A Survey on Detection and Prevention of Web Vulnerabilities

Muhammad Noman[1]
Department of Computer Science
Bahria University Karachi Campus

Muhammad Iqbal[2]
Department of Computer Science
Bahria University Karachi Campus
& School of Information Sciences & Technology
Southwest Jiaotong University, Chengdu, China

Amir Manzoor[3]
Department of Management Science
Bahria University Karachi Campus

***Abstract*—The Internet provides a vast range of benefits to society and empowers the users in a variety of ways to use web applications. Simply, the internet has become the most transformative and fast-growing technology ever built, but it also brings new security challenges to web services in internet applications because of the scattered and open nature of the internet. A simple vulnerability in the program code could favor/benefit an attacker to obtain unauthorized access and perform adversary actions. Hence, the security of web applications from a hacking attempt is of paramount importance. This paper focuses on a literature survey recapitulating security solutions and major vulnerabilities to promote further research by systemizing the existing methods, on a bigger horizon. The data is collected from an absolute of 86 primary studies that are taken from well-known digital libraries. Different methods comprising secure programming, static, Dynamic, Hybrid analysis, and machine learning classify the data from articles. The quantity of references or the significance of a developing strategy is kept in account while selecting articles. Overall, our survey suggests that there is no way to alleviate all the web vulnerabilities therefore more studies is desirable in the area of web information security. All methods' complexity is addressed and some recommendations regarding when to use the application of given methods are provided. Finally, we typify the experience gained and examine future research openings in web application security.**

Keywords*—Web security survey; web vulnerabilities; detection and prevention techniques

I. Introduction

Web-based applications are the best network-based solution to provide standard facilities. It has revolutionized the way standard facilities can be offered. Developing modern web applications is now the best mode. These applications are developed with the combination of a client and server-side development. The server-side portion uses different programming languages (.Net, PHP, Python, and Ruby) and front-end is a client-side portion, which runs on the user's web browser with different programming languages such as JavaScript and CSS/HTML. These two portions are frequently interconnected through HTTP or HTTPS protocol through asynchronous XML (AJAX) and JavaScript [1]. Fig. 1 describes the architecture of server-side and client-side of the website.

The availability of web applications has made them an integral part of everyone's daily life. This is because of their primarily free and internet-accessible availability and ability to handle sensitive data such as banking and payment for e-commerce. Because of their increased popularity, web applications are also the primary focus of hackers [2]. The popular uses of web applications, such as web blogs, social media, banking, and e-commerce, and their vulnerabilities are the focus of hackers to hack web applications with vulnerabilities. The weakness, bug, and loophole in the web application that can be exploited by hacker are called vulnerability [3].

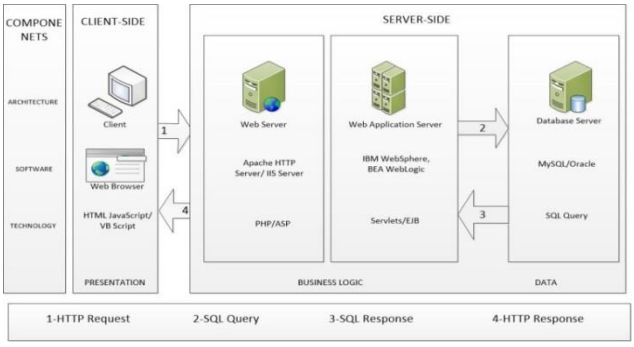

Fig. 1. Overview of Web Architecture.

The most critical vulnerabilities are cross-site scripting (XSS), SQL injections (SQLI), and cross-site request forgery (CSRF) that are listed in the top 10 web vulnerabilities by OWASP. The hackers can use the information of these vulnerabilities to compromise the website. Therefore, website requires security countermeasures to secure web application. A variety of techniques is being used around the globe to overcome these vulnerabilities and these techniques assist to identify the website vulnerabilities. There is a strong need for frequent testing to prevent and minimize web vulnerabilities. However, it requires that the tester have adequate experience

given that the testing procedure itself is an extended manual process [4]. Therefore, some other approaches need to be explored to prevent vulnerabilities.

The methodology to cope with security issues is to find out the bugs before discovery and exploitation by hackers. One of the keen approaches is the use of a white-box Technique. It consists of an analysis of website source code. However, there is a problem with massive false positives and the web application's source code may not be available. There is another procedure called black-box testing to help analyzers and overcome the method of white box testing. The strategy is to examine the vulnerabilities of the application by giving some input for specific vulnerability output. Many researchers have effectively analyzed black box scanner in vulnerability detection. Furthermore, they find out its constraints by repeatedly testing numerous black-box scanners against a wide range of vulnerable applications. A lot of work in this direction is focused on fuzzing. It deals with testing (semi)-random values [5]. Another important method to prevent web vulnerabilities is data mining and machine learning. These learning methods with a variety of web applications are considered a unique approach. However, it can also be used in source code to identify vulnerabilities [6].

We tend to survey the last ten years of existing web vulnerabilities in this study. The goal is to systematize the present methods into a vast picture that supports future research. We categorized the review of web vulnerability detection methods using hybrid analysis, dynamic analysis, static analysis, data mining, and techniques of machine learning. Initially, with traditional approaches, we outline the web vulnerability discovery and analysis difficulty. We also briefly explain the web vulnerabilities and their types. Hybrid analysis, dynamic analysis, Static analysis, and machine learning are different approaches to prevent web vulnerabilities. After that, we discuss each method in detail with the definition, prevention, advantages, and challenges.

We structured the paper as follows. We initially explained the working of a web application with distinctive qualities. Section II describes the classification of web vulnerabilities along with the methods to secure it. Then, discuss and categorize each existing countermeasures in Section III. Section IV, we arrange the analysis method to detect vulnerabilities with the table and discussion. At that point, in Section V, The connected work is debated. In Section VI conclusion of this survey.

II. Background: Web Vulnerabilities Analysis and Methods

We describe the classification of web vulnerabilities and methods to secure web applications. A term vulnerability is a defect referred to error and bug that arises due to defects in the coding of a web application. This result in a severe type of damage to web application upon exploitation [4, 7]. Table I present five types of web vulnerabilities and we categorized these vulnerabilities into three main sections such as improper authentication, improper input validation, and improper session management and. It has been further divided into four web vulnerability categories: Query manipulation, Client-side, Path injection, and session management.

The main issue in security for web applications may be an inappropriate validation of user input. This Input enters into a web application via entry points ($_GET in the PHP language) and hackers can utilize web vulnerability through MySQL query. The major number of attacks occur with the combination of simple input and metadata like ‘And, OR’. Therefore these websites can frequently ensure the input of the user and validate the path and entry points [8].

A. Improper Input Validation

The web application is must validate or sanitize user input properly before its utilization in the web servers. Usually, web developers exercise sanitizing practices (i.e., sanitizers) for the transformation of inputs by the user into trusted data through filtration. For example, an HTML page may include JavaScript code (a PHP document may contain static HTML labels just as PHP declaration [2, 9].

TABLE I. Vulnerability Classes Split by Vulnerability Categories

VULNERABILITY Class	VULNERABILITY CATEGORY	OVERVIEW	VULNERABILITY Name
Improper input validation	Query Manipulation/injection	Vulnerabilities that are related to structures that are store information in the databases.	SQL Injection, NoSQL Injection, Xpath and LDAP Injection
	Client-side injection	Vulnerabilities associated with malicious code injected by a client-side such as JavaScript and processed by the server-side.	Cross-site scripting
	File and Path injection	Class of vulnerabilities that manipulate the relative path and redirect to a different location.	Remote document \local record consideration, Path/Directory Injection and Remote Code infusion
Improper session management	Session Management	Sort of malicious exploit of a site where unapproved directions are transmitted from a client that the web application trusts.	Cross-site request forgery
Improper authentication and authorization (Logic Flaw)	Logic Flaw	Vulnerabilities that can be manipulated with the coding of web application and changing them.	Unreliable Direct Object Reference, missing Functional access Control, Invalidated Redirects and Forwards or application rationale susceptibilities

1) Query Manipulation

Query manipulation is a vulnerability related to structures that store data like databases and where malicious code manipulates queries and changing them. With the help of these web vulnerabilities, a hacker easily manipulatesthe parameter of user input. As a result, the attacker becomes able to change the query's syntax. When the validation of these parameters is not proper, the maliciously infected parameters enter the reliable website due to which unsafe and unreliable information enters the web applications and damage its security. Hence, the missing or improper affirmation of controllable user data is the leading causee for injection vulnerability. There are different types of web vulnerabilities such as SQLi, LDAPI, and NoSQL. These vulnerabilities are related to the construction of filters and queries that are operated by some kind of engine example DBMS. SQL injection is considereda famous and exploited vulnerability. The other vulnerabilities are the same as SQLi, i.e. If a query involves sanitized user inputs with malicious characters then the behavior of the query performed can be altered [2, 9].

B. Client-Side Injections

Client-side injection enables malicious code to be executed by an attacker like JavaScript payloads on victim browsers without a server request. There are different vulnerabilities in this category such as XSS, remote code execution (RCE), and email injection (EI) [6].

1) File and Path Injection Vulnerability

In this class of vulnerability, a hacker manages the entrance to records from web applications or a document framework and URL areas not quite the same as the web application. These are the weakness which has a place with this gathering are RFI, LFI, and Directory traversal (DT) otherwise Path Traversals (PT) [6]. In this category, we have only considered Local file inclusion and remote file inclusion for this study.

C. Improper Authentication and Authorization (Logic Flaw)

Improper authenticating and authorizing procedures imply the invalid exercise of protocols like access control policies also known as "ACPS" as well as functions of authenticating. The logic of web application is generally executed by applying the application's control flow and saving by protecting sensitive information. One can achieve this situation or condition directly by keeping safety measures and checks to the coding of source or indirectly by the path directions provided to users like interface screening. Unsuitable implementation of business logic represents the logic errors, which force the application to behave in different ways as expected from it which results in dropping standard in "QOS" known as quality of service, losing both finance and information through the leakage. Three out of 10 top security-related hazards about applications of web [OWASP Top 10] be able to refer missing Insecure Direct Object Reference, Functional access Control, and Invalidated Readdresses and simply application logic susceptibilities [2, 9].

D. Improper Session Management

Web applications use the web session to recognize and associate multiple web entries from a single user within a specific period. A collection of web sessions is referred to as a session of a web, it may be utilized by the website for keeping the details, path of states from the past web requests and may change the further operations. In web application development, the management of the session is achieved by the cooperation of the client and the server with each other. The general tactic to do this is that an exclusive identifier (like a session ID) sent to the client by the server after the successful verification of the user. Securing alone the session ID will not be enough for managing the protected session. Session hijacking is performed by hackers through a malicious request linked to the authentic session ID. CSRF is a well-known outbreak in this category as listed in OWASP top 10 web vulnerabilities. The vulnerable web application on risk could not identify if web requests are infected or malicious until these are associated with valid session information [2, 9].

1) Session Management

Website use web session to recognize and associate multiple web entries from a single user within a specific period [2, 9]. The vulnerabilities that belong to this group are clickjacking, CSRF, Session fixation, and the hijacking of a session[10]. In CSRF, hacker submitsa malicious request as a legitimate user to web application.Clickjacking is a type of attack that invites a person to click or appeal on objects placed in infected pages and by doing this, some undesirable actions may happen without any consensus of the authentic person . Session fixation and hijacking are those attacks that aim for the user's session ID, on the other cross-Site request forgery and hand clickjacking also CSRF focus on the fact that illegal request on behalf of user [2, 11].

III. Related Work

Existing secondary studies on the topic of securing web applications are discussed (survey papers, review articles). Fig. 2 presents relevant reviews of the literature published over the past 13 years. Much work has been published to identify a taxonomy for vulnerabilities in software. Delgado et al. [12] built up a scientific classification for ordering the runtime programming flaw observing methodologies and monitoring them in light of three elements: component utilized for checking program execution and language.

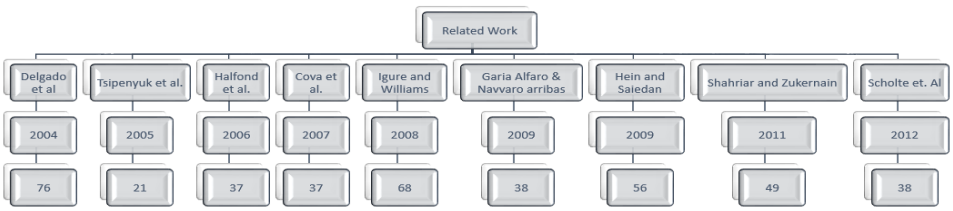

Fig. 2. Related Work.

Tsipenyuk et al. [7] arranged common flaws causing web vulnerabilities, seven out of eight categories are related to environmental and configuration issues. Many attacks and vulnerabilities are classified with various taxonomies developed and submitted in Igure and Williams' comprehensive survey[13]. Krsul [14] classifications classify vulnerabilities in software. The examinations by Halfond et al. [15], Chandrashekhar et al. [16], and Garcia-Alfaro and Navvaro-Arribas [17] give an audit on the strategies for relieving the most dangers vulnerabilities such as SQLI and XSS. The study by Cova et al. [18] features the advantages and disadvantages of weakness examination instruments accessible to secure the website. Fonseca et al. investigation [19] outline the coding flaws that should be avoided in C#, Java, and PHP. In another study, Shahriar and Zulkernine [20] gave the best in class approaches accessible for discovery and the aversion of hack attempts on applications under operation. Furthermore, they discussed the methodologies for moderating web vulnerabilities atthe program level In their study, Hydara et al. [21] discuss the methods of cross-site scripting vulnerability. The XSS systematic literature review highlights various systems for discovering and avoiding XSS attacks.

Wedman et al. [10] presented a definite survey of vulnerabilities aimed at launching session hijacking attacks and available mechanisms to protect users from such attacks for the protection of web applications from vulnerabilities, various methods are utilized and described in Li and Xue [9]. All the previously mentioned audits concentrate on any of the accompanying perspectives such as (I) building up a scientific categorization for characterizing attacks and vulnerabilities, (ii) detect the coding flaws that are abused for propelling attacks, and (iii) categorizing the flaws checking methodologies. The survey on SQLI distributed in 2012 does not take after a methodical strategy that confines the range of their investigation.

Deepa and Santhi [2] provided up-to-date approaches to web vulnerability prevention. This paper is divided into different phases of the software development life cycle with 86 primary studies. There is different web vulnerabilities research paper such as in case of XSS is thirty-five, in case of SQL injection is seventeen and in case of logical bug is thirty-five. Buczak, Anna, and Erhan [22] describe a literature survey on data mining and machine learning for intrusion detection. The latest review by Ghaffarian, Seyed, and Hamid [23] provided a detailed review of the many different methods based on machine learning that analysis and discovery of software vulnerabilities.

IV. Categorize Existing Countermeasures

Numerous researchers around the globe are working on several different ways to detect web vulnerabilities. The following sections present different technique/methods to find web vulnerabilities, such as static analysis, fuzzing and dynamic analysis, hybrid, machine learning technique, and secure programming,

The basic significance of the issue of web vulnerabilities is that many methodologies are researched and proposed. The suggested approaches are not absolute; All of them either need soundness or they are incomplete. Subsequently, all research is working to urge an enhanced approach contrasted with past works, referring to a particular part of the procedure of web vulnerability examination and revelation/discovery; like coverage of vulnerability, discovery exactness, runtime efficiency. Shahriar and Zulkernine [24] presents an extensive review to prevent web vulnerabilities reported during 1994 to 2010.

A. Secure Programming

Secure programming allows programmers to follow secure practices when they are developing the web application. Secure Programming protects coding practices by coding properly, checks the input data; encode correctly the user input, its type further by setting the query's parameter, also by bringing stored procedures to work. Query statements are named to those queries whose parameters are set with placeholders like "?" for referring to user data. SQL code handling placeholders in the string, which is attacking just like input. Queries that are parameterized and procedures already saved bear the same outcome however great measures are considered when programmed. Moreover, in developing of website, SQLIA's still a problem[2, 9].

To protect web apps from attackers, it is important to keep a close eye on the security features at every stage of the lifecycle while developing the web application. It is referred to as SDLC. After setting up a web application we must furnish the secondary security layer [2, 3]. Now day's operating systems are even more secure from the systems years back. The reason for this is the placement of automatic tools of safeguarding and protection within the compilers, core library alike DEP, and .NET respectively. In Linux and windows, stack or canaries cookie may also be used frequently [25]. These tools or systems stops a wide range of attacks without considering about the programmer practicing secure programming practices or not.

The writing of a safe program code has made clear on developers by the deployment of the website. Furthermore, due to the utilization of the stones library, it would be resolved in Java-based applications and Juillerat [26] that applies this technique. This library allows hackersto use databases using OOP and JavaScript payload instead of SQL payloads. The direct replacement of input data provided by the user as a string cannot be possible because it only goes via suitable procedures. Hence the programmers don't have to do much, limiting the additional work as the security features are controlled by the library. It can easily get rid of unsafe string code practice and when the number of queries framed. It can be performed by placing in the data and code a visible partition and Johns et al [27] accomplished. They achieved this by representing query syntax by the ELET (embedded language encapsulation type) introduction. To prevent attacks of XSS, Grabowski et al. [28] The created type system used in Java programming, implement directions of secure and safe programming.

A study was carried out by [29] to allow safe web development by using swift programming model language formed on the Jif language. This language confirms the integrity and confidentiality of information within the program code or declaring annotations description. The locations of the server or client can be recognized to secure placement data.

Another study proposed by Vikram et al., [30]. They give a new method Ripley, a replacement of Swift programming, to evade irregularities within the logic of business across both ends an impression of computational logic is the site on the side of the server that is present on the position of the client. Ripley confirms the reliability of RIAs and prevents from the extra work of within code annotation addition. However, information privacy cannot be guaranty and also it enforces memory, network overhead, and the reason for this is that it moves and positions from client to server every event. A language runtime used for the applications based on the PHP and python known as "Resin" permits the developers to use the already present code of application again to generate assertions that allocate the security policies. A comprehensive study conducted by Yip [31] to avoid Missing Access Control, XSS, and SQLI like multiple issues.

To develop new and secure web application an enormous frameworks of coding are created to preserve the data and important information present in web application with their reliability. To support authorizing rules or directions of web applications as acting like interpreting authority. Additionally, type checker an intermediate coding language created by Jia et al. [32] called "AURA". To allocate and verification that security policies applied properly or not by the integration of information flow and access control for web application Swamy et al. [33] enforced a system kind known as Apologue. To build a secure and safe multi-level web app a coding language is created as SELinks by incorporating language links with a fable type system and this is done by Corcoran et al. [34]. In this type, SELinks compiles the code relating to the implementation of policy to functions within the database defined by the user while fable finds the missed authorization checks. It does not guarantee the security policies relating to the state of the web application is tackle by Swamy et al. [35]. They give stateful approval approaches to the application. Krishnamurthy et al. [36] Proposed a method capluse to secure the web application with secure practices.

Another method proposed by [37] is the intelligent static examination that coordinates static investigation into the Integrated Development Environment (IDE). Additionally, provide secure programming support in-situ that helps developers stop vulnerabilities while building code. There is no need for further training and there are no hypotheses as to how programs are being developed. His work is inspired in portion from the observations that are the number of vulnerabilities introduced because many knowledgeable developers fail to practice secure programming. They have employed an interactive tool for prototype static investigation similarly as a module for Java in Eclipse. Kang and Park [38] suggested a smart fumigation system made in connection with the black box and white box test that could effectively detect/distinguish software weaknesses.

Na Meng [39] study served a wide reception of the validation and approval highlights gave by Spring Security - an outsider system intended to make sure about big business applications. They found that programming difficulties are generally identified with APIs or libraries, including the entangled cross-language information treatment of cryptography APIs. Moreover, discoveries uncover the deficiency of secure coding help and documentation, just as the gigantic hole between security hypothesis and coding rehearses.

The most recent study conducted by Bangani, S et al [40] proposes the educating of secure programming through a bit by bit approach. Our methodology incorporates the distinguishing proof of utilization hazards and secure coding rehearses as they identify with one another and to fundamental programming ideas. We explicitly mean to control instructors on the most proficient method to show secure programming in the .Net condition.The most recent study conducted by Agrawal, A et al [41] proposes an integrated and prescriptive framework intended to identify and mitigate vulnerabilities and provide suggestions for writing a more secure code.The detailed research review on a secure programming method to find XSS, SQLi, CSRF, LFI/RFI, and some other vulnerabilities presented in Table II.

TABLE II. SECURE PROGRAMMING EXISTING STUDIES

			Area of Focus					**Web Vulnerabilities**			
Research Article	***Language/Framework/Tool***	***Year***	***Vulnerability Detection***	***Vulnerability Prevention***	***Attack Detection***	***Attack Prevention***	***Vulnerability Predication***	***Query Manipulation/injection***	***Client side injection***	***File and Path injection***	***Session Management***
Chong, Vikram [29]	Swift	2007		✓		✓		✓	✓	✓	✓
Jia et al. [32]	Aura	2008		✓		✓					✓
Swamy et al. [33]	Fable	2008		✓		✓					✓
Vikram et al. [30]	Riply	2009		✓		✓		✓	✓	✓	✓
Corcoran et al. [34]	SELinks	2009		✓		✓					✓
Swamy et al. [35]	FINE	2010		✓		✓					✓
Krishnamurthy et al. [36]	Capsules	2010		✓		✓					✓
Zhu et al. [37]	ASIDE	2014	✓		✓			✓	✓		
Kang and Park [38]	WVF	2017	✓				✓	✓	✓	✓	✓
Na Meng et al. [39]	Empirical Study	2018	✓					✓	✓		
Bangani, S et al. [40]	Study	2019	✓					✓	✓		
Agrawal, A et al[41]	Framework	2019		✓				✓	✓	✓	✓

1) Discussion

There are numerous existing studies on preventing and detecting vulnerabilities in web applications through secure programming. Developers and firms need to focus on testing each bit of software and each application in their portfolios. By doing this right on time in the website improvement process, both can decrease, the expenses related to security. Application firewalls can be utilized as countermeasures to those attempting to hack information from an IP address. Other encryption, antivirus, antispyware, and confirmation software can be particularly utilized. To protect web applications from attackers its important keeping close eye on the security features at every stage of lifecycle while developing the application software. It is also referred as SDLC further after setting up website and it must be furnished with secondary security layer. XSS, LFI/RFI, RCE, SQLI and CSRF [2]. There are some different approached for safe coding which are distrust user input, input validation and magic switches and some tools to perform automatic source code analysis Rats, Flawfinder and ITS4. From the existing studies we conclude that Agrawal, A et al [41], Kang and Park [38], Zhu et al. [37] useful approaches in secure programming. The methods of secure programming is summarized year wise shows in the Fig. 3.

B. Analysis Method to Detect Web Vulnerabilities

There are different methods to prevent web vulnerabilities such as white box testing, blackbox, and fuzzing methods.

1) WHITE Box Testing

In the white box, the tester accesses the software code and knows the web source code's internal process. While it is possible to check how the input value of the software deduces the result value, however. This test allows access to possibly hidden source codes and code errors. The benefit of this process is that the input value can be easily predicted and a test scenario can be made. However, the white box test requires experienced skills and it is not possible to guarantee that the test specifications are met [38]. The method proposed by Jovanovic et al. [42] is suggested pixy tool.

2) Black-box Testing

The black box test depends on the software for the tester. The tester is unaware of how the software operates internally. It only tests with the result value deduction corresponding to the function-based input value. This method is advantageous for the tester as it does not require source code information or technical skills. However, while testing input values for a short time, some limitations that can not deduce logical errors and make a test case difficult without the knowledge of clear functional specifications [38].

C. Static Analysis

Mechanism of static analysis tools inspecting either binary or intermediate source code. Static examination means to look for potential vulnerabilities by inspecting the code of web applications without executing it [43]. The principal papers right now center around old vulnerabilities, for example, race conditions and buffer overflow. Later this kind of investigation has stretched out for executable programming without source code [44].

Programmers typically use static analysis tools during the development of software, checking if the code does not have vulnerabilities. In any case, these instruments just pursuit and identify the vulnerabilities. These apparatuses are program to, scanning for examples and utilizing rules for the sort of examination that they execute. As a result of this reality, the devices don't distinguish newfound classes of vulnerabilities in source code, potentially imaging the applications with bugs, creating false negatives – a weakness that exists not detailed. The false positives are additionally a worry, however in the feeling of causingwaste of time, since the software engineers need to review the code looking for non-existent bugs. Static investigation procedures arranged in two primary classes, to be specific lexical examination and semantic examination [43]. Next, these strategies are displayed, with more accentuation on taint analysis, a type of semantic examination. Lexical Analysis is a strategy to discover web vulnerabilities from source code. It's examined to scan for library capacities or framework calls that are not viewed as dependable touchy sinks.

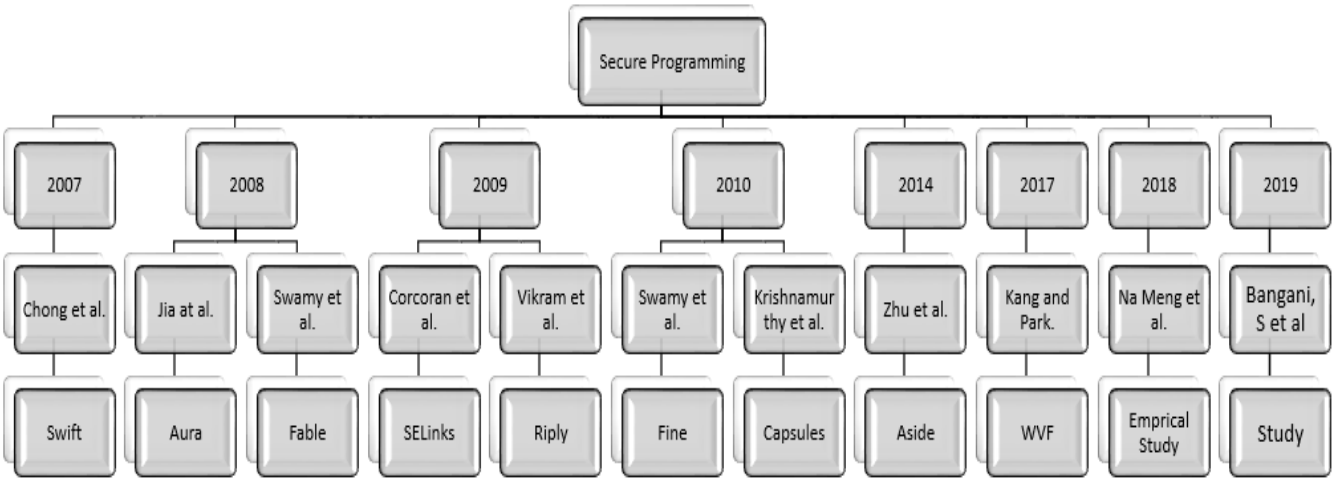

Fig. 3. Methods of Secure Programming.

This investigation includes a lot of three principal methods: control stream examination, type checking, and information stream investigation. In a study by [45], they established a static Analysis scanner WebSSARi to find vulnerabilities in web applications. This scanner provides intra-procedural and flow-sensitive reports established on the base of the lattice model. This broadens the PHP coding including two type states, known as tainted and untainted, also finds every type state of variables in it. Runtime sanitizing objects are introduced at the place the tainted data approaches the sinks. Numerous language features, like recursive functions and array elements, have not been supported, however.Using string taint analysis suitability of sanitizing procedures can be confirmed. Furthermore, Issermann and Su [46] to enhance Minamide[47] string analysis with taint support utilize this. It has analyzed the info string to perceive spoiled substring esteems to prevent any suspicious content from running by the JavaScript mediator. As it needs an understanding of the semantics of the site page the strategy can't discover DOM-based XSS.

In another study by Xie and Aiken [48] to do a reverse interpretation of fundamental blocks, methods, and the complete program to detect SQL vulnerabilities due to injection. The method they have is capable of automatically deriving the set of variables and sanitized after which function invocated by utilizing symbolic execution. Nevertheless, their static analysis technique is bound to a specific set oflanguage features. In a study done by Halfond and Orsob[49] Suggested method AMNESIA to joins static analysis and runtime monitoring to prevent SQL injection web vulnerabilities. In one more study, Shahriar and Zulkernineb[26] put forward an information-theoretic method to prevent SQL injection web vulnerabilities. The entropy of each SQL statement is calculated based on the tokens probability.

In their study, Thomas and Williams [50] SAFERPHP utilizes static analysis to find specific semantic vulnerabilities in PHP code: nullification of administration on account of vast circles, and approving tasks in databases [51]. According to the disavowal of administration, the device utilizes corrupt examination to discover circles, and afterward utilizes representative execution investigation to decide whether assailants can forestall the end of the circles. PHPSAFE (Fonseca and Vieira, [52].; Nunes et al., [63].) taints examination to scan for vulnerabilities in PHP code.Shahriar and Zulkernine [53] proposed a method to detect the vulnerabilities based on static anaylysis. Another study proposed by Shar and Tan [54] to prevent the cross-site scripting XSS method is called XSSsafer and Scholte et al. [55] proposed an IPAAS method to detect the web vulnerabilities.

Yunhui& Zhang [56] describes another use of static analysis. His approach to finding vulnerabilities in remote code execution (RCE) using the inter-procedural path and setting delicate investigation. RCE assaults require as a rule the difference in the string and non-string portions of the customer side data sources; hence, they propose an investigation that handles these parts in a composed and productive manner with the number of PHP contents and demands. They built up a calculation that comprehends these obliges in an iterative and elective style, so endeavors can be made from this arrangement. In one more study, Doup´e et al. [57] created deDacota, an automated tool that gives a clear partition between code and data in web pages. Amira et. al.,[58] proposes another static examination of web applications affirming the program's protection from meeting fixing ambushes called SAWFIX, a PHP static analyzer that outputs web applications for vulnerabilities in meeting fixation. To the best of our understanding, SAWFIX is the principle analyzer that checks extensively for this kind of powerlessness, while exchange strategies simply ensure half-precision that is limited to a modest quantity of plausible executions.

Khalid et al. [59] proposed and built up a WUM defenselessness examining apparatus (web interesting technique) to recognize and forestall every single significant weakness and tells the best way to distinguish unapproved access by recognizing vulnerabilities. The designers can discover possibly defenseless web applications with the assistance of wum Tool. WUM has created an elevated level of accuracy and similarity that is created underneath. Test's outcome shows proposed WUM helplessness scanner apparatus that gives less false positive and more vulnerabilities are identified. Another study is conctducted by Viega et al [60] on static vulnerability scanner for C and C++ code.

They developed Deepa et. al. [61] for recognizing various kinds of rationale vulnerabilities, for example, parameter control, get to control, and work process sidestep vulnerabilities. DetLogic utilizes the discovery approach and models the planned conduct of the application as a clarified limited state machine, which is thusly utilized for determining requirements identified with input parameters, get to control, and work processes. The recent study Nunes and Medeiros [62] the problem of consolidating various ASATs to improve the general identification of vulnerabilities in web applications, considering four advancement situations with various criticality objectives and limitations. These situations run from low spending plans to top of the line (e.g., business basic) web applications. The study of Long et. al, [64] has described some of the major widespread web-based vulnerabilities. These include SQLI, XSS, FI, SI etc. This study proposes an algorithm and improvements that are aimed at increasing efficiency of detecting these web-based vulnerabilities. The algorithm used to develop scanning tool use several software including UTLWebScanner. The algorithm can be compared with software providing similar functionality such as Nesus. The recent study in 2020 conducted by Aliero et al [65] to detect and minimize the occurrence of false positive and false negatives, they focus on enhancing the effectiveness of SQLIVS. They propose an object-based approach for developing SQLIVS. Three different web applications were used to test the accuracy of this approach. Each application had different types of vulnerabilities. The validity of proposed scanner was established using an experimental approach. Analytical evaluation was also used to compare the proposed scanner with other available scanners developed by various academicians. The results of experiments showed significant improvement as evidenced by high level of accuracy. The detailed research review on a secure programming method to find XSS, SQLi, CSRF, LFI/RFI, and some other vulnerabilities presented in Table III.

TABLE III. STATIC ANALYSIS OF EXISTING STUDIES

			Area of Focus					Web Vulnerabilities			
Research Article	***Language/Framework/Tool***	***Year***	***Vulnerability Detection***	***Vulnerability Prevention***	***Attack Detection***	***Attack Prevention***	***Vulnerability Predication***	***Query Manipulation/ injection***	***Client side injection***	***File and Path injection***	***Session Management***
Huang et al. [45]	WebSSARI	2004	✓					✓	✓		
Minamide [47]	analysis	2005	✓						✓		
Halfond and Orso [49]	AMNESIA	2005			✓	✓		✓			
Xie and Aiken [48]	method	2006	✓					✓			
Jovanovic et al. [42]	Pixy	2006	✓					✓		✓	
Wassermann and Su [46]	analysis	2007	✓					✓			
Thomas and Williams [50]	model	2007		✓				✓			
Shahriar and Zulkernine [54]	method	2009				✓		✓			
Son and Shmatikov [51]	SAFERPHP	2011	✓					✓	✓		
Shar and Tan [54]	XSSsafer	2012	✓				✓		✓		
Shahriar and Zulkernine [24]	Program	2012	✓					✓			
Scholte et al. [55]	IPAAS	2012		✓				✓	✓		
Yunhui & Zhang [56]	script	2013					✓		✓		
Doup´e et al. [57]	deDacota	2013		✓							
Fonseca & Vieira [52]	tool	2014		✓				✓	✓	✓	✓
NUNES et al. [63]	phpSAFE	2015					✓	✓	✓		
Khalid et al. [59]	WUM	2017	✓					✓	✓	✓	
Amira et al. [58]	SAWFIX	2017					✓				✓
Deepa et al [61]	DetLogic	2018	✓					✓	✓		
Nunes and Medeiros ss[62]	ASAT Study	2019	✓					✓	✓	✓	✓
Long, et al. [64]	UTLWebScanner	2020	✓					✓	✓	✓	
Aliero et al [65]	Scanner	2020		✓			✓	✓			

1) Discussion

Static analysis tools, either source, binary, or intermediate, mechanize code inspection. The objective of the static examination is to look for vulnerabilities in the source code without running it [43]. Because in the development process, static application security testing tools are used early. Before software is deployed, they can identify vulnerabilities. These tools test line by line the source code, prevent flaws and provide the opportunity to fix them before becoming a true vulnerability on the web. It requires access to source code or binaries that certain organizations or individuals may not want to abandon application testers. In order to detect vulnerabilities before deployment into the live environment, it usually needs to be integrated into the system development lifecycle, which can make implementation difficult.

Each SAST tool tends to focus on a subset of possible weaknesses. The advantages are the ability to detect vulnerabilities that are not visible without access to the source code.The capacity to reveal to you the specific area of any source code shortcomings, including the line number. Probably the greatest test to choose the correct instrument when utilizing SAST is the number of false positives produced. From the current, valuable methodologies in the static examination are NUNES et al [63] Khalid et al. [59] and Nunes, P et al. [62]. The methods of secure programming are summarized year-wise as shown in the Fig. 4.

D. Fuzzing and Dynamic Analysis

Fuzzing and dynamic analysis is another method to identify web vulnerabilities In this method does not analyze web application code for vulnerability detection from static analysis but verifies in runtime whether injected data triggers some vulnerability in application specifications [38]. In this way, it is viewed as a testing procedure that finds bugs in programming by taking care of a program with s unexpected inputs (Evron&Rathaus, [66]; Sutton et al., [67]. In their study, NguyenTuong et al. [68] altered PHP transcriber to exactly infected data of the user on the character's granularity and it traces tainted user data at runtime. In another examination, Haldar et al. [69] formulated the arrangement of Java bytecode that can grow the Java framework with inadequate following assistance. These systems will in general be easy to apply in light of the fact that it doesn't require information about the program to test. Its cooperation with the program is constrained to the program's entrance focuses Jimenez et al. [70]. Mill operator et al. [71] that depicted how they took care of UNIX program utilities with irregular data sources, such as SPIKE [72], improve this thought by giving to the applications distorted sources of info, utilizing a conventional information structure to speak to various information types [67].

In an examination done by Huang et al. [73] utilized a system named as WAVES distinguish both SQLI and XSS vulnerabilities in web applications. Another utilization of Dynamic investigation Huang et al. [45] created white-box instrument b same group called WebSSARI, though WAVES.

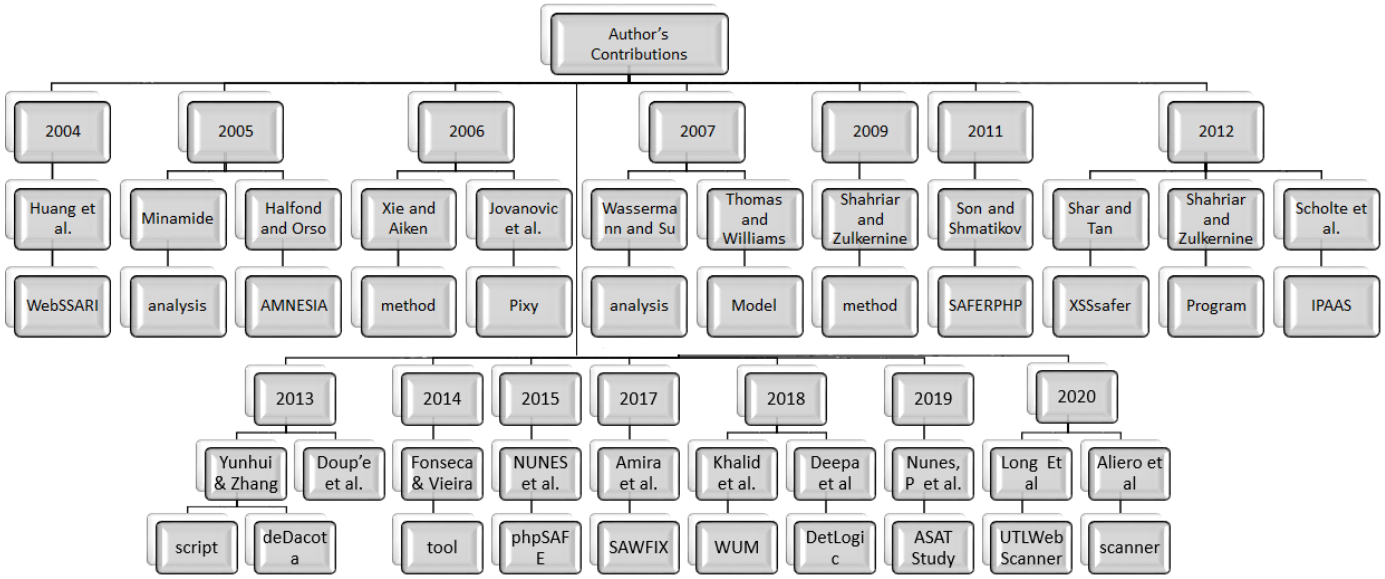

Fig. 4. Methods of Static Analysis.

perceive vulnerabilities with the assistance of a discovery approach In 2006, Kals et al. [74] built up a discovery defenselessness scanner, Secubat, to perceive vulnerabilities. The tool utilizes a crawler to perceive the site pages of the application, possess the structure fields on pages with assault vectors, and afterward break down vulnerabilities to distinguish them.

It is possible to classify fuzzers into two categories: blackbox and whitebox [67]. A blackbox fuzzer executes the method portrayed as yet. As the blackbox approach is generally free of the application and doesn't require setting up the application. Regardless of black-box fuzzers being helpful, they will, in general, find just shallow (bugs that are anything but difficult to discover) and as a rule have low code inclusion (don't practice every conceivable incentive for a given variable), missing numerous pertinent code ways and in this way numerous bugs. KameleonFuzz is a blackbox fuzzer that scans for cross-site scripting. It creates malignant contributions to control cross site sciprting XSS [75].

Another technique of dynamic analysis proposed by Antunes et al., [76] is related to the black box fuzzing in a certain way that is attack injection. A tool that implements this technique intends to mimic the behavior of an attacker, and continuously inject malformed inputs while monitoring the application. The procedure is rehashed to assemble all conceivable execution ways and checking a few properties in runtime [66, 67]. This is a form of white-box fuzzing that is actualized in the SAGE utilizing emblematic execution to practice all conceivable execution ways of the program. Since representative execution is moderate, in any case, it doesn't reach out to huge projects, it is difficult to find profound and complex bugs [84].

In a different study, Ciampa et al. [77] chose the result of the different advance tests on pattern matching the valid and error messages. Data stored in the form of tables and fields are tested by an empirical tactic to evaluate the gathered information. After all the computation, the compiled data is utilized to check attack inputs that are useful to recognize the vulnerabilities. In another study by Lekies et al. [78] used a taint-aware JavaScript engine to sense DOM-based XSS. Whereas, it is out of bound the other available methods to perceive these vulnerabilities. Whitebox fuzzers apply symbolic execution and imperative tackling to the source code Duchène et al., [75]. The working rule of some white-box fuzzers is to produce and to perform dynamic representative execution in an occurrence. It accumulates information stream ways and requirements on contributions from contingent branches that are experienced along with the execution. Then, the collected constraints are negated (constraint solving) and new inputs are injected to collect new execution paths. To deal with this difficulty, Dowser is a combination of symbolic execution with dynamic taint analysis to identify vulnerabilities in buffer overflow buried deep within Haller et al. [79] Programs implemented in their study.

In one, more study Duchene et al. [5] In order to get the auto production of unwanted inputs to access XSS vulnerabilities, the author used a genetic algorithm. Whereas, most of the available techniques do not have such an ability to reach the cause of DOM-based XSS vulnerabilities. In their study, Dohse, and Holz [80] purposed a very first known automatic testing method that uses static code to notice second-order vulnerabilities and correlate more than one step attacks in web applications. The flow of unattended data can be detected by checking the incomings and outgoings from the webserver. It has been a successful identification of unsensitized data streams by linking input and output points of data in databases. Another dynamic Analysis study by Weissbacher.et al., [81] gave a system to strengthen the JavaScript-based web application to protect them from the used side attacks named ZigZag. It is a tool of client-side code. It produces a model that tells how and with whom the client-side section is in the network. It is efficient enough to perform dynamic security code invariant detection by the respective models as well as it

has the ability to handling templated JavaScript bypassing overall re instrumentation in cases where the JavaScript programs are structurally identical.

RanWang et al. [82] propose a unique recognition structure (TT-XSS) for DOM-XSS by methods for the pollute following at the customer side. They modify all JavaScript highlights and DOM APIs to corrupt the rendering procedure of programs and vectors are inferred to check the vulnerabilities naturally. In the recent study Park, et al. [83], a vulnerability detection technique is proposed that develops and manages safe applications and can resolve and analyze these problems. They developed a prototype analysis tool using our technique to test the application's vulnerability–detection ability, and show our proposed technique is superior to existing ones. The recent study in 2020 conducted by Falana [85] used Dynamic Analysis and Fuzzy Inference. The combination of these two techniques allowed them to come up with a hybrid mechanism that can be used for detection of XSS attacks. This approach used scans of website for possible SQL injections. Once this scan was done, they launched an attack vector using a HTTP request. The approach was used to test some active web applications. The results showed a large number of vulnerabilities were detected successfully. the detailed research review on dynamic analysis to prevent web vulnerabilities shown in Table IV.

1) Discussion

Dynamic analysis is a useful technique to prevent web vulnerabilities and does not analyze the source code of the website but verifies in runtime whether injected data triggers some vulnerability in the application. With this strategy, DAST tools offer risk examination and aids in the remediation endeavors, engineers do not generally know where precisely the vulnerabilities are found, nor do they generally know what countermeasures to execute. DAST approach detailing is not as much as agreeable in various examples. From the existing study RanWang et al.[82], Park et al. [83] are useful approaches in dynamic Analysis. The methods of dynamic analysis are summarized year-wise shows in Fig. 5.

TABLE IV. DYNAMIC AND FUZZING ANALYSIS EXISTING STUDIES

			Area of Focus					**Web Vulnerabilities**			
Research Article	***Language/Framework/Tool***	***Year***	***Vulnerability Detection***	***Vulnerability Prevention***	***Attack Detection***	***Attack Prevention***	***Vulnerability Predication***	***Query Manipulation/injection***	***Client side injection***	***File and Path injection***	***Session Management***
Huang et al. [73]	WebSSARI	2005	✓					✓	✓		
Haldar et al.[69]	For java	2005	✓					✓	✓		
Kals et al. [74]	Secubat	2006	✓					✓	✓	✓	✓
Antunes et al. [84]	Tool	2010	✓					✓	✓		
Ciampa et al. [77]	Tool	2010			✓			✓	✓	✓	
Lekies et al. [78]	Approched	2013	✓						✓		
Duchene et al. [5]	black-box fuzzer	2014	✓						✓		
Dahse and Holz [80]	Tool	2014			✓				✓		
Weissbacher et al. [81]	ZigZag	2014		✓					✓		
RanWang et al.[82]	TT-XSS	2018	✓					✓	✓		
Park et al. [83]	TOOL	2019	✓				✓				
Falana et al.[85]	approach	2020		✓				✓			

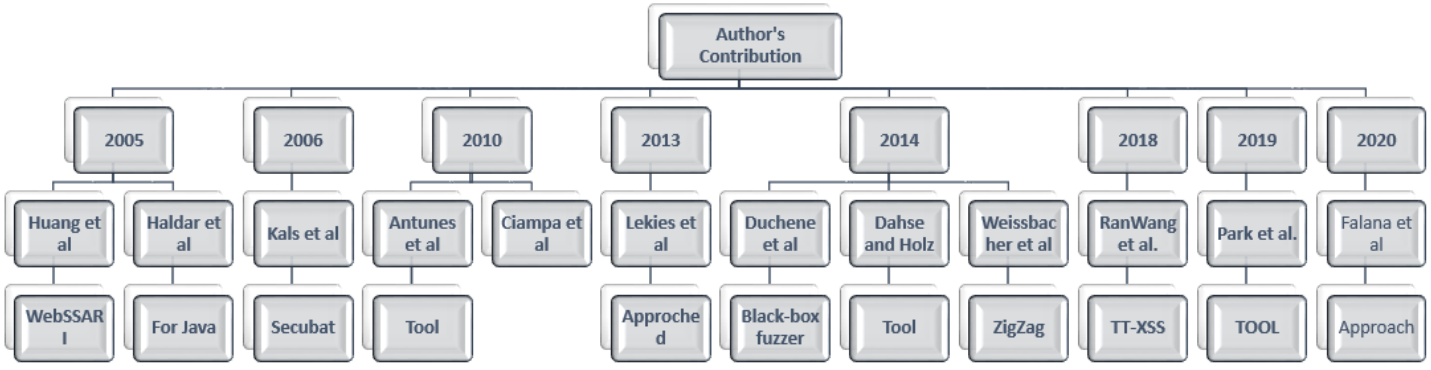

Fig. 5. Methods of Dynamic Analysis.

E. Hybrid Method (Static + Analysis)

Extracts of static and dynamic analysis are mixed to be named as hybrid analysis and provide a path toward precision analysis. In a study by Di Lucca et al. [86] identified vulnerable web pages by studying the application's source code by arrangement control flow graphs. XXS attacks are chiefly because of wrong input sanitization functions. Some web applications are also not successful to separate the suspected entries in the inputs. In another study by Balzarotti et al. [87] claimed various elusive defects could be introduced in the web application due to defective sanitization. These subtle flaws cannot be detected by the static and dynamic practices. The hybrid analysis is utilized by Saner to identify the validity of built-in and customized sanitization procedures. Saner is the first to implement the conventional static string investigation to model the working of user input sanitization. Saner first applies conventional static string analysis to model the sanitization of user input. In order to mark frail or wrong sanitization, a big series of malicious inputs are introduced into the test sanitization procedure.

Livshits et al [88] and Lam et al., [89] studied model checking, static and dynamic inspection, and runtime detection to purpose a holistic method. Which enhances the precision of static analysis by specifically using model checking. Model-checking can analytically search the space of a limited state system. It confirmed the authenticity of the system in reference to the provided conditions or characteristics. As well as this method of checking is capable to automatically produce tangible attacks, produce no false positives in vectors, and exploit path. Another technique of hybrid analysis study by Van Acker et al. [90] found the XSS vulnerabilities in flash applications by setting up Flashover. Whereas, the previous works up until focused on the discovery of conventional XSS web application vulnerabilities, Flashover identifies vulnerabilities in RIAs (Rich Internet Applications).

Another research is led in 2012 Lee et al. [91] struggles for finding SQLIAs by adopting both the static and dynamic methods. He examined the source code then the model of query is deduced from it. It removes the characters involved in SQL queries. After identifying and removing miscellaneous values, the obtained syntax is stored. The syntactic structure of quires is analyzed and compared with the already saved structure, this is how attacks are perceived in the runtime. The pros of using this scheme are that it can identify attach during the process. Another research Lee et[92], also applied both static and dynamic analysis methods for vulnerabilities of web applications. Along with the combination of these other techniques that are also being utilized for the specific application, dynamic black-box testing based on a fuzzing method is included in it. Vogt et al. [93], and Stock et al. [94] deploy the method to prevent the client-side browser from scripting XSS cross-site.

They propose He, X et al [95] a crossover examination strategy consolidating static and dynamic investigation for recognizing noxious JavaScript code that works by first directing grammar examination and dynamic instrumentation to remove inward highlights that are identified with malignant code and afterward performing characterization based identification to recognize assaults. In particular, MJDetector can distinguish JavaScipt assaults in current website pages with high precision 94.76% and de-jumble muddle code of explicit sorts with exactness 100% though the gauge strategy can just identify with exactness 81.16% and has no limit of de-obscurity. The recent study proposes Le et al. [96] E-THAPS which actualizes a novel discovery component, an improved SQL infusion, Cross-site Scripting, and helplessness identification capacities. For vindictive web shell identification, pollute examination, and example coordinating techniques are picked to be actualized in GuruWS. the detailed research review on hybrid analysis to find XSS, SQLi, CSRF, LFI/RFI, and some other vulnerabilities shown in Table V.

TABLE V. HYBRID (STATIC AND DYNAMIC) ANALYSIS EXISTING STUDIES

			Area of Focus					**Web Vulnerabilities**			
Research Article	***Language/Framework/Tool***	***Year***	***Vulnerability Detection***	***Vulnerability Prevention***	***Attack Detection***	***Attack Prevention***	***Vulnerability Predication***	***Query Manipulation/injection***	***Client-side injection***	***File and Path injection***	***Session Management***
Di Lucca et al. [86]	WA	2004	✓						✓		
Livshits et al. [88]	TOOL	2005	✓					✓	✓		
Vogt et al. [93]	novel	2007		✓		✓			✓		
Balzarotti et al. [87]	Saner	2008	✓					✓	✓		
Lam et al. [89]	*	2008	✓					✓	✓	✓	
Van Acker et al. [90]	Flashover	2012	✓						✓		
Lee et al. [91]	*	2012			✓			✓			
Stock et al. [94]	*	2014				✓			✓		
He, X et al [95]	MJDetector	2018	✓					✓	✓		
Le et al. [96]	GuruWS	2019		✓				✓	✓		

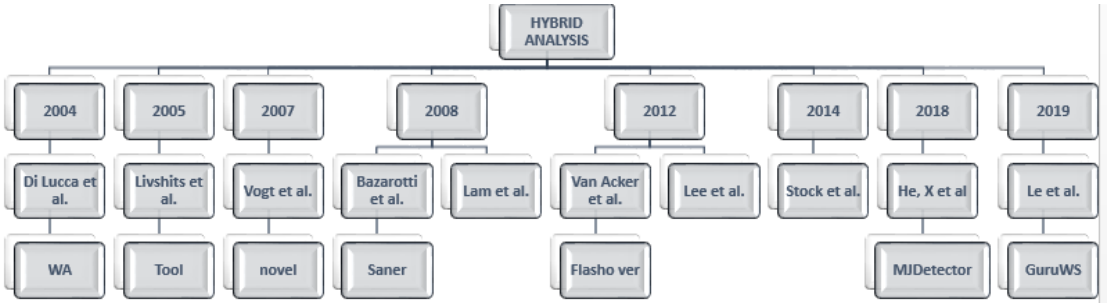

Fig. 6. Methods of Hybrid Analysis.

1) Discussion

Extracts of static and dynamic analysis are mixed to be named as hybrid analysis; it provides a path toward precision analysis. Hybrid Analysis (also called correlation) combines DAST and SAST to correlate and verify the results. Issues identified using dynamic analysis that will be traced to the offending line of code. SAST issues can be automatically prioritized using DAST information. The challenge with hybrid analysis is that DAST relies on data being reflected in the browser, so if a SAST data flow is not reflected in the browser as a DAST issue. From the existing study, Le et al. [96] and Stock et al. [94] are useful approaches in hybrid Analysis. The methods of hybrid analysis are summarized year-wise shows in Fig. 6.

F. Machine Learning Technique

This Technique is utilized in a few application zones (e.g., computer games and robotics). Application security on the web is based on a diverse package of techniques as presented in Fig. 7. It empowers PCs to learn information without programming (coding) it, and afterward to utilize the obtained information to take activities/choices. PCs must be guided to learn before taking activities. They need an informational index of models – preparing informational collection – from which to remove information, gaining from that point. An undertaking is called arrangement on the off chance that it expects to appoint input objects into classes. A classifier is a programmed technique that does classification. A classifier proceeds the dataset to collect the features and classify the dataset and provide the result based on machine learning. Email spamming is a basic example to filter the emails [97]. Machine learning is a different method to prevent web vulnerabilities.

1) Vulnerability Prediction Models in light of Software Metrics

Characterization is a type of information investigation wherein models predict the result. Model is used to predict input data class labels because each training instance's class label is referred to as supervised learning [2].

2) Anomaly Detection Approaches

To extract a program source code model & recognize vulnerabilities as separate from the usual dominant parts and principles, this work class uses unsupervised learning. This technique model isn't utilized to the class in the dataset to prevent web vulnerabilities [2].

3) Vulnerable Code Pattern Recognition

This is another machine learning method that selects the specific patterns of vulnerable code from the data set and utilizes the pattern matching to prevent web vulnerabilities on web applications [2].

4) Miscellaneous Paths

This method is used in the area of AI and data science for programming weakness in software programming and disclosure, which are not suitable other previously mentioned classes

The dataset has some attributes that the set of all instance forms of a training dataset.Attributes are divided into two categories first is numerical and the second type is categorical. Illustrating the first category, it is either discrete of continuous and named as numerical attributes. Whereas, categorical attributes possess non-numerical and distinct values. Categorical attributes have a special kind of binary attributes. Binary attributes have two expected values that are either true or false [98]. Therefore, dividing and arrangement in the form of classes is a type of data examination. It includes extracting models that specify data categorically discrete or unordered class labels, these models are known as classifiers. Classification of data involves two phases: (1) learning, where the classification model is made; (2) classification, where class labels for given input data are predicted by the respective model. Supervised learning is a class in which each training instance is labeled. For example, the input of the classifier is managed in the sense that it is programmed to identify each training instance belongs to which class. An alternative type of machine learning is where each class is unidentified to any attributive vector such a type is known as unsupervised learning. Moreover, the process does not know the set of learned classes prior. [99]. each classifier utilizes an AI calculation that relies upon the learning type (supervised or unsupervised). Furthermore, the training data set is used to classify correctly about the input data. the selection of classifiers depends on the data set factors [100].

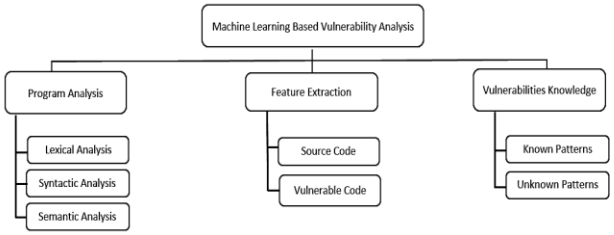

Fig. 7. Machine Learning-based Vulnerability Analysis.

Many researched have focused their studies to enhance the efficiency and precision of different techniques to detect web vulnerabilities. Support Vector Machines (SVM), J48, Artificial Neural Network, and many other classifications of techniques such as C5.0, Naïve Bayes, and linear regression are tested to train different datasets in order to detect web vulnerabilities. They are majorly grouped in two categories: probabilistic and machine learning. These techniques provided algorithms that are proved fruitful to cope with web vulnerability issues. Selected algorithms are analyzed by four metrics of, precision, recall, F1-score, and accuracy.

Suggested vulture tool in a study by Neuhaus et al. [101]. This unique tool will automatically explore existing vulnerabilities in archives for databases and versions. Vulture uses mine information to identify past component vulnerabilities. In addition, the components identified are classified by the most vulnerable to at least one type. This ranking of these components serves as a ground for investigations of the factors, which causes vulnerability to the targeted component. For instance, the study on the history of Mozilla vulnerability reveals an unexpected outcome that the components have one past vulnerability are mostly not affected by more vulnerabilities. In addition to it, those components that have the same functions calls are prone to vulnerabilities.

Machine learning has been utilized in certain attempts to quantify programming quality by gathering traits that uncover the nearness of software defects. Code type, counts of code lines, code metrics complication, and objected-oriented topographies are attributed in various studies. Some studies move ahead to consider the same type of metrics to guess the presence of vulnerabilities in source code. Moreover, other factors like past vulnerability events, called function and complication in codes are also used to conduct various other studies. These studies are not focused to identify bugs and mark their respective location but aim to examine the software codes according to the frequency of defect and vulnerabilities code[102].

Wang Yanya et al. [103] studied rapid density clustering called DSVRDC and intended methods to identify vulnerabilities in software using DCVRDC. Density-dependent clustering of vulnerability orders detected. The classifications examined are determined by the s-order difference. The density clustering methodology based on Rd-entropy is used to create vulnerability sequences in the first stage. Secondly, the respective vulnerabilities of the software are compared by s-order variation. Each order is dedicated to every cluster to calculate the difference in s-order as well as clusters comprises of under investigation software vulnerabilities are also computed.

In their study, Yamaguchi, Lindner, and Rieck [104] proposed a method to examine source code to detect vulnerabilities. This method schematically recognizes the API symbols of each function using lexical analysis. Then by the principal component analysis technique API symbols are introduced in vector space, and in dimension data in order to calculate the usage of API mode. Later on, the API usage mapping is created along with the estimated functions, supervisory code evaluation to classify likely vulnerabilities by utilizing known vulnerability functions as a standpoint. Another research is led in 2012. In order to protect SQLI and XSS vulnerabilities, Scholte et al [55]. combined static analysis and machine-learning and established IPAAS. This collection of information is utilized to deduce authentication policies about input fields that are helpful to save in-process attacks.

In another study by Wijayasekara et al. [105], a text mining technique was studied to remove potential vulnerabilities in the public bug database. This method creates a matrix for the term document. The mentioned process uses a text-mining technique to reach to the final task of classification of feature vectors into normal bug or vulnerability. The author has also purposed the increasing proportion of concealed vulnerabilities influence occurred during the past two years which is 53% for 53% for Linux and 10% for MySQL.

Another research is led in 2012 Nunan et al. [106], [107]. Likewise, recuperate web record and URL based highlights from an enormous box of an assortment of XSS assaults to examine how to depict the assaults and sort new potential XSS vector assaults as vindictive. Due to this enormous assortment, they perceived a lot of highlights (obscurity based, far fetched examples, and HTML/JavaScript plans) that license the specific arrangement of XSS in pages. At that point, they investigate consequently website pages to distinguish XSS assaults. These are the three stages process: one is identification and extraction of muddled highlights, the second one is unraveling of the website pages and includes, and the last one is the arrangement of pages by methods for an AI calculation.

Standard classifiers and other normal information-digging methods just search for the nearness of qualities, without relating them or thinking about their request. This can startthe wrong order and forecast. In earlier years, this perspective has been contemplated for improving exactness. Khosronejad [108] also aim to reduce the time of training during the construction of the HMM. They propose to assemble a model dependent on separated regular normal examples in follow occasions as opposed to taking each follow all alone. The follows are standard calls, since they can recognize the likelihood of deformities, by abnormal capacity call or by illicit utilization of assets as a result of assaults.. Bhole et al. [109] contrast the aftereffects of HMM and standard classifiers for the identification of oddities performed by an IDS. They infer that the HMM performs superior to the others. Another significant investigation shar and Tan [110]set forward their endeavors to recognize web vulnerabilities and to order different info sterilization methods in various classes as a lot of static code and a device called PHPMINER-1. In an investigation by Shar, Tan and Briand [111], they evaluated dynamic ascribe helplessness to supplement static traits. What's more, they utilized directed learning and estimation maps that are made together on the course of action and bunching to figure vulnerabilities. Both of these can perform exclusively in the nearness or nonattendance of marked preparing information. Creator presumed that they are appropriate without marked preparing information also.

In another examination by Soska and Christin [112]. The purpose of the study is to foresee the status of the site that will

get vindictive later on or not before it is truly undermined. This is extremely useful by utilizing AI since they are effectively recovered includes about the server of site and the facilitating subtleties of sites. The highlights removed about the sites are, for example, the structure of record framework (e.g., catalog names that show that the site is made by CMS), the structure of the page (e.g., if the site page is made by a CMS format), and the catchphrases (e.g., presence of some HTML labels). It depends on the event of these highlights; they perceive whether a site will be undermined. In another study using the machine learning technique, Howard et al. [25] proposed the Psigene system to retrieve features from a large SQL injection attack collection box to study how to describe them.

Another study led in 2014 [113], Fabian et al. purposed a technique for efficient big source code data analysis to find the vulnerabilities. The author presented a code property graph for illustration of source code in a new way. These graphs combined the idea of standard program analysis that includes abstract syntax trees, managed flow graphs, and graph of programs into a collective data assembly. We can characterize integer overflows, buffer overflows, vulnerabilities in format strings, or memory disclosures. The purposed collective informative structural model for vulnerabilities in addition to their graphs representation makes a person aware of all the above-mentioned factors. The creditability of this technique is identified by real-time application in some well-known graph database, it is successful in the Linux kernel to find eighteen unfamiliar vulnerabilities in the source code. A technique for detecting RCE, XSS, LFI/RFI, and SQLi was developed by Singh et al. [114]. This study proposed a work to improve the accuracy of the current vulnerability finding scheme. Grieco et al.'s [115] recent study, suggested a method for estimating a vulnerability by blurring. This approach deduces topologies that negate memory by analysis of a binary program. In the consequence of this analysis, all the extracted results are classified to assist machine learning. VDISCOVER is used to check if the test category has vulnerabilities. 1039 program are observed using bug hunter to extract 138308 performance sets in order to statistically investigating 76083 different function calls. Methods are proven effective as the test results have been detected and certain memory leaks have been confirmed.

In another study, Medeiros et al. [116] proposed a new approach to deduce by extraction algorithm the basic and context structure of source code to identify vulnerabilities in web applications. The author also stated context-sensitive security flaws in the prevailing most distinguished XSS (cross-site scripting) technique to find the vulnerability. It is found that the XSS methodology is unable to include user input in the output statements. In Walden, Stuckman, and Scandariato, [117], compared two important feature software metrics and text mining of web vulnerabilities. The author tried to establish a prediction model comprising for PHP. Both the techniques are cross-validated. Application with a version named as Drupal 6.0, PHP My Admin 3.3, and Moodle 2.0. are selected for cross-validation test. Validity test is performed twice; software metrics and term frequency parameters are used respectively to guess vulnerability. After this, results are compared and eminence of guess parameters is analyzed.

In their study YUN et al., [118], gave a new technology VULPREDICTOR that investigates metrics and text mining to guess vulnerable files. At last, it purposes a compound prediction model. First VULPREDICTOR builds 6 basic classifiers on a file under observation in order to produce constructs a Meta classifier. These files are classified according to their text parameters and software algorithms. This method run in two stages firstly it constructs a model then comes prediction stages. In the model construction stage, VULPREDICTOR constructs a composite structure from training source code files with (vulnerable or not) known labels. While in assuming point, this model works as to guess, whether a new source code file is vulnerable or not. In another study, Abunadi, Ibrahim, and mamdouh [119] developed an empirical study method that examines the effectiveness of cross-project prediction to guess vulnerabilities in software. The open-source datasets are incorporated and five famous classifiers are tested. The results of these classifiers are compared to check them in cross-project vulnerability prediction situations.

A study Anbiya et al. [120], focused on using PHP native token and Abstract Syntax Tree (AST) as features then manipulate them to get the best feature. We pruned the AST to dump some unusable nodes or subtrees and then extracted the node type token with Breadth First Search (BFS) algorithm. They were able to get the highest recall score at 92% with PHP token as features and Gaussian Naïve Bayes as a machine learning classification method. Another study in 2018, Kronjee et al [121] built a tool called WIRECAML a contrasted instrument with different devices for powerlessness identification in PHP code. The apparatus performed best for web vulnerabilities. They likewise gave approach a shot various open-source programming applications.

In this study Smitha et al. [122], work investigates the exhibition of calculations like choice woodland, neural systems, bolster vector machine, and strategic relapse. Their exhibition has been assessed utilizing standard execution measurements. HTTP CSIC 2010, a web interruption identification dataset is utilized right now. Test results demonstrate that SVM and LR have been predominant in their exhibition than their partners. Prescient work processes have been made utilizing Microsoft Azure Machine Learning Studio (MAMLS), a versatile AI stage that encourages an incorporated improvement condition to information researchers.

The study conducted by Noman et al. [123], fabricates 6 classifiers on a preparation set of named records spoke to by their product measurements and content highlights. Furthermore, they manufacture a Meta classifier, which consolidates the six hidden classifiers. NMPREDICTOR is assessed on datasets of three web applications, which offer 223 prevalent quality vulnerabilities found in PHPMyAdmin, Moodle, and Drupal. In their study Kudjo et al. [124], directed an observational examination on three open-source helplessness datasets, to be specific Drupal, Moodle, and PHPMyAdmin utilizing five AI calculations. Shockingly, they found that in all instances of the 3 datasets considered, models gave a critical increment in accuracy and precision against the benchmark study. Zhou et al. [125] study presents an improved

algorithm that generates test cases. This algorithm uses a new mutation method to divide test cases into various functional units to preserve their semantic structure. The results showed their algorithm not only generated better cases as compared with standard genetic algorithm and the adaptive genetic algorithm but also detected web vulnerabilities with high accuracy. Another study in machine learning is conducted by Tang et al. [126] that The statistical analysis of normal and SQL injection data was used to design eight feature types and train a machine-learning model. The accuracy of this model was 99%. The study proposed by Williams et al. [127] an integrated framework of data mining. This framework was capable of detecting evolution of web vulnerabilities. This framework three specific techniques i.e. Topically Supervised Evolution Model and Diffusion-Based storytelling technique, and prediction models. Through a series of experiments, it was shown this proposed framework not only discovered the evolution of web vulnerabilities and predict them with high level of accuracy. The methodology proposed by Calzavara et. al., [128] utilized machine learning to detect web application vulnerabilities. They used this methodology in Mitch. Mitch was the first machine-learning based solution to detect cross-site request forgeries.the detailed research review on machine learning to prevent web vulnerabilities shown in Table VI.

TABLE VI. MACHINE LEARNING EXISTING STUDIES

Research article	Language /Framework	Metrics / Feature	Year	Dataset	Classifier	ML Method	Web Vulnerabilities	Performance Parameters	Application
Neuhaus et al. [101]	vulture tool	14 components importing nsNodeUtils.h	2007	Mozilla with 134 vulnerabilities	SVM	A1	Security vulnerabilities	Precision, Recall	Mozilla Firefox
Wang Yanya et al. [103]	DSVRDC	67 series software Apache	2011	open-source web server software Apache httpd 2.2.8	Rd-entropy.	A2	Security vulnerabilities	Accuracy	C++ programming language
Yamaguchi et al. [104]	*	Extracting AST with a parser	2011	*	*	A2	*	*	C++ programming language
Wijayasekara et al. [105]	open bug database	Feature Vector	2012	Linux kernel vulnerabilities (Redhat Bugzilla)	Bayesian	A3	SQLi	Accuracy	hidden impact bugs
Nunan et al. [107]	Experimental study	obfuscation-based, doubtful patterns and html/JavaScript schemes	2012	15.366 websites XSSed database, dmoz.org and (2) 158.847	NB And SVM	A1	XSS	Detection rate, Accuracy rate and False alarm rate	HTML/JavaScript and PHP
Shar and Tan [110]	PHPMINER-1.	bytecode rewriting	2012	Java-based open source applications, Events, Classifieds, Roomba, PersonalBlog, and JGossip	*	A2	XSS, SQL	*	HTML/JavaScript and PHP
Soska and Christin [112]	complementary approach	structure of the file system, the structure of the webpage and the keyword	2014	PhishTank, search redirection attacks,	C4.5	A1	all	accuracy	all
Fabian et al. [113]	Code Property Graph	Extracting AST with a parser	2014	central vulnerability database by MITRE of CVE	*	A2	BO, Memory Mapping, Zero-byte Allocation	*	all
Howard et al. [25]	Psigene system Framework	SQL reserved words, SQLi signatures from the Bro, Snort IDS and the ModSecurity, web of WAF and SQLi reference documents	2014	the Exploit Database, PacketStorm Security, and the Open Source Vulnerability Database	Logistic regression	A2	SQL injection	Accuracy, Precision	PHP

Grieco et al. [115]	VDISCOVER method	N-gram analysis on the function call sequences	2016	bag-of-words, word2vec	Logistic Regression MLP	A2	vulnerabilities in operating systems	Accuracy	*
Medeiros et al. [116]	Method	Aggregate function, Numeric entry point, Complex query, Extract substring, String concatenation, Add char, Replace string, Error/exit, Remove whitespaces, Type checking, An entry point is set, Pattern control,	2016	Custome Dataset	ID3, C4.5/J48, RF, K-NN NB, MLP, SVM ,Logistic, Forest Tree, Bayes Net	*	SQLi, XSS,	Accuracy, Precession, Recall	PHP
Walden, Stuckman, and Scandariato [117]	Dataset Created	Software metrics, text mining	2014	Drupal, PHPMyAdmin, and Moodle with 223 web vulnerabilities	Random Forest	*	SQL injection, XSS, CSRF, and others	Accuracy, Precession, Recall	PHP
YUN et al.., [118]	VULPREDICTOR	Software metrics, text mining	2016	Drupal, PHPMyAdmin, and Moodle with 223 web vulnerabilities	Random Forest, Naïve Bayes, J48	*	SQL injection, XSS, CSRF and others	Accuracy, Precession, Recall	PHP
Abunadi, ibrahim, and mamdouh [119]	Proposed Method	Software metrics, text mining	2016	Drupal, PHPMyAdmin, and Moodle with 223 web vulnerabilities	RF, LR, SVM, J48, and NB	*	SQL injection, XSS, CSRF and others	Accuracy, Precession, Recall	PHP
Anbiya et al[120]	Proposed Method	PHP Tokens	2018	NVD	SVM, DT	A1	SQL injection, XSS,and others	Accuracy, Precession, Recall	PHP
Kronjee et al[121]	WIRECAML	All Features	2018	National Vulnerability Database and the SAMATE	probabilistic classifiers	A1	SQL injection, XSS,	Accuracy,	PHP
Smitha et al [122]	Comparative study ss	All Features	2019	HTTP CSIC 2010	SVM and LR	A1	QLI, XSS, LDAP, and Buffer overflow	Accuracy, Precession, Recall	PHP
Noman et al[123]	NMPREDICTOR	Software metrics, text mining	2019	Drupal, PHPMyAdmin, and Moodle with 223 web vulnerabilities	. J48, Naive Bayes and Random forest	A2	SQL injection, XSS, CSRF and others	Accuracy, Precession, Recall	PHP
Kudjo et al[124]	Model	Software metrics, text mining	2019	Drupal, PHPMyAdmin, and Moodle with 223 web vulnerabilities	RF, SVM, KNN, MLP, C4,5	A2	SQL injection, XSS, CSRF and others	Accuracy, Precession, Recall	PHP
Zhou et al [125]	Proposed Method	All Features	2020	Test cases	genetic algorithm	A3	SQL injection, XSS,	Comparative	PHP
Tang et al [126]	Model	eight Features	2020	Normal Dataset	MLP model	A1	SQL injection	accuracy	*
Williams et al [127]	framework	All Features	2020	*	*	A1 & A2	Vulnerabilties	*	*

* Missing in Paper
A1:Vulnerability Prediction Models based on Software Metrics: A2: Vulnerable Code Pattern Recognition
A3 Miscellaneous Approaches A4: n-grams extraction algorithms.

1) Discussion Machine learning is considered very different approach with a wide range of web applications. However, it can also use to find out web vulnerabilities in source code. It is a very important area in today's collaborative work environment to detect 0day web vulnerabilities and new approaches are always desirable including the current and existing once. Many researched have focused their studies to enhance the efficiency and precision of different techniques to detect web vulnerabilities. Support Vector Machines (SVM), J48, Artificial Neural Network, and many other classifications of techniques such as K-Nearest Neighbor, C5.0, Naïve Bayes, and linear regression are tested to train different datasets in order to detect web vulnerabilities. Furthermore, mostly researcher evaluate their result with machine learning parameters such as precision, recall, F1-score, and accuracy. From the existing, study noman et al. [127], Medeiros et al. [12] are useful approaches in Machine learning.

V. Conclusion

This study provides a comprehensive survey of existing methods in the research area of web applications vulnerabilities. We highlighted several open issues that still needs to be addressed. In this paper, we reviewed classification and detection of web vulnerabilities with different approaches like static analysis, dynamic analysis, hybrid analysis, combined three analyses for scanners and machine learning. We also reviewed various types of web vulnerabilities with different classification. The input validation vulnerabilities and improper session management and methods to perceive web vulnerabilities. There are lot of works that have been performed to cater to such issues. The best approach we identified to secure a web application is concluded such as for secure programming is Agrawal et al (2019), Kang and Park (2017), Zhu et al. (2014), in case of Static analysis is Nunes, P et al. (2019) Khalid et al. (2018) and NUNES et al (2015). Furthermore, in case of Dynamic analysis is Park et al. (2019), Kang and Park (2017) and Zhu et al. (2014), in case of Hybrid analysis is Le et al. 2019 and Stock et al. (2014), however, in case of machine learning is noman et al (2019), Medeiros et al. (2016).

References

[1] Pop, D. P., & Altar, A. (2014). Designing an MVC model for rapid web application development. Procedia Engineering, 69, 1172-1179.

[2] Deepa, G., & Thilagam, P. S. (2016). Securing web applications from injection and logic vulnerabilities: Approaches and challenges. Information and Software Technology, 74, 160-180.

[3] Awoleye, O. M., Ojuloge, B., & Ilori, M. O. (2014). Web application vulnerability assessment and policy direction towards a secure smart government. Government Information Quarterly, 31, S118-S125.

[4] Bozic, J., & Wotawa, F. (2015, August). PURITY: a Planning-based secURITY testing tool. In Software Quality, Reliability and Security-Companion (QRS-C), 2015 IEEE International Conference on (pp. 46-55). IEEE.

[5] Duchene, F., Rawat, S., Richier, J. L., & Groz, R. (2014, March). KameleonFuzz: evolutionary fuzzing for black-box XSS detection. In Proceedings of the 4th ACM conference on Data and application security and privacy (pp. 37-48). ACM.

[6] Medeiros, I., Neves, N. F., & Correia, M. (2014, April). Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. In Proceedings of the 23rd international conference on World Wide Web (pp. 63-74). ACM.

[7] Tsipenyuk, K., Chess, B., & McGraw, G. (2005). Seven pernicious kingdoms: A taxonomy of software security errors. IEEE Security & Privacy, 3(6), 81-84.

[8] Mitropoulos, D., Louridas, P., Polychronakis, M., & Keromytis, A. D. (2017). Defending Against Web Application Attacks: Approaches, Challenges and Implications. IEEE Transactions on Dependable and Secure Computing.

[9] Li, Xiaowei, and Yuan Xue. "A survey on server-side approaches to securing web applications." ACM Computing Surveys (CSUR) 46, no. 4 (2014): 54.

[10] Wedman, S., Tetmeyer, A., & Saiedian, H. (2013). An analytical study of web application session management mechanisms and HTTP session hijacking attacks. Information Security Journal: A Global Perspective, 22(2), 55-67.

[11] Shahriar, H., & Zulkernine, M. (2010, November). Client-side detection of cross-site request forgery attacks. In Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on (pp. 358-367). IEEE.

[12] Delgado, Nelly, Ann Q. Gates, and Steve Roach. "A taxonomy and catalog of runtime software-fault monitoring tools." IEEE Transactions on software Engineering 30, no. 12 (2004): 859-872.

[13] Igure, Vinay M., and Ronald D. Williams. "Taxonomies of attacks and vulnerabilities in computer systems." IEEE Communications Surveys & Tutorials 10, no. 1 (2008)

[14] Krsul, Ivan Victor. Software vulnerability analysis. West Lafayette, IN: Purdue University, 1998.

[15] Halfond, William G., Jeremy Viegas, and Alessandro Orso. "A classification of SQL-injection attacks and countermeasures." In Proceedings of the IEEE International Symposium on Secure Software Engineering, vol. 1, pp. 13-15. IEEE, 2006.

[16] Chandrashekhar, Roshni, Manoj Mardithaya, Santhi Thilagam, and Dipankar Saha. "SQL injection attack mechanisms and prevention techniques." In International Conference on Advanced Computing, Networking and Security, pp. 524-533. Springer, Berlin, Heidelberg, 2011.

[17] Garcia-Alfaro, Joaquin, and Guillermo Navarro-Arribas. "A survey on cross-site scripting attacks." arXiv preprint arXiv: 0905.4850 (2009).

[18] M. Cova, V. Felmetsger, G. Vigna, Vulnerability analysis of web-based applications, in: Test and Analysis of Web Services, Springer Berlin Heidelberg, 2007, pp. 363–394.

[19] NUNES, P., FONSECA, J. & VIEIRA, M. (2015). phpSAFE: A security analysis tool for OOP web application plugins. In Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks.

[20] Shahriar, Hossain, and Mohammad Zulkernine. "Taxonomy and classification of automatic monitoring of program security vulnerability exploitations." Journal of Systems and Software84, no. 2 (2011): 250-269.

[21] Hydara, Isatou, Abu Bakar Md Sultan, Hazura Zulzalil, and Novia Admodisastro. "Current state of research on cross-site scripting (XSS)–A systematic literature review." Information and Software Technology 58 (2015): 170-186.

[22] Buczak, Anna L., and Erhan Guven. "A survey of data mining and machine learning methods for cyber security intrusion detection." IEEE Communications Surveys & Tutorials 18, no. 2 (2016): 1153-1176.

[23] Ghaffarian, Seyed Mohammad, and Hamid Reza Shahriari. "Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey." ACM Computing Surveys (CSUR) 50, no. 4 (2017): 56.

[24] Shahriar, H., & Zulkernine, M. (2012, October). Information-theoretic detection of sql injection attacks. In High-Assurance Systems Engineering (HASE), 2012 IEEE 14th International Symposium on (pp. 40-47). IEEE.

[25] Howard, G. M., Gutierrez, C. N., Arshad, F. A., Bagchi, S., & Qi, Y. (2014, June). psigene: Webcrawling to generalize sql injection signatures. In Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on (pp. 45-56). IEEE.

[26] Juillerat, N. (2007, October). Enforcing code security in database web applications using libraries and object models. In Proceedings of the 2007 Symposium on Library-Centric Software Design (pp. 31-41). ACM.

[27] Johns, M., Beyerlein, C., Giesecke, R., & Posegga, J. (2010). Secure Code Generation for Web Applications. ESSoS, 5965, 96-113. Chicago

[28] Grabowski, R., Hofmann, M., & Li, K. (2011). Type-Based Enforcement of Secure Programming Guidelines-Code Injection Prevention at SAP. Formal Aspects in Security and Trust, 7140, 182-197.

[29] Chong, S., Vikram, K., & Myers, A. C. (2007, August). SIF: Enforcing Confidentiality and Integrity in Web Applications. In USENIX Security Symposium (pp. 1-16).

[30] Vikram, K., Prateek, A., & Livshits, B. (2009, November). Ripley: automatically securing web 2.0 applications through replicated execution. In Proceedings of the 16th ACM conference on Computer and communications security (pp. 173-186). ACM.

[31] Yip, A., Wang, X., Zeldovich, N., & Kaashoek, M. F. (2009, October). Improving application security with data flow assertions. In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles (pp. 291-304). ACM.

[32] Jia, L., Vaughan, J. A., Mazurak, K., Zhao, J., Zarko, L., Schorr, J., & Zdancewic, S. (2008, September). Aura: A programming language for authorization and audit. In ACM Sigplan Notices (Vol. 43, No. 9, pp. 27-38). ACM.

[33] Swamy, N., Corcoran, B. J., & Hicks, M. (2008, May). Fable: A language for enforcing user-defined security policies. In Security and Privacy, 2008. SP 2008. IEEE Symposium on (pp. 369-383). IEEE.

[34] Corcoran, B. J., Swamy, N., & Hicks, M. (2009, June). Cross-tier, label-based security enforcement for web applications. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (pp. 269-282). ACM.

[35] Swamy, N., Chen, J., & Chugh, R. (2010, March). Enforcing Stateful Authorization and Information Flow Policies in Fine. In ESOP (pp. 529-549).

[36] Krishnamurthy, A., Mettler, A., & Wagner, D. (2010, April). Fine-grained privilege separation for web applications. In Proceedings of the 19th international conference on World wide web (pp. 551-560). ACM.

[37] Zhu, J., Xie, J., Lipford, H. R., & Chu, B. (2014). Supporting secure programming in web applications through interactive static analysis. Journal of advanced research, 5(4), 449-462

[38] Kang, J., & Park, J. H. (2017). A secure-coding and vulnerability check system based on smart-fuzzing and exploit. Neurocomputing.

[39] Na Meng, Stefan Nagy, Danfeng Yao, Wenjie Zhuang, and Gustavo Arango-Argoty. Secure coding practices in java: Challenges and vulnerabilities. In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 372--383, 2018.

[40] Bangani, S., Futcher, L., & van Niekerk, J. (2019, July). An Approach to Teaching Secure Programming in the. NET Environment. In Annual Conference of the Southern African Computer Lecturers' Association (pp. 35-49). Springer, Cham.

[41] Agrawal, A., Alenezi, M., Kumar, R., & Khan, R. A. (2019). A source code perspective framework to produce secure web applications. Computer Fraud & Security, 2019(10), 11-18

[42] Jovanovic, N., Kruegel, C., & Kirda, E. (2006, May). Pixy: A static analysis tool for detecting web application vulnerabilities. In Security and Privacy, 2006 IEEE Symposium on (pp. 6-pp). IEEE.

[43] Chess, B., & McGraw, G. (2004). Static analysis for security. IEEE Security & Privacy, 2(6), 76-79.

[44] Durães, João, and Henrique Madeira. "A methodology for the automated identification of buffer overflow vulnerabilities in executable software without source-code." In Latin-American Symposium on Dependable Computing, pp. 20-34. Springer, Berlin, Heidelberg, 2005.

[45] Huang, Y. W., Yu, F., Hang, C., Tsai, C. H., Lee, D. T., & Kuo, S. Y. (2004, May). Securing web application code by static analysis and runtime protection. In Proceedings of the 13th international conference on World Wide Web (pp. 40-52). ACM.

[46] Wassermann, G., & Su, Z. (2008, May). Static detection of cross-site scripting vulnerabilities. In Proceedings of the 30th international conference on Software engineering (pp. 171-180). ACM.

[47] Minamide, Y. (2005, May). Static approximation of dynamically generated web pages. In Proceedings of the 14th international conference on World Wide Web (pp. 432-441). ACM.

[48] Xie, Y., & Aiken, A. (2006, July). Static Detection of Security Vulnerabilities in Scripting Languages. In USENIX Security Symposium (Vol. 15, pp. 179-192).

[49] Halfond, W. G., & Orso, A. (2005, November). AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering (pp. 174-183). ACM.

[50] Thomas, S., & Williams, L. (2007, May). Using automated fix generation to secure SQL statements. In Proceedings of the Third International Workshop on Software Engineering for Secure Systems (p. 9). IEEE Computer Society.

[51] SON, S. & SHMATIKOV, V. (2011). SAFERPHP: Finding semantic vulnerabilities in PHP applications. In Proceedings of the ACM SIGPLAN 6th Workshop on Programming Languages and Analysis for Security.

[52] J. Fonseca, N. Seixas, M. Vieira, H. Madeira, Analysis of field data on web security vulnerabilities, IEEE Transactions on Dependable and Secure Computing 11 (2) (2014) 89–100.

[53] Shahriar, H., & Zulkernine, M. (2009, May). Mutec: Mutation-based testing of cross site scripting. In Proceedings of the 2009 ICSE Workshop on Software Engineering for Secure Systems (pp. 47-53). IEEE Computer Society.

[54] Shar, Lwin Khin, and Hee Beng Kuan Tan. "Automated removal of cross site scripting vulnerabilities in web applications." Information and Software Technology 54, no. 5 (2012): 467-478.

[55] Scholte, T., Robertson, W., Balzarotti, D., & Kirda, E. (2012, July). Preventing input validation vulnerabilities in web applications through automated type analysis. In Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual (pp. 233-243). IEEE.

[56] Zheng, Y., & Zhang, X. (2013, May). Path sensitive static analysis of web applications for remote code execution vulnerability detection. In Proceedings of the 2013 International Conference on Software Engineering (pp. 652-661). IEEE Press.

[57] Doupé, A., Cui, W., Jakubowski, M. H., Peinado, M., Kruegel, C., & Vigna, G. (2013, November). deDacota: toward preventing server-side XSS via automatic code and data separation. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 1205-1216). ACM.

[58] Amira, Abdelouahab, Abdelraouf Ouadjaout, Abdelouahid Derhab, and Nadjib Badache. "Sound and Static Analysis of Session Fixation Vulnerabilities in PHP Web Applications." In Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, pp. 139-141. ACM, 2017.

[59] Muhammad Noman khalid, Muhammad Iqbal, Muhammad Talha Alam, Vishal Jain, Hira Mirza and Kamran Rasheed, “Web Unique Method (WUM): An Open Source Blackbox Scanner for Detecting Web Vulnerabilities” International Journal of Advanced Computer Science and Applications(IJACSA), 8(12), 2017.

[60] Viega, John, Jon-Thomas Bloch, Yoshi Kohno, and Gary McGraw. "ITS4: A static vulnerability scanner for C and C++ code." In Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference, pp. 257-267. IEEE, 2000.

[61] Deepa, G., Thilagam, P. S., Praseed, A., & Pais, A. R. (2018). DetLogic: A black-box approach for detecting logic vulnerabilities in web applications. Journal of Network and Computer Applications, 109, 89-109.

[62] Nunes, P., Medeiros, I., Fonseca, J., Neves, N., Correia, M., & Vieira, M. (2019). An empirical study on combining diverse static analysis tools for web security vulnerabilities based on development scenarios. Computing, 101(2), 161-185.

[63] NUNES, P., FONSECA, J. & VIEIRA, M. (2015). phpSAFE: A security analysis tool for OOP web application plugins. In Proceedings of the

45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks

[64] Long, H. V., Tuan, T. A., Taniar, D., Can, N. V., Hue, H. M., & Son, N. T. K. (2020). An efficient algorithm and tool for detecting dangerous website vulnerabilities. International Journal of Web and Grid Services, 16(1), 81-104.

[65] Aliero, M. S., Ghani, I., Qureshi, K. N., & Rohani, M. F. A. (2020). An algorithm for detecting SQL injection vulnerability using black-box testing. Journal of Ambient Intelligence and Humanized Computing, 11(1), 249-266.

[66] EVRON, G. & RATHAUS, N. (2007). Open Source Fuzzing Tools. Elsevier Inc., 1st edn.

[67] SUTTON, M., GREENE, A. & AMINI, P. (2007). Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley, 1st edn.

[68] Nguyen-Tuong, A., Guarnieri, S., Greene, D., Shirley, J., & Evans, D. (2005). Automatically hardening web applications using precise tainting. Security and Privacy in the Age of Ubiquitous Computing, 295-307.

[69] Haldar, V., Chandra, D., & Franz, M. (2005, December). Dynamic taint propagation for Java. In Computer Security Applications Conference, 21st Annual (pp. 9-pp). IEEE.

[70] Jimenez, W., Mammar, A., & Cavalli, A. (2009). Software vulnerabilities, prevention and detection methods: A review1. Security in Model-Driven Architecture, 6.

[71] Miller, B. P., Fredriksen, L., & So, B. (1990). An empirical study of the reliability of UNIX utilities. Communications of the ACM, 33(12), 32-44.

[72] BRADSHAW, S. (2010). An introduction to fuzzing: Using fuzzers (spike) to find vulnerabilities. http://resources.infosecinstitute.com/intro-to-fuzzing/.bufferoverflow vulnerabilities in executable software without source-code. In Proceedings of the 2nd Latin-American Conference on Dependable Computing, 20–34.

[73] Huang, Y. W., Tsai, C. H., Lin, T. P., Huang, S. K., Lee, D. T., & Kuo, S. Y. (2005). A testing framework for Web application security assessment. Computer Networks, 48(5), 739-761.

[74] Kals, S., Kirda, E., Kruegel, C., & Jovanovic, N. (2006, May). Secubat: a web vulnerability scanner. In Proceedings of the 15th international conference on World Wide Web (pp. 247-256). ACM.

[75] DUCHÈNE, F., RAWAT, S., RICHIER, J. & GROZ, R. (2014). Kameleonfuzz: Evolutionary fuzzing for black-box XSS detection. In Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, 37–48.

[76] Antunes, N., & Vieira, M. (2010, July). Benchmarking vulnerability detection tools for web services. In Web Services (ICWS), 2010 IEEE International Conference on (pp. 203-210). IEEE.

[77] Ciampa, A., Visaggio, C. A., & Di Penta, M. (2010, May). A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications. In Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems (pp. 43-49). ACM.

[78] Stock, B., Lekies, S., & Johns, M. (2013). 25 Million Flows Later-Large-scale Detection of DOM-based XSS. 20th CCS. ACM.

[79] HALLER, I., SLOWINSKA, A., NEUGSCHWANDTNER, M. & BOS, H. (2013). Dowsing for overflows: A guided fuzzer to find buffer boundary violations. In Proceedings of the 22nd USENIX Security Symposium, 49–64.

[80] Dahse, J., & Holz, T. (2014, August). Static Detection of Second-Order Vulnerabilities in Web Applications. In USENIX Security Symposium (pp. 989-1003).

[81] Weissbacher, M., Robertson, W. K., Kirda, E., Kruegel, C., & Vigna, G. (2015, August). ZigZag: Automatically Hardening Web Applications against Client-side Validation Vulnerabilities. In USENIX Security Symposium (pp. 737-752).

[82] RanWang, GuangquanXu, XianjiaoZeng, XiaohongLi, ZhiyongFeng. (2018). TT-XSS: A novel taint tracking based dynamic detection framework for DOM Cross-Site Scripting.Journal of Parallel and Distributed Computing Volume 118, Part 1, August 2018, Pages 100-106

[83] Park, J., Choo, Y., & Lee, J. (2019). A hybrid vulnerability analysis tool using a risk evaluation technique. Wireless Personal Communications, 105(2), 443-459.

[84] CHIPOUNOV, V., KUZNETSOV, V. & CANDEA, G. (2011). S2e: A platform for in-vivo multi-path analysis of software systems. In Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems, 265–278.

[85] Falana, O. J., Ebo, I. O., Tinubu, C. O., Adejimi, O. A., & Ntuk, A. (2020, March). Detection of Cross-Site Scripting Attacks using Dynamic Analysis and Fuzzy Inference System. In 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS) (pp. 1-6). IEEE.

[86] Di Lucca, G. A., Fasolino, A. R., Mastoianni, M., & Tramontana, P. (2004, September). Identifying cross site scripting vulnerabilities in web applications. In Web Site Evolution, Sixth IEEE International Workshop on (WSE'04) (pp. 71-80). IEEE.

[87] Balzarotti, D., Cova, M., Felmetsger, V., Jovanovic, N., Kirda, E., Kruegel, C., & Vigna, G. (2008, May). Saner: Composing static and dynamic analysis to validate sanitization in web applications. In Security and Privacy, 2008. SP 2008. IEEE Symposium on (pp. 387-401). IEEE.

[88] Livshits, V. B., & Lam, M. S. (2005, August). Finding Security Vulnerabilities in Java Applications with Static Analysis. In USENIX Security Symposium (Vol. 14, pp. 18-18).

[89] Lam, M. S., Martin, M., Livshits, B., & Whaley, J. (2008, January). Securing web applications with static and dynamic information flow tracking. In Proceedings of the 2008 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation (pp. 3-12). ACM.

[90] Van Acker, S., Nikiforakis, N., Desmet, L., Joosen, W., & Piessens, F. (2012, May). FlashOver: Automated discovery of cross-site scripting vulnerabilities in rich internet applications. In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (pp. 12-13). ACM.

[91] Lee, I., Jeong, S., Yeo, S., & Moon, J. (2012). A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling, 55(1), 58-68.

[92] Lee, T., Won, G., Cho, S., Park, N., & Won, D. (2012). Experimentation and Validation of Web Application's Vulnerability Using Security Testing Method. In Computer Science and its Applications (pp. 723-731). Springer, Dordrecht.

[93] Vogt, Philipp, Florian Nentwich, Nenad Jovanovic, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. "Cross Site Scripting Prevention with Dynamic Data Tainting and Static Analysis." In NDSS, vol. 2007, p. 12. 2007.

[94] Stock, Ben, and Martin Johns. "Protecting users against XSS-based password manager abuse." In Proceedings of the 9th ACM symposium on Information, computer and communications security, pp. 183-194. ACM, 2014.

[95] He, X., Xu, L., & Cha, C. (2018, December). Malicious JavaScript Code Detection Based on Hybrid Analysis. In 2018 25th Asia-Pacific Software Engineering Conference (APSEC) (pp. 365-374). IEEE.

[96] Le, V. G., Nguyen, H. T., Pham, D. P., Phung, V. O., & Nguyen, N. H. (2019). GuruWS: A Hybrid Platform for Detecting Malicious Web Shells and Web Application Vulnerabilities. In Transactions on Computational Collective Intelligence XXXII (pp. 184-208). Springer, Berlin, Heidelberg.

[97] Hladka, B., & Holub, M. (2015). A Gentle Introduction to Machine Learning for Natural Language Processing: How to Start in 16 Practical Steps. Language and Linguistics Compass, 9(2), 55-76. Chicago

[98] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

[99] Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[100] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 15.

[101] Neuhaus, S., Zimmermann, T., Holler, C., & Zeller, A. (2007, October). Predicting vulnerable software components. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 529-540). ACM.

[102]SHIN, Y., MENEELY, A., WILLIAMS, L. & OSBORNE, J.A. (2011). Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. IEEE Transactions on Software Engineering, 37, 772–787.

[103]Wang, Y., Wang, Y., & Ren, J. (2011). Software Vulnerabilities Detection Using Rapid Density-based Clustering. JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE, 8(14), 3295-3302.

[104]Yamaguchi, F., Lindner, F., & Rieck, K. (2011, August). Vulnerability extrapolation: assisted discovery of vulnerabilities using machine learning. In Proceedings of the 5th USENIX conference on Offensive technologies (pp. 13-13). USENIX Association.

[105]Wijayasekara, D., Manic, M., Wright, J. L., & McQueen, M. (2012, June). Mining bug databases for unidentified software vulnerabilities. In Human System Interactions (HSI), 2012 5th International Conference on (pp. 89-96). IEEE.

[106]Nunan, A. E., Souto, E., dos Santos, E. M., & Feitosa, E. (2012, July). Automatic classification of cross-site scripting in web pages using document-based and URL-based features. In Computers and Communications (ISCC), 2012 IEEE Symposium on (pp. 000702-000707). IEEE.

[107]Sultana, A., Hamou-Lhadj, A., & Couture, M. (2012, June). An improved hidden markov model for anomaly detection using frequent common patterns. In Communications (ICC), 2012 IEEE International Conference on (pp. 1113-1117). IEEE.

[108]Khosronejad, M., Sharififar, E., Torshizi, H. A., & Jalali, M. (2013). Developing a hybrid method of Hidden Markov Models and C5. 0 as a Intrusion Detection System. International Journal of Database Theory and Application, 6(5), 165-174.

[109]Bhole, A. T., & Patil, A. I. (2014). Intrusion Detection with Hidden Markov Model and WEKA Tool. International Journal of Computer Applications, 85(13).

[110]Shar, L. K., & Tan, H. B. K. (2012, June). Mining input sanitization patterns for predicting SQL injection and cross site scripting vulnerabilities. In Proceedings of the 34th International Conference on Software Engineering (pp. 1293-1296). IEEE Press.

[111]Shar, L. K., Briand, L. C., & Tan, H. B. K. (2015). Web application vulnerability prediction using hybrid program analysis and machine learning. IEEE Transactions on Dependable and Secure Computing, 12(6), 688-707.

[112]Soska, K., & Christin, N. (2014, August). Automatically Detecting Vulnerable Websites Before They Turn Malicious. In USENIX Security Symposium (pp. 625-640).

[113]Yamaguchi, Fabian. Golde, N., Arp, D., & Rieck, K. (2014, May). Modeling and discovering vulnerabilities with code property graphs. In Security and Privacy (SP), 2014 IEEE Symposium on (pp. 590-604). IEEE.

[114]Singh, N., Dayal, M., Raw, R. S., & Kumar, S. (2016, March). SQL injection: Types, methodology, attack queries and prevention. In Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on (pp. 2872-2876). IEEE.

[115]Grieco, G., Grinblat, G. L., Uzal, L., Rawat, S., Feist, J., & Mounier, L. (2016, March). Toward large-scale vulnerability discovery using Machine Learning. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy (pp. 85-96). ACM.

[116]Medeiros, I., Neves, N., & Correia, M. (2016). Detecting and removing web application vulnerabilities with static analysis and data mining. IEEE Transactions on Reliability, 65(1), 54-69.

[117]Walden, James, Jeff Stuckman, and Riccardo Scandariato. "Predicting vulnerable components: Software metrics vs text mining." In Software Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium on, pp. 23-33. IEEE, 2014.

[118]Zhang, Yun, David Lo, Xin Xia, Bowen Xu, Jianling Sun, and Shanping Li. "Combining software metrics and text features for vulnerable file prediction." In Engineering of Complex Computer Systems (ICECCS), 2015 20th International Conference on, pp. 40-49. IEEE, 2015.

[119]Abunadi, Ibrahim, and Mamdouh Alenezi. "An Empirical Investigation of Security Vulnerabilities within Web Applications." J. UCS 22, no. 4 (2016): 537-551.

[120]Anbiya, D. R., Purwarianti, A., & Asnar, Y. (2018, November). Vulnerability Detection in PHP Web Application Using Lexical Analysis Approach with Machine Learning. In 2018 5th International Conference on Data and Software Engineering (ICoDSE) (pp. 1-6). IEEE.

[121]Kronjee, J., Hommersom, A., & Vranken, H. (2018, August). Discovering software vulnerabilities using data-flow analysis and machine learning. In Proceedings of the 13th International Conference on Availability, Reliability and Security (pp. 1-10).

[122]Smitha, R., Hareesha, K. S., & Kundapur, P. P. (2019). A Machine Learning Approach for Web Intrusion Detection: MAMLS Perspective. In Soft Computing and Signal Processing (pp. 119-133). Springer, Singapore.

[123]Khalid, M. N., Farooq, H., Iqbal, M., Alam, M. T., & Rasheed, K. (2018, October). Predicting Web vulnerabilities in Web applications based on machine learning. In International Conference on Intelligent Technologies and Applications (pp. 473-484). Springer, Singapore.

[124]Kudjo, P. K., Aformaley, S. B., Mensah, S., & Chen, J. (2019, July). The Significant Effect of Parameter Tuning on Software Vulnerability Prediction Models. In 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 526-527). IEEE.

[125]Zhou, X., & Wu, B. (2020, June). Web Application Vulnerability Fuzzing Based On Improved Genetic Algorithm. In 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (Vol. 1, pp. 977-981). IEEE.

[126]Tang, P., Qiu, W., Huang, Z., Lian, H., & Liu, G. (2020). Detection of SQL injection based on artificial neural network. Knowledge-Based Systems, 190, 105528.

[127] Williams, M. A., Barranco, R. C., Naim, S. M., Dey, S., Hossain, M. S., & Akbar, M. (2020). A vulnerability analysis and prediction framework. Computers & Security, 92, 101751.

[128]Calzavara, S., Conti, M., Focardi, R., Rabitti, A., & Tolomei, G. (2020). Machine Learning for Web Vulnerability Detection: The Case of Cross-Site Request Forgery. IEEE Security & Privacy, 18(3), 8-16.