International Journal of Academic Research and Development
ISSN: 2455-4197
Impact Factor: RJIF 5.22
www.academicsjournal.com
Volume 3; Issue 2; March 2018; Page No. 124-131

Relational revision of web application security testing tools

Manimekalai M, Jayanthi Vagini K
Department of Master of Computer Science, AJK College of Arts & Science, Coimbatore, Tamil Nadu, India

Abstract
Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product. Meets the business and technical requirements that guided its design and development, Works as expected and can be implemented with the same characteristic. Automated software testing utilizes different tools to execute testing activities. In this paper, I have discussed the features of Security Testing Tools: Wapiti, OWASP ZAP and Netsparker. In brief, I have presented a detailed description focusing on multiple feature Authentication, Control and Connection Features, Coverage Features, Input Vector Support, Audit Features, Complimentary Audit Features, The SQL Injection Detection Accuracy of the Scanner, The Reflected XSS Detection Accuracy of the Scanner, The Path Traversal / Local File Inclusion Detection Accuracy of the Scanner, The Remote File Inclusion Detection Accuracy of the Scanner, The WIVET Score of the Scanner. Finally, this research allowed me to draw some solid differences between Security Testing Tools by having real -world experience of testing effectively.

Keywords: wapiti, owasp zap, netsparker, vulnerability scanning

1. Introduction: Web Application

A web application is a computer program that utilizes web browsers and web technology to perform tasks over the Internet.

Web applications use a combination of server-side scripts (PHP and ASP) to handle the storage and retrieval of the information, and client-side scripts (JavaScript and HTML) to present information to users. This allows users to interact with the company using online forms, content management systems, shopping carts and more. In addition, the applications allow employees to create documents, share information, collaborate on projects, and work on common documents regardless of location or device.

How a web application works

Web applications are usually coded in browser-supported language such as JavaScript and HTML as these languages rely on the browser to render the program executable. Some of the applications are dynamic, requiring server-side processing. Others are completely static with no processing required at the server.

The web application requires a web server to manage requests from the client, an application server to perform the tasks requested, and, sometimes, a database to store the information. Application server technology ranges from ASP.NET, ASP and ColdFusion, to PHP and JSP.

Here's what a typical web application flow looks like:

1. User triggers a request to the web server over the Internet, either through a web browser or the application's user interface
2. Web server forwards this request to the appropriate web application server
3. Web application server performs the requested task – such as querying the database or processing the data – then generates the results of the requested data
4. Web application server sends results to the web server with the requested information or processed data
5. Web server responds back to the client with the requested information that then appears on the user's display

Benefits of a web application

- Web applications run on multiple platforms regardless of OS or device as long as the browser is compatible
- All users access the same version, eliminating any compatibility issues
- They are not installed on the hard drive, thus eliminating space limitations
- They reduce software piracy in subscription-based web applications (i.e. SaaS)
- They reduce costs for both the business and end user as there is less support and maintenance required by the business and lower requirements for the end user's computer

What is Security?

Security is set of measures to protect an application against unforeseen actions that cause it to stop functioning or being exploited. Unforeseen actions can be either intentional or unintentional.

What is Security testing?

Security Testing is a variant of Software Testing which ensures, that system and applications in an organization, are free from any loopholes that may cause a big loss. Security testing of any system is about finding all possible loopholes and weaknesses of the system which might result into a loss of

information at the hands of the employees or outsiders of the Organization.

The goal of security testing is to identify the threats in the system and measure its potential vulnerabilities. It also helps in detecting all possible security risks in the system and help developers in fixing these problems through coding.

Types of security testing

There are seven main types of security testing as per Open Source Security Testing methodology manual. They are explained as follows:

- **Vulnerability Scanning:** This is done through automated software to scan a system against known vulnerability signatures.
- **Security Scanning:** It involves identifying network and system weaknesses, and later provides solutions for reducing these risks. This scanning can be performed for both Manual and Automated scanning.
- **Penetration testing:** This kind of testing simulates an attack from a malicious hacker. This testing involves analysis of a particular system to check for potential vulnerabilities to an external hacking attempt.
- **Risk Assessment:** This testing involves analysis of security risks observed in the organization. Risks are classified as Low, Medium and High. This testing recommends controls and measures to reduce the risk.
- **Security Auditing:** This is an internal inspection of Applications and Operating systems for security flaws. Audit can also be done via line by line inspection of code
- **Ethical hacking:** It's hacking an Organization Software system. Unlike malicious hackers, who steal for their own gains, the intent is to expose security flaws in the system.
- **Posture Assessment:** This combines Security scanning, Ethical Hacking and Risk Assessments to show an overall security posture of an organization.

Type style and fonts

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

2. Ease of use

Related works

Security functional testing ensures that the software security functions are implemented correctly and are consistent with security requirements based on their specifications. Software security requirements mainly include data confidentiality, integrity, availability, authentication, authorisation, access control, audit, privacy protection, security management, etc.

Fig 1

Security vulnerability testing is to discover security vulnerabilities as an attacker. Vulnerability refers to the flaws in system design, implementation, operation or management. It may be used to attack, resulting in a state of insecurity. Commonly found vulnerabilities in web applications include cross-site scripting, injection, security misconfiguration, session management and more.

Vulnerabilities of web applications may be accessed by using penetration testing. Source code analysis systems aim to help developers locate vulnerabilities in the underlying code of software programs and applications before they are put into production. Penetration testing is the practice of testing a computer system, network or web application to find vulnerabilities that an attacker could exploit.

Common web application vulnerabilities the common vulnerabilities of web applications are cross-site scripting, SQL injection, broken authentication, cross-site request forgery and session management. These vulnerabilities appear significantly in the Web Hacking Incident Database (WHID).

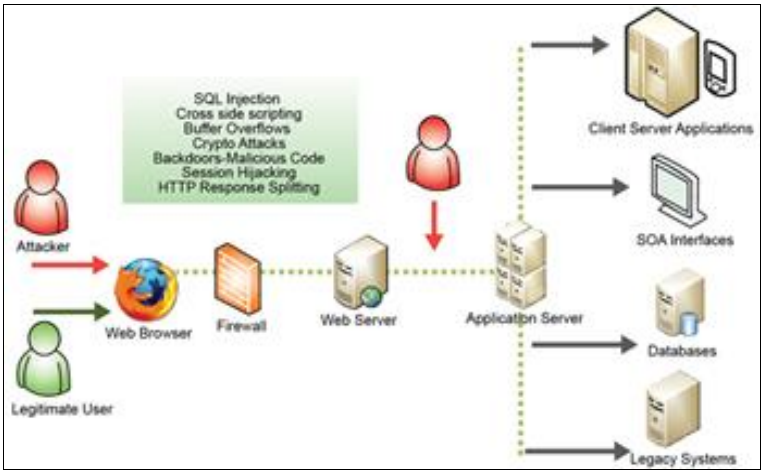


Fig 2: Security testing

SQL injection. It is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution. The best way of finding whether an application is vulnerable to attack is to ensure that all use of interpreters separates untrusted data from query. A rare and best way of attack is to inject malicious code into strings as metadata. Consequently, when the stored string is concatenated into a dynamic SQL query, the malicious code is executed. This can allow the tester to read and modify sensitive data contained in a database and take full control of the database server by exploiting vulnerabilities. Two attack scenarios are:

Scenario #1. The application uses untrusted data in the following vulnerable query:

String query = "SELECT * FROM customers WHERE custNAME='" + request.getParameter("name") + "'";

Scenario #2: Vulnerable query in Hibernate Query Language

Query HQLQuery = session.createQuery("FROM customer WHERE custNAME='" + request.getParameter("name") + "'");

In both scenarios, the attacker modifies the 'name' parameter value to send: ' or '1'='1.

http://testexample.com/app/customerView?name=' or '1'='1

This changes the meaning of queries to return all the records from the customer table. More dangerous attacks could invoke stored procedures or modify data.

Broken authentication and session management. It includes all form of handling user authentication and management of active sessions. A site may be vulnerable if: 1. User authentication credentials aren't protected when stored using hashing or encryption 2. Session IDs are exposed in the URL 3. Session IDs aren't rotated after successful login and passwords 4. Session IDs and other credentials are sent over unencrypted connections

To impersonate a user, an attacker use leaks or flaws, such as exposed session IDs, accounts and passwords, in the authentication or session management. When attack is successful, an attacker may have the same privilege level of the system as that of a legal authorised user. An attack scenario is given below as an example:

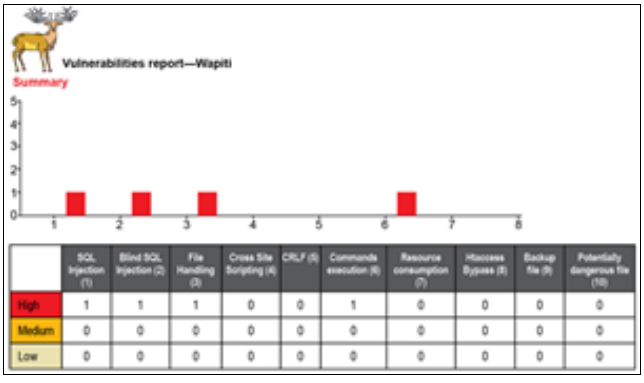
Scenario #1: Ticket reservations application supports URL rewriting:

http://Testexample.com/sale/itemsjsessionid=2P0OC2JSNDL PSKHCJUN2JV?destn =Delhi

An authenticated user of the site mails the above link to his friends without knowing that he is also giving away his session ID. When his friends use this link, they will use his session and credit card information.

Cross site scripting. It is a web vulnerability in which malicious script is injected into the system in the form of input. Three major types of cross-site scripting exist. The first one is Reflected XSS, in which attacker injects browser-executable code within a single HTTP response. It is non-persistent. Second type is Stored XSS, which occurs when some malicious user-submitted data is stored in a database to be used in the creation of pages that will be served to other users later. Thus, visitors of the web page fall victim to this attack. The third, Local XSS, targets vulnerabilities that occur in the source code itself. It is a type of XSS vulnerability which does not originate from the local software of the user.

Wapiti. Wapiti is a web application vulnerability scanner that was created in 2006 by Nicolas Surribus. It does not scan the source code of the application but scans the web pages of the launched web application. Wapiti acts like a fuzzer and injects payloads to see if a script is vulnerable. Wapiti can detect lots of vulnerabilities, such as file-handling errors, database injection, cross-site scripting, LDAP injection and CRLF injection. Wapiti is easy to use, open source and the user does not need security knowledge. But it is not able to find all the vulnerabilities.



	SQL Injection (1)	Blind SQL Injection (2)	File Handling (3)	Cross Site Scripting (4)	CRLF (5)	Commands execution (6)	Resource consumption (7)	Htaccess Bypass (8)	Backup file (9)	Potentially dangerous file (10)
High	1	1	1	0	0	1	0	0	0	0
Medium	0	0	0	0	0	0	0	0	0	0
Low	0	0	0	0	0	0	0	0	0	0

Fig 3: Wapiti test results

OWASP ZAP. OWASP ZAP is a penetration-testing tool which comes with plenty of features. Its main features is active scanning which is used to find certain kind of vulnerabilities, including XSS and others, except some logical vulnerabilities that can never be found by any automated security testing tool. Another interesting feature of ZAP is fuzzing. ZAP provides a list of fuzzers with the help of which we can fuzz any part of the application. This tool is open source, easy to use, well organised and up-to-date. High false positive factors and zero support for multiple scanning profiles are its main disadvantages.

Netsparker Community Edition. Netsparker is a free web application security scanner. It helps the developers who want to scan their web applications instantly and find the vulnerabilities. Netsparker Community Edition scans for SQL injection, XSS, Boolean SQL injection, backup files and static tests. It scans all types of web applications without limiting itself to technology or platform they are built on. It supports Javascripts/AJAX, can show impact of vulnerabilities or remedy for that vulnerability.

Tools results and analysis The web application was tested using the three tools OWASP ZAP, Wapiti and Netsparker Community Edition. The test results obtained using these tools were:

Testing results with Wapiti. Using Wapiti, the following four vulnerabilities were discovered in the Jammu & Kashmir application (the number in brackets refers to the number of occurrence of the vulnerability):

SQL injection (1): SQL injection is a technique that exploits a vulnerability occurring in the database of an application. It is a code injection that can allow the tester to read sensitive information from database and to modify it. SQL Injection occurs when input contains escaped characters, or user input is not strongly typed.

Blind SQL injection (1): Blind injection occurs when SQL injection is already present in the application but its result is not visible. Escaped characters should be handled properly to protect an application from SQL injection.

File handling (1): The attack is known as path traversal or directory traversal. Its aim is to access files which are stored outside the web root folder. The attacker tries to explore the directories stored in web server.

Command execution (1): This attack consists of executing system commands on the server. The attacker tries to inject this command in the request parameter.

Test results with OWASP ZAP Using the active scanning feature of OWASP ZAP, 17 potential vulnerabilities were found, as mentioned below (the number following the vulnerability refers to the number of occurrence of the vulnerability):

Cross site request forgery (2). Also known as one-click attack or session riding, and abbreviated as CSRF or XSRF, it is a type of malicious exploit of a website whereby unauthorised commands are transmitted from a user that the website trusts.

Cookie set without HttpOnly flag (1). If a cookie has been set without the HttpOnly flag, it means the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.

X-Content Type Options header missing (7). The script and styleSheet elements will reject responses with incorrect MIME MIME-type confusion based attacks.

Testing results with Netsparker Using the scanner feature of Netsparker Community Edition, we discovered five potential vulnerabilities in the example site:

X-Frame-Options header not set (7). In this case, X-Frame-Options header is not included in the HTTP response to protect against ClickJacking attacks. The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a < frame >

Cookie not marked as HTTPOnly. If a cookie, which is not marked HTTPOnly, sent by the application, is manipulated by the client side code (JavaScript, Java, etc), it could leave the site to XSS vulnerabilities. HTTP Only cookies cannot be read by client-side script, so marking a cookie as HTTP Only can provide additional layer of protection against types if the server sends the response header 'X-Content-Type- or< iframe >. Sites can use this to avoid clickJacking attacks by ensuring that their content is not embedded into other sites. Password transmitted over HTTP. When a password is transmitted over HTTP, it is vulnerable in many ways. As the password is transmitted over HTTP, network analyser and packet sniffers can easily monitor the traffic and intercept password. HTTP is not considered to be a secure method of user authentication as stated by HTTP specification.

Options: nosniff'. This is a security feature which prevents XSS attacks.

Internal server error. The server responded with an HTTP status 500. This is due to server-side error. Reasons may vary.

Version disclosure. This information can help an attacker gain a greater understanding of the systems in use and potentially develop further attacks targeted at the specific version of application.

3. Evaluation Study

The growing number of web applications combined with an ever growing Internet user mass, highlights the importance of developing high quality products. However, many attributes of quality web based system such as End-to-End Coverage & Methodology, Customization, Issue Workflow, Reporting, Requirement Management, Test case management, Test Set and Execution Management, Issue management are not given due consideration during development. Therefore, proper testing of web based system is needed in ensuring consistency, robust and high performing operation of web applications. Keeping in view the above mentioned features, we have selected 3 Security Testing Tools: Wapiti, OWASP ZAP and Netsparker.

General Features

Table 1

es	Wapiti	Netsparker.	OWASP ZAP
Custom Cookie	✓	✓	✗
Custom Header	✗	✓	✗
BASIC	✓	✓	✗
DIGEST	✓	✓	✗
NTLM	✓	✓	✗
NTLMv2	✗	✓	✗
KERBEROS	✓	✓	✗
FORM	✗	✓	✓
PROXY	✓	✓	✓
GZIP	✗	✓	✗
DEFLATE	✗	✓	✗
SSL	✓	✓	✓
CERT	✗	✓	✗
Logout Detection	✗	✓	✗
Exclude Logout	✓	✓	✗
Exclude URL	✓	✓	✓
Exclude Param	✓	✓	✓

Coverage Features

Table 2

features	Wapiti	Netsparker.	OWASP ZAP
COUNT	2	9	4
Manual Crawl	✗	✓	✓
URL File	✗	✓	✓
Html Crawler	✓	✓	✓
Ajax Crawler	✗	✓	✗
Flash Crawler	✓	✗	✗
Applet Crawler	✗	✗	✗
Silverlight Crawler	✗	✗	✗
WSDL Crawler	✗	✓	✗
REST Crawler	✗	✗	✗
Field Autofill	✗	✓	✗
Smart Autofill	✗	✗	✗

Anti CSRF Support	✗	✓	✓
Viewstate Support	✗	✓	✗
CAPTCHA Bypass	✗	✓	✗
WAF Bypass	✗	✗	✗

Input vector support

Table 3

features	Wapiti	Netsparker.	OWASP ZAP
COUNT	3	16	13
Get	✓	✓	✓
Post	✓	✓	✓
Cookie	✗	✓	✓
Header	✗	✓	✓
Secret	✗	✓	✓
PName	✗	✓	✓
XML	✗	✓	✓
XML ATT	✗	✓	✓
XML TAG	✗	✗	✗
JSON	✗	✓	✓
NetENC	✗	✗	✗
AMF	✗	✗	✗
Java SER	✗	✗	✗
Net SER	✗	✗	✗
WCF	✗	✗	✗
WCF-BIN	✗	✗	✗
Web sock	✗	✗	✗
DWR	✗	✗	✗
Custom	✗	✗	✓

Audit Features

Table 4

features	Wapiti	Netsparker.	OWASP ZAP
Count	15	18	17
SQL i	✓	✓	✓
BSQL i	✓	✓	✓
SSJS i	✗	✗	✗

RXSS	✓	✓	✓
PXSS	✓	✓	✓
DXSS	✗	✓	✓
JSON h	✗	✗	✗
LFI	✓	✓	✓
RFI	✓	✓	✓
CMDExec	✓	✓	✓
UPLOAD	✓	✓	✗
REDIRECT	✗	✓	✓
CRLFi	✓	✓	✓
LDAPi	✓	✗	✓
XPAPHi	✓	✗	✓
MXi	✗	✗	✗
SSI	✗	✗	✓
FORMAT i	✗	✗	✗
CODE i	✗	✓	✓
XML i	✗	✓	✗
EL i	✗	✓	✓
BUFFER o	✗	✗	✗
INTEGER o	✗	✗	✗
CODE Disc	✗	✓	✗
BACKUP f	✓	✓	✗
PADDING	✗	✗	✗
AUTH b	✗	✗	✗
PRIV e	✗	✗	✗
XXE	✓	✓	✗
SESSION	✗	✗	✓
FIXATION	✓	✗	✓
CSRF	✗	✓	✗
ADos	✓	✗	✗

4. Conclusion

Our comparison shows that different tools show different results for the same web application. Wapiti found SQL injection and Blind SQL injection in the application. Several studies show that Wapiti has highest percentage of SQL injection detections. Netsparker does not have Blind SQL Injection module, but is prone to less false positive factors as compared to other tools. OWASP ZAP found cross-site request forgery vulnerability in the web application. OWASP ZAP is prone to high false positive factors as compared to the other tools. It reported SQLi vulnerability as cross-site scripting. So we find that no tool is perfect but each can help to find some basic vulnerability.

5. Acknowledgment

M. Mani mekalai received her M.Sc (Information Technology) from Bharathiar University Coimbatore in 2009, M.Phil from Sri Krishna Arts and Science College in 2014. She has presented many papers on National and International Conferences. She published her works on International and National Journals. At present she is working has an Assistant Professor in AJK College of Arts And Science, Coimbatore.

Her area of interest is Software Testing and Data Structure Algorithms.

K. Jayanthi Vagini received her MCA from Sri Bannari Amman Institute in 2013, M.Phil from Hindustan College of Arts And Science in 2015.she has presented many papers on National Conferences and published her works on International Journals. At present she is working has an Assistant Professor in AJK College of Arts And Science, Coimbatore. Her area of interest is Networking and Big Data.

6. References

1. Http://www.softwaretestinghelp.com/practical-implementation-of-manual-testing/
2. A comparative study of automated software. Testing tools. Nazia islam. Nazia islam, isna1301@stcloudstate.edu
3. Issn-2319-8354(e). 173 page. Comparative study of software. Testing tools on the basis of software. Testing methodologies.
4. A study on Various software Automation Testing Tools Neha bhateja Department of computer science Amity university gurgaon, Haryana, India
5. https://www.guru99.com/testing-tools.html#9
6. International Journal of Information and Computation Technology. ISSN 0974-2239, 2013; 3(7):711-716.© International Research Publications