Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 4, 2026

EXPLORING SOFTWARE VULNERABILITIES AND CYBER THREATS LANDSCAPE USING MACHINE LEARNING: A COMPREHENSIVE REVIEW

Imdad Ali Shah[*1], Sorath Mahar[2], Ali Muhammad[3], Saira Arbab[4]

[*1,2]*FEST, Iqra University, Karachi, Pakistan*

[*1]shahsyedimdadali@gmail.com, [*1]imdad.ali@iqra.edu.pk, [2]drsorathmajid@gmail.com, [2]sorath.mahar@iqra.edu.pk,[3] ali.muhammad@iqra.edu.pk, [3]ali.12msit06@gmail.com, sairarbab@iqra.edu.pk[4], sairarbab@iqra.edu.pk[4]

DOI: https://doi.org/10.5281/zenodo.19850544

Keywords
Software Vulnerabilities, ML Models, Web-Based Apps, and Navigating Threat Landscape.

Article History
Received: 04 March 2026
Accepted: 11 April 2026
Published: 28 April 2026

Corresponding Author: *
Imdad Ali Shah

Abstract

Software vulnerabilities become higher risks for web-based applications, specifically, these vulnerabilities are multiplying several times, with the new era of generative AI (Genai). These challenges can be better addressed with the help of AI. Hence, this comprehensive review will provide in-depth studies, approaches, and mechanisms where Machine Learning (ML) can be a tool/way to handle these recent vulnerability challenges in a better way. According to the Bureau's analysis, 12.5$ billion in 2023 and more than triple in 2019 were recorded in cybercrime complaints. The loss grew significantly in 2024, which shows that data breaches are increasing, which is impacting the various technological software sectors, where web-apps are playing the main role. One of the examples related to the healthcare Industry's 54.4% increase in breach costs, approximately 10.94$ billion in 2023 from 2020. Further, in 2023, the number of cyberattacks that used credentials that had been stolen or compromised increased by 72%. In 2023, manufacturing accounted for more than 26% of attacks, making it the most affected industry. This research mainly common vulnerabilities. including identification and authentication failures, and software and data integrity failures. Using ML to detect, identify, and suggest appropriate solutions/models for the stated vulnerability issues could be the best possible solution based on the literature. This research considered a thorough literature review and found that Machine learning based approaches for the vulnerabilities detection is far better compared to the existing approaches for web apps. Further, this in-depth research found that the SQL injection attack group is severe for web apps. We also consider the manual, static, dynamic and hybrid approaches as well. This research provides the research challenges and opportunities related to web app vulnerabilities in detail.

1. Introduction

As software systems become increasingly integral to critical infrastructure and everyday operations, the threat landscape surrounding them continues to evolve in both scale and sophistication. Traditional methods of identifying and mitigating software vulnerabilities often fall short in the face of rapidly changing attack vectors and complex codebases. The growth of software systems is increasing in significance for the critical infrastructure and daily operations. The increasing importance of software systems in critical infrastructure. While the risk and threats

landscape remains near them, it continues to change both in terms of size and complexity[1, 2]. The quickly changing attack vectors and complex codebases make it difficult to identify, classify and mitigate through traditional methods.

One of the popular private banks in Malaysia also urged customers to remain vigilant and take steps to protect their personal information. It reminded customers to keep their User ID, password, and personal details more secure, and to be wary of potential threats such as malware, phishing sites, and fraudulent messages or phone calls requesting credentials[3, 4]. Between Friday, October 29, and Saturday, October 30, 2021, one of the popular private banks of Pakistan was the subject of one of the most concerning cyberattacks. The bank's backend systems, servers connecting the bank's branches, and backend infrastructure in charge of the ATM network and mobile banking app were all hit by the cyberattack[5, 6]. The impacted systems were immediately isolated so that other customers' financial information was not jeopardized, according to the Express Tribune.

In 2024, cybersecurity risks, including cyberattacks, are a significant global concern, particularly in the financial sector. The World Bank, along with other organizations, is actively working to enhance cyber resilience, especially in developing countries. The World Economic Forum's Global Risks Report 2024 identified cyber risks, such as malware and misinformation, as major threats to supply chains, financial stability, and democratic systems. Figure 1 Top 10 web application security risks by OWASP.

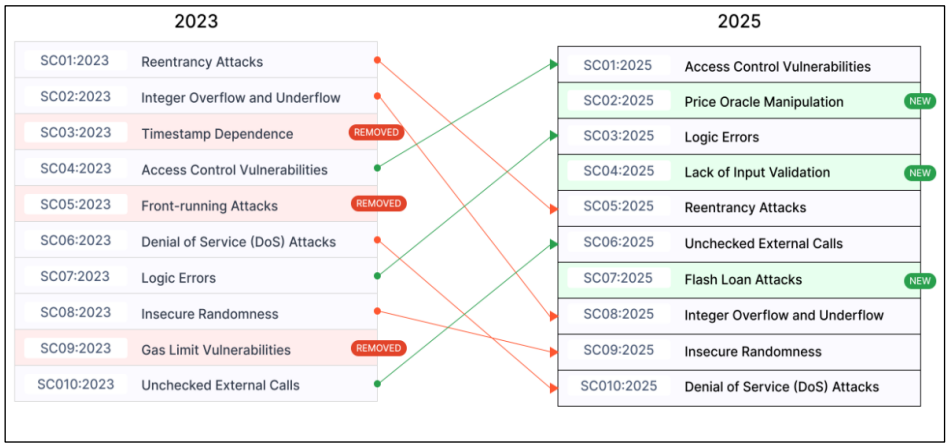

FIGURE 1 TOP 10 WEB APPLICATION SECURITY RISKS BY OWASP [7]

The core contribution of this research is as follows:

- This comprehensive review covers the different vulnerabilities detection, mitigation methods using various approaches such as, Manual, Static, Dynamic, and Hybrid.

- This research provides an analysis of different presented vulnerability approaches/ methods and identifies which approach is better under which conditions.
- This research provides comprehensive possible solutions for vulnerabilities using Machine Learning for current era vulnerabilities challenges, specifically for web apps.
- Further, we provide various challenges and opportunities related the web-app vulnerabilities and their solutions.

Software vulnerabilities significantly compromise computer systems and IT infrastructure security. However, detecting and mitigation methods for software vulnerabilities depend largely on rule-based static analysis, signature-based and manual code review. These approaches and methods are helpful for knowing threats, defined patterns, and struggling with growing, adaptability and detection of zero-day vulnerabilities. Conversely, ML-based techniques give dynamic capabilities through learn a large dataset of code, historical weaknesses. Additionally, the ML takes

automated and continuous monitoring, significantly reducing response time. The ML-based systems are facing challenges, such as data quality vulnerabilities to counterattacks. Conventional techniques also give basic reliability and explainability. Still, challenges remain in software security when Machine Learning advances, such as a lack of high-quality and labelled datasets, which are impacting the training and accuracy in ML models[8, 9]. The exploitation of Machine Learning makes it further complex due to the dynamic and hostile character of Cyber threats. Furthermore, advancements and changes in the software system, it to maintain the Machine Learning Models. In view of these identification issues and challenges in vulnerability classification, detection and mitigation and threat landscape[10, 11]. There are various ways to identify the vulnerabilities, such as manual, static, and dynamic or runtime methods. Static approaches, however, cannot identify vulnerabilities that only manifest during program execution, perhaps because of inputs or runtime circumstances, according to the corresponding [12]run an executable with inputs to observe how it behaves during runtime. Dynamic analysis needs certain input data as test cases to replicate how the program would behave with different inputs. Nonetheless, these solutions frequently suffer from reduced code coverage and may take extensive amounts of time and money to analyse all inputs, especially in big and complex programs. [13, 14] through leveraging both static and dynamic methodologies, hybrid analysis improves the accuracy of static and dynamic analysis and depth of analysis while reducing its drawbacks. In this regard, Hence, hybrid analysis may employ static analysis to find possible areas of code defects and dynamic analysis to confirm the existence of risks under different runtime scenarios. All three methods, however, depend on a small number of patterns derived from known vulnerabilities. To forecast code sections that might have weaknesses, static analysis uses an expert-defined collection of features, security rules, or execution routes; dynamic analysis uses data and experience from vulnerability tests already conducted; and hybrid research uses static rules[15, 16]. The growing number of flaws and identified CVEs continues to be a major concern, even with the advent of numerous vulnerability detection techniques in recent years.

Paper organization in pictorial form:

1. In the first section, we collected data from the different sources through the keywords Software Detection, Figure 8. Second section, we fixed the title and made the abstract, introduction, and Literature Review and focused on current SQL injection attacks, and the software detection accuracy ML model of error-based and union-based detection.
2. In the third section, we focused on research methodology and studied the related papers Figure 9.
3. In the fourth section, we present a critical review and comparative studies of the ML models and SQL injection attacks Figure 12. Finally, we present the conclusion of our research.

2. Literature Review

This section provides a comprehensive background for the studies of software vulnerability detection classification and identification, ML-based. Throughout the past decade, an increasing number of researchers have been highlighting leveraging ML to identify the complex challenges of software vulnerabilities and enhancing the scope of cybersecurity. Early research focused on the static and dynamic analysis techniques, while current developments have investigated the use of supervised, unsupervised techniques. This is a comprehensive review that critically investigates existing approaches, including Manual code analysis, Static code analysis, [17, 18]Dynamic and hybrid code analysis, and this comprehensive analysis/review will provide the challenges and opportunities for current era web-based applications using AI in general and ML in particular. It also provides an overview of research trends and identifies the gaps in current knowledge. Furthermore, sets the basic for knowing how ML can be successfully applied to detect, classify and mitigate software security. [19, 20] it begins by defining software vulnerabilities precisely and clearly, referencing reliable sources and bringing disparate

viewpoints in the field of cybersecurity together. [21] at the field of machine learning-based static vulnerability identification techniques, grouping them into three primary categories: software metrics-based techniques, vulnerable code-based techniques, and anomalous behaviour-based techniques. Web application architecture refers to the framework that defines the interactions between components in a web application.[22-24] It typically includes a client (browser), a web server, an application server, and a database. The process begins when a user sends a request through the browser, which is received by the web server. The request is then forwarded to the application server, where the business logic is processed. If needed, the application server queries the database and retrieves data. Finally, the response is sent back through the web server to the client, displaying the requested information. This layered approach ensures scalability, maintainability, and secure data handling.

However, SSDLC usage is still uneven, even though it is acknowledged as an industry standard. The continued existence of security flaws is highlighted by the growing number of vulnerabilities revealed in 2023, roughly 22,000, of which 15% are classified as extremely critical [25-27]. These impairments are caused by a few variables: First, a lot of businesses are unaware of SSDLC procedures and steer clear of them because they are worried about possible production delays or use security controls based on obscurity, which, although good at identifying known vulnerabilities, frequently generates a large number of false positives because it uses out-of-date databases or is unable to identify complex patterns like segments of code that are obfuscated or encrypted; and third, SSDLC assumes that DAST is used to supplement SAST by examining code as it runs[28-30]. Nevertheless, DAST usually requires more human reviews and lacks the necessary granularity at the source code level. Figure 2 SQLi attacker process.

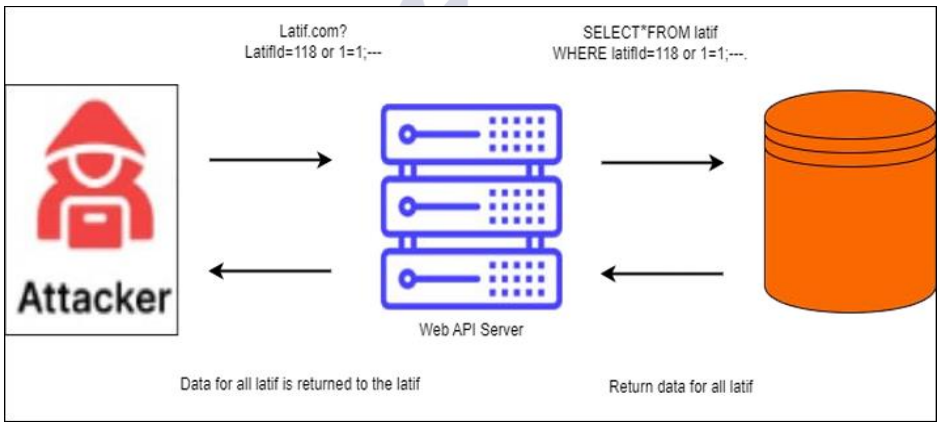

FIGURE 2 SQLI ATTACKER PROCESS [31]

Both SAST and DAST procedures may be optimised by including DevSecOps methods into SSDLC pipelines in conjunction with Continuous Integration (CI) and Continuous Delivery/Deployment (CD) pipelines[32, 33]. The intricacy of tool configuration, false-positive handling, high ongoing maintenance costs, the high fees charged by tool vendors, and the opaqueness of detection and remediation processes are just a few of the noteworthy obstacles that impede this integration. Figure 3 SQLi types.

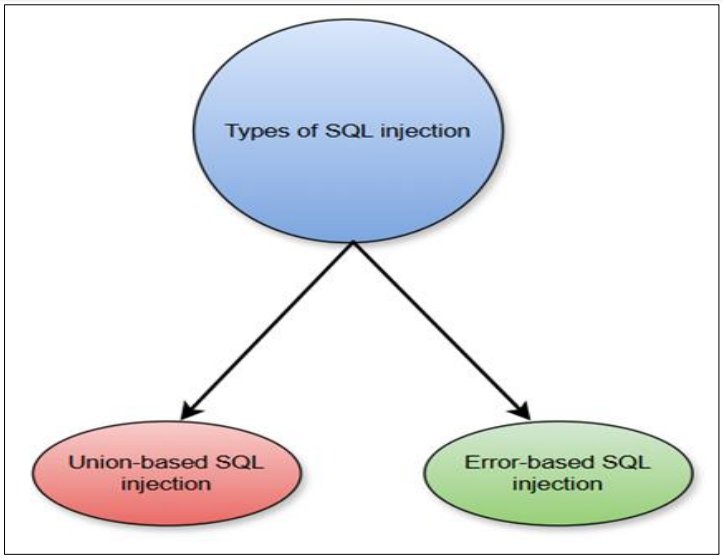

FIGURE 3 F SQLI TYPES

Error-based SQL Injection (SQLi) is a technique where attackers exploit database error messages to gain unauthorized access to data by injecting malicious SQL queries. This method relies on applications displaying detailed error messages that inadvertently reveal database structure or contents. Key challenges include inconsistent error handling across databases, insufficient input validation, poor security configurations, and developers' lack of awareness about secure coding practices [34-36]. Union-based SQL Injection (SQLi) is a technique where attackers use the SQL UNION operator to combine results from a malicious query with legitimate queries, allowing them to extract data from other database tables. This attack requires knowledge of the number and type of columns returned by the original query. Challenges include identifying the correct column count,[37, 38] bypassing input filters, and dealing with database-specific syntax. VulDeeLocator expanded VulDeePecker by using RNN techniques and Natural Language Processing (NLP) to identify vulnerabilities in C code, building on RNN-based approaches. To record data dependencies and control flows between functions and files, it used Static Single Assignment (SSA) representations, segmentation, and ASTs[39-41]. Using roughly 29,000 samples, the processed data was incorporated into a Bi-RNN with LSTM layers, yielding an accuracy of 98.8%.

A study [42, 43] Those that used Generative Pre-trained Transformer (GPT) APIs showed improvements in LLMs. Four crucial Common Weakness Enumerations (CWEs) were examined in the study's analysis of the top 25 dangerous C++ code examples from SANS: buffer overflow, risky functions, integer overflow, and insecure implementations

Several studies have examined the possibility of Machine learning for vulnerability detection, considering the software security detection community's growing interest in machine learning techniques, particularly the development of Machine learning techniques. Examined the SARD and NVD datasets to compare various Machine learning techniques for vulnerability identification [44, 45]. The findings demonstrated that the greatest accuracy of detection was obtained with code representation that included data and control dependencies using a bi-directional recurrent neural network. [46, 47] To compare with CNN and RNN algorithms, they selected several popular traditional machine learning techniques, including Random Forest, Gradient Boosting Decision Tree, and Support Vector Machine, demonstrating that, in comparison to conventional machine approaches, deep learning techniques are more effective in identifying software vulnerabilities [48, 49]. These methods are superior, yet there are still issues that require attention. Figure 2 The process of SQL Injection.

3. Software of Vulnerability Detection Methods

Techniques for classification, detection and mitigation of software vulnerabilities are crucial for locating and reducing security threats in software systems. These approaches can be grouped according to what they employ, the stage of software lifecycles at which they are used, the kind of vulnerabilities they target, and the detection methodology [50-52]. Curiosity, personal interest, and the desire to find a solution are some of the factors that may drive researchers to investigate a given subject. Table I shows machine learning models.

Table I Shows machine learning models

Citation	Problem	Proposed approached	Vulnerabilities	Dataset
[53]	Conventional methods	DL	Overflowing buffers (OB)	NVD, SARD
[54]	No possible danger was discovered.	DL	Injection of SQL	SARD, NVD
[55, 56]	Traditional techniques for security analysis	DL	OB	NVD, SARD
[57, 58]	Finding and taking advantage of security flaws.	DL	OB	21 Binary
[59, 60]	Security Holes	PM	External scripting	Projects
[61, 62]	manual analysis.	Hybrid analysis	OB	Wsmp3d, Kristen
[63, 64]	The current situation necessitates human assistance for classification.	Supervised learning	SQL of injection	CVSS

3.1 Manual Analysis

Manual analysis is the most widely used method in the field of software vulnerability detection. The analysis uses a human expert to test and debug software throughout the software development phase to find security flaws. A large-scale software development project would be labour-intensive, time-consuming, and prone to human error with this approach. The problem of manual analysis attempted to be resolved as early as. Since then, other articles have addressed the problem of manual analysis using a variety of techniques. It could take more time and requires a lot of work to examine the current codes[65, 66]. Point out that manual analysis requires a significant amount of time and resources. Although different security experts have differing levels of knowledge about software security flaws, manual analysis also raises questions about the efficacy and caliber of detection. Therefore, manual analysis is often seen as an unreliable method for discovering software vulnerabilities in large-scale projects that need thorough testing and inspection[67]. Conventional approaches are another popular field of research for software vulnerability identification. Figure 4 Manual Analysis tools.

FIGURE 4 MANUAL ANALYSIS TOOLS

A structured approach to understanding and classifying the different methods used to find security flaws in software systems is provided by the taxonomy of software vulnerability classification and detection methods. By classifying these methods according to their identification approach, the type of stage of the software lifecycle, the type of vulnerabilities targeted, and the techniques used, organizations can more effectively choose and implement the best methods to secure their software[68, 69]. To tackle complicated issues from several perspectives. Manual analysis of software defects is in Table II.

Table II Manual analysis of software defects

Citation	Problem	Proposed approached	Vulnerabilities	Dataset
[70-72]	Conventional methods	DL	Overflowing buffers (OB)	NVD, SARD
[73, 74]	Software Vulnerabilities	DL	Injection of SQL	SARD, NVD
[75]	Traditional techniques for security analysis	DL	OB	NVD, SARD
[76, 77]	Finding and taking advantage of security flaws.	DL	OB	21 Binary
[78, 79]	Security Holes	PM	External scripting	Projects
[80, 81]	manual analysis.	Hybrid analysis	OB	lighted, Wsmp3d, Kristen
[82, 83]	The current situation necessitates human assistance for classification.	Supervised learning	SQL injection	CVSS

3.2 Static Code Analysis (SCA)

The technique of looking at source code without running it to find possible mistakes, weaknesses, or departures from coding standards is known as static code analysis. During development, specialised tools typically carry out this analysis to make sure the code is clear, safe, and maintainable. Early in the software development lifecycle, it assists developers in identifying problems such as syntax errors, memory leaks[14, 84]. Teams can enhance code quality, minimise errors, and preserve adherence to industry standards.

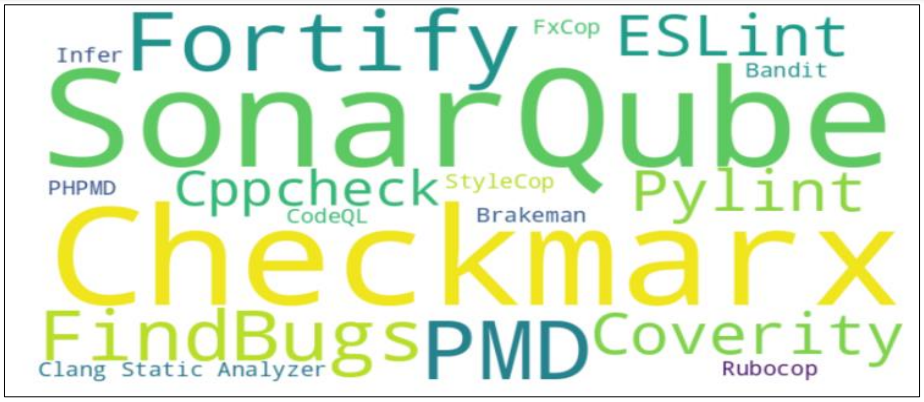

FIGURE 5 STATIC CODE ANALYSIS

It plays a vital part in contemporary software development by automatically checking source code for possible problems prior to execution. It makes it easier and more economical to fix issues by enabling the early discovery of bugs, security flaws, and best practice violations.

3.3 Dynamic Code Analysis (DCA)

The process of assessing a program by running it in a real or simulated environment and observing how it behaves is known as dynamic code analysis. Dynamic code analysis assists in finding runtime problems, including memory leaks, performance snags, input validation errors, and other vulnerabilities that only show up during execution, in contrast to static code analysis, which examines the code without running it[85]. This kind of analysis is very helpful for finding faults that arise during user interactions or conditions.

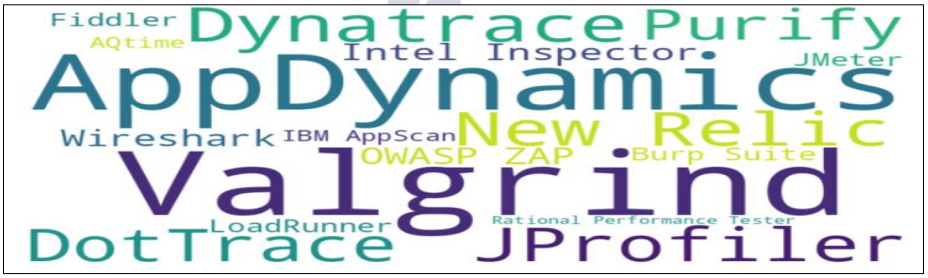

FIGURE 6 DYNAMIC CODE ANALYSIS TOOLS

DCA tools keep an eye on the application while it's running, giving information on how it responds to different inputs, how much system resources it uses, and whether it complies with security standards. Commonly employed methods include behavioural monitoring, runtime instrumentation, and fuzz testing. It is frequently used in the testing stage to make sure software operates as intended under both typical and unusual circumstances. Although dynamic analysis can be slower and require more resources, it offers important real-world validation that static analysis cannot. Table III dynamic analysis.

Table III Dynamic Analysis

Citations	Problem	Techniques	Vulnerabilities	Dataset
[86, 87]	Automated vulnerability detection for PHP	DL	SQLI	SARD
[88, 89]	Dynamic	SL	SQLi	NVD
	Current vulnerability	SL	Unavailable vulnerability data	Bugzilla

	prediction models			
[90, 91]	Vulnerability scanner	SL	Unavailable vulnerability data	288 DARPA
[92]	offline analysis in the modern	SL	OS command injection	PHPMyAdmin

3.4 Hybrid Code Analysis (HCA)

A software testing technique called hybrid code analysis combines the benefits of static and dynamic code analysis to offer a more thorough assessment of the security and quality of a program. While dynamic analysis watches behaviour while the code is being executed, and static analysis looks at the source code without running it, hybrid analysis combines two methods to find a wider variety of problems. This method provides more precise and context-aware insights by identifying runtime vulnerabilities and code structural flaws.

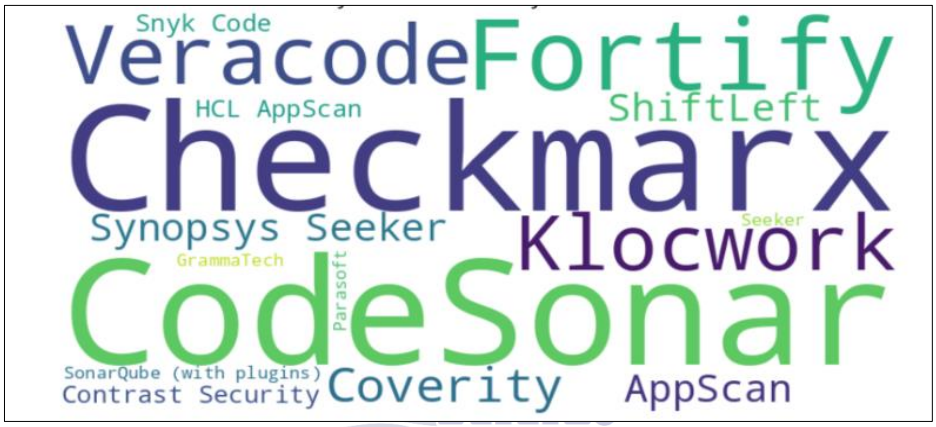

FIGURE 7 HYBRID CODE ANALYSIS TOOLS

HCA Developers and security experts can find intricate bugs, logical mistakes, and security vulnerabilities that would go unnoticed when utilising only one technique by utilising hybrid analysis. For instance, hybrid analysis can reduce false positives and boost confidence in the results by first scanning the code for potential vulnerabilities (static) and then confirming them during execution (dynamic). This integrated approach is particularly useful in high-stakes settings where security and dependability are crucial, such as embedded systems, healthcare software, and financial systems.

4. Methodology for conducting comprehensive Review

Data collection is the process of collecting information from multiple sources to evaluate and make it defensible. Depending on the goals and nature of the data being gathered, it can be broken down into several processes and utilize a variety of techniques and instruments. We have collected more than two hundred articles, international conferences, and book chapters from different databases; the details are in Table IV.

Table IV presents the details of the data collection

Data Source	2020	2021	2022	2023	2024	Total
Articles						
IEEE	21	28	25	25	18	117
Elsevier	18	16	22	19	20	95
ScienceDirect	10	9	12	11	8	50
Google Scholar	6	5	8	7	4	30
International Conference & Book Chapters						

IEEE	3	4	6	5	4	30
Elsevier	2	5	4	3	2	16
ScienceDirect	2	2	5	2	3	14
Google Scholar	1	3	2	3	2	11

The phases involved in the data collection process can be better understood and communicated by using illustrations.

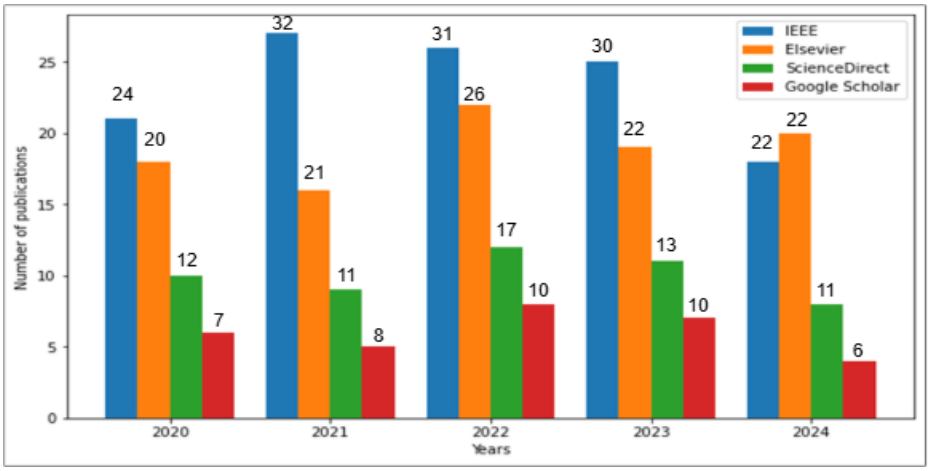

FIGURE 8 DATA COLLECTION

THIS FIGURE PRESENTS THE COLLECTION AND INVESTIGATES PREVIOUS RESEARCH ON CATEGORISATION AND SOFTWARE VULNERABILITY IDENTIFICATION ACROSS A RANGE OF ISSUES AND SOURCE CODES FROM 2020 TO 2024. SINCE THIS STUDY INTENDS TO LOOK MORE CLOSELY AT THE APPLICATION OF MACHINE LEARNING TECHNIQUES IN SOFTWARE FLAW DETECTION AND CLASSIFICATION, WE ALSO EXAMINE ARTICLES THAT USE THESE TECHNIQUES TO IDENTIFY SOFTWARE VULNERABILITIES.

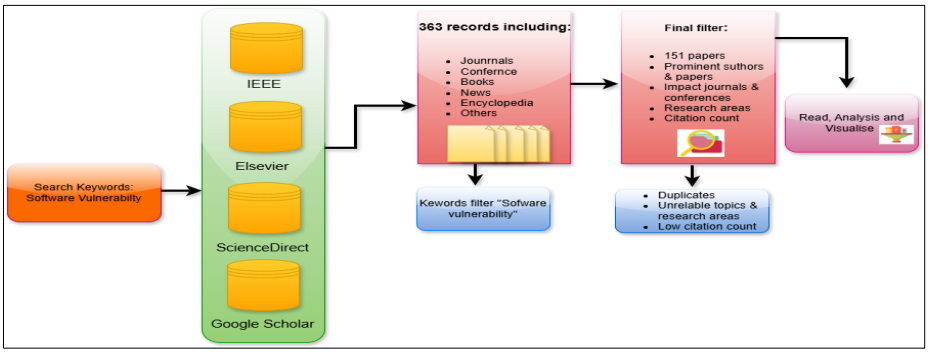

FIGURE 9 FLOWCHART FOR THE DATA COLLECTION

Using "software vulnerability" as our primary keyword, we search three databases—ScienceDirect, IEEE Xplore, and Google Scholar—for pertinent research papers to find the ones used for this study. These databases were selected because they provide a wealth of research articles on a variety of subjects, which raises the visibility of research papers in vulnerability identification. However, many of these records originate from several study fields that are somewhat connected to identifying vulnerabilities. The basis of this research is an analysis of 151 excellent and highly relevant experimental studies on software vulnerability identification. After locating all pertinent articles from the sources, we carefully read and

examined each one, taking note of specifics such as the problem statement and the detection strategy employed. Figure 10 the cumulative total of different categories of research problems, 2020-2025, collected from studies.

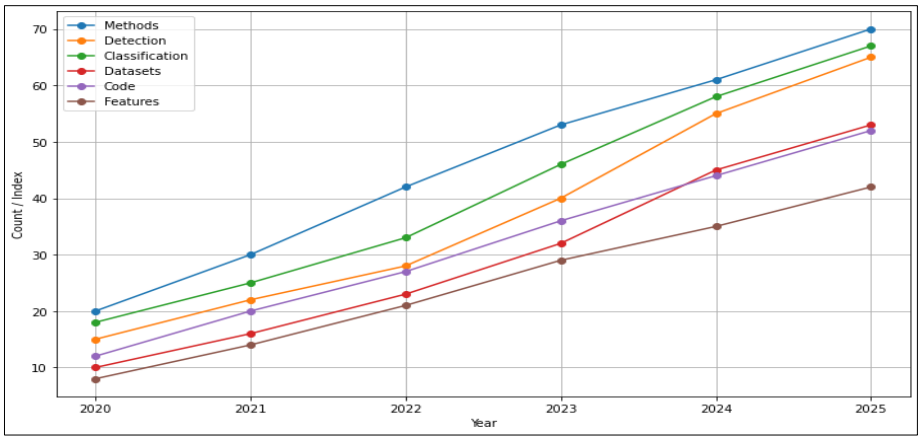

FIGURE 10 THE CUMULATIVE TOTAL OF DIFFERENT CATEGORIES OF RESEARCH, 2020-2025, COLLECTED FROM STUDIES

4.1 SQL injection attacks

A widespread and dangerous hack known as SQL Injection (SQLi) involves inserting malicious SQL queries into input fields to alter or take advantage of a backend database. Applications that improperly sanitise user input are the subject of these attacks, which provide hackers the ability to get around authentication, acquire data without authorisation, edit or remove records, and occasionally obtain administrative access to the system [93, 94]. Sensitive data, including credit card numbers, usernames, passwords, and even entire databases, can be compromised by SQL injection. Even though SQL injection is a well-known weakness, it is nevertheless common, particularly in applications with poor coding or legacy systems. As part of secure coding techniques, prepared statements, input validation, and strong error handling are necessary to prevent SQLi attacks.

Finding SQL injection flaws has been recognised as a serious risk to web applications. Attackers can gain unfettered access to the underlying databases through web apps that are vulnerable to SQL injection. When sensitive user or consumer data is stored in these systems, security breaches can have serious repercussions. Identity theft, the disclosure of private information, and dishonest behaviour are among the consequences [95, 96]. The vulnerabilities can be used by an attacker to run SQL queries on the database. SQL injection attacks target different aspects of SQL database queries and can take many different shapes. Table V types of SQLi and common attack patterns.

Table V Types of SQLi and common attack patterns

SQL Injection Types	Characteristics	Common Attack Patterns
Traditional SQL Injection [97]	Manipulates SQL queries	Injecting malicious SQL code into input fields.
Blind SQL Injection [98, 99]	Exploits vulnerabilities without observing outcomes	Injecting malicious code to test responses.
Time-Based Blind SQL Injection [100, 101]	Introduces delays in SQL queries.	Injecting code with time delays
Error-Based SQL Injection [88, 102, 103]	Injects code to force database errors.	Injecting code to generate errors
Union-Based SQL Injection [88, 104, 105]	Utilizes SQL UNION operator to merge data from multiple tables	Injecting code with UNION statements

Out-of-Band SQL Injection [97, 103, 106]	Exfiltrates data through a separate communication channel	Redirecting data to external channels.
Second-Order SQL Injection [107-109]	Injects data initially without immediate impact	Injecting data without immediate consequences
Content-Based SQL Injection [110, 111]	Manipulates queries based on retrieved content	• Modifying queries based on retrieved data
Boolean-Based SQL Injection [112, 113]	Inserts code based on true or false conditions	Crafting code based on Boolean conditions
Time-Based Blind SQL Injection [114, 115]	Introducing delays in SQL queries	Monitoring response times for confirmation.

Between 2010 and 2024, software-based SQL injection attacks remained one of the most strong and dangerous threats to web applications. These attacks exploit vulnerabilities in application code that participates with databases, allowing attackers to manipulate SQL queries and gain unauthorized access to sensitive data. Over the years, attackers have evolved their techniques, making SQL injections more covert and more automated, often integrating them into larger attack campaigns such as ransomware delivery and credential theft. Despite increased awareness and advances in security practices, such as input validation, parameterized queries, and web application firewalls, SQL injections continue to exploit poorly secured legacy systems and rapid software development cycles. The period also saw the rise of tools and frameworks that simplify detection and prevention, but the widespread use of open-source platforms and third-party Upgrades often introduces new vulnerabilities. Overall, SQL injection attacks during this timeframe highlight the ongoing need for secure coding practices and regular vulnerability assessments. Figure 11 Software-based SQL injection attacks 2010-2024.

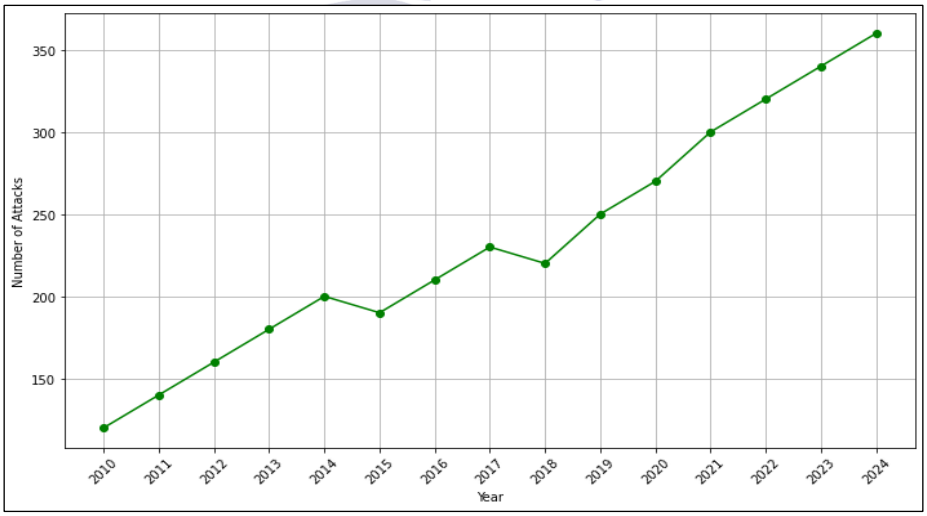

FIGURE 11: SOFTWARE-BASED SQL INJECTION ATTACKS 2010-2024 COLLECTED FROM STUDIES[116-123]

4. Critical Review and Comparative Study

We have studied various vulnerabilities related research studies from different sources, with relevant experimental, theoretical, mathematical modelling and case studies on software web-app vulnerabilities.

We found, based on thorough literature findings, that there are various approaches to detecting software vulnerabilities, such as static, manual, dynamic and hybrid approaches. However, this research found that when we use Machine Learning to detect vulnerabilities, it works with more reliable detection with better

accuracy compared to the way we detect without using Machine Learning. The data has been collected from different database sources including, such as IEEE, Elsevier, ScienceDirect, and Google Scholar, Table III and Figures 8, and 9. We presented the types of attacks on software vulnerability and comparative analysis of various approaches in Figure 12 and Table VI.

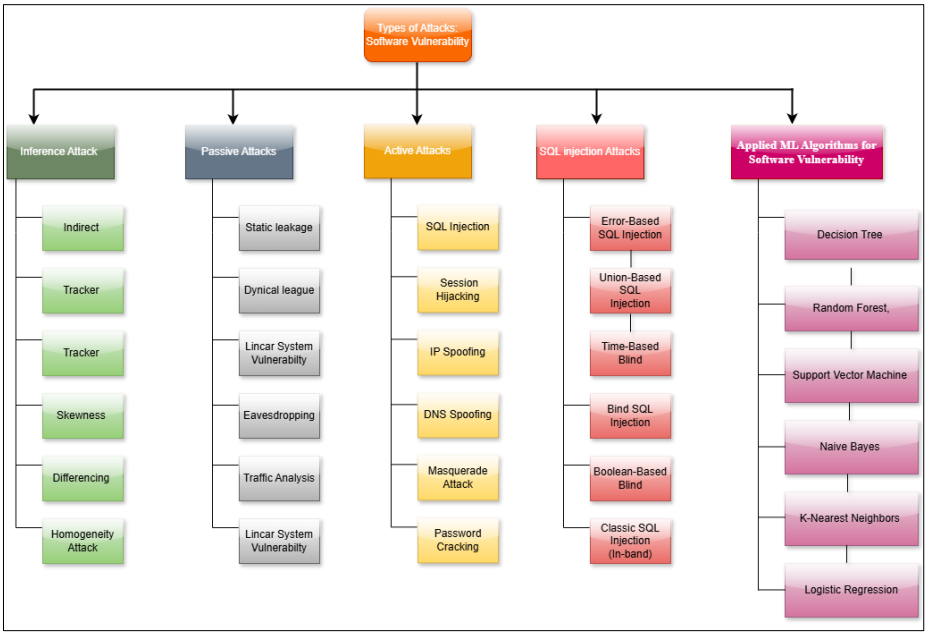

FIGURE 12 SHOWS THE TYPES OF ATTACKS ON SOFTWARE VULNERABILITIES [93, 124-130]

Furthermore, adding Machine Learning (ML) to software vulnerability identification, detection and cyber threat landscape analysis is a new frontier in cybersecurity. Table VI Comparative analysis of various approaches with ML.

Table VI Comparative analysis of various approaches with ML

Execution Required	Static Analysis	Dynamic Analysis	Manual Analysis	Hybrid Analysis	ML Analysis
Best Use Case [131-133]	Early development detection	Runtime behavior analysis	Critical security reviews	Comprehensive security validation	Large-scale automated analysis with evolving threats
Accuracy [88, 134-136]	Moderate – false positives possible	High – observes real behavior	High – depends on expertise	High – balances both strengths, depends on combinations	It varies – depends on data quality and model accuracy. However, generally it is with better accuracy over the other approaches
Automation Level [137-139]	low	Medium	Low	Medium to High	High
Skill Required [140-142]	Low to Medium	Medium to High	High	High	Medium to High
Vulnerability Detection [143-145]	Known patterns, mostly	Runtime vulnerabilities,	Complex, logic, and	Broader detection scope	Known and unknown vulnerabilities

	syntax issues	memory issues	design flaws		
False Positives/Negatives [146-151]	More false positives	Fewer false positives	Fewer false positives, more coverage	Balanced error rates	Can reduce or increase both, depending on training set

When classifying and detecting SQL injection attacks, Static Analysis examines source code without execution, offering fast classification and detection of known patterns but often missing runtime behaviors. **Dynamic Analysis** monitors application behavior during execution, effectively catching real-time SQL injections, but at a higher resource cost. **Manual Analysis**, conducted by experts, provides deep insight and accuracy but lacks scalability and is time-consuming. **Hybrid** Analysis combines static and dynamic methods to enhance detection coverage and reduce false positives. Machine Learning (ML) Analysis introduces automation and adaptability by learning from attack patterns, enabling detection of novel SQL injection techniques, though it requires quality datasets and may struggle with explainability.

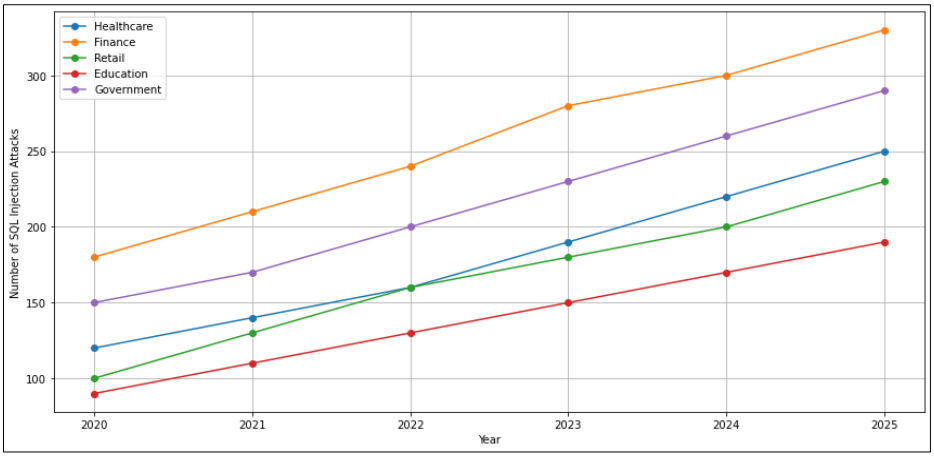

FIGURE 13 SHOWS THE NUMBER OF ATTACKS IN DIFFERENT INDUSTRIES FROM 2020-2025 COLLECTED FROM STUDIES

From 2020 to 2025, SQL injection attacks have steadily increased across major industries. The Finance sector consistently experienced the highest number of attacks, rising from 180 in 2020 to 330 in 2025. The Healthcare industry followed, with attacks growing from 120 to 250 over the same period. Retail, Education, and Government sectors also saw significant increases, reflecting a broader trend of rising SQL injection threats as digital infrastructures expanded. This upward trajectory highlights the urgent need for stronger security measures and intelligent detection systems across all sectors. Figure 13 shows the number of attacks in different industries from 2020-2025.

Finally, we have found from the literature that these challenges in **error-based** SQL Injection (SQLi) include inconsistent error handling across databases, insufficient input validation, poor security configurations, and developers' lack of awareness about secure coding practices. Additionally, error-based SQLi can be difficult to identify and detect when errors are not visible to users but are logged internally. Effective detection, classification and mitigation include using parameterized queries, disabling detailed error messages, and conducting regular security testing. Another challenge in the **Union-based** SQL Injection (SQLi) in the study includes identifying the correct column count, bypassing input filters, and dealing with database-specific syntax Figure 5.

5. Challenges and Opportunities for Software Vulnerability:

Challenges:

Dynamic Nature of Web Technology

The nature of web technology is mainly dynamic and extends the support to several components, including API's, Scripts, Flesh, Ajax, etc., which open several possibilities for vulnerabilities and make it easier for hackers. The rapid evolution of web technologies creates a constant need for updates and new skills, tools and practices. This leads to compatibility issues and potential security vulnerabilities.

Inadequate Input Validation

Inadequate input validation can lead to serious security vulnerabilities such as SQL injection (error-based, Union-based). It compromises data integrity and allows attackers to manipulate system behaviour. This weakness also undermines user trust and system reliability. There are several standards to be followed while designing the input forms. However, there is still a huge gap, which allows the hacker to exploit this for the SQL injection attacks in general.

Insufficient Authentication and Session Management

There are several authentication and authorisation mechanisms in practice for web apps. However, they are not at the required level, only rather the finance applications. Which causes the issues and challenges for the web-apps. Insufficient authentication and session management can allow unauthorised access to sensitive data and user accounts. Poorly managed sessions increase the risk of hijacking and impersonation. This compromises system security and user privacy, making systems vulnerable to attacks.

Third-Party Components and Dependencies

Most of the websites they are using various API's from third party ton enhance the functionalities of the web-apps. These third-party components and dependencies can introduce hidden vulnerabilities and outdated code into web applications. Relying on external sources makes it difficult to ensure consistent security and performance.

Misconfigured Security Settings

Misconfigured security settings can expose applications to unauthorised access and data breaches. Default or weak configurations often go unnoticed, leaving critical vulnerabilities open to exploitation. These issues undermine the overall security posture and trustworthiness of the system.

Human error in the organisation.

Human error within an organization can led to data breaches, system misconfigurations, and accidental information leaks. Mistakes such as weak passwords and phishing emails. These errors highlight the need for ongoing training and strict security protocols. The blue hat hacker could be severe danger for the organizations, as they are aware and part of the organisation.

Security certificate expiration

Security certificate expiration can disrupt website accessibility and break secure connections, leading to user distrust. Expired certificates expose systems to potential man-in-the-middle attacks. Managing renewals across multiple systems is complex and often overlooked, increasing security risks.

Cookies management

Most of the web-apps are asking for the higher-level access to Cookies, which causes damage later for the system. Poor cookie management can lead to privacy violations and unauthorised data tracking. Insecure handling of cookies may expose sensitive user information to attackers.

Opportunities:

Development of Intelligent Detection Models

Since the entire business and data move mainly through the web apps for the various organizations. Hence, breach of the data and exploring the vulnerabilities within the web-app is highly crucial. The new era tools and techniques, including ML and AI, enable us to face this challenge in a better way, by developing better models to detect the web app vulnerabilities. The development of intelligent detection models focuses on leveraging machine

learning and artificial intelligence to identify patterns and anomalies in complex data. These models are designed to enhance accuracy and speed in detecting threats and vulnerabilities.

Automation of Vulnerability Detection
Automation of vulnerability detection streamlines the process of identifying security flaws in software systems using advanced tools and algorithms. It reduces the need for manual code review, increasing efficiency and consistency across large-scale applications. This approach enables faster response times and enhances overall cybersecurity posture. The automation becomes more feasible with the use of ML.

Standardisation of Secure Coding Practices
Standardisation of secure coding practices ensures consistency in writing code that adheres to security guidelines across development teams. It helps prevent common vulnerabilities by enforcing best practices and coding standards. This approach strengthens software security and simplifies code maintenance and auditing.

Modular and Scalable Security Framework
A modular and scalable security framework allows flexible integration of security components tailored to specific system needs. Its architecture supports easy expansion as threats evolve. This research recommends better security frameworks which can fulfil the current vibrant need of security.

Improving User and Developer Awareness
Improving user and developer awareness focuses on educating stakeholders about security best practices and potential risks. This knowledge empowers users to make safer choices and developers to write more secure code. Increased awareness is key to reducing human error and strengthening overall cybersecurity defenses. This is recommended to be adopted this as a continuous practice for the better security of organizations.

6. Conclusion and Recommendations

This research concludes various folds related to the web app security, considering mainly the SQL injection attacks. We discussed the possibilities, challenges, and opportunities related to this. Where we consider different approaches for the vulnerability detection, including manual, static, dynamic, and hybrid. Further we compared the web-app vulnerabilities detection through Machine Learning, as shown in Figure 12. The research concluded that ML based approaches are far better over the existing approaches. Furthermore, Machine Learning is increasing in importance in the software industry, particularly when it comes to improving defensive applications and navigating the complicated threat landscape. Traditional security solutions are frequently insufficient to manage these constantly evolving issues as cyber threats increase in sophistication and frequency. ML offers a strong response to this urgent problem because of its capacity to learn from enormous datasets and spot patterns. Using ML-driven security systems, which can identify anomalies, anticipate risks, and automate responses, can reduce the time and effort required for threat classification and detection. ML is a key player in the software sector, providing innovative ways to improve defensive apps and traverse the threat landscape. Organizations may position themselves to prosper in the quickly changing digital world by utilizing ML to create software that is inventive, safe, and efficient. The software industry's capacity to fully utilize ML and future abstractness into chances for development and progress will determine its future.

REFERENCE

[1] B. Arnold and Y. Qu, "Detecting software security vulnerability during an agile development by testing the changes to the security posture of software systems," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2020: IEEE, pp. 1743-1748.

[2] Y. Fan, J. Li, D. Zhang, J. Pi, J. Song, and G. Zhao, "Supporting sustainable maintenance of substations under cyber-threats: An evaluation method of cybersecurity risk for power CPS," *Sustainability*, vol. 11, no. 4, p. 982, 2019.

[3] S. H. Gill *et al.*, "Security and privacy aspects of cloud computing: a smart campus case study," *Intelligent Automation & Soft Computing*, vol. 31, no. 1, pp. 117-128, 2022.

[4] N. Guo, X. Li, H. Yin, and Y. Gao, "Vulhunter: An automated vulnerability detection system based on deep learning and bytecode," in *Information and Communications Security: 21st International Conference, ICICS 2019, Beijing, China, December 15–17, 2019, Revised Selected Papers 21*, 2020: Springer, pp. 199-218.

[5] Shah, I. A. (2024). Privacy and security challenges in unmanned aerial vehicles (UAVs). Cybersecurity in the Transportation Industry, 93-115.[6] M. Humayun, N. Jhanjhi, M. Niazi, F. Amsaad, and I. Masood, "Securing Drug Distribution Systems from Tampering Using Blockchain. Electronics 2022, 11, 1195," ed: s Note: MDPI stays neu-tral with regard to jurisdictional claims in ..., 2022.

[7] S. Qadir, E. Waheed, A. Khanum, and S. Jehan, "Comparative evaluation of approaches & tools for effective security testing of Web applications," *PeerJ Computer Science*, vol. 11, p. e2821, 2025.

[8] M. I. Khalil, M. Humayun, N. Jhanjhi, M. Talib, and T. A. Tabbakh, "Multi-class segmentation of organ at risk from abdominal ct images: A deep learning approach," in *Intelligent Computing and Innovation on Data Science: Proceedings of ICTIDS 2021*, 2021: Springer, pp. 425-434.

[9] Shah, I. A., Sial, Q., & Fateh, S. (Eds.). (2024). Generative AI Techniques for Sustainability in Healthcare Security. IGI Global.[10] S. Muzafar, M. Humayun, and S. J. Hussain, "Emerging Cybersecurity Threats in the Eye of E-Governance in the Current Era," in *Cybersecurity Measures for E-Government Frameworks*: IGI Global, 2022, pp. 43-60.

[11] S. Sennan *et al.*, "Energy efficient optimal parent selection based routing protocol for Internet of Things using firefly optimization algorithm," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 8, p. e4171, 2021.

[12] I. A. Shah, N. Z. Jhanjhi, F. Amsaad, and A. Razaque, "The role of cutting-edge technologies in industry 4.0," in *Cyber Security Applications for Industry 4.0*: Chapman and Hall/CRC, 2022, pp. 97-109.

[13] R. Sharma, A. Singh, N. Jhanjhi, M. Masud, E. S. Jaha, and S. Verma, "Plant Disease Diagnosis and Image Classification Using Deep Learning," *Computers, Materials & Continua*, vol. 71, no. 2, 2022.

[14] R. M. A. Ujjan, I. Taj, and S. N. Brohi, "E-Government Cybersecurity Modeling in the Context of Software-Defined Networks," in *Cybersecurity Measures for E-Government Frameworks*: IGI Global, 2022, pp. 1-21.

[15] X. Wang, R. Wu, J. Ma, G. Long, and J. Han, "Research on vulnerability detection technology for web mail system," *Procedia computer science*, vol. 131, pp. 124-130, 2018.

[16] Y. Yang, S. Wang, M. Wen, and W. Xu, "Reliability modeling and evaluation of cyber-physical system (CPS) considering communication failures," *Journal of the Franklin Institute*, vol. 358, no. 1, pp. 1-16, 2021.

[17] Z. Yu, C. Theisen, L. Williams, and T. Menzies, "Improving vulnerability inspection efficiency using active learning," *IEEE Transactions on Software Engineering*, vol. 47, no. 11, pp. 2401-2420, 2019.

[18] I. A. Shah, A. Laraib, H. Ashraf, and F. Hussain, "Drone Technology: Current Challenges and Opportunities," *Cybersecurity Issues and Challenges in the Drone Industry*, pp. 343-361, 2024.

[19] M. Zagane, M. K. Abdi, and M. Alenezi, "Deep learning for software vulnerabilities detection using code metrics," *IEEE Access*, vol. 8, pp. 74562-74570, 2020.

[20] M. Zhou *et al.*, "A method for software vulnerability detection based on improved control flow graph," *Wuhan University Journal of Natural Sciences*, vol. 24, no. 2, pp. 149-160, 2019.

[21] N. Z. Jhanjhi and I. A. Shah, *Navigating Cyber Threats and Cybersecurity in the Logistics Industry*. IGI Global, 2024.

[22] I. A. Shah, N. Z. Jhanjhi, and S. K. Ray, "Enabling Explainable AI in Cybersecurity Solutions," in *Advances in Explainable AI Applications for Smart Cities*: IGI Global, 2024, pp. 255-275.

[23] I. Shah, N. Jhanjhi, and S. Ray, "IoT Devices in Drones: Security Issues and Future Challenges. Cybersecurity Issues and Challenges in the Drone Industry," *IGI Global. DOI*, vol. 10, pp. 979-8, 2024.

[24] N. Jhanjhi, I. A. Shah, and S. N. Brohi, "Hardware Vulnerabilities: Taxonomy and Business Security Models," in *Generative AI for Web Engineering Models*: IGI Global, 2025, pp. 129-146.

[25] A. Agrawal *et al.*, "Software security estimation using the hybrid fuzzy ANP-TOPSIS approach: Design tactics perspective," *Symmetry*, vol. 12, no. 4, p. 598, 2020.

[26] M. T. Ahvanooey, Q. Li, M. Rabbani, and A. R. Rajput, "A survey on smartphones security: software vulnerabilities, malware, and attacks," *arXiv preprint arXiv:2001.09406*, 2020.

[27] Z. Alansari, N. B. Anuar, A. Kamsin, S. Soomro, and M. R. Belgaum, "Computational intelligence tools and databases in bioinformatics," in *2017 4th IEEE international conference on engineering technologies and applied sciences (ICETAS)*, 2017: IEEE, pp. 1-6.

[28] M. Alfadel, D. E. Costa, and E. Shihab, "Empirical analysis of security vulnerabilities in python packages," *Empirical Software Engineering*, vol. 28, no. 3, p. 59, 2023.

[29] R. R. Althar, D. Samanta, M. Kaur, A. A. Alnuaim, N. Aljaffan, and M. Aman Ullah, "[Retracted] Software Systems Security Vulnerabilities Management by Exploring the Capabilities of Language Models Using NLP," *Computational Intelligence and Neuroscience*, vol. 2021, no. 1, p. 8522839, 2021.

[30] I. A. Shah, N. Z. Jhanjhi, and R. M. A. Ujjan, "Drone Technology in the Context of the Internet of Things," in *Cybersecurity Issues and Challenges in the Drone Industry*: IGI Global, 2024, pp. 88-107.

[31] I. A. Shah, N. Jhanjhi, and S. N. Brohi, "Proposing Model for Classification of Malicious SQLi Code Using Machine Learning Approach," in *2024 1st International Conference on Innovative Engineering Sciences and Technological Research (ICIESTR)*, 2024: IEEE, pp. 1-6.

[32] A. Bahaa, A. E.-R. Kamal, and A. S. Ghoneim, "A systematic literature review on software vulnerability detection using machine learning approaches," *FCI-H Informatics Bulletin*, vol. 4, no. 1, pp. 1-9, 2022.

[33] M. R. Belgaum, Z. Alansari, S. Musa, M. M. Alam, and M. Mazliham, "Impact of artificial intelligence-enabled software-defined networks in infrastructure and operations: Trends and challenges," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 1, 2021.

[34] M. R. Belgaum, F. Ali, Z. Alansari, S. Musa, M. M. Alam, and M. Mazliham, "Artificial intelligence based reliable load balancing framework in software-defined networks," *CMC–Comput. Mater. Contin*, vol. 70, pp. 251-266, 2022.

[35] M. R. Belgaum *et al.*, "Enhancing the Efficiency of Diabetes Prediction through Training and Classification using PCA and LR Model," *Annals of Emerging Technologies in Computing (AETiC)*, vol. 7, no. 3, pp. 78-91, 2023.

[36] N. Bhatt, A. Anand, and V. S. Yadavalli, "Exploitability prediction of software vulnerabilities," *Quality and Reliability Engineering International*, vol. 37, no. 2, pp. 648-663, 2021.

[37] F. A. Bhuiyan, M. B. Sharif, and A. Rahman, "Security bug report usage for software vulnerability research: a systematic mapping study," *IEEE Access*, vol. 9, pp. 28471-28495, 2021.

[38] Fateh, S., Sial, Q., Dar, S. H., Shah, I. A., & Rani, A. (2024). Smart Healthcare System in Industry 4.0. In Advances in Computational Intelligence for the Healthcare Industry 4.0 (pp. 297-311). IGI Global Scientific Publishing.[39] A. Bosu, J. C. Carver, M. Hafiz, P. Hilley, and D. Janni, "Identifying the characteristics of vulnerable code changes: An empirical study," in *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, 2014, pp. 257-268.

[40] M. Boukhechba, A. R. Daros, K. Fua, P. I. Chow, B. A. Teachman, and L. E. Barnes, "DemonicSalmon: Monitoring mental health and social interactions of college students using smartphones," *Smart Health*, vol. 9, pp. 192-203, 2018.

[41] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 105-114.

[42] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287-1289, 2019.

[43] I. A. Shah, N. Jhanjhi, and S. Brohi, "Cybersecurity issues and challenges in civil aviation security," *Cybersecurity in the Transportation Industry*, pp. 1-23, 2024.

[44] L. Gaur *et al.*, "Disposition of youth in predicting sustainable development goals using the neuro-fuzzy and random forest algorithms," *Human-Centric Computing and Information Sciences*, vol. 11, p. NA, 2021.

[45] L. Gaur, R. M. A. Ujjan, and M. Hussain, "The influence of deep learning in detecting cyber attacks on e-government applications," in *Cybersecurity Measures for E-Government Frameworks*: IGI Global, 2022, pp. 107-122.

[46] Shah, I. A., & Jhanjhi, N. Z. (2025, April). An Intelligent Approach for Classification In-Band SQLi Attacks Using Machine Learning Techniques. In 2025 4th International Conference on Computing and Information Technology (ICCIT) (pp. 125-130). IEEE.

[47] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey," *ACM computing surveys (CSUR)*, vol. 50, no. 4, pp. 1-36, 2017.

[48] M. C. Ghanem and T. M. Chen, "Reinforcement learning for efficient network penetration testing," *Information*, vol. 11, no. 1, p. 6, 2019.

[49] I. A. Shah, N. Jhanjhi, and S. N. Brohi, "IoT Smart Healthcare Security Challenges and Solutions," in *Advances in Computational Intelligence for the Healthcare Industry 4.0*: IGI Global Scientific Publishing, 2024, pp. 234-247.

[50] P. I. R. Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the Internet of Things: Challenges, threats and solutions," *Internet of Things*, vol. 5, pp. 41-70, 2019.

[51] G. Grieco and A. Dinaburg, "Toward smarter vulnerability discovery using machine learning," in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, 2018, pp. 48-56.

[52] I. A. Shah, N. Z. Jhanjhi, and S. K. Ray, "IoT Devices in Drones: Security Issues and Future Challenges," in *Cybersecurity Issues and Challenges in the Drone Industry*: IGI Global, 2024, pp. 217-235.

[53] Shah, I. A. (2025). Security Issues and Challenges in Edge Computing Architecture for the Drone Industry. Computer Vision and Edge Computing Technologies for the Drone Industry, 257-270.[54] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet of Things*, vol. 7, p. 100059, 2019.

[55] M. Humayun, N. Jhanjhi, M. Alruwaili, S. S. Amalathas, V. Balasubramanian, and B. Selvaraj, "Privacy protection and energy optimization for 5G-aided industrial Internet of Things," *Ieee Access*, vol. 8, pp. 183665-183677, 2020.

[56] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges," *IEEE communications surveys & tutorials*, vol. 22, no. 2, pp. 1251-1275, 2020.

[57] N. Jhanjhi, M. Ahmad, M. A. Khan, and M. Hussain, "The impact of cyber attacks on e-governance during the covid-19 pandemic," in *Cybersecurity Measures for E-Government Frameworks*: IGI Global, 2022, pp. 123-140.

[58] A. L. Johnson, "The analysis of binary file security using a hierarchical quality model," Montana State University-Bozeman, College of Engineering, 2022.

[59] H. Kekül, B. ERGEN, and H. ARSLAN, "A multiclass approach to estimating software vulnerability severity rating with statistical and word embedding methods," *Int J Comput Netw Inf Secur*, vol. 12, no. 4, p. 27, 2022.

[60] N. K. Khan, E. Alnatsheh, R. A. Rasheed, A. Yadav, and Z. Alansari, "A quantitative case study in WSNs: Design and implementation of student smart ID card," in *2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, 2020: IEEE, pp. 27-32.

[61] J. Kronjee, A. Hommersom, and H. Vranken, "Discovering software vulnerabilities using data-flow analysis and machine learning," in *Proceedings of the 13th international conference on availability, reliability and security*, 2018, pp. 1-10.

[62] A. Peedikayil Kuruvila, I. Zografopoulos, K. Basu, and C. Konstantinou, "Hardware-Assisted Detection of Firmware Attacks in Inverter-Based Cyberphysical Microgrids," *arXiv e-prints*, p. arXiv: 2009.07691, 2020.

[63] T. H. M. Le and M. A. Babar, "On the use of fine-grained vulnerable code statements for software vulnerability assessment models," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 621-633.

[64] K. Lee and K. Yim, "Cybersecurity threats based on machine learning-based offensive technique for password authentication," *Applied Sciences*, vol. 10, no. 4, p. 1286, 2020.

[65] Jhanjhi, N. Z., Shah, I. A., & Brohi, S. N. (2025). Evaluating Cybersecurity Frameworks Safeguarding the Software Industry Against Evolving Threats. In Navigating Cyber Threats and Cybersecurity in the Software Industry (pp. 111-128). IGI Global Scientific Publishing.

[66] J. Li, "Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST)," *arXiv preprint arXiv:2004.03216*, 2020.

[67] S. I. Ali, Z. Noor, and R. Samina, "Cybersecurity measures for E-government frameworks Advances in electronic government, digital divide, and regional development./[edited by] Noor Zaman, Imdad Ali Shah, Samina Rajper," ed.

[68] Shah, I. A., Jhanjhi+, N. Z., & Brohi, S. N. (2024, December). Secure model for credit card fraud detection using ML approaches. In 8th IET Smart Cities Symposium (SCS 2024) (Vol. 2024, pp. 709-715). IET.

[69] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[70] M. Lim, A. Abdullah, N. Jhanjhi, and M. Supramaniam, "Hidden link prediction in criminal networks using the deep reinforcement learning technique," *Computers*, vol. 8, no. 1, p. 8, 2019.

[71] X. Liu, J. Ospina, and C. Konstantinou, "Deep reinforcement learning for cybersecurity assessment of wind integrated power systems," *IEEE access*, vol. 8, pp. 208378-208394, 2020.

[72] J. Lu *et al.*, "Joint modeling of heterogeneous sensing data for depression assessment via multi-task learning," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1-21, 2018.

[73] G. Márquez and H. Astudillo, "Identifying availability tactics to support security architectural design of microservice-based systems," in *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, 2019, pp. 123-129.

[74] A. Mehrotra and M. Musolesi, "Using autoencoders to automatically extract mobility features for predicting depressive states," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1-20, 2018.

[75] M. H. Miraz, P. S. Excell, A. Ware, S. Soomro, and M. Ali, *Emerging Technologies in Computing*. Springer, 2021.

[76] Shah, I. A. (2025). Security Issues and Challenges in Edge Computing Architecture for the Drone Industry. Computer Vision and Edge Computing Technologies for the Drone Industry, 257-270.[77] V.-P. Ranganath and J. Mitra, "Are free android app security analysis tools effective in detecting known vulnerabilities?," *Empirical Software Engineering*, vol. 25, no. 1, pp. 178-219, 2020.

[78] I. A. Shah, "IoT-Based Smart Transportation Industry: Security Challenges," *Cybersecurity in the Transportation Industry*, pp. 211-239, 2024.

[79] K. Sahu and R. Srivastava, "Revisiting software reliability," *Data Management, Analytics and Innovation: Proceedings of ICDMAI 2018, Volume 1*, pp. 221-235, 2019.

[80] S. Siddique, A.-A. I. Hridoy, S. A. Khushbu, and A. K. Das, "Cvd: An improved approach of software vulnerability detection for object oriented programming languages using deep learning," in *Proceedings of the Future Technologies Conference*, 2022: Springer, pp. 145-164.

[81] S. R. Sindiramutty, N. Z. Jhanjhi, C. E. Tan, N. A. Khan, B. Shah, and A. R. Manchuri, "Cybersecurity Measures for Logistics Industry," in *Navigating Cyber Threats and Cybersecurity in the Logistics Industry*: IGI Global, 2024, pp. 1-58.

[82] T. S. Tata Sutabri, "Design of A Web-Based Social Network Information System," *International Journal of Artificial Intelligence Research*, vol. 6, no. 1, pp. 310-316, 2023.

[83] I. A. Shah, "Privacy and security challenges in unmanned aerial vehicles (UAVs)," *Cybersecurity in the Transportation Industry*, pp. 93-115, 2024.

[84] C. Tiefenau, M. Häring, K. Krombholz, and E. Von Zezschwitz, "Security, availability, and multiple information sources: Exploring update behavior of system administrators," in *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, 2020, pp. 239-258.

[85] R. K. Vigneswaran, R. Vinayakumar, K. Soman, and P. Poornachandran, "Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security," in *2018 9th International conference on computing, communication and networking technologies (ICCCNT)*, 2018: IEEE, pp. 1-6.

[86] Altaf, S., Shah, I. A., & Shah, A. A. (2025). Vulnerability Scanning Tools for Web Application Security Frameworks. In Navigating Cyber Threats and Cybersecurity in the Software Industry (pp. 65-86). IGI Global Scientific Publishing.

[87] C. Lin, Y. Xu, Y. Fang, and Z. Liu, "VulEye: a novel graph neural network vulnerability detection approach for PHP application," *Applied Sciences*, vol. 13, no. 2, p. 825, 2023.

[88] S. Ammagunta, A. Akula, C. S. Pottipati, L. M. Avula, and Y. R. P. Venkata, "Defending against SQL injection: Practical application with open-source tools for improved cyber security," in *AIP Conference Proceedings*, 2025, vol. 3237, no. 1: AIP Publishing LLC, p. 020036.

[89] M. Lin *et al.*, "ToxicSQL: Migrating SQL Injection Threats into Text-to-SQL Models via Backdoor Attack," *arXiv preprint arXiv:2503.05445*, 2025.

[90] A. Biju, M. Ari, V. Gayathri, and S. M. Hatture, "SQL Injection Vulnerability Scanner (Secure Scan)," *Authorea Preprints*.

[91] S. M Hatture, M. Ari, A. Biju, and N. V. Gayathri, "SQL Injection Vulnerability Scanner (Secure Scan)," *Megha and S, Harsha Vardhana and Biju, Aditi and Gayathri, N Valli, SQL Injection Vulnerability Scanner (Secure Scan)(January 24, 2025)*, 2025.

[92] S. S. Nair, "Securing against advanced cyber threats: a comprehensive guide to phishing, XSS, and SQL injection defense," *Journal of Computer Science and Technology Studies*, vol. 6, no. 1, pp. 76-93, 2024.

[93] R. Bakır, "UniEmbed: A Novel Approach to Detect XSS and SQL Injection Attacks Leveraging Multiple Feature Fusion with Machine Learning Techniques," *Arabian Journal for Science and Engineering*, pp. 1-14, 2025.

[94] R. Vinayakumar, K. Soman, and P. Poornachandran, "Evaluating deep learning approaches to characterize and classify malicious URL's," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1333-1343, 2018.

[95] A. M. Begum, M. Arock, and U. S. Reddy, "Channel minimised depth-wise CNN with node weighted tree-LSTM model to detect nested query-based SQL injection attacks," *International Journal of Intelligent Engineering Informatics*, vol. 13, no. 1, pp. 78-113, 2025.

[96] S. Soni, "AN ANALYSIS OF SQL INJECTION AND CROSS-SITE SCRIPTING ATTACKS FOR ENHANCED SECURITY OF THE WEBSITE," *Emerging Research Trends in Computer Science and Information Technology*, p. 48, 2025.

[97] Shah, I. A. (2024). Autonomous shipping: Security issues and challenges. Cybersecurity in the transportation industry, 187-210.

[98] R. Vinayakumar, K. Soman, P. Poornachandran, and S. Sachin Kumar, "Detecting Android malware using long short-term memory (LSTM)," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1277-1288, 2018.

[99] S. Singh Chinthalapudi, "Detecting and Mitigating SQL Injection in. NET Applications Using AI-Based Anomaly Detection," *International Journal of Innovative Science and Research Technology*, vol. 10, no. 3, pp. 2582-2595, 2025.

[100] A. Kumar, S. Dutta, and P. Pranav, "Analysis of SQL injection attacks in the cloud and in WEB applications," *Security and Privacy*, vol. 7, no. 3, p. e370, 2024.

[101] Y. Liu and Y. Dai, "Deep Learning in Cybersecurity: A Hybrid BERT–LSTM Network for SQL Injection Attack Detection," *IET Information Security*, vol. 2024, no. 1, p. 5565950, 2024.

[102] A. Paul, V. Sharma, and O. Olukoya, "SQL injection attack: Detection, prioritization & prevention," *Journal of Information Security and Applications*, vol. 85, p. 103871, 2024.

[103] B. S. Lakshmi, D. Kovvuri, H. V. Bolisetti, D. S. Chikkala, S. Karri, and G. Yadlapalli, "A proactive approach for detecting SQL and XSS injection attacks," in *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, 2024: IEEE, pp. 1415-1420.

[104] M. N. Zaidan, P. Sukarno, and A. A. Wardana, "Collaborative Detection of SQL Injection Attacks using SIEM, Multi-Wazuh Agents, and Diverse Web Application Firewalls," in *2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES)*, 2024: IEEE, pp. 1-6.

[105] A. G. Kakisim, "A deep learning approach based on multi-view consensus for SQL injection detection," *International Journal of Information Security*, vol. 23, no. 2, pp. 1541-1556, 2024.

[106] R. Ala *et al.*, "Database Under Siege: The Hidden Menace of SQL Injection Attacks," in *International Conference on Broadband Communications, Networks and Systems*, 2024: Springer, pp. 144-153.

[107] B. Zhang, R. Ren, J. Liu, M. Jiang, J. Ren, and J. Li, "Sqlpsdem: A proxy-based mechanism towards detecting, locating and preventing second-order sql injections," *IEEE Transactions on Software Engineering*, 2024.

[108] J. Zhao, K. Chen, W. Zhang, and N. Yu, "SQL Injection Jailbreak: A Structural Disaster of Large Language Models," *arXiv preprint arXiv:2411.01565*, 2024.

[109] H. M. Athab, "Critical Evaluation of SQL Injection Security Measures in Web Applications."

[110] Shah, I. A., & Ahmed, M. (2025). Edge computing, emerging trends, and drone technology for the business industry. Computer Vision and Edge Computing Technologies for the Drone Industry, 89-104.

[111] C. Y. Daramola, S. A. Akinpelu, E. O. Akinyemi, S. A. Ajagbe, G. A. Akinpelu, and M. O. Adigun, "Malicious query recognition using chosen machine learning techniques," *SN Computer Science*, vol. 6, no. 3, p. 281, 2025.

[112] B. C. Nareshkumar and N. Vaidya, "Detection and Prevention of Sql Injection Using Machine Learning Based Methods."

[113] Å. Å. Sommervoll, L. Erdődi, and F. M. Zennaro, "Simulating all archetypes of SQL injection vulnerability exploitation using reinforcement learning agents," *International Journal of Information Security*, vol. 23, no. 1, pp. 225-246, 2024.

[114] J. R. Dora, L. Hluchý, and K. Nemoga, "Ontology for blind SQL injection," *Computing and Informatics*, vol. 42, no. 2, pp. 480–500-480–500, 2023.

[115] I. A. Shah, R. K. Murugesan, and S. Rajper, "Supply Chain Management Security Issues and Challenges in the Context of AI Applications," *Navigating Cyber Threats and Cybersecurity in the Logistics Industry*, pp. 59-89, 2024.

[116] V. Luong, "Intrusion detection and prevention system: SQL-injection attacks," 2010.

[117] Q. Xue and P. He, "On defense and detection of SQL server injection attack," in *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing*, 2011: IEEE, pp. 1-4.

[118] I. Schieferdecker, J. Grossmann, and M. Schneider, "Model-based security testing," *arXiv preprint arXiv:1202.6118*, 2012.

[119] V. Malekar and J. Waghmare, "Web application firewall to protect against web application vulnerabilities: A survey and comparison," *International Journal of Computer Technology and Applications*, vol. 4, no. 1, p. 141, 2013.

[120] A. Maleki, F. M. Farahani, and R. Maleki, "Study of Methods of SQL Injection Defence," *Asian Journal of Research in Business Economics and Management*, vol. 4, no. 8, pp. 305-310, 2014.

[121] J. Alves-Foss, J. Song, A. S. Amack, L. Kerr, and S. Steiner, "Evaluating the Use of Security Tags in Security Policy Enforcement Mechanisms," in *2015 48th Hawaii International Conference on System Sciences*, 2015: IEEE, pp. 5201-5210.

[122] P. Gunathilaka, D. Mashima, and B. Chen, "Softgrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions," in *Proceedings of the 2nd ACM workshop on cyber-physical systems security and privacy*, 2016, pp. 113-124.

[123] N. Shah, "Securing Database Users from the Threat of SQL Injection Attacks," 2017.

[124] A. B. Aulianoor and M. Koprawi, "Comparative Analysis of the Performance of Decision Tree and Random Forest Algorithms in SQL Injection Attack Detection," *Journal of Applied Informatics and Computing*, vol. 8, no. 1, pp. 194-202, 2024.

[125] I. Anwar, "Machine Learning Approaches for Detection of SQL Injection Attacks," *Jurnal Sistem Informasi dan Teknik Informatika (JAFOTIK)*, vol. 3, no. 1, pp. 1-6, 2025.

[126] A. H. Farhan and R. F. Hasan, "Detection SQL injection attacks against web application by using support vector machine with principal component analysis," in *AIP Conference Proceedings*, 2024, vol. 3009, no. 1: AIP Publishing.

[127] Z. Lu, "Sql injection detection using Naïve Bayes classifier: A probabilistic approach for web application security," in *ITM Web of Conferences*, 2025, vol. 70: EDP Sciences, p. 04016.

[128] R. Shakya, D. Dharmaratne, and M. Sandirigama, "Detection of SQL Injection Attacks Using Machine Learning Techniques," in *2024 International Conference on Electrical, Communication and Computer Engineering (ICECCE)*, 2024: IEEE, pp. 1-6.

[129] M. Irfa'issurur and B. P. Josaphat, "Machine Learning for Cybersecurity: Web Attack Detection (Brute Force, XSS, SQL Injection)," *InPrime: Indonesian Journal of Pure and Applied Mathematics*, vol. 7, no. 1, pp. 1-15, 2025.

[130] A. H. Farhan and R. F. Hasan, "Detection SQL injection attacks against web application by using K-nearest neighbors with principal component analysis," in *Proceedings of Data Analytics and Management: ICDAM 2022*: Springer, 2023, pp. 631-642.

[131] Y. Yuan, Y. Lu, K. Zhu, H. Huang, L. Yu, and J. Zhao, "A static detection method for sql injection vulnerability based on program transformation," *Applied Sciences*, vol. 13, no. 21, p. 11763, 2023.

[132] A. Tipacti Garcia, "Static Security Testing Models in Inefficiency Reduction Identification of SQL Injection in Web Applications," *Revista Científica y Tecnológica UPSE (RCTU)*, vol. 11, pp. 130-144, 2024.

[133] R. Al Mallah and A. Quintero, "Adversarial Threats and Defense Mechanisms in Machine Learning-Based SQL Injection Detection: A Security Analysis," in *2025 International Conference on Computing, Networking and Communications (ICNC)*, 2025: IEEE, pp. 180-184.

[134] P. Nunes, J. Fonseca, and M. Vieira, "Blending Static and Dynamic Analysis for Web Application Vulnerability Detection: Methodology and Case Study," *IEEE Access*, 2024.

[135] E. Ogawa, T. Yamazaki, and R. Shioya, "Dynamic Controllability Analysis for Preventing Injection Attacks," in *2024 IEEE 29th Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2024: IEEE, pp. 131-142.

[136] G. K. Borana, N. H. Vishwakarma, S. Tamboli, P. Sharma, M. M. Mukhedkar, and N. A. Dawande, "Defending the Digital World: A Comprehensive Guide Against SQL Injection Threats," in *2024 Second International Conference on Inventive Computing and Informatics (ICICI)*, 2024: IEEE, pp. 707-714.

[137] K. Ross, M. Moh, T.-S. Moh, and J. Yao, "Multi-source data analysis and evaluation of machine learning techniques for SQL injection detection," in *Proceedings of the 2018 ACM Southeast Conference*, 2018, pp. 1-8.

[138] M. Hasan, Z. Balbahaith, and M. Tarique, "Detection of SQL injection attacks: a machine learning approach," in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 2019: IEEE, pp. 1-6.

[139] K. Kuroki, Y. Kanemoto, K. Aoki, Y. Noguchi, and M. Nishigaki, "Attack intention estimation based on syntax analysis and dynamic analysis for SQL injection," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2020: IEEE, pp. 1510-1515.

[140] M. Shetty and C. Nalawade, "Hybrid approach for Detection and Analysis of SQL and XSS vulnerabilities," *International Journal of Engineering Trends and Technology*, vol. 59, pp. 37-41, 2018.

[141] H. Mutai, "Hybrid Multi-Agents System Vulnerability Scanner For Detecting SQL Injection Attacks In Web Applications," University of Nairobi, 2019.

[142] O. C. Abikoye, A. Abubakar, A. H. Dokoro, O. N. Akande, and A. A. Kayode, "A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm," *EURASIP Journal on Information Security*, vol. 2020, pp. 1-14, 2020.

[143] I. Jemal, O. Cheikhrouhou, H. Hamam, and A. Mahfoudhi, "Sql injection attack detection and prevention techniques using machine learning," *International Journal of Applied Engineering Research*, vol. 15, no. 6, pp. 569-580, 2020.

[144] S. A. Krishnan, A. N. Sabu, P. P. Sajan, and A. Sreedeep, "SQL injection detection using machine learning," *Revista Geintec-Gestao Inovacao E Tecnologias*, vol. 11, no. 3, pp. 300-310, 2021.

[145] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of sql injection attack using machine learning techniques: a systematic literature review," *Journal of Cybersecurity and Privacy*, vol. 2, no. 4, pp. 764-777, 2022.

[146] B. Brindavathi, A. Karrothu, and C. Anilkumar, "An Analysis of AI-based SQL Injection (SQLi) Attack Detection," in *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, 2023: IEEE, pp. 31-35.

[147] M. A. Imtiaz, M. Z. Ahmed, F. Khalid, A. M. Muhammad, A. Iqbal, and H. Siddique, "Data Exploration with SQL: A Machine Learning Based End-to-End Prediction and Data Security Framework for the Detection of Attack in Emerging Cloud Computing Databases and Integrated Paradigms: Analysis onTaxonomy, Challenges, and Opportunities," *Spectrum of engineering sciences*, vol. 3, no. 2, pp. 272-303, 2025.

[148] I. S. Crespo-Martínez, A. Campazas-Vega, Á. M. Guerrero-Higueras, V. Riego-DelCastillo, C. Álvarez-Aparicio, and C. Fernández-Llamas, "SQL injection attack detection in network flow data," 2023.

[149] N. M. Sheykhkanloo, "A learning-based neural network model for the detection and classification of SQL injection attacks," in *Deep Learning and Neural Networks: Concepts, Methodologies, Tools, and Applications*: IGI Global, 2020, pp. 450-475.

[150] Y. Janardhanan, "A novel method for SQL Injection Prevention," 2020.

[151] L. Sjöström, "Detecting SQL Injection Attacks in VoIP using Real-time Deep Packet Inspection: Can a Deep Packet Inspection Firewall Detect SQL Injection Attacks on SIP Traffic with Reasonable Performance?," ed, 2019.