International Journal Software Engineering and Computer Science (IJSECS)
4 (3), 2024, 1109-1127
Published Online December 2024 in IJSECS (http://www.journal.lembagakita.org/index.php/ijsecs)
P-ISSN: 2776-4869, E-ISSN: 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v4i3.3138.

RESEARCH ARTICLE Open Access

Identifying and Mitigating Web Application Vulnerabilities: A Comparative Study of Countermeasures and Tools

Sayed Elham Sadat *
Information Technology Department, Kabul Education University, Kabul, Afghanistan.
Corresponding Email: s.elham2011@gmail.com.

Mohammed Fahim Naseri
Information Technology Department, Kabul Education University, Kabul, Afghanistan.
Email: fahim.naseri456@gmail.com.

Khosraw Salamzada
Information Technology Department, Kabul Education University, Kabul, Afghanistan.
Email: kh.salamzada@gmail.com.

Received: August 7, 2024; Accepted: November 10, 2024; Published: December 1, 2024.

Abstract: In the current age of technology, web applications and websites have experienced significant growth. This expansion has made their security a critical area of research. Web applications offer benefits, which makes user's lives easier. In this paper, common web application vulnerabilities and effective strategies to mitigate the vulnerabilities are identified using a comparative study of countermeasures and open-source web application vulnerability assessment tools. Specifically, the top ten web application vulnerabilities and their countermeasures are investigated. Accordingly, several open-source vulnerability assessment tools are also introduced. The review highlights that with the developments and deployments of web applications on the internet, users are chased by a remarkable number of cyber-attacks. Attackers take advantage of available vulnerabilities in a web application or website, such as SQL injections, cross-site scripting, and broken authentications. This paper concludes by providing the best practices to mitigate cyber-attacks on web applications and suggests future directions for enhancing vulnerability assessment through machine learning techniques.

Keywords: Web Security; Vulnerabilities; Web Applications; Open-Source Tools; Countermeasures.

1. Introduction

In today's digital world of ever-increasing technology, it would be difficult or almost impossible to live without these technological advancements. In respect to this, cybercrimes have a noticeable increase, and everyday thousands of devices and services are being attacked in cyberspace in order to access important information or organizations. Therefore, cybersecurity has gained much importance and become a part of our lives. Whenever a person is submitting or uploading their data online on the internet, that data becomes vulnerable to a huge number of cyber-attack or cybercrimes, which cause significant damage to the businesses and services associated with government or organizations. Cybersecurity offers security measures and techniques to prevent unauthorized access or exploitation of massive data available online. These security measures are applied to applications, networks used for communication, and devices connected to the network.

Cybersecurity involves the protection of important information and also devices from cyberattacks on cyberspace occurring. Collection of customers information, social platforms where personal and private information is being collected, and government organizations where secret, defense, and political information are maintained and managed in huge databases. It describes how personal and governmental data can be protected from vulnerable attacks on cyberspace. With respect to growth in the number of web- based services users every day, there is a huge increase in the number of threats to information as well, with the cost of cybercrimes estimated in billions [1]. Web applications are one of the great technologies which has had a frequent improvement in the last decade, web application technology improved both the quality and ability of the services offered by most of the organizations. Web applications includ*et al*most all of today's life and have become a major part of today's business which made it easy to have interactions and communications with customers. Websites and web applications are the combination of one or more than one web page which is being accessed using browsers on devices connected to a server.

As the improvements and the need to access the web applications and web sites are increasing day by day, there is a huge risk that unauthorized persons or identity on the network (internet) will access or modify the confidential data and maintain the integrity among them. Software and web applications are designed and developed with the motivation of providing features and functionalities to address the needs of users and customers by providing easier usage. Web application security emphasis on different software bugs available in an application which will cause the application to do something bad [2].

Websites that provide huge services and facilities like Banking, Business transactions, online shopping, social networking are having a higher risk of being victims of unauthorized access or modification of information stored in databases. They should keep the services up to date and also identify the vulnerabilities and loopholes to make their services and confidential information secure. Consequently, the enormous growth in the number of vulnerabilities and attacks on web sites and web applications make it important to study and identify the vulnerabilities to secure web applications and websites from being a victim.

The real attack on a web site or web application happens when the transaction is being initiated between the server and the client and later on the server sends the information to the client. When the application is being rendered it has a greater chance that the intruder or attacker input the malicious code through the browser. Accordingly, common types of miss configurations such as full path disclosure, error reporting, standard passwords, default settings, and many other weaknesses which cause information leakages which may have a good value for intruders to attack and access the services are available in four out of five web applications. In 2018, there was a fall in the percentage of web applications having vulnerability to XML External Entities (XXE). This vulnerability entered the OWASP Top 10 list in 2017, immediately taking fourth place [3], shown in Figure 1.

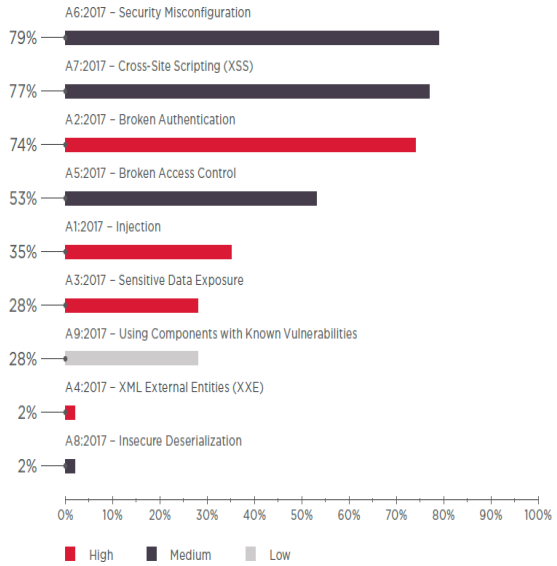

Figure 1. OWASP Top 10–2017 vulnerabilities percentage available in web apps.

The measures to these security threats and attacks on web applications is to identify the vulnerabilities and understand the attacks and countermeasures to them and ask the developer to fix the loopholes available at the application [3]. Accordingly, the main contribution of this paper is to:

1) Investigate the top ten web application vulnerabilities which poses huge risk to organization.
2) Discuses various countermeasures which can be implemented to mitigate web application vulnerabilities and reduce the risk of exploitation.
3) Introduce and evaluate various open-source vulnerability assessment tools which can be used to identify and assess web application vulnerabilities.
4) Highlight the importance of proactive vulnerability management and ongoing monitoring of web applications to ensure that they remain secure over time.

This study aims to provide an overview of the top 10 vulnerabilities in web applications that pose a high risk to organizations, the precautions that can be taken to avoid such risks and the effectiveness of various open-source vulnerability assessment tools in detecting and preventing Web Application security exploits. Therefore, this study illustrates how implementing such a proactive vulnerability management approach, along with continuous monitoring, is critical to securing the long-term resilience of web applications.

2. Research Method

In this paper, a systematic review and analysis of the most common web application vulnerabilities will be conducted together with top ten open-source web application vulnerability assessment tools. The research design consists of a comprehensive review of existing literature, an evaluation of key vulnerability types, and a comparison of the tools' effectiveness in the identification and mitigation of these vulnerabilities.

2.1 Data Sources and Collection

The data in this study were collected using a comprehensive literature review of scientific research articles, reports on web applications which are available publicly along with the performance of open-source web application assessment tools.

1) Research Reports and Scientific Articles
   The studies which were done by Positive Technologies [1], Al-Sanea *et al.* [2], provided insights into the prevalence of different vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS).
2) Industrial Evaluations
   Documents from industries like OWASP or PortSwigger were selected, to determine the effectiveness of OWASP ZAP or W3AF in real world security assessment procedures.

2.2 Evaluation of Web Application Vulnerabilities

The study focuses on the OWASP Top 10 vulnerabilities, such as SQL Injection, Cross-Site Scripting (XSS), and Broken Authentication. These Vulnerabilities are examined based on the following criteria:

1) Prevalence
   The likelihood of the vulnerability in real world web applications, as demonstrated by studies and industry reports [5][6].
2) Impact
   The possible harm or exploitation might result from failing to mitigate vulnerability.
3) Causes
   The Common errors in programming misconfiguration or weaknesses in security procedures that lead to the existence of each vulnerability.

2.3 Selection of Vulnerability Assessment Tools

The web application vulnerability assessment tools were selected based on literature review and public industrial reports. The tools are selected based on the following terms.

1) Open source
   The criterion should be satisfied in terms of freeness, accessibility and popularity.
2) Documented effectiveness
   The selection was based on previous studies, technical and industrial reports that presented the strengths and weaknesses of the tools.
3) Covering OWASP Top 10 Vulnerabilities
   The tools should effectively identify the wide range of vulnerabilities, specifically those on the list of OWASP top 10.

2.4 Comparison of Vulnerability Assessment Tools

The analysis and comparison of open-source web assessment tools was based on documented capabilities, case studies, and review. The tools were compared based on their performance in detecting the OWASP top 10 vulnerabilities.

3. Result and Discussion

3.1 Results

3.1 Working of Web Applications

Many businesses, organizations and government agencies use the internet as a cost-effective, fast and secure communication medium with their customers and users. The effectiveness of this technology is when the organization can collect and store the data and has a proper system to analyze, process and present the data back to the user, they provide the functionality to the users to use the services without downloading and installing any other software. Web applications are composed of one or many web pages that are stored and maintained on a server, and the users can access them using a web browser already installed on their devices, shown in Figure 2. Web applications together use server-side scripting languages like PHP and ASP to manage the storage and retrieval of the information, client-side scripting languages like JavaScript, CSS (Cascading Style Sheet) and HTML (Hyper Text Markup Language) to present the information to the users and back-end databases, where the data is being kept. This allows the users and customers to contact the companies, fill in the forms, do online shopping and it allows the employees to manage the information, share information, create and edit documents and work on projects [4]. Here is the web applications workflow:

1) The user sends a request over the internet or a local network to the server through either a web browser or an application user interface.
2) The request received will be forwarded to an appropriate web application server.
3) The application server will process the request by manipulating data from database, and the result will be delivered to the user.
4) Web server will receive the processed data from the application server.
5) The web server responds to the user's browser, displaying the requested data on the user's screen.

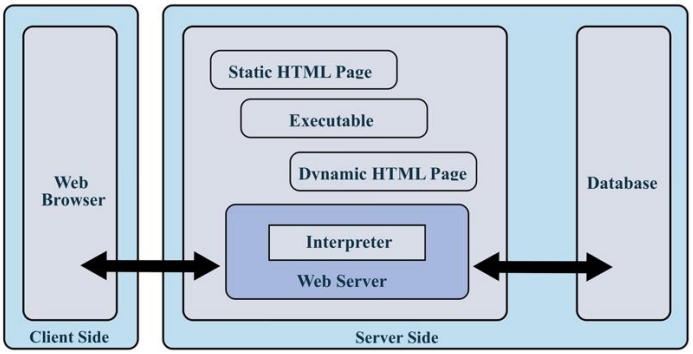

Figure 2. Web Applications Architecture.

3.2 Common Web Application Vulnerabilities and Countermeasures

In cybersecurity, vulnerabilities are the security flaws and weakness which allow the intruder or unauthenticated users to perform an unauthorized action or damage the system or services. An attacker or intruder who wants to exploit the vulnerability of a system should have at least one tool or a technique to establish a connection with the weakness or flaw which is there in the system. There are immense number of vulnerabilities found every day, and it is difficult to remove or eliminate the web application vulnerabilities, because of two main reasons. First, mostly the web application development phase is too rapid which is very short time. Secondly, in most scenarios, the Management Information System engineers (MIS) are involved in the process of developing web applications, most of whom do not have as much training and professional experience at large software firms, like Microsoft, Apple, Google and Facebook [5]. The Open Web Application Security Project (OWASP) is an online community which produces articles, technologies, tools, and documentations for improving application security which are available for free. OWASP Top Ten project: The "Top Ten" project was first published in 2003, and it is regularly updated. The aim of this project is to identify some of the most serious threats which organizations and business are facing and raise awareness about application security [6]. The OWASP top 10-2017 used more than 40 data submissions from application security and by an industry survey which was completed by over 500 individuals, gone through vulnerabilities gathered from hundreds of organizations and more than 100,000 real-world applications [6].

3.2.1 SQL Injections

According to web application architecture, web applications are using databases at the backend, to which web pages are connected in order to store and fetch the user's data to database and display it to the browser or user interface The intruders are trying to exploit database layer vulnerabilities using SQL-Injection which will help to understand the schema of the database which is the most important part of a database, that will allow them to add or include data within the schema of the database. Mostly this attack happens because of lack of a proper checking and filtering mechanism for the users input in the client slide before the data is being input to the database system [7]. In SQL-Injection, the intruder sends unreliable data (malicious code) as a part of SQL Command or query to an interpreter to exploit vulnerabilities in a web application's database. Malicious code executes that type of queries which can give access to an attacker to receive sensitive data from databases. Using the INSERT, DELETE, and ALTER queries, the attacker can modify and change the data available in databases, which affects the integrity of data in database. Consider the following URL which is requested from a Library web application which displays books in different sections.

```
https://abc.com/section?sec=Science
```

An SQL query which will retrieve details of the desired section books from database will be generated.

```
SELECT      * FROM      books WHERE cat = 'Science' AND active = 1
```

It will return all books which belong to the science section and not allowed to be displayed to the users. The active=1 provides a restriction which will hide the books which are not set to be displayed to the users. For the hidden books the value is released = 0. Consider that the application is vulnerable to SQL Injection attack, so attack can be implemented like:

```
https://testing.com/section?sec=Science'--
```

Query generated from the above request will be like:

```
SELECT * FROM books WHERE section = 'Science'--' AND active = 1
```

The double dash (--) sign will make the rest of query as a comment which will not be interpreted by the server, so it eliminates the impact of AND active=1, which will display all books which should be displayed, or which should not be displayed to the users. To avoid SQL-Injection attacks on web applications, the following measures should be implemented:

1) Isolate user input from command structure with the help of parameterizing SQL queries.
2) Periodically update and patch the system.
3) Use of safe APIs for web applications is highly recommended.
4) Deploy appropriate privileges in your system.
5) There should be a proper input validation system to restrict special characters and enforce proper data types.

3.2.3 Broken Authentication and Session Management

The process of verifying an entity or web app which claims to be legitimate is called authentication. Accordingly, the term broken means: inadequate password policies, infinite logon attempts, information leakage or failed logins and insecure password recovery procedures which results in passing the authentication, receive complete control of accounts, account theft and access to sensitive data and damage to data. Authentication and session management is also one of the important parts of web application security, for which the developer should implement proper security practices to secure it. Flaws and bugs in this part can cause many failures in protecting users' credentials and session tokens which leads to serious damage like accessing users accounts or administrator accounts, privacy violations and alteration of credentials in database [3].

In web applications the sessions are established to keep track of requests which are received from users' side. Accordingly, HTTP does not have the above-mentioned feature, thus, the developers are responsible for creating them. The session tokens which are assigned to the users should be secured and protected properly, because the attacker can hijack an active session anytime and undertake the identity of a user. Through this vulnerability the attacker will be able to compromise passwords, keys, or session tokens assigned by the system by which it will assume other user's identities [6]. Mostly all web applications environments, web servers, and application servers are vulnerable to this vulnerability. Assume that Application session timeouts are not set properly in an application. A user is properly authenticated and logged into a public computer system to have access to an application. At the end, instead of clicking the "logout" button to properly logout of the system and end the session. The user just clicks the close button of the browser. A few moments later an attacker can access the same application page using the same browser with that user's authentication. Because the session was still active and not destroyed. The following rules should be focused on and enforced by the developer for providing higher security for applications.

1) Proper mechanism for enforcing the strength, length and complexity of passwords.
2) Proper mechanism of hashing or encrypting the storage of the passwords.
3) Proper mechanism and rules for controlling the changing of the passwords
4) Proper mechanism for protecting the active session ID.
5) Developer should implement a low secure password checking mechanism in web applications.
6) Do not set the default or commonly used username and passwords, especially for admin accounts.
7) A secure built-in session management policy or session manager should be enforced and implemented in server side by the developer, in which uniquely random session IDs are securely created, assigned and maintained only for legitimate users after successful login. And, those IDs should be invalidated and destroyed after the logout, timeouts and being idle for some time. Accordingly, the generated IDs should not be included in the URL of the application.
8) The failed login attempts should be limited and there should be a system to log all failures and alert the administrator when attacks are detected.

3.2.4 Exposure to Sensitive Data

This is one of the widespread attacks with great effect on web applications security. In vulnerability, the attacker is trying to access and receive clear text and keys from the server or perform a man in the middle

attack when the data is being transferred from client's browser to server. Instead of attacking the encrypted or crypto data directly. "773 million Users credentials leakage from different attacks are reported in early 2019, which is really a big security misdeed" [8]. If the sensitive data is not encrypted, weak key generation and management, weak algorithms and protocols for data which is in transit from server side to client side are employed over data are the most common type of flaws. Assume that a web application is using an automatic database encryption mechanism for encrypting credit card numbers in database. However, this means it also decrypts that data automatically when it is being retrieved, this allows a SQL injection flaw to retrieve credit card numbers in clear text [6]. The following steps can be taken to prevent such a scenario.

1) Classifying the data that is processed, stored, or transmitted by an application to categorize the sensitive data.
2) Applying the controls according to classification.
3) Encrypt sensitivity of data at rest.
4) Use only widely accepted implementations of protocols available, crypto algorithms and appropriate key management.
5) Encrypting all data which are in transit from server side to client using TLS (which is highly secure protocol), enforces encryption using directives like HTTP Strict Transport Security (HSTS).
6) Always ensure data integrity and authenticity.
7) Store passwords hashed and salted.

3.2.5 XML – External Entities (XXE)

With this vulnerability the attacker will be able to interfere with application XML data processing. The attacker can exploit vulnerable XML processors by uploading XML or including hostile content in an XML document, exploiting vulnerable code. Mostly it enables the attacker to view and interact with files available on the server filesystem and communicate with backend or any other external system with which the application itself is having access to it. in certain cases, the attacker will be able to deteriorate an XML - External Entity attack to compromise the targeted system or other infrastructures available in the backend, scan internal systems, executing remote request given by the server, perform a denial-of-service attack by leveraging the XXE vulnerability to perform (SSRF) or other types of attacks [9]. There are different types of applications which provide file upload services to the server. Some of those files use XML or may contain XML as their subcomponents. For Example, DOCX for office document formats and SVG for images are XML-based formats. Assume that a web application has the feature that the users can upload images which are being processed and validate on server after they are uploaded. As the SVG format, which is a known type of format for images uses XML, this enables the attacker to submit the malicious SVG image and will be able to get access to the hidden environment to exploit XXE vulnerabilities. The following are some of the ways to avoid such situations.

1) Mostly application's XML parsing library are supporting many dangerous and risky features of XML which does not need to use, therefore the disabling those features will help to prevent XXL attacks effectively.
2) Verify that XML or XSL file upload functionality validates incoming XML using XSD validation or similar [6].
3) Implement whitelisting mechanism for input validations in server-side and filtering to prevent hostile data within XML documents, headers, or nodes.

3.2.6 Broken Access Control

Access control is the procedure that defines how the applications grant access to contents and functions, which is also called Authorization. For developers it is a challenging task to implement a reliable access control system and policy for their applications. This weakness is mostly common due to lack of effective functional testing by application developers or lack of automated detection. An effective method of detecting missing and ineffective access control is manual testing, which also includes HTTP methods as well. Access Control enforces policies for users, such that the users should not be able to act outside their given permissions domain. Mostly failure in the system causes unauthorized access to information, modification and also destruction of data or performing and executing some actions which are outside the limit and permission of the user. By passing access control checks by modifying URL and simply using a custom API attack tool are the common access control vulnerabilities [10]. If a non-admin or an unauthenticated user can access any one of these pages, it is considered as a flaw. To prevent users can follow

1) Log and alert access control failures to admins when appropriate.
2) Use HTTPS instead of HTTP
3) By default, denying access to functionality.
4) By using Access control lists and role-based authentication mechanisms.

3.2.7 Security Misconfiguration

This vulnerability occurs when the developers are not able to implement all security controls for a server or a web application or make errors while implementing those security controls and polices. Hence, the organization is thinking that they are having a safe and secure environment, but it has dangerous mistakes and gaps which can lead to different vulnerabilities and security risks. Attackers will often benefit of unpatched flaws or by accessing default account, unprotected files or unused pages to get unauthorized access to the system. Any kind of lope hole or security weakness can be exploited by an attacker, which can lead to unauthorized access to the system. For having a highly secure web application, it is required to secure th*et al*ready defined and deployed configurations for our web applications, web servers and other entities related to our web application [11]. As the application server admin console is automatically installed in the server and not removed. So, when the attacker discovers the standard admin pages are on your server, logs in with default passwords, and takes over [11]. The following can be some measures to prevent it.

1) For having a high secure system, the organizations should have their own defined policies through which security rules are created.
2) Installing, updating and regular testing of security patches should be done.
3) The effectiveness of deployed configurations should be verified in almost all of the environments using an automated process.
4) Considering roles and permissions like disabling all default accounts or changing passwords in a specific interval of time, disabling administrator accounts.

3.2.8 Cross-Site Scripting (XSS)

This type of attack happens when a web application is used to send malicious code that is generally known as browser-side script, to a different end user. Through this attack, the attacker is injecting malicious scripts into the website code. Generally, such an attack is the result of having improper input validation in the user side. Therefore, this gives the attacker the ability to insert malicious scripts into the application code and will be able to get access to the application. These vulnerabilities will be used to access confidential data, steal identities, perform session hijacking, denial of services attack or bypass the restrictions deployed on web application. XSS is one of the most dominant issues in Top 10 list of OWASP, and is found in around two thirds of all web applications [6]. The following are the types of attacks.

1) Stored/Persistent XSS
   Here the malicious contents which are called payload, mostly Java Script codes, are injected into a targeted application which will be permanently stored or persisted into the application, like in the database of application [12].
2) Reflected XSS
   In this case, the attacker's payload will be injected as a request sent to the server. It will be reflected as the HTTP response will include the payload from the HTTP request. Attacks are delivered via other routes like malicious links or phishing emails. When the user clicks on link or submits data, the browser then executes the code. These attacks have mostly happened through social networks [12].
3) DOM-based XSS
   This is an advanced type of XSS attack. It is mostly a client-side attack in which the infected payload will never be sent to the server. As the application's client-side scripts writes users data to the Document Object Model (DOM), and that data is subsequently read from DOM by application and will be outputted to browser, the attacker can inject payload which will be stored in DOM and executed during read phase from DOM [12].

Assume that an online shopping application, in which users can embed the HTML Tags in comment section which will become the permanent component of the page, and that will cause the browser to parse them along with the rest of code whenever the page is accessed. For example, the attacks comment for an item looks like this:

```
Good product, with good look <script src=http://testing.com/mycode.js> </script>
```

So, anytime that the page is accessed the tag in comment section will be parsed and it will activate the JavaScript file from attacker's website, which is able to steal user session cookies. The attacker could access the user's account and personal information with the help of stolen session cookies. The following are some of the preventive measures.

1) Using Web Application Firewall is a better protection against those attacks.

2) Avoid untrusted HTTP request data which are based on the context in the HTML output.
3) The application code should never output data received as input directly to the browser without checking it for malicious code.
4) Use modern frameworks for application development, most modern frameworks will escape dynamic content by default.

3.2.9 Insecure Deserialization

In web applications the concept takes an object and transfers it to byte stream so that it can be in a proper format to traverse in HTTP network or stored in a database. The reason to use serialization is to save or persist in the state of object, so whenever messages are sent across the network the state of persistence will be there. Turning back the streams of byte to the same object is called deserialization. Many programming languages utilize serialization and deserialization. When the untrusted user input is taken without a proper validation and that input is deserialized from byte stream back into the object, an attacker can take advantage of that and can insert untrusted input and the process of deserialization can lead to remote code execution attacks, which is one of the most sever types of attacks possible [6]. For example PHP forum uses PHP object serialization to save a "super" cookie, which contain the user's login details, role, password hash, and other state. Assume that a forum created using PHP is using PHP object serialization to save certain user's information such as ID, password has, privilege and other related data:

```
a:4:{i:1;i:678;i:1;s:7:"Ali";i:3:"user";
i:3;s:32:"b6a8b3bea87fe0e05033f8f3c99bc9 60";}
```

Here the attacker will change the serialized object to gain Admin privileges.

```
a:4:{i:1;i:0;i:1;s:5:"Ahmad";i:3:"admin";i:3;s:32:"b6a8b3bea87fe0e05033f8f3c99bc
9 60";}
```

The following are some of the preventions.

1) Do not accept untrusted users' input
2) Validate users input properly
3) Network connections with containers or server which deserialize, should be monitored or restricted for both incoming and outgoing connection.
4) Enforcing strict type constraints during deserialization
5) Monitoring deserialization and alerting if a user deserializes constantly.

3.2.10 Using Component with Known Vulnerabilities

It is easy to find already available exploit for huge types of known vulnerabilities, there are some other vulnerabilities which require efforts to develop a custom exploit. In any web application some vulnerable components like working libraries can be exploited with automated tools [6]. Virtually every application has these issues because most development teams don't focus on ensuring their components or libraries are up to date. In most cases, the developers don't know the component they are using or the versions they are using and component dependencies. If the developer doesn't know the versions of all components directly used or nested dependencies used in client and server-sides, the developed application will be highly vulnerable. Accordingly, if the software is vulnerable, unsupported or out of date Operating system, Web/Application server, DBMS, all components and runtime environments and all libraries are used in application development, the developed application will be highly vulnerable to attacks. As the web application and its components have the same privileges, therefore any flaw or vulnerability in any component of an application can have a bad effect on the application itself. Those flaws can be coding errors during development process or already installed backdoors in application [6]. The following can be prevention.

1) Availability of patch management system for application
2) Remove unused:
    a. Dependencies.
    b. Unnecessary features components.
    c. Files, and documentation.
    d. Use components only from official sources.

3.2.11 Insufficient Logging and Monitoring

This vulnerability comes into an action that events which are highly critical in security aspect are not recorded or may be omitted important information about an event when logging, or the system is not logging the current events. The lack of such a system makes it harder to detect and handle the attack which is happening now [13]. Assume that an attacker is trying to download a huge amount of data from the server, which indicates an unusual huge amount of outgoing traffic from the server. With the help of proper monitoring system, such data extraction can be detected and can be prevented such as:

1) Ensure all login, access control failures, and server- side input validation failures can be logged properly which will help to identify suspicious or malicious accounts [6].
2) To make sure that all logs are created in a format which can be easily analyzed by central log management solution.
3) To establish effective monitoring and alerting plans for applications.

3.3 Open-Source Vulnerability Assessment Tools

Vulnerabilities are the key source for malicious activities like cracking web sites and systems. Vulnerability assessment enables you to recognize, categorize and characterize the security flaws and holes in an application, computer systems, or in networks. The developers and system administrators can take advantage of vulnerability assessment methods and tools to make their applications and system safe and secure. Vulnerability scanners and assessment tools are making the security auditing task automated and can play a great role in finding flaws and major security risk in websites and systems.

3.3.1 OWASP Zed Attack Proxy (ZAP)

Zed Attack Proxy, also known as ZAP, is among the world's most popular security tools, which is free and open-source, that is developed by AWASP and is actively maintained by hundreds of international volunteers. ZAP is available for Windows, Unix/Linux and Mac platforms. ZAP is a very simple and easy to use tool which can be used to find wide range of web applications vulnerabilities. This tool will enable you to automatically detect security vulnerabilities of any web application, even if you are in developing and testing stage. Consequently, this tool can also be used to perfume manual security tests over your web application [6]. OWASP Zed Attack Proxy provides you with the ability to detect OWASP top 10 security threats that your website/application might face. Some key features of OWASP ZAP

1) Automatic Scanner
2) Traditional but powerful spiders
3) Fuzzer
4) Web Socket Support
5) Dynamic SSL certificates
6) Plug-n-hack support
7) Smartcard and Client Digital Certificates support
8) Authentication support
9) Intercepting Proxy

3.3.2 W3AF

W3AF is an open source, web application attack and audit framework written in Python. This is a powerful tool which can detect most well-known and most common vulnerabilities in a web application. This tool is equipped with more than 130 plugins which will make it easier to detect the flaws and vulnerabilities. [14]. W3AF is consist of two main parts, the core and plug-in. the core coordinates and manages the process and provides features which are consumed by plugin-ins, which find vulnerabilities and exploit them. It is available for Linux, Mac and windows platforms. Some key features of W3AF:

1) To identify and exploit vulnerabilities, W3AF can implement web and proxy servers
2) To provide Proxy support
3) To use Fuzzing engine
4) Provides Knowledge base
5) Support File upload using multipart
6) DNS cache
7) HTTP Basic and Digest authentication
8) Cookie handling
9) HTTP response cache

10) Support different logging methods like:
    a. Console
    b. Sent by email
    c. Text, CSV, HTML and XML files.

3.3.3 Arachni

Arachni is an open source, modular and high-performance penetration deployment environment which can detect huge number of well-known vulnerabilities. It is free, with its source code public and available for review. Arachni is multi-platform, supporting all major operating systems (All common operating systems) and distributed via portable packages which allow for instant deployment [15]. Some key features of Arachni are as follows:

1) Capable of learning from HTTP response to present better results
2) REST API
3) Support highly complicated web applications which make heavy use of technologies such as:
    a. JavaScript
    b. HTML5
    c. DOM manipulation
    d. AJAX.

3.3.4 Skip Fish

Skip fish is highly quality, high speed, ready to use and an active web security inspection tool developed by google. This tool can formulate an interactive sitemap for a website with crawl recursive in nature and probes based on dictionary [16].

3.3.5 Wapiti

Wapiti is an open-source command-line tool used for auditing web applications security. It scans web pages and injects data to check if a script is vulnerable or not. This tool supports both GET and POST HTTP attacks and also able to detect multiple number of vulnerabilities available in cyber space. This tool is applicable for Windows, Linux and Mac [17]. It can detect the following well-known vulnerabilities:

1) Backup files disclosure
2) CRLF Injection
3) SQL Injection
4) XSS attack
5) File Inclusion
6) File Disclosure
7) Command execution detection
8) Weak .htaccess configuration

3.3.6 Vega

Vega is a free open source, automated web vulnerability scanner and testing platform written in Java which offers a GUI based environment. The tool can be extended with the help of powerful API written in JavaScript. [18] This tool is available for Linux, Mac and Windows platforms.

Table 1. Comparative Analysis of Open-Source Vulnerability Assessment Tools

Name	Language	License	Platform	Authentication	Active	API	Plug-in	Scanner	Spider	Fuzzer	AJAX	CLI	GUI	Learning
ZAP	Java	Apache	Windows, Mac, Linux	√	√	√	√	√	√	√	√	√	√	×
W3AF	Python	GPLv2	Windows, Mac, Linux	×	√	×	√	√	√	√	×	√	√	×
Arachni	Ruby	Apache	Windows, Mac, Linux	√	√	√	√	√	√	√	√	√	√	×
Skipfish	C	GPLv2	Linux, FreeBSD, MacOS	×	×	×	×	√	√	√	×	√	×	×
Wapiti	Python	GPLv3	Windows, Mac, Linux	√	√	×	×	√	√	×	×	√	×	×
Vega	Java	Apache	Windows, Mac, Linux	√	√	×	×	√	√	√	×	√	√	×

3.4 Discussion

Web applications are a critical asset for businesses, organizations, and government agencies, acting as an efficient, fast, and secure route for users. These applications help collect, analyze, and process user data seamlessly. Web applications (webapps) are mapped with a combination of back-end processes (server-side programming languages including PHP and ASP for managing data) and front-end processes (client-side programming languages such as JavaScript, CSS, and HTML for displaying information) alongside database storage for storing information. Web applications consist of user interactions with application servers, web servers, and user browsers, so it is important to understand the architecture and workflow of these interactions for the application to run securely and reliably.

SQL Injection, Broken Authentication, Sensitive Data Exposure, and Cross-Site Scripting (XSS) attack vectors can pose serious threats to web applications, and are among the most commonly exploited vulnerabilities in information systems. Although each type of vulnerability has its own unique mitigation measures. As a solution, the use of parameterized queries and strict input validation to prevent SQL Injection. So, The Final Top 10 List Is As Follows And Provides Solutions For Each Broken Authentication Session Id Secure Based on Session ID that sets restrictions on NUMERIC Failed attempts Set strong passwords While web application firewalls and use of modern frameworks can significantly reduce the risk of XSS, So, to address these vulnerabilities and combat the vulnerabilities, a structured effort with proper technology support is required to maintain data integrity and confidentiality.

Assessment tools like OWASP ZAP, W3AF, Arachni, Skipfish, Wapiti, Vega are the tools required to assess and fix the vulnerabilities. Both manual and automated security testing are supported with the various features provided by these tools. OWASP ZAP, for example, allows in-depth threat detection with an easy-to-use interface. On the other hand, W3AF has the flexibility with 130+ plugins but does not provide API integration. However, Arachni works well when dealing with complex technologies, like JavaScript and AJAX, which are commonly used in web applications. Each tool has its own advantages and disadvantages, which requires careful selection of tools according to the needs of the organization. Efforts to improve web application security should include measures such as implementing data encryption, system updates and access restrictions, and creating adequate logging systems. Modern testing tools and approaches that combine automated and manual methods to detect different types of vulnerabilities also need to be adopted by developers.

4. Related Work

Alzahrani *et al*. (2017), demonstrated the architecture of web applications and also studied and evaluated a number of security vulnerabilities [4]. According to the study, about 49% of the web applications that have been reviewed have vulnerabilities with high-risk life. Furthermore, the tools that have been used to scan those most common vulnerabilities like cross site scripting, SQL Injection, Information leakage and insufficient transport-layer protection ar*et al*so analyzed. It is also stated that in some cases the designing and development errors cause most of the vulnerabilities in web applications. In addition to that, weak and insufficient administration can cause vulnerabilities like information leakage and insufficient transport-layer protection, so they have elucidated the usage of those tools which help in preventing those vulnerabilities. Consequently, each tool's advantages and disadvantages ar*et al*so discussed. According to this paper, security assessment tools can be used in either browser-side or server-side and it can also be used in both sides.

Patel (2019), conducted a study on the common vulnerabilities, resolution of those vulnerabilities, as well as the author proposed some methodologies and tools used for determining those vulnerabilities of the time which helps organizations to secure their services and mitigate the risk of attacks which can be caused from those vulnerabilities [8]. The proposed tools are both commercial and open source. To make the outcome the result of the proposed tools, they have prioritized and explained with the CVE number, that can be utilized from industry standard references. Nagpure *et al*. (2019) provided an analysis of web application vulnerability assessment along with penetration testing techniques. This study highlights that manual penetration testing accuracy is much effective to perform security assessments as compare to automated penetration testing [7]. It is also mentioned that using manual testing technique most famous web vulnerabilities such as Cross site scripting, clickjacking, SQL Injection, file upload and other weakness are detected in many web applications. Relatively, in terms of time and money saving, automation testing technique can also use to detect some web application vulnerabilities and can also preform automated web application penetration tests. Finally, they have proposed that for vulnerability assessment, companies should have a combined vulnerability assessment method of both manual and automated testing techniques which will definitely enhance the accuracy in detecting vulnerabilities in web applications.

Dua *et al*. (2017), have studied a number of common attacks like SQL Injection, Cross Site Scripting (XSS), etc and also proposed a tool which is based on XAMPP server for both server and client environments, which will help the security analysts and students to perfume checkup and analysis of web application vulnerabilities [19][20]. The tool which is proposed is mainly focusing on OWASP list of top 10 attacks, which is a good mean for developers to fix weakness and bugs using those queries which are used for accessing the back-end information. Singh *et al*. (2018) discussed the tools where the cybersecurity professionals can ethically perform attacks to learn about vulnerabilities and also proposed a tool with a legitimate structure that avoids different web attacks. The authors stated that SQLI, broken confirmation, session management and XSS are the basic provision strike found on the web [20]. A large number of systems are reviewed against impediments, accordingly, most of the people are utilizing web administration to their reduction on the globe to relieve themselves, which cause them huge costs.

Tyagi *et al*. (2018) studied OWASP WAP and RIPS, two web application vulnerability detection tools which are basically source code analysis tools. They have done an experiment on (DVWA) and a (bWAPP) web applications which are vulnerable and full of bugs, based on the experiment, they found that OWASP WAP provides better results as compared to RIPS [21]. With the help of vulnerable web applications used in the experimental scenario, we can easily detect vulnerability whether they are TRUE positive or FLASE positive. Accordingly, it is stated that the commercial version of RIPS may generate better results, but in some cases open-source tools can only be needed. Al-Sanea *et al*. (2015) presented the result of assessing and testing the security posture of almost 150 websites from different categories like financial, governmental, commercial and academic website of Sudia Arabia with the help of open source vulnerability assessment tools available in the market [22]. Many vulnerabilities with different level are found, the number of affected websites were large. In addition, government websites were more secure as compared to commercial websites.

Huang *et al*. (2017) have discussed many web application vulnerabilities and proposed several countermeasures and pitfalls. With respect to that, VulScan which is a new vulnerability assessment tool that uses penetration testing and combinative evasion techniques in order to discover cross site scripting and injection vulnerabilities in a system is being introduced, which will improve the system security. To evaluate the accuracy introduced system, a number of real web application like OWASP's WebGoat has been selected for testing [23]. Accordingly, the total vulnerabilities detected by the VulScan is compared with the total number of vulnerabilities detected by ZAP. According to the results of the systems, VulScan was able to detect more SQL Injection and also Cross Site Scripting vulnerability than the OWASP's ZAP.

Moniruzzaman *et al*. (2019) studied a technique to detect maximum number of vulnerabilities using source code analysis and penetration testing techniques with minimum efforts. In this method they have evaluated a number of specific web site in Bangladesh against some most common web vulnerabilities and attacks [24]. The result showed that around 64% of the evaluated web applications are having vulnerabilities, especially governmental organizations are in serious risk. Sandhya *et al*. (2017) enlightened the need to utilize penetration testing, and also with the help of Wireshark, they have done penetration testing in order to assess the security of web sites [25]. They have stated that, Wireshark tool helps ethical hackers to underpin the system security in users at the level of authentication only and proved that its better and rapid wayout to deal with vulnerabilities. Pranathi *et al*. (2018) reviewed the customer side answer to mitigate and solve the cross-site scripting attack. They have claimed that, because of poor web surfing background the client system performance is decreased [26]. Therefore, they have provided a client-side arrangement which is use a well-ordered that deal with ensure cross site scripting web applications. The framework which is proposed is based on how to attack a web site through cross site scripting utilizing the contents with the final goal of preventing attacks.

Efendi *et al*. (2019) studied various deception techniques which could be used as a defined mechanism to sense web application attacks and also act as a distraction and protection mechanism to keep the attacker away from the actual protected services [27]. Yadav *et al*. (2018) reviewed several evolving trends and prevention from web attacks. They hav*et al*so discussed about prevention from several vulnerabilities available in web applications by providing suitable data types to input, restriction in the use of web server, http request and restrictions for users to access the files from the root directories [28]. Consequently, overall security preventions for operating systems and mobile applications ar*et al*so discussed. They hav*et al*so proposed that the government beside promoting digitalization should also focus restricting rules and regulations to protected vital information from attacks and frauds.

Gillman *et al*. (2015) surveyed a number of most common web site attacks and also techniques in order to mitigate them. They mentioned that the attack detection and mitigation is increasingly involves frequently performing the processing operation over a large amount of data which is across multiple and different websites with a specific interval of time [29]. It is also stated that the latest modern attacks are using many different methods to infect a system and can continue for several months. They hav*et al*so described the impact and importance of comebacks to lessons learned from those series of attacks which happened in 2012-2013 on Akamai's customers dubbed operation Ababil. Alenezi *et al*. (2016) have tasted different open source web applications against some most common security vulnerabilities. They have done static security vulnerability test in three different categories: (a) Dodgy code Vulnerabilities (b): Malicious Code Vulnerabilities (c): Security Code Vulnerabilities on different web applications available [30]. They have recommended an intelligent development frame work which can provide suggestion in order to have a secure development, can add missing codes and also learn from other expert developers' practices in order to mitigate and solve common security vulnerabilities on web applications. Nababud *et al*. (2018) study focus on penetration test comparison of use of http and https on websites using Wireshark tool. Wireshark is used to perform penetration testing campus webmail individually and tried to fine its vulnerability [31]. They have stated that if the users want to keep data secure, they should use webmail which is encrypted by HTTPS protocols in data communications.

Shrivastava *et al*. (2016) based on study of pervious research and their implementations, studied the detection and prevention methods of all three types of cross site scripting (XSS) vulnerability. They have tested so many web applications and as a result they found that cross site scripting issues are in a higher priority. As in the existing techniques usually the developers are using Java validation and also input filters in order to prevent client-side infected inputs, on the other hand, some more secure web applications uses Application level firewalls in order to filter the user's input, but still they were not able to totally stop the cross site scripting (XSS) attacks [32]. In their assessment they have found that these are not enough to prevent (XSS) with dangerous payloads. They stated that, the use of those existing techniques can only the direct infected inputs from browsers side, but they are not that much strong to prevent the middleware happing attacks. In their system, they have used the JavaScript validation technique for users input, JavaScript signature mechanism in order to identify valid JavaScript, sanitization mechanism to clean HTML text, token assignment mechanism for client-server request during communication, use of HTTPOnly cookies flag, and lastly they hav*et al*so suggested to use strong application level firewall and proper security assessment and scanner tools to detect XSS. It is stated that their proposed system can help the developers in handling and detecting different cross site scripting (XSS) vulnerabilities. They have suggested that hierarchical system because that single level security is not sufficient to prevent XSS vulnerabilities.

Ariful Haque *et al*. (2022), examines typical web application vulnerabilities and the efficacy of preventative measures. This study proposes a paradigm for preventing exploitation through better design practices and emphasises the need to find vulnerabilities early in the development process [33]. Although the study strongly emphasizes the methods for prevention, it does not provide an exhaustive analysis of web application vulnerability assessment tools. To address this gap, this research focusses on the potential of open-source tools and how well they can identify a variety of vulnerabilities.

Analyzing these previous studies critically results in realizing that while significant achievements have been made in web application security, there remain certain gaps, specifically in integration of manual and automated testing techniques, and the comparison of open-source vulnerability scanning tools. This research not only fill these gaps, but build upon existing knowledge by purposing a more comprehensive, hybrid testing process and offering practical insights into the use of open-source tools. Moreover, the related work in this paper is directly linked to the research outlet. In the critical analysis of literature review, we need a more dynamic and adaptable Vulnerability management model as identified by different authors in previous studies that said it forms basis of this research. We identify gaps in current tools and practices to provide a new lens on how we can improve the security of web applications by testing better or using better tools.

Additionally, the related work in this paper has a profound connection to the main area of study. This study is conducted based on the critical review of prior studies, which emphasizes the need for a more flexible and adaptable vulnerability management strategy. It provides new insights on how to strengthen the security of web applications by improving testing procedures and tool selection by analyzing weaknesses in current techniques and tools.

Table 2. Comparative Analysis of Related Studies

Study	Methodology	Findings	Tools used
Alzahrani *et al*. [4]	Studied and evaluated security vulnerabilities in web applications	49% of reviewed web applications have high-risk vulnerabilities	Scanning tools for common vulnerabilities such as cross-site scripting, SQL injection, information leakage, and insufficient transport-layer protection
Patel [8]	Studied common vulnerabilities and proposed methodologies and tools for determining them	Prioritized vulnerabilities and provided CVE numbers	Both commercial and open-source tools
Nagpure *et al*. [7]	Analyzed web application vulnerability assessment and penetration testing techniques	Manual testing is more effective than automated testing for accuracy	Manual and automated testing techniques
Dua *et al*. [19]	Studied common attacks and proposed a tool for analyzing web application vulnerabilities	Proposed a tool based on XAMPP server for analyzing web application vulnerabilities	Focused on OWASP list of top 10 attacks
Singh *et al*. [20]	Discussed ethical hacking and proposed a tool with a legitimate structure that avoids different web attacks	Stated that SQLI, broken confirmation, session management, and XSS are the basic provisions strike found on the web	Ethical hacking tool
Tyagi *et al*. [21]	Studied OWASP WAP and RIPS, two web application vulnerability detection tools	Found that OWASP WAP provides better results than RIPS	OWASP WAP and RIPS
Al-Sanea *et al*. [22]	Assessed and tested the security posture of almost 150 websites from different categories	Found many vulnerabilities with different levels and that government websites were more secure than commercial websites	Open-source vulnerability assessment tools: W3af and SkipFish
Huang *et al*. [23]	Discussed web application vulnerabilities and proposed	Introduced VulScan, a new vulnerability assessment tool	VulScan and OWASP's ZAP

	countermeasures and pitfalls		
Moniruzzaman *et al*. [24]	Studied a technique to detect vulnerabilities using source code analysis and penetration testing techniques	Around 64% of evaluated web applications have vulnerabilities	Source code analysis and penetration testing
Sandhya *et al*. [25]	Highlighted the need to utilize penetration testing and used Wireshark for penetration testing	Wireshark tool helps ethical hackers to underpin system security	Wireshark
Alazmi and Leon [33]	Reviewed the effectiveness of frequently used web application vulnerability scanners.	The study found that out of 12 study, most of the evaluations tested only two of the OWASP top ten vulnerability types. And only one work evaluated six of OWASP top ten vulnerability types	Open source and closed source.
Shahid *et al*. [34]	A number of application vulnerability assessment tools were surveyed and evaluated based on their capabilities, strength and weaknesses.	The paper found up that ZAP has a higher vulnerability detection as compared to Acunetix and NetSparker.	Open source and closed source, specifically; OWASP-ZAP, Acunetix, and NetSparker

Previous research has revealed achievements in understanding and addressing web application vulnerabilities, including identifying common vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and authentication vulnerabilities. Several tools and techniques, both manual and automated, have been tried to assess their ability to find and fix these vulnerabilities. However, there is still a gap in the integration between manual and automated testing methods, and there is not enough research on the comparison of open-source vulnerability scanning tools. By proposing a more flexible Test Administration Hybridization and Adaptation (tHAd) method than traditional assessment, it will result in more secure web applications by introducing a better testing method and smarter tool usage. Therefore, by making this study a strong foundation for developing a more flexible and effective vulnerability management strategy to date.

5. Conclusion and Future Work

Making a web application secure and finding the vulnerabilities available in a web application is a tough process and needs proper planning and care. Threats for integrity and confidentiality of sensitive data in web applications are increasing day by day. According to OWASP, SQL Injections, Broken Authentication and Sensitive data exposure are the most frequent attacks on web applications. As the attacks are getting complex day by day, it is an important factor that companies, developers and administrators educate themselves on the serious risk that they are facing. The study makes a significant contribution to web application security by evaluating high-risk vulnerabilities, particularly those identified in the OWASP Top 10. It also provides practical countermeasures and compares them with widely used open-source vulnerability assessment tools like OWASP ZAP and W3AF to assess their real-world effectiveness. Using parameterized queries, securing session management, and routinely deploying software patches are important suggestions. In addition, the study recommends using automated techniques in conjunction with manual testing to cover a wider range of potential vulnerabilities, particularly in more complex systems.

Most of the vulnerability are preventable by securing the respective web server like keeping all libraries, plug-ins, database software, frameworks updated with latest security patches. Additionally, the developers should also focus on the possible loopholes and handling methods in development time. The study makes it abundantly clear for practitioners that to safeguard web applications, multi-layered security techniques must be put in place together with regular vulnerability assessments. As Alzahrani *et al*. [5] found out that "some vulnerabilities, such as Cross- Site Scripting, and SQL injection have occurred due to design errors, while information leakage and insufficient transport layer protection are often caused by insufficient administration".

They have recommended the usage of tools to prevent these vulnerabilities. As the attackers are trying to find new methods to bypass the security of web applications, every day new vulnerabilities are added to the list. Therefore, the administrators should be aware of security testing skills and techniques which may be required during the system lifecycle. The study emphasizes that multi-layered security techniques, combined with regular vulnerability assessments, are essential for safeguarding web applications.

In order to choose an appropriate tool which is affordable to most of the organizations, in this document a number of top open-source vulnerability scanners which are developed using highly secure and advanced technologies are discussed, to detect maximum number of vulnerabilities with minim cost and less effort. As the modern web applications are rapidly growing, the traditional vulnerability assessment methods might not be sufficient. Therefore, as a future work researcher, I would like to explore and evaluate the potential use of different techniques in machine learning (ML) which can provide effective mechanisms to identify and classify different web application vulnerabilities based on their severity.

References

[1] ASM Technologies Ltd. (2017). CEBIT - Introduction to Cyber Security. ASM Technologies Ltd.

[2] Curphey, M., & Arawo, R. (2006). Web application security assessment tools. *IEEE Security & Privacy*, *4*(4), 32-41. https://doi.org/10.1109/MSP.2006.108.

[3] Positive Technologies. (2019). Web application vulnerabilities: Statistics for 2018. Positive Technologies.

[4] Alzahrani, A., Alqazzaz, A., Zhu, Y., Fu, H., & Almashfi, N. (2017, May). Web application security tools analysis. In *2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)* (pp. 237-242). IEEE. https://doi.org/10.1109/BigDataSecurity.2017.47.

[5] Revathy, P., & Mukesh, R. (2017, December). Analysis of big data security practices. In *2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)* (pp. 264-267). IEEE. https://doi.org/10.1109/ICATCCT.2017.8389145.

[6] Open Web Application Security Project (OWASP). (2019). OWASP Top Ten Project. Retrieved June 2, 2019, from https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[7] Nagpure, S., & Kurkure, S. (2017, August). Vulnerability assessment and penetration testing of web application. In *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICCUBEA.2017.8463920.

[8] Patel, K. (2019, April). A survey on vulnerability assessment & penetration testing for secure communication. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)* (pp. 320-325). IEEE. https://doi.org/10.1109/ICOEI.2019.8862767.

[9] Portswigger. (2019). XML external entity (XXE) injection. Retrieved June 2019, from https://portswigger.net/web-security/xxe

[10] Blazquez, D. (2019). Broken access control. Retrieved November 20, 2019, from https://hdivsecurity.com/owasp-broken-access-control

[11] Blazquez, D. (2019). Security misconfiguration. Retrieved November 20, 2019, from https://hdivsecurity.com/owasp-security-misconfiguration

[12] Acunetix. (2019). Types of XSS: Stored XSS, reflected XSS, and DOM-based XSS. Retrieved November 2019, from https://www.acunetix.com/websitesecurity/xss/

[13] Hack2Secure. (2018). Insufficient logging and monitoring: A brief walk through. Retrieved January 27, 2018, from https://www.hack2secure.com

[14] Suteva, N., Zlatkovski, D. D., & Mileva, A. (2013). Evaluation and testing of several free/open-source web vulnerability scanners. *The 10th Conference for Informatics and Information Technology (CIIT 2013)*, Macedonia.

[15] Sarosys LLC. (2017). Arachni scanner. Retrieved December 19, 2019, from https://www.arachni-scanner.com

[16] G., S. (2018). Skipfish – Web application security scanner for XSS, SQL injection, shell injection. Retrieved December 22, 2019, from https://gbhackers.com/skipfish-web-application-security-scanner/

[17] Surribas, N. (2019). The web-application vulnerability scanner. Retrieved September 4, 2019, from https://wapiti.sourceforge.io

[18] Mehra, D. (2018). How to start with Vega: The web security scanner? Retrieved February 5, 2018, from https://blog.knoldus.com/start-vega-web-security-scanner/

[19] Dua, M., & Singh, H. (2017, October). Detection & prevention of website vulnerabilities: Current scenario and future trends. In *2017 2nd International Conference on Communication and Electronics Systems (ICCES)* (pp. 429-435). IEEE. https://doi.org/10.1109/CESYS.2017.8321315.

[20] Singh, H., & Dua, M. (2018, July). Website attacks: Challenges and preventive methodologies. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)* (pp. 381-387). IEEE. https://doi.org/10.1109/ICIRCA.2018.8597259.

[21] Tyagi, S., & Kumar, K. (2018, December). Evaluation of static web vulnerability analysis tools. In *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 1-6). IEEE. https://doi.org/10.1109/PDGC.2018.8745996.

[22] Al-Sanea, M. S., & Al-Daraiseh, A. A. (2015, November). Security evaluation of Saudi Arabia's websites using open source tools. In *2015 First International Conference on Anti-Cybercrime (ICACC)* (pp. 1-5). IEEE. https://doi.org/10.1109/Anti-Cybercrime.2015.7351928.

[23] Huang, H. C., Zhang, Z. K., Cheng, H. W., & Shieh, S. W. (2017). Web application security: Threats, countermeasures, and pitfalls. *Computer*, *50*(6), 81-85. https://doi.org/10.1109/MC.2017.183

[24] Moniruzzaman, M., Chowdhury, F., & Ferdous, M. S. (2019, February). Measuring vulnerabilities of bangladeshi websites. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)* (pp. 1-7). IEEE. https://doi.org/10.1109/ECACE.2019.8679426.

[25] Sandhya, S., Purkayastha, S., Joshua, E., & Deep, A. (2017, January). Assessment of website security by penetration testing using Wireshark. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)* (pp. 1-4). IEEE. https://doi.org/10.1109/ICACCS.2017.8014711.

[26] Pranathi, K., Kranthi, S., Srisaila, A., & Madhavilatha, P. (2018, March). Attacks on web application caused by cross site scripting. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 1754-1759). IEEE. https://doi.org/10.1109/ICECA.2018.8474765.

[27] Efendi, A. M., Ibrahim, Z., Zawawi, M. A., Rahim, F. A., Pahri, N. M., & Ismail, A. (2019, May). A survey on deception techniques for securing web application. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)* (pp. 328-331). IEEE. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2019.00066.

[28] Yadav, D., Gupta, D., Singh, D., Kumar, D., & Sharma, U. (2018, December). Vulnerabilities and security of web applications. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)* (pp. 1-5). IEEE. https://doi.org/10.1109/CCAA.2018.8777558.

[29] Gillman, D., Lin, Y., Maggs, B., & Sitaraman, R. K. (2015). Protecting websites from attack with secure delivery networks. *Computer*, *48*(4), 26-34. https://doi.org/10.1109/MC.2015.116

[30] Alenezi, M., & Javed, Y. (2016, September). Open source web application security: A static analysis approach. In *2016 International Conference on Engineering & MIS (ICEMIS)* (pp. 1-5). IEEE. https://doi.org/10.1109/ICEMIS.2016.7745369.

[31] Navabud, P., & Chen, C. L. (2018, July). Analyzing the web mail using wireshark. In *2018 14th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD)* (pp. 1237-1239). IEEE. https://doi.org/10.1109/FSKD.2018.8686871.

[32] Shrivastava, A., Choudhary, S., & Kumar, A. (2016, October). XSS vulnerability assessment and prevention in web application. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)* (pp. 850-853). IEEE. https://doi.org/10.1109/NGCT.2016.7877529.

[33] Alazmi, S., & De Leon, D. C. (2022). A systematic literature review on the characteristics and effectiveness of web application vulnerability scanners. *IEEE Access*, *10*, 33200-33219. https://doi.org/10.1109/ACCESS.2022.3163023

[34] Shahid, J., Hameed, M. K., Javed, I. T., Qureshi, K. N., Ali, M., & Crespi, N. (2022). A comparative study of web application security parameters: Current trends and future directions. *Applied Sciences*, *12*(8), 4077. https://doi.org/10.3390/app12084077.

[35] Haque, A., et al. (2023). Web applications vulnerability analysis and prevention. Retrieved from https://www.researchgate.net/publication/381303883_WEB_Applications_Vulnerability_Analysis_and_prevention.

