2	CWE
dom based page	OWASP-DV-003 Testing for DOM based Cross Site Scripting	OTG-CLIENT-001 Testing for DOM based Cross Site Scripting	WASC-08 Cross-site Scripting	CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Fig. 4 Result of applying the method to get mappings between classifications of web vulnerabilities

TG v4, WASC TC 2.0 and CWE. In CWE just web vulnerability is taken into account. There are several search engines that can be used to do the queries. In this page, Google search engine has been used.

The second step is to look for mappings provided by classifications developers. The Web Application Security Consortium provides a mapping between OWASP TC 2.0 and CWE.

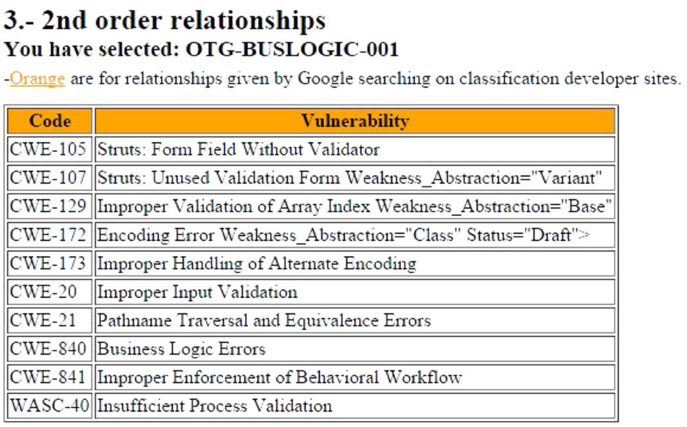
Also OWASP provides relationships between its two classifications OWASP TG v3 and OWASP TG v4. OWASP does not provide a true mapping list, but those mappings can be obtained from the OWASP site. Searching in the OWASP site for vulnerability codes from one classification in redirection pages returns the relationships. The request www.google.es/search?q=site:owasp.org%20+%20%27Redirect%20+%20page%27%20+%20-%20Cate%20gory%20+%20%27OWASP-DV-001%27, for example, returns a page in the OWASP site that link OWASP-DV-001 with OTG- INPVAL-001. An automatic process has been implemented to get all relationships between OWASP classifications v3 and v4. The third step is to add the mappings provided by other organizations. Telligent and Sectoolmarket mappings have been used in this step.

Last, the fourth step is executed to get vulnerability keywords, and use them to search for vulnerability relationships. The process to get vulnerability names with a specific number of words has been implemented in a software program. First the program has been executed to obtain two keywords for each vulnerability, but at some point in the process more than one vulnerability was related with the same two keywords. Then it has been executed to obtain three keywords for each one. In this second execution of the algorithm, the proper result has been obtained.

To get new relationships an automatic process has been implemented to perform search engine queries composed of three-word vulnerability names and vulnerability classification code prefix. For example, the request https://www.google.es/search?q=site:owasp.org+'OTG-'+dos+'buffer+overflows&safe returns pages in the OWASP site that link the vulnerability "fulfilment + api + contract" with an OWASP TG v2 vulnerability.

The final mappings that include all steps mappings can be in the web page [60]. Figure 4 shows the web vulnerability mappings web page. In this web page, selecting a vulnerability code from one classification shows the related vulnerability codes and names in other classifications, and the three keywords that describe the vulnerability. Relationships are separated by first- and second-order relationships.

First-order relationships are those provided by mappings of classifications developers or other organizations. Second-order relationships are those found by Google queries on organization web sites. Second-order relationships shown in [60] are those



3.- 2nd order relationships

You have selected: OTG-BUSLOGIC-001

-Orange are for relationships given by Google searching on classification developer sites.

Code	Vulnerability
CWE-105	Struts: Form Field Without Validator
CWE-107	Struts: Unused Validation Form Weakness_Abstraction="Variant"
CWE-129	Improper Validation of Array Index Weakness_Abstraction="Base"
CWE-172	Encoding Error Weakness_Abstraction="Class" Status="Draft">
CWE-173	Improper Handling of Alternate Encoding
CWE-20	Improper Input Validation
CWE-21	Pathname Traversal and Equivalence Errors
CWE-840	Business Logic Errors
CWE-841	Improper Enforcement of Behavioral Workflow
WASC-40	Insufficient Process Validation

Fig. 5 Result of applying the method to get relationships between web vulnerabilities

relationships obtained on the date the algorithm was executed. Figure 5 shows an example of second-order relationships.

In this developed web page, the user can select a vulnerability code from one of the four selected classifications. Then the web page shows the following data: (i) the three keywords that describe the vulnerability related to the selected code, (ii) vulnerability codes and descriptions of the four classifications. These relationships are the first-order relationships and they have been attained from vulnerability classifications mappings developed by the own organizations and also other organizations, and (iii) vulnerability codes and description, obtained from queries on organizations web sites. These are the second-order relationships.

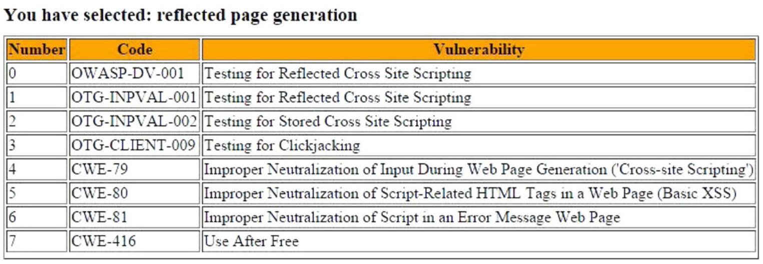
As previously mentioned, in this paper new mappings between web vulnerability classifications are developed. In fact, two new mappings are developed that did not previously exist. Those new mappings are a mapping between OWASP TG v4 and WASC TC v2, and another between OWASP TG v4 and CWE. Also the mapping between WASC TC v2 and CWE is updated.

Table 4 shows several example of (4.4)–(4.6) steps. The first column indicates the vulnerability code in the four selected classifications. Then the next column shows the vulnerability names or descriptions. The third column is the result of joining the words in the descriptions, and then removing repeated words, punctuation marks and other words like prepositions. The last column includes the three-word summary for each vulnerability. Each three-word summary is different for each vulnerability.

Two other web pages have been developed in [60] to provide relationships. Both web pages query the internet online. One page shows the results of searching for web vulnerability relationships on the classification sites. An example can be seen in Fig. 6. The second one searches for relationships on the whole internet. Figure 7 shows an example. The information provided by these two web pages may change from one query to another. They will include the new relationships that can be extracted from new information on the internet about web vulnerabilities.

Table 4 Examples of shortening vulnerabilities descriptions of three keywords

Vulnerability	(4.4)	(4.5)	(4.6)
OWASP-CM-004	Application configuration management testing	Application configuration	Platform server misconfiguration
OTG-CONFIG-002	Test application platform	Management testing	
WASC-14	Configuration Server misconfiguration	Test platform	
CWE-16	Configuration	Server misconfiguration	
OWASP-AT-004	Brute force testing	Brute force	Lockout predictability problems
OTG-AUTHN-003	Testing for weak lockout mechanism	Testing weak	
WASC-11	Brute force	Lockout mechanism	
CWE-350	Predictability problems	Predictability problems	
OWASP-AZ-001	Testing for path traversal	Limitation pathname	Limitation pathname restricted
OTG-AUTHZ-002	Testing for bypassing authorization schema	Restricted testing	
WASC-33	Path traversal	Path traversal	
CWE-22	Improper limitation of a pathname to a restricted directory: ('Path Traversal')	Bypassing authorization	
		Schema improper	
		Restricted directory	

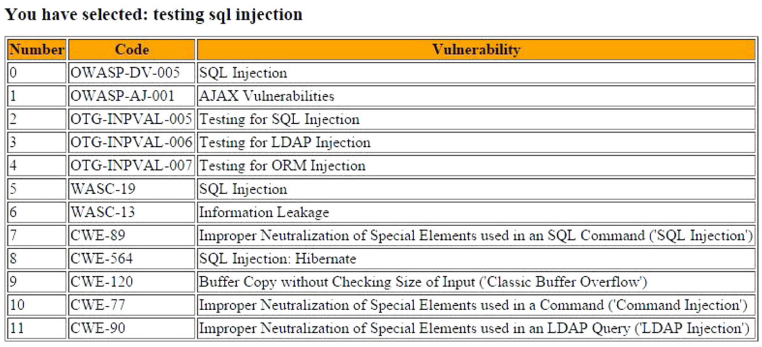

You have selected: reflected page generation

Number	Code	Vulnerability
0	OWASP-DV-001	Testing for Reflected Cross Site Scripting
1	OTG-INPVAL-001	Testing for Reflected Cross Site Scripting
2	OTG-INPVAL-002	Testing for Stored Cross Site Scripting
3	OTG-CLIENT-009	Testing for Clickjacking
4	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
5	CWE-80	Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
6	CWE-81	Improper Neutralization of Script in an Error Message Web Page
7	CWE-416	Use After Free

Fig. 6 Example of results of searching for web vulnerability relationships on the classification sites

4.2 Experimental setup

To test the implementation of the proposed algorithm some vulnerabilities have to be selected. Then the results are checked to test if the relationships obtained are correct. The selected vulnerabilities must clearly have its appropriate cyber-security issue in every classification. The vulnerabilities that do not match plainly with issues in the classifications should be avoided. If one of these vulnerabilities were used, the result could not be judged, because there is no way to compare it with the true relationships.



You have selected: testing sql injection

Number	Code	Vulnerability
0	OWASP-DV-005	SQL Injection
1	OWASP-AJ-001	AJAX Vulnerabilities
2	OTG-INPVAL-005	Testing for SQL Injection
3	OTG-INPVAL-006	Testing for LDAP Injection
4	OTG-INPVAL-007	Testing for ORM Injection
5	WASC-19	SQL Injection
6	WASC-13	Information Leakage
7	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
8	CWE-564	SQL Injection: Hibernate
9	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
10	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')
11	CWE-90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')

Fig. 7 Example of results of searching for web vulnerability relationships on the whole internet

Table 5 Vulnerabilities selected to test the algorithm implementation results (obvious relationships)

Usual name	Keywords	OWASP TG v3	OWASP TG v4	WASC TC v2	CWE
SQL injection	Testing, SQL, injection	OWASP-DV-005	OTG-INPVAL-006	WASC-19	CWE-89
Session fixation	Testing, session, fixation	OWASP-SM-003	OTG-SESS-003	WASC-37	CWE-384
Reflected cross-site scripting, reflected XSS	Refected, page, generation	OWASP-DV-001	OTG-INPVAL-001	WASC-08	CWE-79
Path traversal	Limitation, path, restricted	OWASP-AZ-001	OTG-AUTHZ-002	WASC-33	CWE-22
Cross-site request forgery, CSRF	CSRF, request, forgery	OWASP-SM-005	OTG-SESS-005	WASC-09	CWE-352

So, to test the implementation of the algorithm, five OWASP Top 10 2013 security risks have been selected. The five vulnerabilities have their corresponding item in the four classifications.

To test if the method proposed in this paper is meaningful, three metrics are defined:

– The first metric (1) measures whether searching for vulnerability relationships on organization web sites or on the whole internet find the most obvious relationships. Obvious relations are those that clearly match vulnerability keywords with vulnerability codes from classifications. The better method will produce the more obvious relationships.
– The second metric (2) measures the number of true, but less obvious relationships, that link the vulnerabilities which are really related. For example "limitation path restricted" is related to "CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')" and also with "CWE-36: Absolute Path Traversal" and "CWE-23: Relative Path Traversal".

– The third metric (3) measures if the remaining relationships provide data about detection and mitigation methods. For example, “OWASP-AJ-001: Testing for AJAX Vulnerabilities” which is about testing the potential vulnerabilities in asynchronous JavaScript and XML, includes testing for SQL injection attacks through this technique, that are also true relations.

The second and the third metrics are about the other true relationships, which are not obvious. They are relationships that are not obvious, do not really link the same vulnerability, but include valuable data about detection and mitigation methods.

Table 5 shows the selected vulnerabilities. The first column of the table indicates the usual name given to the vulnerability and the second column shows the keywords that summarizes the vulnerability descriptions. The other four columns are the vulnerability codes for each classification.

4.3 Results

The five vulnerabilities selected in the previous subsection are searched in the three web pages developed, looking for web vulnerability relationships. The results are shown in Table 6. For each vulnerability, four results are shown:

1. First-order relationships: relationships included in web vulnerability classification mappings, developed by classification organizations or other organizations.
2.1. Second-order relationships: relationships found from Google searches on web sites of classification developers. These data were acquired when the algorithm was executed.
2.2. Online second-order relationships: relationships found from Google searches on web sites of classification developers. These data are updated on the fly, at the time the query is sent. They are the same queries like those in the previous item, but the result may be different.
3. Online third-order relationships: relationships found from Google searches on the whole internet. These data are also updated on the fly.

Item (ii) relationships were collected in April 2015. Items (iii) and (iv) relationships shown in this table were collected in June 2015.

4.4 Discussion

From the results shown in Table 6 the three metrics defined previously can be rated. Figure 8 shows the percentage of obvious relationships included for each vulnerability, i.e., the metric (1) values. As can be shown, in most cases the relationships found using the proposed method include the obvious ones. In the worst case, half of the obvious relationships are found, and even in some cases the success rate is 100 %.

The average success rate is 75 %. For example, the data for “testing session fixation” note that, when searching first time for second order relations, three of the four vulnerability codes “OWASP-SM-003”, “OTG-SESS-003”, “WASC-37” and “CWE-384” are include in the results.

Table 6 Relationships between the five web vulnerability selected

Three keywords	First-order relations	Second-order relations		Second-order relations		Third-order relations	
Testing SQL injection		OWASP-DV-005		OWASP-DV-005			
		OTG-INPVAL-005		OWASP-AJ-001			
		OTG-INPVAL-006		OWASP-WS-004		OWASP-DV-005	
		OTG-INPVAL-007		OTG-INPVAL-005		OWASP-AJ-001	
		OTG-AUTHN-004		OTG-INPVAL-006		OTG-INPVAL-005	
		OTG-INPVAL-010		OTG-INPVAL-007		OTG-INPVAL-006	
	OWASP-DV-005	OTG-INPVAL-011		OTG-AUTHN-004		OTG-INPVAL-007	
	OTG-INPVAL-006	WASC-19	WASC-13	OTG-INPVAL-010		WASC-19	WASC-13
	WASC-19	CWE-89	CWE-564	OTG-INPVAL-011		CWE-89	CWE-564
	CWE-89	CWE-120	CWE-94	WASC-19	WASC-13	CWE-120	CWE-77
		CWE-565	CWE-209	CWE-89	CWE-564	CWE-90	
		CWE-20	CWE-79	CWE-120	CWE-94	CWE-78	
		CWE-209	CWE-565	CWE-20	CWE-79		
		CWE-78					
Testing session fixation				OWASP-SM-003			
		OWASP-SM-003		OWASP-AT-009			
	OWASP-SM-003	WASC-37	WASC-47	OTG-SESS-003		OWASP-SM-003	
	OTG-SESS-003	CWE-384	CWE-664	OTG-SESS-001		OTG-SESS-003	
	WASC-37	CWE-732	CWE-287	WASC-37	WASC-47	OTG-SESS-004	
	CWE-384	CWE-698	CWE-79	CWE-384	CWE-664	WASC-18	WASC-37
		CWE-352		CWE-732	CWE-287	WASC-47	CWE-384
				CWE-698	CWE-79		
				CWE-352			

Table 6 continued

Three keywords	First-order relations	Second-order relations		Second-order relations		Third-order relations	
Reflected page generation		OTG-INPVAL-001		OWASP-DV-001		OWASP-DV-001	
	OWASP-DV-001	OTG-INPVAL-002		OTG-INPVAL-001		OTG-INPVAL-001	
	OTG-INPVAL-001	OTG-CLIENT-009		OTG-INPVAL-002		OTG-INPVAL-002	
	WASC-08	CWE-79	CWE-80	OTG-CLIENT-009		CWE-79	CWE-22
	CWE-79	CWE-81	CWE-416	CWE-79	CWE-80	CWE-78	CWE-639
				CWE-81	CWE-416		
Limitation path restricted		OWASP-AZ-001		OWASP-AZ-001			
		OWASP-CM-001		OWASP-CM-001			
		OTG-CRYPST-003		OTG-INFO-008		OWASP-AZ-001	
	OWASP-AZ-001	OTG-CRYPST-001		OTG-CRYPST-003		OWASP-CM-001	
	OTG-AUTHZ-002	WASC-33	CWE-22	OTG-CRYPST-001		WASC-04	WASC-33
	WASC-33	CWE-494	CWE-36	WASC-33	CWE-22	CWE-22	CWE-494
	CWE-22	CWE-23	CWE-41	CWE-494	CWE-36	CWE-23	CWE-264
		CWE-21	CWE-119	CWE-23	CWE-41		
		CWE-73		CWE-21	CWE-119		
				CWE-73			
CSRF request forgery	OWASP-SM-005	OWASP-AJ-001		OWASP-AJ-001			
	OTG-SESS-005	OWASP-SM-005		OWASP-SM-005		WASC-09	
	WASC-09	OTG-SESS-005		OTG-SESS-005		CWE-352	
	CWE-352	CWE-352	CWE-441	CWE-352	CWE-441		
		CWE-442		CWE-442			

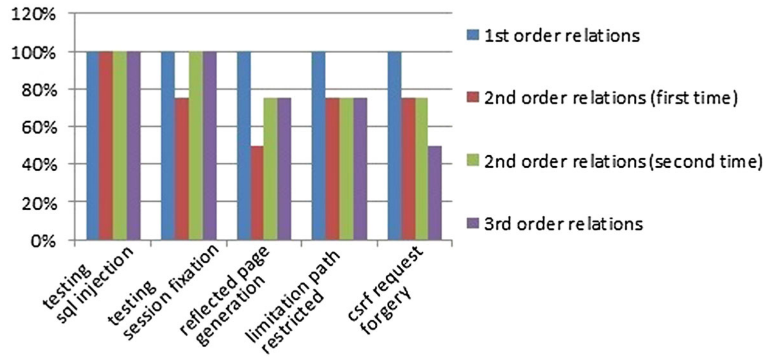


Fig. 8 Percentage of obvious relationships found

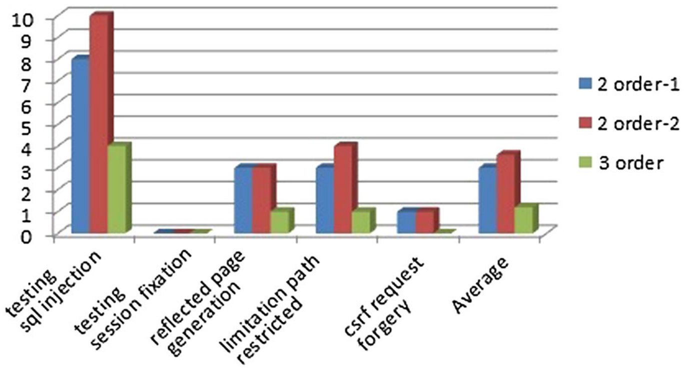


Fig. 9 Metric (2) results for the five vulnerabilities selected

Figures 9 and 10 show the number of true relationships found for the metrics (2) and (3). Although sometimes there are no relations, it usually found some new ones. Joining metrics (2) and (3), Fig. 11 is obtained that for almost every vulnerability some new relationships are found.

From the results shown in Table 6 some other useful conclusions can be derived:

(a) First-order relationships, which are obtained from the organizations that develop classifications, are the same as the obvious relationships shown in Table 5. This result is the expected result, when comparing the obvious relationships with the vulnerability mappings developed by organizations.
(b) There are some differences between the two second-order relationships, although it has only been 2 months from one another. During these two months some new relationships have been defined, and some old relationships have disappeared. For example, CWE code “CWE-494”, which was related to “limitation path restricted” in April 2015, is no longer in June 2015.
(c) As can be seen in Fig. 12, relations are always found. Depending on the vulnerability and the order, the number oscillates from two to about twenty.

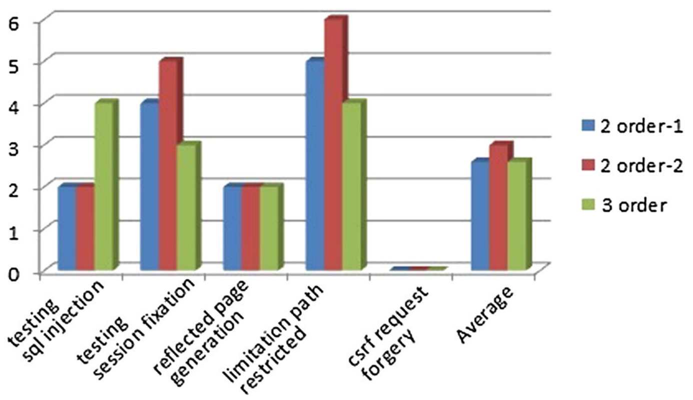


Fig. 10 Metric (3) results for the five vulnerabilities selected

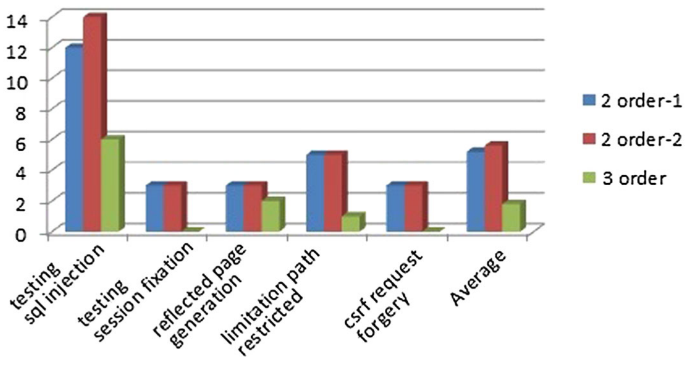


Fig. 11 Metric (2) plus metric (3) results for the five vulnerabilities selected

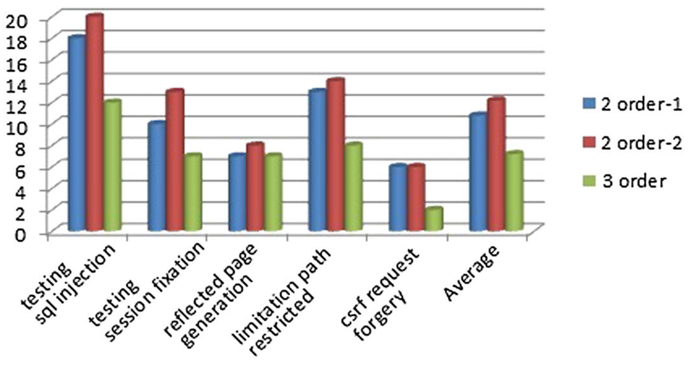


Fig. 12 Total relations found

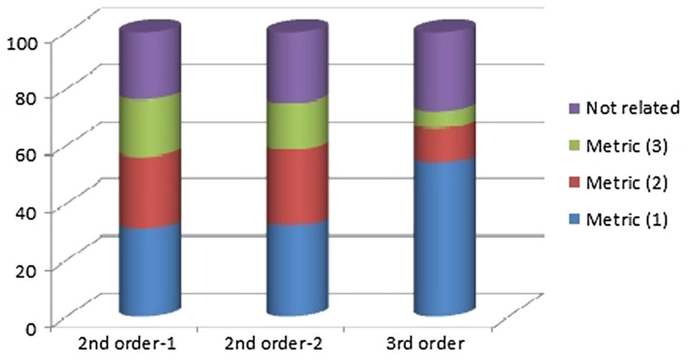


Fig. 13 Distribution by metric of the relationships found for each order

(d) In any of the three kinds of relations (second and third orders) the percentage or true relations are almost 75 %. Figure 13 shows the average percentage of true relationships found in each of the three searches for every metric.
(e) Third-order relationships are fewer than second-order relationships. Looking for relationships on the whole internet produces fewer relationships than looking for relationships on web sites of organizations.
(f) As can be seen in Table 7 and in Fig. 14 almost every tested vulnerability has some false positives. There are vulnerability codes which are unrelated to the searched vulnerability among the relationships found. These codes are found by the described method, but a manual review shows that they are not really related with that vulnerability.
(g) Most of vulnerability codes found in second- and third-order relationships are from CWE. This is because CWE not only includes the generic vulnerability like "SQL injection" but also other vulnerabilities like "SQL injection: Hibernate".

5 Conclusion and future work

One of the main challenges of testing for cyber-security issues, using automatic tools like web vulnerability scanners, is that there are no updated standards defining their capabilities and how to evaluate them. There are vulnerability classifications that could be used to define the scanner capabilities. The drawback of vulnerability classification is that there is not a full mapping between them. In addition, for a web application, there is not a reliable way to know if it is protected from a particular vulnerability of a classification. In this paper, a new method to map web vulnerabilities classifications is presented (Fig. 15). The key of the method is to reduce each vulnerability to a small and unique group of keywords, which represents the vulnerability and search for relations in the internet. Applying this method to current web vulnerability classifications, an up-to-date classification can be obtained. The new classification links vulnerabilities in current classifications, and gives a three-keyword summary of each vulnerability. The tests show that this method can find true relations between vulnerabilities in different classifications and that the majority of the relations found are real. The next step will

Table 7 Measure of the defined metrics

Summary of three keywords	Relations			
	First order	Second order-1	Second order-2	Third order
Testing SQL injection				
Metric (1)	4	4	4	4
Metric (2)		8	10	4
Metric (3)		4	4	2
Total related (1 + 2 + 3)		16 (89 %)	18 (90 %)	10 (83 %)
No related		2	2	4
Total		18	20	12
Testing session fixation				
Metric (1)	4	3	4	4
Metric (2)		0	0	0
Metric (3)		3	3	0
Total related (1 + 2 + 3)		6 (60 %)	7 (54 %)	4 (57 %)
No related		4	5	3
Total		10	13	7
Reflected page generation				
Metric (1)	4	2	3	3
Metric (2)		3	3	1
Metric (3)		0	0	1
Total related (1 + 2 + 3)		5 (71 %)	6 (75 %)	5 (71 %)
No related		2	2	2
Total		7	8	7
Limitation path restricted				
Metric (1)	4	3	3	3
Metric (2)		3	4	1
Metric (3)		2	1	0
Total related (1 + 2 + 3)		8 (62 %)	8 (57 %)	4 (50 %)
No related		5	6	4
Total		13	14	8
CSRF request forgery				
Metric (1)	4	3	3	2
Metric (2)		1	1	0
Metric (3)		2	2	0
Total related (1 + 2 + 3)		6 (100 %)	6 (100 %)	2 (100 %)
No related		0	0	0
Total		6	6	2

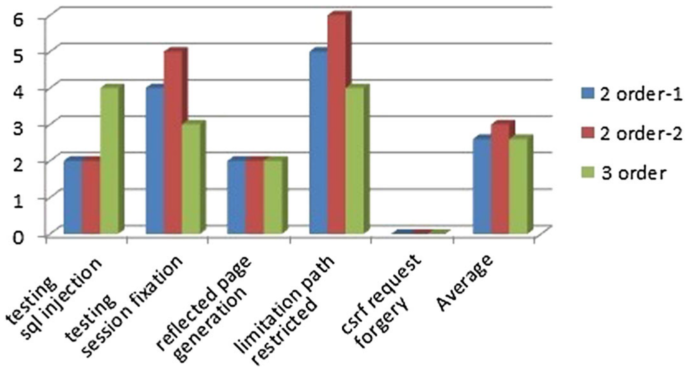


Fig. 14 False-positive relationships

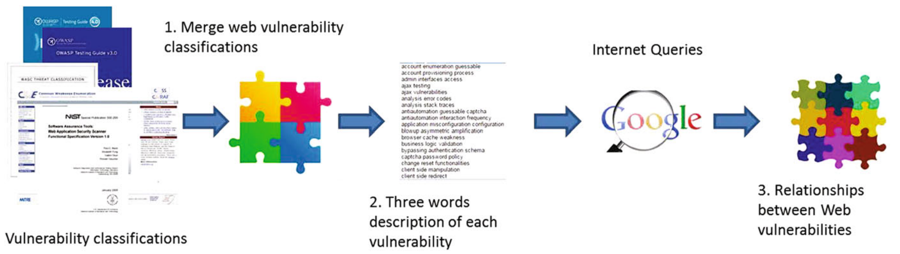

1. Merge web vulnerability classifications
Vulnerability classifications
account enumeration guessable
account provisioning process
admin interfaces access
ajax testing
ajax vulnerabilities
analysis error codes
analysis stack traces
antiautomation guessable captcha
antiautomation interaction frequency
application misconfiguration configuration
blowup asymmetric amplification
browser cache weakness
business logic validation
bypassing authentication schema
captcha password policy
change reset functionalities
client side manipulation
client side redirect
2. Three words description of each vulnerability
Internet Queries
Google
3. Relationships between Web vulnerabilities

Fig. 15 Process to get relationships between web vulnerabilities in different classifications and its main results

be to improve the algorithm described in this paper to reduce the number of false positives. Also a method to test if scanners can detect the vulnerabilities in this new classification has to be developed.

Acknowledgements This work was funded by the European Commission Horizon 2020 Programme under Grant Agreement Number H2020-FCT-2015/700326-RAMSES (Internet Forensic Platform for Tracking the Money Flow of Financially Motivated Malware).

European Commission

References

1. Acunetix (2015) Acunetix web vulnerability scanner. http://www.acunetix.com/. Accessed 30 Sept 2015
2. HP (2015) Hp webinspect. http://www.spidynamics.com/products/. Accessed 30 Sept 2015
3. IBM (2015) IBM rational appscan. http://www-01.ibm.com/software/awdtools/appscan/. Accessed 30 Sept 2015
4. Open Web Application Security Project (2015) Owasp zed attack proxy. https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.Accessed 30 Sept 2015
5. PCI Security Standards Council (2016) PCI SSC data security standards. https://www.pcisecuritystandards.org. Accessed 30 Sept 2015

6. Akowuah F, Lake J, Yuan X, Nuakoh E, Yu H (2015) Testing the security vulnerabilities of openemr 4.1.1: a case study. J Comput Sci Coll 30(3):26–35
7. Assad RE, Katter T, Ferraz F, de Lemos Meira S (2010) Security quality assurance on web-based application through security requirements tests based on owasp test document: elaboration, execution and automation. In: Proceedings of the 2nd OWASP Ibero-American Web Applications Security Conference
8. Austin A, Smith B, Williams L (2010) Towards improved security criteria for certification of electronic health record systems. In: Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care, SEHC '10, New York, pp 68–73
9. Bau J, Bursztein E, Gupta D, Mitchell J (2010) State of the art: automated black-box web application vulnerability testing. In: Proceedings of the IEEE symposium on Security and Privacy (SP), pp 332–345
10. Bergvall J, Svensson L (2015) Risk analysis review. In: Linkopings universitet
11. Bhattacharjee J, Sengupta A, Mazumdar C, Barik MS (2012) A two-phase quantitative methodology for enterprise information security risk analysis. In: Proceedings of the CUBE International Information Technology Conference, CUBE '12, New York, pp 809–815
12. Black PE, Kass M (2005) Software security assurance tools, techniques and metrics (SSATTM). In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, ASE '05, New York, pp 461–461
13. Blanco C, Lasheras J, Valencia-García R, Fernández-Medina E, Toval A, Piattini M (2008) A systematic review and comparison of security ontologies. In: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security, ARES '08. IEEE Computer Society, Washington, DC, pp 813–820
14. Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. J Mach Learn Res 3:993–1022
15. Chen S (2012) General features comparison—web application scanners. http://www.sectoolmarket.com
16. Cornel D (2010) Mapping between owasp top 10 (2004, 2007), wasc 24+2 and sans cwe/25. http://blog.denimgroup.com/denim_group/2010/01/mapping-between-owasp-top-10-2004-2007-wasc-242-and-sans-cwe25.html
17. Corporation TM (2013) Common weakness enumeration. http://cwe.mitre.org
18. Corporation TM (2013) Cwe. http://cve.mitre.org
19. Dahbur K, Mohammad B, Tarakji AB (2011) A survey of risks, threats and vulnerabilities in cloud computing. In: Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications, ISWSA '11, New York, pp 12:1–12:6
20. Daz-Lpez D, Dlera-Tormo G, Gmez-Mrmol F, Martnez-Prez G (2014) Dynamic counter-measures for risk-based access control systems: an evolutive approach. Future Generation Computer Systems
21. Demchenko Y, Gommans L, de Laat C, Oudenaarde B (2005) Web services and grid security vulnerabilities and threats analysis and model. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, GRID '05. IEEE Computer Society, Washington, DC, pp 262–267
22. Doupé A, Cavedon L, Kruegel C, Vigna G (2012) Enemy of the state: a state-aware black-box web vulnerability scanner. In: Proceedings of the 21st USENIX Conference on Security Symposium, Security'12. USENIX Association, Berkeley, p 26
23. Doupé A, Cova M, Vigna G (2010) Why johnny can't pentest: an analysis of black-box web vulnerability scanners. In: Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA'10. Springer, Berlin, pp 111–131
24. Ferreira AM, Klepee H (2011) Effectiveness of automated application penetration testing tools
25. Fong E, Okun V (2007) Web application scanners: definitions and functions. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences, HICSS '07, p. 280b. IEEE Computer Society, Washington, DC
26. Fonseca J, Vieira M, Madeira H (2007) Testing and comparing web vulnerability scanning tools for sql injection and xss attacks. In: Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing, PRDC '07. IEEE Computer Society, Washington, DC, pp 365–372
27. Fonseca J, Vieira M, Madeira H (2014) Evaluation of web security mechanisms using vulnerability & attack injection. IEEE Trans Dependable Secur Comput 11(5):440–453
28. Gonzalez H, Halevy AY, Jensen CS, Langen A, Madhavan J, Shapley R, Shen W, Goldberg-Kidon J (2010) Google fusion tables: web-centered data management and collaboration. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10. NY, USA, New York, pp 1061–1066

29. Grossman J (2013) Wasc threat classification to owasp top ten rc1 mapping . http://jeremiahgrossman.blogspot.com.es/2010/01/wasc-threat-classification-to-owasp-top.html
30. Gupta S, Sharma L (2011) Analysis and assessment of web application security testing tools. In: Proceedings of the 5th National Conference
31. Haley C, Laney R, Nuseibeh B (2004) Deriving security requirements from crosscutting threat descriptions, execution and automation. In: Proceedings of the 3rd International Conference on Aspect-Oriented Software Development, pp 112–121
32. Hashizume K, Rosado DG, Fernández-Medina E, Fernandez EB (2013) An analysis of security issues for cloud computing. J Internet Serv Appl 4(1):1–13
33. Lannacone M, Bohn S, Nakamura G, Gerth J, Huffer K, Bridges R, Ferragut E, Goodall J (2015) Developing an ontology for cyber security knowledge graphs. In: Proceedings of the 10th Annual Cyber and Information Security Research Conference, CISR '15, New York, pp 12:1–12:4
34. International Organization for Standardization (2005) International standard iso/iec 27001
35. Jiang Y, Li X, Meng W (2014) Discword: learning discriminative topics. In: Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), vol 02, WI-IAT '14. IEEE Computer Society, Washington, DC, pp 63–70
36. Jones CL, Bridges RA, Huffer KMT, Goodall JR (2015) Towards a relation extraction framework for cyber-security concepts. In: Proceedings of the 10th Annual Cyber and Information Security Research Conference, CISR '15, New York, pp. 11:1–11:4
37. Jurcenoks J (2013) Owasp to wasc to cwe mapping correlating different industry taxonomy. Critical Watch
38. Khalili A, Sami A, Ghiasi M, Moshtari S, Salehi Z, Azimi M (2014) Software engineering issues regarding securing ics: an industrial case study. In: Proceedings of the 1st International Workshop on Modern Software Engineering Methods for Industrial Automation, MoSEMInA 2014, New York, pp 1–6
39. Kim H, Choo J, Kim J, Reddy CK, Park H (2015) Simultaneous discovery of common and discriminative topics via joint nonnegative matrix factorization. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, New York, pp 567–576
40. Kumar R, Singh H (2013) A qualitative analysis of effects of security risks on architecture of an information system. SIGSOFT Softw Eng Notes 38(6):1–3
41. Loh P, Subramanian D (2010) Fuzzy classification metrics for scanner assessment and vulnerability reporting. IEEE Trans Inf Forensics Secur 5(4):613–624
42. Martin RA, Barnum S (2008) Common weakness enumeration (cwe) status update. Ada Lett 28(1):88–91
43. Martin RA, Barnum S (2008) Creating the secure software testing target list. In: Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead, CSIIRW '08, New York, pp 33:1–33:2
44. Martin RA, Christey S, Jarzombek J (2005) The case for common flaw enumeration. NIST Workshop on Software Security Assurance Tools, Techniques, and Metrics
45. Martirosyan Y (2013) Security evaluation of web application vulnerability scanners strengths and limitations using custom web application. In: California State University
46. Mcquade K (2014) Open source web vulnerability scanners: the cost effective choice? In: Proceedings of the Conference for Information Systems Applied Research, Baltimore
47. Michell S (2013) Programming language vulnerabilities: proposals to include concurrency paradigms. Ada Lett 33(1):101–115
48. Mulwad V, Li W, Joshi A, Finin T, Viswanathan K (2011) Extracting information about security vulnerabilities from web text. In: Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, vol 03, WI-IAT '11. IEEE Computer Society, Washington, DC, pp 257–260
49. National Institute of Standards and Technology: Software assurance tools: web application security scanner functional specification version 1.0. NIST special publication 500-269
50. National Institute of Standards and Technology (2011) Nist special publication 800-30 revision 1: guide for conducting risk assessments. NIST special publication, p 95
51. Njogu HW, Jiawei L, Kiere JN, Hanyurwimfura D (2013) A comprehensive vulnerability based alert management approach for large networks. Future Gener Comput Syst 29(1):27–45

52. Nuno Teodoro CS (2010) Automating web applications security assessments through scanners. In: Proceedings of the OWASP Ibero-American Web Applications Security Conference
53. Open Web Application Security Project (2008) OWASP testing guide v3.https://www.owasp.org
54. Open Web Application Security Project (2014) OWASP testing guide v4.https://www.owasp.org
55. Parmar S (2015) Vulnerability checker for infosecurity. Int J Sci Res (IJSR) 4(3):1593–1596
56. Prashanth S, Sambasiva N (2015) Vulnerability, threats and its countermeasure in cloud computing. Int J Comput Sci Mobile Comput 4(6):126–130
57. Project OTT (2013) Open web application security project. https://code.google.com/p/owasptop10
58. Román Muñoz F, García Villalba LJ (2013) Methods to test web applications scanners. Amman, Jordan
59. Román Muñoz F, García Villalba LJ (2015) Web from preprocessor for crawling. Multimed Tools Appl 74(19):8559–8570
60. Román Muñoz F, García Villalba LJ (2015) Web vulnerability classification mappings 1. http://vulmappings.esy.es
61. Saeed FA (2014) Using wassec to evaluate commercial web application security scanners. Int J Soft Comput Eng (IJSCE) 4(1):177–181
62. SANS (2011) Cwe/sans top 25 most dangerous software errors. http://www.sans.org/top25-software-errors
63. Srivatsa S, Nagasundaram S (2015) Guidelines for security in cloud computing. Netw Commun Eng 7(7):305–306
64. Suto L (2010) Analyzing the accuracy and time costs of web application security scanners. In: Beyond Trust
65. Telligent (2013) Telligent evolution platform. https://community.zimbra.com/documentation/telligentcommunity/w/community7/24580.securing-telligent-evolution
66. The MITRE Corporation (2013) Common attack patterns enumeration and classfication. http://capec.mitre.org/
67. Tripp O, Weisman O, Guy L (2013) Finding your way in the testing jungle: A learning approach to web security testing. Proceedings of the 2013 International Symposium on Software Testing and Analysis, ISSTA 2013, New York, pp 347–357
68. Web Application Security Consortium (2009) Web application security scanner evaluation criteria. http://projects.webappsec.org/w/page/13246986/Web%20Application%20Security%20Scanner%20Evaluation%20Criteria
69. Web Application Security Consortium (2010) The WASC threat classification. http://projects.webappsec.org/w/page/13246978/ThreatClassification
70. Web Application Security Consortium (2012) Threat classification taxonomy cross reference view. http://projects.webappsec.org/w/page/13246975/Threat%20Classification%20Taxonomy%20Cross%20Reference%20View
71. Weber S, Karger PA, Paradkar A (2005) A software flaw taxonomy: aiming tools at security. SIGSOFT Softw Eng Notes 30(4):1–7
72. Weber S, Karger PA, Paradkar A (2005) A software flaw taxonomy: aiming tools at security. In: Proceedings of the 2005 Workshop on Software Engineering for Secure Systems—Building Trustworthy Applications, SESS ’05, New York, pp 1–7