Int. J.Adv.Sig.Img.Sci, Vol. 10, No. 2, 2024

SECURING WEB APPLICATIONS WITH OWASP ZAP FOR COMPREHENSIVE SECURITY TESTING

S. P. Maniraj
Department of Data Science and Business Systems, School of Computing,
SRM Institute of Science and Technology,
Kattankulathur, Chennai, Tamil Nadu, India.
manirajp@srmist.edu.in

Chitra Sabapathy Ranganathan
Mphasis Corporation,
Chandler, Arizona, USA.
chitrasabapathyranganathan@gmail.com

Satheeshkumar Sekar
Mphasis Corporation,
Chandler, Arizona, USA.
satheeshkumar.sekar24@gmail.com

Submitted: *Jun,03, 2024* ***Revised:*** *Aug,08, 2024* ***Accepted:*** *Aug,31, 2024*

Abstract: The powerful Open Web Application Security Project (OWASP)-Zed Attack Proxy (ZAP) tool secures web applications with extensive security testing. Its main goal is to find and fix web application vulnerabilities before they can be exploited. The goal is to improve web application security using OWASP ZAP scans and inspections. Simulating SQL injection and cross-site scripting attacks using the tool reveals an application's security flaws. OWASP ZAP automates testing to protect sensitive data and web application integrity. The purpose is to protect online applications from attacks to reduce security breaches and ensure industry compliance. Modern online applications' security and dependability depend on OWASP ZAP's systematic vulnerability discovery and mitigation. By enhancing scan performance and providing actionable security information, the experimental results show that the OWASP ZAP helps safeguard online applications and reduce the danger of cyberattacks. The objective is to enhance its skills to provide comprehensive, prompt, dependable security evaluations across various situations.

Keywords: Zed attack proxy, web application security, vulnerability detection, security testing, threat mitigation.

I. INTRODUCTION

Many online services and platforms depend on web apps, essential to the digital infrastructure. The growth of web-based applications has increased security risks. Open-source web application security testing tool OWASP Zed Attack Proxy discovers and resolves vulnerabilities. OWASP ZAP simulates numerous attack pathways to help security experts find XSS, SQL injections, and dangerous setups. Flexibility and usability make it vital for web application security testing. This research shows how OWASP ZAP extensively tests web applications for security. Because web apps are online so frequently, many assaults may target them. Active and passive scanning by OWASP ZAP finds web application vulnerabilities. Intercepting, assessing, and reporting userapplication communication discloses security concerns. Allowing administrators to apply critical updates minimizes attack risk.

OWASP ZAP will be tested for web application vulnerability identification and fixation. Scanning, fuzzing, spidering, and attack simulation are OWASP ZAP features. ZAP finds vulnerabilities traditional testing misses due to these traits. The application supports many securities testing frameworks, making it useful in numerous development contexts. Scan for current and upcoming security vulnerabilities to safeguard internet applications. To showcase how OWASP ZAP improves security testing, various web app vulnerability detection and resolution solutions are displayed. Traditional security testing ignores major weaknesses when modeling attacks. OWASP ZAP automates and scales the whole software development lifecycle. As a security expert or beginner, its user-friendly UI, substantial documentation, and community assistance may help protect online apps.

The relevance of security testing in preventing web application attacks is stressed. Online application security gets harder as it becomes more complicated. OWASP ZAP can detect numerous web application security issues with its robust and adaptable solution. Automated testing and regular scans by ZAP enable businesses to safeguard their web applications from known and future dangers. To protect consumers and businesses online, OWASP ZAP stops cyberattacks. The reason and how OWASP ZAP is used for Web Application Comprehensive Security Testing is explained in Section 2. OWASP ZAP's Comprehensive Web Application Security Testing is covered in Sector 3. Section 4 illustrates how Comprehensive Security Testing of Web Applications uses OWASP ZAP across datasets. Finally, Section 5 concludes with a conclusion.

II. RELATED WORKS

An analysis of OWASP ZAP to detect security gaps on websites is discussed in [1]. A detailed presentation on using the OWASP ZAP tool to find website security holes is also included. Automated scanning for vulnerabilities was one of the approaches examined in the evaluation. It demonstrated how OWASP ZAP might improve website security by doing thorough and systematic vulnerability assessments, and it highlighted the tool's usefulness in finding security issues across various online applications. Methods, tools, and techniques for security testing that investigate and fix vulnerabilities are discussed in [2]. A wide range of security testing approaches, tools, and procedures were researched in depth to find and fix security flaws. A variety of security testing methodologies, both old and new, were included in the study. The main objective was to assess how well these methods improved overall cybersecurity and how well they uncovered possible security flaws.

Exploring security vulnerabilities in SIAKAD websites for higher education is described in [3]. The vulnerability assessment methodology allowed for discovering security holes in SIAKAD websites used by institutions of higher learning. The assessment found these educational platforms to have certain security flaws and suggested ways to fix them. The results highlighted the need to conduct frequent security evaluations to guarantee the safety of sensitive information on academic websites. Automated penetration testing can improve cybersecurity for small businesses at a low cost, as described in [4]. An automated penetration testing tool-based approach to improving small businesses' cybersecurity at a reasonable cost was detailed. This method showed how tiny firms may use automated methods to find and fix security flaws without a large financial outlay. These solutions provide a simple and effective way to boost cybersecurity for settings with limited resources.

A case study on OWASP's top 10 methods for website security analysis was conducted in [5]. A comprehensive evaluation of website security was provided via the OWASP top 10 methodology, emphasizing a particular website serving as a case study. The assessment offered key vulnerabilities and ideas on using the OWASP top 10 methodology to analyze and enhance website security. This approach was shown in the case study to detect and reduce typical threats to website security. Assessment of OWASP-based common security vulnerabilities in online resources of state universities and colleges is described in [6]. The OWASP framework showed how state university and college websites are prone to typical security flaws. The examination aimed to find common security concerns and how they affect institutional websites. The results showed how to fix security issues in educational online platforms by following the OWASP standards.

Google's penetration testing and open web application security projects are utilized to evaluate website security [7]. An in-depth analysis of the website's security was conducted utilizing a combination of Google's penetration testing and the OWASP top 10 methodology. This method used two well-known security evaluation tools to find security flaws and evaluate the website's defenses. The testing was used in a dual-layered examination to make the security analysis more robust. The OWASP framework is utilized in [8] to prevent SQL injection attacks [8]. The OWASP framework was used to demonstrate a technique that may reduce the impact of SQL injection attacks on web servers. The methodology details methods for protecting against SQL injection vulnerabilities according to the OWASP standards. One of the most prevalent and harmful forms of assaults on web applications may be better protected with the help of these tactics.

A rundown of the automated technologies that can check the security of a cloud is discussed in [9]. Tools for assessing and protecting cloud infrastructures were the main topic of discussion. The evaluation showed that these technologies were good at finding security flaws and ensuring that cloud-based apps and services were safe. The evaluation of top application security tools, from static analysis to runtime protection, is studied in [10]. A comprehensive review of the best application security technologies covered everything from static analysis to runtime protection. The review aimed to shed light on the strengths and weaknesses of different security solutions in protecting apps across their entire lifespan. The results verified the usefulness of these instruments in all-encompassing application security management.

Secure software development and testing using a model-based approach is described in [11]. This technique provides a systematic way to integrate security into the software development lifecycle. The suggested approach guarantees a strong defense against any vulnerabilities by integrating security measures from the beginning of development through testing. The prevention of firewall and web application security breaches using Vulnerability Assessment and Penetration Testing (VAPT) is described in [12]. VAPT was a method for reducing security risks in firewalls and online applications. Methodical evaluations and testing are at the heart of the strategy, which aims to find and fix security holes. A complete framework for improving the security of network infrastructures and web applications is provided by VAPT implementation.

Predicting assaults on servers hosted on the web was investigated in [13] using expert systems. This method uses sophisticated technology to assess and foresee any security risks. Enhancing proactive security measures is the goal of using expert systems, which may provide early alerts and predictive insights into potential attack routes. An Architecture for cloud-based security risk analysis and penetration testing is discussed in [14]. The cloud computing framework was detailed. The framework offers a methodical way to evaluate and fix security flaws in systems that run in the cloud. From the point of view of penetration testers, the suggested model is made to strengthen the security of cloud environments.

An empirical examination emphasizing continuous security testing is discussed in [15]. The research examined how software development security might be enhanced by including security testing in the DevOps process. The results show how important it is to do security assessments regularly throughout the development lifecycle so that vulnerabilities may be found and fixed. Analyzing website HTTP response headers for vulnerabilities using standard procedures for penetration testing is described in [16]. The study looked for security holes in these headers to determine how to secure HTTP response headers against possible attacks effectively.

The common exposures and vulnerabilities at private universities are examined in [17]. It focused on web application security and thoroughly evaluated typical vulnerabilities and exposures at private universities. Examining the prevalence of security concerns and their effects on private educational institutions was the study's primary goal, which sought to demonstrate the critical need for robust security measures to safeguard confidential data. The impact of web application security flaws and their fixation via experimentation is analyzed in [18]. Web application vulnerability detection and mitigation was the subject of an experimental investigation. Web application security flaws may be discovered and fixed using the methods and tools discussed in this research. The results emphasize practical techniques to improve the security of web-based systems using experimental methodology.

The OWASP-recommended web security assessments are studied in [19]. It demonstrated an examination of online security by OWASP standards. To analyze and enhance online security, the evaluation focused on implementing OWASP guidelines. This research proved that web apps are better protected against typical vulnerabilities and threats when developers follow the OWAS principles. The effectiveness of DevOps security tools for finding and sealing security vulnerabilities is discussed in [20]. Several security technologies integrated into the DevOps architecture were the primary subjects of the evaluation. The results show how these technologies help with security problem detection and resolution, which improves DevOps environment security in general.

III. PROPOSED SYSTEM

Open-source security testing tool OWASP ZAP finds online application vulnerabilities. It is extensively used for manual and automated security testing and is maintained by OWASP. OWASP ZAP helps find XSS, SQL injections, and broken authentication issues. As an intercepting proxy, OWASP ZAP monitors and modifies browser-web application communication. This allows testers to analyze real-time requests and answers, revealing security vulnerabilities. Active scanning by ZAP simulates application assaults to find vulnerabilities. A comprehensive web application security testing approach requires the tool's simplicity and interoperability with multiple development environments. Passive scanning, automated fuzzing, and vulnerability reporting let novices and experts find and fix security vulnerabilities. ZAP will set up a proxy server and redirect all web traffic. Auto scanners are a part of it and may help find security flaws on websites. Figure 1 shows a ZAP Proxy Server.

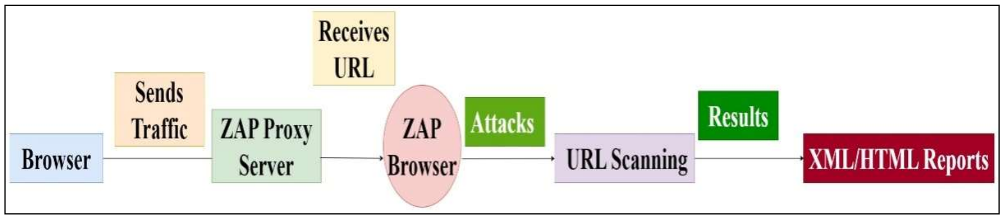

Fig. 1 ZAP Proxy Server

OWASP ZAP's passive and active web application scans help find security issues. Passive scanning detects information leaks, cookie misconfigurations, and SSL/TLS vulnerabilities without altering requests. However, active scanning sends forged requests to the program to attack SQL injections and XSS vulnerabilities. ZAP's spidering is vital for application structure mapping. ZAP crawls the application to find all available pages and links for thorough testing. After collecting website and input data, the spider uses active scanning to find vulnerabilities in form fields, URLs, and other input points. ZAP enables web application fuzzing, which includes delivering random or erroneous data to input fields to find buffer overflows or faulty input validation. Fuzzing adds value to security testing by detecting vulnerabilities that standard scanning may miss. A built-in automatic vulnerability scanner in OWASP ZAP creates reports on found vulnerabilities, detailing each vulnerability and its severity. These reports help developers and security teams prioritize and fix security problems. Figure 2 shows ZAP on a virtual machine running Windows Server 2012. This computer is only accessible to a select few; thus, the final report remains secure. A basic PHP web service is made that TFS can call using the ZAP API. The Webservice returns an OK or a summary of the results.

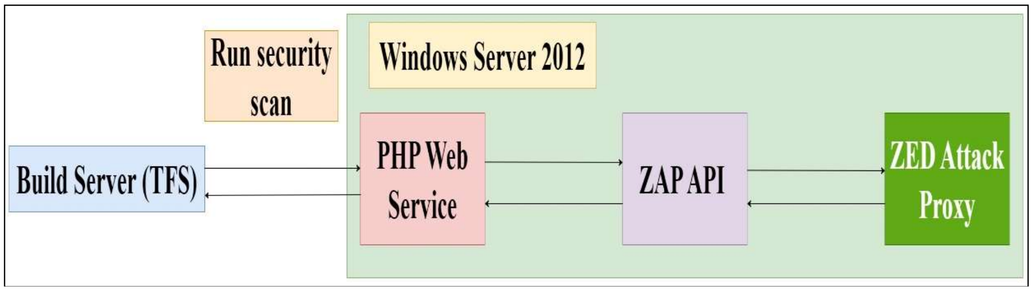

Fig. 2 Zap Application on Windows Server 2012 VM

OWASP ZAP's ability to do thorough security testing without setup or knowledge is a major asset. Its user-friendly interface and interaction with development environments make it accessible to security experts and developers. ZAP may be integrated into DevOps processes to enable consistent security testing, lowering the risk of production vulnerabilities. This study hopes to improve understanding by reviewing the literature, learning how the system works, collecting data, scanning for vulnerabilities, evaluating, testing, and documenting the results. The result may be seen in Figure 3.

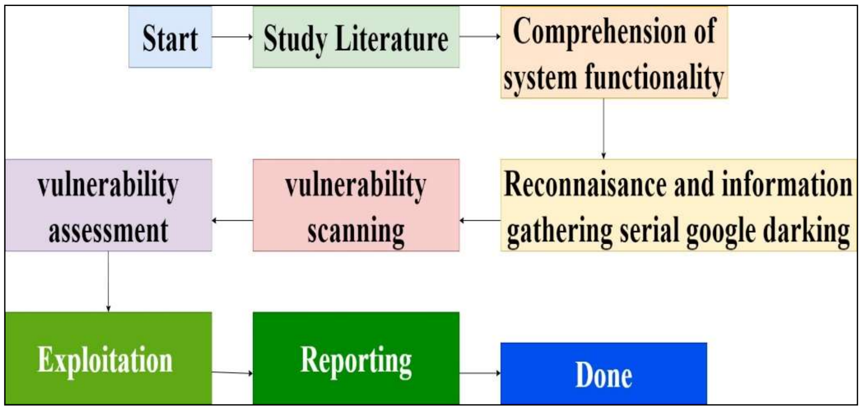
Start
Study Literature
Comprehension of system functionality
Reconnaisance and information gathering serial google darking
vulnerability scanning
vulnerability assessment
Exploitation
Reporting
Done

Fig. 3 Research stages using the OWASP framework

The trend towards DevSecOps, which incorporates security into the CI/CD pipeline, relies on OWASP ZAP. ZAP can automate security tests and interface with CI/CD platforms like Jenkins, GitLab, and CircleCI to test at every step of software development. This detects security issues early, lowering remedial costs and complexity. Developers may detect vulnerabilities before production by automating security scans using OWASP ZAP in the CI/CD pipeline. This proactive strategy addresses security problems during development, decreasing the risk of delivering unsafe apps. As new features and upgrades are released, automated security testing provides regular security evaluations to keep applications safe. ZAP's interaction with CI/CD systems may be customized using scripts and API connectors for security testing. ZAP can automatically scan fresh builds and deployments and provide reports for development and security teams. This engagement across teams promotes shared security responsibility and reduces testing bottlenecks.

OWASP ZAP dramatically minimizes web application vulnerability concerns. ZAP prevents XSS, SQL injection, and sensitive data disclosure by discovering security problems early and offering remedial instructions. Unpatched vulnerabilities may cause data breaches, unauthorized access, and huge financial losses. OWASP ZAP's ability to identify invalid session management, access restrictions, and insecure Direct Object References (IDOR) improves online application security. These vulnerabilities are typically used to elevate privileges, circumvent authentication, or access restricted resources. OWASP ZAP informs security teams of application dangers by scanning and delivering real-time feedback. The ongoing vulnerability assessment technique helps organizations maintain a solid security posture when introducing web application features and functionalities.

OWASP ZAP will grow to address new security issues when web applications adopt new technology. PWA and Web Assembly support might be expanded. ZAP can secure future web technologies by improving the scanning and testing of contemporary apps. Another potential development is integrating AI and ML to boost ZAP's vulnerability detection. AI might improve ZAP's ability to detect complicated attack patterns, reduce false positives, and find new vulnerabilities. AI-driven scanning might let ZAP adapt to different application architectures and setups. Modern online applications increasingly use APIs to link systems and services, making API security a key potential for OWASP ZAP. Increased API scanning by OWASP ZAP will provide more extensive security evaluations of RESTful, SOAP, and GraphQL APIs. The pseudo-code for the proposed algorithm is as follows:

1. *Initialize OWASP ZAP Client*:
   - zap_client: Initialize the ZAP client using the provided API key and base URL.
2. *Define the target URL*:
   - target_url: The URL of the web application to be scanned.
3. *Start a new session*:
   - start_new_session(zap_client, session_name="ExampleSession"): Start a new ZAP session for the scanning process.
4. *Open the target URL in ZAP*:
   - open_url(zap_client, target_url): Open the target URL in ZAP to begin interaction.
5. *Spider the target URL to discover all pages*:
   - start_spider(zap_client, target_url): Start a spider scan to crawl the target URL and discover all linked pages.
   - get_spider_status(zap_client, spider_id): Check the status of the spider until it is completed.

```
# Initialize OWASP ZAP Client
zap_client = initialize_zap_client(api_key, base_url)
# Define the target URL
target_url = "http://example.com"
# Start a new session
start_new_session(zap_client, session_name="ExampleSession")
# Open the target URL in ZAP
open_url(zap_client, target_url)
# Spider the target URL to discover all pages
spider_id = start_spider(zap_client, target_url)
# Monitor the spidering status until the completion
while True:
spider_status = get_spider_status(zap_client, spider_id)
    if spider_status == "completed":
break
    else:
sleep(10)  # Check spider status every 10 seconds
# Start an active scan on the target URL
scan_id = start_active_scan(zap_client, target_url)
# Monitor the scan status until the completion
while True:
scan_status = get_scan_status(zap_client, scan_id)
    if scan_status == "completed":
break
    else:
sleep(10)  # Check scan status every 10 seconds
# Retrieve the scan results
scan_results = get_scan_results(zap_client, scan_id)
# Analyze scan results
def analyze_scan_results(scan_results):
    vulnerabilities = []
    for alert in scan_results['alerts']:
```

```
vulnerabilities.append({
            "alert": alert['alert'],
            "url": alert['url'],
            "risk": alert['risk'],
            "description": alert['description'],
            "solution": alert['solution']
        })
    return vulnerabilities
vulnerabilities = analyze_scan_results(scan_results)
# Log vulnerabilities
log_vulnerabilities(vulnerabilities)
# Notify the security team about high-risk vulnerabilities
for vulnerability in vulnerabilities:
    if vulnerability['risk'] == 'High':
notify_security_team(vulnerability)
# Generate summary report
summary_report = generate_summary_report(vulnerabilities)
# Output the summary report
           output_report(summary_report)
```

III. RESULTS AND DISCUSSIONS

OWASP ZAP can discover numerous vulnerabilities; however, it has limits. Like other automated techniques, ZAP may struggle to find business logic errors. These issues result from how an application manages workflows and procedures, which automated scanners cannot understand. Some vulnerabilities may not be found until later in the development cycle if testing is done on incomplete or obsolete web applications. Figure 4 shows the number of OWASP ZAP vulnerabilities in web application components.

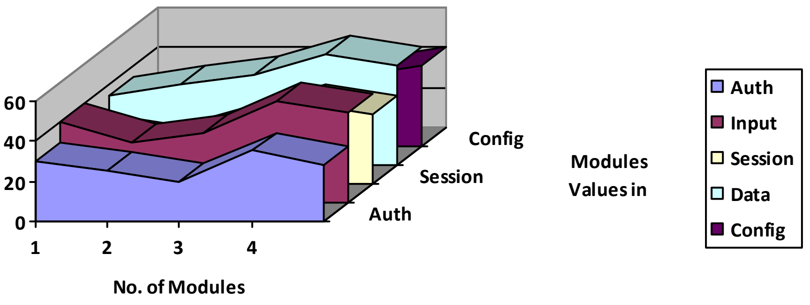

Fig. 4 Vulnerability Count Across Web Application Modules.

Each column shows the number of vulnerabilities in a module, such as Auth, Input Validation, Session Management, Data Handling, and Configuration. For instance, the Input module has 50 vulnerabilities, suggesting a need for security assessment and repair. This data helps developers understand which modules need the greatest effort to improve web application security. Table 1 shows OWASP ZAP data for complete web application security testing. Vulnerabilities Detected shows ZAP's scanning capability by counting web application security issues.

TABLE. 1 Web application security metrics with OWASP ZAP

Metric	**Value 1**	**Value 2**	**Value 3**	**Value 4**	**Value 5**
Vulnerabilities Detected	120	140	110	150	130
High-Risk Issues	25	30	20	35	28
False Positives	10	8	12	9	7
Test Coverage (%)	90%	92%	88%	94%	91%
Average Scan Time (minutes)	30	25	35	28	32

High-risk issues identify the most serious vulnerabilities, demonstrating the tool's ability to detect urgent threats. False Positives—incorrect warnings that demand additional investigation—are an essential scan accuracy statistic. Test Coverage indicates how much of the web application was tested for security vulnerabilities. The tool's average scan time shows its effectiveness in discovering vulnerabilities quickly. These metrics show OWASP ZAP's thorough web application security evaluation, helping organizations decrease risks.

ZAP may be integrated with threat intelligence solutions to provide real-time information regarding exploited threats and vulnerabilities. Threat intelligence integration lets OWASP ZAP scan for zero-day vulnerabilities and new threats using current attack vector knowledge. ZAP provides insight into how attackers could exploit vulnerabilities by tying them to known attack tactics, exploit kits, or malware families. This information lets security professionals assess each vulnerability's effect and modify defenses. Figure 5 displays 12 months of vulnerabilities by severity (Critical, High, Medium, Low).

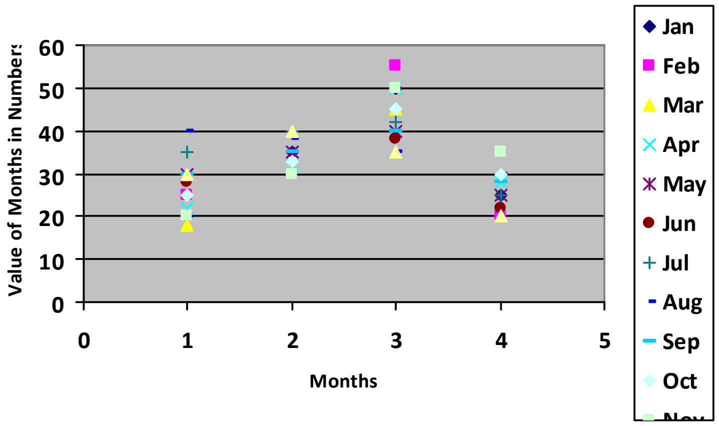

Fig. 5 Vulnerability severity levels in Web application testing

Each cell shows the number of vulnerabilities in each severity category for a month. August saw a surge in Critical vulnerabilities (40), indicating a security risk. Tracking severity levels over time helps identify patterns and prioritize patches by vulnerability severity and frequency. This guarantees that major problems are handled quickly to secure online applications.Table 2 describes how to secure online applications using OWASP ZAP, a comprehensive security testing tool.

TABLE. 2 Aspects of OWASP ZAP for Web Application Security Testing

Aspect	Role	Pros	Cons
Vulnerability Scanning	Identifying Web App Security Flaws	Extensive vulnerability detection	Can produce false positives
Automated Testing	Conducting Security Tests Automatically	Efficient and easy to use	May miss complex or business-logic vulnerabilities
Manual Testing	Customizing Security Analysis	Flexibility in targeted testing	Requires expert knowledge for effectiveness
Reporting	Generating Security Reports	Helps in compliance and audit reviews	Reports may require manual interpretation
Integration	Supporting Development Workflows	Seamless integration with DevOps tools	It may slow down development pipelines if not optimized

Vulnerability Scanning is key to finding web application security problems but may produce false positives. Automated testing saves time and reduces human effort, yet it may overlook complicated vulnerabilities. CI/CD pipeline integration allows continuous security checks throughout development, boosting proactive security management, although it might delay development if not optimized. Each part shows how OWASP ZAP secures web applications during development and testing.

IV. CONCLUSIONS

OWASP ZAP's web application security advantages and difficulties are enormous. Its thorough security testing finds flaws that would otherwise go undetected, strengthening web application defenses. The program relies on human skills to evaluate results, and false positives might lead to needless repair. It can identify simple assaults, but complicated ones need more advanced detection techniques. OWASP ZAP must be updated and improved as web application threats change. Automation, machine learning for better threat detection, and vulnerability mitigation may be future improvements. These areas must be developed to keep OWASP ZAP relevant and successful in web application security's ever-changing industry. Future developments include integrating machine learning for improved vulnerability identification, extending support for mobile and IoT application security, and the provision of advanced real-time reporting dashboards. Highlighting usability for non-experts may enhance its uptake across several sectors. OWASP ZAP is crucial for thorough security testing because of its accessibility, extensive features, and capability to mitigate key vulnerabilities in contemporary online applications successfully.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare they have no conflicts of interest to report regarding the present study.

REFERENCES

[1]. F. P. Putra, U. Ubaidi, A. Hamzah, W. A. Pramadi and A. Nuraini, “Systematic Literature Review: Security Gap Detection on Websites Using Owasp Zap,” Brilliance: Research of Artificial Intelligence, vol. 4, no. 1, 2024, pp. 348-355.

[2]. S. H. Sanne, “Investigations into Security Testing Techniques, Tools, and Methodologies for Identifying and Mitigating Security Vulnerabilities,” Journal of Artificial Intelligence, Machine Learning and Data Science, vol. 1, no. 1, 2024, pp. 626-631.

[3]. N. A. Syarifudinand L. Setiyani, “Analysis of Higher Education SIAKAD Website Security Gaps Using the Vulnerability Assessment Method,” International Journal of Multidisciplinary Approach Research and Science, vol. 1, no. 3, 2024, pp. 332-344.

[4]. Y. Alkhurayyif and Y. S. Almarshdy, “Adopting Automated Penetration Testing Tools: A Cost-Effective Approach to Enhancing Cybersecurity in Small Organizations,” Journal of Information Security and Cybercrimes Research, vol. 7, no. 1, 2024, pp. 51-66.

[5]. B. S. Pradhana, “Website Security Analysis Using the OWASP10 Method (Case Study: almumtazparfumebatam. store),” Jurnal Kewarganegaraan, vol. 8, no. 1, 2024, pp. 588-605.

[6]. C. P. Flores Jr and N. Richard, “Evaluation of Common Security Vulnerabilities of State Universities and Colleges Websites Based on OWASP,” Journal of Electrical Systems, vol. 20, no. 5s, 2024, pp. 1396-1404.

[7]. A. F. Sebrina, A. Junaidi and A. N. Sihananto, “Testing posketanmu website with google penetration testing and OWASP Top 10,” Jurnal Mantik, vol. 8, no. 1, 2024, pp. 636-645.

[8]. A. Fadlil, I. Riadi and M. A. Mu’min, “Mitigation from SQL Injection Attacks on Web Server using Open Web Application Security Project Framework,” International Journal of Engineering, vol. 37, no. 4, 2024, pp. 635-645.

[9]. H. Ghazizadeh, G. Tamm and R. Creutzburg, “Automated Tools for Cloud Security Testing,” Electronic Imaging, vol. 36, 2024, pp. 1-7.

[10]. A. A. Fernandes, “Evaluating the Top Application Security Tools: From Static Analysis to Runtime Protection,” Asian Journal of Research in Computer Science, vol. 17, no. 7, 2024, pp. 119-127.

[11]. V. Casola, A. De Benedictis, C. Mazzocca and V. Orbinato, “Secure software development and testing: A model-based methodology,” Computers & Security, vol. 137, 2024, pp. 1-16.

[12]. A. Alquwayzani, R. Aldossri and M. Frikha, “Mitigating Security Risks in Firewalls and Web Applications using Vulnerability Assessment and Penetration Testing (VAPT),” International Journal of Advanced Computer Science & Applications, vol. 15, no. 5, 2024, pp. 1-17.

[13]. M. Z. Ariffin, and H. F. Hakim, “Use of Expert Systems to Predict Attacks on Web-Based Servers,” Jurnal Inovasi Teknologi dan Edukasi Teknik, vol. 4, no. 2, 2024, pp. 1-11.

[14]. N. E. Ismail, N. H. Ali, M. A. Jalil, F. Yunusand A. D. Jarno, “A Proposed Framework of Vulnerability Assessment and Penetration Testing (VAPT) in Cloud Computing Environments from Penetration Tester Perspective,”

Journal of Advanced Research in Applied Sciences and Engineering Technology, vol. 39, no. 1, 2024, pp. 1-14.

[15]. C. Feio, N. Santos, N. Escravanaand B. Pacheco, "An Empirical Study of DevSecOps Focused on Continuous Security Testing," IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2024, pp. 610-617.

[16]. H. Kurniawan and E. Christianto, "Analysis Vulnerability Website Baleomolcreative dengan Metode Penetration Testing Execution Standard & Vulnerability Assessment Pada Http Response Header Field," Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi), vol. 8, no. 3, 2024, pp. 734-745.

[17]. E. F. Mangaoang and R. N. Monreal, "Common Vulnerabilities and Exposures Assessment of Private Higher Educational Institutions Using Web Application Security," Journal of Electrical Systems, vol. 20, no. 5s, 2024, pp. 668-676.

[18]. R. P. Kollepalli, M. J. Reddy, B. L. Sai, A. Natarajan, S. Mathi and V. Ramalingam, "An Experimental Study on Detecting and Mitigating Vulnerabilities in Web Applications," International Journal of Safety & Security Engineering, vol. 14, no. 2, 2024, pp. 1-10.

[19]. V. I. Sugara, and I. W. Sriyasa, "Analisis Keamanan Web Menggunakan Open Web Application Security Web (OWASP)," Indonesian Journal of Computer Science, vol. 13, no. 2, 2024, pp. 3315-3327.

[20]. S. T. Makani, and S. Jangampeta, "Devops Security Tools Evaluating Effectiveness in Detecting and Fixing Security Holes," International Journal of DevOps (IJDO), vol. 1, no. 2, 2024, pp. 1-12.