Web Application Vulnerability Scanner

[1]Eswar Kumar G, [2]Moses James S, [3]Parasu Raman M

Student
Cyber Forensics And Information Security
Dr. M.G.R. Educational and Research Institute,Chennai, India.

***Abstract* -The web application vulnerability scanner (WAVS) is a tool designed to identify security vulnerabilities within web applications. Its functionality involves scanning application configuration files and server infrastructure to detect potential weaknesses exploitable by attackers. The primary goal is to preemptively identify security issues before they're exploited by malicious hackers, targeting common vulnerabilities such as SQL injection, cross-site scripting (XSS) attacks, directory traversal, and information disclosure. WAVS features a command-line interface, enhancing accessibility for both security professionals and developers.**

INTRODUCTION

A web application vulnerability scanner is an automated tool designed to pinpoint security weaknesses in web applications. It streamlines the process of scanning and evaluating the security status of these applications, identifying potential vulnerabilities exploitable by attackers. These scanners operate by sending diverse HTTP requests and analyzing the responses from the web application. They typically employ a mix of black-box and gray-box testing techniques. Black-box testing involves assessing the application externally, akin to an attacker with no internal knowledge of the application. Conversely, gray-box testing entails having partial knowledge of the application's internals, such as access to the source code or database schema. It works like a digital detective, meticulously searching through the inner workings of web apps to uncover potential vulnerabilities that could be exploited by cyber attackers. These tools aren't just about finding problems; they're proactive guards. They strive to identify and flag vulnerabilities before hackers can exploit them, essentially acting as a shield against cyber threats. They explore the ins and outs of web apps, examining configurations and code, all with the goal of fortifying applications against possible breaches. Once vulnerabilities are discovered, these scanners don't stop there. They provide detailed reports and insights, guiding developers and security teams on how to fix these weaknesses effectively. Ultimately, they're crucial in ensuring that web applications meet security standards and regulations, serving as essential guardians in the ever-evolving realm of cyber threats.

OBJECTIVES

1. **Vulnerability Detection:** Identifying security vulnerabilities within web applications, such as SQL injection, cross-site scripting (XSS), or other common exploits.
2. **Risk Assessment:** Evaluating the severity and potential impact of discovered vulnerabilities to prioritize and address the most critical issues first.
3. **Prevention of Exploits:** Proactively identifying weaknesses before malicious actors can exploit them, thereby enhancing the application's security posture.
4. **Reporting:** Generating detailed reports outlining discovered vulnerabilities.

DESIGN AND IMPLEMENTATION

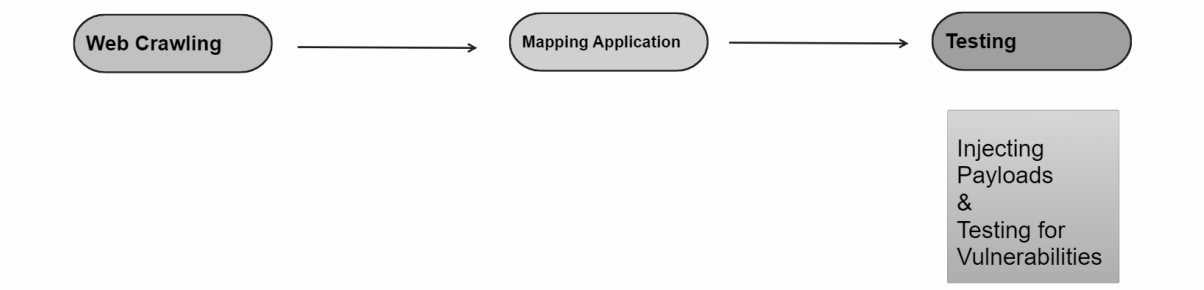

Figure:1 Represents the design of web application vulnerability scanner.

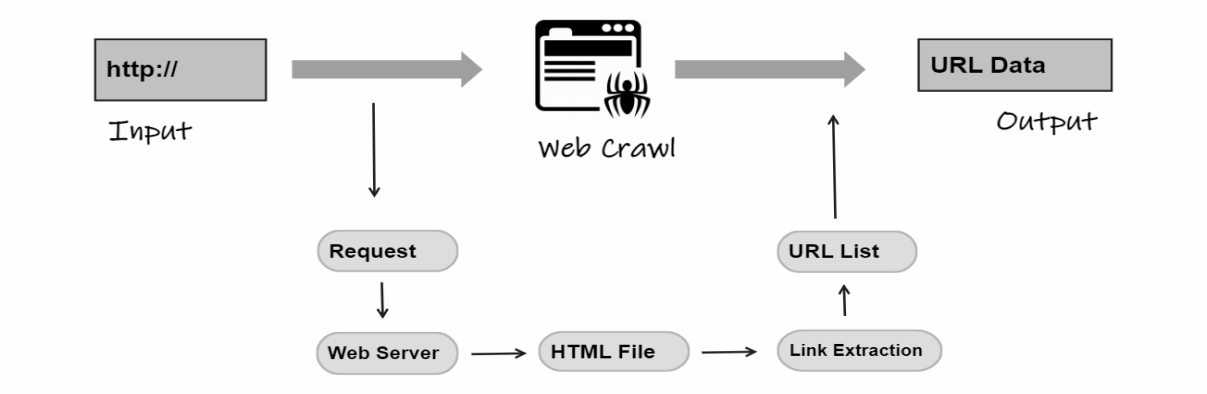

Figure:2 Represents the working of crawler.

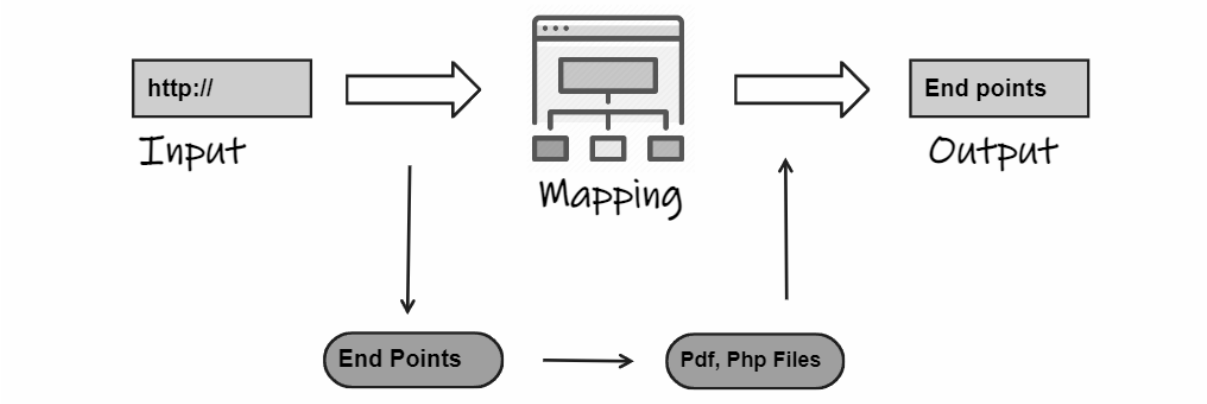

Figure:3 Represents the working of mapper.

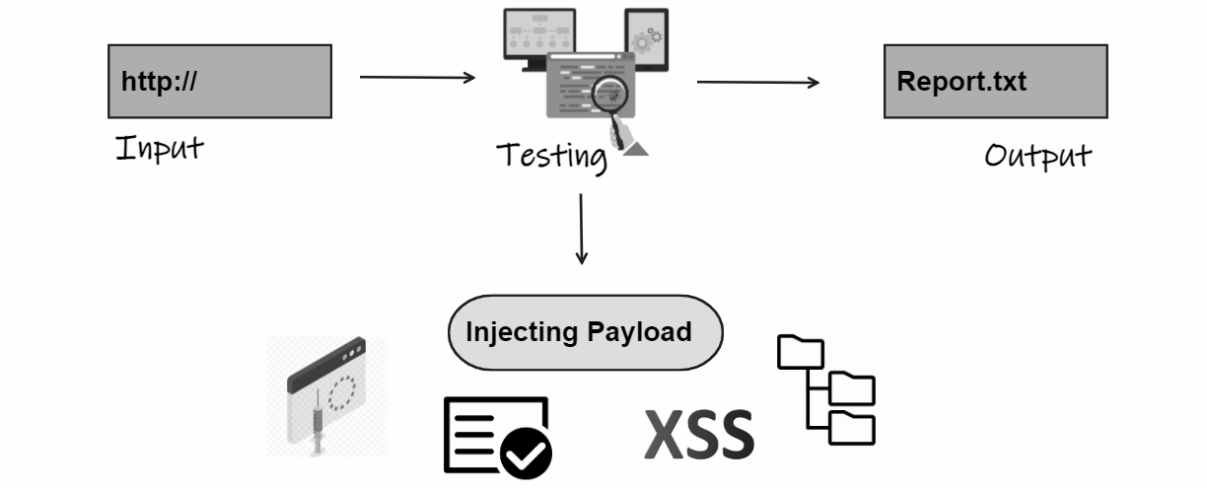

Figure:4 Represents the working of tester

.

METHODOLOGY

Crawler: Maps the web application structure by exploring various paths.
Mapper: Scrapes links and identifies files (pdf, doc, etc.).
Tester: Detects vulnerabilities by sending requests and analyzing responses.
Vulnerability Knowledge Base: Contains known vulnerabilities and attack patterns.
Reporting Module: Generates comprehensive vulnerability reports.

CONCLUSION

The web application vulnerability scanner represents a crucial step in automated security testing. While employing basic checks, diversifying and intensifying test payloads for each vulnerability type can significantly enhance effectiveness. Refining detection criteria is vital for result accuracy, while integrating functionality to handle form submissions via POST requests improves analysis. Restructuring reporting formats for clarity and utility and emphasizing responsible

testing practices, including permissions and ethical guidelines, are imperative. Continuous refinement holds promise for evolving this tool into a robust and trustworthy asset for web application security testing, prioritizing ethical methodologies.

REFERENCES:

[1] Kaur, N., & Kaur, P. (2014). Input Validation Vulnerabilities in Web Applications. Journal of Software Engineering, 8(3), 116-126.
[2] Austin, A., Holmgreen, C., & Williams, L. (2013). A comparison of the efficiency and effectiveness of vulnerability discovery techniques. Information and Software Technology,55(7), 1279-1288.
[3] Zhang, D., Liu, D., Csallner, C., Kung, D., & Lei, Y. (2014). A distributed framework for demand-driven software vulnerability detection. Journal of Systems and Software, 87, 60-73.
[4] Huang, C. C., Lin, F. Y., Lin, F. Y. S., & Sun, Y. S. (2013). A novel approach to evaluate software vulnerability prioritization. Journal of Systems and Software,86(11), 2822-2840.
[5] Lee, I., Jeong, S., Yeo, S., & Moon, J. (2012). A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling, 55(1), 58-68.
[6] Corona, I., Giacinto, G., & Roli, F. (2013). Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. Information Sciences, 239, 201-225.
[7] Balasundaram, I., & Ramaraj, E. (2012). An Efficient Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching.Procedia Engineering, 30, 183-190.
[8] Davanzo, G., Medvet, E., & Bartoli, A. (2011). Anomaly detection techniques for a web defacement monitoring service. Expert Systems with Applications,38(10), 12521-12530.
[9] Shar, L. K., & Tan, H. B. K. (2012). Automated removal of cross site scripting vulnerabilities in web applications. Information and Software Technology, 54(5).43
[10] Goseva-Popstojanova, K., Anastasovski, G., Dimitrijevikj, A., Pantev, R., & Miller, B.(2014). Characterization and classification of malicious Web traffic.Computers & Security,42, 92-115.
[11] Avancini, A., & Ceccato, M. (2013). Comparison and integration of genetic algorithms and dynamic symbolic execution for security testing of cross-site scripting vulnerabilities. Information and Software Technology, 55(12), 2209-2222.
[12] Jang, Y. S., & Choi, J. Y. (2014). Detecting SQL injection attacks using query result size. Computers & Security, 44, 104-118.
[13] Shahriar, H., Weldemariam, K., Zulkernine, M., & Lutellier, T.(2014). Effective detection of vulnerable and malicious browser extensions. Computers & Security, 47, 66-84.
[14] Scholte, T., Balzarotti, D., & Kirda, E. (2012). Have things changed now? An empirical study on input validation vulnerabilities in web applications. Computers & Security, 31(3), 344-356.
[15] Woo, S. W., Joh, H., Alhazmi, O. H., & Malaiya, Y. K. (2011).Modeling vulnerability discovery process in apache and iis http servers. Computers & Security, 30(1), 50-62.
[16] Shar, L. K., & Tan, H. B. K. (2013). Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns. Information and Software Technology, 55(10), 1767-1780.
[17] Wang, S., Gong, Y., Chen, G., Sun, Q., & Yang, F. (2013). Service vulnerability scanning based on service-oriented architecture in Web service environments.Journal of Systems Architecture, 59(9), 731-739.
[18] Awoleye, O. M., Ojuloge, B., & Ilori, M. O. (2014). Web application vulnerability assessment and policy direction towards a secure smart government. Government Information Quarterly, 31, S118-S125.