Received January 18, 2022, accepted February 11, 2022, date of publication March 22, 2022, date of current version March 30, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3161522

A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners

SULIMAN ALAZMI[1,2], (Member, IEEE), AND DANIEL CONTE DE LEON[1], (Member, IEEE)
[1]Department of Computer Science, University of Idaho, Moscow, ID 83844, USA
[2]Department of Computer Science, Majmaah University, Riyadh 11362, Saudi Arabia

Corresponding author: Suliman Alazmi (alaz2105@vandals.uidaho.edu)

The work of Suliman Alazmi was supported by Majmaah University, Saudi Arabia. The work of Daniel Conte de Leon was supported by the University of Idaho, USA.

ABSTRACT Web applications have been a significant target for successful security breaches in the last few years. They are currently secured, as a primary method, by searching for their vulnerabilities with specialized tools referred to as Web Application Vulnerability Scanners (WVS's). Although, these dynamic approaches of testing have some advantages, there is still a scarcity of studies that explore their features and detection capabilities in a systematic way. This article reports findings from a Systematic Literature Review (SLR) to look into the characteristics and effectiveness of the most frequently used WVS's. A total of 90 research papers were carefully evaluated. Thirty (30) WVS's were collected and reported, with only 12 having at least one quantitative assessment of effectiveness. These 12 WVS's were evaluated by 15 original evaluation studies. We found that these evaluations tested mostly only two of the Open Web Application Security Project (OWASP) Top Ten vulnerability types: SQL injection (SQLi) (13/15) and Cross-Site Scripting (XSS) (8/15). Additionally, only one work evaluated six of the OWASP Top Ten vulnerability types and for only one scanner. We also found that the reported detection rates were highly dissimilar between these 15 evaluations. Based on these surprising results we suggest avenues for future directions.

INDEX TERMS Web applications, black-box testing, web vulnerability scanner, effectiveness and performance, OWASP top ten, detection rate.

I. INTRODUCTION

Web application vulnerabilities such as SQL injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) are becoming common and widely reported nowadays. These vulnerabilities give attackers unauthorized access to sensitive information, such as credit card data, accounts, and medical information. Approaches for identifying vulnerabilities in web applications are classified as either Black-box testing or White-box. They are also known as Dynamic Application Security Testing (DAST) or Static Application Security Testing (SAST). White-box testing is used by security consultants who are well versed in different programming languages, creating algorithms, and skilled at inspecting application code [25]. On the other hand, Black-box testing is used by cyber security professionals who are experts in the different technologies, skilled in analyzing user-supplied input, and capable of thinking outside the box. Black-box testing is the most common approach used for identifying vulnerabilities by testing them dynamically [6]. Tools that use a dynamic or Black-box testing approach are usually called Web Vulnerability Scanners (WVS's). Software developers and cyber security experts use these scanners to find vulnerabilities in web applications. Theses scanners have the capability of automatically evaluating the security of web applications with minimal human intervention and are usually marketed as 'point-and-click pentesting' tools [25]. Many WVS's, both commercial and free/open source, are available to help developers and security analysts discover vulnerabilities in web applications. However, these scanners vary in their technical features and vulnerability detection performance. Therefore, the selection of WVS should be based on various factors such as scanner characteristics, availability of documentations and the capability of the scanner to detect vulnerabilities. To the best of our knowledge and date, the present article is

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

the most comprehensive systematic literature review on the effectiveness and characteristics of WVS's. The contributions of this study are the following:

1) This survey performed a systematic literature review of published studies about WVS's. It includes four search engines and four key phrases for a total of 16 searches.
2) The returned publications were then analyzed and classified based on their type of contributions - methodology, approach, evaluation, or survey.
3) Each of the acquired web vulnerability scanners (total 30) were classified based on seven different characteristics: number of citations, license type, last update date, scanner technology, run platform, user interface type, documentation availability, and capability of detecting the OWASP Top Ten vulnerabilities.
4) Data was then collected and tallied regarding the reported effectiveness and vulnerability type detection rate of web vulnerability scanners as reported by the obtained (total 15) evaluation studies.
5) All details of the researchers' method, data, and findings were presented in tables and graphs within this article to make it a complete, self-contained, and state-of-the-art account on web vulnerability scanners.

The rest of this article is organized as follows: Section II: Background introduces web application vulnerability types and web application scanner types and approaches; Section III Related Work, describes related but less comprehensive surveys; Section IV: Research Questions; Section V: Research Methods, describes our systematic approach to this literature search; Section VI Results and Findings, describes in detail the findings and results; Section VII: Discussion, analyzes the results collected by the study.

II. BACKGROUND

A. WEB APPLICATION VULNERABILITIES

The vulnerability can be described as the security flaw which silently exists in a software program or application. Attackers or malicious entities try to exploit the vulnerability to gain illegal access to data owned by the application. The whole exploitation activity includes three actors: the application itself (victim), the implementation of the attack to compromise the application (attack), and the entity that carries the attack (attacker) [63]. OWASP Top 10 2010, OWASP Top 10 2013, OWASP Top 10 2017 and OWASP Top 10 2021 are detailed in Table 1.

B. BLACK-BOX WEB APPLICATION SECURITY TESTING

For many years, the most commonly used approach for testing a web application is Black-box security testing. This strategy serves as a technique to test the running web applications to discover security vulnerabilities and loopholes without priory knowledge of the application's internal coding. Typically, the testing team would be considered users of the application as they are provided with valid access to the user account, where the tester acts like an attacker to find out vulnerabilities and flaws in tested web applications [1].

Adam *et al.* [1] defined four phases for a Black-box vulnerability test:

1) **Planning phase:** The rules and objectives for the test can be set in this phase.
2) **Discovery phase:** This phase is divided into two stages. The first stage includes the initiation of the test and the collection of information. The second stage performs vulnerability analysis, which occurs after the attack phase.
3) **Attack phase:** This phase examines the various vulnerabilities in the target application, which is also known as "the heart of the test".
4) **Reporting phase:** This phase provides documentation with a combination of other phases.

An assessment plan is developed in the planning phase, whereas the discovery and attack phases involve recording and periodical reporting of events to the director. Finally, a report is presented which explains known vulnerabilities, the ranking of risks, and tips for the improvement of the recognized weaknesses [16].

C. BLACK-BOX WEB APPLICATION SECURITY TESTING ADVANTAGES

- Consistent: The Black-box testing is capable of showing the consistencies or inconsistencies of the system's requirements specifications [87].
- Simple: The tester does not have to deal with the tested system's internal structure or code, so the tester does not face many difficulties in performing the Black-box testing. It merely involves examining the inputs and outputs of the tested system, so it is not imperative to have in-depth internal knowledge of the system. Moreover, the source code also does not need to be accessible for conducting the test [87].
- Rapid: Black-box tests do not require a long time for its preparation because the tester is not required to have full knowledge of the system in question. These tests follow the user paths, which are limited in relatively small systems [87].
- Impartial: Black-box tests show the result whether or not the system works. Rather than the "developer" point of view, the tests are taken from a user point of view, creating independence for each party [87].

D. BLACK-BOX WEB APPLICATION SECURITY TESTING DISADVANTAGES

- It is hard to make precise test cases without exact specifications [87].
- It is not easy to distinguish potential and practicable inputs in constrained testing time.
- It is possible that the already executed tests may be re-performed by the coder [87].
- In this testing, several areas of the program may remain untested [87].
- It is difficult to insure covering all functionality of web application [87].

TABLE 1. Merging of the OWASP Top 10 Vulnerability Types for 2010, 2013, 2017 and 2021. Columns labeled 10, 13, and 17 correspond to the OWASP Top Ten Vulnerability Types for 2010, 2013, 2017 and 2021 respectively.

No	Vulnerability Type	10	13	17	21	Description
1	Injection	✓	✓	✓	✓	The injection is a vulnerability in the application that allows user input data to be taken as a command. This allows the system to execute the unwanted command without requisite authorization and exposes the database to the attacker [63], [87].
2	Cross-Site Scripting (XSS)	✓	✓	✓		Cross-site scripting (XSS) is a computer vulnerability of web applications. It injects the attacker's script into the web application. When the user opens the web page, the script gets executed, allowing the attacker to change the web page,deface the web page or redirect the user to bad sites [63], [87], [65].
3	Cross-Site Request Forgery (CSRF)	✓	✓			Cross-Site Request Forgery (CSRF) tricks the user into taking undesired action after the user logs into an application. The user is tricked by malicious email or link that sends the forged request to the web server. The attack may change the user's password or transfer money from his account or steal data etc. [63], [87], [65].
4	Broken Authentication and Session Management	✓	✓	✓		This vulnerability occurs due to improper security implementation of the web application as well as users' carelessness in the use of the system. In this, the attackers get access to the user's credentials if these are stored without encryption or the user is careless about the password, closes the browser without logging out, etc. [63], [65], [87].
5	Broken Access control			✓	✓	This vulnerability happens due to improper implementation of access control of the website. It allows the attacker to get into the system where regular access is not permitted. Having gained access, attackers can steal information and create havoc [63], [65], [87].
6	Insecure Direct Object References	✓	✓			The websites allow the user to access the system by putting a direct reference to the database in the URL. Insecure Direct Object References is a vulnerability of the system that enables an attacker to modify the URL containing direct reference to the database and access the location that regular user is not authorized to access [63], [65], [87].
7	Missing Function Level Access Control		✓			This vulnerability happens due to improper functional level access rights given by developers to the user. The attacker changes the URL to access the unauthorized area by trial and error. This allows attackers to get inside the application without proper authorization if proper protection is not in place [63], [65], [87].
8	Security Misconfiguration	✓	✓	✓	✓	Once a system is installed, it needs to be configured properly to plug all loopholes and remove all vulnerabilities. In the case system is not configured correctly, the attacker can exploit the system and make it vulnerable to attacks [63], [65], [87].
9	Insecure Cryptographic Storage	✓				The sensitive data such as user credentials need to be stored in encrypted form so that in case of attacks, the attacker is unable to access the sensitive information. The encryption level should be as per norms to make stored data secure [63], [65].
10	4XML External Entities (XXE)			✓		XML (Extensible Markup Language) is used by many web service and applications. XML parser or processor is used for the interpretation of XML data. If the XML processor accepts the XML entity without proper verification, then the attacker can push unwanted content that XML processor is unable to process and causes the denial of service [63], [87], [65].

TABLE 1. *(Continued.)* **Merging of the OWASP Top 10 Vulnerability Types for 2010, 2013, 2017 and 2021. Columns labeled 10, 13, and 17 correspond to the OWASP Top Ten Vulnerability Types for 2010, 2013, 2017 and 2021 respectively.**

No.	Vulnerability Type	10	13	17	21	Description
11	Failure to Restrict URL Access	✓				This vulnerability allows users to make up a URL to access hidden content in the system and exploit it. The system needs to be configured so the URL access is allowed only with proper authorization [63], [65], [87].
12	Sensitive Data Exposure		✓	✓		The sensitive data such as password, credit card details, personal information needs to be protected while this type of data is transferred as well as in storage using various encryption mechanisms. The data gets exposed if it is not protected while in transit or storage [63], [65], [87].
13	Insufficient Transport Layer Protection	✓				The transport layer protocol is used for providing requisite encryption to avoid data vulnerability during transit. The data becomes vulnerable to man-in-middle attack and spoofing attack if it is not protected with proper encryption during transit [63], [65], [87].
14	Unvalidated Redirects and Forwards	✓	✓			Many web sites direct the user by providing the link to other sites or web pages. However, if the redirected site's credibility is not checked or verified, the user can become the victim of malware or phishing [63], [65], [87].
15	Insecure Deserialization			✓		The system uses serialization to convert binary data to ASCII characters so that it can be sent using common protocols. On the receipt side, the serialized data is deserialized. However, if the data after serialization is not sent along with integrity checks, an attacker can modify it, then the program code can be changed when it is deserialized leading to serious vulnerability [63], [65], [87].
16	Using Components with Known Vulnerabilities		✓	✓		A program using various components such as framework libraries and software modules during its execution. If any of these components have vulnerabilities, then the attacker can exploit it and launch an attack [63], [65], [87].
17	Cryptographic Failures				✓	Cryptographic failures happen when cryptographic systems break down and can no longer protect sensitive information. These failures can happen at the level of the cryptographic function itself, the cryptographic keys, or even at the level of the person who is trying to use the cryptographic function [92].
18	Insecure Design				✓	Insecure Design refers to risks caused by flaws in the design. This means that more threat modelling, secure design patterns and principles, and reference architectures will be used to move security to the left. It is a wide domain that includes a lot of different types of weaknesses [92].
19	Vulnerable and Outdated Components				✓	All vulnerable, outdated and unsupported software are included in this category. You are probably at risk if you are not aware of the type of the versions of your components, such as direct and indirect dependencies [92].
20	Identification and Authentication Failures				✓	Security risks occur when the identity, authentication, or session management of user is not carefully taken care of, which may enable attackers to take advantage of their passwords, keys, session tokens or flaws in the software to take over users' identities for a short time or for good [92].
21	Software and Data Integrity Failures				✓	This type of failure is concerned with code and infrastructure that fails in shielding against integrity violations. Software updates, critical data, and CI/CD pipelines that are implemented without verification are included in it [92].
22	Security Logging and Monitoring Failures			✓	✓	This category was previously named Using Components with Known Vulnerabilities in 2017. It includes mistakes that people make when they detect, escalate, and respond to active breaches. Without logging and keeping an eye on things, breaches cannot be found. Failures in this category have an impact on visibility, alerting, and forensics [92].
23	Server-Side Request Forgery (SSRF)				✓	SSRF attacks are made in order to draw on how a server handles external information. The main aim of the attack is to steal sensitive information on the server or benefit from other vulnerabilities by using SSRF input validation countermeasures [92].

E. WEB APPLICATION VULNERABILITY SCANNERS(WVS'S) AND THEIR ARCHITECTURE

Black-box WVS's are the automated testing tools used for examining and detecting vulnerabilities in web applications. Several WVS's test the prevalent vulnerabilities in web applications and web servers. These scanners are either academic research projects or open-source tools developed by academic members and researchers who are interested in studying and improving web application vulnerability tools or commercial products that are owned by software companies. The commercialized scanners usually provide more effective results than open-source scanners; however, they can cost from just under 100 to over 6000 US dollar [81].

The design of a WVS includes three core components as per the usage scenario. First, the crawler module grabs the content of the web pages. Second, the attacker module is designated for launching the attacks. Third, the analysis module highlights vulnerabilities.

- The Crawling Module is the most crucial component of a WVS and is performed by utilizing a "crawler" component. It investigates the web application to identify and recover web pages and the related input vectors like input fields in HTML forms, and request parameters GET and POST, and cookies. Moreover, the crawler generates an indexed list of all the crawled web pages. The detection of web vulnerabilities determines the quality of the crawler. If the scanner's attack engine is subpar, a vulnerability may be missed [48].
- The Fuzzing Module is used to investigate the URLs of the pages and input vectors. After the crawling, the attack patterns recognized in the previous step are sent by the crawler to the entry points. It produces a vulnerable value that triggers a type of vulnerability for each entry tested using the WVS. For example, the fuzzer tries to detect XSS vulnerabilities by injecting malicious JavaScript code or SQL injection vulnerabilities by using SQL strings with specific meanings, such as ticks and SQL operators [48].
- The Analysis Module examines the pages that the WVS returns due to the attack that the attacker module launched to detect potential vulnerabilities and provide feedback to other modules. For instance, input testing of SQL injection will return a page that contains a database error message; then, the analysis module may deduce the presence of an SQL injection vulnerability [48].

III. RELATED WORK

Black-box testing has been the focus of many recent studies aimed at improving security in data, systems, and networks. However, only a few surveys and overviews on Black-box web vulnerability scanners were returned by this research.

Bertoglio and Zorzo [18] systemically reviewed 54 primary studies using quality criteria to selected papers to determine reliability and credibility. The criteria grouped papers as 'Good', ' Very good ', and 'Excellent'. The study identified scanners used for penetration testing and their characteristics. Based on their analysis, 13 scanners were identified as the most cited ones. It also identified frameworks, methodologies, and security, testing models. Additionally, it analyzed the relationship between scanners and models besides some challenges of penetration testing. The researchers further identified process efficiency and effectiveness as critical challenges besides the vulnerability assessment process. Also, they noted that challenges in the analysis model and security scanners influence the security of scanners. Our research method is more extensive than the above study. We analyzed 320 studies in our work, and a total of 30 scanners were collected and identified in the paper. All of the returned scanners ' characteristics were provided based on what was indicated in the research papers and information available on the scanners' websites. Similarly, a survey study by Mirjalili *et al.* [65] explored the applications of web penetration testing and its models and highlighted the comparison between web vulnerability scanners. The survey reviewed previous literature on pen test methods and scanners and divided it into three categories. The first category examined and compared various methods and scanners. The second one proposed a new method or scanner for detecting vulnerabilities in web applications. The third category involved proposing a proper testing environment for executing web penetration testing. Moreover, the paper observed a correlation between 13 open-source and seven commercial scanners. It also noted that there are two core factors to judge the effectiveness and efficiency of the scanners. First is the "Structural Design" which deals with the GUI (Graphical User Interface), user ease, customization, and performance. The other key decision factor is the "Supported Features and Functionalities", which incorporate crawling techniques (automatic/manual), analysis techniques, auditing, and logging along with the generation of user reports. The researchers found that some of the reviewed scanners had technical problems such as the inability to detect some types of attacks, such as stored SQLi and stored XSS attacks. Also, some scanners did not support new technologies and were incapable of detecting vulnerabilities attributed to application logic flows. In our survey, we identified 21 free/open source and 9 commercial scanners. In addition, our research covers additional aspects such as the developer that created the scanner, the technology utilized to design it, and the scanner's operating platform (e.g. Windows, Mac OS X or Linux). We also looked at the scanner's user interface, whether it was GUI or CLI. Furthermore, we included the availability of documents, such as the user manual and installation guide. Another study conducted by Kyriakos *et al.* [55] reviewed existing literature on web vulnerability scanners. The researchers delved deep into fundamental open-source scanners and databases. They examined the web vulnerability of fundamental open-source scanners and databases by comparing them based on configuration, functionality, and support. The study also examined the scanners by comparing their accuracy of identifying vulnerability, errors in a web application, and their frequency. Moreover, it evaluated the functionality of the scanners based on categorization, vulnerability coverage, risk assessment inference, and counter-measuring. Besides, the researchers determined configuration using architecture, operation system support, level of usage, required resources, modularity, and access control mode. The researchers concluded that complete benchmarking of vulnerabilities, scanning strategy and workflow is essential to support the execution of the scanners. In comparison to this study, our analysis is more complete because it covers the most prevalent commercial and open-source scanners, whereas this study solely focused on open-source scanners. Additionally, we examined and discussed common web vulnerability scanner features, whereas this study only addressed three: settings, functionality, and support. Furthermore, the performance of the researched scanners in finding vulnerabilities in web applications was not included in this study. However, we reviewed in-depth the findings of evaluation studies undertaken on these scanners, as well as the knowledge gap in this domain.

Kumar and Sheth [56] conducted a review on the Zero-day vulnerabilities and the web application scanners that are used to detect these vulnerabilities in web services. The study explained different techniques used to detect and prevent zero-day vulnerability based on statistical-based methods, behaviour and signature-based methods, and hybrid techniques. The primary objective of each technique is to recognize the exploits' existence, eliminate them in real-time, and minimize the damage induced by the attack. One significant challenge is to ensure that the victim's machine threshold delay for analysis and quarantine is not exceeded. However, in some cases, this can cause undermining of the affected system. The researchers concluded that Zero-day attacks could misuse obscure vulnerabilities due to the absence or lack of antivirus, patches, and intrusion-detection signatures. To combat zero-day attacks, updating the system can disclose patches for most of the unknown vulnerabilities that were not detected during the system's development. In addition to that, the researchers suggested a robust framework designed to help the penetration tester detect and prevent zero-day vulnerabilities and remote code execution. Our research is thorough, and it includes information on all vulnerabilities identified by the Open Web Application Security Project (OWASP), including the OWASP Top Ten - 2010, the OWASP Top Ten - 2013, and the OWASP Top Ten - 2017. Furthermore, we looked into both the commercial and open- source scanners for detecting these security laws.

Seng *et al.* [78] conducted another survey on the available methodologies used to assess web vulnerability scanners regarding test coverage, attack coverage, and vulnerability detection rate. It also highlighted the OWASP Top Ten vulnerabilities in web applications and the popular test-beds used to evaluate the web vulnerability scanners. In this study, the authors investigated some popular web vulnerability scanners, including Acunetix Web Vulnerability Scanner, BurpSuite, N-Sparker, Wapiti, W3af, Vega, Arachni, and Owasp Zap. Nevertheless, the paper could not answer some of the research questions aimed at quantifying the quality of web application security scanners. For instance, the suitable number of testbeds used to benchmark a web application security scanner remained unknown. It only showed that the number of testbeds used to benchmark web vulnerability scanners ranged from zero to thousands. Besides, the researchers did not specify measurement metrics used in describing the test coverage of web application vulnerability scanners, attack coverage, and vulnerability detection rate.

IV. RESEARCH QUESTIONS

This paper investigates the WVS's to address three main Research Questions (RQs):

RQ.1 What are the most cited web application vulnerability scanners?

Ans. Table 3 reports the most cited web vulnerability scanners by other researchers' studies.

RQ.2 What are the general characteristics of the reported scanners?

Ans. Table 4 and Table 5 are built to list all the characteristics of the scanners to satisfy this question.

RQ.3 What are the most common OWASP Top Ten vulnerabilities tested by the reported scanners?

Ans. To answer this question and respond, Table 6 and Table 7 contain the evaluation results of studies conducted by other researchers.

V. RESEARCH METHODS

Systematic Literature Review (SLR) refers to the type of literature review that assists researchers in finding, classifying and investigating the existing literature for any particular research query. As its main objective, SLR assesses the already present literature in accordance with the research question and finds the gap in it. This SLR is following the guidelines provided by PRISMA [91].

A. SEARCH STRATEGY

PRISMA refers to the minimum set of evidence-based items used to detail meta-analyses and systematic reviews [91]. As its primary focus, it makes sure that the systematic reviews are reported transparently and completely and also details information flow through the various phases, such as identification, screening, eligibility and included as given in Figure 1:

1) IDENTIFICATION

Researchers used chains of related words to get relevant papers in order to meet the objective of the study. Some of the keywords used in this search included Black-box, penetration testing, scanner, and tool. The researchers further developed an adequate set of search phrases by studying relevant literature. The selected search phrases include ''vulnerability scanner'', ''web application vulnerability scanner'', ''penetration testing tool'', and ''injection tool''. However, entering ''benchmarking vulnerability scanner'' and ''benchmarking web application vulnerability scanner'' as key words did not give any result. The authors surveyed journal papers and international conference proceedings from databases such as Google Scholar, ACM Digital Library, SpringerLink, and

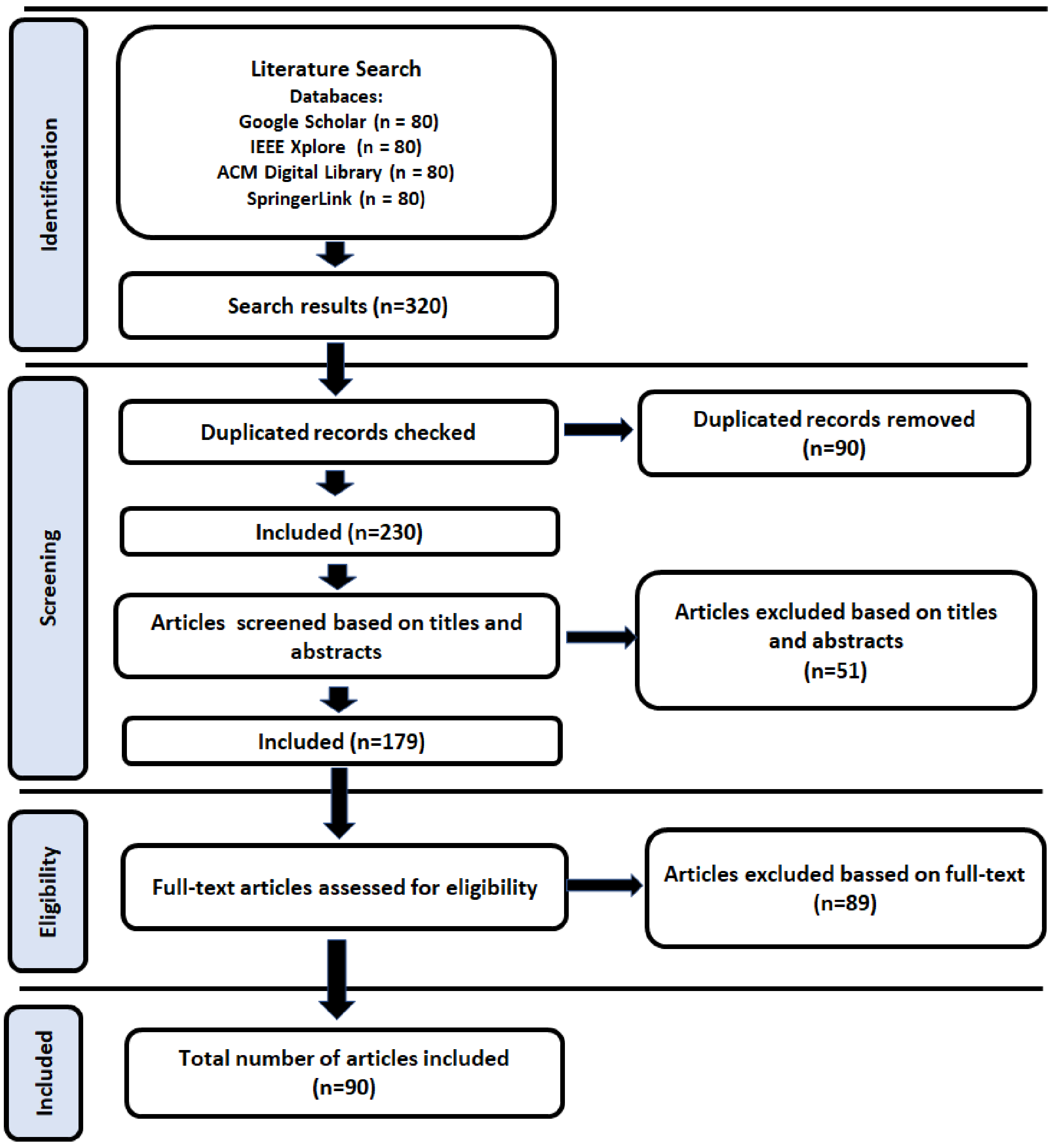

Identification
Literature Search
Databaces:
Google Scholar (n = 80)
IEEE Xplore (n = 80)
ACM Digital Library (n = 80)
SpringerLink (n = 80)
Search results (n=320)
Screening
Duplicated records checked
Duplicated records removed (n=90)
Included (n=230)
Articles screened based on titles and abstracts
Articles excluded based on titles and abstracts (n=51)
Included (n=179)
Eligibility
Full-text articles assessed for eligibility
Articles excluded bassed on full-text (n=89)
Included
Total number of articles included (n=90)

FIGURE 1. PRISMA flow diagram for studies selection.

IEEE xplore to acquire relevant research papers. Overall, 320 manuscripts were retrieved with keywords stated above.

2) SCREENING

The researchers merged the results from all searches, thus eliminating duplicate entries. After removing the duplicated articles, the authors obtained a total of 233 papers from the resources outlined above. They also read the title and the abstract of each paper for screening purposes and they found only 179 articles related to web application vulnerability scanners.

3) ELIGIBILITY

The researchers assessed the full texts of the 179 articles and only collected studies that introduced, compared, evaluated, or reviewed web vulnerability scanners. As a result, 89 studies did not meet the research objectives and were excluded, hence.

4) INCLUDED

The authors considered ninety (90) studies relevant and therefore included them in this study as they fulfilled the

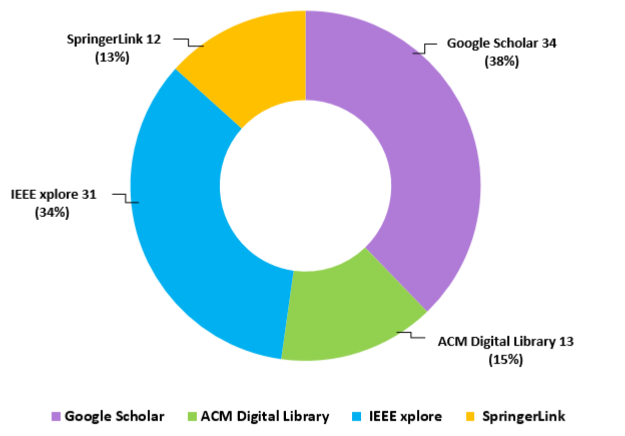


FIGURE 2. Number of papers returned from each source.

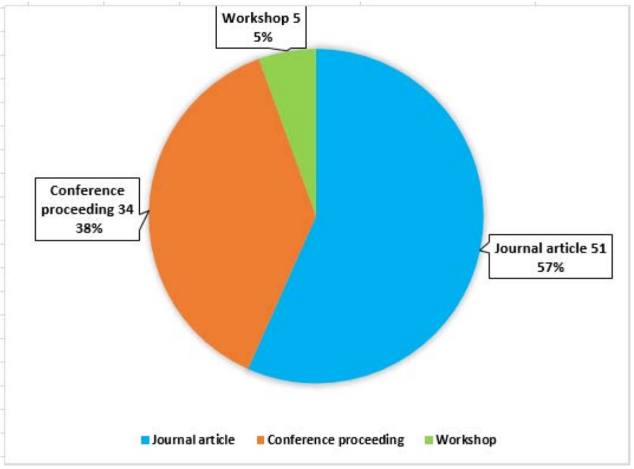


FIGURE 3. Distribution of paper types elicited from the research.

objectives of the study and answered the present research questions.

B. INCLUSION AND EXCLUSION

A set of inclusion and exclusion criteria was used to filter all research papers after their discovery. The following criteria were used to determine papers' inclusion:

- Only peer-reviewed articles must be considered.
- The article should cover Black-box web vulnerability scanners.
- Choosing the most complete version of the study for inclusion if it has been published in more than one journal.

However, the exclusion criteria were as follows:

- Duplicate studies.
- Papers that are unrelated to Black-box web vulnerability scanners were omitted.
- Inaccessible articles: To receive a private copy of them, an email was written to their writers. The articles were discarded if no response was received.
- Articles that are written in a language other than English
- Very brief publications (e.g., posters) that make only a minor contribution

VI. RESULTS AND FINDINGS

This section presents the results of the data collection, as well as how each question was answered. Please note that we only report on what was discovered in the reviewed papers. We do not personalize the information gathered.

A. PAPERS DISTRIBUTION

The 90 papers that were returned are distributed on the basis of the search engines they were obtained from, as depicted in figure 2 below.

There are clear differences in the divisions which are apparent in the graph. For example, Google Scholar elicited the most relevant papers, while SpringerLink produced the lowest amount. This demonstrates the different outcomes from these resources when using the designated search terms. Furthermore, Figure 3 shows that the returned papers fall into three distinct categories: journal articles (51 papers), conference proceedings (34 papers), and workshop papers (5 papers). A majority of these papers (total of 57%), as shown in the figure, are journal articles; the conference papers represent 38% of the papers, and the rest of the studies compose workshop papers (5%). This implies that the workshop papers have a smaller impact than the other types of literature. Conclusively, the majority (95%) of the aforementioned sources constitute journal articles and conference proceedings.

B. STUDIES CLASSIFICATION

The returned studies are classified based on their contribution to the research. They are categorized into methodology, approach, evaluation or survey.

Table 2, summarizes the distribution of returned papers, based on their contribution's classification. This table is divided into three columns as follows:

- Reference: The number of the study, as cited in this article, and the name of the first author.
- Publication Year.
- Paper Contribution: For example, the contribution of the selected paper may be developing methodology for detecting vulnerabilities in web applications or introducing a new approach which includes all studies that developed a new tool, model, framework, or algorithm to detect the flaws and loopholes in a web application. Evaluation studies include those which introduced comparative evaluation for analyzing the effectiveness of the web vulnerability scanners, and all studies that produced analysis and assessment of the web application flaws and loopholes. Finally, survey studies that have been conducted on WVS's or web application vulnerabilities are also returned by this research.

As shown in Figure 4 most of the papers returned from the searches described and presented a new approach with a total of 33 papers which represent 37% of all the returned papers. This includes all the studies which have proposed revolutionary algorithms or frameworks or developed new tool for the purpose of detecting web vulnerabilities. Furthermore, the number of studies that have carried out comparative

TABLE 2. Classification of papers based on their contributions.

Reference		Year	Paper Contribution			
			Methodology	Approach	Evaluation	Survey
[1]	Adam	2012		✓		
[2]	Monika	2013		✓		
[3]	Akhan	2018		✓		
[4]	Muhammmad	2015		✓		
[5]	Muhammad	2020		✓		
[6]	Mansour	2017			✓	
[7]	Nuno	2009	✓			
[8]	Nuno	2009			✓	
[9]	Nuno	2010			✓	
[10]	Nuno	2011		✓		
[11]	Nuno	2012		✓		
[12]	Nuno	2017		✓		
[13]	Lauri	2002		✓		
[14]	Nor	2013			✓	
[15]	Tânia	2010	✓			
[16]	Jason	2010				✓
[17]	Enrico	2016	✓			
[18]	Daniel	2017				✓
[19]	Josip	2014	✓			
[20]	Mariano	2016		✓		
[21]	Mariano	2016	✓			
[22]	Shay	2014			✓	
[23]	Atul	2012		✓		
[24]	Steven	2003	✓			
[25]	Mark	2006			✓	
[26]	Anthony	2011		✓		
[27]	Zoran DHURI	2014		✓		
[28]	Zoran Djuric	2013		✓		
[29]	Adam	2012		✓		
[30]	Adam	2010			✓	
[31]	Derek	2015		✓		
[32]	Naresh	2014		✓		
[33]	Joshua	2019		✓		
[34]	Birhanu	2013		✓		
[35]	Yong	2018		✓		
[36]	José	2013		✓		
[37]	José	2007			✓	
[38]	José	2009	✓			
[39]	José	2013		✓		
[40]	Bernhard	2014		✓		
[41]	Jai Narayan	2016		✓		
[42]	Hardik	2016	✓			
[43]	Sunil	2019		✓		
[44]	Morey	2018	✓			
[45]	William	2006		✓		
[46]	Kevin	2013	✓			
[47]	Hsiu-Chuan	2017				✓
[48]	SE	2017			✓	
[49]	Shilpa	2016			✓	
[50]	Chanchala	2016			✓	
[51]	Konstantinos	2008		✓		
[52]	Chaitali	2015	✓			
[53]	Nidal	2011			✓	
[54]	Divya	2017		✓		
[55]	Kyriakos	2019			✓	
[56]	Pratap	2016				✓
[57]	Rajiv	2018	✓			
[58]	Nicholas	2017	✓			
[59]	Taeseung	2012	✓			
[60]	Xiaowei	2014				✓
[61]	Yuma	2015			✓	
[62]	Yuliana	2012			✓	
[63]	Balume	2018			✓	
[64]	Sean	2008	✓			
[65]	Mahin	2014				✓
[66]	Indraneel	2014			✓	
[67]	Fernando	2018	✓			
[68]	Fernando	2018	✓			
[69]	Bharti	2015		✓		
[70]	Sangeeta	2017			✓	
[71]	Gurvinder	2014				✓
[72]	Muhammad	2015			✓	
[73]	Wu	2014		✓		
[74]	Hassan	2015	✓			
[75]	Albert	2015			✓	
[76]	Ain	2015	✓			
[77]	S Sandhya	2017			✓	
[78]	Seng	2018				✓
[79]	Mandar	2019			✓	
[80]	Sugandh	2015				✓
[81]	David	2010			✓	
[82]	Avinash	2012		✓		
[83]	Shailendra	2018			✓	
[84]	Antonin	2016		✓		
[85]	Natasa	2016			✓	
[86]	Atefeh	2010				✓
[87]	Frank	2011			✓	
[88]	Marco	2009			✓	
[89]	Stefan	2011			✓	
[90]	Xin	2010		✓		

evaluations of the scanners was 28, representing 31% of all the returned papers. Interestingly, only 15 of these evaluate the WVS's in detecting the OWASP Top 10 vulnerabilities. Moreover, 19 studies include methodologies for detecting vulnerabilities in web applications, which represent 21% of all returned papers. Finally, out of the studies collected

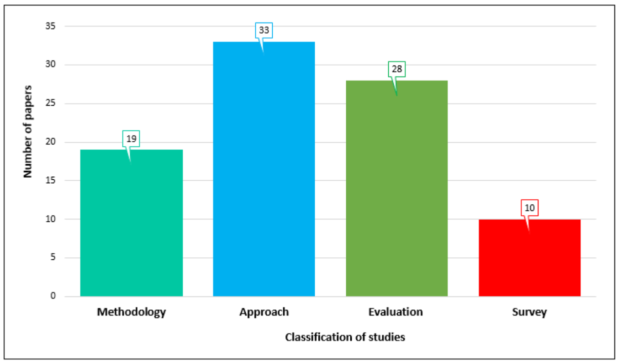


FIGURE 4. Studies' classification.

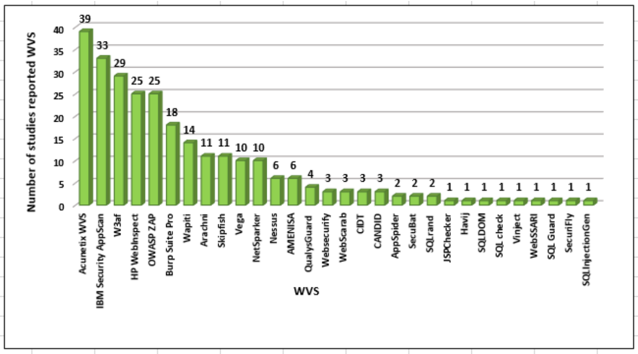


FIGURE 5. Frequency of WVS's in the returned papers.

through this review, ten surveys returned by this study represent 11% of all the returned papers; however, only five surveys were conducted on the Black-box web vulnerability scanners.

C. FREQUENCY OF WEB VULNERABILITY SCANNERS IN THE RETURNED PAPERS

Table 3 provides some answers to research question R1, where it shows a total of thirty WVS's were retrieved through the systematic literature review. The table lists all of the scanners that were reported by the selected studies and also displays the number of citations for each. As evident from Figure 5, Acunetix was the most reported scanner amongst the selected papers, where a total of 39 returned studies mentioned this scanner. This was followed by IBM AppScan as the next highest, which was reported by a total of 33 studies. On the other hand, many of the scanners were discussed in very few of the selected studies. To name a few of these scanners: SQLInjectionGen, SecuriFly, SQL Guard, WebSSARI, VinJect, SQL check, SQLDOM, Havij and JSPChecker, were all only reported on by one study each. Furthermore, it can also be noted that whilst most of the scanners are mentioned in numerous publications, very few studies conducted any critical evaluation with regards to the respective scanners. These evaluative studies would look to compare the scanners based on their capability in detecting vulnerabilities outlined by the OWASP Top 10 Vulnerabilities.The columns of the table are described as below:

- Scanner: A Black-box testing scanner, including both commercial and open source.
- Citation No: Number of citations that mention the given scanner.
- Studies Mentioning Scanner: List of studies that mentioned the scanner, referred to by reference number.
- Number of Evaluation Studies: The number of studies that evaluated or compared a given scanner.
- Studies Evaluating this Scanner: List of studies that evaluate a given scanner, referred to by reference number.

D. GENERAL CHARACTERISTICS OF THE REPORTED VULNERABILITY SCANNERS

Table 4 and Table 5 provide some answers to the research question R2 by illustrating the most important characteristics of the scanners returned by this study, including information such as the scanner name, company, license, technology, run platform, usage, documentation, and OWASP Top 10 Vulnerability detection. These characteristics are detailed as below:

- Scanner: A Black-box testing scanner, including both commercial and open-source.
- Author or Organization: the scanner inventor or producer.
- License: The scanner's license whether it is commercial or open-source.
- Technology: The programming language that a given scanner is implemented in.
- Run Platform: The platform that can be used to run the given scanner.
- Usage: The use of the given scanner can be Graphical User Interface (GUI) as in Windows or Command Line (CLI) as in Linux.
- Documentation: The types of documentation available for installing and using the scanner.
- OWASP Top 10 vulnerabilities detection: It includes the list of scanners that can scan and detect the OWASP Top 10 vulnerabilities.

A pertinent point that should be highlighted is that most of the studies do not include the scanner's features and characteristics required for this purpose. To get the more detailed information regarding the scanner's features and usage, the respective websites were explored.

E. COMPARATIVE EVALUATION

In consultation with the previous research, it was found that numerous comparative studies have been carried out to improve the accuracy of Black-box web vulnerability scanners in detecting web application vulnerabilities. While some of the studies focused primarily on comparing scanners to report the most effective one, others focused on comparing the effectiveness of a new detection approach to the effectiveness of the existing scanners. Based on this research, only fifteen (15) evaluation studies were conducted to compare the capability of twelve (12) WVS's in detecting vulnerabilities found in web applications. The number of studies evaluating the scanners vary among the scanners. As evident from Figure 6, the most evaluated scanners was Acunetix WVS and it was evaluated by nine different studies, whereas the lowest

TABLE 3. The studies reporting on and evaluating WVS's.

No	Scanner	Citation No	Studies Mentioning Scanner	Number of Evaluation Studies	Studies Evaluating this Scanner
1	Acunetix WVS	39	[5], [7], [9], [12], [14], [16], [26], [30], [37], [64], [75], [88], [6], [10], [17], [22], [27], [36], [43], [53], [72], [82], [85], [4], [8], [18], [28], [33], [38], [48], [50], [62], [66], [76], [81] [41], [70]	9	[8], [81], [88], [5], [7], [12], [10], [27], [62]
2	IBM AppScan	33	[5], [9], [12], [13], [15], [16], [26], [30], [36], [39], [65], [90], [4], [7], [8], [10], [17], [18], [22], [38], [44], [48], [68], [81], [21], [41], [54], [58], [60], [72], [73], [85], [88]	8	[5], [7], [8], [88] [7], [10], [12], [81]
3	W3af	29	[1], [5], [17], [26], [30], [49], [64], [73], [83] [6], [22], [27], [29], [34], [59], [76], [81], [85], [89] [4], [28], [33], [41], [61], [66], [77], [80], [87]	2	[5], [27]
4	HP WebInspect	25	[7], [9], [12], [15], [16], [26], [30], [36], [39], [65], [68], [88] [8], [10], [13], [18], [22], [24], [38], [48], [50], [55], [81], [87]	7	[7], [8], [81], [88], [10], [12], [85]
5	OWASP ZAP	25	[5], [6], [17], [22], [27], [31], [61], [63], [68], [72], [73], [83], [4], [28], [33], [40], [46]–[48], [66], [70], [74], [77], [85]	6	[5], [27], [40], [61], [63], [85]
6	Burp Suite Pro	18	[6], [15], [22], [30], [37], [48], [50], [73], [81], [19], [20], [27], [40]–[42], [60], [70], [79]	3	[40], [79], [81]
7	Wapiti	14	[3]–[6], [22], [26]–[28], [66], [81], [83], [85], [87]	4	[5], [6], [27], [85]
8	Arachni	11	[6], [22], [33], [47], [63], [66], [67], [73], [80], [85], [87]	2	[6], [63]
9	Skipfish	11	[1], [6], [26], [29], [34], [49], [61], [73], [77], [80], [85]	2	[61], [85]
10	Vega	10	[4]–[6], [22], [27], [28], [60], [67], [80], [85]	3	[5], [27], [85]
11	NetSparker	10	[6], [17], [22], [35], [48], [73], [76], [81], [85]	2	[81], [85]
12	Nessus	6	[2], [18], [41], [57], [66], [80]	0	
13	AMNESIA	6	[18], [23], [32], [45], [51], [86]	0	
14	QualysGuard	4	[6], [48], [50], [62]	1	[62]
15	Websecurify	3	[34], [66], [87]	0	
16	WebScarab	3	[18], [80], [90]	0	
17	CIDT	3	[23], [52], [71]	0	
18	CANDID	3	[32], [52], [86]	0	
19	AppSpider	2	[6], [64]	0	
20	SecuBat	2	[22], [81]	0	
21	SQLrand	2	[18], [23]	0	
22	JSPChecker	1	[84]	0	
23	Havij	1	[69]	0	
24	SQLDOM	1	[23]	0	
25	SQL check	1	[52]	0	
26	VinJect	1	[3]	0	
27	WebSSARI	1	[86]	0	
28	SQL Guard	1	[52]	0	
29	SecuriFly	1	[86]	0	
30	SQLInjectionGen	1	[40]	0	

evaluated scanner is QualysGuard as it was evaluated by only one study.

Table 6 and Table 7 present the results of all studies evaluated the scanners and provide some answers to the

TABLE 4. Characteristics of the WAVSs outlined in the literature Part [1].

No	Scanner Name	Author or Organization	License		Last Update
			Open Source	Commercial	
1	Acunetix WVS	Acunetix		✓	2020
2	IBM Security AppScan	IBM		✓	2018
3	W3af	Andres Riancho	✓		2015
4	HP WebInspect	HP		✓	2018
5	OWASP ZAP	OWASP	✓		2020
6	Burp Suite Pro	PortSwigger		✓	2020
7	Wapiti	Nicolas Surribas	✓		2020
8	Arachni	Tasos Laskos	✓		2020
9	Skipfish	Michal Zalewski	✓		2012
10	Vega	Subgraph	✓		2020
11	NetSparker	Mavituna Security		✓	2020
12	Nessus	Tenable		✓	2020
13	AMNESIA	William and Orso	✓		unknown
14	QualysGuard	Qualys		✓	2020
15	Websecurify	Gnucitizen	✓		unknown
16	WebScarab	OWASP	✓		unknown
17	CIDT	Choudhary and Dhore	✓		unknown
18	CANDID	Bisht et al.	✓		unknown
19	AppSpider	Rapid7		✓	2020
20	SecuBat	Stefan Kals	✓		unknown
21	SQLrand	Stephen and Angelos	✓		unknown
22	JSPChecker	Sourceforge	✓		unknown
23	Havij	ITSecTeam		✓	unknown
24	SQL DOM	David Rueter		✓	unknown
25	SQL check	An idera		✓	unknown
26	Vinject	Akhan Akbulut	✓		unknown
27	WebSSARI	Huang et al.	✓		unknown
28	SQL Guard	Guardium		✓	unknown
29	SecuriFly	Livshits et al.			unknown
30	SQLInjectionGen	MeiJunjin	✓		unknown

research question R3. The two tables summarize the detection rate of the evaluated scanners against the OWASP Top 10 Vulnerability types. An empty cell indicates that the given scanner has not detected the given vulnerability. Some scanners have been placed into 'groups' in cases where the source paper did not disclose the specific scanner that produced the result. In such cases, the scanners performance hit rate was retrieved, and the mathematical mean of the hit rate was used to calculate the 'group score'. It is clear from the two tables (6 and 7) that SQL Injection and Cross-Site Scripting are the most frequently detected vulnerabilities from the OWASP Top 10 list as evaluated by prior research. It can also be observed that the vulnerability detection rates of the evaluated WVS's, generally fall between 0% and 100% for detecting SQLi, and they fall between 6% and 100%, for detecting XSS. Another interesting observation was made,

TABLE 5. Characteristics of the WAVSs outlined in the literature Part [2].

No.	Scanner	Scanner Technology						Run Platform			Usage		Documentation		Scanner Claims to Detect OWASP Top 10 Vulnerabilities
		C	.Net	Java	Java Script	Ruby	Python	Windows	Mac OS X	Linux	GUI	CLI	User Manual	Installation Guide	
1	Acunetix WAS	✓						✓	✓	✓	✓		✓	✓	✓
2	IBM Security AppScan			✓				✓	✓	✓	✓		✓	✓	✓
3	W3af						✓	✓	✓	✓		✓	✓	✓	✓
4	HP WebInspect			✓				✓	✓	✓	✓		✓	✓	✓
5	OWASP ZAP			✓				✓	✓	✓	✓	✓	✓	✓	✓
6	Burp Suite Pro			✓				✓	✓	✓	✓		✓	✓	✓
7	Wapiti						✓	✓	✓	✓		✓	✓	✓	✓
8	Arachni					✓		✓	✓	✓	✓	✓	✓	✓	✓
9	Skipfish	✓						✓	✓	✓		✓			✓
10	Vega			✓				✓	✓	✓		✓			✓
11	NetSparker		✓					✓	✓	✓	✓		✓	✓	✓
12	Nessus	✓						✓	✓	✓		✓	✓	✓	✓
13	AMNESIA			✓											
14	QualysGuard					✓		✓	✓	✓	✓		✓	✓	✓
15	Websecurify				✓			✓	✓	✓					✓
16	WebScarab			✓				✓	✓	✓	✓			✓	✓
17	CIDT		✓					✓							
18	CANDID			✓						✓					
19	AppSpider			✓				✓			✓		✓	✓	✓
20	SecuBat		✓					✓							
21	SQLrand											✓	✓	✓	
22	JSPChecker											✓		✓	
23	Havij							✓	✓	✓	✓				
24	SQL DOM		✓												
25	SQL check		✓					✓			✓		✓	✓	
26	Vinject			✓						✓					
27	WebSSARI							✓			✓				
28	SQL Guard							✓			✓		✓	✓	
29	SecuriFly			✓											
30	SQL InjectionGen						✓								

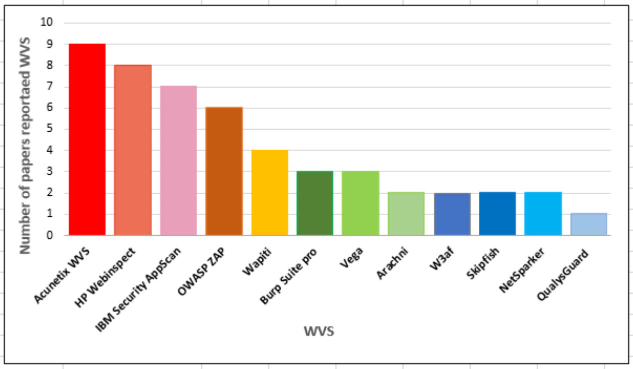


FIGURE 6. Number of studies evaluated each WVS.

in one of the works, OWASP ZAP was able to show a 100% detection rate of SQLi, while in another work, its detection rate was 0%. Vega observed the lowest detection rate for XSS, with that of only 6%; whereas, OWASP ZAP showed the highest rate, 100% the very same sort of vulnerability.

To begin with, Mburano and Si [63] evaluated two available open-source vulnerability scanners, Arachni and OWASP ZAP. Two benchmarks were used in this study, namely OWASP and Web Application Vulnerability Security Evaluation Project (WAVSEP) benchmarks. By combining the performances of the two evaluated scanners in both benchmarks, the researchers concluded that OWASP ZAP performed better than Arachni in detecting SQLi whereas, Arachni performed the best in detecting XSS. Furthermore, Alsaleh *et al.* [6] presented a comparative evaluation of three open-source WVS's. This evaluation includes comparing the security features of the evaluated scanners as well as measuring their capabilities in detecting the common web vulnerabilities. The evaluated scanners include Arachni, Wapiti and Skipfish. While, the results of the conducted experiment showed disagreements between the generated reports by the different scanners, the comparative evaluation study did not show significant differences among the evaluated scanners. The researcher concluded that the latest version of Arachni performed the best among the evaluated scanners in detecting SQLi and XSS vulnerabilities.

Moving forward, Shelly [81] analyzed the flaws and limitations of several WVS's. The evaluated scanners include W3af, Acunetix WVS, Burp Suite Pro, HP WebInspect,IBM Security AppScan and Netsparker. The researcher developed a custom vulnerable web application as a testbed used to test the selected scanners. It had two versions: a secure version for detecting false-positive results and an insecure version for detecting false-negative results. In this evaluation study, the researcher referred to the evaluated scanners as Scanner A, Scanner B, Scanner C, Scanner D, Scanner E, and Scanner F without considering the order. The mean value of these scanners' detection rate in detecting SQLi and XSS vulnerabilities was % 96 and % 43 respectively.

Additionally, Vieira *et al.* [88] conducted an experimental study to evaluate SQLi vulnerability in different web services. In this study, three well-known vulnerability scanners were used to identify the security loopholes in the available web services. The researchers decided not to mention the versions and brands of the evaluated scanners. Thus, they referred to the scanners as VS1.1, VS1.2, VS2, and VS3, where VS1.1 and VS1.2 refer to different versions of the same brand. The four evaluated scanners reported different performances in detecting SQLi. The detection rate of the fours scanners, VS1.1, VS1.2, VS2, and VS3 was 17.5%, 16.8%, 20% and 31.4%, respectively. Therefore, the mean value of the detection rate for all scanners is 21 %.

Further ahead, Makino and Klyuev [61] evaluated and compared OWASP ZAP and Skipfish in detecting (SQLi) and (XSS) in web applications. Two benchmarks were used for evaluating the effectiveness of the compared scanners, WAVSEP (Web Application Vulnerability Scanner Evaluation Project) and DVWA (Damn Vulnerable Web Application). This evaluation study is used to characterize the distinctive features and the detailed analysis of each scanner's reports and features for the vulnerability analysis. After the detailed analysis, it was concluded that OWASP ZAP performed better than Skipfish in detecting vulnerabilities, raising fewer false positives.

Moreover, Antunes and Vieira [7] compared the effectiveness of penetration testing and static code analysis techniques on the detection of SQLi in web services code. They used three popular commercial WVS's to detect vulnerabilities in a set of vulnerable services. The used scanners include HP WebInspect, IBM Rational AppScan and Acunetix Web Vulnerability Scanner. The brands of the scanners were not mentioned to assure neutrality. Thus, the scanners were referred to in this study as VS1, VS2, VS3 (without any order in particular). The performances of the three scanners were 50.8%, 36.1% and 9.8% for VS1, VS2, VS3 respectively. In this evaluation analysis, the mean value was taken to be 32.2%. Moving further, Šuteva *et al.* [85] tested and assessed six open-source or free WVS's (principally aimed at false-negative rates) by using the famous and vulnerable web application, 'WackoPicko'. The rates of false negatives of all the scanners were very high, ranging from 68.8 for IronWasp to 100 for W3af. NetSparker showed a high rate in finding all possible XSS vulnerabilities. Also, Aliero *et al.* [5] conducted an analytical evaluation to compare the effectiveness of their proposed approach-SQLIV- with the effectiveness of existing academic scanners (Acunetix WVS, IBM Security AppScan, OWAZP ZAP, Wapiti, Vega and W3af). The results showed that the two commercial scanners Acunetix WVS and IBM AppScan as well as the open-source scanner W3af achieved a high performance of 80 % in detecting SQLi vulnerabilities. Furthermore, Antunes and Vieira [12] proposed a new approach to designing a vulnerability testing scanner for web services. The researchers executed a case study to demonstrate their scanner's effectiveness in detecting SQLi vulnerabilities in web services. In this experiment, three commercial scanners representing the state-of-the-art vulnerability testing for web applications and web services were used. They include IBM Rational AppScan, HP WebInspect, and Acunetix Web

Vulnerability Scanner. They referred to them as VS1, VS2, and VS3 without any particular order. The coverage of the tools, VS1, VS2, and VS3, stood at 51%, 38%, and 3% respectively. Also, the mean value was calculated to be 31%. Going further, Martirosyan [62] evaluated the effectiveness of Acunetix WVS in detecting OWASP Top ten vulnerabilities. The researcher used the MusicStore web application as a testbed for this study. The evaluation result showed that the scanner detected Insecure Direct Object References vulnerability with a perfect detection rate of 100 %. However, it performed poorly in detecting Insecure Cryptographic Storage with detection rate was only 28%. Moreover, Antunes and Vieira [10] evaluated three commercial scanners anonymously to compare their effectiveness with the effectiveness of their approach (SignWS) in detecting SQLi vulnerabilities. The three commercial scanners include Acunetix, IBM Rational AppScan, and HP WebInspect. The commercial scanners were named VS1, VS2, and VS3 without any consideration for the order. The detection rates of VS1, VS2, and VS3 were 32.28%, 24.05%, and 1.90%, respectively. Additionally, the mean value of the detection rate for the three scanners stood at 19%.

Further, Garn *et al.* [40] provided a methodology for a better detection process of XSS in web applications. They used Burp Suite Pro and OWASP Zed Attack Proxy (ZAP) to test their methodology. Mutillidae II version 2.6.3. was used as a testbed for running this experiment. The result showed that Burp Suite Pro performed better in finding XSS vulnerabilities with a detection rate of 88.9%, whereas the detection rate for ZAP was only 80%. Moving forward, DURIĆ [27] ran an evaluation experiment to compare the performance of his approach for detecting SQLi with the performance of some well-known WVS's. The selected scanners were four open-source scanners: W3af, Nikto, Wapiti, Vega, and ZAP, and one commercial scanner, Acunetix. The author employed six experienced master students to develop the testing environment for this experiment. The result showed that Acunetix achieved the best performance with a detection rate of 50%. Acunetix detected eight vulnerabilities out of 16, Wapiti detected six, W3af detected five, and Vega detected only one vulnerability. Interestingly, ZAP did not detect any vulnerabilities in any of the three applications. Additionally, Antunes and Vieira [8] ran an evaluation experiment to compare their approach (VS.WS) with four commercial vulnerability scanners (two of them were different versions of the same vendor). The goal of this study was to identify SQLi vulnerabilities in web services. The evaluated scanners included HP WebInspect, IBM Rational AppScan and Acunetix. Also, to maintain anonymity and equality, the specific scanner applications names and their corresponding versions were not mentioned by the researchers. They referred to the four scanners in their study as VS1.1, VS1.2, VS2, and VS3, with VS 1.1 and 2.2 being two different versions of the same vendor. The detection rate of SQL vulnerabilities by VS1.1, VS1.2, VS2, and VS3 was 84%, 84%, 30%, and 38% respectively. Also, their mean value was calculated to be 59%.

Lastly, Shah [79] conducted an evaluation study to measure the Burp Suite capability in detecting vulnerabilities in web applications. The researcher used the OWASP Benchmark to evaluate the scanner's detection rate and crawling coverage. The total number of vulnerabilities detected by the scanner is 26, representing 50 % of the SQLi in the tested web application and produced 0.0 false-positive results. The time used to complete the scanning process was six hours and twenty minutes. The two benchmarks used for evaluating the effectiveness of the compared scanners were Web Application Vulnerability Scanner Evaluation Project (WAVSEP) and Damn Vulnerable Web Application (DVWA).

VII. DISCUSSION

The content of this section revolves around examining and discussing the results of the preceding section. Based on the results derived from this research, it was found that only a very small number of surveys and overviews have been conducted on Black-box web vulnerability scanners; a majority of them revolve around merely summarizing the concepts of the approaches without targeting their characteristics and effectiveness [18], [55], [65], [78]. However, the present study contains a systematic literature review on the most cited web vulnerability scanners, summarizing their characteristics and discussing the results of different evaluation studies conducted to compare their effectiveness in detecting the common web applications vulnerabilities. Based on the data collected from the reviewed studies, thirty (30) scanners were identified and it was found that their frequencies in the reviewed studies varied from scanner to scanner. For example, it was found that Acunetix WVS was the most cited scanner as it was cited by 39 papers; however, some scanners including JSPChecker, Havij, SQLDOM, SQL check, Vinject, WebSSARI, SQL Guard, SecuriFly and SQLInjectionGen were only reported by one paper each. We also found that there was no major difference between frequency of the commercial and open-source scanners in the reviewed papers, which may indicate that open-source scanners have similar importance for the researchers as the commercial scanners. Interestingly, among all returned studies, we found that the technical features and characteristics of the web vulnerability scanners were only discussed by a small number of studies. Consequently, we investigated the scanners' official websites, and we documented their main characteristics, including the technology utilized to design the scanner, and the scanner's operating platform (e.g., Windows, Mac OS X or Linux). We also looked at the scanner 's user interface, whether it was GUI or CLI. Moreover, we included the availability of documentations, such as the user manual and installation guide. As a result, we found Java to be the most frequently used language for designing the tools. We also found that all the identified commercial scanners including Acunetix WVS, HP Webinspect, IBM Security AppScan, Burp Suite pro, NetSparker and QualysGuard were provided with a Graphical User Interface (GUI), while some of the open-source scanners such as Wapiti, Skipfish were implemented with a command-line interface (CLI). Users can

TABLE 6. **WVS detection rate results per OWASP Top Ten vulnerability type [1st part].**

No.	1	2	3	4	5	6		
WAS OWASP Top 10 Vulnerability Type (2010, 2013, 2017, & 2021)	Acunetix WVS	IBM Security AppScan	HP WebInspect	Burp Suite Pro	NetSparker	QualysGuard WAS	Group A (1,2,3)	Group B (1,2,3,4,5)
Injections (10,13,17,21)	80% [5] 50% [27]	80% [5]	47% [85]	50% [79]	57% [85]		32% [7] 21% [88] 31% [12] 19% [10] 59% [8]	96% [81]
Cross SiteScripting (XSS) (10,13,17)	72% [62] 50% [27]		49% [85]	89% [40]	60% [85]	33% [62]		43% [81]
Site Request Forgery (CSRF) (10,13)								
Broken Authentication and Session Management(10,13)	50% [62]							
Insecure Direct Object References (10,13)	100% [62]							
Missing Function Level AccessControl (13)								
Security Misconfiguration(10,13,17,21)								
Insecure Cryptographic Storage (10)	28% [62]							
Failure to Restrict URL Access (10)								
Insufficient Transport Layer Protection(10)	29% [62]							
Unvalidated Redirects and Forwards (10,13)								
Sensitive Data Exposure (13,17)								
Using Components with Known Vulnerabilities(13,17)								
Broken Authentication (17)								
XML External Entities (17)								
Broken Access Control (17, 21))								
Insecure Deserialization (17)								
Insufficient Loggingand Monitoring (17)								
Cryptographic Failures (21)								
Insecure Design (21)								
Vulnerable and Outdated Components (21)								
Identification and Authentication Failures (21)								
Software and Data Integrity Failures (21)								
Security Logging and Monitoring Failures (21)								
Server-Side Request Forgery (21)								

interact easily with the scanners that use GUI to perform the scanning process; however, using scanners with CLI mode requires more technical knowledge from the users. Furthermore, even though the present study identified thirty (30) web vulnerability scanners, only twelve (12) of them were evaluated by prior research. The evaluation studies focused on measuring the capabilities of the scanners in detecting the OWASP Top 10 vulnerabilities. Based on the data collected from the evaluative studies, it was found that most of the OWASP Top 10 vulnerabilities tested by the previous studies were SQL Injection and Cross-Site Scripting [6], [7], [10], [12], [27], [40], [79], [81], [85], [88]. Only one work evaluated Acunetix WAS scanner against six vulnerability types from the OWASP Top 10 list, including Broken Authentication and Session Management, Insecure Direct Object References, Insecure Cryptographic Storage, and Insufficient Transport Layer Protection, besides SQL Injection and Cross-Site Scripting. This might be because SQL Injection and Cross-Site Scripting are the most popular web application vulnerabilities and because they are the most exploited web application vulnerabilities that yield effective results. Furthermore,SQLi attacks enable attackers to access the back-end database of web applications and to exfiltrate, destroy, and modify confidential information. XSS attacks may also result in major negative implications. For example, with XSS, attackers might hijack accounts, and steal credentials and/or other sensitive data. It can also be found that the detection rates of the evaluated WVS's fall between 0% and 100% for SQLi, whereas those for XSS fall between 6% and 100%. Interestingly, the evaluations conducted in these studies show inconsistencies in the results reported by the different scanners. Moreover, these scanners significantly vary in the detected vulnerability types, and the detection rates. In turn,this may drastically decrease the

TABLE 7. WVS detection rate results per OWASP Top Ten vulnerability type [2nd part].

No.	1	2	3	4	5	6
WAS / OWASP Top 10 Vulnerability Type (2010, 2013, 2017, & 2021)	W3af	OWAZP ZAP	Wapiti	Arachni	Skipfish	Vega
Injections (10,13,17)	80% [5] 31% [27]	100% [61] 55% [63] 67% [5] 0% [27]	59% [85] 44% [5] 96% [6] 37% [27]	48% [63] 98% [6]	95% [61] 64% [85]	67% [5]
Cross Site Scripting (XSS) (10,13,17)		100% [61] 60% [85] 76% [63] 80% [40]	35% [85] 61% [6]	64% [63] 67.5% [6]	82% [61] 73% [85]	60% [85] 6% [27]
Site RequestForgery (CSRF) (10,13)						
Broken Authentication and Session Management (10,13)						
Insecure Direct Object References (10,13)						
Missing Function Level Access Control (13)						
Security Misconfiguration (10,13,17)						
Insecure Cryptographic Storage (10)						
Failure to RestrictURL Access (10)						
Insufficient Transport Layer Protection (10)						
Unvalidated Redirects and Forwards (10,13)						
Sensitive Data Exposure (13,17)						
Using Components withKnown Vulnerabilities (13,17)						
Broken Authentication (17)						
XML External Entities (17)						
Broken Access Control (17)						
Insecure Deserialization (17)						
Insufficient Logging and Monitoring (17)						
Cryptographic Failures (21)						
Insecure Design (21)						
Vulnerable and Outdated Components (21)						
Identification and Authentication Failures (21)						
Software and Data Integrity Failures (21)						
Security Logging and Monitoring Failures (21)						
Server-Side Request Forgery (21)						

level of trustworthiness that may be attributed to WVS's and subsequently increase the demand for further research that quantitatively evaluates the quality and accuracy of web application vulnerability scanners.

VIII. CONCLUSION AND KNOWLEDGE GAPS

In this article, we have systematically surveyed, collected, organized, and evaluated most of the available knowledge on web vulnerability scanners. We identified the most frequently used scanners and we investigated their features and characteristics. We also collected and analyzed the reported detection rates and accuracy of these scanners to detect OWASP Top Ten vulnerability types. We have achieved this by examining the published research field of web vulnerability scanners in three ways:

1) By examining articles that proposed a new, revolutionary method, algorithm, or scanner for detecting web vulnerabilities.
2) By examining the articles that themselves analyzed and compared the existing scanners for detecting web vulnerabilities.

3) By drawing insights from the existing surveys and literature reviews.

When we analyzed the relatively few (15) published evaluations of the performance of web vulnerability scanners, we discovered two unexpected and, we believe, three very important findings:

1) SQLi and XSS vulnerability types were the most common tested types among the OWASP Top Ten vulnerability types. The other types of vulnerabilities in the OWASP Top Ten list were almost not tested. Only one evaluation was found that reported evaluating four (4) other OWASP Top Ten vulnerabilities and this study evaluated only one commercial web vulnerability scanner. A total of 13 studies evaluated SQLi and 8 studies evaluated XSS performance for several scanners; However, most studies only evaluated one or two scanners against only one or two non-standard, and hence difficult to replicate, web applications.
2) After analyzing and collating the efficacy results as published in the 15 evaluations, we found disparate and inconsistent efficacy reports as detailed in Table 6 and Table 7.
3) We found no published evaluations assessing the usability or quality of use of web vulnerability scanners.

Based on this findings, we would like to make the following recommendations for future directions:

1) The effectiveness evaluation of all web vulnerability scanners should be carried out using a set of ''benchmark'' web applications and for all OWASP Top 10 types of vulnerabilities; Such benchmark web applications currently do not exist. Therefore, new standard and representative benchmark web applications should be created. These benchmarks should cover all specific domains of web applications. This will help ensure web vulnerability scanner results are complete and comparable.
2) Evaluations of web vulnerability scanners should be based on the OWASP Top Ten vulnerability types or other common nomenclature for web vulnerabilities. The lack of standardization in this aspect makes it nearly impossible to adequately measure and compare the efficacy of different scanners.
3) Evaluations of web vulnerability scanners should include disclosures of affiliations or lack-of-thereof with commercial sponsors that may be potential biases for the evaluation.
4) Evaluations of web vulnerability scanners from a usability or quality-of-use perspective should also be performed.

ACKNOWLEDGMENT

The authors would like to thank the faculty and staff of the Department of Computer Science, the Center for Secure and Dependable Systems, and the College of Engineering, at the University of Idaho, and Dr. Mohammad Alshehri of Majmaah University, for supporting this research. They would also like to thank Dr. Dilshani Sarathchandra, Dr. Jia Song, and Dr. Stu Steiner for their time and constructive comments. They would also like to thank the reviewers and editors of IEEE Access for their comments and help improving this article.

REFERENCES

[1] A. Doupé, L. Cavedon, C. Kruegel, and G. Vigna, ''Enemy of the state: A state-awareblack-box web vulnerability scanner,'' in Presented at the 21st USENIX Secur. Symp. (USENIX Secur.), Aug. 2012.

[2] M. Agarwal and A. Singh, *Metasploit Penetration Testing Cookbook*. Birmingham, U.K.: Packt, 2013.

[3] A. Akbulut, ''VinJect: Toolkit for penetration testing and vulnerability scanning,'' *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, vol. 6, no. 4, pp. 779–790, Apr. 2018. [Online]. Available: https://dergipark.org.tr/en/download/article-file/517130

[4] M. S. Aliero and I. Ghani, ''A component based SQL injection vulnerability detection tool,'' in *Proc. 9th Malaysian Softw. Eng. Conf. (MySEC)*, Dec. 2015, pp. 224–229, doi: 10.1109/MySEC.2015.7475225.

[5] M. S. Aliero, I. Ghani, K. N. Qureshi, and M. F. Rohani, ''An algorithm for detecting SQL injection vulnerability using black-box testing,'' *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 1, pp. 249–266, Jan. 2020, doi: 10.1007/s12652-019-01235-z.

[6] M. Alsaleh, N. Alomar, M. Alshreef, A. Alarifi, and A. Al-Salman, ''Performance-based comparative assessment of open source web vulnerability scanners,'' *Secur. Commun. Netw.*, vol. 2017, May 2017, Art. no. 6158107.

[7] N. Antunes and M. Vieira, ''Comparing the effectiveness of penetration testing and static code analysis on the detection of SQL injection vulnerabilities in web services,'' in *Proc. 15th IEEE Pacific Rim Int. Symp. Dependable Comput.*, Nov. 2009, pp. 301–306, doi: 10.1109/PRDC.2009.54.

[8] N. Antunes and M. Vieira, ''Detecting SQL injection vulnerabilities in web services,'' in *Proc. 4th Latin-American Symp. Dependable Comput.*, Sep. 2009, pp. 17–24, doi: 10.1109/LADC.2009.21.

[9] N. Antunes and M. Vieira, ''Benchmarking vulnerability detection tools for web services,'' in *Proc. IEEE Int. Conf. Web Services*, Jul. 2010, pp. 203–210, doi: 10.1109/ICWS.2010.76.

[10] N. Antunes and M. Vieira, ''Enhancing penetration testing with attack signatures and interface monitoring for the detection of injection vulnerabilities in web services,'' in *Proc. IEEE Int. Conf. Services Comput.*, Jul. 2011, pp. 104–111, doi: 10.1109/SCC.2011.67.

[11] N. Antunes and M. Vieira, ''Defending against web application vulnerabilities,'' *Computer*, vol. 45, no. 2, pp. 66–72, Feb. 2012, doi: 10.1109/MC.2011.259.

[12] N. Antunes and M. Vieira, ''Designing vulnerability testing tools for web services: Approach, components, and tools,'' *Int. J. Inf. Secur.*, vol. 16, no. 4, pp. 435–457, Jun. 2016, doi: 10.1007/s10207-016-0334-0.

[13] L. Auronen, ''Tool-based approach to assessing web application security,'' Helsinki Univ. Technol., Espoo, Finland, Tech. Rep. T-110.501, 2002, vol. 11, pp. 12–13. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.104.893&rep=rep1&type=pdf

[14] N. F. Awang and A. A. Manaf, ''Detecting vulnerabilities in web applications using automated black box and manual penetration testing,'' in *Proc. Int. Conf. Secur. Inf. Commun. Netw.* Cairo, Egypt: Springer, Sep. 2013, pp. 230–239, doi: 10.1007/978-3-642-40597-6_20.

[15] T. Basso, P. C. S. Fernandes, M. Jino, and R. Moraes, ''Analysis of the effect of Java software faults on security vulnerabilities and their detection by commercial web vulnerability scanner tool,'' in *Proc. Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2010, pp. 150–155, doi: 10.1109/DSNW.2010.5542602.

[16] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, ''State of the art: Automated black-box web application vulnerability testing,'' in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 332–345, doi: 10.1145/1135777.1135817.

[17] E. Bazzoli, C. Criscione, F. Maggi, and S. Zanero, ''XSS PEEKER: Dissecting the XSS exploitation techniques and fuzzing mechanisms of blackbox web application scanners,'' in *Proc. 31st IFIP Int. Inf. Secur. Privacy Conf. (SEC)*. Ghent, Belgium: Springer, May 2016, pp. 243–258, doi: 10.1007/978-3-319-33630-5_17.

[18] D. Dalalana Bertoglio and A. F. Zorzo, ''Overview and open issues on penetration test,'' *J. Brazilian Comput. Soc.*, vol. 23, no. 1, Dec. 2017, doi: 10.1186/s13173-017-0051-1.

[19] J. Bozic, D. E. Simos, and F. Wotawa, "Attack pattern-based combinatorial testing," in *Proc. 9th Int. Workshop Automat. Softw. Test*, May 2014, pp. 1–7, doi: 10.1145/2593501.2593502.

[20] M. Ceccato, C. D. Nguyen, D. Appelt, and L. C. Briand, "SOFIA: An automated security Oracle for black-box testing of SQL-injection vulnerabilities," in *Proc. 31st IEEE/ACM Int. Conf. Automated Softw. Eng.*, Aug. 2016, pp. 167–177, doi: 10.1145/2970276.2970343.

[21] M. Ceccato and R. Scandariato, "Static analysis and penetration testing from the perspective of maintenance teams," in *Proc. 10th ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2016, pp. 1–6, doi: 10.1145/2961111.2962611.

[22] S. Chen, "The web application vulnerability scanners benchmark," Denim Group, Israel, Tech. Rep. 201108, 2014.

[23] A. S. Choudhary and M. L. Dhore, "CIDT: Detection of malicious code injection attacks on web application," *Int. J. Comput. Appl.*, vol. 52, no. 2, pp. 19–26, Aug. 2012, doi: 10.5120/8174-1493.

[24] S. Cook. (Feb. 2003). *A Web Developers Guide to Cross-Site Scripting*. [Online]. Available: https://www.sans.org/reading-room/whitepapers/securecode/paper/988

[25] M. Curphey and R. Arawo, "Web application security assessment tools," *IEEE Security Privacy*, vol. 4, no. 4, pp. 32–41, Jul. 2006, doi: 10.1109/MSP.2006.108.

[26] A. Dessiatnikoff, R. Akrout, E. Alata, M. Kaaniche, and V. Nicomette, "A clustering approach for web vulnerabilities detection," in *Proc. IEEE 17th Pacific Rim Int. Symp. Dependable Comput.*, Dec. 2011, pp. 194–203, doi: 10.1109/PRDC.2011.31.

[27] Z. C. Dhuri, "WAPTT-Web application penetration testing tool," *Adv. Elect. Comput. Eng.*, vol. 14, no. 1, pp. 93–102, Nov. 2014, doi: 10.4316/AECE.2014.01015.

[28] Z. Djuric, "A black-box testing tool for detecting SQL injection vulnerabilities," in *Proc. 2nd Int. Conf. Informat. Appl. (ICIA)*, Sep. 2013, pp. 216–221, doi: 10.1109/ICoIA.2013.6650259.

[29] A. Doupé, L. Cavedon, C. Kruegel, and G. Vigna, "Enemy of the state: A state-aware black-box web vulnerability scanner," in Presented at the 21st USENIX Secur. Symp. (USENIX Secur.), Aug. 2012.

[30] A. Doupé, M. Cova, and G. Vigna, "Why Johnny can't pentest: An analysis of black-box web vulnerability scanners," in *Proc. 7th Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Bonn, Germany: Springer, 2010, pp. 111–131, doi: 10.1007/978-3-642-14215-4_7.

[31] D. M. Duchesne, "Using CABECTPortal as a case study to extend the capabilities of penetration testing tools," in *Proc. 46th ACM Tech. Symp. Comput. Sci. Educ.*, New York, NY, USA, Feb. 2015, p. 715, doi: 10.1145/2676723.2693629.

[32] N. Duhan and B. Saneja, "A two tier defense against SQL injection," in *Proc. Int. Conf. Signal Propag. Comput. Technol. (ICSPCT)*, Jul. 2014, pp. 415–420, doi: 10.1109/ICSPCT.2014.6884906.

[33] J. Eckroth, K. Chen, H. Gatewood, and B. Belna, "Alpaca: Building dynamic cyber ranges with procedurally-generated vulnerability lattices," in *Proc. ACM Southeast Conf.*, Apr. 2019, pp. 78–85, doi: 10.1145/3299815.3314438.

[34] B. Eshete, A. Villafiorita, K. Weldemariam, and M. Zulkernine, "Confeagle: Automated analysis of configuration vulnerabilities in web applications," in *Proc. IEEE 7th Int. Conf. Softw. Secur. Rel.*, Jun. 2013, pp. 188–197, doi: 10.1109/SERE.2013.30.

[35] Y. Fang, X. Long, L. Liu, and C. Huang, "DarkHunter: A fingerprint recognition model for web automated scanners based on CNN," in *Proc. 2nd Int. Conf. Cryptogr., Secur. Privacy*, Mar. 2018, pp. 10–15, doi: 10.1145/3199478.3199504.

[36] J. Fonseca and F. Matarese, "Using vulnerability injection to improve web security," in *Innovative Technologies for Dependable OTS-Based Critical Systems*. Milan, Italy: Springer, Jan. 2013, pp. 145–157, doi: 10.1007/978-88-470-2772-5_11.

[37] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks," in *Proc. 13th Pacific Rim Int. Symp. Dependable Comput. (PRDC)*, Dec. 2007, pp. 365–372, doi: 10.1109/PRDC.2007.55.

[38] J. Fonseca, M. Vieira, and H. Madeira, "Vulnerability & attack injection for web applications," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun./Jul. 2009, pp. 93–102.

[39] J. Fonseca, M. Vieira, and H. Madeira, "Evaluation of web security mechanisms using vulnerability & attack injection," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 5, pp. 440–453, Sep./Oct. 2013, doi: 10.1109/TDSC.2013.45.

[40] B. Garn, I. Kapsalis, D. E. Simos, and S. Winkler, "On the applicability of combinatorial testing to web application security testing: A case study," in *Proc. Workshop Joining AcadeMiA Ind. Contrib. Test Autom. Model-Based Test. (JAMAICA)*, 2014, pp. 16–21, doi: 10.1145/2631890.2631894.

[41] J. N. Goel, M. H. Asghar, V. Kumar, and S. K. Pandey, "Ensemble based approach to increase vulnerability assessment and penetration testing accuracy," in *Proc. Int. Conf. Innov. Challenges Cyber Secur. (ICICCS-INBUSH)*, Feb. 2016, pp. 330–335, doi: 10.1109/ICICCS.2016.7542303.

[42] H. Gohel and P. Sharma, "Intelligent web security testing with threat assessment and client server penetration," in *Proc. Int. Conf. ICT Sustain. Develop.* Singapore: Springer, Feb. 2016, pp. 555–568, doi: 10.1007/978-981-10-0135-2_54.

[43] S. Gupta, *Web Vulnerability Scanner*. New York, NY, USA: Apress, Nov. 2019, doi: 10.1007/978-1-4842-4341-1_11.

[44] M. J. Haber and B. Hibbert, *Asset Attack Vectors: Building Effective Vulnerability Management Strategies to Protect Organizations*. New York, NY, USA: Apress, 2018.

[45] W. G. J. Halfond and A. Orso, "Preventing SQL injection attacks using AMNESIA," in *Proc. 28th Int. Conf. Softw. Eng.*, New York, NY, USA, May 2006, pp. 795–798, doi: 10.1145/1134285.1134416.

[46] K. P. Haubris and J. J. Pauli, "Improving the efficiency and effectiveness of penetration test automation," in *Proc. 10th Int. Conf. Inf. Technology: New Generat.*, Apr. 2013, pp. 387–391, doi: 10.1109/ITNG.2013.135.

[47] H.-C. Huang, Z.-K. Zhang, H.-W. Cheng, and S. W. Shieh, "Web application security: Threats, countermeasures, and pitfalls," *Computer*, vol. 50, no. 6, pp. 81–85, Jun. 2017, doi: 10.1109/MC.2017.183.

[48] S. Idrissi, N. Berbiche, F. Guerouate, and M. Shibi, "Performance evaluation of web application security scanners for prevention and protection against vulnerabilities," *Int. J. Appl. Eng. Res.*, vol. 12, no. 21, pp. 11068–11076, Jun. 2017. [Online]. Available: https://www.semanticscholar.org/paper/Performance-Evaluation-of-Web-Application-Security-Idrissi-Berbiche/5c56cf1cd211c2810f8217585123c4b99ba5b15a

[49] S. Jose, K. Priyadarshini, and K. Abirami, "An analysis of black-box web application vulnerability scanners in SQLi detection," in *Proc. Int. Conf. Soft Comput. Syst.* New Delhi, India: Springer, Dec. 2016, pp. 177–185, doi: 10.1007/978-81-322-2674-1_18.

[50] C. Joshi and U. Kumar, "Security testing and assessment of vulnerability scanners in quest of current information security landscape," *Int. J. Comput. Appl.*, vol. 145, no. 2, pp. 1–7, Jul. 2016, doi: 10.5120/ijca2016910563.

[51] K. Kemalis and T. Tzouramanis, "SQL-IDS: A specification-based approach for SQL-injection detection," in *Proc. ACM Symp. Appl. Comput. (SAC)*, New York, NY, USA, 2008, pp. 2153–2158, doi: 10.1145/1363686.1364201.

[52] C. Khairnar, "Detection and automatic prevention against SQL injection attack and XSS attacks perform on web application," *Maharashtra India*, vol. 5, no. 11, Nov. 2015. [Online]. Available: https://www.semanticscholar.org/paper/Detection-and-Automatic-Prevention-against-SQL-and-Khairnar/b2f6bee4c95ed56f2ba6df5efdc0f7f441875840

[53] N. Khoury, P. Zavarsky, D. Lindskog, and R. Ruhl, "Testing and assessing web vulnerability scanners for persistent SQL injection attacks," in *Proc. 1st Int. Workshop Secur. Privacy Preserving e-Societies (SeceS)*, 2011, pp. 12–18, doi: 10.1145/2107581.2107584.

[54] D. Kilaru, "Improving techniques for SQL injection defenses," M.S. thesis, Dept. Comput. Sci., University of Colorado Colorado Springs, Boulder, CO, USA, 2017.

[55] K. Kritikos, K. Magoutis, M. Papoutsakis, and S. Ioannidis, "A survey on vulnerability assessment tools and databases for cloud-based web applications," *Array*, vols. 3–4, Sep. 2019, Art. no. 100011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005619300116

[56] P. Kumar and R. K. Sheth, "A review on 0-day vulnerability testing in web application," in *Proc. 2nd Int. Conf. Inf. Commun. Technol. Competitive Strategies (ICTCS)*, 2016, pp. 1–4, doi: 10.1145/2905055.2905357.

[57] R. Kumar and K. Tlhagadikgora, "Internal network penetration testing using free/open source tools: Network and system administration approach," in *Proc. Int. Conf. Adv. Inform. Comput. Res.* Shimla, India: Springer, Jul. 2018, pp. 257–269, doi: 10.1007/978-981-13-3143-5_22.

[58] N. M. Z. Lee, S. Y. Ooi, and Y. H. Pang, "Vulnerability reports consolidation for network scanners," in *Proc. Int. Conf. Comput. Sci. Technol.* Kuala Lumpur, Malaysia: Springer, Feb. 2017, pp. 11–20, doi: 10.1007/978-981-10-8276-4_2.

[59] T. Lee, G. Won, S. Cho, N. Park, and D. Won, "Experimentation and validation of web application's vulnerability using security testing method," in *Computer Science and Its Applications*. Dordrecht, The Netherlands: Springer, Oct. 2012, pp. 723–731, doi: 10.1007/978-94-007-5699-1_74.

[60] X. Li and Y. Xue, "A survey on server-side approaches to securing web applications," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–29, Apr. 2014, doi: 10.1145/2541315.

[61] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in *Proc. IEEE 8th Int. Conf. Intell. Data Acquisition Adv. Comput. Syst., Technol. Appl. (IDAACS)*, Sep. 2015, pp. 399–402, doi: 10.1109/IDAACS.2015.7340766.

[62] Y. Martirosyan, "Security evaluation of web application vulnerability scanners strengths and limitations using custom web application," Ph.D. dissertation, Dept. Comput. Sci., California State Univ.-East Bay, Hayward, CA, USA, Oct. 2012. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.722.5637&rep=rep1&type=pdf

[63] B. Mburano and W. Si, "Evaluation of web vulnerability scanners based on OWASP benchmark," in *Proc. 26th Int. Conf. Syst. Eng. (ICSEng)*, Dec. 2018, pp. 1–6, doi: 10.1109/ICSENG.2018.8638176.

[64] S. McAllister, E. Kirda, and C. Kruegel, "Leveraging user interactions for in-depth testing of web applications," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, Sep. 2008, pp. 191–210.

[65] M. Mirjalili, A. Nowroozi, and M. Alidoosti, "A survey on web penetration test," *Adv. Comput. Sci., Int. J.*, vol. 3, no. 6, pp. 107–121, 2014. [Online]. Available: http://www.acsij.org/documents/v3i6/acsij-2014-3-6-604.pdf

[66] I. Mukhopadhyay, S. Goswami, and E. Mandal, "Web penetration testing using nessus and metasploit tool," *IOSR J. Comput. Eng.*, vol. 16, no. 3, pp. 126–129, 2014, doi: 10.9790/0661-1634126129.

[67] F. Román Muñoz, I. I. Sabido Cortes, and L. J. García Villalba, "Enlargement of vulnerable web applications for testing," *J. Supercomput.*, vol. 74, no. 12, pp. 6598–6617, Dec. 2018, doi: 10.1007/s11227-017-1981-2.

[68] F. Román Muñoz and L. J. García Villalba, "An algorithm to find relationships between web vulnerabilities," *J. Supercomput.*, vol. 74, no. 3, pp. 1061–1089, Mar. 2018, doi: 10.1007/s11227-016-1770-3.

[69] B. Nagpal, N. Singh, N. Chauhan, and A. Panesar, "Tool based implementation of SQL injection for penetration testing," in *Proc. Int. Conf. Comput., Commun. Autom.*, May 2015, pp. 746–749, doi: 10.1109/CCAA.2015.7148509.

[70] S. Nagpure and S. Kurkure, "Vulnerability assessment and penetration testing of web application," in *Proc. Int. Conf. Comput., Commun., Control Autom. (ICCUBEA)*, Aug. 2017, pp. 1–6, doi: 10.1109/ICCUBEA.2017.8463920.

[71] G. K. Pannu, "A survey on web application attacks," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 3, pp. 4162–4166, 2014. [Online]. Available: https://pdfs.semanticscholar.org/0672/d34966bf24aae5fd875111e8c9444c3132d6.pdf

[72] M. Parvez, P. Zavarsky, and N. Khoury, "Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities," in *Proc. 10th Int. Conf. for Internet Technol. Secured Trans. (ICITST)*, Dec. 2015, pp. 186–191, doi: 10.1109/ICITST.2015.7412085.

[73] W. Qianqian and L. Xiangjun, "Research and design on web application vulnerability scanning service," in *Proc. IEEE 5th Int. Conf. Softw. Eng. Service Sci.*, Jun. 2014, pp. 671–674, doi: 10.1109/ICSESS.2014.6933657.

[74] H. Radwan and K. Prole, "Code pulse: Real-time code coverage for penetration testing activities," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Apr. 2015, pp. 1–6, doi: 10.1109/THS.2015.7225269.

[75] A. Sagala and E. Manurung, "Testing and comparing result scanning using web vulnerability scanner," *Adv. Sci. Lett.*, vol. 21, no. 11, pp. 3458–3462, Nov. 2015, doi: 10.1166/asl.2015.6598.

[76] A. Z. M. Saleh, N. A. Rozali, A. G. Buja, K. A. Jalil, F. H. M. Ali, and T. F. A. Rahman, "A method for web application vulnerabilities detection by using Boyer-Moore string matching algorithm," *Proc. Comput. Sci.*, vol. 72, pp. 112–121, Jan. 2015, doi: 10.1016/j.procs.2015.12.111.

[77] S. Sandhya, S. Purkayastha, E. Joshua, and A. Deep, "Assessment of website security by penetration testing using wireshark," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 1–4, doi: 10.1109/ICACCS.2017.8014711.

[78] L. K. Seng, N. Ithnin, and S. Z. M. Said, "The approaches to quantify web application security scanners quality: A review," *Int. J. Adv. Comput. Res.*, vol. 8, no. 38, pp. 285–312, Sep. 2018.

[79] M. P. Shah, "Comparative analysis of the automated penetration testing tools," Nat. College Ireland, Ireland, Tech. Rep. X18139469, 2019.

[80] S. Shah and B. M. Mehtre, "An overview of vulnerability assessment and penetration testing techniques," *J. Comput. Virol. Hacking Techn.*, vol. 11, no. 1, pp. 27–49, Feb. 2015, doi: 10.1007/s11416-014-0231-x.

[81] D. A. Shelly, "Using a web server test bed to analyze the limitations of web application vulnerability scanners," Ph.D. dissertation, Dept. Comput. Eng., Virginia Tech, Blacksburg, VA, USA, Jul. 2010.

[82] A. Kumar Singh and S. Roy, "A network based vulnerability scanner for detecting SQLI attacks in web applications," in *Proc. 1st Int. Conf. Recent Adv. Inf. Technol. (RAIT)*, Mar. 2012, pp. 585–590, doi: 10.1109/RAIT.2012.6194594.

[83] S. Singh and K. Singh, "Performance analysis of vulnerability detection scanners for web systems," in *Cyber Security*. Singapore: Springer, Apr. 2018, pp. 387–399, doi: 10.1007/978-981-10-8536-9_37.

[84] A. Steinhauser and F. Gauthier, "JSPChecker: Static detection of context-sensitive cross-site scripting flaws in legacy web applications," in *Proc. ACM Workshop Program. Lang. Anal. Secur.*, Oct. 2016, pp. 57–68, doi: 10.1145/2993600.2993606.

[85] N. Šuteva, D. Zlatkovski, and A. Mileva, "Evaluation and testing of several free/open source web vulnerability scanners," in *Proc. 10th Conf. for Informat. Inf. Technol.*, Apr. 2013, pp. 221–224. [Online]. Available: http://eprints.ugd.edu.mk/id/eprint/9096

[86] A. Tajpour and S. Ibrahim, "SQL injection detection and prevention tools assessment," in *Proc. Int. Conf. Comput. Sci. Inf. Technol.*, vol. 9, Jul. 2010, pp. 518–522, doi: 10.1109/ICCSIT.2010.5563777.

[87] F. van der Loo, "Comparison of penetration testing tools for web applications," Ph.D. dissertation, Dept. Comput. Sci., Univ. Radboud, Nijmegen, The Netherlands, 2011.

[88] M. Vieira, N. Antunes, and H. Madeira, "Using web security scanners to detect vulnerabilities in web services," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2009, pp. 566–571, doi: 10.1109/DSN.2009.5270294.

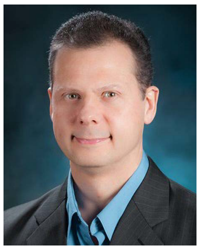
[89] S. Wagner, "The use of application scanners in software product quality assessment," in *Proc. 8th Int. Workshop Softw. Qual.*, Sep. 2011, pp. 42–49, doi: 10.1145/2024587.2024597.

[90] X. Wang, L. Wang, G. Wei, D. Zhang, and Y. Yang, "Hidden web crawling for SQL injection detection," in *Proc. 3rd IEEE Int. Conf. Broadband Netw. Multimedia Technol. (IC-BNMT)*, Oct. 2010, pp. 14–18, doi: 10.1109/ICBNMT.2010.5704860.

[91] A. Liberati, D. G. Altman, J. Tetzlaff, C. Mulrow, P. C. Gøtzsche, J. P. A. Ioannidis, M. Clarke, P. J. Devereaux, J. Kleijnen, and D. Moher, "The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: Explanation and elaboration," *J. Clin. Epidemiol.*, vol. 62, no. 10, pp. e1–e34, Oct. 2009, doi: 10.1016/j.jclinepi.2009.06.006.

[92] (2021). *OWASP Top 10*. Accessed: Mar. 11, 2022. [Online]. Available: https://owasp.org/Top10/

SULIMAN ALAZMI (Member, IEEE) received the bachelor's degree in computer science from King Saud University, Saudi Arabia, in 2004, and the master's degree in computer science with concentration of computer and network security from the University of Wollongong, Australia, in 2009. He is currently pursuing the Ph.D. degree with the Department of Computer Science, College of Engineering, University of Idaho, USA. His research interests include network security, information assurance, web vulnerability analysis and assessment, and digital forensics.



DANIEL CONTE DE LEON (Member, IEEE) received the Ph.D. degree in computer science with a focus on the security and safety of critical systems from the University of Idaho, Moscow, ID, USA, in 2006. He is currently a Cybersecurity Researcher and an Educator at the University of Idaho. He also works on developing innovative and hands-on methods and tools for cybersecurity learning. His teaching experience comes from many years of teaching across the computer science and cybersecurity curriculums. His current research interests include development of methods and tools for the design, development, implementation, configuration, operation, and maintenance of safe and secure critical infrastructure systems.