Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review

Article *in* Artificial Intelligence Review · March 2023

DOI: 10.1007/s10462-023-10433-3

CITATIONS
10

READS
1,394

3 authors:

Jasleen Kaur
Chandigarh University
6 PUBLICATIONS **50** CITATIONS
SEE PROFILE

Urvashi Garg
Chandigarh University
29 PUBLICATIONS **196** CITATIONS
SEE PROFILE

Gourav Bathla
University of Petroleum & Energy Studies
35 PUBLICATIONS **351** CITATIONS
SEE PROFILE

Artificial Intelligence Review
https://doi.org/10.1007/s10462-023-10433-3

Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review

Jasleen Kaur[1] · **Urvashi Garg**[1] · **Gourav Bathla**[2]

Abstract
With the rising demand for E-commerce, Social Networking websites, it has become essential to develop security protocols over the World Wide Web that can provide security and privacy to Internet users all over the globe. Several traditional encryption techniques and attack detection protocols can secure the data transmitted over public networks. However, hackers can effortlessly exploit them to acquire access to the users' sensitive information such as user ID, session ID, cookies, passwords, bank account details, contact numbers, private PINs, database information, etc. Researchers have continuously innovated new techniques to build a secure and robust system that cannot be easily hacked and manipulated. Still, there is much scope for novelty to provide security against contemporary techniques used by intruders. The motivation of this survey is to observe the recent developments in Cross-Site Scripting attacks and techniques used by researchers to secure confidential information. Cross-Site Scripting (XSS) has been recognized as one of the top 10 online application security risks by the Open Web Application Security Project (OWASP) for decades. Therefore, dealing with this security flaw in web applications has become essential to avoid further personal and financial damage to Internet users and business organizations. There is a need for an extensive survey of recent XSS attack detection techniques that can provide the right direction to researchers and security professionals. We present a complete overview of recent machine learning and neural network-based XSS attack detection techniques in this paper, covering deep neural networks, decision trees, web-log-based detection models, and many more. This paper also highlights the research gaps that must be addressed while designing attack detection models. Further, challenges researchers face during the development of recent techniques are also discussed. Finally, future directions are provided to reflect on new concepts that can be used in forthcoming research works to improve XSS attack detection techniques.

Keywords Web vulnerabilities · Cyber-attacks · Web-security · Machine learning · XSS attack · Deep learning · Neural networks

✉ Gourav Bathla
gouravbathla@gmail.com

Extended author information available on the last page of the article

Published online: 23 March 2023

Abbreviations

APT	Advanced persistent threats
CBoW	Continuous bag of words
CMDi	Operating system command injection
DBN	Deep belief network
DDoS	Distributed denial of service
DDQN	Dueling deep q networks
DPI	Deep packet inspection
ECDSA	Elliptic curve digital signature algorithm
FSM	Finite state machine
HTTPS	Hyper text transfer protocol secure
IDS	Intrusion detection system
IoT	Internet of things
kNN	K- Nearest neighbor
LSTM-RNN	Long short-term memory recurrent neural network
MLP	Multilayer perceptron
MNN	Modular neural network
OSI	Open system interconnection
OSN	Online social networks
OWASP	Open web application security project
PCA	Principle component analysis
pdAPSO	Primal dual particle swarm optimization
PNN	Probabilistic neural network
PSO	Particle swarm optimization
QSSVM	Quarter sphere support vector machines
RNN	Recurrent neural network
RSMT	Robust software modelling tool
SARSA	State action reward state action
SCADA	Supervisory control and data acquisition
SDN	Software-defined networks
SQLi	SQL injection
SVM	Support vector machine
TF-IDF	Term frequency-inverse document frequency
URI	Uniform resource identifier
URL	Uniform resource locator
WAF	Web application firewalls
XSS	Cross-site scripting

1 Introduction

The proliferation of online services such as medical care, railway services, airline booking, online shopping, electronic banking, online payments, social networking sites, and many more have paved the way for cybercriminals to hack online systems and steal users' sensitive information (Snehi & Bhandari 2021). The users have become highly dependent upon the internet and trust most web applications without analyzing their credibility. The common public is entirely unaware of the black ethics used by hackers to steal their private data. Since it is impractical to spread cyber security awareness among all individuals, implementing

robust attack detection techniques in web systems is the only possible method to protect user data and privacy. The attack detection and mitigation tools and techniques aim to find flaws in the website. These flaws in a website's design, development, or configuration are commonly referred to as web vulnerabilities (Mohammad & Reza 2017). User ignorance and a lack of formal staff training are frequent factors in the success of web attacks (Gkioulos & Chowdhury 2021). Most web vulnerabilities are located at the application layer of the OSI network model (Shabut et al. 2016) (Khan et al. 2017), while human error in website source code accounts for 93% of data breaches. All public and private organizations refer to security reports offered by OWASP for security analysis. An OWASP organization aims to improve the security standards of software(s) by working on open-source projects and hosting international conferences so that proper action can be taken to improve the web applications' security (owasp 2017). Table 1 illustrates the top 10 web-application-related critical security risks according to OWASP. It is evident from Table 1 that the most prominent internet threat among web applications is cross-site scripting (XSS) attack and thus needs the utmost attention of researchers. Also, the universal cost of threat activities executed by hackers on online platforms crossed $6 trillion at the end of 2021 (Nick 2021).

Several surveys have been conducted in the past to explore XSS attack detection techniques (Sarmah et al. 2018) (Gupta & Gupta 2015) (Jagajeevan Rao et al. 2021) (Kumar & Goyal 2019; Li, 2018; Pitropakis et al. 2019; Rodríguez et al. 2020; Shabut et al. 2016). Nevertheless, these surveys have not covered the latest developments and research challenges in detail. The primary objective of this work is to investigate the XSS attack detection mechanisms, focusing on recent improvements in this field. Further, XSS attack and detection are elaborated, which is the most dangerous attack executed by the hacker by bypassing the Same Origin Policy. This is done to masquerade as a victim user and perform all the actions on a website that a legitimate user can do. Any website is vulnerable to web attack if the input validation and encoding are not performed accurately. It is estimated that almost two-thirds of web applications (owasp 2017) across the globe are at risk of XSS attacks. A comprehensive study investigated by the EDGESCAN organization in 2019 and 2021 claimed that (EOIN KEARY, n.d.) XSS, Brute force and authentication, file path traversal, administrative interface, and information disclosure vulnerabilities still come under the category of high-risk factors. Even though substantial research is being carried out to develop new technologies and methods for detecting, preventing, and mitigating cyberattacks, many websites

Table 1 OWASP Top 10 Vulnerabilities 2020–2021 (owasp 2017)

CVE no	Vulnerability name (2020)	Vulnerability name (2021)
CVE1	Injection	Injection
CVE2	Broken authentication	Broken authentication and session management
CVE3	Sensitive data exposure	Cross-site scripting
CVE4	XML external entities	Insecure direct object references
CVE5	Broken access control	Security misconfigurations
CVE6	Security misconfigurations	Sensitive data exposure
CVE7	Cross-site scripting	Missing function-level access control
CVE8	Insecure deserialization	Cross-site request forgery
CVE9	Using components with known vulnerabilities	Using components with known vulnerabilities
CVE10	Insufficient logging and monitoring	Invalidated redirects and forwards

continue to be targeted by hackers. (Tran et al. 2017). This explains why code-related vulnerabilities must be recognized to prevent cyber-criminal groups from exploiting websites. Moreover, according to EdgeScan report 2021, XSS (Reflected/ DOM) is still at the top, where its percentage of occurrence is 37.20% at the application layer (Keary Eoin, n.d.), as depicted in Fig. 1.

When an XSS flaw exists in a website, it can accept any untrusted malicious string as input. The untrusted input is processed either on the client side or the server side, depending on the vulnerability. In this way, an innocent web user becomes a victim of a web attack by visiting a vulnerable website, downloading malicious files from an untrusted link, or clicking on content that looks mischievous. Moreover, there is a significant possibility that if a website is prone to XSS attack, it is also prone to SQLI, broken authentication, session management attack, and DDoS attacks. The XSS attack targets HTML webpages by inserting malicious code or a malicious URL into vulnerable content, like how the SQLI attack targets SQL-based programs by bypassing SQL queries. Cybercriminals are frequently exploiting XSS vulnerability to steal user sessions to take over their accounts and perform illegal operations by masquerading as authenticated users. Also, it has become the most prominent way to get into the company's social network and hack private information about the company (Rodríguez et al. 2020). The most notable web applications such as Facebook, Twitter, YouTube, and TikTok have also suffered from this attack in the past (Fang et al. 2018).

The attacker performs the following steps for the successful execution of the XSS attack:

- Firstly, the attacker leverages online vulnerability scanning tools to find out websites with XSS flaws to execute the attack vector.
- Secondly, once a vulnerable website search is successful, the attacker passes malicious scripts into the input field such as the comment section, search bar, and URL, and analyzes the processed results produced by the vulnerable webpage.
- The script injected by the attacker can be a JavaScript code, HTML, PHP, or VB Script. The script is either loaded into the web system database or executed by the web browser.
- Finally, by overcoming the SOP, the attacker obtains access to the victim's confidential credentials through a susceptible website. Figure 2 depicts a sample scenario of any web-based attack on the web user.

This work aims to review the causes of web attacks and find out research gaps that can be fulfilled to design an optimized solution for XSS attack detection. The articles selected

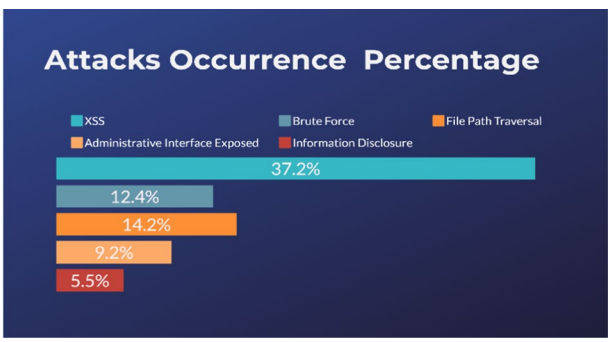

Fig. 1 Percentage of occurrence of attacks

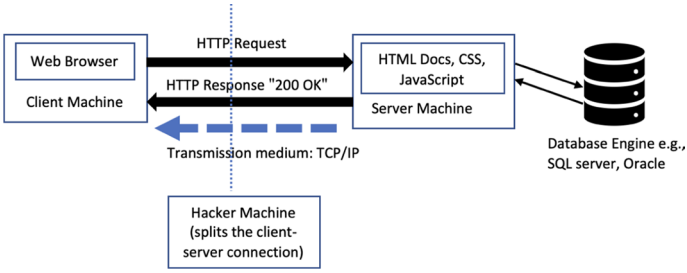

Fig. 2 The general scenario of the XSS attack illustration

for this survey were crawled from Google search, Scholar, Science Direct, Elsevier, Scopus, IEEE explore, Springer, etc. We passed five different types of keywords to search the paper: (i) XSS vulnerability, (ii) site-injection attacks, (iii) machine learning and cyber security, (iv) detection of XSS attacks and machine learning, and (v) cyber-security challenges. The following are the primary research questions of our study:

RQ1: What new advances have been made in the realm of XSS attack detection?
RQ2: Why are traditional techniques not effective at combating XSS attacks?
RQ3: How can XSS attacks be detected and prevented using Machine Learning techniques?
RQ4: What are the most popular and effective cyber-attack mitigation techniques used by active researchers?
RQ5: What are the challenges and opportunities for young researchers in the field of cyber security?

1.1 Motivation and contribution

The key motivation of this extensive survey is to identify the root cause of XSS attacks in today's contemporary era where web applications are already built on secure platforms. During the analysis of existing research works, aforementioned five research questions were found and our survey paper focuses on finding the answers to these research questions. The questions have been answered at the end of the manuscript in Sect. 7. Further, in the past we have observed that researchers have not mentioned recent XSS attack detection techniques. In this survey, we have explored recent techniques and developments. The key motivation is to analyze why accuracy is not adequate and what can be the scope of improvement Fig. 3.

The remaining sections are organized as follows: Sect. 2 gives an outline of the SOP policy, Sect. 3 describes cybersecurity threats and attacks, Sect. 4 discusses in detail the significance of machine learning in combating cyber threats, and Sect. 5 elaborates on XSS attacks, and their types. Section 6 uncovers recent developments in XSS attack detection using machine learning and Deep Learning. Section 7 discusses the various challenges and opportunities in this research field, and finally, Sect. 8 wraps up our study by outlining future directions. Figure 3 depicts the organization of our survey. In the Introduction section, the background of cyber-attacks, including a report on OWASP Top-10 vulnerabilities and percentage occurrence of attacks, are listed. Same Origin Policy, the background of

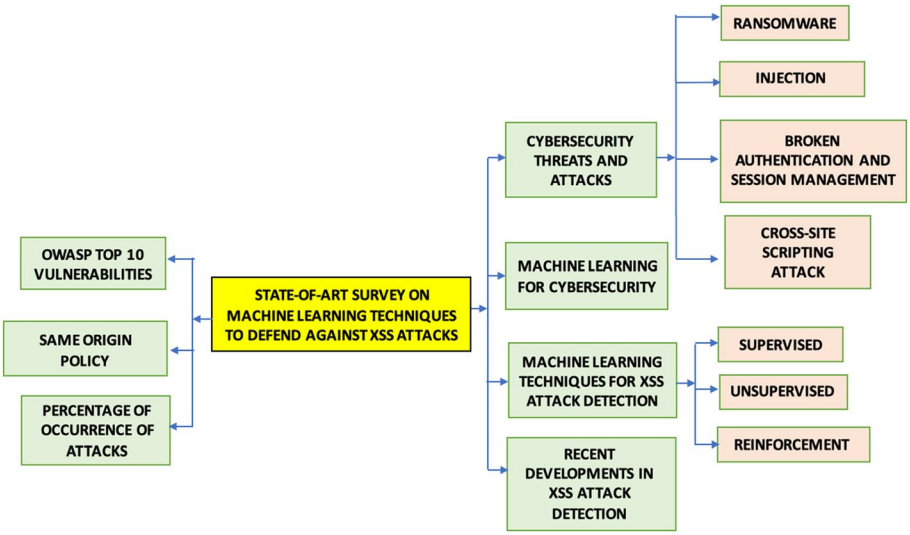

Fig. 3 Flowchart of the survey

Cybersecurity, machine learning techniques applied for XSS attack detection, and recent developments are explained in different sections. Finally, the challenges and concepts covered in the paper are concluded.

2 Same origin policy

Same Origin Policy (SOP) is a vital concept in web applications that comes under the web-security model enforced by web browsers. It defines a set of rules for the interaction between two different web pages within a web application. Bypassing the SOP, the attacker gains access to the victim's confidential credentials via a susceptible website. Web browsers implement SOP by restricting document A of website A from accessing another document B of website B that does not belong to the exact origin. Origin, in simple terms, refers to the three components: URI scheme, host, and port. URI schemes- such as HTTP, HTTPS, FTP, etc.- specify the protocol to access the resources. For example, HTTPS specifies the use of TLS protocol for communication over HTTP. Host- host refers to a particular computer accessing the world wide web. For instance, www.wiki.com is a hostname separated by dots. Every host will have its network address and unique host address. Port- port is a programmable docking port through which the web information flows. It is the communication point where the communication begins and ends.

The main idea behind the implementation of SOP is to prevent data access of website A by malicious website B if both the websites have different origins, i.e., different schemes, hosts, and ports. Therefore, SOP is critical in ensuring the confidentiality and integrity of online applications. (Matt 2021). However, the case study by Schwenk (Guo et al., 2017) stated that since SOP is not defined consistently in academic and non-academic areas, web vulnerabilities allow attackers to avoid SOP checks. Table 2 illustrates

a sample example of how a web page document is granted access by comparing the three components of SOP.

3 Cyber security threats and attacks

Cyber security is a set of security rules, protections, tools, policies, risk-management strategies, and best practices that protect cyberspace, organizations, and users' assets (von Solms & van Niekerk 2013). Cybersecurity aims to preserve the integrity, confidentiality, and availability of user information. Web applications are often versatile in nature because of the different third-party components used, such as AJAX and HTML5, and third-party sign-in options via Google, Facebook, etc. All these web applications are developed using PHP, Ruby, ASP.Net, or Java, where PHP is used by 44% of business organizations for their website development. With the growing demand for PHP web development, there has been a significant decline in vulnerabilities since 2016 (Gupta B.B. & Chaudhary Pooja, 2020). All thanks to the implementation of security patches in PHP scripts by web designers and security professionals that work together to deal with web attacks. However, designing a flawless web application is still a dream that needs to be accomplished (Kumar & Goyal 2019). The most popular attack types leveraged by the attackers for privacy violations are discussed briefly in the following subsections.

3.1 Ransomware attack

Ransomware has grown in popularity as one of the most dangerous cyber threats to businesses. (Brewer 2016). The pandemic year 2020 has seen a vast increase in ransomware attacks on corporate networks (CISCO 2021). When a ransomware infection compromises the victim's system, the victim is unable to access vital data stored in his system and is demanded a ransom amount of money to regain data access or get the decryption key. The victim is given some time to pay in bitcoin. If the victim cannot make the payment, the ransom increases after the deadline. Data access is denied either by encrypting the data or by adopting another criminal tactic. In 2013, CryptoLocker ransomware targeted the Police department and affected almost 250 systems (Kharraz et al. 2015), while in 2016, the healthcare industry was targeted by SamSam ransomware. In addition, WannaCry Ransomware appeared in 2017 and affected more than two lakhs computer systems across 150 countries worldwide (Furnell & Emm 2017). Ransomware is usually spread over the network using phishing emails or exploit kits, allowing infected

Table 2 Verification of web access by SOP

URL	Decision	Reason
https://www.demosite1.com	Accessible	Protocol, host, port match
http://www.demosite2.com	Not accessible	Protocol mismatch
https://www.demosite2x.com	Not accessible	Host mismatch
https://www.demosite2.com:8081	Not accessible	Port mismatch

files to be run on the device. Most crypto-malware add persistence mechanism so that if the system is rebooted during the malware encryption process, it will restart from the previous point until it completes the process. The best practice to defend against a ransomware attack is to save the backup of all critical files and documents.

3.2 Injection attack

Injection attacks are the oldest and are still executed by cybercriminals to obfuscate users into revealing their sensitive information. Since this type of attack can allow attackers to execute remote commands or even install malware onto the system, they pose a severe threat to web security and give an open challenge to security researchers. OS command injection, SQL injection, XSS injection, Buffer Overflow, LDAP injection, CRLF injection, XPath injection, and many more are examples of injection attacks. OWASP defines an injection attack as an injection flaw that allows a hacker to send untrusted data or malware to the program known as an interpreter in the network (owasp 2017). The program further executes the hostile data as a part of code or query that results in alteration of the program. This alteration can lead to further cyber threats, compromising the whole system of victims, data loss, and data integrity (Kaur & Garg 2022).

3.3 Broken authentication and session management attack

The exploitation of a website through broken authentication happens when the user credentials are not appropriately secured. In other words, when a user visits the website, the server provides a unique long session number, commonly referred to as session identifier or session id, to the user for identification on the server. This session id is stored on the user's system for further communication. Whenever the user sends any request to the server, the browser sends the session ID and the request back to the server. Moreover, this session id is used throughout the user's entire session for communication purposes. However, for security reasons, the session id is refreshed every specified interval, for example, 10 s. The problem occurs when a website cannot update the session id after some specified time interval or the session id is visible in the website URL due to poor session management or authentication management. In such a case, the attacker can use this vulnerability as an opportunity to hijack the account by using the unsecured session-id (J. Kaur & Garg 2022). To make predicting the next session ID difficult for the user, the session ID is randomly produced as a long set of letters and numbers. Depending on the requirements of the web application, the session id is stored on the server or the client's web browser (Wikipedia 2021), (Hassan et al., 2018). Most developers choose to store the session inside the authentication cookie on the client side due to the complexity of server-side session management. However, if data quality, legitimacy, and secrecy are not assured, this typical practice creates more hazards. Thus, it allows hackers to compromise the system through the existing loopholes in the website.

3.4 Cross-site scripting attack

Another prominent type of code injection-based online assault is a Cross-Site Scripting attack, commonly referred to as an XSS attack. XSS exposure has affected 80% of web

applications over the globe (Dora and Nemoga, 2021) (Gao et al., 2010). Due to this reason, it has become the most dangerous web vulnerability and a recent open challenge among security researchers and web designers (Geetha and Thilagam, 2020). Hence, research on security in the XSS attack detection model needs to be overlooked (Morzeux and Http-ParamsDataset, 2021) (Gkioulos and Chowdhury, 2021). An XSS vulnerability is the root cause of online data breaches and privacy violations in big organizations. In 2018, the largest airline in the UK- British Airways, suffered a data breach caused due to XSS vulnerability in the online payment module where data of 3,80,000 users was stolen (Gupta and Pooja, 2020). In 2020, Facebook was found vulnerable to an XSS attack, and the flaw was in its login button where window.postmessage () function allowed cross-communication between window objects (Leyden 2020). XSS vulnerability earned $20,000 for security researchers via a Bug Bounty Program. In 2021, more than 98.2 million individuals from the healthcare industry and financial, professional, and retail services were affected by data breaches only in the first half of the year, as reported by CRN news on July 2021 (Novinson Michael 2021). In March 2021, Gmail's AMP (Accelerated Mobile Page) feature was found to be vulnerable to XSS by a group of security researchers (*XSS in Gmail's AMP For Email Earns Researcher $5,000 | The Daily Swig*, n.d.). AMP was introduced by Google in 2019 to make email content dynamic and more interactive for customers. Further in 2022, XSS vulnerability was exposed in Chrome browser extension- Screencastify, WordPress 5.8.3, Microsoft Teams, Google Cloud, and Google Play (*Screencastify Chrome Extension Flaws Allow Webcam Hijacks*, n.d.),(*XSS Vulnerabilities in Google Cloud, Google Play Could Lead to Account Hijacks | The Daily Swig*, n.d.), (*WordPress Stored XSS Vulnerability—Update Now*, n.d.), (*Microsoft Teams Security Vulnerability Left Users Open to XSS* via *Flawed Stickers Feature | The Daily Swig*, n.d.).

Any website is vulnerable to an XSS attack when it accepts untrusted content and inserts it into the benign website without proper input encoding or input validation. The attacker-supplied code can be written in any technology that a web browser supports such as HTML/JavaScript, VBScript, ActiveX, Java, Flash, or any other browser-based technology (Rodríguez et al. 2020). Upon injecting the malicious query into the web browser, the injected script will execute in the confidential zone of the website and thus allowing admin privileges to transmit sensitive information to the attacker. For instance, if a hacker inserts a malicious JavaScript code into the comments section or any other section that accepts some input from the user, and the browser further executes the malicious code/URL with the dynamic content, then it means that website is vulnerable to an XSS attack. XSS vulnerability with high-risk factors can even disclose the user's session cookies, thereby allowing the attacker to compromise the entire user's account. If a web application is exposed to an XSS attack, it is also vulnerable to DDoS (Distributed Denial of Service) and ATO (Account Takeover) attacks. Thus, a single XSS flaw can result in malware installation, session hijacking, sensitive data disclosure, and social engineering attacks. Table 3 lists some of the popular XSS payloads utilized by hackers to compromise a system (Conti et al., 2018) (Rodríguez et al. 2020).

XSS attacks can be classified into four different categories depending upon the nature of the threat vector

Table 3 XSS Payloads

Sr. no	Types of XSS payloads
1	Steal cookies without page redirection
2	Accessing page HTML
3	Creating alert boxes and commands
4	Reauthentication dialogs
5	Injection of Java payload
6	Keyloggers
7	Webpage redirection
8	Tracking browser details
9	Web-page defacement
10	Auto-run exploits when connected to BeEf
11	Network enumeration
12	Ping sweep network
13	Rewriting of links
14	Stealing clipboard
15	Detection of TOR
16	Metasploit AutoPWN

3.4.1 Persistent/ stored XSS attack

A website is vulnerable to persistent XSS attacks when the malicious script injected by the hacker gets stored on the web server and database; hence, the name stored XSS attacks. Since the attack vector injected into the website gets stored in the database, it can affect millions of users visiting the webpage simultaneously. Moreover, the malicious script gets loaded into the system automatically even if the user does not click on any link. Therefore, it is also considered the most dangerous type of vulnerability. Figure 4 illustrates the sample scenario where a victim accesses a vulnerable webpage with a malicious script. As it is crystal clear from the diagram below, a web user loses access to his confidential data when he pays a visit to a vulnerable web page, and a malicious script gets loaded into his system implicitly.

3.4.2 Non-Persistent/reflected XSS attack

A reflected XSS attack is another common attack performed by the hacker. A website is prone to this attack when it echoes back the content or a string inserted by the user in its input fields, such as the search bar, comment section, and log-in text box. When the hacker identifies a reflected XSS flaw, he creates a malicious payload and inserts it into the webpage as a link or shares it directly with the user via email. Once the user clicks on the malicious link, the hacker can steal the user's cookies and login credentials. Both reflected and stored XSS are server-side vulnerabilities. Figure 5 illustrates the basic scenario of the reflected XSS attack where the user clicks on an untrusted malicious link, thinking it came from a trusted source. Within a single click, the victim gets directed to a malicious web page and compromises his system unintentionally.

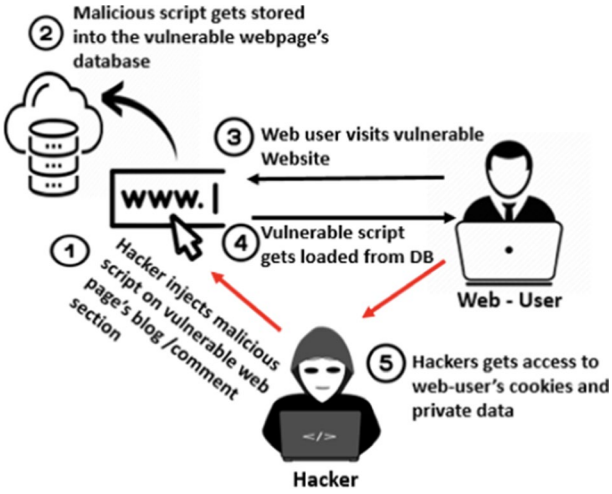

Fig. 4 Persistent XSS attack

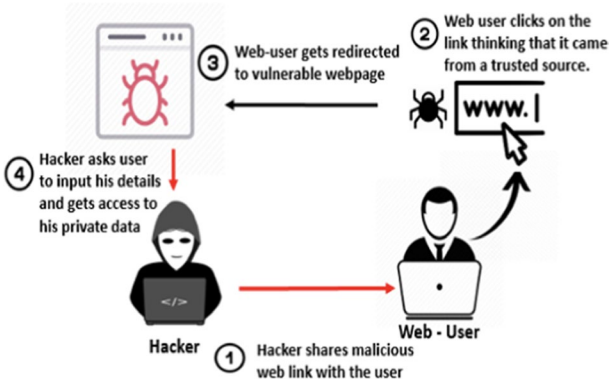

Fig. 5 Reflected XSS attack

3.4.3 DOM-based XSS attack

It is a client-side vulnerability where the malicious script is loaded in the web browser's DOM (Document Object Model). Whenever a victim visits a web page with a DOM vulnerability, injected code can alter the content of DOM and the values of the object's properties. Figure 6 illustrates the sample scenario of a DOM-based XSS attack on a web application. This vulnerability is exploited through client-side scripts such as JavaScript, VB Script, AJAX, ActiveX, JQuery, or any other technology supported by web browsers. There is a common myth among people that AJAX (Asynchronous JavaScript and XML) and JQuery are secure concepts and thus enhance web security. However, with the usage of image tags and AJAX requests, it is possible to execute direct attacks on third-party applications via DOM-XSS (Hickling 2021). The attacker can thus retrieve cookies and sensitive information like user credentials and IP addresses. One thumb rule to detect DOM-XSS flaw is to inject malicious script into the browser and check whether the script is visible in the server code. If the malicious payload is visible in the browser's

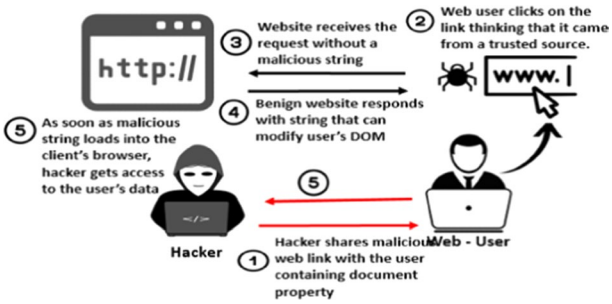

Fig. 6 DOM XSS Attack

page source but not in the server code, it concludes that there is a DOM-XSS flaw in the web page.

3.4.4 Mutation-based XSS attack

In 2013, Dr. Mario found a new type of XSS attack called mutation XSS (Heiderich et al., 2013). Due to the obvious variances in browser interpretation of HTML standards, M-XSS is self-referential, and the iteration is very uncertain. InnerHTML verifies the automated alteration of HTML text inside the browser, allowing the attacker to circumvent the DOM. Google was formerly vulnerable to a mutated XSS attack, which was disclosed to Google by security researcher Masato Kinugawa in February 2019 (Nidecki Tomasz Andrzej 2019). Figure 7 depicts an example of a mutation-based XSS attack scenario. In this attack, when the hacker inserts malicious HTML content into the web application, the web application filters the injected content and passes it to the browser. This further modifies the content of innerHTML while parsing the markups. However, in this process, the browser mutates the contents, and unknowingly it leads to an XSS attack. Owing to this, MXSS is challenging to track.

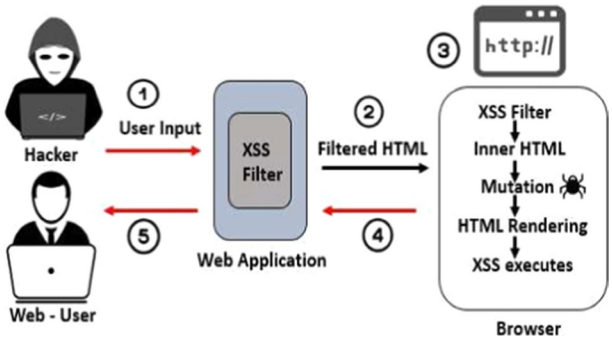

Fig. 7 Mutation-based XSS Attack

4 Cyber security threats and attacks

To combat web security-related issues, researchers have raised an urgent need to utilize machine learning techniques. This is so because machine learning techniques can detect new attacks within a small timeframe without performing any code or file analysis (Khan et al. 2017) (Thakkar & Lohiya 2021). In this section, we will delve deeper into the related work done by several researchers. In the end, we will come out with the research gaps that must be resolved to overcome the vulnerability issues. Artificial Intelligence (AI) has become ubiquitous because of its ability to solve complex problems such as language analysis, speech recognition, machine translation, robotics, cyber defense, and many more. The rising interest in AI is driven by its advanced computing power (Zhang et al. 2021b).

The authors (Jian-hua Li 2021) presented a detailed review of the role of machine learning in cyberspace. The author discussed the various machine learning and Deep Learning algorithms used to combat security issues in web applications. Furthermore, the author studied numerous attacks that an AI model itself may suffer. The researcher also focused on developing a secure AI model by encrypting the deep neural network. Support Vector Machine (SVM) requires more training time, which is a limitation. However, it can be trained with AVL Tree to reduce the training and attack detection time. Attribute-QSSVM and Temporal Attribute-QSSVM have less computational complexity than SVM. Since most of the data contain irrelevant fields and is noisy, reducing the amount of data for yielding effective results by applying dimensionality reduction algorithms is of utmost importance (Laghrissi et al. 2021). Traditional machine learning techniques such as Support Vector Machine (SVM), Decision Trees, Random Forests, K-nearest neighbors, and feature selection techniques uses shallow learning and, therefore, are ineffective at performing well in a real-time environment. The authors (Chauhan et al. 2019) represented a comparison between shallow and Deep Learning methods for predicting the performance of stroke patients. The comparison was made between CNN and traditional machine learning methods, including Ridge Regression (RR), PCA, and SVR (Support Vector Regression), a kernel-based machine learning regression model. Their results indicate that deep Convolutional Neural Networks (CNN) did not perform better than the shallow learning methods. However, their hybrid RR model (with CNN and PCA features as input) outperformed other methods (CNN, PCA+RR, SVR). Moreover, their observations conclude that CNN yields the best results only when more data is available.

On the one hand, machine learning attack detection models can learn the general patterns related to web attacks and can quickly analyze different types of attacks by differentiating their properties using classification algorithms. On the other hand, the performance of machine learning algorithms strongly depends upon the accuracy of feature extraction (Onan et al. 2016) (Onan 2021a, 2021b; Onan et al. 2017) (Onan 2018b) and recognition, size of the training set, and training time. That means that ML algorithms will classify the data samples as per the selected features only, and features that are not mentioned will escape attack detection (Li, 2018) (Fang et al. 2019). Despite its significant progress in recent years, machine learning techniques can be attacked or deceived (Li, 2018). For instance, training or testing samples can be modified, the working environment can be manipulated, and an external model can be added to affect the system's performance. Since the AI model is evaluated upon the testing dataset, it is impossible to verify the accuracy of the AI model in a real-world scenario. Therefore, most of the traditional ML algorithms may generate false true positives and false true negatives. Another limitation of the ML algorithm is that features are compiled manually, which is costly and time consuming

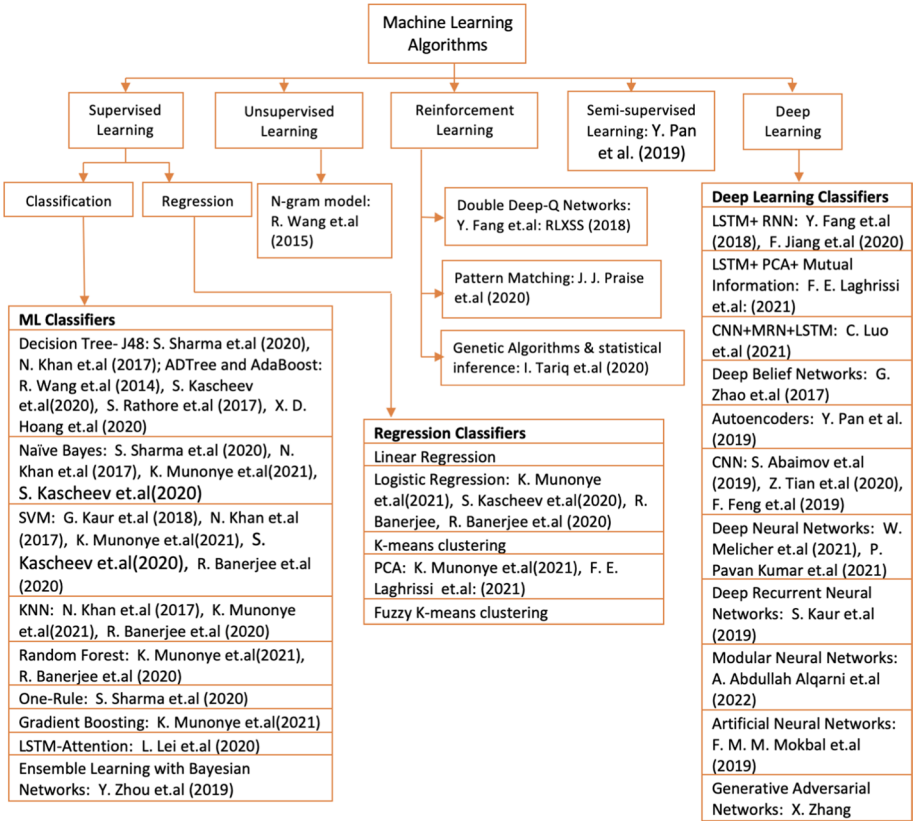

Fig. 8 Classification of Machine Learning Techniques

(Onan 2022). Furthermore, many security mechanisms do not implement a detailed packet analysis, leading to insufficient feature generation to detect zero-day attacks (Wang et al. 2022). However, Deep Learning can effortlessly produce positive results with a large amount of data in relatively less time as the models can be trained without implementing feature extraction (Onan, 2020). The next section presents several aspects of machine learning that can be leveraged in security standards and protocols to safeguard websites from cyber criminals. Figure 9 represents the existing ML techniques available for identifying web attacks. ML techniques are of five types: (i) supervised learning models, which include classification and regression problem solving, (ii) unsupervised learning models mainly cover clustering algorithms such as K-means clustering, PCA, fuzzy-k clustering, (iii) semi-supervised which integrate benefits of supervised and unsupervised learning models while covering graph-based algorithms, self-training models, SGAN, etc. (iv) Reinforcement learning models, and (v) Deep Learning models which include several neural network-based models. The following Fig. 8 represents some of the popular machine learning techniques used by researchers to tackle with XSS attacks detection mechanism.

5 Machine learning techniques for XSS attack detection

Researchers have shifted their focus from traditional XSS attack detection techniques to machine learning-based detection techniques because ML techniques have several advantages over conventional mechanisms such as detecting malicious patterns automatically and higher accuracy scores. Owing to the weaknesses in traditional XSS attack detection mechanisms, researchers are shifting their focus to machine learning based detection approached. The following Table 4 represents the Retrospective analysis and inferences of existing literature with a comparison of best practices followed by the researchers and scope of improvements in their proposed approaches.

XSS attack mitigation techniques using machine learning can be classified into the following categories:

5.1 Supervised machine learning

Supervised learning is the first major category of machine learning prediction models where the labeled dataset is fed to the input section of the learning algorithm. Several techniques have been used from supervised learning to prevent and detect XSS vulnerabilities in web applications. Supervised machine learning is one of the prominent areas leveraged for the implementation of XSS attack detection techniques. Decision trees such as Random Forests, ADTree, J48 along with boosting algorithms are implemented by researchers. Apart from Decision Trees, other supervised models such as Support Vector Machines, Principal Component Analysis, kNN, mutual information, Naïve Bayes and SVM are also popular. Some of them are discussed in the upcoming paragraphs below:

Kaur Gurpreet et al. (G. Kaur et al. 2018) introduced an ML-based model that can identify the malicious attack vector before it is processed by the browser on the victim's system. They used the Linear Support Vector classification algorithm to identify blind XSS and stored XSS attacks. During the features generation phase, authors leveraged the JS events and scripts that the hackers inject to attack the website. The dataset used for experimentation purposes was Linear separable while the experiment was simulated on Mutillidae, a free susceptible website. With a recall value of 0.951 and a false positive rate of 0.111, the detection accuracy is 95.4 percent.

According to S. Sharma et al. (Machine Learning Based Intrusion Detection System for Web-Based Attacks, 2020), feature set extraction is crucial in detecting web-based attacks. Therefore, the authors proposed a feature set extraction approach that, if applied with ML based intrusion detection model, can improve the results to a significant extent. They performed the experiment using CSIC HTTP 2010 dataset and the Weka tool. The experiment was performed in three steps. The first step involves preprocessing the data of the dataset with a python script, while the second step focuses on extracting features from the dataset that contains specific keywords such as GET, POST, DROP, DELETE, MODIFY, UNION, DROP, etc., before feeding it into the Weka for data modeling. In the final phase, the data is fed into the three ML models, namely, J48, OneR, and Naïve Bayes in Weka, where J48 produced the best results compared to other classifiers. The same results were reported by (Alam & Pachauri 2017) regarding the performance of the J48 classifier, where they leveraged Weka tool to detect fraudulent activity in credit card payments. J48 decision tree requires less time to construct the model, prediction accuracy is higher, and the numbers of correctly classified instances are also more. A

Table 4 Retrospective analysis and inferences of existing literature with a comparison:

ML solution	Reference	Best practices followed	Scope of improvement
Supervised learning	(G. Kaur et al. 2018)	Their technique can detect XSS attack before the malicious script is executed on the vulnerable web page	Focus area is only blind XSS attack while DOM XSS, reflected XSS are not considered. Research is carried on using only JavaScript events feature set. Accuracy of attack detection could have been improved
	(R. Wang et al. 2015)	Features are extracted based on keywords and their frequency related to URL, JavaScript and HTML tags. It considered the spreading speed of attack	Covers attacks detection only on social networking sites. Mechanism to prevent overfitting problem and biasness is not covered
	(Kascheev & Olenchikova, 2020)	A comparison of four machine learning algorithms is made: Decision tree, Naïve Bayes classifier, Logistic regression, and support vector machine, where the decision tree showed the best performance rates	Using t-SNE for data dimensionality results in slow computation when applied over a dataset of more than 10000 records. Therefore, it can be combined with PCA to reduce the noise in dataset and to improve the computational speed
	(Machine Learning Based Intrusion Detection System for Web-Based Attacks, 2020)	The authors proposed a feature set extraction approach that, if applied with ML based intrusion detection model, can improve the results to a significant extent. The authors used three models i.e., J48, OneR, and Naïve Bayes in Weka, where J48 produced the best results	Their model does not work on higher layers and considers only HTTP requests. Features such as JavaScript keywords, HTML tags, and malicious URLs and IP addresses could be used to improve the detection results in real time
	(Munonye & Péter 2021)	Researchers focused on fixing OAuth vulnerabilities that can assist in identification of XSS attacks. They utilized Principal Component Analysis and Factor analysis to extract features that are more likely to impact the output	The technique considers only HTTP requests for feature generation. Features such as JavaScript keywords, HTML tags, and malicious URLs and IP addresses could be used to improve the detection results in real time
	(Rathore et al. 2017)	10 ML classifiers were evaluated to measure the performance of the detection model, tree classifiers (Random Forest and ADTree) showed the best results. Features are extracted using URL, HTML and Social networking sites keywords	The approach does handle complex dataset and extracting SNS features is more time-consuming. Considering JavaScript keywords can help improve results in real-time. Overfitting and biasness problems can be handled while working with decision trees

Table 4 (continued)

ML solution	Reference	Best practices followed	Scope of improvement
	(Khan et al. 2017)	Researchers used lexical analysis LALR for generation of JavaScript token and to detect obfuscated code which is used by hackers to avoid attack detection	Performing static analysis on JavaScript code is time consuming. Model may give inaccurate results with large dataset as Decision Tree can result in overfitting. Large dataset could have been used as in real time, dataset used is quite large
Unsupervised learning	(Hoang 2020)	PCA and CART decision tree algorithm is applied for XSS detection. Experiment is executed on real web logs collected from web servers	Overfitting and biasness problems can be handled while working with decision trees. Techniques to detect obfuscated code can be included
	(Fang et al., 2019)	Researchers proposed four new attack escaping strategies: replacement of sensitive words, encoding obfuscation, positional morphological transformation, and adding special character	Technique can be implemented on multiple datasets to get better results in real-time scenarios
	(Tariq et al. 2021)	They used a threat intelligence model, reinforcement learning, GA, and statistical inference, and a combination of GA and statistical inference to guard against any XSS assault	When one of the patterns is altered, slight change in accuracy is observed
	(Praise et al., 2020)	Signature-based pattern matching algorithm is integrated with reinforcement learning to block malicious attacks in cloud environments	Attack escape detection strategy is not applied in the firewall implementation
Deep learning	(Feng et al. 2019)	Researchers implemented a plug-and-play ready-to-use device to detect DOS attacks, XSS attacks, and SQL attacks using DNN, CNN, and LSTM models	KD99 dataset is used which is old dataset. The technique can be implemented with latest datasets to verify the accuracy. Also, the dataset does not consider XSS attack. Techniques to detect obfuscated code can be included
	(Laghrissi et al. 2021)	Comparison between three LSTM classifier-based Intrusion Detection Systems is done: LSTM, LSTM-PCA, and LSTM-MI, where the PCA-based model produced the best results for both binary and multiclass (3 class: Normal, Remote to Local Attack, DoS attack) classifications. PCA and MI is applied to reduce the data dimensions	KD99 dataset is used which is old dataset. The technique can be implemented with latest datasets to verify the accuracy. Also, the dataset does not consider XSS attack. Techniques to detect obfuscated code can be included

Table 4 (continued)

ML solution	Reference	Best practices followed	Scope of improvement
	(Abaimov & Bianchi 2019)	The concept of the pre-processing module, based on the type of keyword or symbol, removes noisy data from input queries so that ambiguous data value pairs from the dataset can be removed. This directly impacts the neural network's performance, thereby increasing the detection accuracy from 75 to 95%	Domain specific semantic knowledge is required. Techniques to detect obfuscated code can be included
	(Yan et al. 2018)	Abstract Syntax Tree is used to extract features of JavaScript code. Chi-square test is done to select the top n-grams	Techniques to detect obfuscated code can be included
	(Pan et al. 2019)	Their method uses RSMT to detect many sorts of assaults with distinct features without manual extraction. Web security domain knowledge is not required. Autoencoders is used to evaluate the model	Technique can be implemented on multiple datasets to get better results in real-time scenarios. Techniques to detect obfuscated code can be included
	(Mokbal et al. 2019)	Their proposed approach is integrated with the dynamic feature extractor and Back-Propagation algorithm for XSS attack detection	Multiple datasets can be used
	(Abdullah Alqarni et al. 2022)	Decoder is used to decode any encoded malicious script obfuscated the attacker	Word2vec is not suggested for XSS attack detection as it is time consuming, therefore, it should be avoided. The dataset used in this study is small and unbalanced, i.e., 100000 benign samples, while malicious samples are only 1000. Attention mechanism can be included to yield maximum results
	(Luo et al. 2021)	A combination of three deep learning-based models i.e., MRN, CNN, and LSTM is utilized to analyze the URL requests for anomalies. Search is carried on real world dataset collected from security company with good amount of records	Using MLP as an ensemble model could result in complex and time-consuming computations
	(Tian et al. 2020)	For data discrimination, the FastText deep learning neural model has been used with some modifications. The model is trained over three datasets: HTTP CSIC 2010, FWAF, and HttpParams dataset	Word2vec is not suggested for XSS attack detection as it is time consuming, therefore, it should be avoided

Table 4 (continued)

ML solution	Reference	Best practices followed	Scope of improvement
	(Kaur & Singh 2019)	Researchers leveraged Misuse Detection Engine (MDE), where the main job is to differentiate the common attacks from the network traffic	Techniques to detect obfuscated code can be included

study by K. Munonye and M. Peter (Munonye & Péter 2021) was solely focused on fixing OAuth vulnerabilities. They utilized Principal Component Analysis and Factor analysis to extract features that are more likely to impact the output. To identify the various challenges related to the domain and OAuth workflow, the Finite State Machine model was implemented. Features were extracted manually because of the unavailability of the existing dataset for the research. For detecting the successful workflows, Gradient Boost Classifier (GBC) was used, which produced 0.82 accuracy and 0.71 value for the ROC curve.

R Wang et al. in (R. Wang et al. 2014) developed an approach that detects XSS worms in Online Social Network web pages. The benign and malicious web pages are quite different and can be differentiated by counting the frequencies of scripting functions in both. In their model, the researchers opted for the ADTree decision tree and AdaBoost.M1 (Adaptive Boosting) algorithms for the classification because of the higher accuracy of ADTree than other decision trees and AdaBoost.M1 produces a strong classifier. A feature extractor model is also developed to automatically capture the features from the webpages because feature extraction and database creation play a crucial role in generating a classification model. Benign and malicious samples are collected from DMOZ and XXSed Database. The researchers extracted four groups of features from web pages, namely, (1) keyword features, (2) JavaScript features, (3) HTML tag features, and (4) URL features, where each of these features further covers sub-features. Researchers in this paper used the Weka tool to generate and evaluate the classification models and the tenfold cross-validation method for evaluation. However, the technique proposed produces a high false-positive rate of 4.20% and gives a shallow detection rate. ADTree generates precision value 0.938, recall value 0.936, F-Measure 0.936 while AdaBoost.M1 gave higher values i.e., precision = 0.941, recall = 0.939, F-measure = 0.939.

S. Kascheev and T. Olenchikova in (Kascheev & Olenchikova 2020) proposed supervised machine learning algorithms to detect XSS attacks using the cross-validation method to train 70% of the data and generate 30% of test samples. The model's performance is evaluated based on correct responses, accuracy, completeness, and F-measure. Their dataset contains 40,000 malicious codes and 200,000 lines of benign code. First, each request is decoded into Unicode characters, and regular expressions are used to extract the query's parameters. However, abnormal queries with many parameters are removed from the dataset. The source data set is represented using Word2Vec. To reduce the data dimension and restore the data nesting structure, T-distributed Stochastic Neighbor Embedding (t-SNE) is used. A comparison of four machine learning algorithms is made: Decision tree, Naïve Bayes classifier, Logistic regression, and support vector machine, where the decision tree showed the best performance rates. Metrics used are accuracy, precision, recall, and F-measure.

R. Banerjee et al. (Banerjee et al. 2020) investigated the use of four ML algorithms, namely SVM, KNN, Random Forest, and Logistic Regression, for the detection of XSS attacks. The true and false values in the dataset were mapped by employing the Logistic Regression model. The experiment was performed using python and scikit library on the dataset having 24 attributes based on URL and JavaScript features. Of the four classifiers, Random Forest Classifier had shown promising results with high accuracy and a low false positive rate. S. Rathore et al. (Rathore et al. 2017) proposed XSSClassifier to detect an XSS attack on social websites such as Facebook and Twitter. Their approach used a machine learning classifier for attack detection. First, the features are extracted from URLs, social networking sites (SNSs), and HTML content. During the evaluation phase, it was

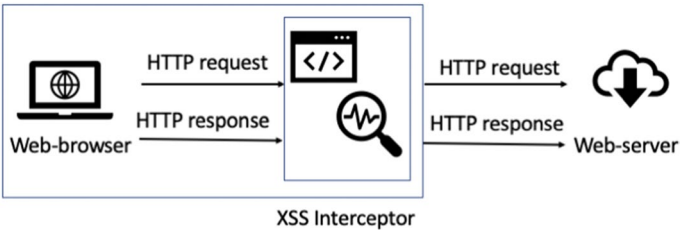

Fig. 9 XSS Vulnerability Interceptor

found that the detection model generates higher accuracy when fed with SNS features to the classifier. While a total of 10 ML classifiers were evaluated to measure the performance of the detection model, tree classifiers (Random Forest and ADTree) showed the best results with 97.2% accuracy and a false positive rate of 0.87 in real-time.

Nayeem Khan et al. in (Khan et al. 2017) also used four different machine learning classifiers: SVM, KNN, J48, and Naïve Bayes for XSS attack detection, where J48 outperformed with 99.22% accuracy when the dataset was split into training and testing data. Moreover, an interceptor plugin is implemented that ensures secure communication between the browser and server, as shown in Fig. 9.

The HTTP GET response is sent to the server via the interceptor. If any malicious activity is found in JavaScript code, the web page is blocked automatically before being sent to the browser. While the approach is browser and platform-independent and efficiently detects XSS attacks in real-time, it is also lightweight in nature and requires minimal runtime overhead. Features to be fed into the classifier are extracted through static analysis. To break down the website's source code into a list of tokens, a yacc parser is used in the detection model. Furthermore, lexical analysis is performed to remove any obfuscated code used by hackers to hide malicious activity. Lexical analysis is a security scanning component that breaks down a source code into tokens before parsing (C.C. Michael & Steven Lavenhar, 2013).

5.2 Unsupervised machine learning

Unlike supervised machine learning, unsupervised learning is trained on the dataset which is unlabeled. Therefore, these models carry the ability to predict unknown hidden patterns and cluster the data without any human supervision. K-means clustering algorithms, PCA, dimensionality reduction, and autoencoders are some of the unsupervised learning models. Now let us discuss further some of the popular unsupervised machine learning models to detect an anomaly or malicious behavior of web applications in the below paragraphs.

(Wang et al. 2015) proposed an improved version of a model motivated by the n-gram model for identifying XSS in online social networks. They utilized a set of correlated data values for the feature extraction purpose. They extracted features based on keywords related to URL, JavaScript, and HTML tags and developed their classifier using the Alternating Decision Tree. Taking into account the spreading rate of XSS attacks in online social networks, they considered the spread speed of suspicious identified features. The detection model is iterated twice in the implementation stage, where they used their proposed model in the second iteration to detect malicious samples. In terms of precision and memory, their strategy was successful.

Laghrissi F. et al. in (Laghrissi et al. 2021) presented a comparison between three LSTM classifier-based Intrusion Detection Systems (IDS): LSTM, LSTM-PCA, and LSTM-MI, where the PCA-based model produced the best results for both binary and multiclass (3 class: Normal, Remote to Local Attack, DoS attack) classifications. KD99 dataset was used and divided into three categories in Google Colab: testing data, training data, and validation data (6:2:2). For evaluation, F-1 score, accuracy (training + testing), sensitivity, and precision are used. LSTM produced more effective results for binary classification than multiclass classification, while MI performed better with four feature sets. LSTM-PCA produced a recall value of 0.99 for both binary and multiclass, precision of 1.0 for binary and 0.99 for multiclass, F1 score of 0.99 for both types of classes, training accuracy of 99.49 for binary and 99.39 for multiclass, and testing accuracy as 99.44 for binary and 99.36 for multiclass. X.D. Hoang (Hoang 2020) proposed a web logs-based machine learning detection model for common web attacks like SQLi, XSS, Path traversal, and CMDi. The HTTP Param dataset contains 20000 payload requests which are used as a training set and 11067 as a testing set. Under the detection stage, URIs are first extracted from the weblogs to add in input further to the detection model using the n-gram approach (n=3). In the second step, URIs are pre-processed to convert them into vector representation by incorporating the TF-IDF method. Finally, the PCA method is implemented to reduce the number of features. The CART decision tree model is implemented for training purposes as it is the fastest and best match for real-time detection systems. For detection purposes, the classifier produced from the training stage is utilized to classify the URI vector and generates either normal or attack results. The experiments conclude that the overall detection rate is 98.52%. To explain, the results for average payload, SQLi, and path traversal only are acceptable; however, for XSS and CMDi attacks, the detection rate (accuracy) is meager i.e., 85.88 and 73.33%, respectively, because of the amount of training data used. Table 5 below represents a confusion matrix that is commonly used to evaluate the performance of XSS attack detection model that predicts whether an instance belongs to a positive (malicious) or negative (genuine) class.

5.3 Reinforcement machine learning

Reinforcement machine learning has shown favorable results in the cybersecurity field. It uses trial-and-error strategy where the learning model performs a sequence of steps and receives a reward based on the outcomes. Algorithms such as SARSA, Q-Learning, and deep Q networks come under this category.

Yong Fang et al. (Fang et al. 2019) developed the RLXSS approach for XSS attack detection using reinforcement learning. The authors came up with the idea of implementing the adversarial and retraining models, where the former is supposed to deal with adversarial attacks, and the latter would re-identify the malicious samples fed by the former model. For optimization reasons, adversarial samples collected from the adversarial model were put into the retraining model. Apart from this, four new attack escaping strategies were also proposed: replacement of sensitive words, encoding obfuscation, positional morphological transformation, and adding special characters. Ultimately, this technique converts the XSS escape attack into a fleeing tactic. Their methodology can detect adversarial assaults that are undetectable by white-box and black-box testing. Both white-box and black-box situations are considered when determining the reward function. The reward for the white-box environment is decided based on confidence level, while for the black-box model the parameter for reward is whether it escapes attack detection or not. The following reward function is applied:

Table 5 Confusion matrix

		Actual values	
		Genuine	Malicious
Predicted values	Genuine	True negative	False positive (Type-I error)
	Malicious	False negative (Type-II error)	True positive

$$r_t = \begin{cases} result * score, \ if \ black_{box} = 1 \\ \frac{1 - result}{1 - threshold} * score, \ if \ white_{box} = 0 \end{cases} \quad (1)$$

where 1 indicates True value, 0 indicated False value, the result is the feedback value for a reward under a black-box environment, the threshold is the confidence value under a white-box environment and the score indicates success in escape. To evaluate their model, DR (Detection rate) and ER (Escape rate) are used and their formulas are mentioned below:

$$DR = \frac{Count \ of \ malicios \ samples \ detected}{Total \ count \ of \ malicious \ samples} \quad (2)$$

$$ER = 1 - DR \quad (3)$$

After the calculation of ER and DR, these evaluation metrics are further tested using SafeDog, XSSChop, LSTM, ADTree and AdaBoost where the LSTM model gave the higher results. However, the results were slightly lower than SafeDog and XSSChop.

Praise J et al. (2020) in (Praise et al. 2020) proposed an effective firewall filtering method that utilizes machine learning for malicious activity. Signature-based pattern matching algorithm is integrated with reinforcement learning to block malicious attacks in cloud environments. By using a two-way pattern-matching strategy, results are calculated in less time complexity. Two-way pattern marching compares the signature of every incoming packet for possible attacks. Furthermore, DPI, a packet filtering technique, is utilized to scan the payload and header content of IP packets that are passed from public networks to private networks.

Figure 10 represents Reinforcement based and pattern-matching RLPM architecture for malicious packet filtration by the firewall. In RLPM, the input data is first passed to the firewall, and firewall log then further classifies the data based on log information to generate new rules. The new rules are further compared with performance log data to generate optimized rules. These updated rules are further fed into the firewall rule database for classifying the data packet as malicious or non-malicious. The process is repeated for all the incoming packets passing through the network and their signatures are matched with relevant packet signatures available in the database. If a signature is not matched, the packet is blocked to enter the network. ECDSA is used to generate the dataset.

In 2021, (Tariq et al. 2021) created a hybrid solution for combating XSS in web applications. They claim to be the first to combine a machine learning approach with a metaheuristic algorithm, such as a Genetic Algorithm. They used a threat intelligence model, reinforcement learning, GA, statistical inference, and a combination of GA and statistical inference to guard against any XSS assault. Their findings show that using only genetic algorithms to identify XSS attacks leads to a large number of false positives. XSS

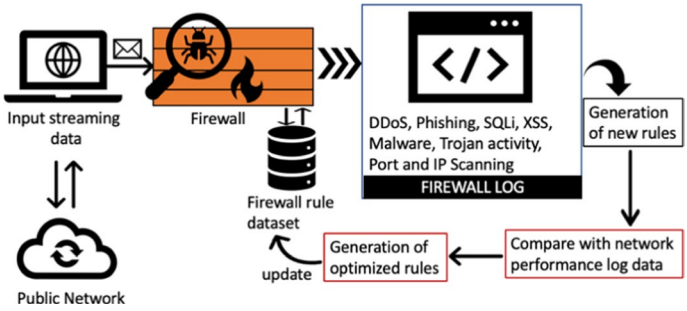

Fig. 10 RLPM based firewall Architecture

vulnerabilities can be accessed via static analysis, dynamic analysis, or hybrid analysis. However, one limitation of this approach is that only limited raw inputs must be fed into the system to reduce the training time. Although there are innumerable techniques available for the detection of XSS attacks such as static analysis, dynamic analysis, hybrid analysis (Mohammad & Reza 2017), input filtering methods, output escaping techniques, and signature-based detection. However, these techniques have proved to be ineffective and time-consuming due to numerous reasons. First and foremost, cybercriminals have developed stealth techniques to bypass the detection models. To explain, attackers use evasion techniques (Jiang et al., 2020) (Li, 2021) and HTML/ URL/ Hex/ Base64/UTF-7 encoding (Kascheev and Olenchikova, 2020) (Conti et al., 2018) to bypass static filters, escape sequences, and comment symbols. In addition, although the static analysis method can detect vulnerability before the code is exploited, it has high error and false-positive rates (Katsikeas et al., 2021). Moreover, signature-based detection, heuristics, firewalls, and dynamic analysis fail to detect new types of modern attacks. Antivirus software whether online or offline implements signature-based attack detection models and hence, requires subsequent updates of the attack vector definitions. Signature-based detection tools fail to keep an eye on new types of attack vectors as the definition of a new attack is unavailable in their detection mechanism. In signature-based detection methods, the signature of the attack is captured and its pattern is then stored in its database to protect the device against the same type of attack in the future. Owing to this, machine learning techniques have gained attention for the past decade (Kaur et al., 2018) and researchers are enlisting ML models in the detection of malware. They can even prove to be fruitful in the detection of threats that are highly dynamic (Jiang et al., 2020).

The following evaluation parameters are used in machine learning XSS attack detection models:

Confusion matrix:

Another important term in machine learning prediction models is a confusion matrix. It's an N-by-N matrix deployed for assessing the efficiency of prediction algorithms where columns represent actual values, and rows represent predicted values, as depicted in Table 6.

(i) Accuracy

The number of observations accurately recognized from the total number of predictions is called accuracy. Accuracy is calculated using the following Eq. (4):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Even if the model's consequent value is 99%, accuracy alone is insufficient to judge its success. The reasons are two-fold. Firstly, accuracy will always result in a higher score if the dataset is imbalanced as results will be higher for the class with the majority of samples (Onan 2019a). For instance, if the total number of benign links is 90 and malicious links are only 10, you will consistently achieve accuracy greater than 90%. That's what an imbalanced dataset means. Therefore, in this case, accuracy alone is inefficient. Secondly, if multiple classes exist in the dataset, predicting accuracy for each category from the overall accuracy result becomes ambiguous. Hence, it is always suggested to include precision, recall, and F1-score to verify the model's performance. (Note: FN refers to missed vulnerabilities)

(ii) Precision

Precision is the overall number of positive values divided by the total number of anticipated positive values. Precision is calculated via the following Eq. (5):

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

(iii) Detection rate

The detection rate is the percentage of true positive values that are accurately anticipated. This is used to detect the malicious samples. Finding a balance between accuracy and detection rate is critical. It can be calculated through the following Eq. (6):

$$Detection\ rate = \frac{TP}{TP + FN} \tag{6}$$

(iv) Escape rate

The escape rate is the percentage of malicious attack vectors incorrectly identified as benign payloads. The following Eq. (7) is used to determine the escape rate:

$$Escape\ rate = 1 - Detection\ rate \tag{7}$$

(v) F1-Score

The F1-Score is used to achieve a balance of accuracy and recall levels. This statistic accounts for both false positives and false negatives without altering the prediction model's accuracy score. The following Eq. (8) is used to determine the F1-score:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

(vi) FP-Rate

The false positive rate is the percentage of incorrectly identified attack results. The following Eq. (9) is used to determine the FP- rate:

Table 6 Comparative analysis of existing web-attack and XSS attack detection techniques as per the literature survey

ML solution	Reference	Data set	Platform	XSS included	Accuracy
Supervised learning	(Kaur et al. 2018)	Linear separable dataset collected from GitHub, GBHackers, XSS-payloads	Vulnerable website Mutillidae	✓	95.4%, 95.1% recall, false positive rate 0.111
	(Syarif & Gata 2018)	KDD CUP 1999 (*KDD* Cup 1999 *Data*, n.d.)	MATLAB	✗	99.9% accuracy for DoS
	(Dada 2017)	KD99	MATLAB	✗	98.55% classification accuracy
	(Olalere et al. 2016)	23582 malicious and benign URLs collected manually	Weka data mining tool	✗	96.95%
	(Kokila et al. 2015)	2000 DARPA (*2000 DARPA Intrusion Detection Scenario Specific Datasets | MIT Lincoln Laboratory*, n.d.)	Python	✗	95.11%
	(Kascheev & Olenchikova 2020)	40,000 malicious samples ((XSS Archive), 200,000 benign samples (Kaggle)	Python	✓	93.70% accuracy for Decision Tree
	(Machine Learning Based Intrusion Detection System for Web-Based Attacks, 2020)	CSIC HTTP dataset 2010: 36000 normal samples and more than 25000 malicious sample	Python, Weka	✓	94.7% Precision, 94.5% Recall, 93.4% F1 Score with J48 ML Classifier using Weka tool
	(Munonye & Péter 2021)	Set I-manual dataset generated using fuzzing technique, set II- PHP security vulnerability dataset	DEKANT tool	✓	Gradient boosting classifier generated 99.4% accuracy, 80.34% precision, 74.70% recall, 71.71% F1-score

Table 6 (continued)

ML solution	Reference	Data set	Platform	XSS included	Accuracy
	(Rathore et al. 2017)	XSSed (*XSSed | cross site scripting (XSS) attacks information and archive*, n.d.) (benign), Alexa (*Alexa—Top Sites*, n.d.) (malicious), Elgg (*introducing a powerful open source social networking engine*, n.d.) (SNSs infected by XSS worm)	Weka	✓	Accuracy 97.2%, false positive rate 0.87
	(Khan et al. 2017)	409 malicious samples, 1515 benign samples	Weka	✓	Accuracy 99.22% with J48, 97.14% with KNN
Decision- Tree Based Cyber Security	(Vuong et al. 2015)	Manual	R programming	✗	86.9%
	(Moon et al. 2015)	Manual	Weka	✗	84.7%
Neural Networks based solution	(Gao et al. 2010)	DOS	MSU SCADA security laboratory	✗	97.1%
	(Vollmer & Manic 2009)	snort.org, emerging threats	Nemesis	✗	60% (DoS, misc activity, attempted-recon)
Unsupervised Machine Learning Solutions	(Laghrissi et al. 2021)	KD99	Google Colab	✓	Model based on PCA generated best accuracy 99.49%, 99.15% Recall
	(R. Wang et al. 2015)	DMOZ and XSSed dataset	Python	✓	90% Precision, 96.5% Recall
	(Shahid et al. 2012)	Not mentioned	MATLAB	✗	99% fault-classification, 100% fault-detection
	(Hoang 2020)	The HTTP param dataset (*GitHub—Morzeux/HttpParamsDataset*, n.d.)	Python	✓	98.56% Accuracy
Reinforcement Machine Learning Solutions	(Fang et al. 2019)	XSSed (33,426 samples)	Python3, Keras-rl, and OpenAIGym	✓	99.5% Accuracy, 97.9% Recall, 98.7% F1-score

Table 6 (continued)

ML solution	Reference	Data set	Platform	XSS included	Accuracy
	(Tariq et al. 2021)	Deurgan dataset	NetBeansIDE 8.0.2 (Java)	✓	Accuracy 99.89%
	(Praise et al. 2020)	Manually constructed	Web	✓	96.7% Efficiency
Deep Learning Solutions for defending against cyber-space attacks	(Dali Zhu et al. 2017)	Automatic using flowdroid	Android application	✗	96.76%
	(Zhao et al. 2017)	KDD CUP 1999	MATLAB	✗	99.14%
	(Vinayakumar et al. 2018)	AndroidManifest	GPU enables TensorFlow in Ubuntu 14.04	✗	93.9% on dynamic analysis, 97.5% on static analysis
	(Feng et al. 2019)	WAF logs and KDD Cup 1999	Python, NS-2	✓	98%
	(Abaimov & Bianchi 2019)	SQL payload dataset, XSS payload dataset (Github repository)	Python	✓	95% accuracy, 99% precision, 92% recall
	(Pavan Kumar et al. 2021)	Kaggle benchmark dataset (phishing.csv)	Python	✗	94.8% accuracy with loss of 0.2024%
	(Yan et al. 2018)	1336 normal apps, 422 vulnerable apps	Python with Keras on Ubuntu 14.04 OS	✓	97.55%
	(Pan et al. 2019)	XSSed	Weka	✓	F1-score 91% (No domain knowledge)
	(Mokbal et al. 2019)	Manually collected using web crawlers (148,157 records) + XSSed + Open Bug Bounty (38,569 malicious samples)	Python 3.6.7 + Keras 2.2.4 + TensorFlow 1.12.0	✓	Accuracy 99.32%, detection probability98.35%, False positive rate 0.3%, AUC-ROC score 99.02%
	(Abdullah Alqarni et al. 2022)	figshare.com: 1000 malicious samples and 100,000 benign samples	Weka	✓	Accuracy: 99.96%

Table 6 (continued)

ML solution	Reference	Data set	Platform	XSS included	Accuracy
	(Luo et al. 2021)	HTTP CSIC 2010, CSIC 2020	DVWA, sqlmap, burpsuite, XSStrike	✓	Accuracy: 0.9947, TPR: 0.9929, FPR: 0.0033, Precision:0.9970
	(Tian et al. 2020)	HTTP CSIC 2010, FWAF, HttpParams	Python, tensorflow library	✓	Accuracy: 0.9941, TPR: 0.9891, DRN: 0.9955
	(S. Kaur & Singh 2019)	NSL-KDD and CICIDS 2017	Not mentioned	✓	Accuracy: 0.991
	(Jiang et al. 2020)	NSL-KDD	Tensorflow (v1.2)	✗	Accuracy: 0.9894

$$FP\ Rate = \frac{FP}{FP + TN} \tag{9}$$

6 Recent developments in XSS attack detection

This section discusses the recent developments in XSS attack detection techniques. Several researchers have utilized Genetic algorithms, hybrid approaches, neural network models, Deep Learning, and dynamic analysis-based approaches to optimize the accuracy of XSS attack detection. Machine learning algorithms have been increasingly becoming popular for detecting and preventing XSS assaults, although they are ineffective in innovative and diverse XSS attacks (Tariq et al. 2021)(Zhou & Wang 2019)(S. Kaur & Singh 2019). Deep Learning models have proved their efficiency by improving accuracy, reliability, and scalability in cybersecurity applications (Dixit & Silakari 2021) (G. Zhang et al. 2022) (Onan & Tocoglu 2021) (Onan 2019b). Therefore, security researchers are exploring novel methods of web attack detection using Deep Learning (Luo et al. 2021). In Deep Learning, the word 'deep' refers to multiple hidden layers (up to 150) inside neural networks (Geetha & Thilagam 2020). The learning task is more difficult since data is handled at each layer, and thus learning task becomes more complicated. The authors provided a study on Deep Learning approaches for cybersecurity (Dixit & Silakari 2021). Deep Learning detection techniques are separated into three categories: supervised, unsupervised, and reinforced.

The authors (Pavan Kumar et al. 2021), leverage Deep Learning techniques and Swarm based Intelligence to detect website phishing attacks. The primary purpose of the study is to use the Binary Bat Algorithm to distinguish fraudulent URLs from valid ones and to enhance speed using the Adam optimizer. After pre-processing, 30 features are extracted for the classification, such as URL length, IP address, user ID, iframe, port, redirect, mouseover, etc. SI-BBA model is evaluated using loss and accuracy parameters where loss is 0.2%, and accuracy is 94.8%. F. Jiang et al. (Jiang et al. 2020) presented a scheme that detects DoS, U2R, Probe, and R2L attacks using multiple channels. This is the only paper that has implemented multi-channel feature extraction and training for attack detection. Their model also considers new variants of threat vectors. They used the NSL KDD training dataset to evaluate the performance of their model. The data is preprocessed first during the training stage and second during the testing stage. The multi-channel training is leveraged to generate classifiers based on LSTM-RNN neural networks. Their model is evaluated on three parameters: Detection rate, False alarm rate, and accuracy. F. Feng et al. in 2019 (Feng et al. 2019) implemented a plug-and-play ready-to-use device to detect DOS attacks, XSS attacks, and SQL attacks using DNN, CNN, and LSTM models. If a malicious payload is detected, an alarm is produced that could stop the spread of the attack to other sites. LSTM performed best by generating 98% accuracy however took more time in training. The authors experimented with KDD Cup 99 dataset and WAF (Web Application Firewall) logs for attack detection.

S. Kaur et al. (S. Kaur & Singh 2019) leveraged a hybrid model based on a deep recurrent neural network that not only searches for malicious signatures but also performs anomaly detection for web attacks and named their approach as D-Sign. Their hybrid model consists of three sub-models: (1) Misuse Detection Engine (MDE), where the main job is to differentiate the common attacks from the network traffic, (2) Anomaly Detection Engine (ADE) identifies the change in the normal pattern that could occur due to presence of anomaly and (3) signature generation for malicious vector through Signature Generation

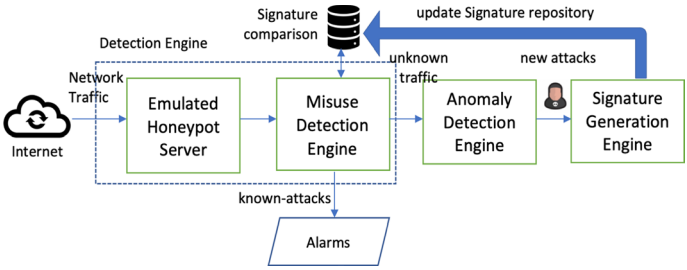

Fig. 11 Overview of D-Sign

Engine (SGE). Figure 12 illustrates the execution flow of their proposed XSS attack detection model.

In Fig. 11, emulated honeypot server acts as an emulated vulnerable server, attracting hackers to the server for executing attack vectors. The NSL-KDD and CICIDS 2017 datasets were utilized in the experiments. The honeypot server catches any suspicious behavior on the network and sends it to MDE as quickly as possible as the sensor detects it. MDE further analyses the captured network packets using the rule-based approach for any suspicious activity and, if found generates an alarming signal. If no suspicious packet is found, traffic is forwarded to ADE for further analysis. ADE leverages a Deep Learning-based model. i.e., LSTM to classify normal and malicious packets in the incoming traffic.

In 2020, Z. Tian et al. (Tian et al. 2020) proposed another approach for web attack detection that can be applied to multiple CNN models on cloud-based edge devices. Their technique involves preparing data to be further fed to the data discrimination model, and finally, action to be taken for the normal or malicious requests. URLs are converted into vector form in the data preparation stage using the CBOW word2vec model. For the optimization of the CNN model, the ReLU function, dropout layers, and pooling layers are leveraged in CNN architecture. Some changes are also implemented in Residual Network, while ResNet is used to detect malicious URLs. For data discrimination, the FastText Deep Learning neural model has been used with some modifications. The model is trained over three datasets: HTTP CSIC 2010, FWAF, and HttpParams dataset. W. Melicher et al. (Melicher et al. 2021) focus on DOM XSS vulnerabilities in the browser, which predominantly executes dynamic JavaScript code using Deep Learning. The authors presented a strategy of combining the taint tracking method with DNN to detect DOM-based XSS attacks. First, taint tracking is applied to identify vulnerable functions, and then, a heuristic is applied to those functions to confirm if they are vulnerable. They found that a relatively small deep neural network with four layers can be an effective pre-filter for taint tracking. However, on the negative side, the algorithm captures only 50% of confirmed vulnerabilities when used alone. The observed vulnerabilities may not be applicable to browsers other than chromium v57. The restrictions of the dynamic analysis utilized for labeling also limit the data. Moreover, the paper does not consider attacks on machine learning algorithms, such as evasion and poisoning attacks. Moreover, there is a massive class imbalance across labels in training data. A study by Abdullah Alqarni A. et al. (Abdullah Alqarni et al. 2022) proposed a modular neural network approach based on neural networks. Using a deep neural network eliminates the need for security domain-related knowledge and expertise, as illustrated in Fig. 12.

First, the pre-processing is done to remove noisy data and to transform the decoded code into the original format. The first stage of pre-processing consists of the execution

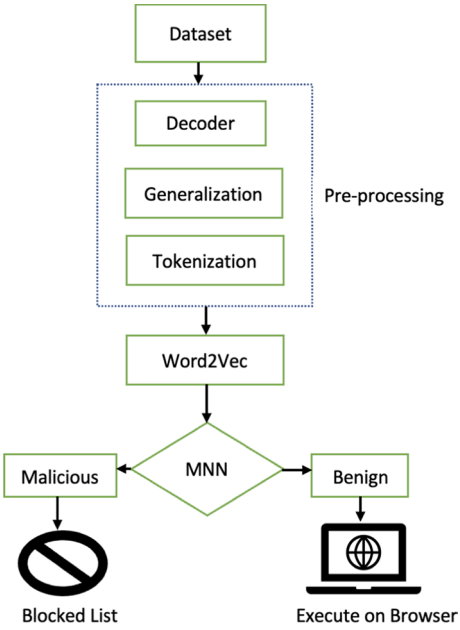

Fig. 12 MNN-XSS Architecture

of sub-three steps, namely, data decoding, generalization, and performing tokenization to reduce the data length. After pre-processing, Word2Vec and CBOW are deployed for the numerical representation of code in simple plain text. Apart from word2vec and bag of words, improved text analysis schemes could have been utilized that cover using Deep Learning, unsupervised learning algorithms (Onan 2019c), ensembled approach, genetic algorithms (Onan & KorukoGlu 2017), and swarm optimization (Onan 2018a) to get improved results in feature extraction. Feature selection is performed through Pearson's Correlation method. Equation 8 depicts Pearson's Correlation method.

$$r = \sum \frac{\left(\left(x-\bar{x}\right)\left(y-\bar{y}\right)\right)}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}} \tag{10}$$

Finally, MNN is applied to break down the features into further sub-features. For this, MNN implements the divide and conquer method. MNN (Shukla et al. 2010) can merge different types of independent neural networks and produce the desired output in the form of a single modular network, as illustrated in Fig. 13.

Since the sub-network acts as a sub-module to build a vast network, MNN is named a modular neural network. The challenging issue of processing a significant amount of training data by ANN is simplified by MNN. If the MNN network generates an output value greater than 0.5, then the code is marked as malicious, else benign. MATLAB is used for the implementation of MNN, while Weka is used for feature selection. The study achieved exceptional results where the accuracy score is 0.9996, Precision score is 0.9995, recall score is 0.9991, and F1-score is 0.9992 for malicious code. However, the dataset used in this study is unbalanced, i.e., 10000 benign samples, while malicious samples are only 1000.

L. Lei et al. (Lei et al. 2020) added an attention mechanism to the hidden layer of the already developed LSTM model to improve the attack detection rates. LSTM (Yu et al.

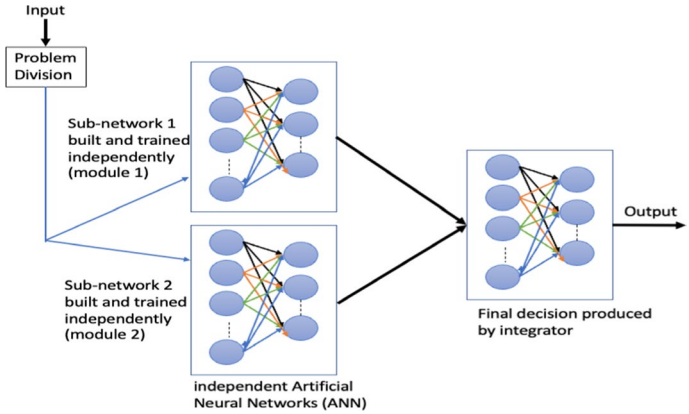

Fig. 13 Modular Neural Network (MNN) Architecture

2019) is an advanced Recurrent Neural Network with four hidden layers capable of remembering the state values for long distances via cell states. A neural network is a model of computation biologically inspired by the human brain and consists of artificial neurons with a set of edges between each neuron (Lipton et al. 2015). Figure 14 represents RNN architecture where it is the input sequence, and h_t is the output sequence.

LSTM includes a hidden layer with the complex structure of gates as a permanent solution to the vanishing gradient problem faced by RNNs. Although RNNs also have a single hidden layer, their design is simple. As RNNs have short-term memory, it is easier to forget the old information while covering the long-distance and may miss important details. Unlike RNNs, LSTM, an artificial RNN, can regulate the flow of information via their internal mechanism known as gates, as depicted in Fig. 15.

The attention mechanism, which V. Mnih introduced in 2014 (Mnih et al. 2014), focused on the part of a sentence while performing the translation. The attention model was first used in machine translation and can be applied to other applications. Firstly, the data is preprocessed and cleaned using a decoder. Further, tokenization is applied to reduce the data dimension as a part of preprocessing. Secondly, word2vec is applied for the feature extraction process and maps them to feature vectors for vectorization. Finally, the LSTM model is trained and tested. For evaluation, the LSTM-Attention model is compared with three classifiers: ADTree, AdaBoost, and SVM (Support Vector Machine), where their model achieved higher Precision, Recall, and F1 as 99.3%, 98.2%, and 98.5%, respectively. The model also performed better than other Deep Learning models, such as Recurrent Neural Networks, Gated Recurrent Unit, and LSTM. The layout of the LSTM attention model is illustrated in Fig. 16. Comparative analysis of existing literature has been covered in Table 6.

A study by Yong Fang et al. (Fang et al. 2018) used a decoder and word2vec for data cleaning and feature extraction, respectively. The data is then fed into an LSTM neural network model. Finally, the suggested method's performance is compared to that of the ADTree and AdaBoost algorithms using a tenfold cross-validation methodology. Using DeepXSS, they were able to get results with 99.5% accuracy. However, the limitation of their approach is the high time complexity. Rubio Yan et al. (Yan et al. 2018) proposed hybrid Deep Learning based on the neural model HDLN to detect XSS attacks in hybrid mobile applications. First, n-gram features are generated from AST, followed by key

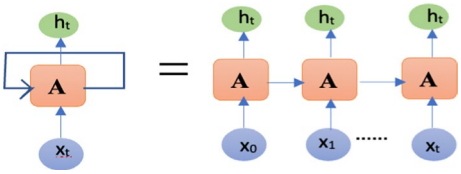

Fig. 14 Recurrent Neural Network Architecture

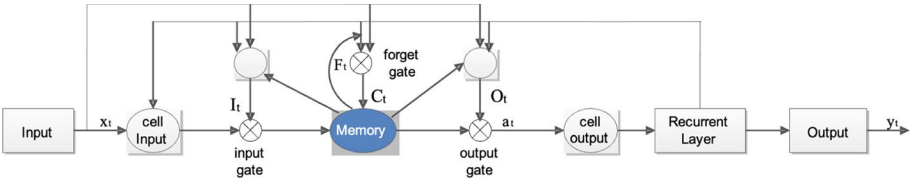

Fig. 15 LSTM Architecture

feature selection. Furthermore, a fivefold cross-validation technique is employed to validate the classification model. While the best accuracy is achieved with eight hidden layers, the precision of malicious data is 99.24%. Results improve as the number of hidden layers is increased. According to Yao Pan et al. (Pan et al. 2019), traditional intrusion detection systems differentiate between normal and malicious traffic depending on supervised machine learning. This necessitates a massive amount of data required for training the ML model, which is expensive, inaccurate, and time-consuming. Furthermore, intrusion detection technologies used to protect against online threats frequently need a thorough understanding of the security domain. Furthermore, the current unsupervised machine learning method necessitates manual feature selection and has a high probability of false positives. The researchers claim that Deep Learning can automatically make categorization and feature selection. Their method uses RSMT to detect many sorts of assaults with distinct features.

(Mokbal et al. 2019) have proposed MLPXX: a multilayer perceptron scheme. Their proposed approach is integrated with the dynamic feature extractor and Back-Propagation algorithm for XSS attack detection. Three models were developed that work together to detect XSS attacks. The researchers constructed a large dataset by selecting the optimal approach of random walk and random jumping method for the training and testing of the samples. Apart from this, a novel technique for dynamic feature extraction and a Multilayer Perceptron for the detection has also been proposed in this paper. Their model has achieved promising results with a detection probability of 98.35%. Data is collected manually using python script scraping framework-based web crawler; extracting the raw data from websites and deleting duplicate samples. The authors extracted the raw data using their web crawler from XSSed and Bug Bounty websites for malicious data. The authors also suggested avoiding using word vectors for text sample representation as XSS code is a more generalized JavaScript code, and hence, word vector often is time-consuming for XSS. The BeautifulSoup library and HTML5lib parser are integrated with the model to extract the HTML raw files; for tokenization and extraction of abstract syntax tree from the JS code, Esprisma JavaScript parser is utilized. The model developed is composed of three sub-model having different feature sets such as HTML features (tags, events, attributes) for developing attack vectors, JavaScript features, and URL features. Their results improved on adding the additional hidden layer where mean accuracy increased to 0.991195 and

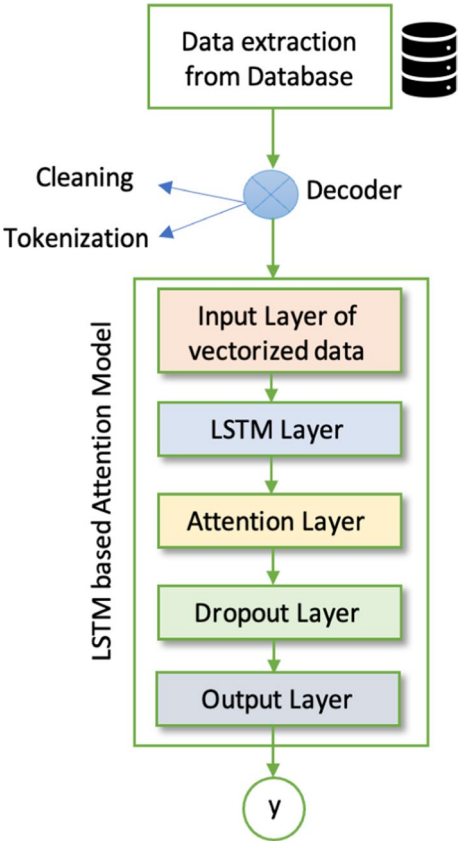

Fig. 16 LSTM Attention model for XSS attack detection

decreased variance value to 0.000759. The experiment was executed on Python framework 3.6.7 and Keras (2.2.4) with Tensorflow (1.12.0). For Non-XSS class, precision is 0.9924, detection rate (True Positive Rate) is 0.9969, F-score is 0.9953. While for XSS class, precision is 0.9921, detection rate is 0.9835, F-score is 0.9877, and false-positive rate is 0.0031.

Abaimov & Bianchi (2019) proposed a supervised approach based on Deep Learning for code injection attacks such as SQLi and XSS. Their model achieved an overall accuracy of 95% with 99% precision and 92% recall value with a preprocessing module that can be integrated with the detection model before the training phase or in the online query injection phase. The concept of the pre-processing module, based on the type of keyword or symbol, removes noisy data from input queries so that ambiguous data value pairs from the dataset can be removed. This directly impacts the neural network's performance, thereby increasing the detection accuracy from 75 to 95%. Furthermore, a local search optimization algorithm is used in CODDLE to automate the process of parameter selection and of effective neural network shape that otherwise requires considerable time for trial and error. Figure 17 represents the CODDLE architecture in brief.

C. Luo et al. in (Luo et al. 2021) proposed a Deep Learning and ensemble learning based web attack detection model for IoT (Internet of Things) networks. A combination of three Deep Learning-based models i.e., MRN, CNN, and LSTM is utilized to analyze the URL requests for anomalies. CBOW and TF-IDF text analysis algorithms are employed

to transform URL requests into vectors. Finally, the results produced by Deep Learning models are fed into the MLP ensemble classifier to improve the predictive model results. The experiments were performed on DVWA vulnerable website and used HTTP CSIC 2010 dataset and another dataset provided by the security company. Their model achieved 99.47% of accuracy and 99.70% of precision. However, the performance score was reduced when CSIC 202 dataset was applied. Moreover, the model does not consider attacks on ML models.

7 Discussions, challenges and opportunities

Our study was solely based upon the clarification of the following research questions.

RQ1: What new advances have been made in the realm of XSS attack detection?
RQ2: Why are traditional techniques not effective at combating XSS attacks?
RQ3: How can XSS attacks be detected and prevented using Machine Learning techniques?
RQ4: What are the most popular and effective cyber-attack mitigation techniques used by active researchers?
RQ5: What are the challenges and opportunities for young researchers in the field of cyber security?

During our survey, extensive analysis of recent advancements is observed that suggests that for RQ1 and RQ4, several researchers are using Deep Learning in the web security field, where hybrid neural techniques perform better than other techniques. Apart from this, some researchers are focusing on merging metaheuristic algorithms with Deep Learning algorithms for cyber-attack detection as metaheuristic algorithms can optimize Deep Learning-based solutions (Akay et al. 2021). With the ever-increasing usage of web access and advancements in the digital world, hackers have started implementing new techniques and threats to deceive the victim. The adoption of new networking terminologies such as SDNs, OSNs, IoT, and Cloud computing induces the need for advanced intrusion-detection systems based on machine learning techniques. One of the most critical challenges is to defend against cyberattacks by searching for vulnerabilities before the attacker can exploit them (Conti et al. 2018). Furthermore, the heterogeneous nature of new attacks and anti-forensics-evasion techniques used by hackers makes it challenging to detect and mitigate cyberattacks. Most XSS attack detection models based on machine learning algorithms always assume that the ML attack detection environment is benign. However, this assumption is not always valid (Fang et al. 2019), and the adversary can take advantage of this assumption by targeting the training data (poisoning attacks) or testing data (evasion attacks) (Pitropakis et al. 2019) as illustrated in Fig. 18.

Moreover, according to X. Zhang et al. in (X. Zhang et al. 2020), many Deep Learning algorithms are vulnerable to adversarial attacks. There is limited research on defense mechanisms against adversarial attacks on machine learning models. While doing this survey, we found only five articles (Chen et al. 2019; Fang et al. 2019; Zhang et al. 2020),(Q. Wang et al.2022a 2022b), (Aldahdooh et al. 2022) focusing on adversarial attacks in web security based upon machine learning algorithms in the context of neural networks. This answers our RQ1 and RQ2. For RQ3, it has been observed that to prevent XSS attacks,

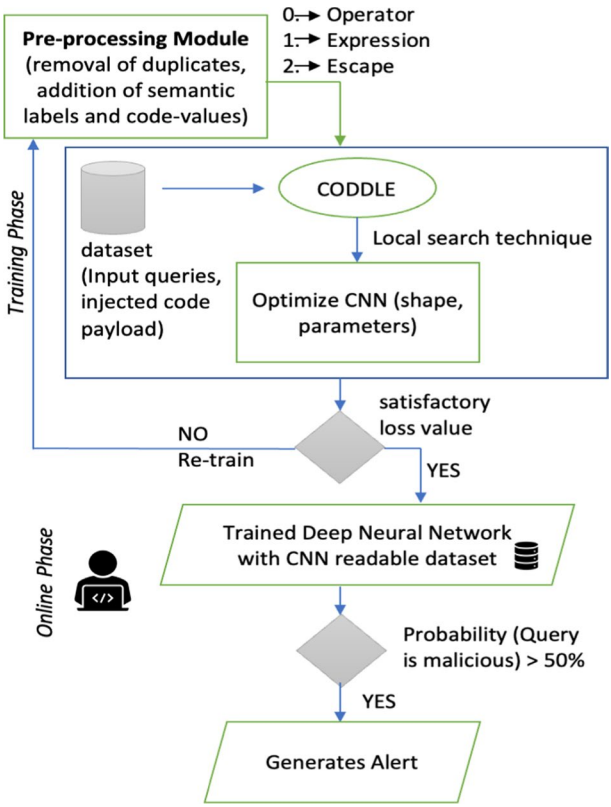

Fig. 17 CODDLE XSS attack detection flow diagram

proper input sanitization must be done at the backend of a website, password strength must be kept as strong, the web developer must enforce high encryption on the network traffic, and regular updates in security patch must be performed by the web application owner. Moreover, web users must be warned about online threats via emails, or text messages as most often attack is executed by sending a malicious link to the user either on their email accounts or via text messages.

Adversarial attacks have affected the performance of several ML-based applications, such as intrusion detection systems, credit card fraud detection, biometric recognition system, and recommendation systems. To defend against these attacks, Adversarial Learning has also become popular among the cyber research communities over the past two decades [80]. In addition, innumerable dynamic web applications leverage the use of their-party tools and components for user comfort such as AJAX, JavaScript, HTML /HTML5, XHTML, RDF, OWL, etc. (Tekli 2021). However, these trends give birth to new hidden vulnerabilities that require extensive research and analysis to get exploited. Developing detection algorithms with few false positives, false negatives, and 100% accuracy is still complex and requires more research. Advances in communication technology have made the attack detection task complicated because of the differences in syntax implementations of underlying platforms deployed by multiple third-party organizations (Abaimov & Bianchi 2019). For example, MySQL differs greatly in the implementation from Oracle, MSSQL, Postgre SQL, etc. The majority of the XSS attack detection models are

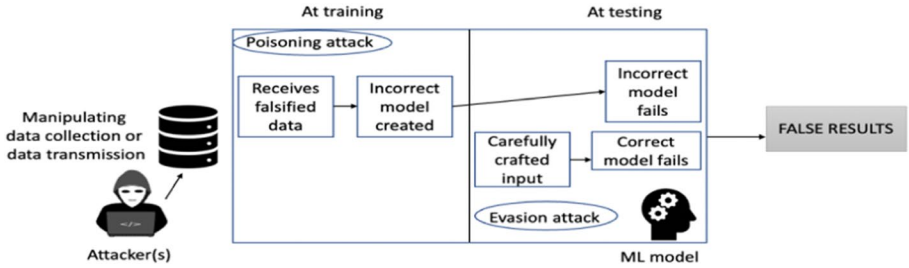

Fig. 18 Adversarial attack on Machine Learning-based system

weak at defending against zero-day attacks. These challenges open the door for innumerable research opportunities for young researchers. On the one hand, artificial intelligence, machine learning, and Deep Learning have gained the attention of researchers in resolving cybersecurity-related threats and risks. On the other hand, Deep Learning suffers from the semantic gap, training of the neural network, complex result evaluation strategies, and high cost of misclassification. For RQ5, we have observed that Deep Learning models aimed especially at cyber security applications must not focus on a single attack vector and should consider multiple attack vectors. Most ML and DL algorithms focus solely on TPR and FPR for performance evaluation and ignore training speed, storage consumption, and suspension of data poisoning parameters (Dixit & Silakari 2021). In addition, the usage of genetic algorithms and fuzzy algorithms can be better utilized to improve accuracy. Finally, metaheuristics algorithms can improve the efficiency of Deep Learning-based XSS attack detection techniques.

8 Conclusion and future directions

Cyber security threats have consistently risen over the past few decades, and several organizations are at potential risk. Every year cyber minds come up with sophisticated cyber-attacks that are challenging for traditional attack detection tools. Organizations are facing many challenges and losses due to XSS attacks on their web applications. There is an urgent need for a survey that can cover the limitations of traditional attack detection techniques and the scope of improvements required in these techniques to tackle novel cyber-attacks. Our survey provides the direction to researchers and security analysts with a detailed description of sophisticated vulnerability analysis techniques as compared to traditional attack detection techniques. Machine learning techniques provide better accuracy in detecting XSS attacks as compared to state-of-the-art approaches. In this survey, supervised, unsupervised and semi-supervised techniques are discussed along with advantages and limitations. We have highlighted five Research Questions for which we have tried to find answers from recent research works. After doing this survey, all RQs are answered which will be beneficial for researchers.

After doing this survey, it can be concluded that the ML techniques have proved to scan subtler bugs and reduce false alarms. However, there are a few challenges that are observed while doing a survey of existing techniques. The applications must be tested for XSS attack vulnerability at regular intervals. Leaving web applications for years without any security assessment is one of the biggest mistakes that many developers and application owners do.

They must have sound knowledge of security risks related to XSS. Moreover, most existing approaches do not consider new XSS threat vectors and payloads that must be identified through an appropriate machine-learning model. It should be noted that XSS attack detection models can also be compromised by hackers. Therefore, it is significant to retrain the attack detection model further through effective machine learning models and training. To obtain high accuracy, a feature subset selection mechanism must be implemented to extract high-profile features from overall generated features that will significantly impact the detection model's performance. To eliminate XSS attacks from their roots, Deep Learning techniques act as an alternative to inefficient traditional techniques of machine learning. Social networking sites are more affected by XSS attacks as the attack is easier to pass on to other users through a single infected victim via social connections. Deep Learning models such as LSTM, AutoEncoder, and MLP are applied by researchers to detect XSS attacks more effectively. In our survey, these techniques are covered for a better understanding of the reader. It is concluded that more sophisticated Deep Learning models should be proposed to achieve effective results.

References

2000 DARPA Intrusion Detection Scenario Specific Datasets | MIT Lincoln Laboratory. (n.d.). https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scenario-specific-datasets. Accessed 2 Nov 2021

Abaimov S, Bianchi G (2019) CODDLE: Code-injection detection with deep learning. IEEE Access 7:128617–128627. https://doi.org/10.1109/ACCESS.2019.2939870

Abdullah Alqarni A, Alsharif N, Ahmad Khan N, Georgieva L, Pardade E, Alzahrani YM (2022) MNN-XSS: modular neural network based approach for XSS attack detection. Comput Mater Continua 70(2):4075–4085. https://doi.org/10.32604/CMC.2022.020389

Akay B, Karaboga D, Akay R (2021) A comprehensive survey on optimizing deep learning models by metaheuristics. Artif Intell Rev 55(2):829–894. https://doi.org/10.1007/S10462-021-09992-0

Akrout R, Alata E, Kaaniche M, Nicomette V (2014) An automated black box approach for web vulnerability identification and attack scenario generation. J Brazilian Comput Soc 20(1):1–16. https://doi.org/10.1186/1678-4804-20-4

Alam F, Pachauri S (2017) Comparative study of J48, naive bayes and one-R classification technique for credit card fraud detection using WEKA. Adv Comput Sci Technol 10(6):1731–1743

Aldahdooh A, Hamidouche W, Fezza SA, Déforges O (2022) Adversarial example detection for DNN models: a review and experimental comparison. Artif Intell Rev 2022:1–60. https://doi.org/10.1007/S10462-021-10125-W

Alexa - Top sites. (n.d.). https://www.alexa.com/topsites. Accessed 2 Nov 2021

Banerjee, R., Baksi, A., Singh, N., & Bishnu, S. K. (2020, October 2). Detection of XSS in web applications using Machine Learning Classifiers. *2020 4th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*. https://doi.org/10.1109/IEMENTech51367.2020.9270052

Brewer R (2016) Ransomware attacks: detection, prevention and cure. Netw Secur 2016(9):5–9. https://doi.org/10.1016/S1353-4858(16)30086-1

Chauhan S, Vig L, Filippo De Grazia M, Corbetta M, Ahmad S, Zorzi M (2019) A Comparison of shallow and deep learning methods for predicting cognitive performance of stroke patients from mri lesion images. Front Neuroinform. https://doi.org/10.3389/fninf.2019.00053

Chen T, Liu J, Xiang Y, Niu W, Tong E, Han Z (2019) Adversarial attack and defense in reinforcement learning-from AI security view. Cybersecurity 2:1. https://doi.org/10.1186/s42400-019-0027-x

Cimpanu Catalin. (2018). *British Airways breach caused by the same group that hit Ticketmaster | ZDNet*. ZDNET, A RED VENTURES COMPANY. https://www.zdnet.com/article/british-airways-breach-caused-by-the-same-group-that-hit-ticketmaster/

CISCO. (2021). *Defending Against Critical Threats*. https://www.cisco.com/c/en/us/products/security/defending-against-critical-threats.html?CCID=cc000160&DTID=odicdc000016&OID=rptsc024689

Conti, M., Dargahi, T., & Dehghantanha, A. (2018). Cyber threat intelligence: Challenges and opportunities. In *Advances in Information Security* (Vol. 70, pp. 1–6). Springer New York LLC. https://doi.org/10.1007/978-3-319-73951-9_1

Dada, E. G. (2017). A Hybridized SVM-kNN-pdAPSO Approach to Intrusion Detection System. In *University of Maiduguri Faculty of Engineering Seminar Series* (Vol. 8). https://www.researchgate.net/publication/316145216_A_Hybridized_SVM-kNN-pdAPSO_Approach_to_Intrusion_Detection_System

Zhu Dali, Jin Hao, Ying Yang Wu, D., & Weiyi Chen. (2017) DeepFlow: deep learning-based malware detection by mining android application for abnormal usage of sensitive data. 2017 IEEE Symposium Comput Commun (ISCC). https://doi.org/10.1109/ISCC.2017.8024568

Dixit P, Silakari S (2021) Deep learning algorithms for cybersecurity applications: a technological and status review. Comput Sci Rev 39:100317. https://doi.org/10.1016/J.COSREV.2020.100317

Dora JR, Nemoga K (2021) Ontology for cross-site-scripting (XSS) attack in cybersecurity. J Cybersecur Privacy 1(2):319–339. https://doi.org/10.3390/jcp1020018

EOIN KEARY. (n.d.). *2019 VULNERABILITY STATISTICS REPORT*. https://www.edgescan.com/wp-content/uploads/2019/02/edgescan-Vulnerability-Stats-Report-2019.pdf. Accessed 2 Sep 2021

Fang Y, Huang C, Xu Y, Li Y (2019) RLXSS: Optimizing XSS detection model to defend against adversarial attacks based on reinforcement learning. Future Internet 11:8. https://doi.org/10.3390/FI11080177

Fang, Y., Li, Y., Liu, L., & Huang, C. (2018). DeepXSS: Cross site scripting detection based on deep learning. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3194452.3194469

Feng F, Liu X, Yong B, Zhou R, Zhou Q (2019) Anomaly detection in ad-hoc networks based on deep learning model: a plug and play device. Ad Hoc Netw 84:82–89. https://doi.org/10.1016/J.ADHOC.2018.09.014

Furnell S, Emm D (2017) The ABC of ransomware protection. Comput Fraud Secur 2017(10):5–11. https://doi.org/10.1016/S1361-3723(17)30089-1

Gao W, Morris T, Reaves B, Richey D (2010) On SCADA control system command and response injection and intrusion detection. General Mem Meet ECrime Res Summit ECrime. https://doi.org/10.1109/ECRIME.2010.5706699

Guo Y, Pan Y, Zhang Z, Li L, Jamshed MA, Moon Y, Kim D, Han D, Park K, Jamshed M A, Berger DS, Sitaraman RK, Harchol-Balter M, Pfaff B, Pettit J, Koponen T, Jackson E, Zhou A, Rajahalme J, … Security I. T (2017) Same-origin policy: Evaluation in modern browsers. In Proceedings of the Same-Origin Policy: Evaluation in Modern Browsers. Nsdi, 40(4): 97–112

Geetha R, Thilagam T (2020) A review on the effectiveness of machine learning and deep learning algorithms for cyber security. Arch Comput Methods Eng 28(4):2861–2879. https://doi.org/10.1007/S11831-020-09478-2

GitHub - Morzeux/HttpParamsDataset. (n.d.). https://github.com/Morzeux/HttpParamsDataset. Accessed 2 Nov 2021

Gkioulos V, Chowdhury N (2021) Cyber security training for critical infrastructure protection: a literature review. Comput Sci Rev 40:100361. https://doi.org/10.1016/J.COSREV.2021.100361

Gupta B.B., & Chaudhary Pooja. (2020). *Cross-Site Scripting Attacks: Classification, Attack, and Countermeasures.* (First). CRC Press. https://www.google.co.in/books/edition/Cross_Site_Scripting_Attacks/697SDwAAQBAJ?hl=en&gbpv=0&kptab=overview

Gupta S, Gupta BB (2015) Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. Int J Syst Assurance Eng Manag 8(1):512–530. https://doi.org/10.1007/S13198-015-0376-0

Hassan MdM, Nipa SS, Akter M, Haque R, Deepa FN, Rahman MM, Siddiqui Md, Sharif MdH (2018) Broken authentication and session management vulnerability: a case study of web application. Int J Simul. https://doi.org/10.5013/ijssst.a.19.02.06

Heiderich, M., Schwenk, J., Frosch, T., Magazinius, J., & Yang, E. Z. (2013). mXSS attacks: Attacking well-secured web-applications by using innerHTML mutations. *Proceedings of the ACM Conference on Computer and Communications Security*, 777–788. https://doi.org/10.1145/2508859.2516723

Hickling J (2021) What is DOM XSS and why should you care? Comput Fraud Secur 2021(4):6–10. https://doi.org/10.1016/S1361-3723(21)00040-3

Hoang XD (2020) Detecting common web attacks based on machine learning using web log. Lecture Notes Networks and Syst 178:311–318. https://doi.org/10.1007/978-3-030-64719-3_35

Introducing a powerful open source social networking engine. (n.d.). https://elgg.org/. Accessed 2 Nov 2021

Jagajeevan Rao L, Nazeer Basha SK, Rama Krishna V (2021) Prevention and analysing on cross site scripting. Adv Intell Syst Comput 1171:731–739. https://doi.org/10.1007/978-981-15-5400-1_69

Jiang F, Fu Y, Gupta BB, Liang Y, Rho S, Lou F, Meng F, Tian Z (2020) Deep learning based multi-channel intelligent attack detection for data security. IEEE Trans Sustain Comput 5(2):204–212. https://doi.org/10.1109/TSUSC.2018.2793284

Jian-hua Li. (2021). Cyber Security Meets Machine Learning. In *Cyber Security Meets Machine Learning*. Springer Singapore. https://doi.org/10.1007/978-981-33-6726-5

Kascheev, S., & Olenchikova, T. (2020). Detecting Cross-Site Scripting (XSS) Using Machine Learning Methods. *2020 Global Smart Industry Conference (GloSIC)*. https://doi.org/10.1109/GloSIC50886.2020.9267866

Katsikeas S, Johnson P, Ekstedt M, Lagerström R (2021) Research communities in cyber security: a comprehensive literature review. Comput Sci Rev. https://doi.org/10.1016/j.cosrev.2021.100431

Kaur, G., Malik, Y., Samuel, H., & Jaafar, F. (2018). Detecting blind cross-site scripting attacks using machine learning. *ACM International Conference Proceeding Series*, 22–25. https://doi.org/10.1145/3297067.3297096

Kaur, J., & Garg, U. (2022). State-of-the-Art Survey on Web Vulnerabilities, Threat Vectors, and Countermeasures. In: Dr. R. Aggarwal, Dr. J. He, Dr. E. Shubhakar Pilli, & Dr. S. Kumar (Eds) *Cyber Security in Intelligent Computing and Communications*. Springer, Singapore. (pp. 3–17).

Kaur S, Singh M (2019) Hybrid intrusion detection and signature generation using deep recurrent neural networks. Neural Comput Appl 32(12):7859–7877. https://doi.org/10.1007/S00521-019-04187-9

KDD Cup 1999 Data. (n.d.). http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed 2 Nov 2021

Keary Eoin. (n.d.). 2021 VULNERABILITY STATISTICS REPORT EDGESCAN. In *EdgeScan Report*. https://info.edgescan.com/hubfs/Edgescan2021StatsReport.pdf?hsCtaTracking=9601b027-23d3-443f-b438-fcb671cfda06%7Cb222011c-0b6d-440b-aed8-64d37dec66e2. Accessed 2 Sep 2021

Khan N, Abdullah J, Khan AS (2017) Defending malicious script attacks using machine learning classifiers. Wireless Commun Mobile Comput. https://doi.org/10.1155/2017/5360472

Kharraz A, Robertson W, Balzarotti D, Bilge L, Kirda E (2015) Cutting the Gordian knot: a look under the hood of ransomware attacks. Lecture Notes Comput Sci 9148:3–24. https://doi.org/10.1007/978-3-319-20550-2_1

Kokila, R. T., Thamarai Selvi, S., & Govindarajan, K. (2015). DDoS detection and analysis in SDN-based environment using support vector machine classifier. *6th International Conference on Advanced Computing, ICoAC 2014*, 205–210. https://doi.org/10.1109/ICOAC.2014.7229711

Kumar R, Goyal R (2019) On cloud security requirements, threats, vulnerabilities and countermeasures: a survey. Comput Sci Rev 33:1–48. https://doi.org/10.1016/J.COSREV.2019.05.002

Laghrissi FE, Douzi S, Douzi K, Hssina B (2021) Intrusion detection systems using long short-term memory (LSTM). J Big Data 8:1. https://doi.org/10.1186/s40537-021-00448-4

Lei, L., Chen, M., He, C., & Li, D. (2020). XSS Detection Technology Based on LSTM-Attention. *2020 5th International Conference on Control, Robotics and Cybernetics, CRC 2020*, 175–180. https://doi.org/10.1109/CRC51253.2020.9253484

Leyden, J. (2020). *XSS vulnerability in 'Login with Facebook' button earns $20,000 bug bounty*. PortSwigger. https://portswigger.net/daily-swig/xss-vulnerability-in-login-with-facebook-button-earns-20-000-bug-bounty

Li J, Hua (2018) Cyber security meets artificial intelligence: a survey. Front Inform Technol Electron Eng 19(12):1462–1474. https://doi.org/10.1631/FITEE.1800573

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). *A Critical Review of Recurrent Neural Networks for Sequence Learning*. http://arxiv.org/abs/1506.00019

Luo C, Tan Z, Min G, Gan J, Shi W, Tian Z (2021) A novel web attack detection system for internet of things via ensemble classification. IEEE Trans Industr Inf 17(8):5810–5818. https://doi.org/10.1109/TII.2020.3038761

Matt, F. (2021). Application Model & Same-Origin Policy. In *Lecture Notes on Web Security: Application Model & Same-Origin Policy*.

Melicher, W., Fung, C., Bauer, L., & Jia, L. (2021). Towards a lightweight, hybrid approach for detecting DOM XSS vulnerabilities with machine learning. *The Web Conference 2021—Proceedings of the World Wide Web Conference, WWW 2021*, 2684–2695. https://doi.org/10.1145/3442381.3450062

C.C. Michael, & Steven Lavenhar. (2013). *Source Code Analysis Tools - Overview | CISA*. Cybersecurity & Infrastructure Security Agency. https://us-cert.cisa.gov/bsi/articles/tools/source-code-analysis/source-code-analysis-tools---overview

Microsoft Teams security vulnerability left users open to XSS via flawed stickers feature | The Daily Swig. (n.d.). https://portswigger.net/daily-swig/microsoft-teams-security-vulnerability-left-users-open-to-xss-via-flawed-stickers-feature. Accessed 10 Oct 2022

Mnih, V., Heess, N., Graves, A., & Kavukcuoglu, K. (2014). Recurrent Models of Visual Attention. *Advances in Neural Information Processing Systems*, *3*(January), 2204–2212. https://arxiv.org/abs/1406.6247v1

Mohammad G, Reza S (2017) Software vulnerability analysis and discovery using machine-learning and data-mining techniques. ACM Comput Surv (CSUR) 50:4. https://doi.org/10.1145/3092566

Mokbal FMM, Dan W, Imran A, Jiuchuan L, Akhtar F, Xiaoxi W (2019) MLPXSS: an Integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique. IEEE Access 7:100567–100580. https://doi.org/10.1109/ACCESS.2019.2927417

Moon D, Im H, Kim I, Park JH (2015) DTB-IDS: an intrusion detection system based on decision tree using behavior analysis for preventing APT attacks. J Supercomput 73(7):2881–2895. https://doi.org/10.1007/S11227-015-1604-8

Munonye K, Péter M (2021) Machine learning approach to vulnerability detection in OAuth 2.0 authentication and authorization flow. Int J Inf Secur 2021:1–15. https://doi.org/10.1007/S10207-021-00551-W

Nidecki Tomasz Andrzej. (2019). *Mutation XSS in Google Search*. THE ACUNETIX BLOG. https://www.acunetix.com/blog/web-security-zone/mutation-xss-in-google-search/

G. Nick. (2021). *The Most Telling Cyber Security Statistics in 2021 [Infographic]*. Cyber Security Stats—Infographic. https://techjury.net/blog/cyber-security-statistics/

Novinson Michael. (2021). *The 10 Biggest Data Breaches Of 2021 (So Far)*. CRN News. https://www.crn.com/slide-shows/security/the-10-biggest-data-breaches-of-2021-so-far-/2

Olalere, M., Abdullah, M. T., Mahmod, R., & Abdullah, A. (2016). Identification and Evaluation of Discriminative Lexical Features of Malware URL for Real-Time Classification. *2016 International Conference on Computer and Communication Engineering (ICCCE)*. https://doi.org/10.1109/ICCCE.2016.31

Onan A (2018) Biomedical text categorization based on ensemble pruning and optimized topic modelling. Comput Math Methods Med. https://doi.org/10.1155/2018/2497471

Onan A (2018b) An ensemble scheme based on language function analysis and feature engineering for text genre classification. J Inf Sci 44(1):28–47. https://doi.org/10.1177/0165551516677911

Onan A (2019) Consensus clustering-based undersampling approach to imbalanced learning. Sci Program. https://doi.org/10.1155/2019/5901087

Onan A (2019b) Topic-enriched word embeddings for sarcasm identification. Adv Intell Syst Comput 984:293–304. https://doi.org/10.1007/978-3-030-19807-7_29/COVER

Onan A (2019c) Two-stage topic extraction model for bibliometric data analysis based on word embeddings and clustering. IEEE Access 7:145614–145633. https://doi.org/10.1109/ACCESS.2019.2945911

Onan A (2020) Mining opinions from instructor evaluation reviews: a deep learning approach. Comput Appl Eng Educ 28(1):117–138. https://doi.org/10.1002/CAE.22179

Onan A (2021a) Sentiment analysis on massive open online course evaluations: a text mining and deep learning approach. Comput Appl Eng Educ 29(3):572–589. https://doi.org/10.1002/CAE.22253

Onan A (2021) Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks. Concurrency Comput 33(23):e5909. https://doi.org/10.1002/CPE.5909

Onan A (2022) Bidirectional convolutional recurrent neural network architecture with group-wise enhancement mechanism for text sentiment classification. J King Saud University—Comput Information Sci 34(5):2098–2117. https://doi.org/10.1016/J.JKSUCI.2022.02.025

Onan A, KorukoGlu S (2017) A feature selection model based on genetic rank aggregation for text sentiment classification. J Inf Sci 43(1):25–38. https://doi.org/10.1177/0165551515613226

Onan A, Korukoğlu S, Bulut H (2016) Ensemble of keyword extraction methods and classifiers in text classification. Expert Syst Appl 57:232–247. https://doi.org/10.1016/J.ESWA.2016.03.045

Onan A, Korukoğlu S, Bulut H (2017) A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification. Inform Processing Manag 53(4):814–833. https://doi.org/10.1016/J.IPM.2017.02.008

Onan A, Tocoglu MA (2021) A term weighted neural language model and stacked bidirectional LSTM Based framework for sarcasm identification. IEEE Access 9:7701–7722. https://doi.org/10.1109/ACCESS.2021.3049734

owasp. (2017). *OWASP Top Ten*. OWASP. https://owasp.org/

Pan Y, Sun F, Teng Z, White J, Schmidt DC, Staples J, Krause L (2019) Detecting web attacks with end-to-end deep learning. J Internet Serv Appl 10(1):1–22. https://doi.org/10.1186/S13174-019-0115-X

Pavan Kumar P, Jaya T, Rajendran V (2021) SI-BBA—a novel phishing website detection based on swarm intelligence with deep learning. Mater Today. https://doi.org/10.1016/J.MATPR.2021.07.178

Pitropakis N, Panaousis E, Giannetsos T, Anastasiadis E, Loukas G (2019) A taxonomy and survey of attacks against machine learning. Comput Sci Rev 34:100199. https://doi.org/10.1016/J.COSREV.2019.100199

Praise JJ, Raj RJS, Benifa JVB (2020) Development of reinforcement learning and pattern matching (RLPM) based firewall for secured cloud infrastructure. Wireless Personal Commun 115(2):993–1018. https://doi.org/10.1007/S11277-020-07608-4

Rathore S, Sharma PK, Park JH (2017) XSSClassifier: an efficient XSS attack detection approach based on machine learning classifier on SNSs. J Inform Processing Syst 13(4):1014–1028. https://doi.org/10.3745/JIPS.03.0079

Rodríguez GE, Torres JG, Flores P, Benavides DE (2020) Cross-site scripting (XSS) attacks and mitigation: a survey. Computer Networks 166:106960. https://doi.org/10.1016/J.COMNET.2019.106960

Sarmah U, Bhattacharyya DK, Kalita JK (2018) A survey of detection methods for XSS attacks. J Netw Comput Appl 118:113–143. https://doi.org/10.1016/J.JNCA.2018.06.004

Screencastify Chrome extension flaws allow webcam hijacks. (n.d.). https://www.bleepingcomputer.com/news/security/screencastify-chrome-extension-flaws-allow-webcam-hijacks/. Accessed 10 Oct 2022

Shabut, A. M., Lwin, K. T., & Hossain, M. A. (2016). Cyber attacks, countermeasures, and protection schemes—a state of the art survey. *2016 10th International Conference on Software, Knowledge, Information Management & Applications (SKIMA)*. https://doi.org/10.1109/SKIMA.2016.7916194

Shahid N, Aleem SA, Naqvi IH, Zaffar N (2012) Support vector machine based fault detection & classification in smart grids. *2012 IEEE Globecom Workshops*. GC Wkshps 2012:1526–1531. https://doi.org/10.1109/GLOCOMW.2012.6477812

Machine Learning based Intrusion Detection System for Web-Based Attacks, Proceedings - 2020 IEEE 6th Intl Conference on Big Data Security on Cloud, BigDataSecurity 2020, 2020 IEEE Intl Conference on High Performance and Smart Computing, HPSC 2020 and 2020 IEEE Intl Conference on Intelligent Data and Security, IDS 2020 227 (2020).

Shukla A, Tiwari R, Kala R (2010) Modular neural networks. Stud Comput Intell 307:307–335. https://doi.org/10.1007/978-3-642-14344-1_14

Snehi M, Bhandari A (2021) Vulnerability retrospection of security solutions for software-defined cyber-physical system against DDoS and IoT-DDoS attacks. Comput Sci Rev. https://doi.org/10.1016/j.cosrev.2021.100371

Syarif, A. R., & Gata, W. (2018). Intrusion detection system using hybrid binary PSO and K-nearest neighborhood algorithm. *Proceedings of the 11th International Conference on Information and Communication Technology and System, ICTS 2017*, *2018-January*, 181–186. https://doi.org/10.1109/ICTS.2017.8265667

Tariq I, Sindhu MA, Abbasi RA, Khattak AS, Maqbool O, Siddiqui GF (2021) Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning. Exp Syst Appl 168:114386. https://doi.org/10.1016/J.ESWA.2020.114386

Tekli G (2021) A survey on semi-structured web data manipulations by non-expert users. Comput Sci Rev 40:100367. https://doi.org/10.1016/J.COSREV.2021.100367

Thakkar A, Lohiya R (2021) A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. Artificial Intell Rev 55(1):453–563. https://doi.org/10.1007/S10462-021-10037-9

Tian Z, Luo C, Qiu J, Du X, Guizani M (2020) A distributed deep learning system for web attack detection on edge devices. IEEE Trans Industr Inf 16(3):1963–1971. https://doi.org/10.1109/TII.2019.2938778

Tran NK, Sheng QZ, Babar MA, Yao L (2017) Searching the web of things: state of the art, challenges, and solutions. ACM Comput Surv 50(4):1–34. https://doi.org/10.1145/3092695

Vinayakumar R, Soman KP, Poornachandran P, Sachin Kumar S (2018) Detecting android malware using long short-term memory (LSTM). J Intell Fuzzy Syst 34:3. https://doi.org/10.3233/JIFS-169424

Vollmer, T., & Manic, M. (2009). Computationally efficient neural network intrusion security awareness. *Proceedings - ISRCS 2009—2nd International Symposium on Resilient Control Systems*, 25–30. https://doi.org/10.1109/ISRCS.2009.5251357

von Solms R, van Niekerk J (2013) From information security to cyber security. Comput Secur 38:97–102. https://doi.org/10.1016/j.cose.2013.04.004

Vuong, T. P., Loukas, G., Gan, D., & Bezemskij, A. (2015). Decision tree-based detection of denial of service and command injection attacks on robotic vehicles. *2015 IEEE International Workshop on Information Forensics and Security, WIFS 2015 - Proceedings*. https://doi.org/10.1109/WIFS.2015.7368559

Wang Q, Yang H, Wu G, Choo KKR, Zhang Z, Miao G, Ren Y (2022) Black-box adversarial attacks on XSS attack detection model. Comput Secur 113:102554. https://doi.org/10.1016/J.COSE.2021.102554

Wang, R., Jia, X., Li, Q., & Zhang, D. (2015). Improved N-gram approach for cross-site scripting detection in Online Social Network. *Proceedings of the 2015 Science and Information Conference, SAI 2015*, 1206–1212. https://doi.org/10.1109/SAI.2015.7237298

Wang, R., Jia, X., Li, Q., & Zhang, S. (2014). Machine Learning Based Cross-Site Scripting Detection in Online Social Network. *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS)*, 823–826. https://doi.org/10.1109/HPCC.2014.137

Wang Z, Fok KW, Thing VLL (2022) Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. Comput Secur 113:102542. https://doi.org/10.1016/J.COSE.2021.102542

Wikipedia. (2021). *Session ID*. Online. https://en.wikipedia.org/wiki/Session_ID

WordPress Stored XSS Vulnerability—Update Now. (n.d.). https://www.searchenginejournal.com/wordpress-core-vulnerability-2022/441795/#close. Accessed 10 Oct 2022

XSS in Gmail's AMP For Email earns researcher $5,000 | The Daily Swig. (n.d.). https://portswigger.net/daily-swig/xss-in-gmails-amp-for-email-earns-researcher-5-000. Accessed 8 Oct 2022

XSS vulnerabilities in Google Cloud, Google Play could lead to account hijacks | The Daily Swig. (n.d.). https://portswigger.net/daily-swig/xss-vulnerabilities-in-google-cloud-google-play-could-lead-to-account-hijacks. Accessed 8 Oct 2022

XSSed | Cross Site Scripting (XSS) attacks information and archive. (n.d.). http://xssed.com/. Accessed 2 Nov 2021

Yan R, Xiao X, Hu G, Peng S, Jiang Y (2018) New deep learning method to detect code injection attacks on hybrid applications. J Syst Softw 137:67–77. https://doi.org/10.1016/J.JSS.2017.11.001

Yu Y, Si X, Hu C, Zhang J (2019) A review of recurrent neural networks: Lstm cells and network architectures. Neural Comput 31(7):1235–1270. https://doi.org/10.1162/NECO_A_01199

Zhang G, Liu B, Zhu T, Zhou A, Zhou W (2022) Visual privacy attacks and defenses in deep learning: a survey. Artif Intell Rev 2021:1–55. https://doi.org/10.1007/S10462-021-10123-Y

Zhang X, Zhou Y, Pei S, Zhuge J, Chen J (2020) Adversarial examples detection for XSS attacks based on generative adversarial networks. IEEE Access 8:10989–10996. https://doi.org/10.1109/ACCESS.2020.2965184

Zhang Z, Ning H, Shi F, Farha F, Xu Y, Xu J, Zhang F, Choo KKR (2021a) Artificial intelligence in cyber security: research advances, challenges, and opportunities. Artif Intell Rev. https://doi.org/10.1007/s10462-021-09976-0

Zhang Z, Ning H, Shi F, Farha F, Xu Y, Xu J, Zhang F, Choo KKR (2021) Artificial intelligence in cyber security: research advances, challenges, and opportunities. Artif Intell Rev 55(2):1029–1053. https://doi.org/10.1007/S10462-021-09976-0

Zhao, G., Zhang, C., & Zheng, L. (2017, July). Intrusion Detection Using Deep Belief Network and Probabilistic Neural Network. *22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. https://doi.org/10.1109/CSE-EUC.2017.119

Zhou Y, Wang P (2019) An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence. Comput Secur 82:261–269. https://doi.org/10.1016/J.COSE.2018.12.016

Authors and Affiliations

Jasleen Kaur[1] · Urvashi Garg[1] · Gourav Bathla[2]

Jasleen Kaur
jasleenkaur2136@gmail.com

Urvashi Garg
urvashi.mittal80@gmail.com

[1] Chandigarh University, Punjab, India

[2] University of Petroleum and Energy Studies, Dehradun, India