[1]Tliahun Ejigu Belay,
[2]Dr. Shalu Gupta

Scientific Research Methodology for Strategic Investigation of Open-Source Web Application Vulnerability Testing Tools: A Holistic Examination and Comparative Study

Abstract: - Now a time web applications are essential to business functions, understanding their security vulnerabilities is crucial. This study offers a thorough examination of open-source web application vulnerability testing tools, using a solid scientific research methodology. My approach includes a comprehensive analysis of various tools, assessing their effectiveness, usability, and adaptability in real-world situations. Through comparative analysis, I review key metrics such as detection rates, false positives, and the ease of integrating these tools into existing workflows. By synthesizing both qualitative and quantitative data, we aim to provide insights that assist developers and security professionals in selecting the right tools while contributing to the broader conversation this research serves as a foundational resource for future studies and practices, enhancing understanding of open-source solutions in the ever-evolving cybersecurity landscape. Additionally, I conduct comparative case studies to investigate key metrics and variables that influence outcomes, allowing us to draw significant insights and identify best practices. My research not only seeks to clarify the complexities of the subjects studied but also aims to enrich the academic discourse by providing actionable recommendations. This comprehensive approach promotes a deeper understanding of the interconnections among various elements, paving the way for future research and practical applications in the field. By establishing a rigorous methodological framework, this study seeks to improve the effectiveness of comparative studies and guide decision-making processes across different contexts.

Keywords: Open Source Tools, Web Application Security, Vulnerability Testing, Comparative Study, Holistic Examination, Cybersecurity, Tool Effectiveness, Usability Assessment, Detection Rates, False Positives, Integration Ease, Qualitative and Quantitative

3.1 INTRODUCTION

This research uses a mixed methods study that includes both quantitative and qualitative research techniques. This study begins with a systematic review of the existing literature on security testing tools for open-source web applications. This overview provides the basis for deciding which instruments to evaluate in the study. The evaluation is performed using a reference framework developed from a literature review. This framework evaluates different tools based on functionality, ease of use, accuracy, and versatility. Vulnerability assessment, in a broader sense, is the process of identifying, documenting, and classifying vulnerabilities within a system. In the context of information technology systems, this definition narrows down to finding, documenting, classifying, and possibly mitigating security weaknesses within an information system. Additionally, this study also evaluates the effectiveness of the selected tools in detecting vulnerabilities in various web applications. Data is collected from a variety of sources, including online surveys, case studies, and interviews with developers using selected tools. The methodology used to achieve these goals includes the following steps [1]:

- The planning and requirements definition phase was critical in developing a research methodology that was well-structured, practical, and closely aligned with the overall objectives of the study. This thorough upfront work laid a strong foundation for the successful execution of the subsequent stages of the research project.
- Literature Review: Conduct a literature review to understand the current state of web application vulnerability tools and the various open-source options available.
- Select tools and Examination Criteria: Identify and define criteria for evaluating the performance of the selected tools, including: Scanning speed, accuracy, reporting, ease of use, and scalability. Comparative analysis: Perform a comparative analysis of the selected tools based on established criteria.
- Organized the expected finding Based on the comprehensive literature review, the established conceptual framework, and the defined research objectives, the key expected findings from this study are focused on analyzing and comparing the performance of various open-source tools: Analyze the comparative

[1]Research Scholar, Dept. of Computer Applications, Guru Kashi University, Talwandi Sabo, Punjab, India
[2.]Assistant Professor, Dept. of Computer Applications, Guru Kashi University, Talwandi Sabo, Punjab, India

effectiveness of the open-source tools in addressing the core research objectives and end-user requirements.

3.2 Data collection

Data collection is the process of collecting and analyzing information on relevant variables in a predetermined, methodical way so that one can respond to specific research experiment, test hypotheses, multiple sources and assess results. Whether the research is qualitative or quantitative, accurate data gathering is necessary to maintain the integrity of the study. Errors during data collection can be minimized by carefully choosing the right instruments and tools, which might be new, modified, or already-existing ones, and by providing clear instructions for how to use them.

3.2.1 Data collection methods

A. Primary data collection the method of obtaining authentic, first-hand information straight from the source is known as primary data collecting. Either the researcher or the organization itself collects this kind of data. In computer science, test experiments, surveys, and observation are a few popular primary data collection techniques.
B. Secondary data collection methods is The practice of obtaining data that has already been gathered and released by others is known as secondary data collection. The primary data is frequently supported or supplemented by this kind of data collecting. The literature review, the internet and online sources, and industry reports are a few popular secondary data collection techniques in computer science however for this research I used both data collection methods the following table show How to collect data and when to use.

3.2.2 Show How to collect data and when to use

Method	**When to use**	**How to collect data**
Testing and Experiment	To test a causal relationship between each tools	Analysis and measure their effectiveness of tools on each other's.
Survey	To understand the general characteristics open source tools and their testing behavior.	Distribute a list of questions to a sample online and paper related to my paper.
Observation	To understand something in it's natural of experiment and result.	Measure or survey of related work and try to conclude the observation.
Archival research	I will refer different research paper that related to this paper to understand current or historical events, conditions or practices.	I will Access manuscripts, documents or records from libraries, depositories or the internet sources.

3.3 Types of Data in Research

Different applications and research fields in computer science require an understanding of the differences between quantitative and qualitative data. Complement each other and offer a more comprehensive picture of the subject at hand, which is why computer science research and applications frequently benefit from their combination. The precise study issue, the resources at hand, and the required level of understanding all influence the choice of data type and use for research output analysis. Data are organized into two broad categories: qualitative and quantitative.

- Quantitative data is any numerical information that can be measured, tallied, or expressed as a numerical value. This kind of data can be further broken down into two subcategories and is commonly utilized for statistical analysis. The performance of software programs, algorithms, and computer systems is frequently assessed using quantitative data. Metrics including processing speed, memory utilization, throughput, error rates, and reaction time are included in this. In computer science, controlled experiments frequently use quantitative data to compare algorithms, test theories, and assess the efficacy of novel methods or tools.
- Qualitative data can be studied using techniques like content analysis, theme coding, or narrative analysis, such as user interviews, focus groups, and usability testing. Qualitative data is commonly obtained by methods like interviews, focus groups, or open-ended surveys. Insights into the viewpoint, requirements, and problems of the user are provided, which is essential for creating user-friendly applications and interfaces.

Both quantitative and qualitative data are crucial for a variety of activities in computer science research, including software testing, decision-making, system performance analysis, and user experience research. The particular study issue, the kind of information required, and the resources available all influence the choice of data type. Frequently, combining quantitative and qualitative data allows for a deeper comprehension of an issue or to [2]**.**

3.4 Sampling of Research

Sampling is a critical aspect of data collection in computer science research and other research domains. Developing a well-designed sampling plan is essential to ensure the collected data is representative, reliable, and can be used to draw meaningful conclusions in research result. When planning a sampling approach, researchers need to clearly define the experimental target - the entire group I want to draw conclusions about the research requirement. This could be all users of a software application, all network devices in an organization, or all potential participants in a research study will be determined.

Researchers then need to determine the sample of the study - the subset of the experiment target from which actually collect data. Obtaining a representative sample is crucial to ensure the findings can be generalized to the broader target.

There are several common sampling methods used in computer science research, such as:

- Random Sampling: Selecting participants or measurements randomly from the target to ensure each element has an equal chance of being included.
- Stratified Sampling: Dividing the target into distinct subgroups (strata) and then selecting a random sample from each stratum to ensure representation of different characteristics.
- Cluster Sampling: Dividing the target into clusters (e.g., geographic regions, organizational units) and then randomly selecting a sample of clusters to include in the study.

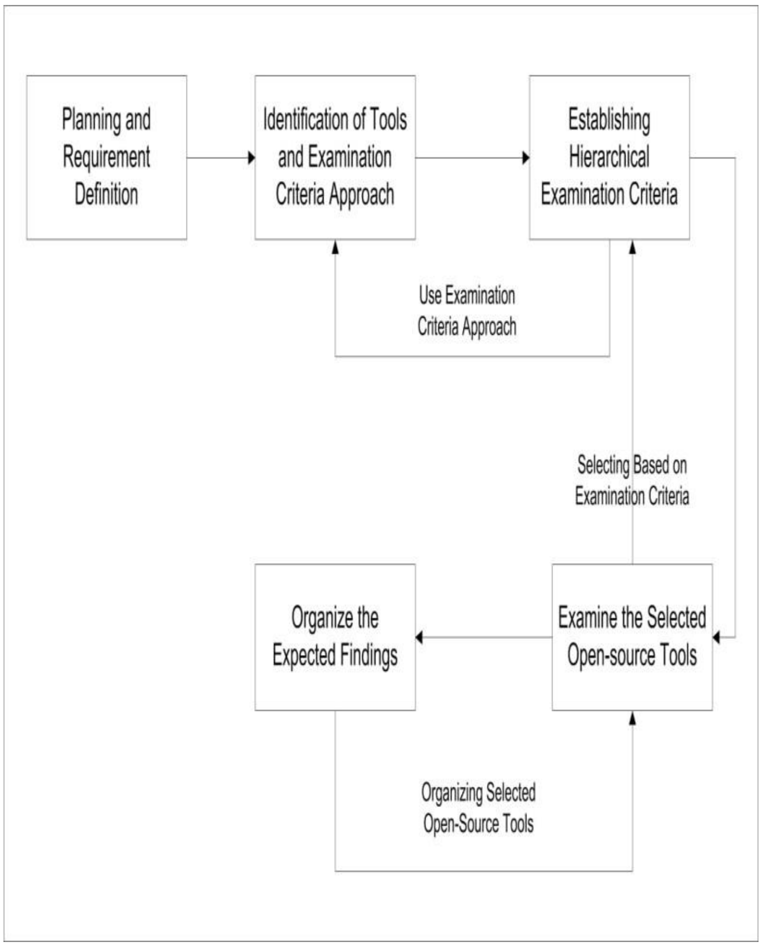
In this research I will used random sampling technique to determining the appropriate sample size is critical to ensuring the data collected is statistically meaningful and representative of the target. Factors like the expected effect size, desired statistical power, and the variability within the target should be considered when calculating the required sample size.

3.5 Research design

Research design focused on overarching strategy or blueprint that guides the execution of a research study. It establishes the framework and guidelines that dictate how the research will be conducted, including the specific methods and techniques that will be used to collect and analyse the necessary data. Essentially, the research design is defined as a detailed plan for systematically addressing a research problem in a rigorous and methodical manner. This research design will enable the comparison of multiple tools based on various metrics. The structured framework allows for a systematic evaluation and comparison of different tools or approaches based on predefined performance measures or criteria.

The objective of this study is to develop a tool that performs static code analysis. This analysis is conducted using the Taint Analysis approach, which involves identifying variables that may pose a risk (tainted) as they originate from user input. The tool then tracks these variables to potentially dangerous functions known as sinks. If a tainted variable reaches a sink without undergoing proper filtering or sanitization, it is considered vulnerability. The evaluation of the developed tool yielded satisfactory results in detecting vulnerabilities. [3].

Designing a research study to evaluate open-source vulnerability analysis tools requires a multi-step approach to ensure a comprehensive, systematic, and scientifically valid investigation. This involves carefully planning and structuring the research process to enable a rigorous comparison and assessment of the various tools under examination. In this research design, the independent variables are the different open-source vulnerability analysis tools being evaluated. The dependent variables include metrics such as accuracy, coverage, performance, usability, and user satisfaction. These dependent variables serve as the outcome measures that will be assessed and compared across the various open-source vulnerability analysis tools under investigation. The methodology used to achieve these goals includes the following flow diagram


Planning and Requirement Definition
Identification of Tools and Examination Criteria Approach
Establishing Hierarchical Examination Criteria
Use Examination Criteria Approach
Selecting Based on Examination Criteria
Organize the Expected Findings
Examine the Selected Open-source Tools
Organizing Selected Open-Source Tools

3.6 Research Testing Methods

Vulnerability scanners incorporate or extend upon different forms of penetration testing, namely white box testing and black box testing. Penetration testing is a simulated cyberattack against the computer system that checks for exploitable vulnerabilities. White box testing can be defined as testing from the view of a developer, where the tester has full access and complete knowledge regarding the target being tested as well as features the target has. This form of testing results in more vulnerabilities being found since the tester has complete access to the codebase, However, it can be more time consuming since there is a lot of information at hand. In contrast, black box testing is a form of external testing from the view of an attacker, where the tester has no knowledge of the technologies or frameworks the target was built on, only provided with the target URL using those techniques In the world of software development, the discussion around manual testing and test automation remains a central focus. Both approaches offer unique advantages and challenges, and the decision to utilize one or the other (or a combination) can significantly impact the quality and delivery pace of software projects. Penetration testing of the web applications is necessary prior to their launching and during their operation. The test can be performed either automatically or manually [4].

A. Automated Testing: Is a technique of using software tools to scan web application pages to discover vulnerabilities and generate reports at the end of the test. There are several tools used for automated testing such as OWASP ZAP, Burp Suite, Paros, W3af, etc.

- Automated tools scan the web application for known vulnerabilities, misconfigurations, and security weaknesses.
- These tools can cover a wide range of common web application vulnerabilities, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery
- Automated testing can be integrated into the application's continuous integration and deployment processes, providing a consistent and scalable.

B. Manual Testing: Sometimes automated testing is not enough to assess the vulnerabilities of the web application and there is a need for human intervention to perform the attacks as in social engineering.

- Experienced security professionals conduct manual analysis of the web application to uncover vulnerabilities.
- Manual testers can use a combination of automated tools and specialized techniques to identify complex, application-specific vulnerabilities.

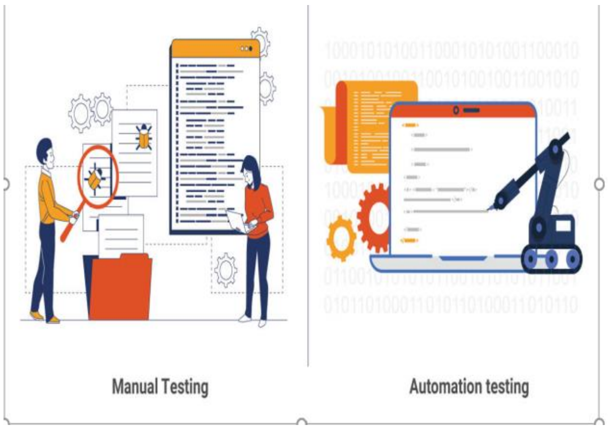


Figure 1 types of testing.

General definition the above testing methodology Web -based vulnerability assessments typically involve specialized software tools and techniques that automat or manually to identified vulnerabilities using black box and wait box penetration testing [5].

- Black box testing checks the accuracy of your system without too much concern about the internal implementations. It identifies the interface-level bugs and resolves them.
- White box testing deeply scrutinizes and analyzes your system by checking its internal coding.
- Manual penetration testing offers in-depth tests and avoids false positives
- Automated scans are faster and cover wider areas. They complement each other to create a strong security posture for your company

3.7 Identified vulnerability analysis tools

Selecting the right vulnerability assessment tool is critical to significantly impact your organization's cybersecurity posture. By understanding your specific needs, considering your budget, evaluating the tool's features, and assessing vendor reputation and support, you can make an informed choice that strengthens your defenses against cyber threats. Remember that cybersecurity is an ongoing process, and the right tool, coupled with a proactive and vigilant approach, can help your business stay resilient against evolving cyber threats. The time and effort invested in choosing the ideal vulnerability assessment tool will improve your organization's and its stakeholder's security and peace of mind.

Choosing vulnerability analysis tools is How to choose the right web application vulnerability assessment tool based on research comparison criteria and ranks the top open-source web-application-vulnerability-scanners. In this section, select from the most popular categories of tools for detecting vulnerabilities for online applications. It entails checking your web application and chooses a group of web application vulnerability analysis tools to look at. Think about commonly used commercial and open-source tools in the field. Use a variety of tools to address various facets of vulnerability assessment, including manual testing, dynamic analysis, and static analysis.

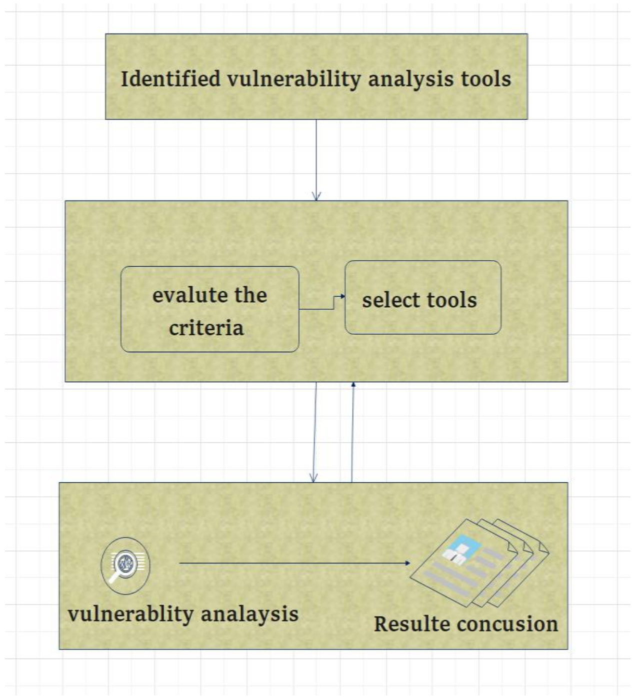


Figure 2. Identified vulnerability analysis methods.

3.8 Established hierarchical Examination criteria

Design Framework for Tools Examination is Design-Testing-Framework is a guideline used for creating and designing vulnerability analysis and comparative test-cases. A Framework is composed of a combination of practices and tools, which are designed to help analyze vulnerability and comparative the scan result from the given scenario more efficiently. Developing a framework to evaluate open-source web application vulnerability testing tools can aid in making sure that each tool is thoroughly assessed and contrasted. This is a recommended framework:

3.9 Examine the selected open-source tools

The technology industry has seen a significant rise in the adoption of open-source software, which now offers a wide array of tools and solutions for developers, data scientists, and other professionals. These open-source tools provide a valuable alternative to proprietary software, often boasting the added benefits of transparency, flexibility, and a thriving community of contributors.

In this examination, we will explore some of the most prominent and widely-used open-source tools across various domains, including version control, programming languages, machine learning, web development, and databases, containerization, and media creation. By understanding the capabilities, features, and use cases of these tools, we can better evaluate their suitability for different projects and tasks.

Evaluating the chosen open-source tools generally entails a thorough assessment and analysis of a group of freely available tools released under an open-source license. The goal is to understand their capabilities, limitations, and potential uses. The key steps involved in this process include: Identification and Selection of Tools:

- **Define Purpose and Requirements**: Clearly outline the specific needs or use case for which you are assessing the open-source tools.
- **Research Available Tools**: Conduct a comprehensive investigation to identify relevant open-source tools that meet your criteria.

- **Testing criteria** In order to evaluate the efficacy and quality of a software system or application, testing criteria are crucial components of the testing process. Consider the following important testing parameters Insecure websites and web applications are susceptible to well-known security risks, including cross-site scripting, SQL injection, security misconfiguration, cookie theft, self-propagating worm attacks, and session hijacking [6].

3.10 Perform web application vulnerability analysis

A vulnerability analysis is the testing process used to identify and assign severity levels to as many security defects as possible in a given timeframe. This process may involve automated and manual techniques with varying degrees of rigor and an emphasis on comprehensive coverage to compare and contrast the final result.

Perform web application scanning is open-source web application scanners to analysis vulnerabilities on Web servers, proxy servers, and other Web services. The essential elements of web-based application the capacity to evaluate various Web technologies, assess the security of a web application, and gather information regarding web security status. The results of this scanning process help to this study comparison of automated open-source web vulnerability analysis tools capacity. Define the KPIs that will be used to gauge the weakness of the tools' effectiveness. Vulnerability detection rate, false positives rate, scan time, configuration ease of use, and reporting capabilities are a few examples. The metrics have to be aligned with the goals specified in step.

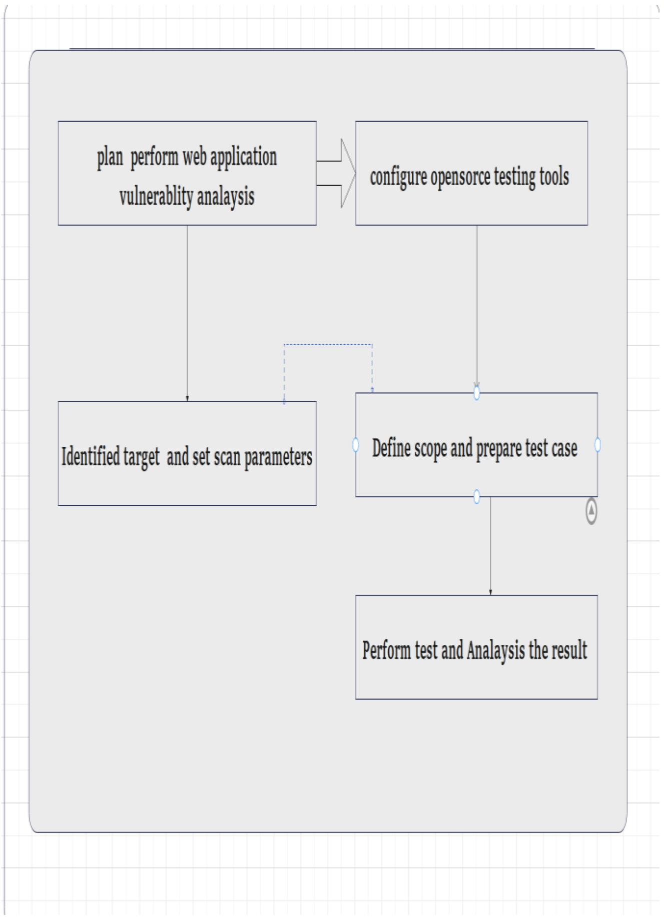

plan perform web application vulnerablity analaysis
configure opensorce testing tools
Identified target and set scan parameters
Define scope and prepare test case
Perform test and Analaysis the result

Figure 3. Perform web application vulnerability analysis

3.11A Holistic Examination and Comparative Study

Existing web application vulnerability visualization tools do not have a holistic approach that combines vulnerability scanner results with the environment properties and timeline of security-related activities. They also lack clear metric definitions and the number of metrics they present is very low in general, resulting in incomplete picture of the security status of web applications. Security Information and Event Management (SIEM) tools, on the other hand, provide a more holistic approach, with a large number of metrics for the enterprises. However, rather than focusing on web application security, they aim to serve other purposes, such as reporting and managing security alerts in real time based on the analysis of data obtained from the entire IT infrastructure [7].

3.12 Experiment Environment and requirements

The experiment utilized intentionally vulnerable web applications to create a controlled environment for testing and evaluating the effectiveness of the vulnerability scanning tools. By leveraging the capabilities of Kali Linux, the web application vulnerability scanners, and the vulnerable web applications, the experiment aimed to assess the performance and effectiveness of the selected security assessment tools in identifying and reporting vulnerabilities in web-based systems. To perform the experiment, we utilized the following tools and resources:

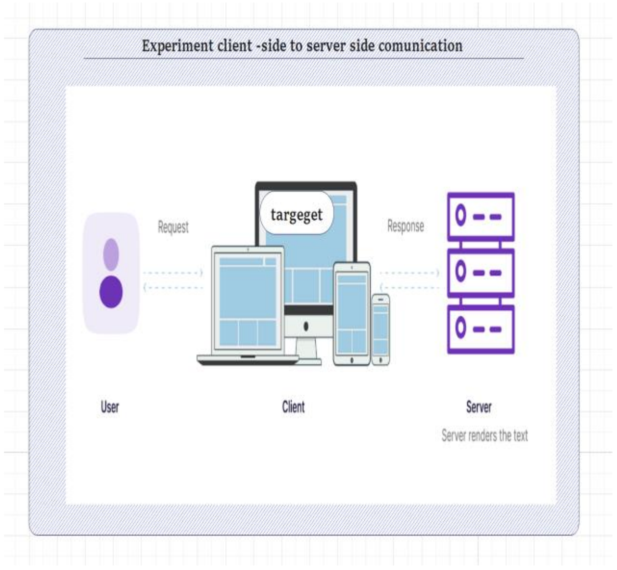


Figure 4 Experiment Environment setup.

- Client-side/user is that the action takes place on the user's (the client's) computer.
- Server-side is that the action takes place on a web server.
- A target is a website, web application or server that you would like to scan for security vulnerabilities

3.13 Operating System:

Linux is a Unix-like, open source and community-developed operating system (OS) for computers, servers, mainframes, mobile devices and embedded devices. It is supported on almost every major computer platform, including x86, Scalable *Processor Architecture* (*SPARC*) making it one of the most widely supported operating systems.

Kali Linux (formerly known as Backtrack Linux) is an open-source, Debian-based Linux distribution which allows users to perform advanced penetration testing and security auditing. It runs on multiple platforms and is freely available and accessible to both information security professionals and hobbyist it is popular penetration testing and ethical hacking distribution, was the operating system of choice for this experiment to perform the experiment, the researchers utilized Kali Linux as a virtual machine running in Oracle Virtual Box. Kali Linux is an open-source operating system based on Debian that is widely used for penetration testing and digital forensics.

Employing Kali Linux provided the researchers with a comprehensive suite of specialized security testing tools and utilities out-of-the-box. Kali Linux's extensive collection of hundreds of penetration testing programs, vulnerability scanners, and hacking tools made it a popular choice for security professionals and researchers conducting security assessments and experiments. By running Kali Linux within the Oracle Virtual Box virtualization environment, the researchers were able to create a controlled, isolated testing setup separate from their primary operating system. This virtualized approach ensured the security tools and exploits could be safely tested without impacting the host system. The combination of Kali Linux's security-focused distribution and the isolation provided by the Oracle Virtual Box virtual machine created an ideal platform for the researchers to thoroughly analyze the target web applications and identify potential vulnerabilities [8].

Vulnerability Analysis is one of the most important phases of Hacking. It is done after Information Gathering and is one of the crucial steps to be done while designing an application. The cyber-world is filled with a lot of vulnerabilities which are the loopholes in a program through which hacker executes an attack. These vulnerabilities act as an injection point or a point that could be used by an attacker as a Launchpad to execute the attack.

Kali Linux comes packed with 300+ tools out of which many are used for vulnerability analysis. Though there are many tools in Kali Linux for vulnerability analysis here is the list of most used tools.

An essential Kali Linux feature Kali ISOs with complete customization. An optimal version of Kali for your particular demands is always simple to build thanks to the use of metapackages geared for the distinct need sets of a security expert and an extremely accessible ISO customization process.

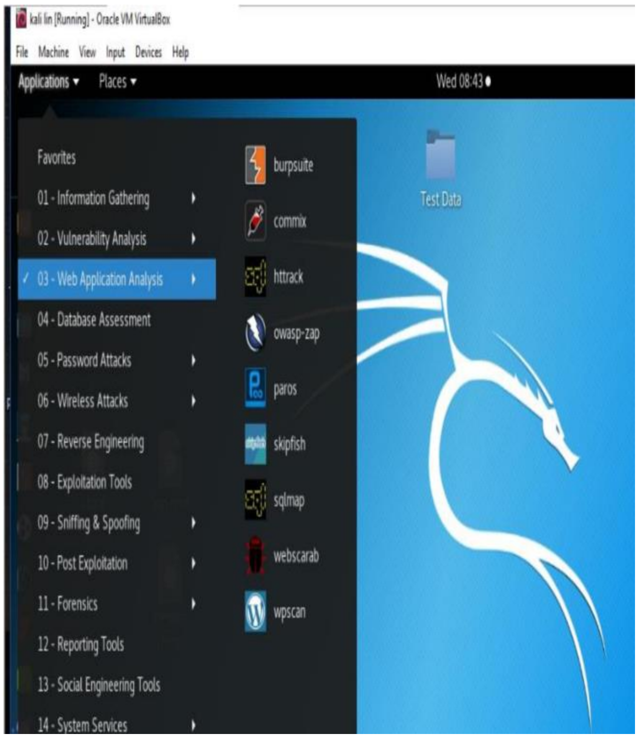


Figure 5 kali linux oprating system

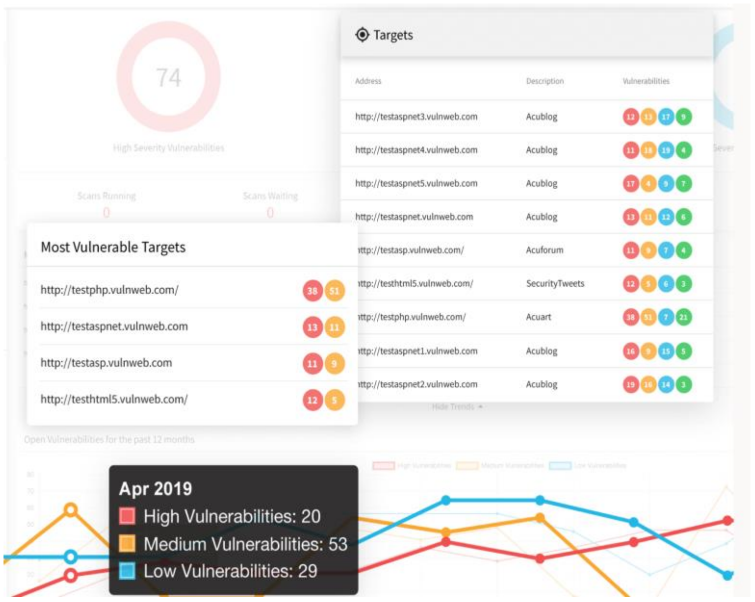
3.14Web Application open source Vulnerability Scanners:

The researchers utilized several industry-standard web application vulnerability scanning tools to assess the security posture of the target web applications. These scanning tools are designed to automatically identify and report on a broad range of vulnerabilities that could potentially be exploited by malicious actors. These industry-leading vulnerability scanning tools systematically analyzed the target web applications, identifying a wide range of potential weaknesses that could be exploited, such as cross-site scripting (XSS), SQL injection, authentication flaws, and more. The findings from these scans provided valuable insights that could be used to improve the overall security posture of the tested web application. Various web application vulnerability scanning tools to assess the security posture of the target web applications. Even with the knowledge of how to write secure code, knowing what inputs that are invalid or not, the risk of making a mistake that opens up the possibilities of an attack is still probable. If it is about making a small mistake in a larger system or just ignorance in a smaller system, making mistakes are in human nature and this leads up to the need to continuously test the application for any part that might be vulnerable to an attack. This can be done in several distinct ways which all have different advantages and disadvantages. They can mainly be divided into two groups; these are white box and black box testing [9]. These tools included industry-standard scanners such as:

- Burp Suite an automated dynamic application security testing (DAST) web vulnerability scanner.
- OWASP Zap (Zed Attack Proxy)
- w3af (Web Application Attack and Audit Framework
- Nikto - An open-source web server scanner used to identify potential vulnerabilities.
- Uniscan - An open-source web vulnerability scanner that can identify a wide range of web application
- Acunetix, a leading web application vulnerability scanner, was one of the tools used for experiment to assess *the security of the target web applications such as blow image.*

3.15Performance-Based Comparative Analysis Open Source Web Vulnerability Scanners.

The widespread adoption of web vulnerability scanners and the differences in the functionality provided by these tool-based vulnerability detection approaches increase the demand for testing their detection effectiveness. Although there are many previously conducted research studies that addressed the performance characteristics of web vulnerability detection tools by either quantifying the number of false alarms or measuring the corresponding crawler coverage, the scope of the majority of these studies is limited to commercial tools. The results of our comparative evaluation of a set of open source scanners highlighted variations in the effectiveness of security vulnerability detection and indicated that there are correlations between different performance properties of these scanners (e.g., scanning speed, crawler coverage, and number of detected vulnerabilities) [10]



Targets

Address	Description	Vulnerabilities
http://testaspnet3.vulnweb.com	Acublog	12 13 17 9
http://testaspnet4.vulnweb.com	Acublog	11 18 19 4
http://testaspnet5.vulnweb.com	Acublog	17 4 9 7
http://testaspnet.vulnweb.com	Acublog	13 11 12 6
ttp://testasp.vulnweb.com/	Acuforum	11 9 7 4
ttp://testhtml5.vulnweb.com/	SecurityTweets	12 5 6 3
ttp://testphp.vulnweb.com/	Acuart	38 51 7 21
ttp://testaspnet1.vulnweb.com	Acublog	16 9 15 5
ttp://testaspnet2.vulnweb.com	Acublog	19 16 14 3

Most Vulnerable Targets

http://testphp.vulnweb.com/	38 51
http://testaspnet.vulnweb.com	13 11
http://testasp.vulnweb.com	11 9
http://testhtml5.vulnweb.com/	12 5

Apr 2019
🟥 High Vulnerabilities: 20
🟧 Medium Vulnerabilities: 53
🟦 Low Vulnerabilities: 29

Figure 6 Web Application open source Vulnerability Scanners

3.16 Vulnerability metrics

The Common Vulnerability Scoring System (CVSS) is a technique for providing a subjective assessment of vulnerability. The CVSS is not a danger indicator. Base, temporal, and environmental are the three metric groupings that make up CVSS v2.0 and v3.x. The Base, Threat, Environmental, and Supplemental metric categories make up the somewhat altered CVSS v4.0. A numerical score between 0 and 10 is produced by metrics. Another way to see a CVSS assessment is as a vector string, which is a condensed text representation of the scores' derived values. For businesses, institutions, and governments that require precise and reliable vulnerability severity scores, CVSS is a suitable standard measurement approach. The severity of vulnerabilities found on one's systems can be calculated using two popular applications of CVSS.

In addition to the qualitative severity ratings for CVSS v3.x and CVSS v4.0 as specified in their respective specifications, the NVD also notes the qualitative severity ratings of "Low," "Medium," and "High" for CVSS v2.0 base score ranges.

3.17 Vulnerabilities Detected by Web Application Testing Tools

Most critical security risks to web applications are OWASP Top 10 vulnerabilities are The OWASP Top 10 is the most well-known OWASP document, which outlines the most prevalent web application vulnerabilities. In contrast, the document you're referring to focuses on defensive techniques and controls, rather than risks. For each control in this document, there is a mapping to one or more items from the risk-based OWASP Top 10, which is provided at the end of the control descriptions.

The detection of these issues underscores the importance of conducting regular security assessments on web applications. Addressing these vulnerabilities promptly is crucial to safeguarding the application and protecting its users. Regular security evaluations using open source tools can help identify and mitigate potential threats, ensuring the application remains secure and resilient against malicious attacks. Proactively addressing these critical vulnerabilities is a fundamental aspect of maintaining the overall security posture of the web application security some of Open source web application security tools are frequently capable of detecting critical vulnerabilities such as:

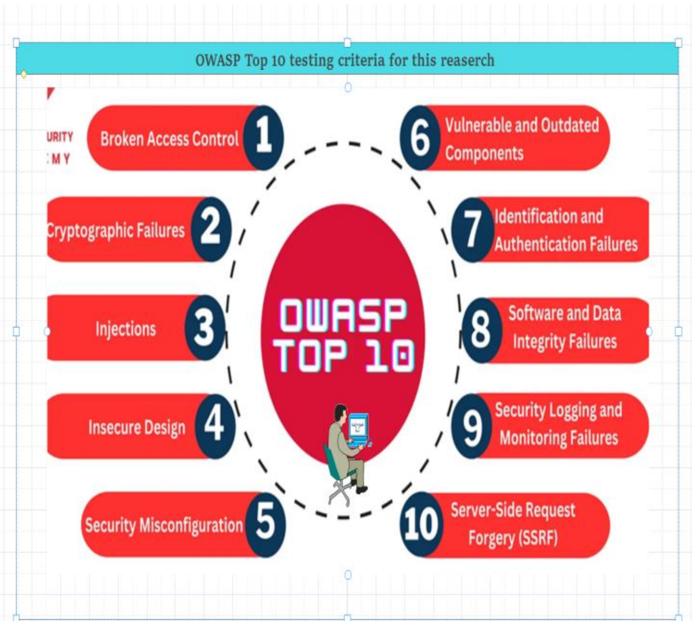

OWASP Top 10 testing criteria for this reaserch
OWASP TOP 10
1 Broken Access Control
2 Cryptographic Failures
3 Injections
4 Insecure Design
5 Security Misconfiguration
6 Vulnerable and Outdated Components
7 Identification and Authentication Failures
8 Software and Data Integrity Failures
9 Security Logging and Monitoring Failures
10 Server-Side Request Forgery (SSRF)

Figure 7 Vulnerabilities Detected by Web Application Testing Tools

Some definition OWASP TOP 10 is defined blow.

A. **Broken Access Control**

This occurs when users can improperly access functionality or data that they should not have permission to access. This can happen due to flaws in how the application implements and enforces access control checks, insecure API design, or a failure to properly restrict access to sensitive data and features.

Some examples of Broken Access Control include:

- Accessing administrative functions or sensitive data without the appropriate authorization
- Modifying URL parameters to gain access to resources the user should not be able to access
- Accessing other users' accounts by manipulating session tokens or IDs
- Vertical privilege escalation - accessing higher-level functionality beyond what the user's role should permit
- Horizontal privilege escalation - accessing data or features that belong to other users

B. **Cryptographic Failures**

This Occur when sensitive data is not adequately protected through the proper use of cryptography. This can result in the exposure of sensitive information, such as passwords, credit card numbers, and other personal data.

Some examples of Cryptographic Failures include:

- Using weak or outdated encryption algorithms that are no longer considered secure
- Improper key management practices, like storing encryption keys in plaintext
- Failing to encrypt sensitive data while it is in transit or at rest
- Using predictable initialization vectors or salts, which can compromise the cryptographic protections.

C. **Injection vulnerabilities**

This occur when untrusted data is passed to an interpreter as part of a command or query. This can lead to the execution of unintended commands or the access of unauthorized data.

Some examples of Injection flaws include:

- SQL Injection: Where malicious SQL queries are inserted into application inputs, potentially allowing unauthorized access to the database.
- OS Command Injection: Where malicious system commands are injected into application inputs, potentially allowing the execution of arbitrary commands on the underlying operating system.
- LDAP Injection: Where malicious LDAP queries are inserted into application inputs, potentially allowing unauthorized access to directory services.

D. **Insecure Design**

This refers to flaws in the conceptual security design and architectural decisions that can leave applications vulnerable to attacks, even if the implementation is otherwise done correctly.Some examples of Insecure Design include:

- Failure to consider security requirements and risks during the initial design and planning phase of the application or system.
- Lack of thorough threat modeling and risk analysis to identify and mitigate potential security threats.
- Inadequate security controls and safeguards built into the overall system architecture.

E. Security Misconfiguration

This arises when the security-related settings and configurations in an application, framework, library, or underlying platform are not properly set, leaving the application vulnerable to potential attacks.

Some examples of Security Misconfiguration include:

- Leaving default or weak credentials or configurations in place, which can be easily exploited.
- Enabling unnecessary features or services that may introduce additional security risks.
- Improperly setting file or directory permissions, potentially allowing unauthorized access.
- Exposing sensitive information, such as error messages or logs that could be leveraged by attackers.

F. Vulnerable and Outdated Components.

This can expose the application to exploitation and potential compromise, as attackers may leverage these vulnerabilities to gain unauthorized access or perform malicious activities.

Some examples of Vulnerable and Outdated Components include:

- Using versions of libraries or frameworks that have publicly disclosed security vulnerabilities.
- Failing to regularly update the components used in the application to the latest versions that address known security issues.
- Neglecting to monitor for and promptly address newly discovered vulnerabilities in the components being utilized.

G. Identification and Authentication Failures

This vulnerability occur when an application's mechanisms for identifying and authenticating users are not properly implemented, leading to vulnerabilities that can be exploited by attackers to compromise user accounts or escalate their privileges.

Some examples of Identification and Authentication Failures include:

- Allowing the use of weak or commonly used passwords that are easily guessed or cracked.

- Failing to implement robust multi-factor authentication, which provides an additional layer of security beyond just a password.
- Improper handling of session management or session tokens, which can allow attackers to hijack user sessions.
- Lack of account lockout or brute-force protection mechanisms, which can leave the application vulnerable to automated attacks.

H. Software and Data Integrity Failures

This refer to situations where an application does not properly verify the integrity of software updates, critical data, or the underlying infrastructure it relies on. This can lead to the introduction of malicious code or the compromise of sensitive data.

Some examples of Software and Data Integrity Failures include:

- Failing to implement robust mechanisms to verify the integrity and authenticity of software updates or patches before installing them on the application.
- Lack of controls to detect and prevent the tampering or unauthorized modification of sensitive data, either at rest or in transit.
- Improper handling of user-supplied inputs, which could allow the injection of malicious content that compromises the application's integrity.
- To address Software and Data Integrity Failures, organizations should:
- Implement secure software update processes that include verifying the digital signatures or hash values of updates to ensure their authenticity and integrity.
- Employ data integrity checks, such as cryptographic hashing or digital signatures, to detect any unauthorized changes or tampering of sensitive data.
- Implement strict input validation and sanitization techniques to prevent the injection of malicious code or content into the application.
- Regularly review and test the integrity of the application, its dependencies, and the underlying infrastructure to identify and address any potential weaknesses.
- Maintain a comprehensive inventory of all software components and dependencies, and monitor for any security updates or vulnerabilities that may affect the application's integrity.

I. Security Logging and Monitoring Failures

This vulnerability occur when an application does not properly record security-relevant events or monitor for suspicious activity, making it challenging to detect and respond to security incidents in a timely manner.

Some examples of Security Logging and Monitoring Failures include:

- Lack of logging or insufficient logging of security-relevant events, such as failed login attempts, privilege escalations, or data access.
- Failure to actively monitor and analyze the application's logs for signs of potential security incidents or anomalous behavior.
- Absence of alerting or notification mechanisms to promptly inform security teams or administrators of critical security events.

J. Server-Side Request Forgery (SSRF)

This vulnerability occurs when an application allows an attacker to indirectly make arbitrary HTTP requests from the server-side. This can potentially grant the attacker unauthorized access to internal resources or sensitive data.

Some examples of SSRF vulnerabilities include:

- Allowing users to specify arbitrary URLs that the server will then fetch and process, potentially exposing internal or sensitive resources.
- Failure to properly validate and sanitize user-supplied URLs, which can lead to the application making requests to unintended or malicious destinations.

- Lack of protective measures to prevent the application from accessing internal or sensitive resources, such as through internal IP addresses or services.

3.18 Organized and conclude the Examine the result,

Which is the final evaluation of open-source vulnerability scanner tool analysis results for web applications. Vulnerability identification includes recording the vulnerabilities found, their impact, and their severity. It also includes looking at the final result design matrix based on features, testing performance, accuracy, community support, ease of use, and testing scope. This examines the result and makes sense for this study's comparative analysis because the research results are compared based on this result.

The purpose of the proposed study is to provide a comprehensive evaluation of open source web site security testing tools. The aim of the research is to identify the most effective means of detecting vulnerabilities in web applications. The results of this study provide developers with insight into the selection of web application testing tools. Ultimately, the study contributes to the security of web applications, which benefits society as a whole finally after testing the research experiment I will organize and conclude the Examine the result compared based on criteria format in table and graph.

3.19CONCLUSION

This study provides a comprehensive examination of vulnerability testing tools for open source web applications using a strategic and comparative methodology. It introduces a variety of tools, including OWASP ZAP, Burp Suite Community Edition, and Nikto, each with their unique strengths in identifying different vulnerabilities. The effectiveness of these tools has been found to be highly context-dependent and influenced by factors such as the architecture of the target web application and the tester's familiarity with the tool. Ease of use and user experience proved to be crucial criteria for selecting these tools. With users preferring those that have intuitive interfaces and thorough documentation. This emphasis on usability highlights the need for a balance between a tool's functionality and its usability to ensure that security professionals can effectively use these tools in their workflows. The study also highlights the importance of community support and regular updates are crucial to the sustainability and further development of these tools. Tools supported by active communities tend to be more responsive to emerging threats and enable organizations to maintain a proactive security posture. Additionally, the study highlights the importance of integrating vulnerability testing tools into continuous integration/continuous deployment (CI/CD) pipelines. This integration facilitates automation and increases the efficiency of security practices, making it easier for organizations.

This holistic review and comparison study highlights the importance of strategically selecting and applying open source tools for testing web application vulnerabilities. By leveraging the results of this research, organizations can improve their security frameworks and better defend against web-based threats identification and use the best open source web application tools The OWASP Top 10 Benchmark Project can be extended to other benchmarks and real-life vulnerable environments as long as it is possible to provide deep results that help in choosing the best tool depending on the required task [11].

3.20 REFERENCES

1 References

[1] J. R. A. ,. A. J. L. O. DINIS BARROQUEIRO CRUZ, Open Source Solutions for Vulnerability Assessment: A Comparative Analysis, ieee access/2https://cwe.mitre.org, 2023.

[2] S. M. S. Kabir, METHODS OF DATA COLLECTION, https://www.researchgate.net/, 2016.

[3] A. F. Maskur, "Static Code Analysis Tools with the Taint Analysis Method for Detecting Web Application Vulnerability," *2019 International Conference on Data and Software Engineering (ICoDSE),* p. 1, 14 May 2020.

[4] C. Khounborine, A Survey and Compar y and Comparative Study on V e Study on Vulnerability Scanning T ability Scanning Tools, ScholarWorks@UARK, 2023.

[5] M. BAYKARA, International Journal of Computer Science and Mobile Computing, Monthly Journal of Computer Science and Information Technology, 2018.

[6] O. N.-B. Isaac Kofi Nti, "Investigating Websites and Web Application Vulnerabilities: Webmaster's Perspective," *International Journal of Applied Information Systems (IJAIS),* 2017.

[7] ,. (. I. A. B. G. K. FERDA ÖZDEMIR SÖNMEZ, "Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic," *https://ieeeaccess.ieee.org/,* February 4, 2021,.

[8] 1. H. S. Abdullah, Evaluation of Open Source Web Application Vulnerability Scanners, Academic Journal of Nawroz University (AJNU), 2020.

[9] E. Matti, Evaluation of open source web vulnerability scanners and their techniques used to find SQL in jection and cross-site scripting vulnerabilities, Linköpings universitet, 2021.

[10] 2. A. A. Noura Alomar, "Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners," *Security and Communication Networks,* 2017.

[11] *. ,. D. A. 1. a. A. J. 2. Marwan Albahar 1, "An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities," *Electronics is an international, peer-reviewed,* Published: 21 September 2022.

[12] P. b. E. B.V., "How-to conduct a systematic literature review: A," *journal homepage: www.elsevier.com/locate/mex,* p. 1, 2022.

[13] S. S. A. Ömer Aslan, "A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions," *electronics,* p. 3, 11 March 2023.

[14] b. C.-H. T. b. Yao-Wen Huang a, "A testing framework for Web application security assessment q," *Y.-W. Huang et al. / Computer Networks 48 (2005) 739–761,* p. 140, 12 February 2005.

[15] G. Vishveshwarya, "COMPARATIVE STUDY AND EVALUATION OF WEB BASED AUTOMATION TESTING TOOLS," *International Journal of Computer Application,* April 2017.

[16] A. S. E B Setiawan*, "Web vulnerability analysis and implementation," *IOP Conf. Series: Materials Science and engneering,* p. 1, 2018.

[17] 2. (. I. A. D. C. D. L. SULIMAN ALAZMI, "A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners," *ieee acess,* y 18, 2022.

[18] E. B. D. G. J. M. Jason Bau, "State of the Art: Automated Black-Box Web Application Vulnerability Testing".

[19] 1. S. J. 1. Student, "Vulnerability Scanning," International Journal of Engineering Development and Research (www.ijedr.org), 2020.

[20] N. B. S. M. Anuja Bokhare#1, "Benchmarking of testing tools used for web services," *International Journal of Applied Engineering Research,* 2015.

[21] K. S. Arif and U. Ali, "Mobile Application testing tools and their challenges: A comparative study," *International Conference on Computing, Mathematics and Engineering Technologies,* 2019 .

[22] J. C. P. K. K. D. T. Richard Amankwah, "An empirical comparison of commercial and open-source web vulnerability scanners," *https://doi.org/10.1002/spe.2870,* 03 July 2020.

[23] J. Serra-Ruiz, V. Cavaller and J. Cano, "A Comprehensive Cybersecurity Audit Model to Improve Cybersecurity Assurance: The CyberSecurity Audit Model (CSAM)," *2017 International Conference on Information Systems and Computer Science (INCISCOS),* November 2017.

[24] R. v. Roessing, "Improving the IT Security Audit Framework: Standards, Common Ground, and Strategic Alignment," *Proceedings of Security and Protection of Information, 2005,* 2005.

[25] A. O. A. A. A. I Atoum, "A holistic cyber security implementation framework implementation framework," *Information Management & Computer Security,* 2014.

[26] N. M. Vithanage and N. Jeyamohan, "WebGuardia - an integrated penetration testing system to detect web application vulnerabilities," *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET),* 2016 .

[27] Z. ĐURIĆ, "WAPTT - Web Application Penetration Testing," *Advances in Electrical and Computer Engineering ,* 2014.

[28] N. Singh, V. Meherhomji and B. R. Chandavarkar, "Automated versus Manual Approach of Web Application Penetration Testing," *11th International Conference on Computing, Communication and Networking Technologies (ICCCNT),* 2020.

[29] J. H. J. M. JRB Higuera, "Benchmarking Approach to Compare Web Applications Static Tools Detecting OWASP Top Ten Security Vulnerabilities," *researchgate.net,* 2020.

[30] D. A. A.-H. Assem I. Mohaidat1, "Web Vulnerability Scanning Tools: A Comprehensive Overview,Selection Guidance, and Cyber Security Recommendations," *International Journal of Research Studies in Computer Science and Engineering (IJRSCSE),* 2024.

[31] A. Fuggetta, "Open source software—an evaluation," *The Journal of Systems and Software ,* p. 77–90, (2003) .

[32] R. V. P. Urshila Ravindran1, "A Review on Web Application Vulnerability Assessment and Penetration Testing," *IIETA | Advancing the World of Information and Engineering,* p. 2, 22 December 2021.

[33] L. Zhang, A. Stoffel, S. M. Michael Behrisch, T. Schreck and R. Pompl, "A comparative review of state-of-the-art commercial systems," *IEEE,* 2012 IEEE Conference on Visual Analytics Science and Technology (VAST).

[34] ·. J. A. C. Paul A. Wortman1, "A framework for evaluating security risk in system design," *Discover Internet of Things,* 2022.

[35] R. P. Petar Cisara, "Some ethical hacking possibilities in Kali Linux environment," *Journal of Applied of Technical and Educational Sciences jATES,* 2019.

[36] R. N. Seema Rani1, "PENETRATION TESTING USING METASPLOIT FRAMEWORK: AN," *International Research Journal of Engineering and Technology (IRJET),* 2019.

[37] M. A. a. R. Ibrahim, "A Comparative Study of Web Application," *International Conference on Communication and Computer EngineeringAt: Malaysia,* November 2014.

[38] N. S. a. J. G. Samer Zein a, "A systematic mapping study of mobile application testing techniques," *Journal of Systems and Software,* July 2016.

[39] A. S. A. Susana Paola Lainez Garcia, "A Comparative Analysis of Web Application Vulnerability Tools," *Journal of Information Systems Applied Research,* July 2023.

[40] P. Kaur, I. Sharma and A. Kaur, "5th International Conference on Information Systems and Computer Networks (ISCON)," *IEEE,* 2021 .

[41] G. Deepa, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," *Information and Software Technolog,* 2016.

[42] S. N. F. Holik, "Vulnerabilities of Modern Web Applications," *2017 40th International Convention on Information and ..., 2017•ieeexplore.ieee.org,* 2017.

[43] ∗. P. L. M. P. a. A. D. K. Dimitris Mitropoulos, "Defending Against Web Application Attacks:Approaches, Challenges and Implications," *IEEE Transactions on Dependable and Secure Computing, 2017•ieeexplore.ieee.org,* 2017.

[44] M. A. I. N. O. N.-B. V Appiah, "Survey of Websites and Web Application Security Threats Using Vulnerability Assessment," *Journal of Computer Science, 2018•academia.edu,* 2018.

[45] A. S. N. Shuvalaxmi Dass, "Vulnerability coverage for adequacy security testing," *35th Annual ACM Symposium on Applied ComputingMarch,* p. Pages 540–543, 2020.

[46] Frank der Loo - Netherlands, "Comparison of penetration testing tools for web applications," *Masters thesis from Radboud Universiteit,* August 15, 2011.

[47] A. Parashar, "Web-based OWASP MITM Attack Model for Vulnerability Assessment and Penetration Testing of Web Applications : WebProbe," *Samrat Ashok Technological Institute,* 2024.

[48] V. F. F. Matteo Esposito, "An Extensive Comparison of Static Application Security Testing TOOLS," *28th International Conference on Evaluation and Assessment in Software Engineering,* 2024.