Vulnerability Assessment of Web Applications using Hybrid Algorithm

Prof. Smita Krishna Patil, Adarsh Patil, Akhil Penta, Mayur Pardeshi, Rishikesh Salunkhe
Department of Information Technology,
Atharva college of engineering,
Mumbai, Maharashtra, India

***Abstract*—Web vulnerability scanners (WVS) are tools for discovering vulnerabilities in a web application.However, they are not 100% accurate. We intend to develop a hybrid algorithm for detecting web based applications vulnerabilities. The hybrid algorithm is based on the concept of carefully, combining desirable features of various components so that the new algorithm has the ability to discover more vulnerabilities. However the combination is not done blindly, it is based on various factors such as optimization and sophistication among others with an aim of increasing efficiency.**

Keywords—AppScan, BurpSuite, nikto, OWASP, penetration testing security, security assessment tools, threat modelling, vulnerability assessment, w3af, web server

I. INTRODUCTION

The number and importance of web applications have increased rapidly over the years, many organizations have embraced these technologies to explore new business opportunities and some companies have been forced to adopt the electronic commerce by their customers or competitors.

Web applications have gained popularity and have become part of our daily lives interaction. Web applications are present and always accessible on internet. But, attacker can abuse these web applications through vulnerabilities present in Application. Attacker can use several well known techniques using which they can penetrate the web applications. Security is not a one-time event [2]. It is insufficient to secure your code just once. A secure coding initiative must deal with all stages of a program's lifecycle. Therefore, manual code inspection or security audits must be done by highly trained experts, who are labour-intensive, expensive, and prone to errors. For this reason, there is need to automate vulnerability discovery. OWASP conducts security survey every year to identify the vulnerabilities that have caused attacks and the top ten vulnerabilities of 2010 are Injection, Cross-Site Scripting(XSS), Broken Authentication and Session Management, Insecure Direct Object References, Cross-Site Request forgery(CSRF), Security Misconfiguration, Insecure cryptographic Storage, Failure to Restrict URL Access, Insufficient Transport Layer Protection and Invalidated Redirects and Forwards[1,2].

II. LITERATURE REVIEW

A. Overview on Vulnerabilities

Detecting vulnerabilities is generally not an easy task, and not all of the common vulnerabilities can be successfully detected by automated scanners. As our digital infrastructure gets increasingly complex and interconnected the difficulty of achieving application security increases exponentially [1,2]. We can no longer afford to tolerate relatively simple security problems like those presented below.

a. Remote code execution
b. SQL injection
c. Cross Site Scripting (XSS)
d. Insecure Direct data Object
e. Security Misconfiguration
f. Sensitive Data Exposure
g. Missing Function Level Access Control

a. Remote code execution

Improper coding leads to this type of vulnerability. It allows attacker to run arbitrary system level code on server and can retrieve any desire information present at server. It is difficult to discover this vulnerability during penetration testing process but such problems are often revealed while doing a source code review [1,2].

b. SQL Injection

SQL injection is a very old approach but it's still popular among attackers. This technique allows an attacker to retrieve information from a Web server's database. Depending on the application's security measures, the impact of these type of attacks may vary from basic information disclosure to remote code execution and total system compromise [1].

c. Cross site scripting

The success of these type of attacks require the victim to execute a malicious URL or URLs which are crafted in such a manner to appear to be legitimate at first look. When visiting such a crafted URL, an attacker can effectively execute something malicious in the victim's browser. Some malicious scripts, for example, will be run in the context of the web site which possesses the XSS bug [2].

d. Insecure data object

Insecure data object comes into picture when web application shows or exposes some of the internal data to user. Open Redirects and Directory Traversal are two classic examples of insecure direct object reference vulnerability [3].One essential defense is to check access control. On each use of a direct object reference from an untreated source, the application should perform an access control check to ensure the user is authorized for the requested object or service.One essential defense is to check access control. On each use of a direct object reference from an untreated source, the application should perform an access control check to ensure the user is authorized for the requested object or service [4].

e. Security misconfiguration

It is equally important to have the software up to date. Anonymous external attacker as well as user with their own accounts may attempt to compromise the system. It can cause all of important data could be stolen or change with the time. If Directory listing is not disabled on the server and if attacker discovers the same then the attacker can simply list directories to find any file and execute it. It is also possible to get the actual code base which contains all your custom code and then to find a serious flaws in the application [5].

f. Sensitive data

Sensitive Data Exposure occurs when an application does not adequately protect sensitive information. The data can vary and anything from passwords, session tokens, credit card data to private health data and more can be exposed[7]. A few examples would be exposed data that someone mistakenly uploaded somewhere, weak crypto that means an attacker would be able to read the data if they successfully compromised the target and the lack of headers that prevent browser caching. In short, every possible way where it would have been possible to better protect the sensitive data [7].

g. Missing Function Level Access Control

This name suggests, user can spoof the URL to invoke the hidden or important information display function if they know how to do that, or they could view the HTML and JavaScript code of a page to see how to call the function. If users do this and they can get access to parts of the application that they shouldn't then this is a case of Missing Function Level Access Controls [9].

Table I. Vulnerabilities and Required Scanning Tool

Tool	Scanning	Vulnerability
Highly Critical	AppScan,	Remote Code Execution
Moderate to Highly Critical	Vega, AppScan, W3af, Nikto	SQL Injection
Less to Moderate Critical	Vega, W3af	XSS
Moderately Critical	AppScan, Nikto	Insecure Direct Data Object
Less to Moderate Critical	W3af, AppScan	Security Misconfiguration
Highly Critical	Nikto	Sensitive Data Exposure
Moderately Critical	AppScan,Vega	Missing Function Level Access Control

B. Vulnerability Scanning Tools

i. APP SCAN

IBM Security Appscan (AppScan) is an automated dynamic commercial security testing tool AppScan tests almost 2000 tests but not all tests are necessary. This report also gives a proper step to take on the vulnerabilities detected. AppScan comes in several editions to address the web application security testing needs throughout the Software Development Life Cycle [16]..

ii. Wikto (Nikto for Windows OS)

There are 3 main sections of the tool: Back-End miner, Nikto-like functionality and google [15]. Wikto is coded in C# and requires the .NET platform [14]. Nikto can crawl a website in the least amount of time. It uses a technique called mutation [11], whereby it creates combinations of various HTTP tests together to form an attack, based on the Web server configuration and the hosted code. Thus, it finds critical loopholes such as file upload misconfiguration, improper cookie handling, cross-scripting errors.

iii. W3af

This open source tool is widely used to scan websites, mainly because it supports HTTP and HTTPS, and also provides findings in an interactive fashion. w3af is divided into two main parts, the core and the plug-ins. The core coordinates the process and provides features that are consumed by the plug-ins, which find the vulnerabilities and exploit them. The plug-ins are connected and share information with each other using a knowledge base [15].

IV. WORKING

A hybrid algorithm is explained in below diagram. The main aim of the algorithm is to detection accuracy. The hybrid algorithm is derived from available algorithms with a aim is to scanning a vulnerability of web application in less time period. Although the accuracy may not be achieved 100%, an effort has been put to raise it above the available tools. The results of the tests will be standard with OWASP results which are updated on a regular basis. The hybrid algorithm is based on the concept of combining sensible features of various components so that the new algorithm can determine more vulnerability.

The scanning process involves crawling and parsing and the identification of the vulnerabilities and repeated until all the vulnerabilities have been discovered. Once this process is completed, the analysis is done and finally a report is displayed showing the discovered vulnerabilities discovered and their location.

The scanning process includes, crawling and fuzzing. After the scanning process is completed, the results are submitted for analysis and a report is displayed.

1. Inspection: This phase focuses on information about the web application. It acquiring various parameters available on the source code.
2. Scanning: Once the first phase is completed, the Scanning process begins, which involves, recognizing the weaknesses that exist in the web application. The more the details found on this phase, the more successful the entire scanning process will be.
3. Vulnerabilities: Vulnerabilities are found based on various tests included within the algorithm.
4. Analysis: Once the vulnerabilities are discovered they are analyzed in the next phase. The vulnerabilities will pass through various test cases.
5. Report: After the analysis report is displayed at the end of the process. The report generated will be in a proper format intended to better understanding and with complete details.

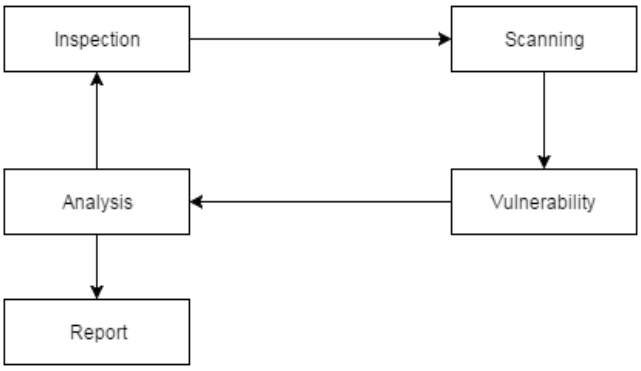

Fig 1. Scanning Web Application

V. EXPECTED OUTPUT

The output will be provided in a proficient manner so that the process of assessment of vulnerabilities and their removal can be streamlined for a penetration tester. The output will be based on various phases of identification, analysis and tests.

VI. CONCLUSION

This paper provides an approach for developing a web Application assessment tool using an hybrid algorithm which can address a wide range of vulnerabilities thereby trying to incorporate the benefits of various tools & their beneficiary available in the market.

REFERENCES

[1] Ashwani Garg, Shekhar Singh "A Review on Web Application Security Vulnerabilities" International Journal of Advanced Research in Computer Science and Software Engineering Research Paper, Issue 1, January 2013
[2] Dhanya Pramod, "A Study of Various Approaches to Assess and Provide Web based Application Security", International Journal of Innovation, Management and Technology, Vol. 2, No. 1, February, 2011
[3] Brett Hardin. July 22, 2009. Insecure Direct Object Reference. Retrieved from http://bretthard.in/post/insecure-direct-object-reference
[4] Hui Wang. (n.d.). Preventing Insecure Direct Object References In App Development Retrieved From http://www.cs.tufts.edu/ comp/ 116/ archive/fall2014/hwang.pdf
[5] Security Testing - Security Misconfiguration. (n.d.). Retrieved from https://www.tutorialspoint.com/security_testing/testing_security_misconfiguration.htm
[6] Security Misconfiguration (n.d.) Retrieved From https:// codedx. com/ security-misconfiguration/
[7] Linus Särud. 2016.07.01. OWASP TOP 10: Sensitive Data Exposure . Retrieved From https://blog.detectify.com/2016/07/01/owasp-top-10-sensitive-data-exposure-6/
[8] Josh Hamit. March 27, 2014. Top Ten Web Security Risks: Sensitive Data Exposure [web blog post] Retrieved from https://www. credera. com/blog/technology-insights/open-source- technology-insights/top-ten-web-security-risks-sensitive-data-exposure-6/
[9] Maurice McMullin. Dec 9 2015. OWASP Top Ten Series: Missing Function Level Access Control [web blog post]. Retrieved from https://kemptechnologies.com/blog/owasp-top-ten-series-missing function-level-access-control
[10] Josh Hamit. March 27, 2014. Top Ten Web Security Risks: Sensitive Data Exposure [weblog post]. Retrieved from https://wwwcredera com/ blog/technology-insights/open-sourcetechnology-in sights/top-ten-web-security-risks-sensitive-data-exposure- 6/
[11] http://opensourceforu.com/2010/05/website-vulnerabilities-and - nikto/
[12] http://resources.infosecinstitute.com/14-popular-web-application - vulnerability-scanners/
[13] http://www-03.ibm.com/software/products/en/appscan-standard
[14] https://github.com/sensepost/wikto
[15] https://www.security-database.com/toolswatch/+-wikto-+.html
[16] http://www.ibm.com/developerworks/library/se-appscan/