Advances in Computer Science and Information Technology (ACSIT)
p-ISSN: 2393-9907; e-ISSN: 2393-9915; Volume 5, Issue 2; April-June, 2018, pp. 102-103

http://www.krishisanskriti.org/Publication.html

Web Application Security Testing Approaches

Aquil Ahmad Khan[1] and Mayank Jain[2]

[1,2]ICERT, New Delhi
E-mail: [1]akhan2786@gmail.com, [2]engineermayankjain@gmail.com,

Abstract—*Web applications go through rapid development phases with extremely short turnaround time, making it difficult to eliminate vulnerabilities. Security of the website deals with testing security of the confidential data and make the data remain confidential. This Paper proposed an approach to test the web application platform. The World Wide Web is capable of delivering a broad range of sophisticated applications. The approach of security testing is based on understanding of how the client and the server communicate using HTTP Protocol. Here we analyze the design mechanisms of Web application security assessment in order to identify poor coding practices that render web applications vulnerable to attacks such as SQL injection and cross-site scripting. SQL injection and XSS attack are very critical as attacker can get a vital information from the server database. The proposed methodology is to create a researchable test suite based on the collected user session with genetic heuristic. The main aim of this paper is to explain the security of the web application to generate a test case of the web application on the bases of user session. We describe the use of a number of software testing techniques including dynamic analysis, black-box testing, gray-box testing, white-box testing, behavior monitoring and suggest mechanisms for applying these techniques to Web applications.*

Keywords*: Web Application Testing, Security Assessment, Gray Box Testing, White Box Testing, Black-Box Testing.*

1. INTRODUCTION

Web application testing is a very expensive process in terms of time and resources due to the nature of web application. Testing, designing and generating test cases. The number of reported web application vulnerabilities is increasing dramatically all over the world is witnessing a rapid increase in the number of attacks on Web applications. Developer are becoming more adept at writing secure code and developing & distributing patches to counter. Using our proposed Topic Model, the knowledgebase selects the best injection patterns according to experiences learned through previous injection feedback, and then expands the knowledgebase as more pages are crawled. Both previous experiences and knowledge expansion contribute to the generation of better injection patterns. Besides, this paper presents a new approach to vulnerability analysis which incorporates advantages of static testing and dynamic analysis. This approach effectively utilizes the extended Web Application Security testing model.

The web application is considered as one of the distributed system, with a client and server or multi-tier architecture. Web Application always run in different environment such as different hardware, network connections, operating systems, Web servers, and Web browsers. It is able to generate software components at run time according to user inputs and server status.

Web application Security testing approaches: Security society actively develops automated approaches to finding security vulnerabilities. But the most efficient way of finding security vulnerabilities in web applications is Manual code review. This technique is very time-consuming and requires expert skills which is prone to overlooked errors. These approaches can be divided into two wide categories: black box testing and white box testing:

Black Box Testing: This testing methodology looks at what are the available inputs for an application and what the expected outputs are that should result from each input. Most black-box testing tools employ either coordinate based interaction with the applications graphical user interface (GUI) or image recognition. An example of a black-box system would be a search engine. You enter text that you want to search for in the search bar, press “Search” and results are returned to you. In such a case, you do not know or see the specific process that is being employed to obtain your search results, you simply see that you provide an input – a search term – and you receive an output – your search results. It is not concerned with the inner workings of the application, the process that the application undertakes to achieve a particular output.

White BoxTesting: This testing methodology looks into the subsystem of an application. Whereas black-box testing concerns itself exclusively with the inputs and outputs of an application, white-box testing enables you to see what is happening inside the application. Whitebox testing provides a degree of sophistication that is not available with black-box testing as the tester is able to refer to and interact with the objects that comprise an application rather than only having access to the user interface. An example of a white-box system would be in-circuit testing where someone is looking at the interconnections between each component and verifying that each internal connection is working properly. Another

example from a different field might be an auto-mechanic who looks at the inner-workings of a machine to ensure that all of the individual parts are working correctly to ensure the properly working of machine.

OWASP Top 10 Most Critical Web Application Security Risks: One of the awareness documents provided by the OWASP community is the ten most critical web application vulnerabilities. The latest version that received consensus by the start of this thesis is the OWASP top ten 2013. The list was developed through analysing vulnerabilities found in thousands of applications. The vulnerabilities are rated based on number of occurrences, exploitability, detectability and their impact. Through awareness documents like OWASP top ten, OWASP Application Testing guide, code review guide and development guide, they aim to provide the necessary information to build and maintain secure applications.

A1: Injection

A2: Broken Authentication

A3: Cross-Site Scripting (XSS)

A4: Insecure Direct Object References

A5: Security Misconfiguration

A6: Sensitive Data Exposure

A7: Missing Function Level Access Control

A8: Cross-Site Request Forgery (CSRF)

A9: Using Components with Known Vulnerabilities

A10: Unvalidated Redirects and Forwards

Injection Vulnerability: There are different types of Injections i.e. SQL, LDAP, XPath, XSLT, HTML, XML and OS command. These type of issues occur when an user input is sent to an interpreter as a part of a command or a query without encoding or validation. To avoid such type of vulnerabilities it is recommended to use safe APIs having strongly typed parameterized queries and object relational mapping (ORM) libraries, avoid providing detailed error messages and properly validate input data for length, type and syntax.

Cross-site scripting (XSS) vulnerability: Reflected XSS occurs when a page reflects user provided data back to user. Stored XSS occurs when hostile data could be stored in a file or a database and at later stage displayed back without filtering. To overcome this vulnerability, it is recommended that Output encoding should be implemented on both the client and server side. Server-Side input validation should be done and white-listing of incoming data should be performed.

2. CONCLUSION

The approach of this paper is to assessing Web application security was constructed from a software engineering approach. A large number of defacement, phishing incident reported due to web applications vulnerabilities is increasing dramatically. Most of them result from improper coding or none input validation by the web application. We also introduced a new approach to automatic penetration testing by leveraging it with knowledge from dynamic analysis. Web Application security support analysis of data flows through other data storage types or implemented by means of stored procedures & triggers and special attention to development of automatic crawling mechanisms will be given. Here automatic form filling is the main issue. Static analysis can be used to detect data flows from HTTP parameters to the database fields. The investigation should also be done on augmentation of the generated test suit to meet a full coverage from a structural analysis.

REFERENCES

[1] OWASP WebScarab Project. http://www.owasp.org/index.php/OWASP_WebScarab_Project FastCGI Home. http://www.fastcgi.com/

[2] Scripting) Cheat Sheet. http://ha.ckers.org/xss.html

[3] OWASP WSFuzzer Project. http://www.owasp.org/index.php/Category:OWASP_WSFuzzer _Project

[4] CVE - Common Vulnerabilities and Exposures. http://cve.mitre.org/

[5] Ricca, F., Tonella, P. "Analysis and Testing of Web Applications." In: Proceedings of the 23rd IEEE International Conference on Software Engineering (Toronto, Ontario, Canada, May 2001), 25 –34.

[6] Sanctum Inc. "Web Application Security Testing – AppScan 3.5." http://www.sanctuminc.com

[7] E. Hieatt and R. Mee, "Going Faster: Testing the Web Application," *IEEE Software*, Vol. 19, No. 2, 2002, pp. 60-65.

[8] D. C. Kung, C. H. Liu and P. Hsia, "An Object-Oriented Web Test Model for Testing Web Applications," *Proceedings of the* 1*st Asia-Pacific Conference on Web Applications*, New York, 2000, pp. 111-120.

[9] User Session-Based Test Case Generation and Optimization Using Genetic Algorithm* Zhongsheng Qian *J. Software Engineering & Applications*, 2010, 3, 541-547.

[10] Mod_python – Apache/Python Integration. http://modpython.org/

[11] SCGI: Simple Common Gateway Interface. http://python.ca/scgi/

[12] PEP 333 – Python Web Server Gateway Interface v1.0. http://www.python.org/dev/peps/pep-0333/

[13] The Django framework. http://www.djangoproject.com/

[14] Pylons Python Web Framework. http://pylonshq.com/

[15] CherryPy Framework. http://www.cherrypy.org/

[16] Spyce – Python Server Pages (PSP). http://spyce.sourceforge.net/XSS (Cross Site