applied sciences

Article

A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions

Jahanzeb Shahid [1], Muhammad Khurram Hameed [2], Ibrahim Tariq Javed [3,*], Kashif Naseer Qureshi [3], Moazam Ali [3] and Noel Crespi [4]

[1] School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan; 13msccsjshahid@seecs.edu.pk
[2] Paperlessly Pakistan, Islamabad 44000, Pakistan; khurram@paperlessly.com
[3] Center of Excellence in Artificial Intelligence (CoE-AI), Department of Computer Science, Bahria University, Islamabad 44000, Pakistan; knaseer.buic@bahria.edu.pk (K.N.Q.); muazzam.buic@bahria.edu.pk (M.A.)
[4] Institut Mines-Télécom, Télécom SudParis, CEDEX, 91011 Evry, France; noel.crespi@mines-telecom.fr
* Correspondence: itariq.buic@bahria.edu.pk

Abstract: The growing use of the internet has resulted in an exponential rise in the use of web applications. Businesses, industries, financial and educational institutions, and the general populace depend on web applications. This mammoth rise in their usage has also resulted in many security issues that make these web applications vulnerable, thereby affecting the confidentiality, integrity, and availability of associated information systems. It has, therefore, become necessary to find vulnerabilities in these information system resources to guarantee information security. A publicly available web application vulnerability scanner is a computer program that assesses web application security by employing automated penetration testing techniques that reduce the time, cost, and resources required for web application penetration testing and eliminates test engineers' dependency on human knowledge. However, these security scanners possess various weaknesses of not scanning complete web applications and generating wrong test results. Moreover, intensive research has been carried out to quantitatively enumerate web application security scanners' results to inspect their effectiveness and limitations. However, the findings show no well-defined method or criteria available for assessing their results. In this research, we have evaluated the performance of web application vulnerability scanners by testing intentionally defined vulnerable applications and the level of their respective precision and accuracy. This was achieved by classifying the analyzed tools using the most common parameters. The evaluation is based on an extracted list of vulnerabilities from OWASP (Open Web Application Security Project).

Keywords: security vulnerabilities; NIST; OWASP; OWASP top 10; evaluation; comparison; analysis; web application scanners; SQL injection; cross-site scripting; broken authentication; sensitive data exposure; XML injection; security misconfiguration; insecure deserialization; insufficient logging and monitoring; automated vulnerability detection

check for updates

Citation: Shahid, J.; Hameed, M.K.; Javed, I.T.; Qureshi, K.N.; Ali, M.; Crespi, N. A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions. *Appl. Sci.* **2022**, *12*, 4077. https://doi.org/10.3390/app12084077

Academic Editors: Manuel Armada and Gianluca Lax

Received: 28 January 2022
Accepted: 7 April 2022
Published: 18 April 2022

1. Introduction

The internet plays a significant role in our lives. In their daily lives, people interact with each other with the help of customized and user-friendly web applications developed by using different languages, platforms, and scripting environments. This heterogeneous nature of web applications makes it difficult for a developer to update and secure those against emerging threats and attacks properly.

Initially, these web applications were developed in either stand-alone or client-server models with meagre security issues. Therefore, it was easy to secure and analyze a web application. However, the abundant reliance of information systems on web applications has given rise to many security problems; internet banking systems, e-commerce systems and governmental transaction platforms are severely affected whenever their respective web applications are compromised.

Statistically, 19% of scanned web applications' vulnerabilities give the attacker control the application and operating system. In 2019, a report on vulnerabilities in corporate information systems [1] found that nearly 75% of Local Area Network (LAN) penetration vectors have vulnerabilities in web application protection. Similarly, loopholes during the development process cause serious threats to web applications. Moreover, changes in configuration settings are adequate to affix only 17% of vulnerabilities. Most of these have low severity levels. Fifty percent of leaks cause the disclosure of account credentials and personal data leakage, and around 91% of scanned web applications store and process personal data [2].

On average, 33 vulnerabilities are found in each web application, six of which are of high severity. Severe vulnerabilities per web application have increased three-fold compared to 2017.

The selection of the right penetration testing methodology for a specific type of vulnerability [3] plays an instrumental part in scanning the web application for vulnerability detection. Moreover, the automation of web application testing is helpful to pen-testers, which has not only reduced labour work, time, resources, and cost, but has also alleviated pen-tester dependency on human knowledge [4]. Furthermore, the scanner saved the human knowledge of pen-testing by creating executable computer programs. Therefore, the development of automated web application security scanners has made pen-testing a popular research trend. In this field, developers convert the web application pen-testing methods into executable programs to detect web application vulnerabilities better.

Web application assessments are becoming more pervasive with the development of automated software and tools that simulate the actions of an experienced web application penetration tester [4]. Different programs and tools are available to help a penetration tester gather information. Similarly, tools such as Metasploit [5], and Commix [6] are developed to attack the target system's confidentiality, integrity, and availability.

Web assessment tools are classified into two categories: open-source tools and proprietary tools. There are some limitations and capabilities associated with these categories. Open-source tools are free to download, but the developers also have access to their source codes. Therefore, developers can customize their functions and features according to their requirements. There is a big community linked with these open-source technologies, where a developer can find help to customize open-source tools. On the other hand, proprietary or paid tools are not freely available. However, their beta versions or trial versions are freely available online. By downloading a trial version, one can experience very limited features of these tools. In addition, their source codes are not freely available, so these tools cannot be customized.

Research led by WhiteHat Security [7] found that 86% of scanned web applications had, on average, 56% of vulnerabilities per web application, at least one of which was classified as serious. In another study [8], Symantec executed 1400 scans and identified that 76% of websites have at least one serious vulnerability [9], and 20% of servers consist of critical vulnerabilities.

However, the statistics and classification of detected vulnerabilities differ from one scanner to another. For example, some web assessment tools will successfully identify all "true" vulnerabilities with a low false-positive rate. In contrast, others will fail to detect true positive vulnerabilities with a considerably high false-positive rate.

This research is also motivated by the need for using suitable web assessment tools for scanning and vulnerability assessments of web applications. As we know, every execution performed by a computer is based on an algorithm. Therefore, to achieve the aim of automated web application assessment, it is very important to write a sophisticated algorithm. However, these algorithms are not always seamless, and false positives and false negatives are common for automated web application penetration testing. The false positives require extra effort to be made by the pen-tester and time to eliminate the false vulnerability. In contrast, the false negatives impair the pen-tester's judgement in deciding an under-test web application security. However, some research works claims web security scanners as untrustworthy and inaccurate. This raises the importance of experimentation for the assessment of web application security scanners' performances [10]. Moreover, there is no standard document issued by authorized parties that defines a good approach for assessing a web application security scanner's performance.

Black box and White box testing [11] are the primary techniques for the security assessment of web applications. White box testing examines the internal structures of web applications. However, web applications usually consist of multiple technologies and programming languages, including client and server-side languages. Therefore, such testing may fail to capture all security flaws and vulnerabilities due to the code complexity. On the other side, the black box testing technique, also known as behavioural testing, is in which the internal design or structure of the testing item is unknown to the tester. These tests can be functional or non-functional; however, most of the time, they are functional.

For comparison, there is a well-defined standard implemented for assessment purposes: OWASP (Open Web Application Security Project) [12]. This standard provides software developers and purchasers with a means to decide whether the tools in consideration for use should meet the software assurance requirements or not. OWASP addresses only the top 10 security risks every three years. It publishes the ranking of the ten most dangerous security vulnerabilities in the document "OWASP Top 10" [13] and enables the security team to protect the web application against the most important threats [7].

In this survey paper, a comprehensive comparative analysis is presented with different open-source and proprietary tools for vulnerabilities detection in web applications [14]. Moreover, these tools have also been assessed for their precision value to detect vulnerabilities in web applications. Twelve tools are compared according to their capabilities. Five tools (Acunetix [15], Nessus [16], HP WebInspect [17], IBM APPSCAN [18], and NetSparker [19]) are proprietary; their vulnerability detection statistics were collected from the literature. Other open-source tools were installed and configured in the laboratory. These open-source tools were run against DVWA (Damn Vulnerable Web Application) [20], a vulnerable application that is used to check the scanner's detection capabilities. The detection results of these scanners were compared to OWASP defined vulnerabilities.

OWASP Framework was chosen to compare different tools concerning OWASP TOP 10 Vulnerabilities. OWASP is open source and easily accessible to anyone to improve web application security-related problems. It is an important framework that can be used for web application security as it provides documentation, videos, and forums. One of its best-known projects is OWASP Top 10. Therefore, our survey work covers OWASP Top 10 Vulnerabilities

The literature review of web application assessment is presented in Section 2. A brief description of security assessment tools for web application is presented in Section 3. Section 4 consists of comparison results for selected web assessment tools. Our contributions are presented in Section 5. Open challenges and future directions are illustrated in Section 6. Our conclusions are discussed in Section 7.

2. Literature Review

Several research studies have discussed the issue of evaluating the performance and effectiveness of web application assessment tools. Several researchers researched only commercial or open-source tools, while others combined open-source and commercial tools. Ricardo Araújo et al. [14] present a recent and detailed comparison of security scanners that are available for use by organizations with lower resources, such as small and medium-sized enterprises. In addition, they compared open-source tools and highlighted their performances according to the number of vulnerabilities detected and false positives.

Roy Shanto et al. [21] provide a survey that is based on taxonomy and a detailed overview of adversarial reconnaissance techniques. The taxonomy categorizes reconnaissance techniques based on the technical approach, including target footprinting, social engineering, network scanning, and local discovery.

Doupé et al. [22] assessed ten tools which are Acunetix, AppScan [18], BurpSuite [23], Arachni, Hailstorm, NTOSpider, Paros, N-Stalker, Webinspect, and W3af for web application assessment. These tools were selected to consider a variety of open-source and commercial scanners.

Lilan hu et al. [24] presented an in-depth study of the existing security vulnerability detection technology, combined with the development process of machine learning security vulnerability detection technology, the requirements of the security vulnerability detection model are analyzed in detail, and a cross-site scripting security vulnerability detection model for web application is designed and implemented. Based on the existing network vulnerability detection technology and tools, the verification code identification function is added, which solves the problem that the data can be submitted to the server only by inputting the verification code.

Jinfeng Li [25] synced industry-standard OWASP top 10 vulnerabilities and CWE/SANS top 25 dangerous software errors in a matrix with Checkmarx vulnerability queries, producing an application security framework that helps development teams review and address code vulnerabilities to minimize false positives discovered in static scans and penetration tests, targeting an increased accuracy of the findings. Mapping the OWASP/SANS with Checkmarx vulnerabilities queries, flaws and vulnerabilities are demonstrated to be mitigated with improved efficiency.

Amir Manzor et al. [26] presented a literature survey recapitulating security solutions and major vulnerabilities to promote further research by establishing a system of the existing methods on a bigger horizon. They suggested that there is no way to alleviate all the web vulnerabilities; therefore, further study is desirable in web information security. They also addressed the complexity of the surveyed methods and suggested some recommendations when using these methods.

Ouissem Ben Fredge et al. [27] identified the most critical web vulnerabilities according to OWASP Top 10, including their related attacks with their countermeasures. They also proposed an application that guarantees the protection of web applications against the most severe attacks and prevents some zero-day exploits.

Richard Amankwah et al. [28] provide a comparison work to determine the vulnerability detection capabilities of eight web vulnerability scanners (both open and commercial) using two vulnerable web applications: WebGoat and Damn vulnerable web application. The eight WVSs studied were: Acunetix; HP WebInspect; IBM AppScan; OWASP ZAP; Skipfish; Arachni; Vega; and Iron WASP. The performance was evaluated using multiple evaluation metrics: precision, recall, Youden index, OWASP web benchmark evaluation, and the web application security scanner evaluation criteria. The experimental results show that, while the commercial scanners effectively detect security vulnerabilities, some open-source scanners (such as ZAP and Skipfish) can also be effective.

The evaluation focused mainly on the selected web application assessment capabilities against XSS (Cross-Site Scripting), SQL injection, code injection, and broken access controls. Exposed vulnerabilities were categorized to enable the assessment process for seventeen different vulnerabilities. In addition, the authors developed their test application called WackoPicko [29], which is geared toward the assessment and evaluation process to test the scanner under realistic conditions. The results show that the running time of N-Stalker is higher than Acunetix, Webinspect, and BurpSuite, which provides competitive results for true positive vulnerabilities detection. The authors also noted that the crawling process is challenging and needs further investigation to improve the automated identification of vulnerabilities.

Bau et al. [30] evaluated eight commercial scanners, namely Acunetix WVS, Cenzic HailStrom Pro, HP WebInspect, IBM AppScan, McAfee SECURE, N-Stalker QA Edition, QualysGuard PCI, and Rapid7 NeXpose. In addition, they used black box scanning to test the scanners against a custom web application with known vulnerabilities. The vulnerability categories targeted in this study are XSS, SQL Injection, Secure Socket Layer (SSL), Session Management, Cross-Site Request Forgery (CSRF), SSL/Server Configuration, and Information Leakage. The results presented focused on the fact that all scanners successfully detected straightforward XSS and SQL injection vulnerabilities but failed to detect second-order (stored) SQL injection and XSS vulnerabilities.

Parvez et al. [31] evaluated two commercial scanners, Acunetix WVS and IBM AppScan, and one open-source scanner, OWASP ZAP. Their evaluation was performed against a custom web application with intentional vulnerabilities. This work focused on only two web application vulnerabilities, XSS and SQL Injection. The analysis revealed that the scanners show some improvement over previous studies in detecting second-order XSS and SQL Injection vulnerabilities.

Suteva et al. [32] evaluated six open-source scanners, namely NetSparker Community Edition, N-Stalker Free 2012, OWASP ZAP, W3AF, Iron WASP and Vega. The evaluation was performed against a vulnerable web application called WackoPicko [29]. The tested vulnerabilities were XSS, SQL Injection, Command Injection, File Inclusion, File Exposure, and several others. The total number of detected vulnerabilities and the number of false positives were identified. The results showed that NetSparker performed better than the other tested scanners.

A recent study by El Idrissi et al. [33] was conducted on eleven WAVSs (Web Application Vulnerability Scanners). Five were commercial tools, BurpSuite, Acunetix, Netsparker, AppSpider, and Arachni, and six were open-source tools: Wapiti, SkipFish, W3AF, IronWASP, ZAP and Vega. The scanners were evaluated against the WAVSEV application. The tested vulnerabilities were XSS, SQL Injection, Local and Remote File Inclusion, and Path Traversal. The results showed that most scanners had a better detection rate for XSS and SQL injection vulnerabilities than other vulnerabilities. The authors focused on comparing the performance of commercial and open-source tools. The results showed that some open-source tools such as ZAP and Vega have better results than other commercial tools such as AppSpider and Arachni.

Elahen et al. [34] assessed and ranked methodologies for User-Generated Content (UGC) on the web; their research provides supplementary content but varies in quality. They highlighted existing approaches and techniques for assessing and ranking UGC. Their survey consists of an efficient review for assessing and ranking UGC. The classification of frameworks used for assessment and ranking processes gives a systematic review of their functions. Conclusively, their research accomplishes the goal of assessing the healthcare information system domain by a comprehensive assessment of databases, networks, applications, and infrastructure.

Mohit et al. [35] proposed a model for web vulnerability detection based on probabilistic classification. Their research mainly focused on the top four OWASP vulnerabilities: CSS, SQL injection, Remote code execution, and File inclusion. The results were based on improving performance parameters such as false alarm and accuracy for the detection of the four mentioned vulnerabilities. The scope of their work is very limited, which addresses only four mentioned vulnerabilities on OWASP's list.

Gaurav et al. [36] presented a survey for monitoring web services using the cloud. It provides an assessment system for penetration testing for vulnerability and malware detection. They claimed that their proposed architecture was a cost-effective framework for web assessment using cloud services, which ensured better security for the web application, where web services were used in back-end communication.

Mehreen et al. [37] present a state-of-the-art survey for evaluations. They performed web application assessments based on a questionnaire with very positive results. The basic Human–Computer Interaction (HCI) design principle was used in questionnaire formation. However, survey evaluations might not be 100% accurate for usability testing; this method effectively collects user responses about the usability of the web application.

Ashikali and Divyakant [38] surveyed the literature to determine the major techniques for the Vulnerability Assessment and Penetration Testing (VAPT) process and suggested software and tools which can be helpful for the VAPT process. This approach audited web application security and analyzed the penetration testing process with its limitations. They discussed different tools with their features such as W3af, Havij, Fimap, Metasploit, Acunetix, and Nexpose. However, they did not perform a comparative analysis of these discussed tools.

Rawaa [39] performed an assessment of web scanner tools and evaluated the effectiveness and differences to find flaws and limitations in different web scanner tools. The evaluation of black box web application vulnerability scanners was focused on SQLI and XSS attacks due to their severity level. The author used open-source tools such as Paros, Wapiti, Skipfish, Nikto [40], Wfuzz, NetSparker, and HP webinspect.

Mark Rudolph [41] categorized web application security assessment tools in the context of their usage. According to their research, tool selection for web application security assessment depends on the process, web application's complexity, and environment in which they are being used. They provided an extensive review for web application penetration testing and assessment tool selection.

Fang et al. [42] presented a framework, "DarkHunter", that could categorize the fingerprints of automated tools and analyzed the payload features of different web assessment tools by using a deep learning algorithm. The five commercial and community web scanners chosen in this research are AppScan, AWVS, NetSparker, Vega, and W3af. Their proposed framework accuracy rate is 94.6% and recall rate is 95%. The limitation of this research is that the model cannot be dynamically enhanced. However, once the training is complete, the model is fixed.

Alsaleh et al. [43] published an article based on a comparative assessment of the performance of an open-source web vulnerability scanner. He evaluated four different open-source scanners with detection of FPR (False Positive Rate), TPR (True Positive Rates), FNR (False Negatives Rate), and TNR (True Negative Rates). The primary task of this work is to analyze the performance of open-source web application scanners in the context of correlation and detection capabilities. However, their comparative assessment did not present vital performance differences in assessed scanners.

Terry et al. [44] performed a HIS (Healthcare Information System) domain assessment within the domain and then automated the assessment process to ensure time and resource efficiency. Furthermore, automating the comprehensive assessment ensured that no steps could be skipped and that the assessment of the HIS is HIPAA (Health Insurance Portability and Accountability) compliant. This research is valuable as it highlights the problems faced in the healthcare industry that are not publicized and make healthcare systems entirely vulnerable within the domain. For this purpose, mostly open-source tools such as OpenVAS, Kismet, Aircrack, UCSniff, SQLMAP, and Wapiti are used.

There are a variety of tools discussed in this literature review. Although, a comparison and analysis summary of the literature review is presented in Table 1, some of these research works are limited to several detected vulnerabilities, and others just compared limited scanners. However, in this survey, 11 proprietary and open-source web application scanners have been compared with the detection capability of OWASP top 10 defined vulnerabilities.

Table 1. Comparison and analysis of literature review.

Researcher	Evaluated Scanner	Evaluated Vulnerabilities	Main Contributions	Limitation
Doupé et al. (2010) [22]	Acunetix 174, AppScan Burp, Grendel-Scan, Hailstorm, Milescan, N-Stalker, NTOSpider, Paros, W3af, Webinspect	Cross-Site Scripting, SQL Injection, Code, Injection, Broken Access Controls	accuracy, execution times and Threat scores	Vulnerabilities detection scope is very limited
Bau et al. (2010) [30]	Acunetix, Cenzic, McAfee SECURE, WebInspect, N-Stalker, QualysGuard	Cross-Site Scripting, SQL Injection, Cross Channel Scripting, Session Management, Cross-Site Request Forgery	Scanner footprint, Vulnerability detection, false positive	Scanners can detect straightforward XSS and SQL injection attacks, but failed to detect second order form of XSS and SQL injection attack
Parvez et al. (2015) [31]	Acunetix, Rational AppScan, ZAP	SQL Injection, Stored XSS	Detection rate for SQLI and XSS	Compare only two commercial scanners
Suteva et al. (2012) [32]	NetSparker, N-Stalker, OWASP-ZAP, W3af, Iron WASP, Vega	XSS, SQL Injection, Command Injection, File Inclusion	NetSparker has better detection rate then others	Linux, MAC, Windows
El Idrissi et al. (2017) [33]	BurpSuite, Acunetix, Netsparker, AppSpider, Arachni, Wapiti, SkipFish, W3AF, IronWASP, ZAP and Vega	XSS, SQL Injection, Local and Remote File Inclusion, Path Traversal	XSS, and SQL vulnerabilities have higher detection rate, Arachni perform better in open-source tools	Linux, MAC, Windows
Elahen, Claire et al. (2013) [34]	User Generated Content (UGC) evaluation	Assessing and Ranking UGC Assessment of healthcare systems	False Positive Rate	
Mohit et al. (2017) [35]	BurpSuite, Acunetix, Wapiti, SkipFish, Netsparker, W3AF, AppSpider, Arachni, ZAP, Vega	Cross site scripting, SQL injection, Remote code execution, File inclusion	Precision, Recall, and F-measure	Limited to false alarm and accuracy
Gaurav et al. (2018) [36]	Penetration system for malware detection and web assessment using cloud services	OWASP's Vulnerabilities	Cost-effective solution for web assessment	Limited to specific web applications
Mehreen et al. (2018) [37]			assessment based on questionnaire with positive results	No technical details are provided for assessment purposes
Ashikali et al. (2018) [38]	W3af, Havij, Fimap, Metasploit, Acunetix, Nexpose	OWASP'S Vulnerabilities	Major techniques for vulnerability assessment and penetration testing	Didn't perform any comparative analysis
Rawaa (2016) [39]	Paros, Wapiti, Skipfish, Nikto, Wfuzz, NetSparker, HP WebInspect	SQL Injection, Cross Site Scripting	Analytical comparison of six open-source tools	Very limited number of vulnerabilities are discussed
Mark, and Rudolph (2006) [41]	Commercial and Free/Open-source Tools		Benefits and Drawbacks of Selected Tools	Selected tools are not popular
Fang et al. (2018) [42]	AppScan, AWVS, Netspark, Vega, W3af		Extracted feature by using Convolutional Neural Network and accurately identify scanners	model cannot be dynamically enhanced
Alsaleh, Mansour et al. (2017) [43]	Arachni, Wapiti, Skipfish	SQL, Cross Site Scripting	Arachni scanner can test 100 percent of the SQL test cases	Not present vital performance difference between selected scanners
Terry et al. (2018) [44]	OpenVAS, Kismet, Aircrack, SQLMAP, Wapiti	Buffer overflow, Cross Site Scripting		Limited to Healthcare Information system only

3. Security Assessment for Web Application Tools

Web application assessment tools can be found as open-source and proprietary. Proprietary tools usually offer free trial packages for users and pen-testers; however, their features and functions are limited. There are several aspects involved in selecting one scanner over another. A scanner should:

- Support the protocols and authentication algorithms used by web applications.
- Support the main types of input delivery methods and be able to detect vulnerabilities in the web application with a low false-positive rate.
- Be within the technical abilities of the person who will use it.
- Be stable and regularly updated with the latest security updates to cover the ongoing vulnerabilities being discovered.
- Be selected whilst keeping its cost and licensing within the budget limit.

This research aims to evaluate and compare the scanner's capabilities to detect the true vulnerabilities in web applications based on multiple vulnerable web applications. In this research work, the scanners were evaluated against different vulnerable web applications to increase the granularity and variation of the detected vulnerabilities.

3.1. Analyzed Tools

The following is a brief representation of selected web application assessment tools.

Acunetix WVS: This is a leading web vulnerability scanner used to check web applications for vulnerabilities automatically. Acunetix WVS is used to discover web application security levels by crawling and analyzing the web application to exploit SQL injections, XSS, Host Header Injection and over 3000 other web vulnerabilities. Acunetix is a commercial tool that only works on Windows Operating Systems (OS). Older versions of Acunetix were evaluated.

Nessus: This is a vulnerability scanner with one of the largest knowledge bases of security vulnerabilities and hundreds of plugins that can be activated for detailed, customized scans. Nessus can detect security vulnerabilities in the OS of targeted hosts, patches, and services. It also demonstrates the ability to propose solutions that can mitigate these security vulnerabilities [45]. Nessus is a commercial tool that works on Linux, MAC OS, and Windows OS. For this work, a cloud-based version of the Nessus vulnerability scanner was evaluated. The Nessus vulnerability scanner was not evaluated in any of the previous works mentioned.

NetSparker: A web application security scanner is designed to discover and audit web application vulnerabilities such as SQL Injection and XSS possibilities. NetSparker is used by cybersecurity space professionals and is considered by many researchers [19]. However, NetSparker is a commercial tool that works on Windows OS only. Older versions of NetSparker were evaluated in [35].

APPSCAN: This tool provides centralized control with new advanced application scanning, remediation capabilities, enterprise application security status metrics, key regulatory compliance reporting, and seamless integration with AppScan Standard. It provides user administration capabilities for AppScan Source. AppScan Enterprise provides security teams with the security intelligence of the application needed to make informative risk-based decisions, including the capability to build a consolidated inventory of all application assets [46]. Assets can be described in terms of a list of attributes meaningful to the organization. Security specialists can query the portfolio to identify types of applications that match certain characteristics.

HP WebInspect: This is a powerful continuous diagnostics and mitigation tool. It can launch more than 3000 attacks on web systems and report back each time a vulnerability is exploited. WebInspect supports Java and .NET applications. WebInspect can create a virtual environment of an online bank system, so one can initiate scans and generate reports. The software itself does not require any special installation; a standard PC will be fine, though a dedicated workstation or server would likely yield results more quickly.

OWASP-ZAP: The OWASP Zed Attack Proxy (ZAP) is one of the most popular web application security testing tools. It is made available for free as an open-source project and is contributed to and maintained by OWASP [47]. The OWASP is a vendor-neutral, non-profit group of volunteers dedicated to making web applications more secure. The OWASP ZAP tool can be used during web application development by web developers or experienced security experts during penetration testing to assess web applications for vulnerabilities.

Wapiti: This is a generic command-line tool to automate the audit process of a web application [48]. It is an open-source tool and has some current updates. It can detect vulnerabilities such as Injection, CSS, Command Execution Attacks, CRLF Injection, and File Disclosure [49]. In addition to detecting mentioned vulnerabilities, Wapiti can execute some additional penetration functions such as finding potentially dangerous files on servers, finding configuration errors in .httaccess files that can lead to the security breach, and finding backup copies of the applications on the server that could compromise the security of said web applications if an attacker manages to get a hand on those files. It generates results in an HTML file.

It also carries out a black box testing technique (but does not read the source code) by crawling the web pages of the targeted web application. Moreover, it identifies scripts and forms to inject data. After obtaining the list of URLs, forms, and inputs, it conducts fuzzing to identify vulnerable scripts.

Arachni: This is an open-source, full-featured, modular, Ruby framework designed to assist penetration testers in scanning and evaluating web applications [50]. Arachni trains itself by learning from HTTP (HyperText Transfer Protocol) responses it receives during the scanning and testing process. Unlike other scanners, it detects the dynamic nature of web applications while crawling through paths of a web application’s cyclomatic complexity. This approach detects attack vectors more seamlessly. Arachni is a high-performance tool due to its asynchronous HTTP model. Therefore, its performance utterly depends on the hardware specification of the server and the available bandwidth of the network. There are also some limitations due to the server’s responsiveness underscan. It can support complicated web applications due to its integrated browser environment developed on technologies such as JavaScript, HTML5, and AJAX [51].

Nikto: This is an open-source web application scanner that can execute a scan for multiple items, including malicious files and programs, outdated versions of server machines, and version specific software in web servers. In addition, it performs scans on configuration files of web servers such as multiple index files, server fingerprinting, and HTTP calls settings. Its plugins are regularly updated for newly detected vulnerabilities. It is not a stealthy tool and performs scans swiftly; any IDS/IPS can detect nikto scans easily [52].

It has SSL and proxy support, and it generates reports in HTML, XML, and CSV format. It updates itself via the command line and detects server software with the help of headers, files, and favicons.

BurpSuite: This is a Java-based popular penetration testing toolkit created by PortSwigger web security for security experts. It helps to verify attack vectors and detects vulnerabilities directly affecting web applications. While browsing the target web application, it intercepts HTTP calls. It acts like an MITM (Man In The Middle) and captures and analyzes requests from the target web application server for malicious parameters and injection points [53].

It is freely available with limited features but highly capable functions. However, its fully-featured commercial and licensed versions are also available at an affordable price for professionals.

W3af: W3af (Web Application Attack and Audit Framework) detects and exploits web application vulnerabilities. It is a command-line open-source toolkit but is also available in the GUI (Graphical User Interface) package. This toolkit is also called the "Metasploit of the Web". It detects vulnerabilities using black box scanning techniques [54]. W3af core engine and plugins are written in Python, and it scans with more than 140 plugins that specifically exploit CSS, SQL injection, and remote file inclusion vulnerabilities. Its plugins share information while performing scans by using a knowledge base technique.

3.2. Reviewed Tools

Several other tools were not chosen but have been reviewed for different reasons, and these reasons are listed in Table 2. The category "Proxy" means that tools also act as a proxy. The term "Not Available" means that these tools are not available any more for download. The term "Did not Work" means that we could not obtain or install these programs. Finally, the term "Other" category covers various reasons for not choosing a particular tool.

Table 2. List of tools not chosen.

Tool Name	Not Available	Did Not Work	Proxy	Other
Prox Mon			✓	
Suru Web Proxy			✓	
Paros			✓	
WebScarab			✓	
SPIKE			✓	
XSSScan	✓			
HTMangLe	✓			
WebFuzz	✓			
ASP Auditor	✓			
WhiteAcid's XSS Assistant	✓			
screamingCobra	✓			
Overlong UTF	✓			
Web Hack Control Center	✓			
Web Text Converter	✓			
Grabber		✓		
Wfuzz		✓		
WSTool		✓		
Uber Web Security Scanner		✓		
Grendel-Scan		✓		
DirBuster		✓		
J-Baah		✓		
JBroFuzz		✓		
XSSFuzz				Only for CSS

Table 2. *Cont.*

Tool Name	Not Available	Did Not Work	Proxy	Other
WSFuzzer				Only for SOAF Services
RegFuzzer				Only for Regular Expressions
RFuzz				Only for HTTP requests
XSSFuzz				Only for CSS
Watcher				Depends on another application (Fiddler)
Falcove				Could not be Installed
NeXpose community edition				Not for Web Applications
SQLMap				Only for SQL Injection
X5S				Not Automated, Only for CSS

3.3. Evaluation of Web Application Scanner

Web application scanner evaluations are very important for designers and users. The fingerprint identification feature in web scanners helps to identify the attackers' automated tools by log files and network traffic. Several researchers pointed out that most cyber-attacks happen due to the misuse of scanners [55]. The traditional approach to solving such a problem is to match special characters in URL (Uniform Resource Locator) payloads or HTTP headers from scanners. The limitation of this approach is that attackers can modify special characters because scanning is a finite process [56]. Therefore, it is better to explore new technologies to improve fingerprint detection. Black box pen-testing technique is important for finding security vulnerabilities in an automated fashion. Black box pen-testing tools work in a point-and-shoot manner, assessing any web application independent from server-side language. However, black box tools have some limitations, especially when dealing with complex applications and multiple functionalities.

3.4. Evaluation Methodology and Criteria

In this subsection, the evaluation methodology has been explicitly defined. The black box penetration testing technique is used in this paper to evaluate open-source web application assessment tools. The real advantage of this approach is that it provides a uniform scenario for launched attacks. The black box approach evaluates web application scanners with no prior knowledge of web application internal structure. The major components of web application security perform three significant tasks: crawling, fuzzing, and reporting. For evaluation purposes, these three tasks are needed to be discussed.

The OWASP [13] is an international non-profit organization primarily focused on web application security. OWASP project regularly maintains an updated list of severe security vulnerabilities in web applications. Being an open-source solution, all of their material is freely available and easily accessible on the internet. This material includes documentation, tools, videos, and a forum for user guidance. Their popular project is the OWASP Top 10.

The OWASP Top 10 [13] is a frequently updated document focusing on web application security on the top ten critical risks. This document is composed of a team of security experts from all over the world. OWASP Top 10 is also known as "Awareness Document" it provides extensive statistical analysis of the most severe vulnerabilities in web applications. A list of Web application security vulnerabilities for OWASP Top 10 2021 [12] report is given below:

1. Broken Access Control;
2. Cryptographic Failures;
3. Injection;
4. Insecure Design;
5. Security Misconfiguration;
6. Vulnerable and Outdated Components;
7. Identification and Authentication Failures;
8. Software and Data Integrity Failures;

9. Security Logging and Monitoring Failures;
10. Server-Side Request Forgery.

4. Comparison of Selected Web Assessment Tools

The assessment of a web application scanner depends on the quality, and it depends on these characteristics. In this section, these three characteristics have been defined, which dictate the quality of the web application scanner. These are as follows:

- Quality of Crawling;
- Quality of Fuzzing;
- Quality of Analysis.

4.1. Quality of Crawling

Crawling is the process of identifying pages of the web application that are vulnerable to a certain attack. The number of pages crawled by a scanner determines its quality of crawling. Crawling is not dependent on vulnerability that is analyzed for or presented in the web application.

4.2. Quality of Fuzzing

The process of entering input to expose a vulnerability is called fuzzing. The quality of fuzzing depends on the inputs, and a fuzzer enters to find a certain vulnerability. Therefore, these inputs should exploit particular vulnerabilities regardless of web or SQL server.

4.3. Quality of Analyzing

It is the responsibility of the analyzer to analyze the results that the fuzzer generates. It should detect vulnerabilities without generating false positives.

This section presents the obtained results from different research reviews with our contributions using the following measures: vulnerabilities detected by each scanner, categorization of vulnerabilities based on their severity level, the classification of detected vulnerabilities, and accuracy of each scanner.

Moreover, DVWA (Damn Vulnerable Web Application) is a vulnerable web application used as a test bench [20]. It is a PHP MySQL web application that is thoroughly vulnerable. It is developed to test tools and professionals' skills in a legal environment. According to the survey, there is no accurate measure for detecting vulnerabilities. Therefore, there is a need to discover more vulnerabilities by using scripts and plugins in open-source tools. Results from proprietary tools such as Acunetix, APPSCAN, Nessus, NetSparker, and HP WebInspect were collected from different research studies. The results collection process aimed at a state-of-the-art comparison of these proprietary tools' performances on OWASP and NIST defined vulnerabilities. Open-source tools such as OWASP-ZAP, Burp Suite, Nikto, Arachni [51], and Wapiti [48] are tested on the DVWA framework. As a result, this framework is vulnerable to the following attacks: Brute Force Attack, Command Injection Attack, CSRF, File Inclusion, File Upload, Insecure Captcha, SQL injection, SQL Injection blind, CSS stored, CSS reflected, CSS-DOM, CSP (Content Security Policy) bypass, and Weak Session ID's. Most of these attacks' vulnerabilities are defined by OWASP listed in Table 3 [9]. Tables 4 and 5 classify and present the detection rate of each vulnerability for open-source and proprietary evaluated scanners. For example, as presented in Table 5, the detection rate of Acunetix is higher in the category of CSS, File Injection, Sensitive Data Exposure, CSRF, and Broken Authentication vulnerabilities. On the other hand, an open-source tool, OWASP-ZAP, in Table 4 has a lower detection rate in several categories such as CSS, File Injection, and Insecure Communication.

Table 3. Total number of vulnerabilities classified and detected by proprietary scanners.

Vulnerabilities	Acunetix	HP WebInspect	NetSparker	APPSCAN	Nessus
Broken Access Control	115	67	60	58	54
Cryptographic Failures	3	0	0	0	0
Injection	145	90	113	88	29
Insecure Design	2	2	1	0	1
Security Misconfiguration	51	30	15	25	5
Vulnerable and Outdated Components	23	27	25	21	15
Identification and Authentication Failures	165	180	140	130	100
Software and Data Integrity Failures	133	52	115	95	49
Security Logging and Monitoring Failures	12	15	11	10	15
Server-Side Request Forgery	65	95	55	45	52

Table 4. Total number of vulnerabilities classified and detected by open-source scanners.

Vulnerabilities	ZAP	Nikto	W3af	Wapiti	Arachni	BurpSuite
Broken Access Control	26	18	22	5	28	29
Cryptographic Failures	0	0	1	0	0	0
Injection	11	18	31	42	15	90
Insecure Design	11	0	0	1	1	1
Security Misconfiguration	8	4	11	18	17	12
Vulnerable and Outdated Components	0	10	13	2	27	27
Identification and Authentication Failures	743	545	411	223	180	179
Software and Data Integrity Failures	13	8	15	10	8	55
Security Logging and Monitoring Failures	4	6	8	0	13	6
Server-Side Request Forgery	36	43	26	39	25	52

Table 5. Precision rate of evaluated scanners.

Evaluated Scanner	CSS		SQLI		Precision
	TP	FP	TP	FP	
Acunetix	139	0	66	0	100%
IBM APPSCAN	6	5	49	0	84%
Nessus	200	21	43	5	90.88%
BurpSuite	136	3	62	0	98.5%
Wapiti	11	7	4	6	53%
Arachni	136	5	60	0	81.32%
WebInspect	8	7	11	17	44.1%
Nikto	9	2	11	6	71.4%
Netsparker	136	3	64	0	98.5%
W3af	81	3	19	3	94.3%
OWASP-ZAP	136	0	63	0	100%

4.4. Comparative Analysis of Proprietary Tools

- *Broken Access Control:* In the survey process of proprietary scanners, it was observed that Acunetix detected most of the broken access control vulnerabilities, and HP WebInspect detected 67 CSS vulnerabilities. Results of NetSparker and AppScan tools were similar. Nessus detected 54 vulnerabilities. Detection of broken access control vulnerability by Acunetix is far better than other proprietary tools.
- *Cryptographic Failures:* The Acunetix scanner only detected this vulnerability; in other scanners, we're unable to detect broken authentication vulnerability.
- *Injection:* Acunetix detected this vulnerability with 145 scores, NetSparker detected 113 vulnerabilities, HP WebInspect detected 90 vulnerabilities, AppScan detected 88 data exposure vulnerabilities, and in this comparison, Nessus detected only 29 vulnerabilities.

- *Insecure Design:* The detection score of this vulnerability is quite low; Acunetix and HP WebInspect detected only two vulnerabilities. NetSparker and Nessus detected only one vulnerability. AppScan could not find any malicious file inclusion vulnerability.
- *Security Misconfiguration:* Acunetix detected 51 vulnerabilities of this type, whereas HP WebInspect and AppScan detected 30 and 25 vulnerabilities.
- *Vulnerable and Outdated Components:* HP WebInspect detected more vulnerable and outdated components vulnerabilities than other proprietary tools with a 27 score. NetSparker detected 25, and Acunetix detected 23 vulnerabilities. AppScan detected 21, and Nessus detected 15 vulnerabilities.
- *Identification and Authentication Failures:* HP WebInspect scanner detected more identification and authentication failures vulnerabilities with a 180 score. Acunetix stood second with 165 scores. NetSparker detected 140 vulnerabilities, AppScan detected 130 vulnerabilities, and Nessus detected 100 vulnerabilities.
- *Software and Data Integrity Failures:* Acunetix scanner had detected more vulnerabilities of this type compared to other listed proprietary scanners. Acunetix detected 133 software and data integrity failure vulnerabilities. NetSparker detected 115, and AppScan detected 95 vulnerabilities. HP WebInspect detected 52, and Nessus detected 54 vulnerabilities. Therefore, the detection results of this vulnerability were good in Acunetix and NetSparker.
- *Security Logging and Monitoring Failures:* HP WebInspect and Nessus detected 15 vulnerabilities, Acunetix detected 12, NetSparker detected 11, and AppScan detected ten vulnerabilities.
- *Server-Side Request Forgery:* HP WebInspect detected 95 server-side request forgery vulnerabilities. Acunetix stood second with a 65 score. NetSparker detected 55, and Nessus detected 52 vulnerabilities. AppScan detected 45 vulnerabilities.

4.5. Comparative Analysis of Open-Source Tools

We assessed six well-known tools in an open-source web application security scanner tools: ZAP, Nikto, W3af, Wapiti, Arachni, and BurpSuite. We set the same parameters for the assessment purpose as used in proprietary tools assessment. We assessed these six tools in the DVWA environment, and the generated results were matched with proprietary tools results.

- *Broken Access Control:* In the open-source tools category, BurpSuite detected 29 of these vulnerabilities, Arachni detected 28 vulnerabilities, and ZAP and W3af detected 26 and 22 vulnerabilities, respectively. On the other hand, Wapiti detected only 5 of these types of vulnerabilities.
- *Cryptographic Failures:* W3af only detects this vulnerability in an open-source category. Other tools were unable to detect this vulnerability.
- *Injection:* BupSuite detected this vulnerability with the highest score. W3af detected 31 injection vulnerabilities. Nikto detected 18, Arachni detected 15, and ZAP detected 11 vulnerabilities.
- *Insecure Design:* ZAP detected 11 insecure design vulnerabilities; Nikto and W3af could not detect any vulnerability of this type. Wapiti and Arachni only detected one vulnerability.
- *Security Misconfiguration:* Wapiti and Arachni's score was 18 and 17, respectively. BurpSuite detected 12 vulnerabilities, W3af detected 11 vulnerabilities, and ZAP and Nikto detected 8 and 4 vulnerabilities.
- *Vulnerable and Outdated Components:* BurpSuite and Arachni detected 27 vulnerabilities. W3af detected 13, Nikto detected 10, and Wapiti detected only two vulnerabilities. ZAP could not detect any vulnerability.
- *Identification and Authentication Failures:* This vulnerability was detected with high scores as compared to other vulnerabilities. ZAP detected 743 vulnerabilities, Nikto detected 545 vulnerabilities, W3af score was 411, Wapiti detected 223 vulnerabilities, and Arachni detected 180 vulnerabilities.

- *Software and Data Integrity Failures:* BurpSuite detected 55 vulnerabilities, and W3af and ZAP detected 15 and 13 vulnerabilities. Arachi and Nikto detected eight vulnerabilities. Wapiti detected ten vulnerabilities. Comparatively, BurpSuite detected more data integrity vulnerabilities than other tools.
- *Security Logging and Monitoring Failures:* Arachni detected 13 vulnerabilities, and Burp-Suite and Nikto detected six vulnerabilities. W3af and ZAP detected 8 and 4 vulnerabilities, respectively. Wapiti could not detect any vulnerability of this type.
- *Server-Side Request Forgery:* BurpSuite detected 52 vulnerabilities of this type. Nikto detected 43, Wapiti detected 39, ZAP detected 36, W3af detected 26, and Arachni detected 25 server-side request forgery vulnerabilities.

As presented in Figure 1, Acunetix is ranked first, and NetSparker stood second in detecting high severity vulnerabilities. NetSparker stands second in detecting high severity vulnerabilities. In Figure 2, OWASP-ZAP has a high detection rate in the low severity vulnerabilities category. Open-source tools such as Burp Suite and Nikto depicted a high detection ratio of informational categories.

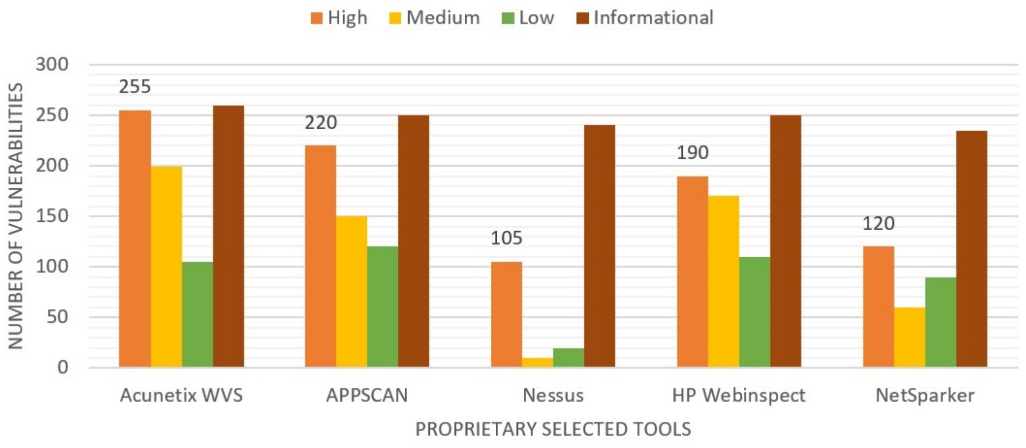

Figure 1. Detected Vulnerabilities on Proprietary Scanners.

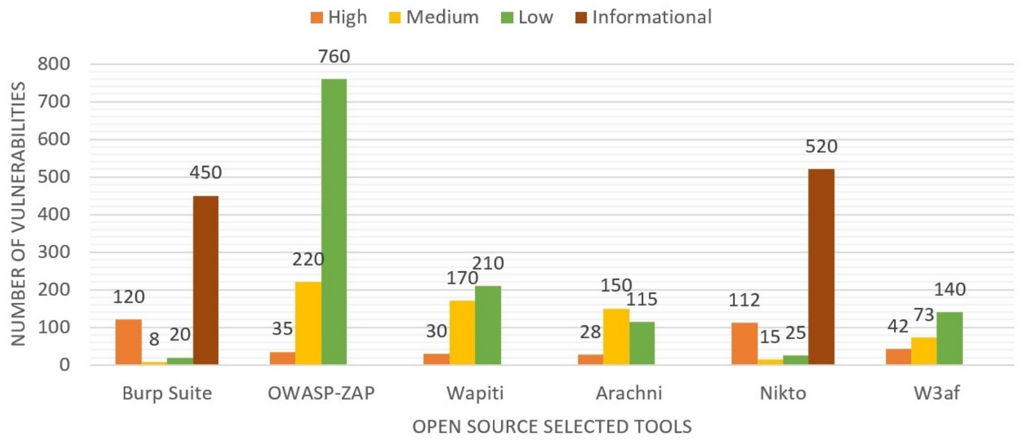

Figure 2. Detected vulnerabilities in open-source scanners.

A scanner could generate false-positive results in the black box evaluation of vulnerable web applications. In the context of black box testing, the scanners mostly rely on the information from service banners and signature matching checks, which leads to the false detection of vulnerabilities. Hence, it is important to verify the results manually by checking their validity and precision. Two approaches are used to validate web scanner obtained results. False Positive (FP) is referred to as a vulnerability that a scanner wrongly reported and a False Negative (FN), which is a vulnerability not detected or reported by a scanner. In this survey, we focused on false positive (FP) calculation.

4.5.1. Precision of Scanner

The precision of the scanner is measured as the percentage of correctly detected vulnerabilities or true positives (TP) to the total number of detected vulnerabilities, i.e., TP and false positives (FP). It is positively collected information, and the formula of precision can be presented as:

$$Precision = \frac{TP}{TP + FP}$$

Results of six evaluated scanners, the precisions of these open-source scanners for SQLI and CSS vulnerabilities are presented in Table 6.

Table 6. Recall rate of evaluated scanners.

Evaluated Scanner	CSS		SQLI		Precision
	TP	FN	TP	FN	
Acunetix	139	0	66	0	100%
IBM APPSCAN	6	0	49	0	100%
Nessus	200	0	43	0	100%
BurpSuite	136	1	62	0	99.2%
Wapiti	11	2	4	0	90%
Arachni	136	1	60	0	99.5%
WebInspect	8	0	11	0	100%
Nikto	9	1	11	0	99.5%
Netsparker	136	0	64	0	100%
W3af	81	0	19	0	100%
OWASP-ZAP	136	0	63	0	100%

4.5.2. Accuracy of Scanner

Accuracy is the percentage of the sum of correctly detected true positives and true negatives to the summation of all classified instances, i.e., True Positives (TP), False Positives (FP), False negatives (FN), and True negatives (TN). It depicts the closeness of the true value to the predicted value and can be calculated by the formula given below.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

4.5.3. Recall

The recall is the calculation of correctly identified positive instances that are really positive. It is also referred to as sensitivity. It can be calculated by the formula below, and a comparative analysis of the recall rate is provided in Table 6.

$$Recall = \frac{TP}{TP + FN}$$

4.6. Limitations

Penetration tools have some limitations as well, some of which are described here:

- These tools cannot detect all the vulnerabilities in web applications. Therefore, these tools are not efficient enough to find vulnerabilities such as information disclosure and encryption flaws.

- Access control flaws are harder to detect with these tools. Detecting hard-coded back doors or identifying multi cover attacks are difficult to detect [57].
- There are no specific goals associated with automated penetration testing tools. Therefore, these automated tools must check every possible flaw in web applications [58].

5. Our Contributions

As discussed in Section V, the quality of a web application scanner depends on three properties, Crawling, Fuzzing and Analyzing. First, crawling quality depends on the number of pages crawled by a scanner. Secondly, the quality of Fuzzing depends on the inputs that a fuzzer enters to find a certain vulnerability. During the development process of an indigenous scanner for detecting web application vulnerabilities, different open-source tools and technologies were used to develop a toolkit that can scan a given URL as a commercial scanner does. We focused on the crawling and detection process of web application scanning to enhance the performance of the developed toolkit.

After surveying different commercial and open-source web application scanning tools, it was assessed that different tools have different features and characteristics. For example, some scanners have better crawling techniques and others have better detection techniques. After analyzing the performance and accuracy of surveyed scanner, it was discovered that after improving the crawling and fuzzing properties of a scanner, the performance of these scanners could be improved.

Our first objective is to improve the crawler coverage of the scanner. An enhanced scanner coverage can be achieved by replacing the crawler component of a scanner with a better one. This technique might be helpful for some scanners but is not good for others, specifically if the integrated crawler works well. Therefore, an approach was chosen to integrate a better crawler component with other scanners. This newly integrated compiler provides additional URLs for a web application. To get the additional URLs of a web application, two approaches can be used: endpoint extraction from the source code of a web application and then using these detected URLs with an integrated crawler.

Endpoint extraction can be conducted by searching the web application underscan's source code (including configuration files). Using this approach, a crawler can uncover hidden parameters of requests (e.g., debugging parameters). However, this approach is only feasible when the source code of the web application under test is available. If it is not available, then the second approach comes into play. This approach consists of using the best available crawler to obtain additional URLs. Our survey assessed that Arachni provides better crawling performance compared to other open-source crawlers such as ZAP, W3af, and Wapiti.

We used the second approach to develop the web application scanner toolkit. To check the crawling coverage, WIVET (Web Input Vector Extractor Teaser) version 4 was used, which is a benchmarking project specifically designed to assess crawling coverage. Table 7 compares crawling coverage results of tools with default and integrated crawlers.

Table 7. Site coverage rate before and after using Arachni Crawler.

Scanner	Raw Coverage	Coverage when Seeded with Arachni Crawled Results
Arachni	95%	
OWASP-ZAP	68%	93.5%
W3af	22%	93.5%
Wapiti	49%	93%
Nikto	41%	93.5%

This table shows that raw crawling coverage of Arachni is already good, while ZAP, Wapiti, and W3af find half of all resources. However, reliably performing authenticated scans is difficult due to some reasons. For example, it finds difficulties with various authentication methods, preventing logouts during scans, and re-authenticating sessions (e.g., when a web application with integrated protection mechanisms invalidates the authenticated session when scanned). Therefore, some scanners have difficulties with the authentication process.

To cope with these difficulties, a module OAuth v 2.0 is implemented It provides a wizard to configure authentication requests, store the authentication cookies received from web applications, and injects them into subsequent requests from the scanner to ensure that requests are requested handled as authenticated one by the web application. Moreover, Arachni uses PhantomJS, a headless browser, and it extracts more information by allowing request and response information from HTTP. One can use only the crawler option by using the switch(-checks=-) option in the Arachni command to not load any checks or (–checks=trainer) to load the trainer, enabling the crawling function. URLs generated after crawling from a web application under test are used as seed values for other web scanners.

Figure 3 presents a graphical representation of old and new detected vulnerabilities by using an integrated crawler. Results have been significantly improved by using the Arachni crawler for generating URLs lists with the OAuth authentication framework. In the next step, these URLs are provided as a seed value to other open-source scanners (ZAP, W3af, and Wapiti), and their vulnerabilities and detection capabilities have been improved.

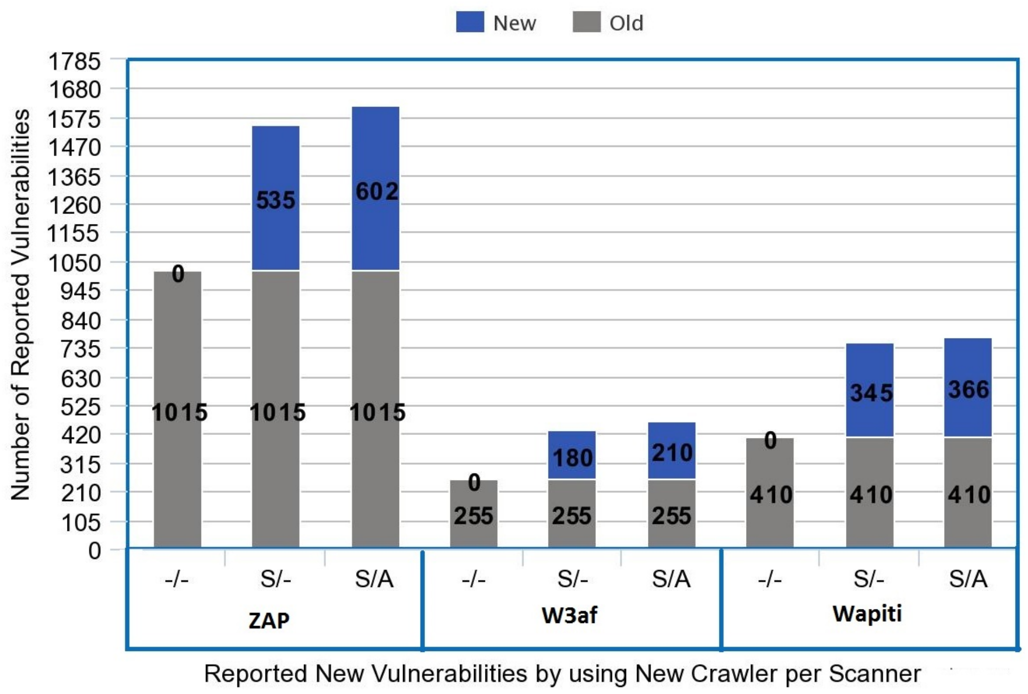

Figure 3. Reported vulnerabilities using before and after Arachni crawler.

Table 8 depicts the configurations settings for the executed scans. In configuration (-/-), we set the scanner without any seed values, i.e., no URLs list is provided by using the Arachni crawler and no authentication framework is used. Scanners are used with their default settings. In configuration (S/-), seed values, i.e., URLs' list, are provided by using Arachni crawler to another scanner but without any authentication framework. In configuration (S/A), the seed value and authentication framework are used to obtain better detection results.

Table 8. Configurations used during the evaluation.

Config.	Scans Executed
-/-	Without Seeding and Authentication Configuration
S/-	With Seeding and Non-Authentication Configuration
S/A	With Seeding and Authentication configuration (using both technical solutions)

Moreover, most open-source tools are available in a desktop application form or installed within the application server. This toolkit is available in desktop applications and can be accessed through a web link. A pen-tester can install a desktop application of this toolkit on its machine and scan a given URL like the Acunetix web scanner provides this functionality.

The detection results have also been improved by using a new crawler with ZAP, W3af, and Wapiti. Figure 3 depicts that the detection rate of vulnerabilities has been increased significantly.

6. Open Challenges/Future Directions

While performing penetration testing, automated tools work swiftly. Contrary to this, a manual approach needs more time than an automated approach. Reconnaissance and information gathering related tasks can also be conducted with automated tools. Therefore, in such scenarios, automated tools can be used for reconnaissance and scanning purposes to find the right target for further vulnerability exploitation. Automated testing using open-source and proprietary tools is much more effective because it covers all the aspects. We cannot be sure of the vulnerabilities it detects by using these automated tools.

6.1. Automated Scripts Generation

With automated tools, the false-positive rates are appeared to be high. Some manual analysis is required to confirm the reported vulnerabilities. Even in cases where the size of the application is large, an automated security scan comes in handy. However, the result given by the automated tool is not necessarily the conclusion. Manual analysis is often required to confirm the vulnerabilities. Manual techniques are also helpful in finding business logic flaws. Therefore, there is a need to research manual testing techniques that make this process less time consuming and convenient for a pen-tester. Research on automatic script generation for exploitation purposes can make manual testing easier.

6.2. Lower False Positives in Automation

When a new vulnerability/exploit is released, most of the automated tools have to wait for the next update to use it in their tests, whereas a human can learn about a new technique and implement it the very next day. However, this again requires an expert. Updated automated tools are reliable to use. If a zero-day vulnerability has been detected in the environment, it is cumbersome to detect and identify security threats. A tester can write an exploit in manual testing depending on the vulnerability. This leads to a more comprehensive testing methodology with lower false positives than most automated tools cannot perform. Sometimes in automated tools, exploitation of the detected vulnerability is not possible due to false positives. Therefore, another open challenge is the deduction of false positives in automated tools to obtain more accurate results.

6.3. Insufficient Skills for Exploitation

Automated tools generate false positives and negatives (10 to 70% depending on methodology and product) that present insufficient security requirements. Moreover, insufficient capabilities of tools also cause inaccuracies. It all depends on the expertise of a manual tester that uses automated tools, validates the generated results, and determines the true security findings. We cannot be sure or rely fully on these tools whether the vulnerabilities they are detecting are vulnerabilities that exist in the system or not. Automated tools are not good for penetration testing of logical vulnerabilities of the system. Automation is enabled to determine that the application is completely assessed from the perspective of security loopholes. Several vulnerabilities, for example, CSRF (Cross-Site Request Forgery) and source code logical errors, require skilful certified security experts to identify and exploit security loopholes.

We concluded for future work to mature web vulnerability scanners for the minimization of false-positive detection rate. False-positive vulnerabilities mostly appear at a high rate by using automated tools that may result in improper security evaluation of the target websites. Moreover, the automation process cannot determine that an application is completely tested from all types of security loopholes. Therefore, these vulnerability scanners should have a proper mechanism for understanding the limitation of the application to determine security concerns. Further, we know that automated tools are only as reliable as their updates. Not timely updates of tools may result in not finding the latest thread and vulnerabilities of websites. Therefore, all tools must ensure their regular update to detect Zero-day and latest vulnerabilities. One other aspect we see is that remediation of the tool should be in more detail. Remediation of found vulnerability should include technical details and codes for patching that vulnerability. It will assist the Penetration tester much more if patching of vulnerabilities is a concern. Every tool cannot detect all vulnerabilities present in a web application; therefore, there is a need for mutual compliance to detect vulnerabilities detected by other tools.

7. Conclusions

The assessment of web applications contains many actions that aim to increase the overall security and robustness against different cyber-attacks. Developers and pen-testers use different tools to scan web applications and detect every possible vulnerability dynamically. In this paper, eleven web application assessment tools are surveyed and evaluated with different capabilities by intentionally scanning web applications such as DVWA. The selected web application assessment tools were the top listed scanners in 2019. The web application scanning tools are evaluated using measures such as detection rate accuracy, precision, capability to detect different vulnerabilities, and their severity level. According to some research papers and our assessment, OWASP-ZAP has a higher vulnerability detection rate in the open-source tools category; however, others say Acunetix and NetSparker have fewer false-positive rates, i.e., they have a better vulnerability identification capability in the proprietary tools category. Moreover, this survey provides compared results of different detected vulnerabilities. A comparative analysis will be performed by developing plugin scripts and algorithms for open-source web assessment tools. This approach will help to increase crawling coverage with better authentication features and helps to decrease false positives.

In the future, other web application assessment tools can also be compared with these tools for a better false positive detection rate. Some of these tools can be customized in terms of their features and functions to detect web application vulnerabilities better. Open-source tools such as Nikto and Arachni use scripts and plugins for vulnerability detection. Therefore, more plugins and scripts are developed to detect more significant vulnerabilities. All tools cannot generate the same results, and if one tool can detect any potential vulnerability that others cannot detect, it can be customized and developed in such a way that it can detect the same kind of vulnerability.

Author Contributions: Conceptualization, J.S.; methodology, J.S.; writing—original draft preparation, J.S., M.K.H. and I.T.J.; writing—review and editing, I.T.J., K.N.Q., M.A. and N.C.; funding acquisition, N.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ivanov, D.; Kalinin, M.; Krundyshev, V.; Orel, E. Automatic security management of smart infrastructures using attack graph and risk analysis. In Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 27–28 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 295–300.
2. Jyothi, P.D.; Lakshmy, K. Vuln-Check: A Static Analyzer Framework for Security Parameters in Web. In *Soft Computing and Signal Processing*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 233–247.
3. Bhorkar, G. Security Analysis of an Operations Support System. 2017. Available online: https://aaltodoc.aalto.fi/handle/123456789/29252 (accessed on 12 April 2022).
4. Seng, L.K.; Ithnin, N.; Said, S.Z.M. The approaches to quantify web application security scanners quality: A review. *Int. J. Adv. Comput. Res.* **2018**, *8*, 285–312. [CrossRef]
5. Holik, F.; Horalek, J.; Marik, O.; Neradova, S.; Zitta, S. Effective penetration testing with Metasploit framework and methodologies. In Proceedings of the 2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 19–21 November 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 237–242.
6. Stasinopoulos, A.; Ntantogian, C.; Xenakis, C. *Commix: Detecting and Exploiting Command Injection Flaws*; White Paper; Department of Digital Systems, University of Piraeus: Piraeus, Greece, 2015.
7. Grossman, J. Whitehat website security statistics report. *Retrieved March* **2007**, *8*, 2010.
8. Symantec. Welcome to Integrated Cyber Defense. Available online: https://docs.broadcom.com/docs/integrated-cyber-defense-reference-guide (accessed on 12 April 2022).
9. Qasaimeh, M.; Ala'a, S.; Khairallah, T. Black box evaluation of web application scanners: Standards mapping approach. *J. Theor. Appl. Inf. Technol.* **2018**, *96*, 4584–4596.
10. Babovic, Z.B.; Protic, J.; Milutinovic, V. Web Performance Evaluation for Internet of Things Applications. *IEEE Access* **2016**, *4*, 6974–6992. [CrossRef]
11. Nidhra, S.; Dondeti, J. Black box and white box testing techniques—A literature review. *Int. J. Embed. Syst. Appl. (IJESA)* **2012**, *2*, 29–50. [CrossRef]
12. OWASP. Welcome to the OWASP Top 10—2021. 2021. Available online: https://owasp.org/Top10/ (accessed on 12 April 2022).
13. OWASP. OWASP Foundation. Available online: https://owasp.org/ (accessed on 12 April 2022).
14. Araújo, R.; Pinto, A.; Pinto, P. A Performance Assessment of Free-to-Use Vulnerability Scanners-Revisited. In Proceedings of the IFIP International Conference on ICT Systems Security and Privacy Protection, Oslo, Norway, 22–24 June 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 53–65.
15. Acunetiex. Acunetix Web Vulnerability Scanner. 2019. Available online: https://www.acunetix.com/?ab=v1 (accessed on 12 April 2019).
16. Tenable. Nessus Vulnerability Scanner. 2011. Available online: https://www.tenable.com/products/nessus (accessed on 12 April 2022).
17. HP. HP WebInspect. 2011. Available online: http://www.hp.com/hpinfo/newsroom/press_kits/2011/risk2011/HP_WebInspect_data_sheet.pdf (accessed on 12 April 2022).
18. IBM. IBM AppScan. 2013. Available online: https://www.ibm.com/support/pages/sites/default/files/inline-files/$$$FILE/AppScanUserGuide_0.pdf (accessed on 12 April 2022).
19. Netsparker. Netsparker Web Application Security Solution. 2019. Available online: https://www.invicti.com/ (accessed on 24 June 2019).
20. DVWA. Damn Vulnerable Web Application. Available online: https://dvwa.co.uk/ (accessed on 24 June 2019).
21. Roy, S.; Sharmin, N.; Acosta, J.C.; Kiekintveld, C.; Laszka, A. Survey and Taxonomy of Adversarial Reconnaissance Techniques. *arXiv* **2021**, arXiv:2105.04749.
22. Doupé, A.; Cova, M.; Vigna, G. Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Bonn, Germany, 8–9 July 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 111–131.
23. PortSwigger. Burp Suite Scanner. Available online: https://portswigger.net/burp/vulnerability-scanner (accessed on 12 April 2022).
24. Hu, L.; Chang, J.; Chen, Z.; Hou, B. Web application vulnerability detection method based on machine learning. In *Journal of Physics: Conference Series*; IOP Publishing: Guiyang, China, 2021; Volume 1827, p. 012061.
25. Li, J. Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST). *Ann. Emerg. Technol. Comput. (AETiC)* **2020**, *4*, 1–8. [CrossRef]

26. Noman, M.; Iqbal, M.; Manzoor, A. A Survey on Detection and Prevention of Web Vulnerabilities. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 521–540. [CrossRef]
27. Hamam, H.; Derhab, A. An OWASP top ten driven survey on web application protection methods. In Proceedings of the Risks and Security of Internet and Systems: 15th International Conference, CRiSIS 2020, Paris, France, 4–6 November 2020; Revised Selected Papers; Springer Nature: Berlin/Heidelberg, Germany, 2021; Volume 12528, p. 235.
28. Amankwah, R.; Chen, J.; Kudjo, P.K.; Towey, D. An empirical comparison of commercial and open-source web vulnerability scanners. *Softw. Pract. Exp.* **2020**, *50*, 1842–1857. [CrossRef]
29. Aldeid. WackoPicko. Available online: https://github.com/adamdoupe/WackoPicko (accessed on 12 April 2022).
30. Bau, J.; Bursztein, E.; Gupta, D.; Mitchell, J. State of the art: Automated black-box web application vulnerability testing. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Berleley/Oakland, CA, USA, 16–19 May 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 332–345.
31. Parvez, M.; Zavarsky, P.; Khoury, N. Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities. In Proceedings of the 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 14–16 December 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 186–191.
32. Suteva, N.; Zlatkovski, D.; Mileva, A. Evaluation and testing of several free/open source web vulnerability scanners 2013. In Proceedings of the 10th Conference for Informatics and Information Technology (CIIT 2013), Bitola, Macedonia, 18–21 April 2013.
33. Idrissi, S.; Berbiche, N.; Guerouate, F.; Shibi, M. Performance evaluation of web application security scanners for prevention and protection against vulnerabilities. *Int. J. Appl. Eng. Res.* **2017**, *12*, 11068–11076.
34. Momeni, E.; Cardie, C.; Diakopoulos, N. A survey on assessment and ranking methodologies for user-generated content on the web. *ACM Comput. Surv. (CSUR)* **2016**, *48*, 41. [CrossRef]
35. Kumar, M.; Majithia, S.; Bhushan, S. An Efficient Model for Web Vulnerabilities Detection based on Probabilistic Classification. *Int. J. Technol. Comput. (IJTC). Techlive Solut.* **2016**, 50–58. Available online: https://www.semanticscholar.org/paper/An-Efficient-Model-for-Web-Vulnerabilities-based-on-Kumar-Majithia/f09ddc0501358e234a5f8e9ebec359beb91db8f1 (accessed on 12 April 2022).
36. Raj, G.; Mahajan, M.; Singh, D. Security Testing for Monitoring Web Service using Cloud. In Proceedings of the 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), Paris, France, 22–23 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 316–321.
37. Ahmed, M.; Adil, M.; Latif, S. Web application prototype: State-of-art survey evaluation. In Proceedings of the 2015 National Software Engineering Conference (NSEC), Rawalpindi, Pakistan, 17 December 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 19–24.
38. Hasan, A.; Meva, D. Web Application Safety by Penetration Testing. *Int. J. Adv. Stud. Sci. Res.* **2018**, *3*. Available online: https://www.academia.edu/38248493/Web_Application_Safety_by_Penetration_Testing (accessed on 12 April 2022).
39. Mohammed, R. Assessment of Web Scanner Tools. *Int. J. Comput. Appl.* **2016**, *133*, 1–4. [CrossRef]
40. InfoSec. Introduction to the Nikto Web Application Vulnerability Scanner. Available online: https://resources.infosecinstitute.com/topic/introduction-nikto-web-application-vulnerability-scanner/ (accessed on 12 April 2022).
41. Curphey, M.; Arawo, R. Web application security assessment tools. *IEEE Secur. Priv.* **2006**, *4*, 32–41. [CrossRef]
42. Fang, Y.; Long, X.; Liu, L.; Huang, C. DarkHunter: A Fingerprint Recognition Model for Web Automated Scanners Based on CNN. In Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, Guiyang, China, 16–19 March 2018; ACM: New York, NY, USA, 2018; pp. 10–15.
43. Alsaleh, M.; Alomar, N.; Alshreef, M.; Alarifi, A.; Al-Salman, A. Performance-based comparative assessment of open source web vulnerability scanners. *Secur. Commun. Netw.* **2017**. Available online: https://www.hindawi.com/journals/scn/2017/6158107 (accessed on 12 April 2022).
44. Terry, M.; Oigiagbe, O.D. A comprehensive security assessment toolkit for healthcare systems. *Colon. Acad. Alliance Undergrad. Res. J.* **2015**, *4*, 1–6.
45. Black, P.E.; Fong, E.; Okun, V.; Gaucher, R. Software assurance tools: Web application security scanner functional specification version 1.0. *Special Publication* **2008**. Available online: https://www.govinfo.gov/content/pkg/GOVPUB-C13-4be4dc7c71623dd4fdc2be6167bc769e/pdf/GOVPUB-C13-4be4dc7c71623dd4fdc2be6167bc769e.pdf (accessed on 12 April 2022).
46. Introduction to AppScan Enterprise. Available online: https://www.ibm.com/docs/en/dsm?topic=guide-appscan-enterprise-scanner-overview (accessed on 12 April 2019).
47. OWASP Zed Attack Proxy Project. Available online: https://owasp.org/www-project-zap/ (accessed on 12 April 2022).
48. Automated Audit Using WAPITI. 2016. Available online: https://owasp.org/www-community/Automated_Audit_using_WAPITI (accessed on 12 April 2022).
49. Wapiti—The Black Box Vulnerability Scanner for Web Applications. Available online: https://wapiti-scanner.github.io (accessed on 12 April 2019).
50. Mukhopadhyay, I.; Goswami, S.; Mandal, E. Web penetration testing using nessus and metasploit tool. *IOSR J. Comput. Eng.* **2014**, *16*, 126–129. [CrossRef]
51. Arachni, Web Application Security Scanner Framework. 2017. Available online: https://www.arachni-scanner.com/ (accessed on 12 April 2019).
52. CIRT.net. Nikto2. 2019. Available online: https://cirt.net/ (accessed on 12 April 2022).
53. Geek, P. What is Burp Suite. 2019. Available online: https://portswigger.net/burp (accessed on 1 December 2019).

54. Tools, P.T. w3af Package Description. 2019. Available online: http://docs.w3af.org/en/latest/install.html (accessed on 1 December 2019).
55. Catakoglu, O.; Balduzzi, M.; Balzarotti, D. Attacks landscape in the dark side of the web. In Proceedings of the Symposium on Applied Computing, Marrakech, Morocco, 3–7 April 2017; ACM: New York, NY, USA, 2017; pp. 1739–1746.
56. Doupé, A.; Cavedon, L.; Kruegel, C.; Vigna, G. Enemy of the state: A state-aware black-box web vulnerability scanner. In Proceedings of the 21st USENIX Security Symposium (USENIX Security 12), Bellevue, WA, USA, 8–10 August 2012; pp. 523–538.
57. Sutton, M.; Greene, A.; Amini, P. *Fuzzing: Brute Force Vulnerability Discovery*; Pearson Education: London, UK, 2007.
58. McAllister, S.; Kirda, E.; Kruegel, C. Leveraging user interactions for in-depth testing of web applications. In Proceedings of the International Workshop on Recent Advances in Intrusion Detection, Cambridge, MA, USA, 15–17 September 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 191–210.