METHODS TO TEST WEB APPLICATION SCANNERS

Fernando Román Muñoz, Luis Javier García Villalba

Group of Analysis, Security and Systems (GASS)
Department of Software Engineering and Artificial Intelligence (DISIA)
School of Computer Science, Office 431, Universidad Complutense de Madrid (UCM)
Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain
Email: {froman, javiergv}@fdi.ucm.es

Abstract

Web application vulnerability scanners are automated tools that probe web applications for security vulnerabilities. In order to assess the current state of the art options, many studies have been conducted. The studies are carried out selecting a set of tools and a web application vulnerable to know web vulnerabilities. The results show if the web vulnerability scanners can detect the vulnerabilities. In this work we compare the results of several studies. We propose some improvement points to web vulnerability scanner assessment. Some studies also include web vulnerability scanner challenges. One of the challenges is crawling web applications. In this work we also analyze and compare how web vulnerability scanners crawl web applications.

Keywords - Web vulnerability scanner, web vulnerabilities, security testing, crawling.

1 INTRODUCTION

A very popular option for web vulnerability assessment is to use an automated web vulnerability scanner. These are tools that firstly crawl a web application trying to enumerate all the pages and its associated input vectors. Secondly the tool generates special input values that are submitted to the web application, and lastly the tool checks the application's response looking for vulnerability evidences.

The accuracy of web vulnerability scanners has been assessed in some papers. Studies that assess web vulnerability scanners takes three inputs: a set of commercial or open-source tools, a list of vulnerabilities to detect, and a web application vulnerable to the vulnerabilities of the list. The tools are configured to test the vulnerable web application, and the results are analyzed. The conclusions of the studies are the crawling capabilities and vulnerabilities detected by each tool.

In this paper we compare the results of some evaluations of web vulnerability scanners. The goal is to check if a common approach can be defined to test web vulnerability scanners. We also compare the crawling feature of some tools to find out how these tools fill out web forms.

2 VULNERABILITIES IN WEB APPLICATIONS

Nowadays Web applications have extended its use to almost every area of our daily life, such as in our jobs, leisure or relationships with the Administration. Given that these applications are typically available to large numbers of users at any time, it is critical to detect and fix the vulnerabilities.

2.1 Detection of vulnerabilities in web applications

The methods to detect vulnerabilities in Web applications can be grouped into two classes: (1) static analysis from the web application code and its functional description, provided by the developer, and (2) dynamic analysis from the initial URL and a reduced set of credentials if needed.

In (1) the web application code is analyzed to detect fragments that correspond to known vulnerability patterns. This method achieves large code coverage and reaches almost all execution paths. The disadvantages of static analysis are that every single tool or human tester can only analyze a set of languages, and since the code is not executed, the analysis can produce too many false positives.

In (2) the application is run sending input values and recording and analyzing the outputs. This type of analysis does not depend on which language is used to write the code, and its use is quite widespread. One of its main disadvantages is that you can only analyze the pages of the web application that are reached, so it is necessary to first crawl the web application. Another disadvantage is that this method does not shows where the problem in the code is.

A dynamic vulnerability analysis is mainly divided into two phases: passive phase, also called crawling, and active phase. The objective of passive phase is to pass through the web application to locate all its pages and the associated input vectors, typically GET and POST parameters. In the active phase, once the scanner identifies all the inputs on theapplication's pages, it then attemptstoinject values for each parameter and observes the response. The key point in a dynamic vulnerability analysis is to crawl the Web application as deep as possible;otherwise it won't be possible to test every input vector.

One of the most widely used approaches to detect vulnerabilities in Web applications is to use tools, usually known as web vulnerability scanners, that perform the analysis in an automatic or semi-automatic way. From information like the initial URL and maybe a set of credentials provided by the web application administrator, these tools carry out the analysis following the phases described in the previous section.

3 EVALUATIONS OF WEB VULNERABILITY SCANNERS

There are severalstudies available that present evaluations of web vulnerability scanners. We have chosen five leading studies [1 to 5] which evaluate some web vulnerability scanners. In this section we summarize those studies.Those studies present evaluations of several web vulnerability scanners both commercial and open-source. The evaluations use vulnerable web applications to check the crawling capabilities of the tools and different types of vulnerabilities.

3.1 Tools and web applications

This subsection explains what tools and web applications are tested in the studies introduced in the previous paragraph.

Table 1 shows the web vulnerability scanners that are evaluated in thestudies. Although each study evaluates at least seven tools, there are only two tools evaluated in all studies. There are also many tools that are only evaluated in one study. Thereby it is very difficult to compare the studies' results.The crawling feature in [4] has been checked in 12tools. For the sake of clarity these 12 tools have been chosen for the comparison.

	[1]	[2]	[3]	[4]	[5]
Acunetix (Free or Commercial Editions)	X	X	X	X	X
AppScan	X	X	X	X	X
WebInspect	X	X	X	X	-
Hailstorm	X	X	X	-	-
N-Stalker	X	X	-	-	-
Burp Suite (Free or Professional Edition)	X	-	X	X	X
NTOSpider	X	-	X	-	X
Paros, Andiparos, Zaproxy or Milescan	X	-	-	X	X
W3AF	X	-	-	X	X
Grendel Scan	X	-	-	X	-
Qualys	-	X	X	X	X
McAfee SECURE	-	X	-	-	-
Rapid7 NeXpose	-	X	-	-	-
Arachni	-	-	-	X	-
JSky (Commercial Edition)	-	-	-	X	-
Netsparker (Commercial or Community Edition)	-	-	-	X	-
ProxyStrike	-	-	-	X	-
Core Impact	-	-	-	-	X
Skipfish	-	-	-	-	X

Table 1. Tools evaluated in studies

Table 2 shows the kind of web application that is tested in the studies. No tool uses the same web applications as another, so no conclusions can be drawn about the kind of web application. Some studies use custom web applications build ad-hoc to the evaluation and others use known web applications containing vulnerabilities.

	[1]	[2]	[3]	[4]	[5]
Custom web application	X	X			X
Web applications containing known vulnerabilities		X			
Vendor's test sites			X		
[7]				X	

Table 2. Web applications used in studies

3.2 Crawling and vulnerabilities

From tools and web applications, studies follow the two phases of vulnerability analysis: crawling and dynamic phase.The findings about crawling vary from crawling challenges, to a percentage of web pages detected in the web application.

In [1] two major challenges about crawling are multi-step process and the execution of client-side code. In [2] and [3] the major challenge about crawling is also to properly execute the client-side code. In [5] there is an indication that these tools have to improve.

Percentages speaking two studies give its results. In [2] the average successful links crawled over total existent links in the applications tested is 63.06%. In [4] the value is 44.1%.From the conclusion about crawling of these studieswe can not get a real value of the successful of web vulnerability scanners when crawling, but it clearly needs improving.Table 3 shows the vulnerabilities tested in the studies. Overall twentyseven vulnerabilities are tested, but only the two most popular vulnerabilities are tested in all evaluations: Cross Side Scripting and SQL Injection.

	[1]	[2]	[3]	[4]	[5]
SQL Injection	X	X	X	X	X
Cross Site Scripting	X	X	X	X	X
Command Injection	X	X	X	X	-
File upload	X	X	X	X	-
Path Traversal	X	X	-	X	X
Weak Session Identifier	X	X	-	X	-
Privilege escalation	X	-	-	X	X
Logic Flaw	X	-	-	-	-
Parameter Manipulation	X	-	-	-	-
Weak password	X	-	-	-	-
Cross Site Request Forgery	-	X	X	X	X
Other injections	-	X	X	X	-
Error Message Disclosure	-	X	X	-	-
Authentication Bypass	-	X	X	-	-
Session Fixation	-	X	-	X	-
Source Code Disclosure	-	X	-	X	-
SSL Misconfiguration	-	X	-	X	-
Insecure HTTP Methods	-	X	-	-	-
Insecure Temp File	-	X	-	-	-
HTTP Response Splitting	-	-	X	X	-
SOAP/AJAX Attacks	-	-	X	-	-
Application Denial of Service	-	-	-	X	-
Backup Files	-	-	-	X	-
JSON Hijacking	-	-	-	X	-
Open Redirect	-	-	-	X	-
Xml External Entity	-	-	-	X	-

Table 3. List of vulnerabilities tested

Comparing the results of the studies, as can be seen in Table 4, it is not clear which tool is more accurate. There is no tool in the top 3 in all studies. Besides, there are tools in top positions of a study but in low position of others. [2] Doesn't appear because it doesn't include tool names in its results. If the study evaluates two or more editions of the same product, only the best valuate edition is included in the table. If the study tests the tools twice, point and shoots and trained, first result is taken.

Position	[1]	[3]	[4]	[5]
1	Acunetix (Free or Commercial Editions)	NTOSpider	AppScan	AppScan
2	Webinspect	AppScan	WebInspect	Skipfish
3	Burp Suite (Free or Professional Edition)	Acunetix (Free or Commercial Editions)	Acunetix (Free or Commercial Editions)	Burp Suite (Free or Professional Edition)
4	N-Stalker	Hailstorm	W3AF	Qualys
5	AppScan	WebInspect	Burp Suite (Free or Professional Edition)	NTOSpider
6	w3af	Qualys	Qualys	Acunetix (Free or Commercial Editions)
7	Paros, Andiparos, Zaproxy or Milescan	Burp Suite (Free or Professional Edition)	Arachni	Core Impact
8	Hailstorm		JSky (Commercial Edition)	Paros, Andiparos, Zaproxy or Milescan
9	NTOSpider		Netsparker (Commercial or Community Edition)	W3AF
10	Grendel-Scan		Paros, Andiparos, Zaproxy or Milescan	
11			Grendel Scan	
12			ProxyStrike	

Table 4. Analysis of web vulnerability scanner

3.3 Web vulnerabilities list

In previous subsections five studies of web vulnerability scanner have been analysed. The studies can not been compared because everyone analyses different vulnerabilities in various web applications without the same tools.

To solve part of the problem a list of web application vulnerabilities like [6] or [12] could be used. This way a vulnerable web application could be developed that allow testing web vulnerability scanners always with the same set of vulnerabilities.

4 CRAWLING WEB FORMS IN DYNAMIC ANALYSIS

One of the major challenges when properly crawlingweb applications is to follow multi-step process. This occurs when a web application process need more than one step to execute an action. For example to complete a check out process the user fills out two or more forms in different web pages. This section explains how web forms works, and how web vulnerability scanners try to crawl web applications.

4.1 Web forms

A web form is a form on a web page that allows a user to enter or select data. This data is then sent to the Web server for processing. Web form fields can mainly be of two types: selection or text. Based on its properties they can also be hidden, mandatory, and dependent on the input value of another field. Web forms also contain at least one button to submit the data to the web server.

A processor running on the web server processes the form data and sends the response to the web browser. This response will be different depending on whether the values sent are valid or invalid. If the submitted values are valid the web server will return a web page with different content than the

previous page. If the values are invalid the web server will return a web page with the same form waiting for valid values, or an error message.

4.2 Crawling web forms

The web vulnerabilities scanners typically incorporate some options to specify, prior to or during the crawling, which values or types of values should be entered in form fields. The processing of the form on the web server can involve queries or changes on database records.

Actual solutions for filling out web form automatically can be grouped in four categories(a) crawl the web application collecting links, and introducing default field values in the web forms, (b) collect the values entered by a user while browsing the web application for later use in the crawling phase, (c) a hybrid approach using usersupplied values, in addition to collecting values of selection fields, and (d) keeping a repository of issues, descriptors and possible input values, also initialized with the values of the selection fields.

Both commercial and open source web vulnerability tools usually include (a) and (b) of the previous paragraph. These tools also used to include some improvements. For example some of them allow defining input values by wild-card characters or by its type. Other tools allow assigning multiple values to the same field name, and ordering then by its priority, or by web application.In (b) the tool can record the user's browsing prior to the vulnerability scan or can prompt for the values when it finds a web form.A particular case of web forms are the login forms, where the user generally enters an identifier and a password that the web application must validate.

Table 5 shows several characteristics of some commonly used scanners, as regards the automatic form filling.

	[8]	[9]	[10]	[11]
Save user's browsing	X	-	X	X
Ask for data	-	X	-	-
Default value	X	X	X	X
Define values	X	X	X	-
Values by web URL	X	-	X	-
Define field	X	X	X	-
Define value	X	-	-	-
Login	X	-	X	-

Table 5. Crawling features

In table 5 each row is:

- Save user's browsing: the tool can record the user's browsing and use the values used by the user in the vulnerability scan (active phase) later on.
- Ask for data: the tool can be configured so that it will ask for the values of form fields while crawling.
- Default value: The tool has a default value to fill input field when no other value is found.
- Define values: The user can configure the tool with values associated with fields that will be used later on.
- Values by URL: the tool allows defining specific web field values for each different web application.
- Define field: the tool allows using "wildcards" for the name of the fields.
- Define value: the tool allows to use "wildcards" for the field values
- Login: the tool can record a login macro to use in the scan.

The main drawback of this kind of procedure for filling out forms in crawling is that it relies heavily on the user. He or she has to browse through the web application entirely and enter valid input

values in the web form fields. An additional drawback is that this tool generally does not embody a method to determine if the values entered in the web form fields have been successful or not, beyond detecting HTTP error code 404 (Not Found).

5 CONCLUSIONS AND FUTURE WORK

This paper presented the comparison of five studies about web vulnerability scanners. The results of the comparison clearly show that is necessary to define a list of vulnerabilities to test web vulnerability scanners.A vulnerable web application that includes vulnerabilities should subsequently be developed.There are already web vulnerability lists that can be the beginning point to fulfil this requirement.

We also describe how web vulnerability scannerfill out web form fields to crawl the web application. Future work should focus onimproving the crawling capabilities of web vulnerability scanners, in order to reach all the web applications' content and face web crawling challenges. This new method should have two features: it has to propose a candidate value for each form field, and it has to execute client-side code.

ACKNOWLEDGMENTS

This work was supported by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID, Spain) through Acción Integrada MAEC-AECID MEDITERRÁNEO A1/037528/11.

References

[1] Doupe, A., Cova, M. and Vigna, G., Why Johnny can not pentest: an analysis of black-box vulnerability scanners web, In Proceedings of the 17th international conference on Detection of intrusions and malware, and vulnerability assessment, 111-131 (2010)

[2] Bau, J., Bursztejn E., Gupta, D., and Mitchell, J., State of the Art: Automated Black-Box Testing Web Application Vulnerability, In Proceedings of the 2010 IEEE Symposium on Security and Privacy, 332-345 (2010)

[3] Analyzing the Accuracy and Time Costs of Web Application Security Scanners

[4] http://www.sectoolmarket.com

[5] Effectiveness of Automated Application Penetration Testing Tools

[6] OWASP Testing Guide v3

[7] The WIVET (Web Input Vector Extractor Teaser) Test Score of Web Application Scanners

[8] Acunetix: http://www.acunetix.com

[9] Burp Suite: http://portswigger.net/burp

[10] HPWeb Inspect: https://www.fortify.com/products/web inspect.html

[11] Zaproxy: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[12] Black, PE, Fong, E., Okun, V. and Gaucher, R., Software Assurance Tools: Web Application Security Scanner functional Specification Version 1.0 U.S, Department of Commerce National Institute of Standards and Technology (2008)