NFSU JOURNAL OF CYBER SECURITY & DIGITAL FORENSICS

NFSU – Journal of Cyber Security and Digital Forensics
Volume – 1, Issue – 1, December 2022
E – ISSN – 2583-7559

WEB APPLICATION PENTESTING APPROACH & REMEDIATION

Purva Bhesaniya, Animesh Kumar Agrawal

National Forensic Sciences University
Gandhinagar, India
purva15ub@gmail.com

National Forensic Sciences University
Gandhinagar, India
akag9906@gmail.com

***Abstract*— In this technophile world, the expansion in web apps is enormous, and breaches and unauthorized access of sensitive data from various platforms over the internet. Hackers concentrate on web-based applications like shopping carts. Web applications are hard to protect against security flaws known as web application vulnerabilities. Web application pen testing is fundamental to identifying existing vulnerabilities. Buffer overflows, XSS attacks, CSRF attacks, and SQL injections are all examples of these types of attacks. In other words, once new technologies are demanded by the globe, security testing could become a growing need. This paper aims to understand the testing techniques of web apps that penetrate and identify proper counter measurements by understanding the various vulnerabilities precisely. The OWASP top 10 vulnerabilities are studied in detail, and these vulnerabilities need to be addressed with precautions and used manual and automatic approaches.**

Keywords—Pentesting, OWASP Top 10 vulnerability

I. INTRODUCTION

The development of web applications, relying on various phases, possibly distributed over multiple platforms and service providers, sometimes raise problems, specifically in terms of security. Web application security is difficult to assess, and web apps play an essential role in our daily lives. In order to maintain users' trust, security measures must include confidentiality, integrity, and other safeguards. There, however, are some vulnerabilities that might allow attackers to damage users due to the fact that no system is completely safe. The erosion of trust among users has impacted a significant part of the web application industry model. Thus, it is essential to make sure that web pages are secure. Penetration testing of online apps is required to find/detect security flaws to preserve users' trust and data. It entails determining whether a web application is vulnerable to assaults using various tools and techniques often used by penetration testers. Manual and automated testing are the two main ways of this type of testing.

The manual approach is appropriate for complete app inspection since it detects problems and loopholes that automated tests bypass. In contrast, automated test results are faster, more efficient, more straightforward, and more trustworthy. It also automatically assess whether a machine is susceptible to risk and how it can be mitigated. This method does not require any prior knowledge. Nessus, Metasploit, and OpenVAs are tools for automated penetration testing. This document conducts manual and automated procedures research and mentions a few OWASP vulnerabilities and mitigation. It is usually preferable to analyze faster; an automated technique is a fantastic example of this, rather than a manual one. With the use of practical results and examples, discuss the comparison. Various examples are focused on to differentiate the perspectives of approaches. Specifically, Section II elaborates on previous studies concluded by experts on web application pen-testing and their discussion of actual attack scenarios. For understanding some OWSAP vulnerabilities, Section III utilizes automated and manual approaches with attack scenarios. Section IV explains the Mitigation strategies. According to our observations and research findings, Section V and VI summarizes our conclusions.

II. LITERATURE REVIEW

Web app pen-testing analyses by the researcher, the field within web security. Penetration testers eagerly try to look for vulnerabilities in websites and other platforms. Much study has gone into Detecting Web Application Vulnerabilities Using Penetration Testing and Threat Modelling, which explains how to use various penetration tools. This tool aids in the detection of Web application vulnerabilities and provides Web attack trees for a better understanding of attacks such as SQL injection and CSRF attacks[1]. An open-source application called Penetration Testing Analysis with Standardized Report Generation distinguishes between automated and manual testing processes. Aside from that, it uses a survey to highlight the lack of a defined report format [2]. An approach to detecting XSS vulnerabilities in web applications that employs both static and dynamic approaches, which include static, dynamic, safe programming, and modelling [3]. An automated penetration testing process maintains the security of a cloud application, as per Towards Automatic Security Research for Cloud Platforms [4]. This study paper will assist in comprehending the necessary countermeasures for OWSAP vulnerabilities [5]. This article helps understand

neural networks to create a decision support tool and clarify their advantages and disadvantages [6]. Using an IDS to analyze penetration testing tool traffic can help identify weaknesses in web applications and evaluate the importance of using Snort to analyze tools, which helps identify vulnerabilities [7]. GuruWS is a tool that analyses experiments and functional improvements to the GuruWS platform to identify web application vulnerabilities and help resolve them by providing a solution [8]. Various tools and techniques implemented in web apps describe the limitations of automated penetration testing technologies and recommend that manual testing complete the assessment [9]. This paper will focus on understanding the situation where the hacker can easily hack the users' sensitive information from the various web applications using OWSAP top 10 vulnerabilities, which would demonstrate. It will also discuss the two approaches to web application penetration testing.

III. Attack Scenarios

OWSAP's 2021 flaws describe using Manual and Automated techniques that a hacker might use to obtain access to the website—followed by a specific scenario that an attacker could utilize. The flaws of OWSAP discuss. There are numerous other ways that attackers try to harm websites. There is a discussion of Only the most common vulnerabilities.

A. Lab Setup

The following lab setup is required in a virtual environment to perform vulnerability scanning. It would understand a few OWSAP 2021 vulnerabilities to understand the difference between the two main penetration testing techniques. Fig 1 explains the methodology.

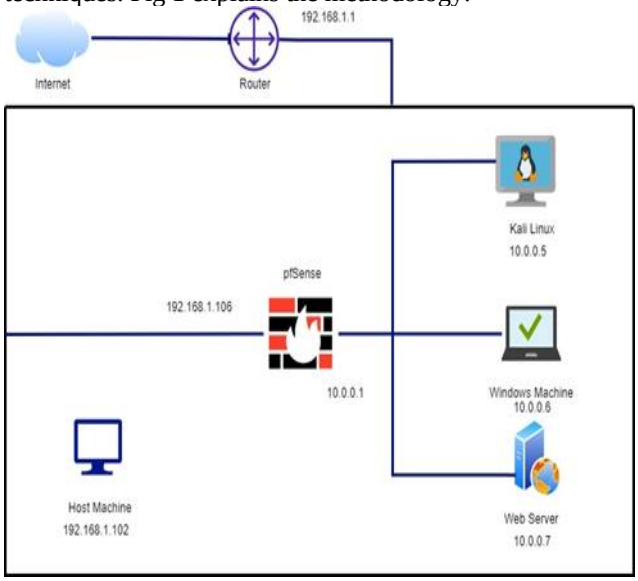

Fig. 1. Lab setup

B. Sample attack scenario

For this research paper, a website that would use download from the git-hub plat-form. After that, configure the website on windows server 2022 with the IIS and SQL database management plugins. The website can access Windows ten pro, a client machine, and Kali Linux is an attacker machine. Then use a PFsense Tool to isolate the virtual environment from the host machine and configure a private network. Following, it focuses on OWSAP 2021 vulnerabilities.

C. Injection Flows

A vulnerability known as an injection flaw allows untrusted data to be accessed and executed by a code or query contained in a web application. A malicious attack would construct malicious orders or queries to exploit injection flaws. In the worst scenario, data would be lost, or corrupted, accountability would be lacking, or users would be denied access.

D. Broken Access Control

The notion of Broken Access Control refers to the possibility of attackers accessing, modifying, deleting, or performing actions outside an application or system's intended permissions. As a consequence of Broken Access Control, users can change parameters in URLs, view or modify one's own data, or gain privileges if attacker change parameters in the URL.

E. Security Misconfiguration

In most cases, security misconfigurations occur at the platform, web server, application server, framework, and custom code levels within an application stack. An incorrectly configured server, for example, could cause various issues that could compromise the site's security.

F. Identification and Authentication Failures

Passwords can still be guessed by automated attacks such as credential stuffing and brute-force attacks. There are flaws in the process of resetting a password or recovering it. No handling of identifying session identifiers after email/password updates, logouts, inactivity, or logging.

IV. Countermeasures

This session addresses the mitigation and counter-measurement of web application vulnerabilities. There are a few counterarguments to take to secure the web application.

A. Web application firewalls

Web application firewalls are Software and hardware that help block traffic and monitor it.

B. Information gathering

Information gathering Support in examining and collecting third-party data and material to identify client-side codes and access points.

C. Authorization

Track traversals should be tested correctly in the web application, and allowed access should be blocked. It improves the prevention of insecure logins and the loss of sensitive information.

D. Cryptography

Encrypts specific data, checks for random flaws, and bypasses incorrect algorithms to ensure that all data transmissions are secure.

E. Denial of service

Using its anti-automation, HTTP protocol DoS, account lockout, and SQL wildcard DoS tools, applications can be made more resilient to denial-of-service attacks. Utilize

scalable resources and filtering solutions together to prevent DDoS and DoS attacks.

V. RESULTS

Developers of web applications are vigilant when it comes to mitigating common security flaws. Even with their best efforts, there will still be some vulnerabilities in the application. There are a number of mitigation techniques explained in the previous section. This section usually consists of sound tactics for preventing assaults, as long as developers use alerts when implementing. Vulnerabilities will still exist despite the adoption of those strategies. There is a link between the Several specific conditions and vulnerabilities listed here. This section consists of automatic and manual procedures using open-source software.

An overall view of the study is provided by combining web penetration testing with application testing. In addition, the paper provides mitigation options to overcome the vulnerabilities in the web applications.

A. Footprint of Web Infrastructure

It is a process of gathering complete information about the target web application, its related component, and how it works with the Nmap tool's help. this command Nmap -sS -O 192.168.1.104 && Sudo Nmap -sS -A 192.168.1.104

B. Auditing Web Application Framework Using Vega

Vega is a free, open-source, graphical web-auditing tool. It helps to identify XSS and SQL injection Vulnerabilities. Vega gives us summary alert vulnerability with High/ low/ medium on the website.

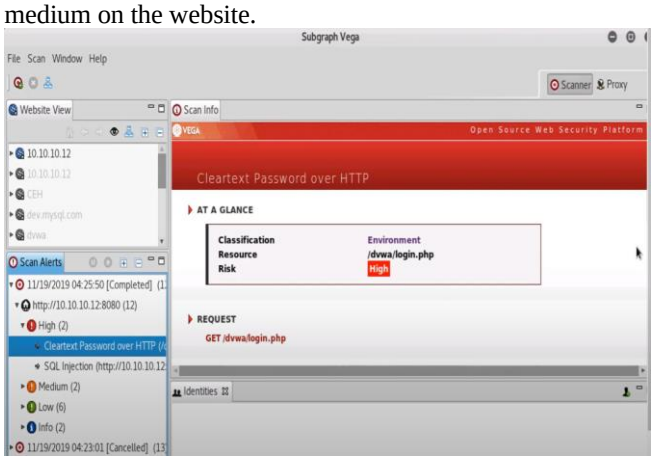

Fig. 2. Vega Framework

C. XSS Vulnerability in Web Application

With the help of a script, an attacker can enter malicious scripts in the database from the table content page, like entering contact information.

script <script>alert("You are hacked")</script>

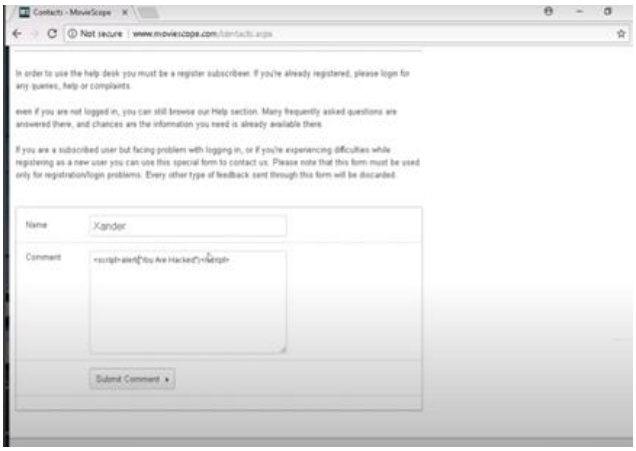

Fig. 3. XSS Vulnerability

D. SQL injection

The attacker login the web application with the help of the database query. Moreover, log in with a fake identity and access the users' sensitive data. Apart from that attacker create, update and delete the SQL database.

Type the query blah' or 1=1: Use names as login names in the Username, and leave the password blank. To log in, click the Login button.

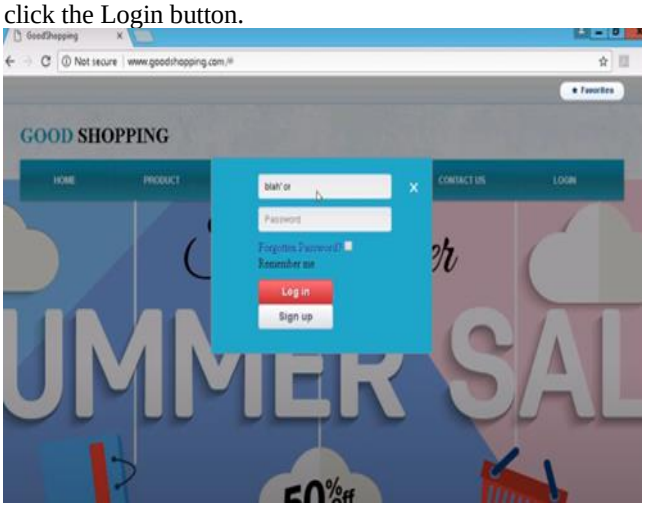

Fig. 4. SQL injection- Fake login

In the username field, type query blah';insert into login values ('john','apple123"); --

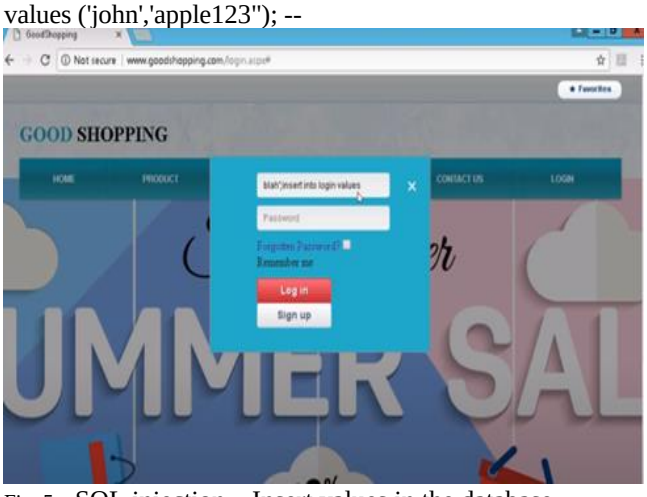

Fig. 5. SQL injection – Insert values in the database

This query creates a new database. Type the query blah'; create a database; --

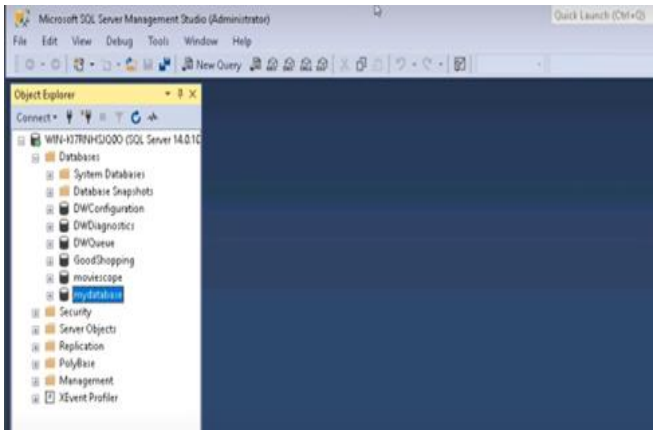

Fig. 6. SQL injection – Create a new database.

VI. Conclusion

To find vulnerabilities in web applications, pen testing requires. Additionally, vulnerability testing is divided into two types. It depends on the circumstances whether manual or automatic approaches work best. As evidenced by the above situations, manual testing is more effective in detecting unique vulnerabilities. Because automatic testing upon the most prevalent flaws frequently ignores the uncommon ones. Indeed, automated scanners are more successful in circumstances when common vulnerabilities must examine, saving the tester a significant amount of time. It is also better to look for sensitive information where it could keep. There are many possibilities for web application attacks, and this paper does not cover them all. In this study, a select handful is highlighted. The tests for these situations also use locally run web applications explicitly designed to test them. Performing manual penetration testing on a real-world web application under these circumstances may not be able to detect such flawthr. This research will help the read team, penetration testers, and security analysts understand the complete scenario of web app hacking from the attacker's perspective.

References

[1] Van-Giap Le, Huu-Tung Nguyen, Duy-Phuc Pham, Van-On Phung, and Ngoc-Hoa Nguyen, GuruWS: A Hybrid Platform for Detecting Malicious Web Shells and Web Application Vulnerabilities, 2019.

[2] Ms. Shweta Thakre, Studying the Effectiveness of Various Tools in Detecting the Protecting Mechanisms Implemented in Web-Applications, 2018.

[3] Mallick, P. K., Bhoi, A. K., Chae, G.-S., & Kalita, K. (Eds.). (2021). Advances in Electronics, Communication and Computing. Lecture Notes in Electrical Engineering.

[4] Debahuti Mishra, Rajkumar Buyya, Prasant Mohapatra, Srikanta Patnaik, Series: Smart Innovation, Systems and Technologies 153 Publisher: Springer Singapore;Springer, Year: 2021.

[5] Casola, V., De Benedictis, A., Rak, M., & Villano, U. (2018). Towards Automated Penetration Testing for Cloud Applications. 2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative.

[6] Tetskyi, A., Kharchenko, V., & Uzun, D. (2018). Neural networks based choice of tools for penetration testing of web applications. 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT).

[7] Muñoz, F. R., Armas Vega, E. A., & Villalba, L. J. G. (2016). Analyzing the traffic of penetration testing tools with an IDS. The Journal of Supercomputing.

[8] K Nagendran, A Adithyan, R Chethana, P Camillus and Sri Varshini K B Bala, "Web Application Penetration Testing", International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 10, August 2019, ISSN 2278-3075

[9] Kristian Beckers, Sebastian Pape, Peter Schaab and Daniel Schosser, Conference: International Conference on Trust and Privacy in Digital Business, August 2017.

[10] Tae Hyun Kim and Douglas Reeves, A survey of domain name system vulnerabilities and attacks, January 2020.

[11] Rizdqi Akbar Ramada, Redho Maland and Dedi Hariyadi, "Sudomy: Information Gathering Tools for Subdomain Enumeration and Analysis", The 2nd International Conference on Engineering and Applied Sciences 2019 (2nd InCEAS 2019), vol. 771, March 2020.

[12] Mayur Parmar, Google Dorks -Advance Searching Technique, August 2019.

[13] Mamta Bhavsar, Priyanka Sharma and Manik Gokani, "Port Scanning using Nmap", published at International Journal of Engineering Development and Research, December 2017.

[14] Mandala Mounica, R Vijayasaraswathi and R Vasavi, "Detecting Sybil Attack In Wireless Sensor Networks Using Machine Learning Algorithms", IOP Conference Series: Materials Science and Engineering, 2021.

[15] Delaitre AM, Stivalet BC, Black PE, Okun V, Cohen TS, Ribeiro A (2018) Sate v report: Ten years of static analysis tool expositions. NIST SP 500-326, National Institute of Standards and Technology (NIST).

[16] Doupé A, Cova M, Vigna G (2010) Why johnny can't pentest: An analysis of black-box web vulnerability scanners. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, pp. 111–131

[17] Fonseca J, Vieira M, Madeira H (2007) Testing and comparing web vulnerability scanning tools for sql injection and xss attacks. In: 13th Pacific Rim international symposium on dependable computing (PRDC 2007), IEEE, pp. 365–372

[18] MITRE (2021) Cwe view: Weaknesses in owasp top ten (2021). In: (MITRE 2021b), https://cwe.mitre.org/data/definitions/1344.html. Accessed 09 Dec 2021.