International Journal on Emerging Technologies **16(2): 120-139(2025)**

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Web Vulnerabilities using Machine Learning for Prevention and Detection: A Critical Review

Oduleye B.E.*, Asuquo P. and Bliss U.S.
Department of Computer Engineering, University of Uyo, Nigeria.

(Corresponding author: Oduleye B.E.)*
(Received 10 June 2025, Revised 06 August 2025, Accepted 25 August 2025)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Deep learning has emerged as a powerful tool in addressing the complex and evolving nature of cybersecurity threats, particularly in the realm of web security. This paper explores the challenges and opportunities associated with integrating deep learning techniques into web security measures. Web vulnerabilities, ranging from injection attacks to authentication flaws, pose significant risks to digital systems and necessitate comprehensive cybersecurity strategies. Deep learning offers promising avenues for enhancing detection accuracy and efficiency by automatically learning from vast datasets. However, the integration of deep learning into web vulnerability identification encounters challenges such as data availability, interpretability, and susceptibility to adversarial attacks. Despite these challenges, recent advances in deep learning have revolutionized web security, enabling the development of state-of-the-art models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs). These models exhibit remarkable performance in detecting malware, identifying anomalies, and preventing intrusions, thus mitigating potential security breaches. Moreover, emerging techniques such as generative adversarial networks (GANs) and interdisciplinary approaches incorporating natural language processing (NLP) and graph analysis are shaping the future of web security. However, challenges remain, including data requirements, computational resources, and vulnerability to adversarial attacks. Addressing these challenges requires innovative strategies like transfer learning, federated learning, and privacy-preserving approaches to enhance the generalization capabilities and resilience of deep learning models in web security applications. By leveraging the strengths of deep learning techniques and traditional methods, comprehensive web security solutions can be designed to effectively mitigate risks and safeguard against evolving threats in today's interconnected digital landscape.

Keywords: Adversarial Attacks, Cybersecurity, Deep Learning, Malware Detection, Vulnerability Identification.

INTRODUCTION

Web vulnerabilities relate to defects or flaws in web applications, websites, or web servers that can be exploited by hostile actors to undermine the security of the system or the data it holds (Nirmal *et al.*, 2018; Naveen & Mishra 2020). These vulnerabilities can occur at multiple levels of the web architecture, including the application layer, server layer, and network layer (Priyanka & Smruthi 2020; Al-Kahla *et al.*, 2021; Bararia & Choudhary 2023). They pose considerable hazards to the security, integrity, and availability of information stored or sent via the web.

One of the most popular types of web vulnerabilities is the injection attack, when attackers inject malicious code or commands into input fields or parameters of web applications to change their behavior or access sensitive information (Gu *et al.*, 2020). Examples include SQL injection, cross-site scripting (XSS), and command injection. Another prominent sort of online vulnerability is authentication and session management problems, in the study of (Hassan *et al.*, 2018), they found 56% to be vulnerable to these flaws. These vulnerabilities allow attackers to bypass authentication systems, hijack user sessions, or escalate privileges to gain unauthorized access to web sites or sensitive data. Web vulnerabilities can also come from unsafe direct object references, security misconfigurations, cross-site request forgery (CSRF), and sensitive data exposure, among others (Bararia & Choudhary 2023). Each form of vulnerability has its distinct hazards and potential influence on the security posture of web systems.

The discovery and treatment of online vulnerabilities are key components of cybersecurity and web application development techniques. Failure to address these vulnerabilities can lead to serious consequences, including data breaches, financial losses, reputational damage, and legal penalties. Moreover, in today's interconnected digital landscape, where web applications serve as gateways to sensitive information and critical systems, the impact of web vulnerabilities extends beyond individual websites to affect entire organizations and their stakeholders (Anwar *et al.*, 2018; Mishra *et al.*, 2021) Identifying online vulnerabilities is crucial for preserving the security and integrity of web-based systems and protecting sensitive information from unauthorized access, alteration, or disclosure.

Proactive vulnerability detection and mitigation solutions are critical for enterprises to manage security risks, comply with regulatory requirements, safeguard user privacy, and preserve business continuity. Web vulnerabilities expose enterprises to different threats, such as data breaches, malware infections, and service disruptions (Tang *et al.*, 2019). By proactively identifying and addressing these vulnerabilities, companies can lower the likelihood of successful cyberattacks and limit the impact on their operations and stakeholders. Compliance with rules like GDPR, PCI DSS, and HIPAA often needs robust security measures (Chhetri *et al.*, 2022). Failure to identify and address web vulnerabilities can expose user privacy, erode confidence, and ruin organizations' brand. Web-based services and applications are key components of modern company operations, providing e-commerce, online banking, communication, and collaboration. Vulnerabilities that interrupt web systems can result in financial losses, productivity decreases, and consumer unhappiness. Therefore, identifying and mitigating vulnerabilities is critical for assuring continued web service functioning and sustaining business continuity.

Traditional approaches to web vulnerability identification comprise a variety of strategies and processes targeted at finding and mitigating vulnerabilities in web applications and systems. Web-based risks have changed over time, with traditional methods such as manual code review, vulnerability scanning tools, penetration testing, web application firewalls (WAFs), and secure coding practices (Asaduzzaman *et al.*, 2020). Manual code review involves human professionals carefully evaluating online application source code to discover potential vulnerabilities, such as injection holes, authentication problems, and vulnerable configuration settings (Marashdih *et al.*, 2018). However, it is time-consuming and resource-intensive, making it unfeasible for large-scale applications or frequent code modifications.

Vulnerability scanning solutions automatically scan online applications and servers for known vulnerabilities, misconfigurations, and security issues. They employ predefined signatures and attack patterns to identify common vulnerabilities, such as SQL injection, cross-site scripting (XSS), and directory traversal (Amankwah *et al.*, 2020; Alazmi & De Leon 2022). However, they may produce false positives or miss developing risks. Penetration testing, often known as ethical hacking, mimics real-world intrusions to assess the security posture of web services and systems. It gives useful insights into the effectiveness of security controls and helps organizations prioritize remedial efforts based on the severity of identified vulnerabilities.

Web application firewalls (WAFs) monitor and filter HTTP traffic between web applications and clients, assessing incoming requests and responses for suspicious patterns, abnormalities, or known attack signatures (Muzaki *et al.*, 2020). While they offer real-time security and threat intelligence, they may bring performance overhead and require regular upgrades. Secure coding techniques comprise following established rules, best practices, and coding standards to design online applications resilient to common vulnerabilities and security threats (Anis *et al.*, 2018; Nembhard *et al.*, 2019). Integrating security into the software development lifecycle (SDLC) helps minimize vulnerabilities during development and save remedial costs.

Traditional approaches to web vulnerability identification comprise a number of techniques, including manual code inspection, vulnerability scanning, penetration testing, web application firewalls, and secure coding practices. While each strategy has its strengths and limits, organizations can increase their security posture by adopting a holistic and layered approach to web application security that integrates numerous techniques and strategies.

Deep learning, a subset of machine learning, has emerged as a promising tool for addressing difficult cybersecurity concerns, including web security. Deep learning algorithms, inspired by the structure and function of the human brain, can automatically learn and extract subtle patterns and features from massive volumes of data, enabling them to detect and mitigate web-based risks efficiently (Vinayakumar *et al.*, 2019; Tian *et al.*, 2020). Deep learning algorithms are used in web security to detect and categorize malware, abnormalities, intrusions, and phishing attempts. These techniques can be used to file signatures, behavior patterns, and network traffic to discover new threats with high accuracy. They can also detect aberrant behavior in web traffic, user sessions, and system logs, indicating probable security breaches or unauthorized access attempts (Vinayakumar *et al.*, 2019). These models can also detect and prevent breaches, exploits, and cyberattacks affecting web servers, apps, and databases. They can recognize patterns linked with known attack vectors and developing threats in real-time utilizing deep neural networks and convolutional architectures. Deep learning models can identify phishing attempts and fraudulent activity by evaluating email headers, message content, and user interactions. These algorithms may categorize emails as valid or suspect based on linguistic signals, contextual information, and sender reputation.

Deep learning is a significant technique in web security, giving scalability, adaptability, and accuracy. It can process massive volumes of data and detect threats without manual involvement, making it appropriate for large-scale web environments and dynamic attack landscapes (Tian *et al.*, 2020). It can learn from varied datasets, detecting fresh threats and zero-day vulnerabilities with minimal human supervision. Compared to traditional rule-based or signature-based techniques, deep learning minimizes false positives and false negatives. Deep learning in web security confronts issues such as data quality and quantity, interpretation, and adversarial assaults (Liu *et al.*, 2021). Large, diversified, and labeled datasets are essential for effective learning, which can be challenging to get due to privacy concerns, data shortages, and class imbalance. Interpretability is also an issue, as deep learning models are frequently regarded black-box systems, making it difficult to understand their decision-making process and rationale (Ali *et al.*, 2024). Therefore, robustness and resilience are critical in deep learning-based web security systems.

Web vulnerabilities refer to flaws in online applications, websites, or servers that criminal actors exploit to gain access to digital systems. These flaws exist at several levels, including the application, server, and network layers (Gupta & Gupta 2015; Priyanka & Smruthi 2020). Recognizing the type and severity of these vulnerabilities is critical for implementing effective cybersecurity solutions. Web vulnerabilities include injection attacks, authentication problems, unsafe direct object references, security setup errors, and cross-site request forgery (Asati *et al.*, 2024). These flaws jeopardize data integrity, confidentiality, and availability, resulting in breaches, malware infections, service disruptions, reputational harm, and legal liabilities. Deep learning, a type of machine learning, has emerged as an effective technique in cybersecurity. It provides benefits like scalability, adaptability, and accuracy in identifying and managing various cyber threats such as malware, anomalies, intrusions, and phishing attacks (Mahdavifar & Ghorbani 2019). Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), scan large datasets for patterns that indicate security threats. However, issues like data quality, interpretability, and vulnerability to adversarial attacks impede the wider use of deep learning in online security. Combining classical approaches with deep learning techniques can improve detection performance while resolving constraints. Feature extraction, anomaly detection, ensemble learning, transfer learning, and domain adaptation are examples of deep learning-based vulnerability identification strategies (Bamane *et al.*, 2023). These approaches use deep learning's capabilities to enhance accuracy and efficiency in discovering web vulnerabilities, thereby improving overall web security measures.

Web application security architecture encompasses various layers and components designed to protect web applications from potential threats and vulnerabilities (Aborujilah *et al.*, 2022). A well-designed security architecture ensures the confidentiality, integrity, and availability of data and services provided by the application as shown in (Fig. 1).

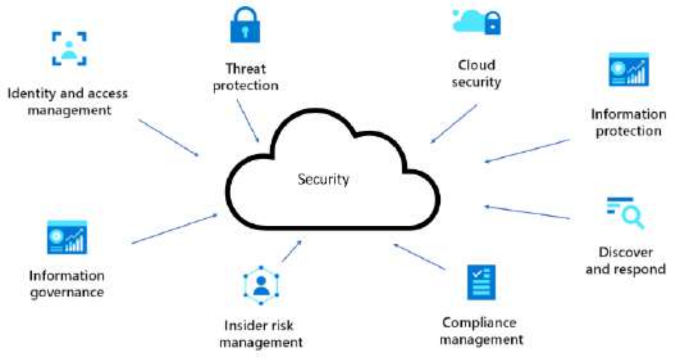

Fig. 1. Web Security Architecture.

Deep learning shows potential for increasing web security by providing automated threat identification, anomaly detection, intrusion detection, and phishing detection in web-based systems. By using the capabilities of deep learning, organizations may increase their defenses against evolving cyber threats and safeguard their web apps, systems, and users against exploitation and compromise. The detection of web vulnerabilities is crucial for maintaining the security, privacy, and continuity of web-based services. By adopting proactive security measures, firms can manage risks, comply with legal obligations, secure user privacy, and uphold the trust and confidence of their stakeholders.

WEB VULNERABILITIES

Web vulnerabilities indicate serious weaknesses or defects inside web apps, websites, or web servers that bad actors might exploit to jeopardize the security and integrity of digital systems. These vulnerabilities can occur at many levels of the web architecture, including the application layer, server layer, and network layer (Nirmal *et al.*, 2018; Dwivedi 2021). Understanding the nature, forms, and impact of web vulnerabilities is vital for successful cybersecurity measures and the protection of sensitive information in today's linked digital landscape.

Web vulnerabilities comprise a wide range of security issues that expose web applications and systems to exploitation by attackers. These vulnerabilities might develop due to programming errors, misconfigurations, design flaws, or insufficient security measures within web-based technology.

Common examples of web vulnerabilities include:

Injection Attacks: Injection attacks occur when malicious code or commands are injected into input fields or parameters of online applications to change their behavior or obtain unauthorized access to sensitive data (Bararia & Choudhary 2023). Examples include SQL injection, where attackers insert harmful SQL queries into input fields to modify databases, and XSS, where attackers inject malicious scripts into web pages seen by other users.

Authentication and Session Management Flaws: Authentication and session management vulnerabilities allow attackers to bypass authentication methods, hijack user sessions, or escalate privileges to obtain unauthorized access to web sites or sensitive data (Hassan *et al.*, 2018). Weak password restrictions, insecure session tokens, and inadequate access controls are common instances of authentication and session management issues.

Insecure Direct Object References (IDOR): Insecure direct object references occur when web applications disclose internal objects, files, or resources directly to users without sufficient authorization checks (Pratama & Rhusuli 2022). Attackers can exploit IDOR vulnerabilities to obtain sensitive data or perform unwanted operations by changing object references in URLs or request parameters.

Security Misconfigurations: Security misconfigurations arise when web servers, databases, or application frameworks are poorly configured, leaving them exposed to exploitation (Liu *et al.*, 2019). Common misconfigurations include default settings, open ports, unneeded services, and obsolete software versions, which can be exploited by attackers to obtain unauthorized access or jeopardize system integrity.

Cross-Site Request Forgery (CSRF): CSRF attacks occur when attackers deceive authenticated users into unintentionally executing harmful activities on web applications where they have an active session (Kour, 2020). By leveraging the trust relationship between the user and the online application, attackers can undertake unauthorized operations, such as moving funds, modifying account settings, or submitting fraudulent transactions.

Web vulnerabilities jeopardize data integrity, confidentiality, and availability, prompting breaches, malware, interruptions, reputation damage, and legal concerns (Bararia & Choudhary 2023) Data breaches can result in illegal access, disclosure, or theft of sensitive information, leading to financial, legal, and reputational implications for companies. Malware infestations can halt corporate operations, ruin data, and risk the confidentiality of sensitive information. Service disruptions can originate from denial-of-service attacks, server breakdowns, and application failures, impacting the availability and operation of web-based services and applications. Public exposure of web vulnerabilities or security breaches can harm the reputation and credibility of organizations, undermining customer confidence and investment faith. Legal obligations may develop for non-compliance with data protection legislation, industry regulations, or contractual responsibilities. Regulatory frameworks such as the General Data Protection Regulation (GDPR), Payment Card Industry Data Security Standard (PCI DSS), and Health Insurance Portability and Accountability Act (HIPAA) impose strict requirements for safeguarding sensitive data and protecting user privacy in web environments.

The OWASP Top 10 list typically includes the most critical security risks to web applications based on community feedback and data analysis as shown in Table 1 to table 10.

Injection: Injection attacks occur when untrusted data is sent to an interpreter as part of a command or query. Types of injection attacks include SQL injection, NoSQL injection, OS command injection, and LDAP injection. Attackers exploit vulnerabilities in input validation to inject malicious code, leading to data breaches, data loss, and unauthorized access

Table 1: Injection.

Type	Description	Impact
SQL Injection	Allows attackers to execute arbitrary SQL	Data loss, unauthorized access, data leakage
NoSQL Injection	Similar to SQL injection but targets NoSQL databases	Data loss, unauthorized access, data leakage
OS Command Injection	Exploits vulnerabilities in command execution	Remote code execution, system compromise
LDAP Injection	Attacks LDAP (Lightweight Directory Access Protocol) queries	Unauthorized access to directories, data leakage

Broken Authentication: This vulnerability occurs when authentication and session management mechanisms are implemented incorrectly. Attackers exploit weaknesses such as weak passwords, session fixation, and session hijacking to gain unauthorized access to user accounts.

Table 2: Broken Authentication.

Type	Description	Impact
Weak Passwords	Users choose easily guessable or common passwords	Account takeover, unauthorized access
Session Fixation	Attackers set session identifiers for victims	Session hijacking, unauthorized access
Session Hijacking	Attackers steal session cookies or session IDs	Account takeover, unauthorized access
Credential Stuffing	Attackers use leaked credentials to gain access	Account takeover, unauthorized access

Sensitive Data Exposure: This vulnerability involves the exposure of sensitive data such as passwords, credit card numbers, or personal information due to weak encryption, insecure storage, or improper transmission.

XML External Entities (XXE): XXE attacks occur when an attacker can upload or modify XML input containing external entity references. These entities can be used to disclose internal files, execute remote code, or perform denial of service attacks.

Broken Access Control: This vulnerability occurs when restrictions on what authenticated users are allowed to do are not properly enforced. Attackers exploit these weaknesses to gain unauthorized access to resources or perform actions beyond their privileges.

Security Misconfiguration: This vulnerability occurs due to insecure configuration settings, default configurations, or unnecessary features enabled on the web application, server, or platform.

Table 3: Sensitive Data Exposure.

Type	Description	Impact
Weak Encryption	Data is encrypted using weak algorithms or keys	Data theft, unauthorized access
Insecure Storage	Data is stored in plaintext or with weak protection	Data theft, unauthorized access
Insecure Transmission	Data is transmitted over unencrypted channels	Data interception, unauthorized access
Improper Access Control	Inadequate access controls on sensitive data	Unauthorized access, data leakage

Table 4: XXE.

Type	Description	Impact
Entity Expansion	Attackers expand external entities to read files or execute commands	Data theft, remote code execution
Entity Injection	Insertion of malicious entities into XML data	Data leakage, remote code execution
Entity Manipulation	Manipulation of existing entities for malicious purposes	Data theft, unauthorized access
Blind XXE	Attackers exploit XXE without receiving direct responses	Data theft, remote code execution

Table 5: Broken Access Control.

Type	Description	Impact
Insecure Direct Object References	Accessing resources directly without proper authorization	Unauthorized access, data leakage
Missing Function Level Access Control	Access control checks are missing or not properly implemented	Unauthorized access, privilege escalation
Overly Permissive Access Controls	Permissions are set too broadly, allowing excessive access	Unauthorized access, data leakage
Insecure Access Control Methods	Weaknesses in access control mechanisms	Unauthorized access, data leakage

Table 6: Security Misconfiguration.

Type	Description	Impact
Default Accounts and Passwords	Use of default credentials or accounts	Unauthorized access, data breach
Unnecessary Services	Running unnecessary services or components	Attack surface expansion, potential vulnerabilities
Open Cloud Buckets	Cloud storage buckets are left open or misconfigured	Data exposure, unauthorized access
Misconfigured Security Headers	Improperly configured HTTP headers	Cross-site scripting, data leakage

Cross-Site Scripting (XSS): XSS vulnerabilities occur when attackers inject malicious scripts into web pages viewed by other users. These scripts can steal sensitive data, hijack sessions, or deface websites.

Insecure Deserialization: This vulnerability occurs when untrusted data is deserialized by a program, leading to remote code execution or other malicious actions.

Using Components with Known Vulnerabilities: This vulnerability arises when outdated or vulnerable components, libraries, or frameworks are used in a web application.

Insufficient Logging & Monitoring: This vulnerability occurs when an application lacks proper logging and monitoring mechanisms, making it difficult to detect and respond to security incidents.

Table 7: XSS.

Type	Description	Impact
Reflected XSS	Injected script reflects off a web application	Data theft, session hijacking
Stored XSS	Injected script is stored on the server and executed when accessed	Data theft, session hijacking
DOM-based XSS	Client-side vulnerabilities in JavaScript code	Data theft, session hijacking
Blind XSS	Attacker cannot directly observe the effects of the payload	Data theft, session hijacking

Table 8: Insecure Deserialization.

Type	Description	Impact
Remote Code Execution	Allows attackers to execute arbitrary code	System compromise, unauthorized access
Data Tampering	Modifies serialized data to alter application behavior	Data corruption, system compromise
Denial of Service	Exploits deserialization vulnerabilities to disrupt services	Service interruption, system compromise
Session Hijacking	Manipulates serialized data to hijack sessions	Unauthorized access, session hijacking

Table 9: Using Components with Known Vulnerabilities.

Type	Description	Impact
Outdated Libraries	Use of older versions of libraries with known vulnerabilities	Exploitation of known vulnerabilities, system compromise
Unpatched Software	Failure to apply security patches and updates to components	Exploitation of known vulnerabilities, system compromise
Vulnerable Dependencies	Dependencies with known security flaws	Exploitation of known vulnerabilities, system compromise
Third-party APIs	Integration with insecure or untrustworthy APIs	Data leakage, unauthorized access

Table 10: Insufficient Logging & Monitoring.

Type	Description	Impact
Lack of Monitoring	Failure to monitor system activities and security events	Missed security incidents, delayed response
Inadequate Logging	Insufficient logging of events and actions	Difficulty in detecting and investigating security breaches
No Alerts or Notifications	Absence of alerts or notifications for suspicious activities	Delayed response to security incidents
Limited Retention	Short retention periods for logs and monitoring data	Difficulty in forensic analysis, incident response

These tables provide a comprehensive overview of each vulnerability type, including their classifications, descriptions, and impacts. Developers and security professionals need to understand these vulnerabilities and implement appropriate measures to mitigate the associated risks in web applications.

The OWASP Top 10 list for 2021 incorporates various adjustments and new categories to better represent current trends and difficulties in online application security, as described in (Fig. 2) (*OWASP Top Ten | OWASP Foundation*, n.d.).

A01:2021-Broken Access Control: This category has risen up from the fifth place, demonstrating its growing occurrence in web application security vulnerabilities. Broken access control refers to inadequate limits on what authorized users may perform within an application. With 94% of apps assessed for some type of failed access restriction, its ubiquitous presence is obvious.

A02:2021-Cryptographic Failures: Previously known as Sensitive Data disclosure, this category now focuses on errors linked to cryptography, which often lead to sensitive data disclosure or system penetration. By tackling the core causes rather than merely the symptoms, this category tries to promote cryptographic practices in online applications.

A03:2021-Injection: Injection vulnerabilities have dropped down to the third place but remain a serious risk. Ninety-four percent of apps were checked for some sort of injection, underlining the necessity to limit injection risks, which can lead to numerous assaults such as SQL injection and command injection.

A04:2021-Insecure Design: This new category emphasizes dangers connected to design errors in online applications. By advocating approaches like as threat modeling, safe design patterns, and reference architectures, it seeks to address vulnerabilities at the design stage, eventually enhancing overall security posture.

A05:2021-Security Misconfiguration: This category has moved up from #6 in the previous edition, showing the increased significance of resolving misconfigurations in highly changeable software. It now includes the old category for XML External Entities (XXE), highlighting the need to resolve misconfigurations that might lead to security breaches.

A06:2021-Vulnerable and Outdated Components: Previously named Using Components with Known Vulnerabilities, this category stresses the complexity of testing and analyzing risks associated to vulnerable and obsolete components. By stressing the necessity of addressing these components, it tries to prevent severe security dangers to online applications.

A07:2021-Identification and Authentication Failures: Formerly known as Broken Authentication, this category now contains CWEs more linked to identity issues. This reflects changes in standardized frameworks and authentication methods, underlining the significance of effective identity and authentication systems.

A08:2021-Software and Data Integrity Failures: Another new category concentrating on making assumptions related to software updates, critical data, and CI/CD pipelines without validating integrity. This underlines the necessity of preserving the integrity of software and data throughout the development and deployment process to prevent possible security concerns.

A09:2021-Security Logging and Monitoring Failures: Previously Insufficient Logging & Monitoring, this category is broadened to encompass more sorts of failures. It emphasizes the problems in testing for and fixing faults in logging and monitoring, which can directly affect incident response and forensics.

A10:2021-Server-Side Request Forgery: Added from the Top 10 community survey, this category reflects circumstances where security community members underline its relevance. Despite not being prominently displayed in the data, it illustrates the possibility of attackers using server-side request forgery vulnerabilities to undertake illegal operations on behalf of the server.

These updates and additions in the OWASP Top 10 list reflect the dynamic environment of web application security threats and underline the significance of addressing vulnerabilities at various phases of the software development lifecycle.

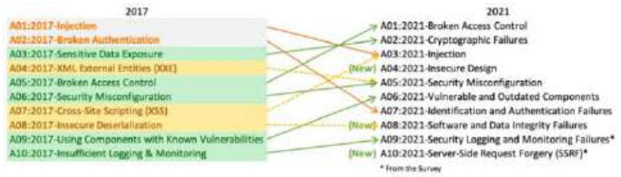
2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

Fig. 2. OWASP Top 10 list for 2021 (*OWASP Top Ten | OWASP Foundation*, n.d.).

Web vulnerabilities have impacted several prominent firms, resulting in substantial implications such as data leaks, financial losses, and brand harm. Below is a concise overview of several noteworthy occurrences: In 2017, Equifax, a major credit reporting agency, suffered a significant data leak as a result of a vulnerability in the Apache Struts framework. Malicious individuals used this vulnerability to illicitly obtain access to highly confidential personal data of more than 147 million individuals, encompassing their names, Social Security numbers, birth dates, and residences. The breach resulted in a broad public outcry and legal consequences, such as several lawsuits, inquiries conducted by regulatory bodies, and a substantial erosion of customer confidence (Fruhlinger, 2023). Equifax incurred financial losses due to expenses related to litigation bills, settlements, and the negative impact on the company image.

Yahoo experienced many data breaches from 2013 to 2016, impacting billions of user accounts. These hacks encompassed a range of vulnerabilities, such as SQL injection and counterfeit cookies. Unauthorized individuals successfully obtained users' data, including their names, email addresses, phone numbers, and passwords that were encoded for security purposes. The security failures inflicted significant harm on Yahoo's brand and eroded its standing as a trustworthy provider of safe services. Yahoo suffered legal ramifications, including regulatory probes and class-action lawsuits (*Takeaways From Yahoo's 500-Million-Account Breach*, n.d.). The episodes also had ramifications for Yahoo's acquisition by Verizon, resulting in a decreased acquisition price.

In 2013, Target had a data breach stemming from malware planted on their payment card system. The virus exploited flaws in Target's network architecture, allowing attackers to steal credit card data and personal information from millions of consumers. The breach harmed Target's brand and reduced consumer trust, resulting to a fall in sales and stock prices (Jones, 2022). Target sustained enormous financial losses owing to lawsuit settlements, regulatory fines, and expenditures connected with cybersecurity upgrades and consumer remediation initiatives.

Marriott International experienced a significant data breach in 2018, impacting around 500 million guests. The attack came from weaknesses in Marriott's Starwood guest reservation database, which had been exposed since 2014. Attackers got illegal access to guests' personal information, including names, contact details, passport numbers, and credit card data. The breach has severe ramifications for Marriott, including regulatory inquiries, litigation, and brand harm (Fruhlinger, 2020). Marriott suffered large expenditures connected to legal settlements, cybersecurity enhancements, and compensation for affected consumers.

In 2014, Sony Pictures Entertainment faced a high-profile hack ascribed to a group known as the Guardians of Peace (GOP). The hack exploited vulnerabilities in Sony's network architecture and exposed critical company data, including internal emails, personnel information, and unreleased films. The hack caused severe interruption to Sony's business, resulting to disclosed private information, reputational harm, and financial losses (Young, 2021). Sony experienced criticism from stakeholders, regulatory scrutiny, and legal challenges, including lawsuits brought by workers and shareholders.

These instances underscore the significance of comprehensive cybersecurity measures and proactive risk management to guard against online vulnerabilities and limit the possible effect on enterprises.

CUTTING-EDGE STRATEGIES FOR ADDRESSING WEB APPLICATION VULNERABILITIES: STATE-OF-THE-ART APPROACHES

Addressing each vulnerability in web applications requires a multifaceted approach that combines technical solutions, best practices, and ongoing vigilance.

Broken Access Control vulnerabilities offer a danger to the security of online applications, allowing unauthorized users to gain access to sensitive resources or do actions outside their rights. To overcome this issue, state-of-the-art techniques rely on developing comprehensive access control systems. One alternative is to implement Role-Based Access Control (RBAC) or Attribute-Based Access Control (ABAC) systems. RBAC gives responsibilities to users, and access permissions are issued based on these roles, simplifying administration and guaranteeing that users only have access to the resources essential for their jobs. ABAC, on the other hand, permits access based on qualities linked with users, resources, and environmental factors, giving more flexible and fine-grained control over access rights (Bansal, 2023). Access Control Lists (ACLs) can be deployed to set access rights for particular resources. ACLs let administrators to designate which users or groups have read, write, or execute rights for specified resources, boosting access control granularity. Proper session management is also vital in reducing Broken Access Control issues (Rana & Mishra 2021). This includes incorporating methods such as session expiration, ensuring that sessions automatically end after a specified time of inactivity to avoid unwanted access. Session invalidation at logout is critical to withdraw access permissions promptly when a user logs out, lowering the possibility of session hijacking. Secure session storage is equally critical to safeguard session data from unwanted access or alteration, employing techniques such as encryption and secure cookie characteristics. By adopting these state-of-the-art measures, companies may successfully minimize Broken Access Control vulnerabilities in their online applications, ensuring that access to resources is correctly restricted and unauthorized access attempts are blocked. However, it's vital to regularly monitor and update access control measures to respond to emerging security risks and guarantee comprehensive protection against illegal access.

Adopting strong cryptographic methods and key lengths required by industry standards increases the security of data encryption. Algorithms such as AES (Advanced Encryption Standard) with proper key lengths provide comprehensive security against cryptographic assaults (Althamir et al., 2023). Implementing safe key management procedures is vital to avoid unwanted access to encryption keys. This entails integrating key rotation methods to routinely swap encryption keys and defend against key leakage using secure storage and transmission protocols (Abu-Faraj *et al.*, 2022). Leveraging hardware security modules (HSMs) or key management services (KMS) supplied by cloud providers can further increase key security. Using cryptographic libraries and frameworks given by credible sources is crucial for safe encryption and decryption procedures. These libraries frequently undergo extensive security assessments and upgrades to address emerging threats and vulnerabilities. By adopting well-established cryptographic libraries such as OpenSSL or Bouncy Castle, developers may limit the risk of implementation errors and assure compliance with industry best practices. Additionally, incorporating features like digital signatures and message authentication codes (MACs) utilizing cryptographic primitives promotes data integrity and authenticity (Abu-Faraj *et al*., 2022).

Injection vulnerabilities in online applications needs extensive safeguards to prevent attackers from exploiting vulnerabilities such as SQL injection and cross-site scripting (XSS). Implementing parameterized queries or prepared statements is a fundamental strategy to prevent SQL injection attacks. These strategies ensure that user-supplied input is processed as data rather than executable code, considerably lowering the danger of injection vulnerabilities (Abdullayev & Chauhan 2023). By isolating SQL instructions from user input, parameterized queries prevent attackers from manipulating SQL queries to execute harmful commands. Implementing input validation and output encoding is crucial to combat many sorts of injection threats, including XSS. Input validation analyzes user input against specified rules to verify it fulfills anticipated criteria, hence avoiding fraudulent input from being handled. Output encoding, on the other hand, entails encoding user-generated material before displaying it on web pages, preventing dangerous scripts from being performed in the context of the application (Abdullayev & Chauhan 2023). Employing Object-Relational Mapping (ORM) frameworks and query builders can assist eliminate injection vulnerabilities by automatically cleaning inputs and providing safe database queries. These frameworks encapsulate database interactions and manage parameterization internally, lowering the danger of injection attacks. By adopting ORM frameworks like as Hibernate in Java or Sequelize in Node.js, developers may decrease the possibility of creating injection vulnerabilities in their code.

Performing threat modeling exercises is critical for methodically detecting potential design defects and security vulnerabilities. By examining the application's design, data flow, and possible risks, developers may predict and prioritize security protections. Threat modeling helps ensuring that security considerations are included into the design phase, allowing for more effective risk reduction. Applying secure design concepts such as the principle of least privilege and defense-in-depth is critical for minimizing design-related vulnerabilities (Purba *et al.*, 2023). The notion of least privilege argues for providing users just the minimum rights necessary to complete their duties, decreasing the attack surface and limiting the consequences of possible breaches. Defense-in-depth includes building numerous levels of security measures to offer redundancy and resistance against attackers.

Employing safe design patterns and reference designs offers developers with proven solutions to typical security concerns. Secure design patterns incorporate best practices for implementing certain security elements, such as authentication and access control, whereas reference architectures give blueprints for constructing secure and scalable applications (Patil & Thokal 2025). By accessing these tools, developers may guarantee that their apps comply to recognized security standards and avoid frequent mistakes.

Establishing automated tools for configuration management is crucial to enforce consistent and secure configurations across diverse environments, such as development, testing, and production. Tools like Ansible, Puppet, or Chef automate the deployment and configuration of infrastructure components, ensuring that security settings are implemented consistently and lowering the possibility of misconfigurations. Frequent audits and reviews of setups are critical for discovering and remedying any risks. Organizations should undertake frequent audits to check the security posture of their systems, including examining default settings, disabling superfluous services, and ensuring compliance with security policies and standards. Automated scanning solutions can assist discover misconfigurations and provide ongoing monitoring of security settings. Following security best practices and standards published by platform manufacturers and security groups is vital for securing settings efficiently (Shakirov & Karmanov 2020). Platform manufacturers typically offer security rules and advice for setting their systems securely. Organizations should be updated about the latest security warnings and updates and implement patches swiftly to address identified vulnerabilities.

Software composition analysis (SCA) techniques is critical for discovering and tracking risky dependencies in third-party libraries and components. These tools assess the dependencies of a web application and give insights on potential security vulnerabilities existing in the software stack. By incorporating SCA tools into the development and deployment pipelines, businesses may detect susceptible components early in the lifecycle and take relevant repair measures (Zhao *et al.*, 2023). Building a comprehensive patch management procedure is crucial for swiftly implementing security updates and fixes for identified vulnerabilities. Organizations should design methods for analyzing the severity of vulnerabilities, prioritizing fixes based on risk, and releasing updates in a timely way. Automated patch management systems can simplify the process of discovering, testing, and applying updates, decreasing the window of exposure to known vulnerabilities. Monitoring vulnerability databases and security advisories is critical for remaining updated about developing threats and vulnerabilities in third-party components. Organizations should frequently monitor sources such as the National Vulnerability Database (NVD), vendor security advisories, and open-source repositories for information on newly identified vulnerabilities and accessible patches (Chen *et al.*, 2020). By keeping proactive and responsive to developing threats, companies may limit the risk caused by susceptible and outmoded components.

Multi-factor authentication (MFA) adds an extra layer of protection to user authentication procedures by forcing users to give various forms of verification. This often includes combining something the user knows (e.g., a password) with something they have (e.g., a smartphone or hardware token) or something they are (e.g., biometric data). MFA greatly strengthens security by minimizing the chance of unwanted access, even if credentials are stolen. Adopting established authentication protocols such as OAuth and OpenID Connect guarantees safe authentication and authorization operations (ALSaleem & Alshoshan 2021). These protocols provide industry-standard procedures for authentication, authorization, and safe exchange of user credentials across different systems. By utilizing OAuth for permission and OpenID Connect for authentication, developers may design secure authentication processes and safeguard user credentials from interception and modification. Utilizing secure authentication procedures such as password hashing, salting, and password complexity restrictions is vital for avoiding authentication-related risks (Petcu *et al*., 2023). Password hashing and salting methods encrypt user passwords before saving them in the database, making it substantially more complex for attackers to get plaintext passwords in the case of a data breach. Enforcing password complexity criteria, such as minimum length and character variety, further strengthens security by preventing the use of weak or readily guessable passwords.

State-of-the-art techniques focus on establishing code signing and integrity verification mechanisms, leveraging secure development practices, and creating secure CI/CD pipelines. Integrating code signing and integrity verification techniques is vital to secure the integrity of software updates and critical data (Thangavel & Varalakshmi 2020). Code signature includes digitally signing software updates and other executable files using cryptographic keys, allowing users to verify the validity and integrity of the code before installation. Integrity verification technologies, such as checksums or digital signatures, can be used to verify the integrity of essential data, guaranteeing that it has not been tampered with during transit or storage. Employing safe development approaches is critical for preventing integrity-related problems. This involves doing code reviews, performing static code analysis, and adhering to secure coding rules throughout the development process. Code reviews help developers to discover and resolve possible security vulnerabilities and integrity concerns before they are incorporated into the codebase, while static code analysis tools can automatically uncover common coding mistakes and security weaknesses. Adhering to secure coding principles guarantees that developers follow best practices for building secure and resilient code (Thangavel & Varalakshmi 2020). Creating safe CI/CD pipelines with built-in security controls is critical for identifying and preventing tampering with software and data throughout the deployment process. Secure CI/CD pipelines automate the process of developing, testing, and delivering software updates, enabling for quicker and more reliable release cycles. By incorporating security measures like as code signing, integrity checks, and vulnerability scanning into the CI/CD pipeline, businesses can guarantee that only trusted and validated code is sent to production environments.

Providing thorough logging methods is critical to record security-relevant events and actions within the program. This entails capturing crucial security events, such as authentication attempts, access control changes, and unusual activity, to create a thorough audit trail for security analysis and incident response. By documenting essential information with suitable degrees of detail, companies may get insight into possible security events and identify malicious activity. Employing centralized logging solutions and SIEM systems helps firms to collect, correlate, and analyze data effectively. Centralized logging systems collect logs from numerous sources, such as web servers, application frameworks, and network devices, and store them in a centralized repository for simple access and analysis (Hristov *et al.*, 2021). SIEM systems provide additional features for log aggregation, correlation, and alerting, allowing enterprises to discover security risks and abnormalities more effectively. Integrating real-time monitoring and alerting methods is critical for recognizing and responding to security events swiftly (Hristov *et al*., 2021). Real-time monitoring solutions continually monitor security-relevant events and activities, alerting security professionals to possible risks and abnormalities as they occur. By establishing automated alerting systems based on predetermined thresholds and criteria, companies may speed incident response and limit the consequences of security breaches.

Server-side request forgery (SSRF) vulnerabilities in web applications involves comprehensive procedures to check and manage user-supplied information and restrict access to external resources. State-of-the-art techniques emphasize on integrating input validation, stringent server-side restrictions, and steps to prevent attackers from creating harmful requests. Verifying and sanitizing user-supplied information is critical to prevent attackers from creating fraudulent requests and exploiting SSRF vulnerabilities (Kour, 2020). Input validation entails evaluating the format, type, and content of user input to ensure that it complies to specified requirements. Sanitization techniques delete or escape potentially harmful characters or sequences from user input to avoid injection attacks and other types of malicious input. Providing strong server-side input validation is crucial to guarantee that requests are authentic and intended. Server-side input validation checks the integrity and authenticity of incoming requests, ensuring that they originate from trustworthy sources and include accurate data. By imposing tight validation criteria, companies may limit the danger of SSRF attacks and prevent attackers from influencing server-side behavior. Implementing server-side restrictions such as allowlists and denylists helps restrict the sorts of requests that the server may make to external sites. Allowlists describe approved destinations or resources that the server is permitted to visit, whereas denylists block access to known harmful or prohibited destinations. By establishing server-side controls correctly, companies may limit the attack surface and prevent SSRF vulnerabilities from being exploited.

By adopting these state-of-the-art approaches, organizations can effectively mitigate the vulnerabilities listed in the OWASP Top 10 and enhance the security posture of their web applications. Additionally, ongoing education, training, and collaboration within the security community are essential to staying informed about emerging threats and evolving security best practices.

Table 11 provide an overview of the related works in each stage for each vulnerability category, encompassing software design and implementation, software analysis and testing, as well as deployment and monitoring.

Table 11: Comprehensive Overview of Vulnerability Mitigation Strategies in Software Development Lifecycle.

Vulnerability Category	Software Design and Implementation	Software Analysis and Testing	Deployment and Monitoring
Broken Access Control	Implement RBAC or ABAC, Use ACLs for access permissions, Proper session management	Security code reviews, Penetration testing, Static code analysis	Logging and monitoring of access control events, Regular review of access control configurations
Cryptographic Failures	Use strong cryptographic algorithms, Employ secure key management practices	Cryptographic vulnerability assessments, Penetration testing	Logging and monitoring for cryptographic operations, Regular audit of cryptographic protocols
Injection	Use parameterized queries, Employ input validation and output encoding	Code reviews, Dynamic analysis, Penetration testing	Logging and monitoring for injection attempts, Regular review of application logs
Insecure Design	Conduct threat modeling exercises, Follow secure design principles	Architectural risk analysis, Security reviews, Threat modeling	Logging and monitoring for design-related security events, Regular review of architectural diagrams
Security Misconfiguration	Implement automation tools for configuration management, Regularly audit configurations	Configuration analysis, Vulnerability scanning, Configuration audits	Logging and monitoring for configuration changes, Regular review of configuration settings
Vulnerable and Outdated Components	Utilize SCA tools for dependency analysis, Establish patch management process	Dependency analysis, Software composition analysis, Vulnerability scanning	Logging and monitoring for third-party component usage, Regular review and update of third-party dependencies
Identification and Authentication Failures	Implement MFA, Utilize standardized authentication protocols, Employ secure authentication mechanisms	Security assessments, Authentication bypass testing, Vulnerability scanning	Logging and monitoring for authentication events, Regular review of authentication logs
Software and Data Integrity Failures	Implement code signing and integrity verification mechanisms, Utilize secure development practices	Integrity testing, Static analysis, Code review	Logging and monitoring for integrity-related events, Regular review of integrity logs
Security Logging and Monitoring Failures	Implement comprehensive logging mechanisms, Utilize centralized logging solutions, Implement real-time monitoring	Logging and monitoring testing, Penetration testing, Threat modeling	Logging and monitoring for security-relevant events, Regular review of logs
Server-Side Request Forgery	Validate and sanitize user input, Implement strict server-side input validation	Input validation testing, Fuzz testing, Penetration testing	Logging and monitoring for suspicious request patterns, Regular review of request logs

EXPLORING THE LANDSCAPE: STATE-OF-THE-ART MACHINE LEARNING TECHNIQUES IN WEB APPLICATION DEVELOPMENT

Web applications and machine learning techniques have become increasingly intertwined, with machine learning being leveraged to enhance various aspects of web application development, such as personalization, recommendation systems, security, and performance optimization.

Supervised learning stands as a cornerstone in web application development, employing algorithms like Support Vector Machines (SVM), Random Forests, and Gradient Boosting Machines for classification, regression, and ranking tasks. Recent improvements see the incorporation of deep learning models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) into web applications, catering to tasks like image classification, natural language processing, and time-series prediction. However, issues remain, showing gaps that deserve attention. The necessity for big labeled datasets remains a constraint, reducing the scalability and performance of supervised learning algorithms. Developing model interpretability poses a substantial issue, especially with the emergence of complicated deep learning systems (Gupta *et al.*, 2022). Preventing overfitting, particularly in sophisticated deep learning models, remains an ongoing challenge, stressing the demand for strong regularization strategies and architectural alterations to increase generalization capabilities. Addressing these shortcomings is crucial to unlock the full potential of supervised learning in the field of web application development.

Unsupervised learning stands at the forefront of innovation in web application development, where techniques like K-means clustering, hierarchical clustering, and Principal Component Analysis (PCA) are harnessed for pivotal tasks such as customer segmentation, anomaly detection, and data preprocessing. Recent breakthroughs see a trend towards deep learning-based unsupervised approaches, like autoencoders and Generative Adversarial Networks (GANs), which excel in tasks like data denoising, dimensionality reduction, and producing synthetic data for augmentation.

Nevertheless, major issues continue, showing crucial gaps that demand addressing. The scalability of algorithms to handle enormous datasets remains a serious topic, as standard unsupervised learning approaches may struggle to process vast volumes of data efficiently. The difficulty of managing high-dimensional data creates hurdles, demanding the development of unique algorithms and strategies to extract relevant insights. Evaluating the quality of unsupervised representations remains an ongoing problem, stressing the need for rigorous evaluation metrics and approaches to measure the performance of unsupervised learning models appropriately (Li & Sheu 2021).

Reinforcement learning (RL) stands as a cutting-edge strategy in web application development, with algorithms like Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) leading the push. These RL approaches are adeptly employed in a range of tasks such as content suggestion, dynamic pricing strategies, and game playing within online applications. Recent research in RL has pushed the boundaries further, investigating fresh applications such as tailored information delivery, adaptable user interfaces, and improving website design to maximize user engagement. These improvements underline the adaptability and promise of RL in tackling difficult issues in web application design and user interaction.

Considerable gaps continue, presenting notable barriers to the mainstream adoption of RL in real-world web applications. These include challenges relating to sample efficiency, wherein RL algorithms may require significant training data to reach optimal performance. The exploration-exploitation trade-offs inherent in RL algorithms provide issues in balancing the investigation of new tactics with the exploitation of established effective behaviors (Nian *et al.*, 2020). Safety concerns arise in deploying RL agents in production contexts, stressing the necessity for strong procedures to maintain the dependability and stability of RL-driven systems.

Deep learning stands as a transformational force in web application development, with approaches such as deep neural networks, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformer models altering the landscape. These state-of-the-art deep learning algorithms permit web applications to perform different tasks including picture recognition, natural language understanding, and speech synthesis with unparalleled accuracy and efficiency (Dong *et al.*, 2021). Recent research initiatives in deep learning have concentrated on developing the subject further, with a concentration on advances in model architectures, transfer learning approaches, and self-supervised learning techniques customized for web application domains (Dong *et al.*, 2021). These approaches aim to boost model performance, stimulate generalization across varied datasets, and allow more efficient model training and deployment procedures.

However, some problems exist, throwing shadows on the mainstream use of deep learning in online applications. These include the pressing requirement for model interpretability, where complicated deep learning models frequently behave as "black boxes," hampering human ability to comprehend and trust their decision-making processes (Dong *et al.*, 2021). Additionally, issues regarding the resilience of deep learning models to adversarial assaults raise worries about their trustworthiness in real-world circumstances. Moreover, the computational expense associated with training and deploying large-scale deep learning models creates practical hurdles, especially for resource-constrained online settings.

Online learning approaches provide a dynamic approach to online application development, where algorithms like Online Gradient Descent and Online Passive-Aggressive algorithms play a crucial role in supporting real-time prediction, customization, and adaptive systems. These state-of-the-art online learning algorithms are adeptly employed in numerous tasks such as dynamic pricing, click-through rate prediction, and user behavior modeling inside online platforms. Recent research attempts in online learning have investigated fresh applications and approaches specialized for web applications (Cesa-Bianchi & Orabona 2021). Advancements in this subject include employing online learning techniques for dynamic pricing strategies, forecasting user engagement metrics like click-through rates, and modeling user behavior patterns to boost customization efforts in online platforms.

However, considerable problems exist, creating large gaps in the use of online learning approaches in web applications. These problems include properly addressing idea drift, where the underlying data distribution may shift over time, creating hurdles to model flexibility and performance. Additionally, establishing model stability and minimizing catastrophic forgetting are significant considerations in online learning settings, especially when dealing with high-velocity input streams (Cesa-Bianchi & Orabona 2021). Furthermore, scalability remains a significant challenge, as online learning algorithms must rapidly handle and react to massive amounts of data in real-time.

DEEP LEARNING IN CYBERSECURITY

In recent years, deep learning has emerged as a useful tool in several sectors, including cybersecurity. With its ability to automatically learn and extract subtle patterns and features from massive volumes of data, deep learning provides considerable potential for boosting security measures and minimizing cyber risks. Table 12 we provide an overview of deep learning, explore its applications in cybersecurity, and highlight its advantages for web security specifically.

Deep learning is a subtype of machine learning that incorporates artificial neural networks with numerous layers of interconnected nodes, known as neurons or units. These networks are inspired by the structure and function of the human brain, where each layer of neurons evaluates input data and extracts increasingly sophisticated information to create predictions or classifications.

Table 12: Components of Deep Learning.

Component	Description
Neural Networks	Fundamental building blocks comprising interconnected layers of neurons—input, hidden, and output—processing data via mathematical transformations.
Activation Functions	Functions introducing non-linearity, crucial for learning complex patterns; include sigmoid, tanh, ReLU (Rectified Linear Unit), and softmax.
Backpropagation	Optimization algorithm adjusting neural network parameters (weights, biases) to minimize the difference between predicted and actual outputs using gradient descent.
Deep Architectures	Models with multiple layers of neurons enabling hierarchical data representations; examples include CNNs for images, RNNs for sequences.

Applications of Deep Learning in Cybersecurity. Deep learning has numerous applications in cybersecurity, where it is employed to detect and mitigate various types of cyber threats.

Advantages of Deep Learning for Web Security. Deep learning offers several advantages for enhancing web security measures and protecting against cyber threats.

Table 13: Applications of Deep Learning in Cybersecurity.

Application	Description
Malware Detection	Analyzing file signatures, behavior patterns, and network traffic to classify viruses, ransomware, and trojans. Learns from vast datasets to identify new threats accurately.
Anomaly Detection	Detecting abnormal behavior in network traffic, user sessions, and system logs. Identifies potential security breaches by learning normal patterns and recognizing deviations.
Intrusion Detection & Prevention	Analyzing network traffic, packet payloads, and system logs to prevent intrusions and cyberattacks on web servers and applications. Recognizes known attack vectors and emerging threats in real-time.
Phishing Detection	Analyzing email headers, content, and user interactions to detect phishing attempts. Learns from historical data to classify emails based on linguistic cues, context, and sender reputation.

Table 14: Advantages of Deep Learning in Web Security.

Advantage	Description
Scalability	Deep learning models process vast data volumes and adapt to evolving threats autonomously. Ideal for large-scale web environments and dynamic attack landscapes. Continuously learning, they update knowledge and enhance performance.
Adaptability	Deep learning models learn and generalize from diverse datasets, detecting novel threats and zero-day vulnerabilities with minimal human oversight. Unlike rule-based methods, they identify subtle, unseen patterns indicating cyber threats effectively.
Accuracy	Achieving high accuracy and precision, deep learning models reduce false positives and negatives compared to traditional approaches. Leveraging advanced algorithms, they effectively distinguish between legitimate and malicious activities, enhancing overall security.

DEEP LEARNING FOR ENHANCED WEB SECURITY: ALGORITHMS AND APPLICATIONS

As the digital landscape evolves, the danger landscape evolves with it, needing enhanced security measures to defend online applications, systems, and users. Deep learning, a subset of machine learning, has emerged as a strong tool in addressing the complex and dynamic nature of cyber threats. Numerous deep learning algorithms specifically specialized for web security, including malware identification, anomaly detection, intrusion detection and prevention, and phishing detection.

Malware, including viruses, ransomware, and trojans, poses a substantial danger to web security, capable of compromising sensitive data, interrupting processes, and incurring financial losses. Deep learning models offer an effective technique to detect and categorize malware based on file signatures, behavior patterns, and network traffic (Vinayakumar *et al.*, 2019).

CNNs and RNNs are extensively used architectures for malware detection. CNNs are adept in extracting geographical and temporal properties from raw data, making them appropriate for analyzing file content and network packets. RNNs, on the other hand, are well-suited for processing sequential data, such as system logs and command sequences.

By training on massive datasets of known malware samples, deep learning models can learn to differentiate between benign and dangerous files, enabling proactive detection and mitigation of malware risks in web settings.

Anomaly detection is critical for recognizing variations from regular behavior patterns in web traffic, user sessions, and system logs, which may indicate potential security breaches or unauthorized access attempts. Deep learning algorithms offer robust solutions for anomaly detection by understanding the regular patterns and behaviors of online applications and users.

Autoencoders, a form of neural network architecture, are extensively employed for unsupervised anomaly detection. Autoencoders learn to rebuild input data by compressing it into a lower-dimensional representation and then reconstructing it back to its original form (Zavrak & Iskefiyeli 2020). Anomalies are recognized when the reconstruction error exceeds a predetermined threshold, suggesting deviations from typical behavior.

RNNs and Long Short-Term Memory (LSTM) networks are also excellent for time-series data analysis, making them useful for detecting anomalies in online traffic and system logs (Smagulova & James, 2019). By learning temporal dependencies and sequential patterns, RNNs and LSTMs can recognize abnormalities and suspicious activity suggestive of security concerns.

Intrusion detection and prevention systems (IDPS) are critical components of web security, tasked with monitoring and protecting against unauthorized access, exploits, and cyberattacks against web servers, applications, and databases (Ahmad *et al.*, 2020). Deep learning approaches boost the capabilities of IDPS by providing real-time detection and prevention of intrusions.

Deep Neural Networks (DNNs) and Deep Reinforcement Learning (DRL) are useful for intrusion detection, exploiting large-scale datasets of network traffic, packet payloads, and system logs to identify malicious behaviors and attack patterns (Naseer *et al.*, 2018). DNNs scan network traffic and system records to detect anomalies and suspicious behavior, while DRL models learn optimal techniques for responding to recognized threats and preventing further intrusions.

By continuously learning from fresh data and reacting to developing threats, deep learning-based intrusion detection and prevention systems can boost the resilience of online environments and decrease the risk of security breaches and data compromises.

Phishing attacks, when hostile actors imitate genuine entities to trick users into providing personal information, constitute a substantial danger to web security and user privacy (Lawson *et al*., 2020). Deep learning approaches offer powerful solutions for identifying and minimizing phishing attempts by evaluating email headers, message content, and user interactions.

NLP models, such as RNNs and Transformer-based architectures like BERT (Bidirectional Encoder Representations from Transformers), are often employed for phishing detection (Gualberto *et al*., 2020). These algorithms examine linguistic signals, semantic patterns, and contextual information to discern between valid and questionable emails.

By training on varied datasets of phishing emails and legitimate communications, deep learning models can learn to recognize common phishing strategies, including faked sender addresses, harmful links, and false content. By integrating phishing detection methods into email systems and web browsers, enterprises may protect users from falling victim to phishing assaults and safeguard their sensitive information.

CHALLENGES AND CONSIDERATIONS IN IMPLEMENTING DEEP LEARNING FOR WEB SECURITY

Deep learning has emerged as a strong technique in several sectors, including web security. However, despite its potential, the implementation of deep learning in web security brings significant obstacles and constraints.

One of the key obstacles in employing deep learning for web security is the availability and quality of data. Deep learning systems require vast volumes of data to train efficiently, particularly in web security where diverse and dynamic threats constantly evolve (Dai & Boroomand 2021). However, getting labeled data for training deep learning models in the domain of web security might be problematic due to various factors.

Obtaining labeled datasets that accurately depict real-world threats and attack scenarios is tough. Anomalies and malevolent behaviors are often unusual phenomena, making it tough to acquire adequate samples for training. Moreover, classifying data for online security duties involves specialist expertise and manual work, which can be time-consuming and costly. The quality of the data used to train deep learning models strongly effects their performance and generalization capabilities. In online security, data may be noisy, incomplete, or biased, which can lead to suboptimal model performance and higher false positive or false negative rates (Dai et al., 2017). Furthermore, the imbalance between normal and malicious samples in the dataset can impact the model's capacity to detect rare or emerging threats accurately.

Addressing data quality and quantity challenges in deep learning for web security needs new approaches such as data augmentation techniques, active learning methodologies, and the use of synthetic datasets generated through simulation or adversarial techniques.

Another difficulty linked with deep learning in online security is the lack of interpretability of sophisticated neural network models. Deep learning models, particularly deep neural networks, are generally characterized as black boxes, making it tough to grasp the rationale behind their predictions and judgments, especially in the context of security-sensitive applications (Lei *et al*., 2021).

Interpretability is critical in online security because it helps security analysts and practitioners to understand how and why a certain judgment or prediction was made by the model. Without interpretability, it becomes difficult to trust and validate the outputs of deep learning models, leading to problems surrounding accountability and transparency in security-critical applications. Several approaches have been proposed to enhance the interpretability of deep learning models in web security, including feature visualization techniques, attribution methodologies, and model-agnostic interpretability frameworks (Lei *et al.,* 2021). These techniques aim to provide insights into the internal workings of deep learning models and help detect any weaknesses or biases that may impair their performance and reliability.

Despite these attempts, attaining interpretability remains an ongoing difficulty in deep learning, particularly in complicated structures such as deep neural networks. Balancing model complexity with interpretability is a trade-off that demands careful attention in the creation and evaluation of deep learning-based security solutions.

A key shortcoming of deep learning models in online security is their sensitivity to adversarial assaults. Adversarial assaults entail generating input data with imperceptible perturbations that are explicitly designed to trick deep learning algorithms, resulting to inaccurate predictions or behavior. Web security, adversarial attacks pose major challenges to the integrity and resilience of deep learning-based security systems (Ren *et al.,* 2020).

Adversarial attacks can emerge in different forms, including evasion attacks, poisoning attacks, and model inversion attacks. Evasion attacks try to avoid detection methods by changing input data, while poisoning attacks include manipulating the training data to threaten the integrity of the model (Ren *et al*., 2020). Model inversion attacks leverage flaws in deep learning models to infer sensitive information about the training data or the model's parameters.

Mitigating the vulnerability of deep learning models to adversarial attacks involves a multi-faceted approach involving robust model design, adversarial training methodologies, and the integration of security mechanisms such as input sanitization and anomaly detection. Additionally, continuing research efforts are focused on building adversarial robustness metrics and adversarial training methods to better the resilience of deep learning models against sophisticated attacks.

RECENT ADVANCES AND TRENDS IN DEEP LEARNING FOR WEB SECURITY

Deep learning has changed the world of web security by giving strong tools and approaches to detect and mitigate numerous threats and vulnerabilities. In recent years, substantial breakthroughs have been achieved in exploiting deep learning for web security, ranging from state-of-the-art models to developing methodologies and strategies.

State-of-the-Art Deep Learning Models. State-of-the-art deep learning models have proven amazing performance in handling different web security concerns, including malware detection, intrusion detection, and vulnerability assessment. These models leverage advanced neural network designs, large-scale datasets, and complex training procedures to attain remarkable levels of accuracy and robustness (Mahdavifar & Ghorbani 2019).

One noteworthy example of a state-of-the-art deep learning model in web security is the use of CNNs for virus detection. CNNs excel in learning hierarchical features from raw input data, making them well-suited for analyzing complicated malware binaries and identifying dangerous patterns (Marin *et al.,* 2022). Researchers including Cui *et al.* (2019); He & Kim (2019); Wang *et al.* (2021) have created CNN-based malware detection systems that achieve high detection rates while limiting false positives, hence boosting the security of web apps and networks.

Another major innovation in deep learning for web security is the application of RNNs and long short-term memory (LSTM) networks for intrusion detection. RNNs and LSTMs are capable of capturing temporal dependencies in sequential data, making them effective for detecting aberrant behavior and recognizing network intrusions in real-time (Li *et al.,* 2021). By monitoring network traffic and system records, RNN-based intrusion detection systems may detect suspicious activity and potential security breaches with high accuracy and efficiency.

Deep learning models such as GANs have been deployed for creating realistic adversarial examples and evaluating the robustness of security systems against complex attacks (Ren & Xu 2019). By simulating realistic attack scenarios, GAN-based techniques enable security researchers to analyze the effectiveness of protection mechanisms and design more resilient security solutions.

Emerging Techniques and Strategies. In addition to state-of-the-art models, various developing methodologies and strategies are influencing the landscape of deep learning for web security. These solutions combine novel methodologies and interdisciplinary insights to confront new and growing challenges in cyberspace. One developing trend is the integration of deep learning with other fields such as NLP and graph analysis for complete threat detection and analysis. By merging numerous modalities and data sources, interdisciplinary approaches enable holistic security monitoring and threat information gathering, boosting the resilience of web systems against diverse attack vectors.

Another new strategy is the use of transfer learning and domain adaptation to increase the generalization capabilities of deep learning models in web security. Transfer learning allows models to leverage knowledge learnt from related tasks or domains, helping them to adapt to new contexts and datasets with less labeled data. By pre-training models on large-scale datasets or auxiliary tasks, transfer learning mitigates the requirement for significant labeled data and accelerates the implementation of deep learning solutions in real-world scenarios.

Emerging technologies such as federated learning and privacy-preserving approaches are gaining popularity in web security, particularly in cases requiring sensitive or confidential data. Federated learning provides collaborative model training across dispersed devices or servers without sharing raw data, ensuring user privacy and confidentiality while boosting model performance and resilience (Tan *et al.*, 2023). Privacy-preserving approaches such as differential privacy and homomorphic encryption ensure that sensitive information remains secure during data processing and model inference, safeguarding user privacy in web-based apps and services (Loya & Bana 2021).

COMPARATIVE ANALYSIS: TRADITIONAL VS. DEEP LEARNING APPROACHES IN WEB SECURITY

In the field of online security, both traditional methods and deep learning techniques play essential roles in safeguarding against cyber threats. A comparative review of different approaches reveals distinct strengths and weaknesses. Traditional approaches rely on rule-based systems, while deep learning techniques employ complicated neural networks to find abnormalities and patterns. Understanding the advantages and challenges of each strategy is vital for establishing effective web security measures.

Strengths and Weaknesses of Traditional Methods. Traditional methods in web security, such as signature-based detection and rule-based systems, have been the cornerstone of defense mechanisms for many years.

Table 15: Strengths and weakness of Traditional methods of web security.

Traditional Methods	Strengths	Weaknesses
Signature-based Detection	- Effectively identifies known threats	- Limited to known signatures
	- Low false positive rates	- Ineffective against zero-day attacks
	- Fast and efficient deployment	- Requires frequent updates and maintenance
Rule-based Systems	- Customizable rules for specific environments	- Limited scalability to handle complex threats
	- Transparency in decision-making	- Prone to false positives and false negatives
	- Well-understood and widely adopted	- Inflexible to adapt to evolving threats

Traditional techniques excel at identifying known dangers and enforcing established rules but suffer with adaptation and resistance against developing threats and sophisticated attack routes. They provide a baseline degree of protection but may fall short in tackling the dynamic nature of cyber threats in today's linked world.

Advantages and Challenges of Deep Learning Techniques. Deep learning techniques, powered by artificial neural networks, offer a paradigm shift in web security by enabling automated learning and detection of complex patterns.

Table 16: Deep learning techniques: Advantages and Challenges.

Deep Learning Techniques	Advantages	Challenges
Neural Network Architectures	- Capable of learning intricate patterns and features	- Requires large amounts of labeled training data
	- Adaptable to diverse and evolving threat landscapes	- Computationally intensive and resource-demanding
	- Potential for continuous improvement through	- Lack of interpretability and explainability
	feedback loops and model refinement	- Vulnerability to adversarial attacks
	- Generalizes well to unseen data	- Ethical and privacy concerns

Deep learning approaches use the potential of neural networks to autonomously learn from data and adapt to changing contexts. They offer the possibility for greater accuracy and scalability in web security solutions. However, difficulties such as data requirements, processing resources, interpretability, and adversarial weaknesses offer substantial impediments to their widespread adoption and application.

The comparative research illustrates the various characteristics and trade-offs between standard methods and deep learning techniques in web security. Traditional solutions provide reliability and transparency but may struggle to keep pace with the dynamic nature of cyber threats. Deep learning techniques provide the promise of automation and adaptability but come with inherent limitations related to data, computation, and interpretability.

The future of web security lies in utilizing the complimentary qualities of both classical methods and deep learning techniques. Integrating rule-based systems with deep learning models can boost detection capabilities while maintaining transparency and interpretability. Moreover, continual research and innovation are needed to solve the obstacles associated with deep learning, including data scarcity, computational efficiency, and adversarial robustness.

By using the strengths of traditional methods and the capabilities of deep learning techniques, companies may design comprehensive web security plans that effectively reduce risks and safeguard against developing threats in an increasingly interconnected digital ecosystem.

STRATEGIES FOR DEEP LEARNING INTEGRATION IN WEB VULNERABILITY IDENTIFICATION

Integrating deep learning models into web vulnerability identification comprises many important tactics targeted at exploiting the capabilities of neural networks to boost detection accuracy and efficiency. Here are some techniques for efficiently integrating deep learning into the vulnerability detection process:

a. Feature Extraction and Representation: Deep learning models excel at learning hierarchical representations of data, making them well-suited for extracting features from raw web traffic logs, server logs, and application data. By preprocessing and translating raw data into high-dimensional feature vectors, deep learning models can capture subtle patterns and anomalies suggestive of possible vulnerabilities.

b. Anomaly identification and Pattern Recognition: Deep learning techniques, such as autoencoders and RNNs, can be applied for anomaly identification and pattern recognition in web traffic and system logs. These models understand the regular behavior of online apps and infrastructure and highlight deviations or unexpected patterns that may signal security vulnerabilities or malicious activity.

c. Ensemble Learning and Model Fusion: Ensemble learning techniques, such as mixing multiple deep learning models or integrating deep learning with classical machine learning algorithms, can boost the robustness and generalization capabilities of vulnerability identification systems. By employing multiple models and pooling their predictions, ensemble techniques decrease the chance of false positives and increase overall detection performance.

d. Transfer Learning and Domain Adaptation: Transfer learning approaches enable the transfer of knowledge from pre-trained deep learning models to new vulnerability identification tasks or domains with insufficient labeled data. By fine-tuning pre-trained models on domain-specific datasets or adapting them to target contexts, transfer learning speeds the deployment of deep learning solutions in real-world scenarios and boosts their effectiveness in detecting web vulnerabilities.

CONCLUSIONS

Web vulnerabilities represent critical weaknesses within digital systems that adversaries can exploit to compromise security and integrity. These vulnerabilities, occurring at various levels of web architecture, demand a comprehensive understanding of effective cybersecurity measures. Common examples include injection attacks, authentication flaws, insecure direct object references, security misconfigurations, and cross-site request forgery. These vulnerabilities can lead to breaches, malware infections, service disruptions, and reputational damage, highlighting the urgent need for robust security strategies.

Deep learning has emerged as a promising approach in cybersecurity, offering significant potential to enhance detection accuracy and efficiency. By automatically learning from vast datasets, deep learning models can effectively identify malware, detect anomalies, prevent intrusions, and detect phishing attempts. Moreover, deep learning exhibits scalability, adaptability, and accuracy, making it suitable for large-scale web environments and dynamic threat landscapes. However, integrating deep learning into web vulnerability identification poses challenges, including data availability and quality, interpretability, and susceptibility to adversarial attacks. Addressing these challenges requires innovative approaches such as data augmentation, interpretability techniques, and adversarial robustness measures.

As the digital landscape evolves, so do the associated dangers, necessitating enhanced security measures to safeguard online applications, systems, and users. Deep learning, a subset of machine learning, has emerged as a potent tool for addressing the intricate and evolving nature of cyber threats, particularly in web security. This review has explored numerous deep learning algorithms specialized for various aspects of web security, including malware identification, anomaly detection, intrusion detection and prevention, and phishing detection. Malware, a significant threat to web security, can lead to severe consequences such as data compromise and financial losses. Deep learning models, particularly CNNs and RNNs, have proven effective in detecting and categorizing malware based on diverse data sources. Moreover, deep learning excels at anomaly detection, identifying deviations from normal patterns in web traffic and system logs indicative of potential security breaches. Despite its potential, implementing deep learning in web security presents challenges, including data availability and quality, interpretability, and vulnerability to adversarial attacks. Addressing these challenges requires innovative approaches such as data augmentation, interpretability techniques, and adversarial robustness measures.

Recent advances in deep learning have revolutionized web security by providing powerful tools and methodologies to detect and mitigate various threats and vulnerabilities. State-of-the-art deep learning models, such as CNNs for malware detection and RNNs for intrusion detection, have demonstrated remarkable performance in addressing critical security concerns. Additionally, emerging techniques like GANs and interdisciplinary approaches incorporating NLP and graph analysis are shaping the future of web security by enabling comprehensive threat detection and analysis. While deep learning offers significant advantages in terms of automation, adaptability, and scalability, it also presents challenges such as data requirements, computational resources, interpretability, and vulnerability to adversarial attacks. Addressing these challenges requires innovative strategies like transfer learning, federated learning, and privacy-preserving approaches to enhance the generalization capabilities and resilience of deep learning models in web security applications.

Ultimately, the integration of deep learning techniques with traditional methods holds promise for designing comprehensive web security solutions that effectively reduce risks and safeguard against evolving threats in today's interconnected digital landscape. Continued research and innovation in deep learning for web security are essential to overcome existing challenges and ensure the development of robust and reliable security measures for the future.

Acknowledgement. The authors would like to express their sincere gratitude to the anonymous reviewers and the editor for their valuable comments, insightful suggestions, and constructive feedback, which have significantly improved the quality of this manuscript.

Conflict of Interest. None.

REFERENCES

Abdullayev, V. & Chauhan, D. A. S. (2023). SQL Injection Attack: Quick View. *Mesopotamian Journal of Cyber Security*, 30–34.

Aborujilah, A., Adamu, J., Shariff, S. M. & Awang Long, Z. (2022). Descriptive Analysis of Built-in Security Features in Web Development Frameworks. *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*.

Abu-Faraj, M., Al-Hyari, A. & Alqadi, Z. (2022). A Complex Matrix Private Key to Enhance the Security Level of Image Cryptography. *Symmetry*, *14*(4), 664.

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J. & Ahmad, F. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32*(1).

Alazmi, S. & De Leon, D. C. (2022). A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners. *IEEE Access*, *10*, 33200–33219.

Ali, S., Raslan, A. & Fattouh, L. (2024). Model for the prediction of default risk of funding requests using data mining. *International Journal on Emerging Technologies*, *15*(2), 05–12.

Al-Kahla, W., Shatnawi, A. S. & Taqieddin, E. (2021). A Taxonomy of Web Security Vulnerabilities. *12th International Conference on Information and Communication Systems (ICICS)*, 424 – 429.

ALSaleem, B. O. & Alshoshan, A. I. (2021). Multi-Factor Authentication to Systems Login. *2021 National Computing Colleges Conference (NCCC)*.

Althamir, M., Alabdulhay, A. & Yasin, M. M. (2023). A Systematic Literature Review on Symmetric and Asymmetric Encryption Comparison Key Size. *2023 3rd International Conference on Smart Data Intelligence (ICSMDI)*.

Amankwah, R., Chen, J., Kudjo, P. K. & Towey, D. (2020). An empirical comparison of commercial and open-source web vulnerability scanners. *Software: Practice and Experience*, *50*(9), 1842–1857.

Anis, A., Zulkernine, M., Iqbal, S., Liem, C. & Chambers, C. (2018). Securing Web Applications with Secure Coding Practices and Integrity Verification. *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*.

Anwar, A., Khormali, A., Nyang, D. & Mohaisen, A. (2018). Understanding the hidden cost of software vulnerabilities: measurements and predictions. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 377–395.

Asaduzzaman, M., Rawshan, P. P., Liya, N. N., Islam, M. N. & Dutta, N. K. (2020). A Vulnerability Detection Framework for CMS Using Port Scanning Technique. *Cyber Security and Computer Science*, 128–139.

Asati, S., Gangele, R. K. & Kumar, B. (2024). The Black-Scholes Model: A comprehensive analysis. *International Journal on Emerging Technologies, 15*(2), 13–18.

Bamane, K. D., Patankar, A. J., Gupta, P., Jambi, R. K. & Rajgure, N. (2023). Enhanced Study of Deep Learning Algorithms for Web Vulnerability Scanner. *International Journal on Recent and Innovation*

Trends in Computing and Communication, *11*(7s), 610–616.

Bansal, A. (2023). CRABAC: Combined Role-Attribute Based Access Control Model. *Interantional Journal of Scientific Research in Engineering And Management*, *07*(09).

Bararia, A. & Choudhary, M. V. (2023). Systematic review of common web-application vulnerabilities. *Interantional Journal of Scientific Research in Engineering and Management*, *07*(01).

Cesa-Bianchi, N., and Orabona, F. (2021). Online Learning Algorithms. *Annual Review of Statistics and Its Application*, *8*(1), 165–190.

Chen, Y., Santosa, A. E., Sharma, A. & Lo, D. (2020). Automated identification of libraries from vulnerability data. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*.

Chhetri, T. R., Kurteva, A., DeLong, R. J., Hilscher, R., Korte, K. & Fensel, A. (2022). Data protection by design tool for automated gdpr compliance verification based on semantically modeled informed consent. *Sensors*, 22(7): 2763

Cui, Z., Du, L., Wang, P., Cai, X. & Zhang, W. (2019). Malicious code detection based on CNNs and multi-objective algorithm. *Journal of Parallel and Distributed Computing*, *129*, 50–58.

Dai, D. & Boroomand, S. (2021). A Review of Artificial Intelligence to Enhance the Security of Big Data Systems: State-of-Art, Methodologies, Applications, and Challenges. *Archives of Computational Methods in Engineering*, *29*(2), 1291–1309.

Dai, W., Yoshigoe, K. & Parsley, W. (2017). Improving Data Quality Through Deep Learning and Statistical Models. *Advances in Intelligent Systems and Computing*, 515–522.

Dong, S., Wang, P. & Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, *40*, 100379.

Dwivedi, A. (2021). Vulnerability Scanning Technology on Web Applications. *International Journal for Research in Applied Science and Engineering Technology*, *9*(6), 991–995.

Fruhlinger, J. (2020). *Marriott data breach FAQ: How did it happen and what was the impact?* CSO Online. https://www.csoonline.com/article/567795/marriott-data-breach-faq-how-did-it-happen-and-what-was-the-impact.html

Fruhlinger, J. (2023). *Equifax data breach FAQ: What happened, who was affected, what was the impact?* CSO Online. https://www.csoonline.com/article/567833/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html

Gu, H., Zhang, J., Liu, T., Hu, M., Zhou, J., Wei, T. & Chen, M. (2020). DIAVA: A traffic-based framework for detection of sql injection attacks and vulnerability analysis of leaked data. *IEEE Transactions on Reliability*, *69*(1), 188–202

Gualberto, E. S., De Sousa, R. T., De Brito Vieira, T. P., Da Costa, J. P. C. L. & Duque, C. G. (2020). The Answer is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering. *IEEE Access*, *8*, 223529–223547.

Gupta, S., and Gupta, B. B. (2015). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, *8*(S1), 512–530.

Gupta, V., Mishra, V. K., Singhal, P. & Kumar, A. (2022). An Overview of Supervised Machine Learning Algorithm. *2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)*.

Hassan, M. M., Nipa, S. S., Akter, M., Haque, R., Deepa, F. N., Rahman, M. M., Siddiqui, M. & Sharif, M. H. (2018). broken authentication and session management vulnerability: a case study of web application. *International Journal of Simulation: Systems, Science and Technology*, *19*(01).

He, K. & Kim, D. S. (2019). Malware Detection with Malware Images using Deep Learning Techniques. *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*.

Hristov, M., Nenova, M., Iliev, G. & Avresky, D. (2021). Integration of Splunk Enterprise SIEM for DDoS Attack Detection in IoT. *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)*.

Jones, C. (2022). *Warnings (&#038; Lessons) of the 2013 Target Data Breach*. Red River | Technology Decisions Aren't Black and White. Think Red. https://redriver.com/security/target-data-breach

Kour, P. (2020). A Study on Cross-Site Request Forgery Attack and Its Prevention Measures. *International Journal of Advanced Networking and Applications*, *12*(02), 4561–4566.

Lawson, P., Pearson, C. J., Crowson, A. & Mayhorn, C. B. (2020). Email phishing and signal detection: How persuasion principles and personality influence response patterns and accuracy. *Applied Ergonomics*, *86*, 103084.

Lei, X., Fan, Y., Li, K. C., Castiglione, A. & Hu, Q. (2021). High-precision linearized interpretation for fully connected neural network. *Applied Soft Computing*, *109*, 107572.

Li, C., Wang, Y., Miao, C. & Huang, C. (2020). Cross-Site Scripting Guardian: A Static XSS Detector Based on Data Stream Input-Output Association Mining. *Applied Sciences*, *10*(14), 4740.

Li, H., and Sheu, P. C. Y. (2021). A scalable association rule learning heuristic for large datasets. *Journal of Big Data*, *8*(1).

Liu, M., Zhang, B., Chen, W. & Zhang, X. (2019). A Survey of Exploitation and Detection Methods of XSS Vulnerabilities. *IEEE Access*, *7*, 182004–182016.

Loya, J. & Bana, T. (2021). Privacy-Preserving Keystroke Analysis using Fully Homomorphic Encryption & Differential Privacy. *2021 International Conference on Cyberworlds (CW)*.

Mahdavifar, S. & Ghorbani, A. A. (2019). Application of deep learning to cybersecurity: A survey. *Neurocomputing*, *347*, 149–176.

Marashdih, A. W., Zaaba, Z. F. & Suwais, K. (2018). Cross Site Scripting: Investigations in PHP Web

Application. *2018 International Conference on Promising Electronic Technologies (ICPET)*.

Marin, D., Orozco-Rosas, U. & Picos, K. (2022). Malware classification through image processing with a convolutional neural network. *Optics and Photonics for Information Processing XVI*.

Mishra, S., Alowaidi, M. A. & Sharma, S. K. (2021). Impact of security standards and policies on the credibility of e-government. *Journal of Ambient Intelligence and Humanized Computing*, 1-12.

Muzaki, R. A., Briliyant, O. C., Hasditama, M. A. & Ritchi, H. (2020). Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall. *2020 International Workshop on Big Data and Information Security (IWBIS)*.

Naseer, S., Saleem, Y., Khalid, S., Bashir, M. K., Han, J., Iqbal, M. M. & Han, K. (2018). Enhanced Network Anomaly Detection Based on Deep Neural Networks. *IEEE Access*, *6*, 48231–48246.

Naveen, M. & Mishra, D. P. (2020). Addressing web application security issues and vulnerabilities assessment pen testing. *International Journal of Recent Technology and Engineering (IJRTE)*, 8(6): 2314–2321

Nembhard, F. D., Carvalho, M. M. & Eskridge, T. C. (2019). Towards the application of recommender systems to secure coding. *EURASIP Journal on Information Security*, *2019*(1).

Nian, R., Liu, J., and Huang, B. (2020). A review On reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, *139*, 106886.

Nirmal, K., Janet, B. & Kumar, R. (2018). Web Application Vulnerabilities - The Hacker's Treasure. *International Conference on Inventive Research in Computing Applications (ICIRCA)*, 58-62.

Patil, S. S. & Thokal, G. N. (2025). Analysis of predictive mechanical maintenance using artificial intelligence, machine learning, and data science. *International Journal on Emerging Technologies*, *16*(1), 45–53.

Petcu, A., Pahontu, B., Frunzete, M., and Stoichescu, D. A. (2023). A Secure and Decentralized Authentication Mechanism Based on Web 3.0 and Ethereum Blockchain Technology. *Applied Sciences*, *13*(4), 2231.

Pratama, I. P. A. E., and Rhusuli, A. M. (2022). Penetration Testing on Web Application Using Insecure Direct Object References (IDOR) Method. *2022 International Conference on ICT for Smart Society (ICISS)*.

Priyanka, A. K. & Sai Smruthi, S. (2020). Web Application Vulnerabilities: Exploitation and Prevention. *International Conference on Electrotechnical Complexes and Systems (ICOECS)*. 1-5.

Purba, M. D., Ghosh, A., Radford, B. J. & Chu, B. (2023). Software Vulnerability Detection using Large Language Models. *2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW)*.

Rana, S. & Mishra, D. (2021). An authenticated access control framework for digital right management system. *Multimedia Tools and Applications*.

Ren, C. and Xu, Y. (2019). A Fully Data-Driven Method Based on Generative Adversarial Networks for Power System Dynamic Security Assessment with Missing Data. *IEEE Transactions on Power Systems*, *34*(6), 5044–5052.

Ren, K., Zheng, T., Qin, Z. & Liu, X. (2020). Adversarial Attacks and Defenses in Deep Learning. *Engineering*, *6*(3), 346–360.

Shakirov, M. B. & Karmanov, I. N. (2020). Information Security Audit of an Optoelectronic Device Engineering Enterprise. *Interexpo GEO-Siberia*, *6*(2), 146–151.

Smagulova, K. & James, A. P. (2019). A survey on LSTM memristive neural network architectures and applications. *The European Physical Journal Special Topics*, *228*(10), 2313–2324.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. (2023). Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(12), 9587–9603.

Tang, M., Alazab, M., and Luo, Y. (2019). Big Data for Cybersecurity: Vulnerability Disclosure Trends and Dependencies. *IEEE Transactions on Big Data*, *5*(3), 317–329.

Thangavel, M. & Varalakshmi, P. (2020). Enabling Ternary Hash Tree Based Integrity Verification for Secure Cloud Data Storage. *IEEE Transactions on Knowledge and Data Engineering*, *32*(12), 2351–2362.

Tian, Z., Luo, C., Qiu, J., Du, X. & Guizani, M. (2020). A Distributed Deep Learning System for Web Attack Detection on Edge Devices. *IEEE Transactions on Industrial Informatics*, *16*(3).

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A. & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, *7*, 41525–41550.

Wang, S., Wang, J., Song, Y. & Li, S. (2021). Malicious Code Variant Identification Based on Multiscale Feature Fusion CNNs. *Computational Intelligence and Neuroscience*, *2021*, 1–10.

Young, K. (2021). *Cyber Case Study: Sony Pictures Entertainment Hack*. CoverLink Insurance - Ohio Insurance Agency.

Zavrak, S. & Iskefiyeli, M. (2020). Anomaly-Based Intrusion Detection from Network Flow Features Using Variational Autoencoder. *IEEE Access*, *8*, 108346–108358.

Zhao, L., Chen, S., Xu, Z., Liu, C., Zhang, L., Wu, J., Sun, J. & Liu, Y. (2023). Software Composition Analysis for Vulnerability Detection: An Empirical Study on Java Projects. *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering.*

How to cite this article: Oduleye B.E., Asuquo P. and Bliss U.S. (2025). Web Vulnerabilities using Machine Learning for Prevention and Detection: A Critical Review. *International Journal on Emerging Technologies, 16*(2): 120–139.