future internet

MDPI

Systematic Review

Deep Learning for Vulnerability and Attack Detection on Web Applications: A Systematic Literature Review

Rokia Lamrani Alaoui * and El Habib Nfaoui

LISAC Laboratory, Computer Science Department, Faculty of Sciences Dhar EL Mahraz (F.S.D.M.), Sidi Mohamed Ben Abdellah University, Fez 30000, Morocco; elhabib.nfaoui@usmba.ac.ma
* Correspondence: rokia.lamranialaoui@usmba.ac.ma

Abstract: Web applications are the best Internet-based solution to provide online web services, but they also bring serious security challenges. Thus, enhancing web applications security against hacking attempts is of paramount importance. Traditional Web Application Firewalls based on manual rules and traditional Machine Learning need a lot of domain expertise and human intervention and have limited detection results faced with the increasing number of unknown web attacks. To this end, more research work has recently been devoted to employing Deep Learning (DL) approaches for web attacks detection. We performed a Systematic Literature Review (SLR) and quality analysis of 63 Primary Studies (PS) on DL-based web applications security published between 2010 and September 2021. We investigated the PS from different perspectives and synthesized the results of the analyses. To the best of our knowledge, this study is the first of its kind on SLR in this field. The key findings of our study include the following. (i) It is fundamental to generate standard real-world web attacks datasets to encourage effective contribution in this field and to reduce the gap between research and industry. (ii) It is interesting to explore some advanced DL models, such as Generative Adversarial Networks and variants of Encoders–Decoders, in the context of web attacks detection as they have been successful in similar domains such as networks intrusion detection. (iii) It is fundamental to bridge expertise in web applications security and expertise in Machine Learning to build theoretical Machine Learning models tailored for web attacks detection. (iv) It is important to create a corpus for web attacks detection in order to take full advantage of text mining in DL-based web attacks detection models construction. (v) It is essential to define a common framework for developing and comparing DL-based web attacks detection models. This SLR is intended to improve research work in the domain of DL-based web attacks detection, as it covers a significant number of research papers and identifies the key points that need to be addressed in this research field. Such a contribution is helpful as it allows researchers to compare existing approaches and to exploit the proposed future work opportunities.

Keywords: deep learning; systematic literature review; web applications security; web attacks detection; web vulnerabilities

Citation: Alaoui, R.L.; Nfaoui, E.H. Deep Learning for Vulnerability and Attack Detection on Web Applications: A Systematic Literature Review. *Future Internet* **2022**, *14*, 118. https://doi.org/10.3390/fi14040118

Academic Editors: Miltiadis D. Lytras and Andreea Claudia Serban

Received: 21 February 2022
Accepted: 24 March 2022
Published: 13 April 2022

1. Introduction and Background

Due to the extensive use of websites and web applications, web vulnerabilities are continuously growing. A survey conducted in 2019 [1] found that nine of 10 web applications are vulnerable and that sensitive data breaches are possible on 68% of web applications. Furthermore, that network intrusion was caused by unauthorized access to web servers in 8% of cases. The immeasurable and disparate use of the Internet makes it a hackers' aim. The main goal of web vulnerabilities detection techniques is to protect websites and web applications from cyber-attacks such as Cross-Site Scripting (XSS), SQL injection, etc. The subsections below present the necessary background for understanding the remainder of this paper, including the general architecture of web-based applications, types of web vulnerabilities, and different web vulnerabilities prevention and detection methods.

1.1. Web Applications Architecture

Web-based applications are the essential network-based solution to offer standard web services. The development of these applications is based on client and server-side development. The server side involves a web server, a web application, and a database server; it utilizes backend scripting languages including .NET, PHP, and JEE (Jakarta Enterprise Edition). The client-side works on the user's web browser with front-end scripting languages, including CSS/HTML, Javascript, etc. These two are usually interconnected via HTTP protocol. Figure 1 presents server-side and client-side web application architecture. Web applications became an integral part of individuals' daily life because of their accessibility and convenience. However, this increased popularity is a double-edged sword. Indeed, web-based applications are the main highway for attackers to jeopardize critical services in vital sectors such as healthcare, education, banking, and e-commerce.

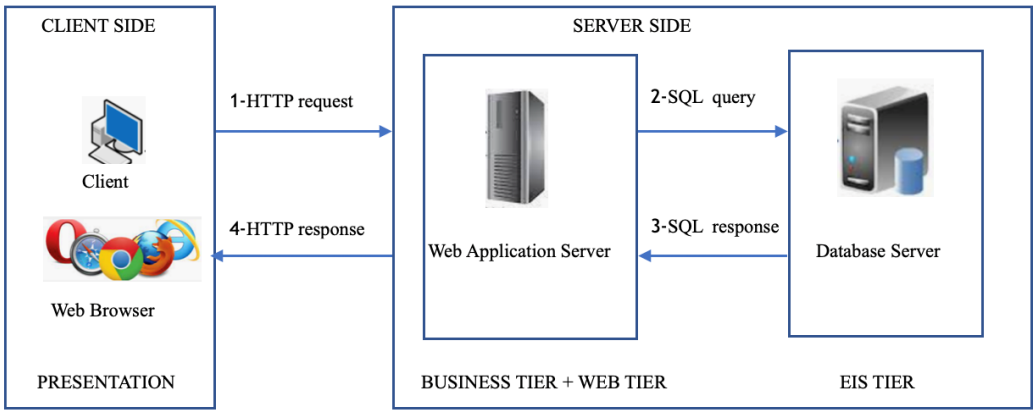
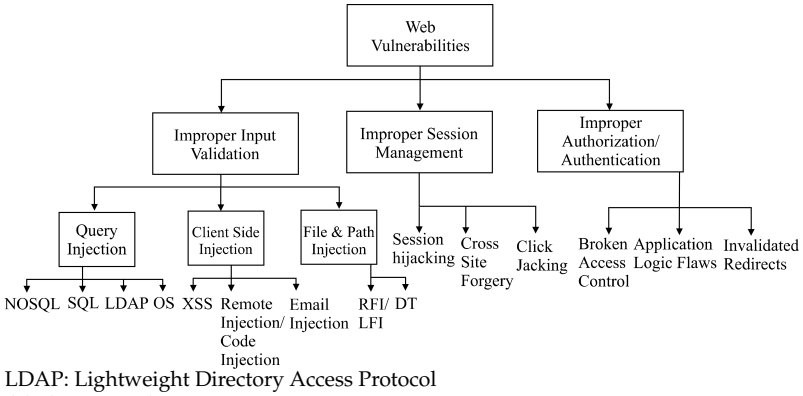


Figure 1. Overview of web architecture.

1.2. Web Vulnerabilities

The term vulnerability is a weakness, glitch, or loophole in web application development. An exploit is when the vulnerability is exploited, and the attack succeeds. Based on the research work [2], web application vulnerabilities can be classified into three categories (Figure 2): *Improper input validation* refers to an incorrect validation and sanitization of user input. SQL injection and Cross-Site Scripting (XSS) are examples of web attacks caused by improper input validation vulnerability. Second, *Improper session management* refers to when the web session is not secured correctly, and thus, the application could not identify if web requests are malicious until these are linked with a proper valid session identifier. Cross-Site Request Forgery (CSRF) and session highjacking are examples of web attacks caused by improper session management. Finally, *Improper authorization and authentication* vulnerability implies a logic flaw in the exercise of access control policies as well as functions of authenticating. Broken access control is one of the web attacks that could happen if the web application does not correctly manage authentication and authorization procedures.



LDAP: Lightweight Directory Access Protocol
OS: Operating Systems
RFI: Remote File Inclusion
LFI: Local File Inclusion
DT: Directory Traversal

Figure 2. Types of web vulnerabilities [2].

1.3. Web Vulnerabilities Countermeasures

Numerous researchers have proposed different methodologies to counter web vulnerabilities. Figure 3 presents the main web applications security approaches. These methods can be used mutually at different stages of the web application development life-cycle, and they can either be implemented and placed at the client side or server side of web applications.

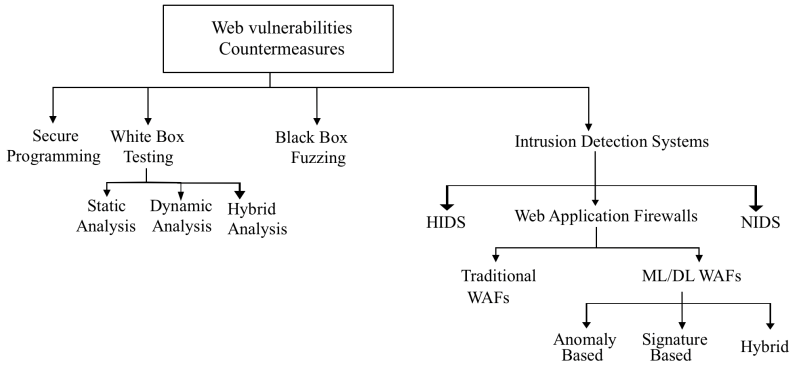


Figure 3. Web vulnerabilities countermeasures.

1.3.1. Secure Programming

It is a set of rules and good practices enabling the development of secure web applications. The secure coding standards include queries parametrization (i.e., query parameters are replaced with placeholders and parameter values are supplied at execution time), input validation (i.e., checks if the input meets a set of criteria such as a string contains no standalone single quotation marks), and sanitization of user input (i.e., modifies the input to ensure that it is valid). The OWASP project (Open Web Application Security Project) has proposed different standards to allow developers to follow secure practices when they are coding web applications (ASVS (Application Security Verification Standard) [3], ESAPI (Enterprise Security API), SAMM (OWASP Software Assurance Maturity Model) [4]). Although secure programming can help to prevent web vulnerabilities, it imparts time overhead and is not enough because of the complexity of web applications and the diversity of technologies and external libraries involved in the development of web applications.

1.3.2. Static Analysis

It aims at finding web vulnerabilities by inspecting source or binary code without running it. There exist several research works on web vulnerabilities detection using static

analysis. For instance, ref [5] proposed a contextual, inter-function, and data-flow analysis to discover taint-style vulnerabilities such as SQL, command injection, and XSS attacks. Ref. [6] combined source code taint-analysis (i.e., track user inputs to verify if they reach a sensitive sink (a function that can be exploited)) to find web vulnerabilities with data mining to reduce false positives. Ref. [7] described a static analysis method that automatically detects access control vulnerabilities in web applications. Ref. [8] proposed a Machine Learning-based static analysis method to discover web vulnerabilities. They precisely used Hidden Markov Model and annotated code slices to train a model to discover vulnerabilities in source code. Ref. [9] presented a methodology and tool based on symbolic code execution (i.e., instead of running the program with concrete inputs, symbolic execution runs them with symbolic ones and finds vulnerabilities along with the inputs that will trigger them) to identify vulnerabilities in web-based applications.

Overall, static analysis-based tools are a solution to find web vulnerabilities, but they tend to generate false positives, they are time-consuming, and they may never converge for large code bases.

1.3.3. Dynamic Analysis

It runs the application web and tries to identify security violations by using techniques such as code instrumentation (i.e., inserting checks into the program) and fuzzing (i.e., it inputs random test data to a target program in order to explore all possible paths).

Ref. [10] improved the detection of XSS attacks in web applications by using dynamic analysis and a fuzzy engine. They extracted Application Entry Points (AEP) using a web crawler and then used a fuzzy engine that generates invalid strings for each AEP until the Web Application Firewall is defeated, in which case its signatures database is updated. Ref. [11] presented a method for the detection of DOM-XSS attacks: they used dynamic taint analysis of JavaScript code (i.e., taint traces are obtained while parsing web pages), and fuzzing to automatically derive attack vectors based on those taint traces, and then, they verify DOM-XSS vulnerability by rendering HTTP responses on the browser. Ref. [12] used code instrumentation to generate models that describe how and with whom client-side components interact, which allows protecting JavaScript-based web applications against client-side validation attacks. Refs. [13,14] presented a concolic (concrete + symbolic) execution-based approach for the detection of XSS attacks, and SQL injection and OS command injection, respectively.

Dynamic analysis-based approaches incur no false positives but can not achieve high code coverage.

1.3.4. Black-Box Fuzzing

It sends random malicious data to a web application without regard to its logic and identifies whether the application is vulnerable based on its responses. Black-box fuzzers are generally composed of (i) a crawler which identifies all possible web pages and entry points in the analyzed web application, (ii) a test data generator that generates random data for each application entry point, and (iii) a monitor that detects errors in the application runtime behavior.

Ref. [15] proposed *KameleonFuzz*, a black-box XSS fuzzer for web applications that can generate malicious inputs to exploit XSS vulnerabilities and also to detect how close it is revealing a vulnerability. They used a genetic algorithm guided by an attack grammar for malicious inputs generation and evolution. Ref. [16] described a black-box fuzzing approach to detect XQuery injection and parameter tampering vulnerabilities in web applications driven by XML databases. The proposed approach takes place in two phases: (1) a training phase in which the application behavior is learned. It involves (i) a crawler that identifies injection points, (ii) a model constructor that constructs legitimate query models, and (iii) an HTML/JavaScript analyzer that extracts constraints on HTTP parameters. (2) The testing phase involves three components: (i) an attack generator that generates attack strings related to XQuery injection and parameter tampering vulnerabilities, (ii) a model

constructor that constructs illegitimate query models resulting from the execution of attack queries, and (iii) a detector that identifies the type of vulnerability exploited by comparing the query model generated during the testing phase against the appropriate query model generated during the training phase. Ref. [17] proposed a black-box fuzzing technique to detect logic vulnerabilities in web applications. They firstly collected HTTP traces in which users interact with a certain application. Then, they build a navigation graph model that synthesizes web resources (i.e., data types, URLs, forms, HTTP parameters, JSON objects, etc.). Afterward, they extracted behavioral patterns that model actions usually done by users and actions allowed by the navigation graph model. Then, they generated test cases that would break those behavioral patterns. Finally, they called a test oracle that collects from the executed test a partially ordered set of events and verifies whether all sequences satisfy the provided LTL (Linear Temporal Logic) formula. The test oracle returns true if a certain predefined logic property is violated and false otherwise. Likewise, ref. [18] proposed a method for the detection of XSS vulnerability in which they used a state automaton to obtain knowledge about the application behavior and a genetic algorithm to automatically generate inputs with better fitness values toward triggering an instance of the given vulnerability.

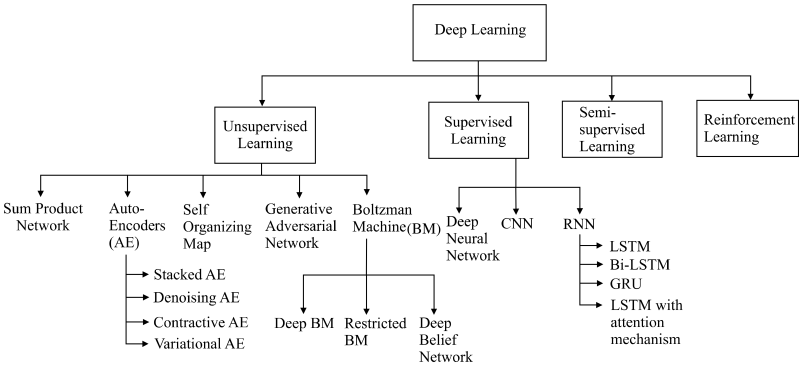
Web Vulnerability Scanning Tools can also be considered in the category of black-box fuzzers. They scan web applications from the outside to look for security vulnerabilities. They are frequently used by security analysts and hackers to find web application vulnerabilities. A large number of both commercial and open-source tools of this type are available (e.g., Burp Suite, Nessus, Nekto, etc.).

Black-box fuzzing does not require the availability of application source code, but it can incur a high rate of false negatives and false positives depending on the efficiency of the crawler and the attack data generator, respectively.

1.3.5. Intrusion Detection Systems (IDS)

They are defined as systems built to monitor host systems (Host Intrusion Detection System (HIDS)) or network communications (Network Intrusion Detection System (NIDS)) or web applications (Web Application Firewall (WAF)). HIDSs are usually employed for malware detection (i.e., malicious software that infects computers). NIDSs usually detect network attacks such as DoS (Denial of Service), MITM attacks (Man-In-the-Middle), as well as some types of web attacks. WAFs help protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. They detect client-side (e.g., DOM-XSS attack) and server-side (e.g., SQL injection attack) web attacks. We distinguish between traditional WAFs and WAFs based on Machine Learning (ML) ([19–21]) or Deep Learning algorithms (Figure 4 summarizes the main Deep Learning models that might be used for web attacks detection).

Traditional WAFs (e.g., ModSecurity) use static pattern rules matching to detect attacks. Thus, they can not detect new-unknown attacks. In addition, updating rules can be a tedious task if the attack pattern is complicated. However, they generate a few false positives. As for ML/DL-based WAFs, there exist three main approaches: (1) the anomaly-based approach in which models are trained using normal data instances uniquely and unsupervised or hybrid ML algorithms, (2) the signature-based approach (the most used) in which models are trained with normal and abnormal data instances and offline supervised ML algorithms, and (3) the hybrid approach, which combines the signature-based approach and anomaly-based detection approach. The first approach can detect zero-day attacks but suffers from a high rate of false positives because it is not obvious to define with certainty what is normality in a certain field. The second type of approach can not detect zero-day attacks but can detect known attacks accurately and with fewer false positives than the first approach. The third approach can make the best of both worlds. It can achieve high accuracy in detecting known attacks while generating low false positives in detecting new unknown attacks. Furthermore, the IDS signature database can be improved by adding the signature of newly detected attacks.



LSTM: Long Short-Term Memory
GRU: Gated Recurrent Network
RNN: Recurrent Neural Networks
CNN: Convolutional Neural Networks

Figure 4. Classification of Deep Learning techniques.

In this work, we are particularly interested in exploring existing studies on web attacks detection using Deep Learning in a systematic way. Thus, researchers and practitioners willing to use Deep Learning for web attacks detection will hopefully find valuable ground on which they can base to develop new and efficient DL-based web attacks detection models.

To the best of our knowledge, this study is the first of its kind on SLR in this field. We studied 63 DL-based web attacks detection papers published between 2010 and September 2021. Additionally, we categorized the papers according to several perspectives and came up with some interesting research opportunities. Our main contributions are summarized below:

- Identifying the Primary Studies (PS) related to the DL-based web attacks detection and getting different insights from the studies.
- Performing a quality analysis on the PS.
- Presenting the results of the investigation including publication information, datasets, detection models, detection performance, research focus, and limitations.
- Summarizing the findings and identifying some interesting opportunities for future work in the domain of DL-based web attacks detection.

The remainder of this article is organized as follows. Section 2 describes surveys and systematic literature reviews related to DL-based web attacks detection. Section 3 presents the methodology followed in conducting this SLR. Section 4 presents the results and analysis. Section 5 describes the limitations of this study. Finally, Section 6 concludes this study.

2. Related Work

As far as we know, this is the first study to systematically investigate Deep Learning for vulnerability and attack detection on Web applications. However, there are also other interesting partial surveys in the area.

Ref. [2] conducted a survey on the detection and prevention of web vulnerabilities. They explained in detail web vulnerabilities and the different methods used to counter them. However, they only reviewed traditional Machine Learning-based web attacks detection research works. Related surveys, such as [22,23], have described Machine Learning or Deep Learning applications to cyber-security problems but without paying a particular attention to web applications security. Refs. [24,25] are two surveys about web vulnerabilities classification and countermeasures. They both do not focus on Deep Learning-based approaches for web vulnerabilities detection and do not follow the SLR protocol. Ref. [26] is an SLR on web services attacks and security. Ref. [27] is a recently published survey presenting the latest Machine Learning and Deep Learning-based approaches used for detecting XSS attacks.

3. Review Methodology

The systematic review in this study conducted by adapting the strategy proposed by [28,29] consists of three main steps; including (i) planning, (ii) conducting, and (iii) reporting the review results. The detail of these steps is summarized in Figure 5. The planning phase (first step) determines if there is a need to conduct a systematic review. The second step develops a review protocol including (i) identifying research questions, (ii) creating a search strategy, (iii) defining the study selection criteria, (iv) developing quality assessment rules, (v) determining the data extraction strategies that will be used, and (vi) defining the methods that will be used to synthesize the extracted data. We provide details about the proposed protocol in the following subsections. The second phase explains the necessary primary steps to conduct a systematic review of the study. In the two first steps (steps 3 and 4), we select the PS by applying the selection criteria and quality assessment rules defined in the planning phase, and then we describe their contents. In the second step (step 5), we extract from selected PS the data that will help answer the research questions. In the third step (step 6), we use different methods to synthesize the extracted data to facilitate the answer to the research questions. In the fourth step (step 7), we answer the research questions based on the synthesized data. In the reporting phase, we discuss the review results, and we state the limitations of selected PS.

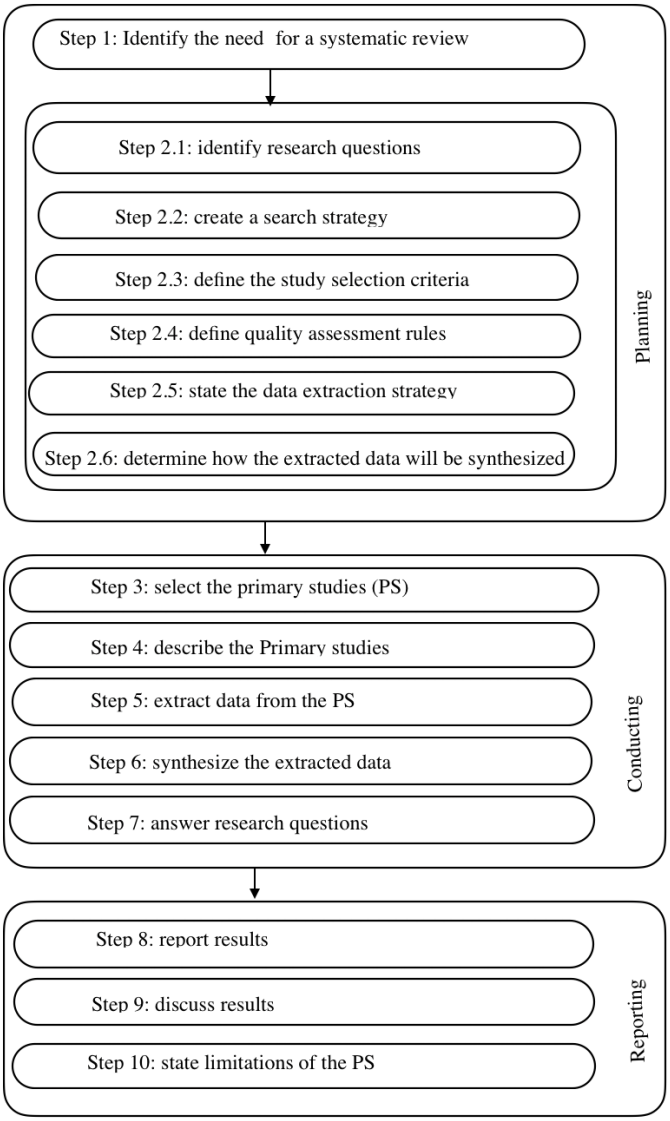


Figure 5. Systematic review process.

3.1. Research Questions

The main focus of our study is to analyze scientific literature on web attacks detection using Deep Learning techniques available from 2010 to September 2021 inclusively. Based on that, we specify the research questions (RQ1–RQ10) that we detailed in Table 1.

Table 1. Research questions.

RQ No.	Question	Motivation
RQ1-1	What is the annual number of studies on DL-based web attacks detection?	Estimate the published articles per year on the DL-based web attacks detection.
RQ1-2	What is the percentage of studies published in journals and conferences?	Compare the percentage of studies on the DL-based web attacks detection published in journals and conferences.
RQ2	What datasets are used to evaluate the proposed approaches for DL-based web attacks detection?	Identify the datasets that are commonly used in the evaluation of prediction models.
RQ3	What frameworks and platforms are used to implement the proposed solutions for DL-based web attacks detection?	Give an overview of the available frameworks and platforms used for developing DL-based web attacks detection models.
RQ4	What performance metrics are used in DL-based web attacks detection literature?	Enumerate the most commonly used performance metrics in web intrusion detection systems.
RQ5	What are the feature selection and extraction approaches used in DL-based web attacks detection literature?	Identify the feature extraction and selection approaches used for DL-based web attacks detection models.
RQ6	What classification models are used to detect web vulnerabilities?	Identify the classification models that are commonly used for detecting web attacks.
RQ7	What types of web attacks do the proposed approaches detect?	Identify whether the proposed solutions for DL-based web attacks detection have a general purpose or target a specific type of web attacks.
RQ8	What is the performance of DL-based web attacks detection models?	Report the experimentation details of the proposed DL-based web attacks detection models.
RQ9	What is the research focus of the PS?	Identify the main objective of the PS.
RQ10	What limitations do the proposed solutions for DL-based web attacks detection have according to the authors?	Identify the limitations of the studies as stated by their authors.

3.2. Search Strategy

We performed exhaustive searches on different online libraries. The following search string yielded most appropriated results:

- (“deep learning” OR “neural networks”) AND (“web attacks” OR “web security” OR “web application security” OR “web vulnerabilities”)

We adapted search strings to be suitable for each database according to their specific requirements. Then, we queried each database by title, abstract, and keywords. The digital libraries utilized in this study include Scopus, Web Of Science, ScienceDirect, IEEE, ACM, and Springer. The first part of Figure 6 (blue) indicates the steps followed in conducting the search strategy.

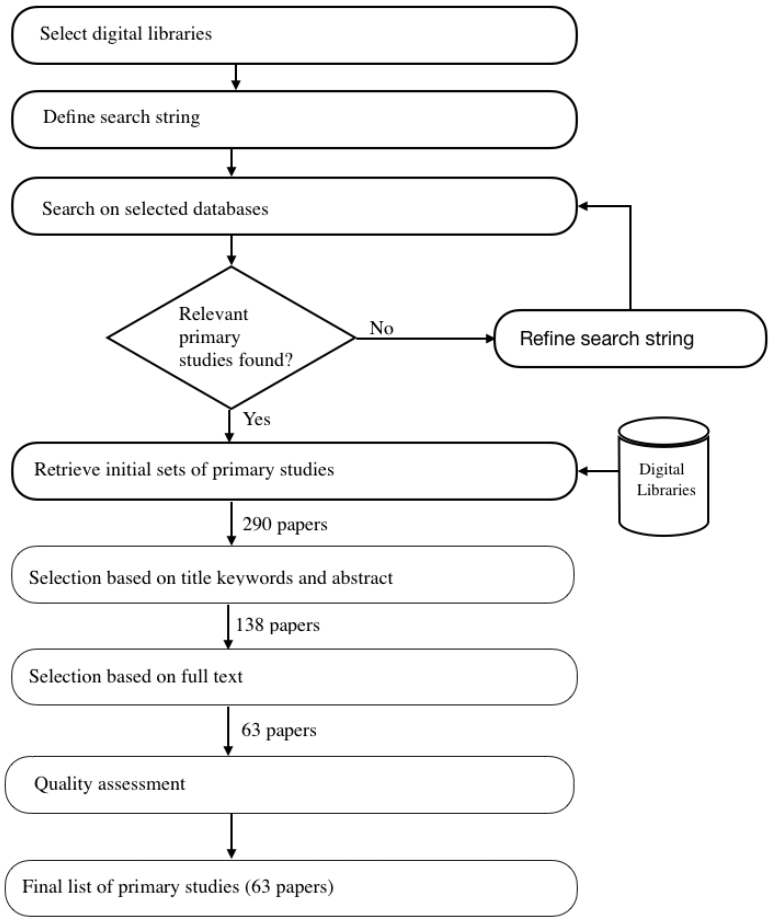


Figure 6. Search and study selection process.

3.3. Study Selection

Figure 6 explains our search strategy achieved an initial set of 290 PS. Then, we apply additional filtering that aims at:

1. Getting rid of duplicate PS.
2. Applying inclusion and exclusion criteria to determine the relevant PS.
3. Performing a quality assessment of selected PS.

Our criteria for inclusion included the following:

- Research works include web attacks and Deep Learning terms in title or abstract.
- Research works whose main topic is detecting web attacks using Deep Learning.
- Research works published between 2010 and September 2021.
- Research works that contain a quantitative evaluation of the proposed solutions.

Our criteria for exclusion included:

- Studies involve detecting web attacks using methods other than Deep Learning.
- Studies involve the development of Deep Learning models for the detection of Malware, Spam, Network intrusions, or Phishing attacks.
- Studies not considered as published journal papers or conference proceedings.
- Studies written in a language other than English.

After the phase mentioned above, we found 63 journal and conference articles fulfilling our selection strategy. The second part of Figure 6 (green) indicates the steps followed in conducting the study selection strategy.

3.4. Quality Assessment Criteria

To assess the selected PS quality, we performed a quality analysis questionnaire. This quality assessment aims at giving a quality score to each PS and not intended to eliminate any PS selected at the previous phase of the SLR. Table 2 explains a total of eight quality assessment questions. Each question was scored as follows: "fully answered" = 1, "partly answered" = 0.5, "not answered" = 0. Between 0 and 8, quality scores are assigned to each PS by summing the individual question scores. Table 3 shows the selected PS with their QA scores.

Table 2. Quality assessment questions.

QQ No.	Question
QQ1	Is the objective of the study clear?
QQ2	Is the data collection procedure clearly defined?
QQ3	Does the study provide any tool or source online?
QQ4	Is there a comparison among techniques?
QQ5	Does the author provide sufficient details about the experiment?
QQ6	Are problems of validity or trust of the results obtained adequately discussed?
QQ7	Does the study clearly define the performance parameters used?
QQ8	Is there a clearly defined relationship between objectives, data obtained, interpretation, and conclusions?

Table 3. Quality assurance scores of selected PS.

ID	Ref	Score	ID	Ref	Score
PS1	[30]	6.5	PS29	[31]	6
PS2	[32]	5.5	PS30	[33]	6
PS3	[34]	6	PS31	[35]	4
PS4	[36]	6	PS32	[37]	4.5
PS5	[38]	4	PS33	[39]	4.5
PS6	[40]	5.5	PS34	[41]	6
PS7	[42]	5	PS35	[43]	4.5
PS8	[44]	5	PS36	[45]	4.25
PS9	[46]	5	PS37	[47]	5
PS10	[48]	5.5	PS38	[49]	5.5
PS11	[50]	5	PS39	[51]	6
PS12	[52]	5	PS40	[53]	5.5
PS13	[54]	5	PS41	[55]	5.75
PS14	[56]	6.5	PS42	[57]	6
PS15	[58]	4.5	PS43	[59]	4.5
PS16	[60]	5.5	PS44	[61]	3.5
PS17	[62]	5	PS45	[63]	6.5
PS18	[64]	7	PS46	[65]	4
PS19	[66]	5	PS47	[67]	5
PS20	[68]	5	PS48	[69]	4.5
PS21	[70]	6	PS49	[71]	6
PS22	[72]	5	PS50	[73]	4.5
PS23	[74]	4.75	PS51	[75]	5.5
PS24	[76]	6	PS52	[77]	5.5
PS25	[78]	5.5	PS53	[79]	5
PS26	[80]	5	PS54	[81]	5
PS27	[82]	5.5	PS55	[83]	8
PS28	[84]	6	PS56	[85]	5
			PS57	[86]	5
PS58	[87]	4.5	PS59	[88]	7
PS60	[50]	5	PS61	[89]	6.5
PS62	[90]	6	PS63	[91]	6

The quality analysis of selected studies shows that the DL-based web attacks detection research domain is yet to be explored properly. Few studies obtained good scores; still, most of the studies reported average scores.

3.5. Data Extraction

In this step, the main focus is to extract important information from each PS that helps answer research questions and store that extracted data in spreadsheets to use in the data synthesis process later on. Table 4 provides the data extraction form used in this SLR study.

Table 4. Data extraction form.

No	Attribute Name	Research Question
01.	Study identifier	
02.	Year of publication	RQ1
03.	Type of study	RQ2
04.	Datasets	RQ3
05.	Frameworks	RQ4
06.	Performance measures	RQ5
07.	Feature extraction	RQ6
08.	Classification models	RQ7
09.	Type of web attacks	RQ8
10.	Experimental performance of proposed models	RQ9
11.	Limitations	RQ10
12.	Main objective	RQ11

3.6. Data Synthesis

Data synthesis aims at using various methods to synthesize the data extracted from the selected PS to answer the research questions. Therefore, we considered different synthesis methods, including narrative synthesis, tables, and visualization tools such as bar charts, pie charts, and line graphs.

4. Results and Discussion

In the following subsections, the findings of this SLR study will be presented and discussed for each Research Question (RQ).

4.1. RQ1: What Are the Trend and Types of Studies on DL-Based Web Attacks Detection?

This question aims at reviewing bibliometric studies in the DL-based web attacks detection domain; the answers reflect publication information of Primary Studies.

RQ1-1: What is the annual number of studies on DL-based web attacks detection? Figure 7 is the year-by-year presentation of selected studies. The year started in 2010 and ended in September 2021, and we have shown ten years of data of 63 articles. The figure shows the disparate distribution of papers according to the years. In 2012, we found the first and only one research article on the DL-based web attacks detection. Since that day, several research articles have been published on the topic. The number of papers published reaches the maximum in 2019 and then decreased in 2020. We finished the SLR by September 2021. Thus, the 12 papers published in 2021 are not representative of the papers published in that year. This scenario suggests that the distribution of published research papers is not equal, and web vulnerability detection topics using Deep Learning will gain more attention in coming years.

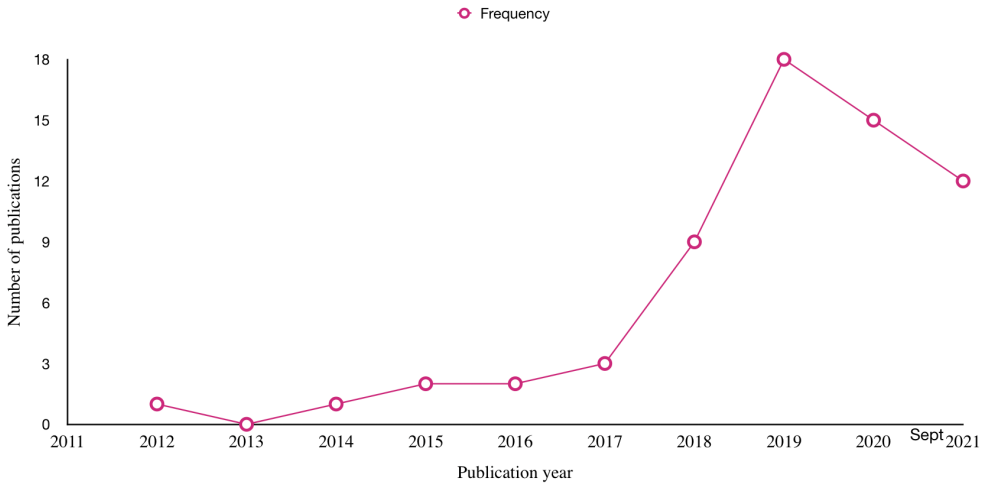


Figure 7. Year-wise distribution of studies.

RQ1-2: What is the percentage of studies published in journals and conferences? Figure 8 shows that 57% of PS are published in conferences while 43% are published in journals.

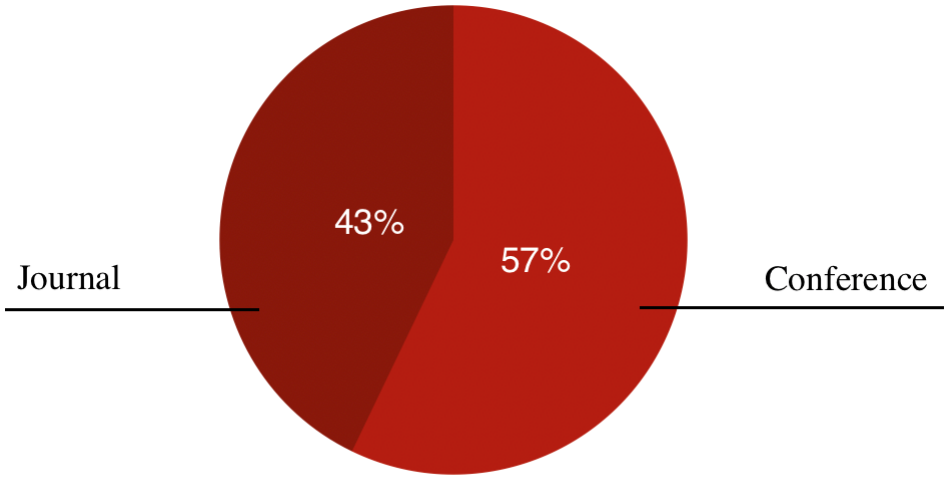


Figure 8. Percentage of studies published in journals and conferences.

The RQ1 answer indicates that interest in detecting web vulnerabilities using Deep Learning models is very recent; since 2019, the number of articles published in this research field has increased significantly. In addition, the number of journal articles is less than the number of conference papers.

4.2. RQ2: What Datasets Are Used to Evaluate the Proposed Approaches for DL-Based Web Attacks Detection?

In Machine Learning approaches, the choice of the dataset is a key point in the evaluation of detection models' performance. Thus, this question aims at reviewing and discussing the limitations of datasets commonly used in web attacks detection.

According to the dataset type, we shaped the studies into the following two classes: (i) public datasets—free and open access and (ii) private datasets—not open access.

We found that some studies combine more than one public dataset or even use private and public datasets at the same time to conduct the experiments. Figure 9 gives an overview of the datasets percentage utilized in articles reviewed in this study.

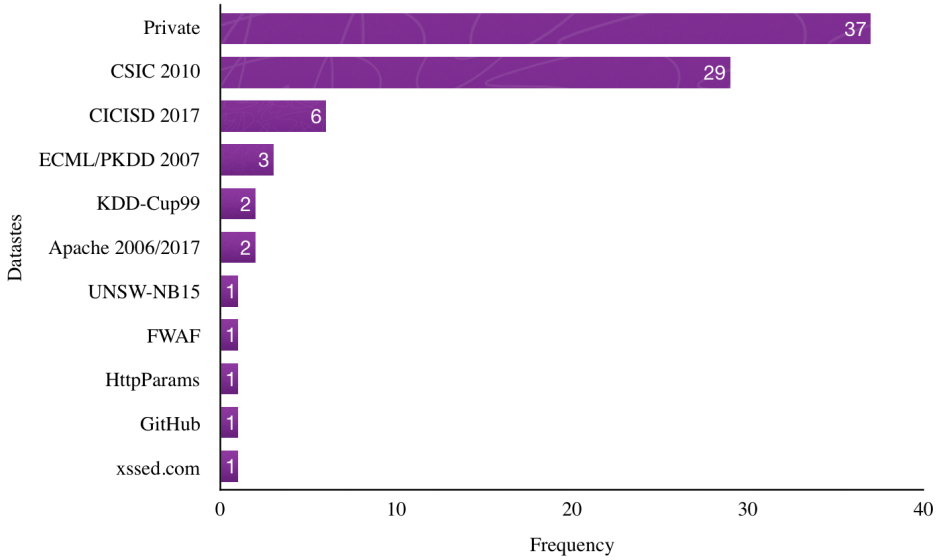


Figure 9. Histogram of datasets used in selected PS.

We found that 37 studies used private datasets, 29 studies used the CSIC-2010 dataset, three studies used the ECML/PKDD 2007 dataset, two studies used KDD-Cup99, six studies use the CICIDS 2017 dataset, and six studies used publicly available datasets not very commonly used in research works (e.g., xssed.com, Apache 2006/2017, HttpParams). We detail below the public datasets that have been used in the majority of the reviewed studies:

- KDD-Cup99: The dataset contains 41 features. It can get in three following versions: (i) complete training set, (ii) 10% of the training set, and (iii) testing set. It is mainly used for building networks intrusion detection models.
- UNSW-NB15: The dataset combines actual modern normal activities and synthetic contemporary attack behaviors. It has nine types of attacks, which are mostly related to network intrusion. The training dataset includes 175,341 instances whereas 82,332 instances are in the testing set. It is also mostly used in networks intrusion detection.
- CICIDS-2017: The Canadian Institute for Cybersecurity created this dataset. It has 2,830,540 distinct instances and 83 features containing 15 class labels (1 normal + 14 attack labels). The dataset contains only 2180 web attacks instances, which means it is insufficient for evaluating a web attacks detection model.
- CSIC-2010: The dataset contains the generated traffic targeted to an e-commerce web application. It is an automatically generated dataset that contains 36,000 normal requests and more than 25,000 anomalous requests (i.e., web attacks).
- ECML/PKDD 2007: The dataset is part of ECML and PKDD conferences on Machine Learning. The dataset contains 35,006 normal traffic and 15,110 malicious web requests. The dataset was developed by collecting real traffic and then processed to mask parameter names and values—replacing them with random values.

DL-based web attacks detection models development faces the limitation of problems related to datasets available for evaluating detection models' performance. In fact, existing public datasets are outdated and simplistic, meaning that they do not include newly discovered web attacks and they do not reflect the complexity of real-world web applications. Moreover, comparing research works in this field is almost impossible because most researchers use private datasets, and even when they adopt public datasets, they use different portions and apply different pre-processing techniques, which results in different versions of the same dataset. Additionally, public datasets need thorough pre-processing. Otherwise, evaluation results can not reflect the real models' performance. For instance, a

dataset with many duplicated instances results in biased accuracy. More, a poor feature selection/extraction strategy can produce over-fitted or under-fitted models.

4.3. RQ3: What Frameworks and Platforms Are Used to Implement the Proposed Solutions for DL-Based Web Attacks Detection?

The main objective behind this research question is (i) to give an overview of software and platforms commonly used in DL-based web attacks detection models development and (ii) to gauge the interest of researchers to give technical implementation details.

Figure 10 summarizes the frequency of usage of frameworks and platforms that are used for developing DL-based web attacks detection models. First, it shows that the most used frameworks and platforms are Keras and TensorFlow. Then, various frameworks such as PyTorch, Theano, Scikit-learn, Weka, and MATLAB are used by few studies. However, 23 studies did not provide implementation details.

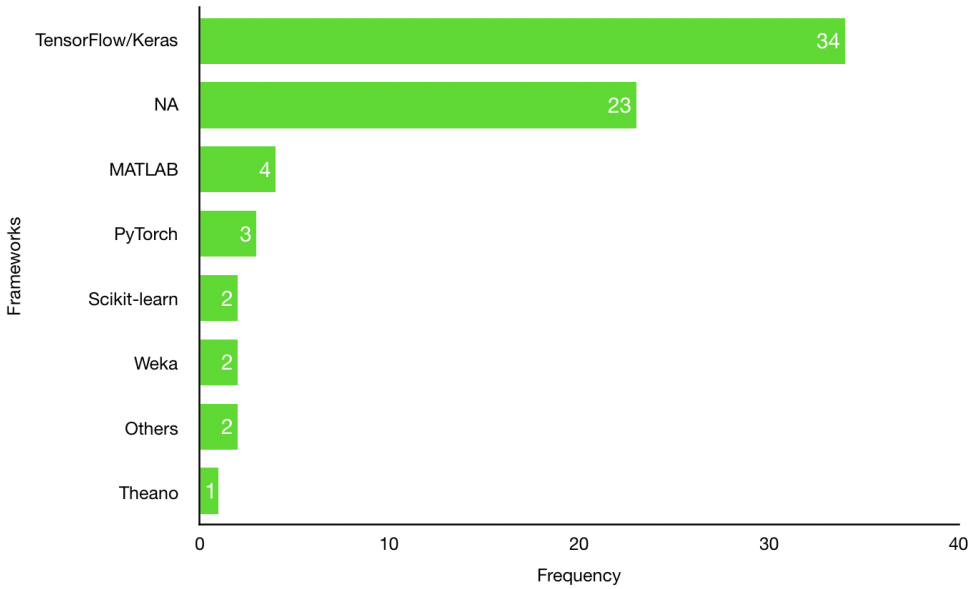


Figure 10. Histogram of frameworks and platforms used in selected PS.

4.4. RQ4: What Performance Metrics Are Used in DL-Based Web Attacks Detection Literature?

The objective of this research question is to present and discuss the performance metrics that are commonly used to evaluate DL-based web attacks detection models.

Using DL models for web attacks detection amounts to developing a classification model that can identify whether a web application is vulnerable or not (i.e., binary classification), or whether it is vulnerable to a specific web attack (e.g., vulnerable or not to XSS attacks), or determine to which web attack it is vulnerable (multi-classification problem).

Different performance metrics have been used in the literature to evaluate DL-based web attacks detection models. However, we detail below the most widely used metrics as reported in Figure 11.

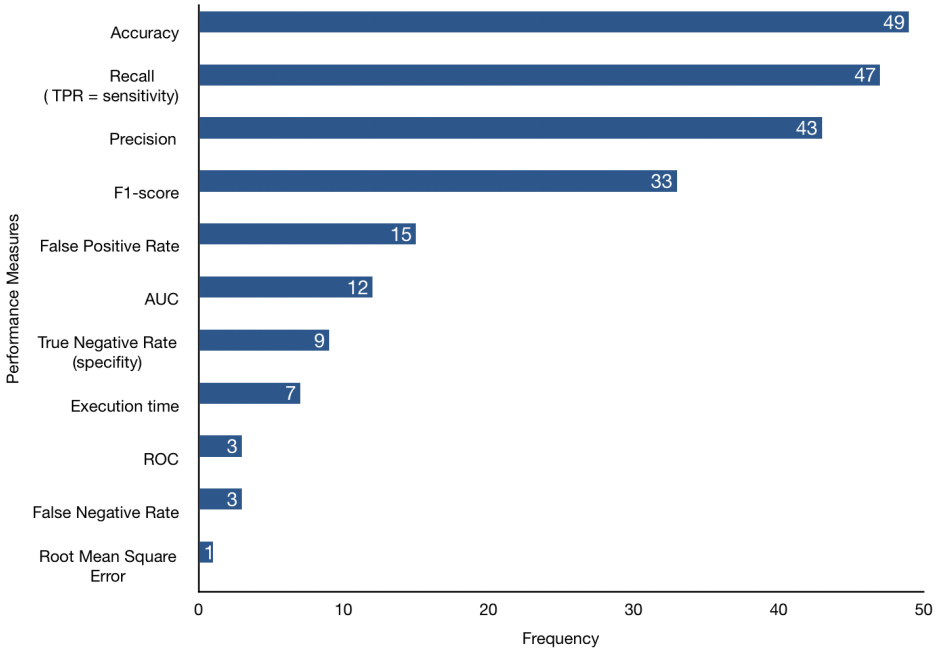


Figure 11. Histogram of performance metrics used in selected PS.

- Accuracy: measures the ratio of the number of samples classified correctly over the total number of samples. Accuracy is not useful when the classes are unbalanced (i.e., there are a significantly larger number of examples from one class than from another). However, it does provide valuable insight when the classes are balanced. Usually, it is recommended to use recall and precision along with accuracy.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

- Recall or Sensitivity or True Positive Rate or Detection Rate: measures the proportion of actual positives that are correctly identified.The higher value of sensitivity would mean a higher value of true positive and lower value of false negative. The lower value of sensitivity would mean lower value of true positive and higher value of false negatives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

- Precision: measures the number of positive class predictions that actually belong to the positive class. Precision does not quantify how many real positive examples were predicted as belonging to the negative class, that is why it is advisable to compute the True Negative Rate (TNR) metric.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

- F1-score: weighted harmonic mean of precision (P) and recall (R) measures. It is recommended to use F1-score rather than accuracy if we need to seek a balance between precision and recall and there is an uneven class distribution:

$$\text{F1-score} = \frac{1}{\alpha.\frac{1}{P} + (1-\alpha).\frac{1}{R}} \tag{4}$$

α is chosen such that recall is considered α times as important as precision. If α is $\frac{1}{2}$, then precision and recall are given equal importance. The choice of α, and thus the trade-off between precision and recall, depends on the classification problem.

- False Positive Rate: the ratio of all benign samples incorrectly classified as malicious. It is used to plot the ROC curve:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{5}$$

  In intrusion detection systems, it is important to have a low FPR. Otherwise, the detection system is considered not reliable.
- Area Under the Curve (AUC): measures two-dimensional area under the ROC curve, ranging from 0 to 1, indicating a model's ability to distinguish between classes. Models should have a high value of AUC, so-called models with good skill. The ROC (Receiver Operating Characteristic) curve is the plot between the TPR (y-axis) and the FPR (x-axis).
- True Negative Rate or Specificity: measures the proportion of actual negatives correctly identified. The higher value of specificity would mean a higher value of true negative and lower false positive rate. The lower value of specificity would mean a lower value of true negative and higher value of false positive.

$$\text{TNR} = \frac{\text{TN}}{\text{FP} + \text{TN}} = 1 - \text{FPR} \tag{6}$$

For multi-classification problems, it is straightforward to calculate accuracy; however, metrics such as precision, recall, FPR, F1-Score, and AUC cannot be calculated easily because TP and TN do not exist for such problems. These metrics can only be determined for three or plus class problems by collapsing the problem into a two-class problem (i.e., all classes versus one class), where the metrics are calculated for each class. Usually, for multi-class problems, only accuracy is used.

4.5. RQ5: What Are the Feature Selection and Extraction Approaches Used in DL-Based Web Attacks Detection Literature?

Although Deep Learning performs automatic feature selection and extraction during models training, in the context of web attacks detection, the model input is textual in most cases, which means that prior to model training, the input should be processed beforehand. In this research question, we will shed the light on the feature selection and extraction approaches that are used to process the input to DL-based web attacks detection models.

Feature engineering is an essential step in the construction of Machine Learning models. It includes feature selection and feature extraction. Feature selection starts from a set of attributes and retains the most relevant ones. Feature extraction starts from a set of attributes and derives attributes intended to be informative and non-redundant.

According to reviewed papers, we can group feature selection and extraction approaches into three categories. (1) In intrinsic feature selection and extraction, features are selected and extracted in the course of model training, thereby, the feature selection and extraction are performed by the DL classification model. In this category, the model input is either a set of numerical features likely selected using a feature selection method, or a result of a simple conversion from textual to numerical format. (2) Extrinsic and intrinsic feature selection and extraction consists of applying external feature selection and extraction methods on the input; then, the resulting features are fed to the DL classification model, which extracts and selects more abstract and complex features during training. In this category, the external feature extraction methods are more sophisticated; they are either based on techniques that are employed in Natural Language Processing (NLP) problems, namely word-level and character-level embedding, or on manual feature extraction (i.e., features can be extracted using automatic tools, but these tools are built according to experts instructions). (3) In extrinsic feature selection and extraction, feature selection

and extraction are external to the classification model construction. It can involve text classification techniques, manual feature extraction, and/or DL models. In this category, a traditional Machine Learning model is used for classification.

As seen in Figure 12, extrinsic and intrinsic feature selection and extraction is the most used approach in reviewed studies, which is followed by intrinsic feature selection and extraction, and then extrinsic feature selection and extraction. As for external feature extraction techniques, we can see from Figure 13 that the most used methods are word-level embedding, followed by manual feature extraction, and then character-level embedding and Encoder–Decoder models. Feature selection came in the fourth position after manual feature extraction, but only one study among the seven studies has identified the technique used.

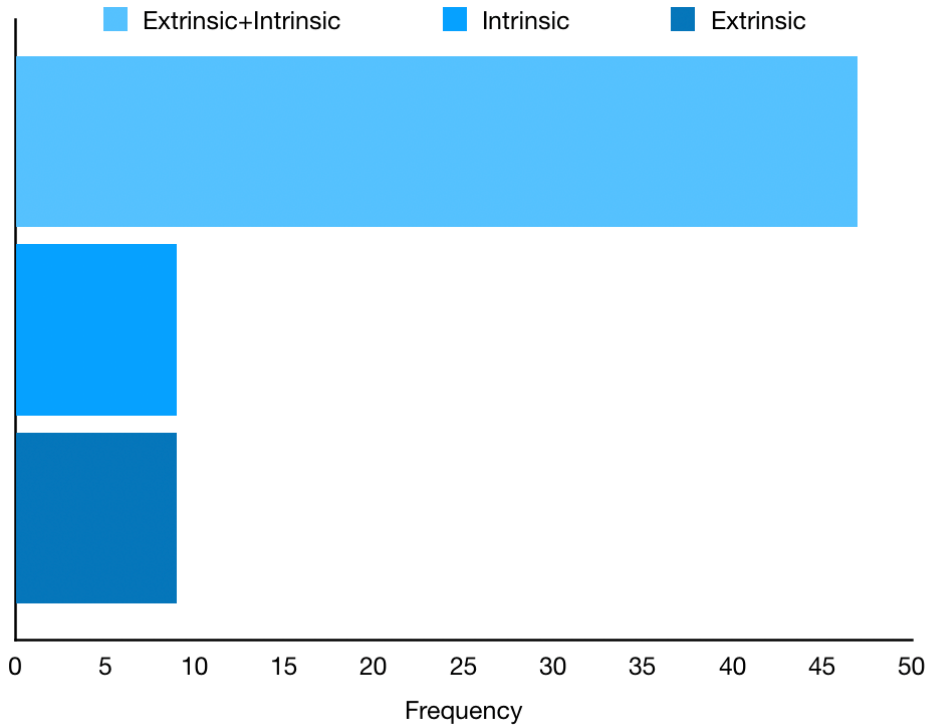


Figure 12. Histogram of feature selection and extraction categories used in selected PS.

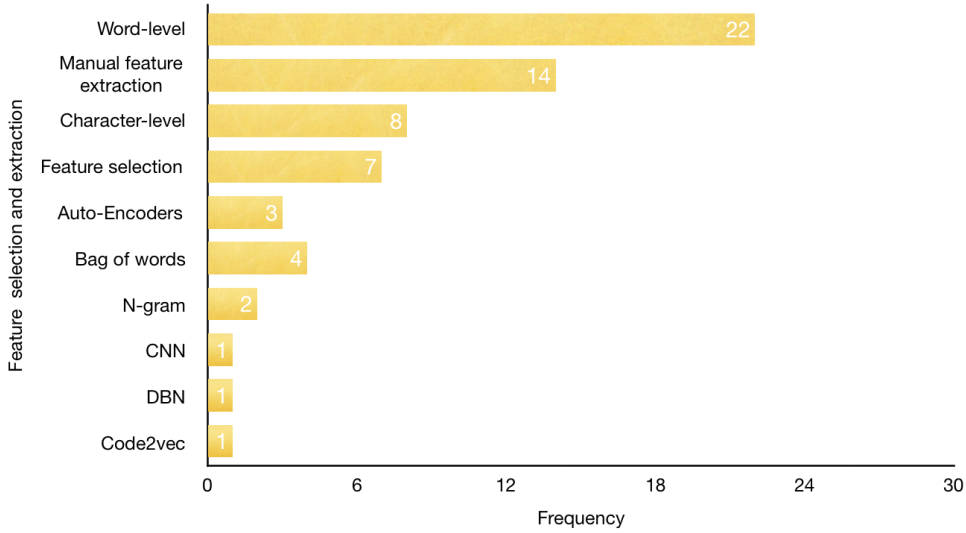


Figure 13. Histogram of feature selection and extraction methods used in selected PS.

We observed that most reviewed studies do not discuss feature engineering in detail. Indeed, even if Deep Learning can extract abstract and complex features in the course of model training, it is important to identify whether the input to the DL model is a result of external feature extraction and selection or of a simple conversion from textual to numerical format and that feature selection and extraction is performed by the classification model during training. Yet, the studies that pay more attention to feature engineering are those using traditional ML models for classification and DL models for feature selection and extraction.

4.6. RQ6: What Classification Models Are Used to Detect Web Vulnerabilities?

The main objective of this research question is to identify DL models used for the classification of web attacks.

In this section, we extract from the selected PS the classification models used for detecting web attacks. As seen in Figure 14, there exist different Deep Learning models with the exception of a few models such as Generative Adversarial Network and variants of Encoder–Decoders models. Few studies use Deep Learning models as feature extractors and Machine Learning (ML) algorithms as web attacks classifiers. This is why they were part of this study. Finally, we noticed that CNN, LSTM, and DFFN are the most Deep Learning models used in the reviewed studies.

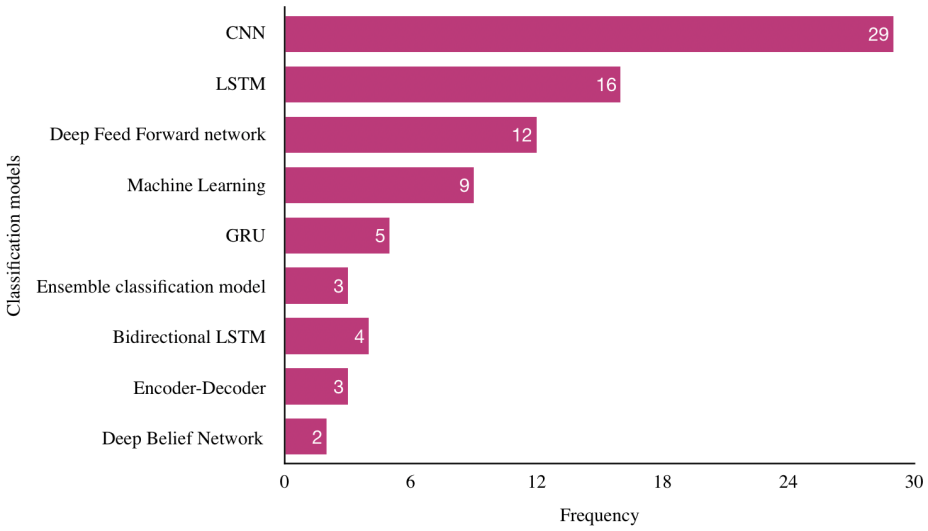


Figure 14. Histogram of classification models used in selected PS.

Since Generative Adversarial Network and Encoder–Decoder models have shown promising results in networks intrusion detection (e.g., [92–95]), exploring these models in web attacks detection is an important area for improvement.

4.7. RQ7: What Types of Web Attacks Do the Proposed Approaches Detect?

In this research question, we attempt to identify the types of web attacks that reviewed studies try to detect using Deep Learning models.

As seen in Figure 15, most studies develop Deep Learning models for detecting web attacks without targeting a specific type of web attacks. Still, some studies focus on detecting specific web attacks, namely query injection attacks, XSS attacks, and file and path injection attacks. Moreover, 46 studies develop binary classification models, while only 15 studies develop multi-classification models (Figure 16).

Since most if not all studies do not mention whether the proposed DL models were trained on a dataset that contains an even number of instances for each type of web attack, it is important to evaluate how well these DL models will perform in detecting specific

types of web attacks. This can be achieved by giving more attention to the development of multi-classification models.

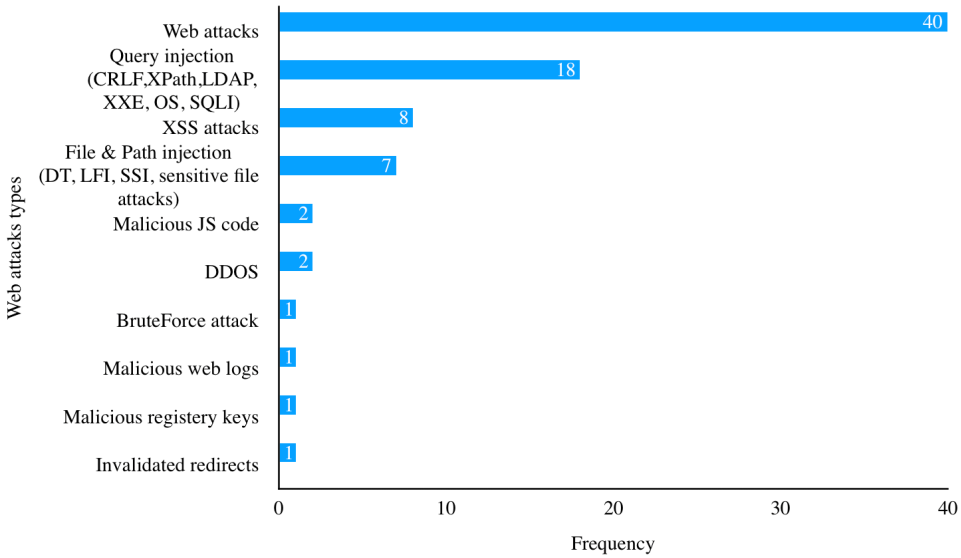


Figure 15. Histogram of web attacks types used in the selected PS.

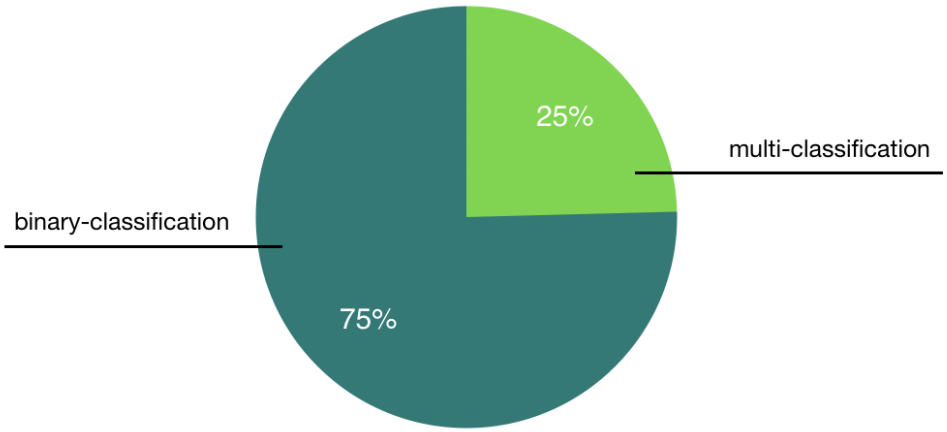


Figure 16. Percentage of studies proposing binary or multi-classification models.

4.8. RQ8: What Is the Performance of DL-Based Web Attacks Detection Models?

In this research question, we report the experimentation details of DL-based web attacks detection models proposed in the selected PS.

Table 5 summarized the experiments conducted in the reviewed studies: targeted web attacks, classification models, datasets, performance metrics, as well as limitations of studies as stated by their authors. If a given reviewed study performed many experiments, we report the experiment that yielded the best results.

Table 5. A summary of experiments conducted in selected PS.

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS1	[30]	SQLI	CNN	Combination of CSIC-2010, KDD-Cup99, UNSW-NB15, and private dataset	binary	-Accuracy: 0.9950 -Precision: 0.9898 -F1-score: 0.99 -TPR: 1	-Restricted datasets -Limited to one type of attack
PS2	[32]	Web attacks injected in HTTP requests	CNN	CSIC-2010	binary	-Accuracy: 0.9820 -TPR: 0.98 -TNR: 0.97	-Online learning -Adversarial attacks
PS3	[34]	JavaScript attacks	Logistic regression	Private dataset	binary	-Accuracy: 0.9482 -Precision: 0.949 -TPR: 0.94	-Minification and obfuscation of JS code -Long training time
PS4	[36]	Malicious HTML pages	DFFN	Private dataset	binary	-ROC: 0.975	-Establishing the ground truth
PS5	[38]	Web attacks injected in HTTP requests	CNN	CSIC-2010	binary	-Accuracy: 0.9649 -TPR: 0.934 -FPR: 0.13	-Consider the whole HTTP request message in the detection system.
PS6	[40]	Web attacks injected in HTTP requests	Weighted average ensemble of ResNet	CSIC-2010	binary	-Accuracy: 0.9941 -TNR: 0.99	-Evaluate other ensemble models and experiment with other feature representation techniques
PS7	[42]	Web attacks injected in HTTP requests	CNN + GRU	CSIC-2010	binary	-Accuracy: 0.99 -Precision: 0.9982 -F1-score: 0.987 -TPR: 0.97	Not Available
PS8	[44]	XSS attacks	CNN + LSTM	xssed.com	binary	-Accuracy: 0.993 -Precision: 0.999 -F1-score: 0.995 -TPR: 0.99 -AUC: 0.95	Scarcity of datasets in the field of web security
PS9	[46]	Web attacks injected in HTTP requests	Neural Network	CSIC-2010	binary	-Accuracy: 0.84 -Precision: 0.83 -F1-score: 0.79 -TPR: 0.82 -AUC: 0.86	Not available

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS10	[48]	SQLI	CNN	Private dataset	binary	-Accuracy: 0.953 -Precision: 0.954 F1-score: 0.734 -TPR: 0.59	-Collect a dataset for SQL injection attack detection -Build a word2vec model for PHP source code -Use Control Flow Graph attributes
			MLP	Private dataset	binary	-Accuracy: 0.953 -Precision: 0.91 -F1-score: 0.746 -TPR: 0.637	
PS11	[50]	Web attacks injected in HTTP requests	CNN	CSIC-2010	binary	-Accuracy: 0.9797 -Precision: 0.9743 -F1-score: 0.975 -TPR: 0.97 -FPR: 0.03 -TNR: 0.96 -AUC: 0.96	-Datasets available for evaluating DL-based web attacks detection model are limited -Consider multi-class classification instead of binary classification
PS12	[52]	SQLI and DDoS	Neural Network	Private dataset	multi-class	-Accuracy: 0.97	-Variable size network -Memory and time constraints
PS13	[54]	Web attacks injected in HTTP requests	Ensemble classification of Neural Network and ML algorithms	CSIC-2010	binary	-Accuracy: 0.9098	-Compare the proposed method with other ensemble classification models -Apply the proposed method in malware
				ECML-PKDD	binary	-Accuracy: 0.9056	
PS14	[56]	-Password guessing and authentication -SQLI -Application vulnerability attack	CNN + LSTM	Private dataset	multi-class	-Accuracy: 0.9807 -Precision: 0.9706 -F1-score: 0.981 -TPR: 0.99 -TNR: 0.99	-Re-validation and retraining of DL-based misuse intrusion detection tools -Combine misuse detection tools and signature based tools
PS15	[58]	Web attacks injected in HTTP requests	Isolation Forest	CSIC-2010	binary	-Accuracy: 0.8832 -Precision: 0.8029 -F1-score: 0.841 -TPR: 0.88 -TNR: 0.88	-Use other deep learning techniques

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS16	[60]	Web attacks injected in GET HTTP requests	LSTM + Multi-Layer Perceptron (MLP)	CSIC-2010	binary	-Accuracy: 0.9842 -TPR: 0.97 -TNR: 0.99	-Long URLs are not handled very well -The proposed model can not dynamically leverage between true positives and false positives
			GRU + MLP	Private dataset	binary	-Accuracy: 0.9856 -TPR: 0.98 -TNR: 0.98	
PS17	[62]	Web attacks injected in HTTP requests	LSTM	CSIC-2010	binary	-Accuracy: 0.9997 -Precision: 0.995 -TPR: 0.995	Not Available
PS18	[64]	Web attacks injected in system calls	LSTM	Private dataset	binary	-AUC: 0.96	-Certain classes of attacks could be missed -Privacy leakage -Retraining ML models for each new considered web application -Adversarial attacks
PS19	[66]	-SQLI -XSS -Brute-Force	GRU based Encoder-Decoder with attention mechanism	CSIC 2010 + CICIDS 2017	multi-class	-TPR: 0.94 -FPR: 0.003	Not Available
PS20	[68]	Web attacks injected in HTTP requests	Bi-directional LSTM	CSIC 2010	binary	-Accuracy: 0.9835 -Precision: 0.99 -F1-score: 0.985 -TPR: 0.98 -FPR: 0.014	Not Available
PS21	[70]	Web attacks injected in HTTP requests	CNN + LSTM	CSIC 2010	binary	-Accuracy: 0.9779 -Precision: 0.9854 -F1-score: 0.9872 -TPR: 0.9604	-Include more web attacks in the dataset -Consider scenario based attacks (i.e., correlated requests) -Deploy the model in a practical web service
PS22	[72]	Web attacks hidden in HTTP requests	GRU	Private dataset	binary	-Accuracy: 0.985	Not Available

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS23	[74]	Web attacks hidden in HTTP requests	CNN	CSIC 2010	binary	-Accuracy: 0.9726 -Precision: 0.9771 -F1-score: 0.965 -TPR: 0.95 -FPR: 0.01	-HTTP requests are misclassified if containing strings that never appear in the training set
			LSTM	CSIC 2010	binary	-Accuracy: 0.9699 -Precision: 0.9882 -F1-score: 0.962 -TPR: 0.937 -FPR: 0.007	-The way in which the request is split into a sequence of words influences false positives occurrence
			CNN-LSTM	CSIC 2010	binary	-Accuracy: 0.9550 -Precision: 0.9463 -F1-score: 0.944 -TPR: 0.94 -FPR: 0.03	-Inspect all the fields of the HTTP request
			LSTM-CNN	CSIC 2010	binary	-Accuracy: 0.9602 -Precision: 0.9652 -F1-score: 0.950 -TPR: 0.935 -FPR: 0.02	-Use more sophisticated models
PS24	[76]	XSS attacks	LSTM	Private dataset	binary	-Precision: 0.995 -F1-score: 0.987 -TPR: 0.979 -AUC: 0.98	-Collect more XSS attacks
PS25	[78]	Web attacks hidden in HTTP requests	CNN-GRU	Private dataset	binary	-Accuracy: 0.9961 -Precision: 0.9963 -F1-score: 0.9961 -TPR: 0.9958	-Reduce memory consumption -Online update of the trained model
PS26	[80]	Web attacks hidden in HTTP requests	SVM	CSIC 2010	binary	-Accuracy: 0.9897 -Precision: 0.9970 -TPR: 0.986	Not Available

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS27	[82]	SQLI	Random Forest	SQL injection datasets published in Github	binary	-Accuracy: 0.998 -Precision: 0.999 -F1-score: 0.999 -TPR: 0.986 -TNR: 0.999 -AUC: 0.999	-Non-malicious data is biased toward domain-specific traffic
			DFFN	SQL injection datasets published in Github	binary	-Accuracy: 0.984 -Precision: 0.934 -F1-score: 0.873 -TPR: 0.820 -TNR: 0.995 -AUC: 0.992	
PS28	[84]	-SQLI -XSS -Object de-serialization	Stacked Denoising Auto-Encoder	Private dataset	multi-class	-Precision: 0.906 -F1-score: 0.918 -TPR: 0.928	-Investigate more complex neural networks -Detect zero-day attacks -Online updating of trained models -Distributed machine learning analysis
PS29	[31]	DDOS	Stacked Auto-Encoder	Private dataset	multi-class	-TPR: 0.98 -FPR: 0.012	Not Available
PS30	[33]	SQLI	LSTM	Private dataset	binary	-Accuracy: 0.9917 -Precision: 0.9110 -TPR: 0.99 -FPR: 0.90	Not Available
			MLP	Private dataset	binary	-Accuracy: 0.9975 -Precision: 0.9727 -TPR: 0.99 -FPR: 0.26	
PS31	[35]	SQLI	MLP	Private dataset	binary	-Accuracy: 1 -TPR: 1 -TNR: 1	Not Available
PS32	[37]	Web attacks hidden in web logs	DFFN	Apache 2006	binary	-Accuracy: 0.9973 -Precision: 0.9976 -F1-Score: 0.995 -TPR: 0.99	Not Available
			DFFN	Apache 2017	binary	-Accuracy: 0.9838 -Precision: 0.9984 -F1-score: 0.972 -TPR: 0.94	

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS33	[39]	Web attacks hidden in HTTP requests	LSTM	CSIC 2010	binary	-Precision: 0.977 -F1-score: 0.978 -TPR: 0.979	Not Available
			CNN	CSIC 2010	binary	-Precision: 0.986 -F1-score: 0.985 -TPR: 0.983	
			LSTM+ CNN	CSIC 2010	binary	-Precision: 0.989 -F1-score: 0.989 -TPR: 0.988	
PS34	[41]	-Malicious URLs -Malicious registry keys -Malicious file paths	CNN	Private dataset	multi-class	-TNR: 0.993 (URLs detection)	-Poor detection performance of some web attacks -Long URLs length induce computational cost overhead
						-TNR: 0.978 (file paths detection)	
						-TNR: 0.992 (registery keys detection)	
PS35	[43]	Web attacks hidden in network traffic	LSTM	CICIDS 2017	binary	-Accuracy: 0.9908 -TPR: 0.987 -TNR: 0.992	-Available datasets for web attacks detection are restricted -Reduce detection time by using GPU
				NSL-KDD	binary	-Accuracy: 0.9914 -TPR: 0.995 -TNR: 0.996	
				CICIDS 2017	multi-class	-Accuracy: 0.9910 -TPR: 0.994 -TNR: 0.993	
				NSL-KDD	multi-class	-Accuracy: 0.994 -TPR: 0.985 -TNR: 0.992	
PS36	[45]	Web attacks hidden in network traffic	Genetic algorithm and Shallow Neural Network (SNN)	CICIDS-2017	multi-class	-Accuracy: 0.9790 (web attacks detection) 0.9676 (web attacks detection) -TPR: 0.97 (web attacks detection)	Not Available
				CICIDS-2017	multi-class	-Accuracy: 0.9758 (normal traffic) -Precision: 0.9540 (normal traffic) -TPR: 0.96 (normal traffic)	

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS37	[47]	Web attacks hidden in HTTP requests	Stacked-AE + Isolation Forest (IF)	CSIC 2010	binary	-Accuracy: 0.8924 -Precision: 0.8158 -F1-score: 0.853 -TPR: 0.894 -TNR: 0.8911 -AUC: 0.96	-Extract effective features
			Deep Belief Network (DBN) + IF	CSIC 2010	binary	-Accuracy: 0.8687 -Precision: 0.8109 -F1-score: 0.813 -TPR: 0.815 -TNR: 0.89 -AUC: 0.94	-Port the WAF as cloud service
			Stacked-AE + Elliptic Envelop	ECML-PKDD	binary	-Accuracy: 0.8378 -Precision: 0.8240 -F1-score: 0.842 -TPR: 0.863 -TNR: 0.8117 -AUC: 0.92	-Extend the proposed method for big data environments and data streams
			DBN + Elliptic Envelop	ECML-PKDD	binary	-Accuracy:0.8413 -Precision: 0.8086 -F1-score:0.849 -TPR: 0.895 -TNR: 0.78 -AUC: 0.94	
PS38	[49]	-SQLI -XSS attacks	CNN	CSIC 2010	multi-class	-Accuracy: 0.998 -Precision: 1 -F1-score: 0.991 -TPR: 0.982 -FPR: 0	-The proposed system can only deal with SQLI and XSS attacks
			CNN	Private dataset	multi-class	-Accuracy: 0.999 -Precision: 1 -F1-score: 0.999 -TPR: 0.986 -FPR: 0	

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS39	[51]	Malicious JS code	Bi-directional LSTM	Private dataset	binary	-Accuracy: 0.9771 -Precision: 0.9868 -F1-score: 0.9829 -TPR: 0.979	-Static analysis cannot detect malicious JS code generated dynamically -Experiment more complex neural networks
PS40	[53]	-SQLI -RFI -XSS -DT	CNN	Private dataset	multi-class	-Precision: 1 (benign) -TPR: 0.99 (benign) FPR: 0.02 (benign)	-Test the model performance in more practical applications
						-Precision: 1 (Directory Traversal (DT)) -TPR:1 (DT)	
						-Precision: 0.9983 (Remote File Inclusion (RFI)) -TPR:1 (RFI)	
						-Precision: 0.9979 (SQLI) -TPR:1 (SQLI)	
						-Precision: 1 (XSS) -TPR:1 (XSS)	
PS41	[55]	XSS	DFFN	Private dataset	binary	-Accuracy: 0.9932 Precision: 0.9921 -F1-score: 0.987 -TPR: 0.98 -FPR: 0.31 -AUC: 0.99	-Deploy the proposed model in a real-time detection system
PS42	[57]	Web attacks injected in web logs and HTTP requests	CNN	Apache 2006	binary	-Accuracy: 0.9971 -Precision: 0.9964 -F1-score: 0.9921 -TPR: 0.9879	-Exploit model uncertainty in other security scenarios -Combine softmax output and the model uncertainty as a unified standard to evaluate the prediction confidence
				CSIC 2010	binary	-Accuracy: 0.9581 -Precision: 0.8612 -F1-score: 0.987 -TPR: 0.9291	
				Apache 2007	binary	-Accuracy: 0.9959 -Precision: 0.9878 -F1-score: 0.9931 -TPR: 0.9984	

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS43	[59]	XPath injection	LSTM	Private dataset	binary	-TPR: 0.84 -FPR: 0.16 -TNR: 0.83	-Use other techniques and algorithms to enhance accuracy while considering the important factor of response time
PS44	[61]	Web attacks hidden in URLs	SAE	Private dataset	binary	-Accuracy: 0.99	-Identify other types of web attacks that appear in user agent strings and cookies
			RNN	Private dataset	binary	-Accuracy: 0.93	
PS45	[63]	SQLI	CNN	Private dataset	binary	-Accuracy: 0.9993 -Precision: 0.9996 -F1-score: 0.99 -TPR: 0.99 -TNR: 0.99	-Implement a multi-classification model that is not limited to SQL injection attacks detection
PS46	[59]	SQLI	DFFN	Private dataset	binary	-Accuracy: 0.968 -TPR: 0.032	Not Available
PS47	[67]	SQLI	LSTM	Private dataset	binary	-Accuracy: 0.9535 -Precision: 0.9651 -TPR: 0.96	-Use real PHP applications to generate datasets
PS48	[69]	Web attacks hidden in HTTP requests	CNN	CSIC 2010	binary	-Accuracy: 0.988	Not Available
PS49	[71]	Web attacks hidden in HTTP requests	Encoder–Decoder	Private dataset	binary	-Precision: 0.9937 (average value) -F1-score: 0.993 (average value) -TPR: 0.99 (average value)	-Class imbalance problem -Poisoning attack
PS50	[73]	SQLI	DBN	Private dataset	binary	-Accuracy: 0.96	Not Available
PS51	[75]	-XSS -SQLI	MLP network as an ensemble classifer of: CNN, LSTM, and a variation of Residual Networks (ResNet)	CSIC 2010	multi-class	-Accuracy: 0.9947 -Precision: 0.9970 -TPR: 0.99 -FPR: 0.00	-Explore other DL models
				Private dataset	multi-class	-Accuracy: 0.9998 -Precision: 0.9997 -TPR: 1 -FPR: 0.04	
				Private dataset	multi-class	-Accuracy: 0.9917 -Precision: 0.9917 -TPR: 0.99 -FPR: 0.00	-Detect other web attacks

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS52	[77]	Web attacks hidden in network traffic	Tree CNN with Soft-Root-Sign (SRS) activation function	CICIDS 2017	multi-class	-Accuracy: 0.99	-Test other activation functions
			Tree CNN with SRS activation function	private dataset	multi-class	-Accuracy: 0.98 -Precision: 0.95 -F1-score: 0.97 -TPR: 0.97	-Existing datasets do not have characteristics of IoT devices
			Tree CNN with SRS activation function	private dataset	multi-class	-Accuracy: 0.99 -Precision: 0.98 -F1-score: 0.98 -TPR: 0.98	
PS53	[79]	Web attacks hidden in HTTP requests	CNN	CSIC 2010	binary	-Accuracy: 0.981 -Precision: 0.977 -F1-score: 0.962 -TPR: 0.97	-Time overhead in the training and testing phases
PS54	[81]	Web attacks hidden in HTTP requests	CNN with word-level embedding	CSIC 2010	binary	-Accuracy: 0.976	-Investigate new embedding approaches
			CNN with character-level embedding	CSIC 2010	binary	-Accuracy: 0.961	
PS55	[83]	DOM-XSS attack	DFFN	private dataset (Unconfirmed vulnerabilities)	binary	-Precision: 0.267 -TPR: 0.95	-Browser-specific vulnerabilities
PS56	[85]	Web attacks hidden in IoT networks	DBN	CICIDS 2017	multi-class	-Accuracy: 0.987 -Precision: 0.972 -F1-score: 0.97 -TPR: 0.98	-Detect other attacks against IoTs -Evaluate the model against other intrusion detection datasets
PS57	[86]	Web attacks	DFFN (8 input neurons)	private dataset	binary	-TPR: 0.92 -FPR: 0.07	-Consider every parameter in web pages -Achieve more coverage of users behavior
			DFFN (7 input neurons)	private dataset	binary	-TPR: 0.95 -FPR: 0.04	

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS58	[87]	Web attacks hidden in HTTP requests	CNN	CSIC 2010 (train/test split)	binary	-Accuracy: 0.9812 -Precision: 0.9483 -F1-score: 0.962 -TPR: 0.97	-ASCII code based conversion causes a time overhead in the training and testing phases
				CSIC 2010 (5-fold cross-validation)	binary	-Accuracy: 0.9824 -AUC: 0.97	
				CSIC 2010 (10-fold cross-validation)	binary	-Accuracy: 0.9820 -AUC: 0.97	
PS59	[88]	XSS hidden in PHP and JS code	Path Attention [96] (DFFN-based network with attention mechanism)	Private D1 (PHP included as code) and word2vec used for vectorization	binary	-Accuracy: 0.733 -Precision: 0.68 -F1-score: 0.70 -TPR: 0.735	-Created datasets consider synthetic data only
				Private D2 (PHP included as text) and word2vec used for vectorization	binary	-Accuracy: 0.707 -Precision: 0.724 -F1-score: 0.615 -TPR: 0.534	-The current code representation techniques do not take into account the invocation between different files: they only analyze single files
				Private D1 (JS included as code) and word2vec used for vectorization	binary	-Accuracy: 0.720 -Precision: 0.728 -F1-score: 0.676 -TPR: 0.631	
				Private D2 (PHP included as code) and code2vec used for vectorization	binary	-Accuracy:0.9538 -Precision: 0.9538 -F1-score: 0.918 -TPR: 0.999	-The current code representation techniques do not scale to large source code files
				Private D1 (JS included as code) and code2vec used for vectorization	binary	-Accuracy: 0.797 -Precision: 0.894 F1-score: 0.740 -TPR: 0.632	

Table 5. *Cont.*

ID	Paper	Targeted Web Attacks	Classification Model	Dataset	Classification Type	Performance Metrics	Limitations
PS60	[50]	Web attacks hidden in HTTP requests	CNN	CSIC 2010	binary	-Accuracy: 0.9684 -Precision: 0.9743 -F1-score: 0.9751 -TPR: 0.9759 -FPR: 0.0368 -TNR: 0.9631 -AUC: 0.9696	-Consider multi-classification technique using the proposed model
PS61	[89]	-Different web attacks injected in HTTP Web requests	Bi-LSTM	private dataset	binary	-Accuracy: 0.975 -Precision: 0.976 -F1-score: 0.975 -TPR: 0.975	-Training Bi-LSTM models consume time and computational resources
				ECML-PKDD	binary	-Accuracy: 0.995 -Precision: 0.994 -F1-score: 0.996 -TPR: 0.998	-The DA-SANA method does not consider the files uploaded as part of the web HTTP request length
				ECML-PKDD	binary	-Accuracy: 0.927 -Precision: 0.926 -F1-score: 0.923 -TPR: 0.927	-The DA-SANA method does not consider the files uploaded as part of the Web Http request length
				CSIC2010	multi-class	-Accuracy: 0.986 -Precision: 0.982 -F1-score: 0.981 -TPR: 0.984	
PS62	[90]	-Malicious HTTP requests -XSS -SQLI -DT	CNN	private	binary and multi-class	-Precision: 0.9333 -F1-score: 0.9340 -TPR: 0.9348	-Consider complex web attacks such as webshell -Detect encrypted malicious HTTP requests -Consider datasets with only normal samples (i.e., anomaly-based detection) -Consider non-textual elements of the HTTP request
PS63	[91]	Web attacks	CNN	private dataset	binary	-Accuracy: 0.9994 -Precision: 0.9995 -F1-score: 0.9993 -TPR: 0.992	-Implement a deception mechanism that analyzes the characteristics of detected web attacks
				CSIC2010	binary	-Accuracy: 0.987 -Precision: 0.994 -F1-score: 0.991 -TPR: 0.988	

Due to different performance measures and datasets, it is hard to compare and rank studies. However, we observed that most if not all reviewed studies achieved high-performance metrics, but few of them had discussed the threats to validity of their experiments (i.e., what are the things that may invalidate the results of the experiments).

4.9. RQ9 and RQ10: What Is the Research Focus and Limitations According to the PS?

In this section, we report the research focus as well as the limitations of the studies as stated by their authors. This part will help researchers and practitioners to have an idea of what is already done in previous research works and to develop more effective and improved detection models. We organize this section according to the Deep Learning classification models used in the reviewed studies. In the end, we discuss papers that used Deep Learning feature extraction methods for traditional Machine Learning algorithms.

4.9.1. CNN or CNN Combined with LSTM or GRU

Ref. [30] proposed a CNN-based method for detecting SQLI. They showed that the proposed model outperforms ModSecurity—a rule matching-based firewall for detecting web attacks.

Ref. [32] introduced a method for detecting malicious HTTP GET requests using a new architecture of CNNs for classification, and they used NLP-based analysis and Auto-Encoders for URL representation and extraction. However, the authors state that the proposed model can not be updated easily when new training data are available and can be defeated by adversarial attacks.

Ref. [38] described a method for detecting web attacks injected in web HTTP requests using word embedding and CNNs. The proposed method can not detect web attacks hidden in parts of the HTTP request message other than the URL.

Ref. [42] proposed a method for detecting web attacks using CNN and GRU along with word-level embedding-based features augmented with manually extracted features.

Ref. [44] provided a method for detecting XSS attacks using CNN, LSTM, and word-level embedding. They identify the problem of scarcity of datasets in the field of web security as a limitation of their study.

Ref. [48] worked on the detection of SQL injection attacks in PHP code. They compared different classification algorithms and feature representation techniques. They reported that the best algorithms are CNN and Multi-Layer Perceptron (MLP) applied to manually extracted features and TF-IDF bag of words model. As future work, they propose to collect a dataset for SQL injection attack detection and to build a Word2vec model for PHP source code as well as to develop an SQL injection attack detection model using CFG (Control Flow Graph) attributes.

Ref. [50] detected web attacks hidden in web HTTP requests using Bag of Words and CNN models. They plan to consider multi-class classification instead of binary classification in future works.

Ref. [56] introduced AI-IDS, which is a Deep Learning model for detecting three types of web attacks: password guessing and authentication bypass, SQL injection, and application vulnerability attack. The proposed model works in parallel with a signature-based NIDS to correct or improve its detection rules. Thus, after repeated manual re-validation and daily retraining, the model can be used as a standalone tool when it reaches an acceptable rate of false positives. The study limitations include the need to re-validate and retrain Deep Learning-based misuse intrusion detection tools due to the low tolerance for a high rate of false alarms. Moreover, their study shows that misuse detection tools can be used in parallel with signature-based tools to improve the detection rules of the latter and the detection quality of the former by checking the malicious events that are detected by one and not by the other.

Ref. [70] introduced an anomaly detection method of web attacks using character-level embedding and CNN followed by LSTM. They trained the model on a dataset that contains only two out of three attacks and then tested the model on the attacks that did not belong to

the training set in order to enhance the model's capability to detect unknown attacks. One significant issue of this work is that more web attacks need to be included in the dataset. In addition, scenario-based attacks through several correlated requests need to be considered, and the deployment of the model in a practical web service is to be tested.

Ref. [74] evaluated different Deep Learning models based on LSTM, CNN, and CNN combined with LSTM, for the detection of web attacks hidden in HTTP GET and HTTP POST requests. Moreover, they analyzed some false positives and found that they occur because either the HTTP requests contain strings that never appear in the training set or because of the way the request is split into a sequence of words. Therefore, they should inspect all the fields of the HTTP request and use more sophisticated models to convert request strings to numerical vectors.

Ref. [78] used character and keyword level embedding along with CNN without a pooling layer followed by GRU to detect web attacks through the classification of URLs into different categories: normal or type of attack. To improve their proposed model, they should consider reduced memory consumption and online update of trained models.

Ref. [39] implemented and compared three Deep Learning models for the detection of web attacks. First, they empirically showed that the classification performance of CNN combined with LSTM is better than that of LSTM or CNN. In addition, they showed how the dropout rate, the number and width of filters, the number of hidden units, and the size of local max-pooling influence the performance of CNN, LSTM, and CNN combined with LSTM models, respectively.

Ref. [41] used character-level embedding with CNN to extract relevant features from URLs, registry keys, and File paths. Then, they fed the output features to three fully connected layers to classify URL, registry keys, or File paths as normal or malicious. They showed that compared with other feature extraction methods, the proposed method has better classification performance but incurs a computational overhead if training long strings. They faced difficulties in collecting and labeling registry keys and file paths datasets, which resulted in poor classification performance in comparison with results obtained for URLs classification.

Ref. [51] proposed a CNN-based system for detecting web attacks. The system comprises two networks that are trained separately; the first one locates suspicious payloads in the HTTP request by identifying their start and end positions. They took the top three suspicious payloads returned by the first network and passed them to the second network, which identifies the attack type in each payload. If the three payloads are benign, the request is classified normal; otherwise, it is anomalous. Although the proposed system can only detect SQLI and XSS attacks, it reduces the computational cost by 82.6% and increases the detection accuracy by 22.3% compared with character-level CNN.

Ref. [55] proposed a multi-classification system for detecting malicious HTTP requests by identifying malicious HTTP parameters. The system is composed of a character-level embedding layer and a convolutional-pooling layer. Compared with SVM and Random Forest, the proposed system is better with respect to different performance metrics. In addition, the proposed model can be updated by retraining the model with new or rectified instances of HTTP requests. Finally, they should consider testing the performance of the model in more practical applications.

Ref. [63] proposed a novel CNN-based model for the detection of SQL injection attacks. The approach novelty consists of modifying the pooling layers so as to retain the maximum of information about the SQL query string. However, they plan to implement a multi-classification model that is not limited to identifying SQL injection attacks.

Ref. [69] proposed a Deep Learning-based approach for the detection of web attacks. They concatenated four models with the same architecture—a character-level embedding layer, two convolutional-pooling layers, and a dense layer. Then, they fed the four outputs to a dense layer. They demonstrated that the proposed model is more accurate and takes less time and memory resources than if only one model is used.

Ref. [77] proposed an IDS for detecting web attacks, DDoS, Infiltration, and Brute Force, based on the analysis of IoT networks traffic. The IDS consists of a trained Tree-CNN model that uses Soft-Root-Sign (SRS) activation function. They justified this choice by the fact that Tree-CNN shows better performance in other fields such as image classification and that the SRS activation function allows faster training and detection time. They compared the proposed model with other ML algorithms, Deep Belief Network, as well as Tree-CNN that uses other activation functions (RELU, Softmax). The results showed the out-performance of the proposed model. However, the evaluation of proposed detection models is limited by the scarcity of IoT-based datasets.

Ref. [79] developed a Deep Learning-based model for the detection of malicious HTTP requests. They used an ASCII code to convert the HTTP requests to a two-dimensional matrix and then fed it to a CNN network. They showed that the proposed model is more accurate than other state-of-the-art models. However, compared with word-level and character-level embedding, the ASCII code-based conversion causes a time overhead in the training and testing phases.

Ref. [81] aimed at finding the best hyper-parameters and the embedding approaches that should be advised for building CNN models that help to improve web attacks detection accuracy and to reduce detection time overhead. To this end, they built different CNN models using character-level and word-level embedding, different values of hyper-parameters (activation functions, kernel sizes, optimizers, number of layers, etc.), and validation methods. The comparison showed that word-level embedding, Relu function, Adam optimizer, two fully connected layers, 128 filters for each kernel size (2, 3, 4), and 10 fold cross-validation method bring the best detection accuracy with less time overhead. They plan to investigate new embedding approaches that could outperform word-level and character-level embedding.

Ref. [87] proposed a new method for representing HTTP web requests, which consists of substituting each character or symbol in the HTTP web request with its corresponding ASCII code. Then, they fed the resulting integer vector into a CNN network to classify HTTP web requests into benign and malicious. They compared the proposed method with word-level and character-level embedding and showed that it produces better classification results. They raise the problem of time overhead incurred by the new method at the training and testing phases.

In [37,57], the authors introduced a model uncertainty to web attacks detection that aims at finding annotations errors in web log datasets: they wrongly tagged some web logs of the dataset (normal web logs are tagged attacked and attacked web logs are tagged safe), and they included these web blogs in both the training and testing set. Then, they trained a CNN network followed by two fully connected layers on the resulting dataset to classify web logs as attacked or safe. Afterwards, they computed a Bayesian variance of classified web logs, which represents the model uncertainty; that is how the model is confident about its predictions. They showed that web logs with high variance are most likely to be wrongly labeled, which can help security experts to correct the dataset and retrain the model on more clean data. They compared the model proposed with Softmax (i.e., if the softmax output is 0.5, it means the model is not sure about the prediction), and they showed that in most cases, the model uncertainty finds more annotation errors because mislabeled inputs have the highest variances in most cases. As for the prediction time, the model uncertainty takes a long time compared with Softmax, but the time overhead is nearly imperceptible for the user. The authors plan to exploit model uncertainty in other security scenarios such as locating the adversarial web request samples and to combine softmax output and the model uncertainty as a unified standard to evaluate the prediction confidence.

In [50], the authors presented a method for classifying malicious HTTP requests based on the URL and payload. They used the Bag of Words technique to convert the textual input to a two-dimensional numerical matrix, which they input to a two-layer CNN. They evaluate the model using the CSIC2010 dataset, and they compare it with state-of-the-art

methods. The proposed approach achieved the best performance results, but it only handles binary classification.

In [90], the authors propose a novel approach for the detection of malicious HTTP requests in the particular case where the datasets available for the target system are limited in size and of low degree of diversity to build performant DL models. The approach consists of three phases. (i) In model initialization, a large public dataset is used to build a DL model that detects malicious HTTP requests. They used word2vec for the vectorization of HTTP payloads and TextCNN for their classification. (ii) In data augmentation, noise is added to the original samples of the targeted system-based dataset while keeping the keywords unchanged. This way, the samples get diversified but keep their semantic meaning. They used TF-IDF to define keywords. (iii) Third, in Transfer Semi-Supervised Learning, they froze the first n layers of the TextCNN model built at the first phase, and they trained the rest on the original and generated labeled and unlabeled targeted system-based datasets. This way, the obtained model has the knowledge learned by the initial model and adapted to the new target system without the risk of over-fitting due to the small size of the dataset. The results of the conducted experiments show that the proposed approach achieved better performance in comparison with other baseline models and methods such as Bi-LSTM, SVM, AE, character-level embedding, and N-gram. However, as future work, they plan to (i) include more complex web attacks, (ii) handle encrypted malicious HTTP requests, (iii) consider non-textual elements of the HTTP web requests, and (iv) build anomaly-based detection models.

4.9.2. Recurrent Neural Networks

Ref. [60] introduced an anomaly detection method that used (i) word-level embedding to represent URLs, (ii) two separate GRU or LSTM networks to predict the next token in the URLs path or query parameter given a set of previous tokens, and finally, (iii) an MLP to predict if a URL is normal or anomalous based on the probabilities vectors returned by the GRU or LSTM networks. However, the proposed model can not handle some kind of long URLs properly and also can not dynamically leverage between true positives and false positives.

Ref. [62] proposed an anomaly detection method of web attacks using the LSTM network and lexical and statistical features extracted manually from HTTP web requests. They showed that compared with other methods where selecting a subset of extracted features was necessary to increase the classification model accuracy, the LSTM model was able to achieve high accuracy without feature selection.

Ref. [64] suggested APPMINE, an anomaly-based method for web vulnerabilities detection which uses an LSTM network and web application system calls as features. However, APPMINE could miss certain classes of attacks not manifested in the system call sequences generated by the web application. Moreover, APPMINE has other issues, including privacy leakage: monitoring a system to collect application system calls could impact users' privacy. In addition, the training of APPMINE should be done for each web application to detect anomalies specific to that web application. Finally, APPMINE is also prone to adversarial attacks.

Ref. [68] presented an anomaly detection method of web attacks using a word-level embedding for feature extraction and bi-directional LSTM for the classification of malicious URLs.

Ref. [72] proposed a multi-classification model that can detect six categories of web attacks hidden in URLs using one-hot encoding and GRU networks. Furthermore, they showed that GRU networks outperform Random Forest in terms of accuracy even when small training sets are used.

Ref. [76] proposed an anomaly detection method to detect XSS attacks using LSTM and word-level embedding. However, they need more XSS-oriented datasets to assess the model.

Ref. [33] implemented and compared two Deep Learning models, namely MLP and LSTM, for the detection of SQL injection attacks. They used statistical features of URLs as input to the MLP model, whereas they used lexical features as input to the LSTM. They found that MLP is better than LSTM both in classification performance and time. They used an external dataset different from the training and test set to evaluate the capability of the models to detect unknown attacks.

Ref. [43] filtered out known attacks using a signature-based intrusion detection system. Afterward, they used a trained two-layer LSTM model as an anomaly-based intrusion detection system to detect attacks that the misuse detection system could not identify. Then, a signature generation module is called to update the signature repository with new signatures extracted from the new attacks detected by the anomaly-based IDS. They compared the proposed model with other traditional Machine Learning algorithms, and they showed that it is better in every performance metric.

Ref. [53] proposed a Deep Learning-based approach for the detection of malicious JavaScript programs. The pre-processing phase is based essentially on a static analysis of JavaScript code that consists of six main steps—(i) JavaScript de-obfuscation, (ii) Abstract Syntax Tree (AST) generation, (iii) Program Dependency Graph (PDG) generation, (iv) Program slices generation, (v) Tokenization and generalization of program slices, (vi) Vectorization—and outputs an 80-dimensional vector. Then, they applied a two-layer Bi-LSTM on the output vector to classify JavaScript codes as malicious or benign. Compared with other Machine Learning algorithms and signature-based open-source antivirus tools, the proposed model achieves the best detection performance provided that the JavaScript code is de-obfuscated. The limitation of this study is induced by the limitation of static analysis which does not allow to detect malicious JavaScript code that is generated dynamically. In future work, they will explore other types of Neural Networks such as tree and graph structure neural networks.

Ref. [59] proposed a modular neural network for the detection of XPath injection attacks. The model comprises two LSTM networks; the first one classifies login attempts as valid or malicious. The second network classifies user input as valid, invalid, or malicious. The final decision about the user request is made based on the classification output of the two LSTM models. If the user request is classified as malicious, fake data are returned in place of real data. If the user request is classified as invalid, a message error is returned; otherwise, the user request is processed by the web server.

Ref. [67] proposed a Deep Learning approach for the detection of SQLI vulnerability in PHP codes. They used a tool that captures the opcode of PHP code before it is executed. Then, they converted the opcode slice to an integer vector that they pass to an embedding layer to obtain a matrix that they fed to an LSTM layer followed by two dense layers. The issue in that study is training and testing sets are generated from the same dataset. Thus, it is possible that the selected dataset may not reflect real PHP applications.

4.9.3. Encoder–Decoder Models

Ref. [66] proposed an anomaly detection method of web attacks based on a GRU-Encoder–Decoder model with an attention mechanism. They justify the use of the attention mechanism with the fact that malicious strings may occur in different locations of the HTTP request payload. Therefore, it is essential to consider all hidden states instead of the last hidden state only. In addition, the attention mechanism allows verifying if the model is well-trained thanks to the visualization techniques that can be built using the attention weights.

Ref. [84] proposed an anomaly-based detection method of web attacks. They collected traces of normal behavior of java web applications and trained a Stacked Denoising Auto-Encoder to distinguish normal requests from malicious requests based on execution traces. In the test phase, the trained model takes as input a trace vector and tries to reconstruct it; if the reconstruction error is more significant than a certain threshold, then the web request is classified as abnormal; otherwise, it is normal. The threshold is defined during the training

phase so that the F1-score remains maximal. However, they should investigate more complex networks, such as LSTM Auto-Encoder or CNN Auto-Encoder, and evaluate the performance of the proposed model on zero-day attacks. Moreover, they should consider updating trained models using online data from actual world usage without incorporating attack data into the normal behavior dataset.

Ref. [31] trained a Stacked Auto-Encoder with Logistic Regression classifier to distinguish DDOS attacks from normal web requests based on eight features representative of DDOS attacks. The Stacked Auto-Encoder is intended to extract an abstract representation of features, while Logistic Regression is used for binary classification.

Ref. [61] implemented two Deep Learning models to detect web attacks hidden in URLs. The first one is the Stacked Auto-Encoder model, and the second is an RNN. In addition, they should consider identifying other types of web attacks that appear in URLs and user-agent strings and cookies.

Ref. [71] proposed a zero-wall system for the detection of web attacks. It operates behind a signature-based Web Application Firewall (WAF). The WAF drops known attacks, and the zero-wall systems intercept allowed web requests and classifies them as benign or malicious. If it is malicious, then the web requests are analyzed by security experts to ensure it is not a false positive. If it is a false positive, then a rule set is added to a white list so it is not rejected again by the zero-wall system. If it is a true positive, then a rule set is added to the WAF signature database. The zero-wall system uses an LSTM-based Encoder–Decoder model to distinguish between benign and malicious web requests. Indeed, if it fails to reconstruct the input token sequence corresponding to a web request, then the latter is considered malicious. To speed up the detection time taken by the zero-wall system, they used hash tables to not process web requests that have already been seen and classified by the system. However, the system has a few issues, including class imbalance problem—too small volume of training data, and poisoning attack—an attacker may inject many malicious samples in the normal traffic hoping the system would learn from the wrong dataset.

4.9.4. Deep Belief Networks

Ref. [73] used a deep belief network along with word2vec and statistical features to detect SQL injection attacks hidden in GET or POST web requests.

Ref. [85] developed a Deep Belief Network-based IDS for the detection of different types of attacks including web attacks in IoT systems. They showed that the proposed model outperforms different Machine Learning algorithms such as RNN and SVM. They plan to improve their work by detecting other attacks against IoTs and by evaluating the proposed model using other datasets.

4.9.5. Ensemble Classification Models

Ref. [40] focused on detecting web attacks in HTTP requests sent by edge devices to the cloud using a weighted average ensemble model composed of two Residual Networks (ResNet) that use different feature representations of URLs. As with previous studies, the online updating of detection models and adversarial attacks are two problems that are not addressed by this work.

Ref. [54] implemented a detection method of web attacks in heterogeneous and adversarial environments using an adaptive IDS, which is an ensemble classification model that aims at tracking the best classification model for each data instance. As future work, they plan to apply the proposed method to malware and network intrusion detection and compare it with other ensemble classification algorithms.

Ref. [75] proposed a web attacks detection system for IoT networks, which consists of an ensemble classifier that identifies malicious URLs based on the predictions obtained from three sub-models—CNN, LSTM, and ResNet—and an update module that fine-tunes each sub-model in order to detect novel web attacks. Moreover, they showed that the proposed system outperforms each sub-model as well as other state-of-the-art classification

models. However, they plan to explore other alternative Deep Learning models to improve the performance of the overall system as well as to detect web attacks other than XSS attacks and SQL injection attacks.

In [89], the authors proposed a data-augmentation method based on self-adapting noise adding, which consists of adding noisy data to web attacks datasets in proportion with the web HTTP request length. The method aims at overcoming the problem of classification in unbalanced datasets. They evaluated the method by implementing different state-of-the-art DL models and comparing their classification performance with and without data augmentation on public datasets as well as on a synthetic dataset for both binary classification and multi-classification problems. The results showed that the Bi-LSTM model with the DA-SANA method has the best classification performance values. The method presents the limitations of time cost and computational resources required to train Bi-LSTM models. In addition, the DA-SANA method does not consider the files uploaded as part of the web HTTP request length.

In [91], the authors propose a detection, mitigation, and attacker profiling system for securing web applications against web attacks. The system is composed of a Cookie Analysis Engine (CAE) and a DL classifier. First, the HTTP request is checked for forged cookies. Then, it is forwarded to the DL classifier. The decision of whether the request is normal or malicious is based on the combined outcome of the CAE and the DL classifier. Furthermore, if a user is profiled as suspicious or attacker, the subsequent requests will be blocked without calling neither the CAE nor the DL classifier. Then, the attacker's profiling option aims at reducing the processing time, which makes the proposed system suitable for real-time environments. A state-of-the art comparison showed that the proposed system has the best performance results on the private dataset as well as on the CSIC2010 dataset and in the real-time environment. They also mentioned that most web attacks are injected in the HTTP payloads rather than in the cookies. They plan to join the proposed system with a deception mechanism that is intended to analyze the characteristics of the attacks and the tactics of attackers.

4.9.6. Deep Feed Forward Networks

Ref. [36] worked on a method for the detection of malicious HTML pages using DFFNs and a custom representation of HTML pages. One limitation of the study is that the proposed model may not be able to detect malicious web content that experts mislabeled as normal in the training set.

Ref. [46] compared Machine Learning-based web attacks detection methods.

Ref. [52] detected two types of web attacks (SQL injection and distributed denial of service) in network packets using statistical methods for labeling web packets and supervised Machine Learning algorithms for building predictive models. However, the study has a few limitations, including variable size network and memory and time constraints, which constitute an obstacle for real-time web intrusion detection.

Ref. [82] implemented and compared different Machine Learning algorithms (Random Forest, Decision Tree, AdaBoost, Deep NN, etc.) for the detection of SQL injection attacks. They used manually extracted features that are relevant to the domain of SQL injection attacks. They empirically showed that Random Forest outperforms all the other algorithms.

Ref. [35] trained a simple MLP to detect SQLI attacks by converting each URL to a binary vector whose length is 13, which is the number of the most popular SQLI attack keywords and patterns, and it can be extended to include other SQLI keywords.

Ref. [55] proposed a Neural Network-based method for the detection of XSS attacks that can be used on both the client side or server side. They extracted 41 numerical features including URL, JavaScript, and HTML-based aspects that characterize XSS attacks, and they fed them to a two-layer neural network. They compared the proposed method with other Machine Learning algorithms and found that it is more performant with respect to different aspects of evaluation measurements. They also highlighted that traditional Machine Learning algorithms have also good classification performance, which they explained

by the quality and method of collecting the training dataset and the strategy used for feature extraction.

Ref. [59] proposed a multi-classification model that classifies each URL as benign or an SQLI attack in which case it specifies its type. First, they extracted 32 keywords that characterize different types of SQLI attacks. Then, they represented each URL with a binary vector that indicates the presence (1) or absence (0) of the keyword. Finally, each URL is assigned a class between eight classes—benign, tautologies, illegal/logically incorrect queries, piggy-backed query, union queries, stored procedures, inference SQLI attack, and alternate encoding—by using a trained MLP network.

Ref. [83] developed a three-layer Deep Neural Network model to identify functions in JavaScript code that are vulnerable to DOM-XSS attack. They used the classification model as a pre-filter to taint tracking, which is a dynamic analysis method for detecting DOM-XSS vulnerabilities. They experimentally showed that the combination of these two approaches improves precision and recall while decreasing time overhead. They identify two limitations to their study: the dataset used to evaluate the proposed model may not apply to other browsers and may contain mislabeled instances (i.e., noisy ground truth).

Ref. [86] developed an artificial neural network for identifying web attacks that were not detected at the stage of signature-based analysis. They defined custom features to identify user behavior and to decide whether it is normal or not. They plan to extend the proposed system to take into account every parameter in web pages in order to achieve more coverage of the user's behavior.

Ref. [88] proposed a DL approach for detecting XSS attacks in PHP and JavaScript source code. They used two representation techniques to transform source code files into numerical vectors: word2vec and code2vec ([96]), which extract features from the source code Abstract Syntax Tree. They implemented a DFFN with an attention mechanism to classify the source code as vulnerable or not to XSS attacks. The proposed approach outperforms existing static analysis tools in every classification performance metric. However, it also has different limitations that undermine its applicability to real-world scenarios. Indeed, the model is evaluated using a synthetic dataset. In addition, the code2vec method does not scale to large source code files, and both representation techniques do not take into account invocations between different source code files.

4.9.7. Deep Learning-Based Feature Extraction

Ref. [34] represented a static analysis method for detecting malicious JS code by using a Stacked Denoising Auto-Encoder for feature extraction and logistic regression for classification. However, the limitations of the study include the need for minification and obfuscation of JS code and the long time it takes for training the proposed model.

Ref. [58] used n-gram and Stacked Auto-Encoder for feature construction and extraction, as well as Isolation Forest for classification to build an anomaly web attacks detection method.

Ref. [80] used word-level embedding to convert URLs to numerical matrices. Then, a Convolutional-Pooling layer is used to extract features from the embedded matrices. Afterward, the extracted features are merged with statistical features before they are fed into an SVM classifier. They built a function that takes as input the output of the max-pooling layer and outputs the corresponding words in the input URL, which may help researchers validate the trained model and understand which words contribute to a specific URL classification. Moreover, they showed the impact of testing a model on a dataset containing duplicated data: the proposed model accuracy goes from 99.33 to 100% when tested on the CSIC 2010 dataset—which contains a repetition rate of 81%—without de-duplication.

Ref. [47] used genetic algorithms to distinguish normal network traffic from abnormal network traffic. Then, they used a shallow Neural Network to classify attack types: either web attacks, infiltration attacks, PortScan, BruteForce, DoS, or Botnet attacks. Finally, they

showed that the proposed model is better than other Machine Learning techniques in terms of accuracy, detection rate, recall, and precision.

Ref. [49] exploited different Deep Learning models, namely Stacked Auto-Encoder and Deep Belief Network, for feature extraction and used one-class learning algorithms, namely Isolation Forest, one-class SVM, and Elliptic Envelop, for web requests classification.

5. Limitations

This study aims at conducting an SLR of DL-based web attacks detection research works. The lack of comparison between selected studies is the primary limitation of this study. Although we reported the results of selected studies, we could not compare their performance because they used different types of datasets and performance measures. In the searching phase, we collected studies from the most extensively utilized six digital libraries (Scopus, Web Of Science, ScienceDirect, IEEE, ACM, and Springer). Still, some resources might be left out. Thus, it is hard to declare that our search strategy covered all relevant studies. Moreover, our selection was conducted in two phases: firstly, we selected title and abstract-based studies, and secondly, the selection was based on the full text of each study. However, there is still a possibility that a suitable study might be excluded mistakenly. We would also like to mention that we genuinely reported the experiment results of the reviewed studies, but we did not perform their models and we did not reproduce their experimentation.

6. Conclusions

Web applications are prone to many security threats. Therefore, many techniques have been proposed to detect and prevent web attacks. This study represents an SLR of the existing DL-based web attacks detection research papers. Initially, we searched and selected journals and conference papers focused on DL-based detection of web vulnerabilities that have been published from 2010 until September 2021. Afterward, we selected relevant studies based on the title, abstract, and content. After the selection phase, we obtained 63 Primary Studies (PS), on which we applied a non-eliminatory quality analysis. We studied the selected PS from different perspectives and synthesized the results of their studies using different synthesis and visualization techniques.

Based on this analysis, we have learned different lessons from which we have inferred interesting research opportunities for future work in the DL-based web attacks detection domain:

- Generate standard public real-world datasets: it is important to generate web attacks detection datasets to resolve the current datasets issues. Indeed, most researchers used private datasets. Additionally, public datasets do not reflect the complexity of real-world web applications and do not include newly discovered web attacks. Moreover, when the same public dataset is adopted, different portions are used for training and testing, which complicates the comparison between approaches. Therefore, there is a need for standard realistic public datasets to allow the research communities to contribute efficiently in this field, to facilitate comparative analyses between research works, and to make the proposed models applicable in real-world web applications.
- Consider the standard classification performance metrics: some studies used a single performance metric (e.g., Accuracy) to evaluate their detection model, which is not sufficient in the case of imbalanced datasets. Moreover, most reviewed studies do not consider computational overhead as an evaluation metric, although it is a very critical issue in real-world web applications. Moreover, few researchers provided a detailed report of false alarms (FPR), which is an important point because it helps in model retraining while saving analysts time. In addition, most studies reported their results, but none of them outlined the threats to the validity of the experiments. Therefore, it is important to identify standard performance metrics that researchers should consider in order to ensure that the proposed models are accurate, cost-effective, reliable, and reproducible.

- Explore advanced DL models in the field of web attacks detection: the existing DL-based web attacks detection literature lacked some advanced DL models. In particular, applying Generative Adversarial Networks (GAN) and Encoders–Decoders to web attacks detection is interesting because they have been successfully exploited in a similar domain that is Networks Intrusion Detection (e.g., Refs. [92–95]).
- Bridge the expertise in web application security and expertise in Machine Learning in order to build theoretical Machine Learning models tailored for web attacks detection: most existing DL models were designed with other applications in mind. For instance, Convolutional Neural Networks and Recurrent Neural Networks were originally developed to answer the specific requirements of image processing and Natural Language Processing problems, respectively. Additionally, because each web application has its own business logic, it is interesting to have a theoretical Deep Learning model for detecting web attacks without a need to learn each web application separately.
- Support secure learning: Machine Learning models are prone to adversarial attacks. In such attacks, attackers evade intrusion detection systems by exploiting the underlying Machine Learning model. For instance, an adversarial attack can disrupt the model training by contaminating training data with malicious data, thereby tricking the detection model into misclassifying malicious web requests as benign.
- Support online learning: almost all reviewed studies adopted offline learning in building their detection models. In offline and online learning, the model is trained using batch algorithms (i.e., the cost function is computed over a group of instances). However, while in offline learning, the model is tested and validated using batch algorithms, in online learning, the model is tested using real-time data; thereby, the cost function is re-valuated over a single data instance at a time. In industry, models trained online are preferred over models trained offline because the latter are generally evaluated against outdated and simplistic datasets. Therefore, it is important that researchers give more consideration to online learning in order to reduce the gap between research and industry in the DL-based web attacks detection area.
- Support adaptive incremental learning: according to a recent study conducted in 2020 [97], 42% of attacks are zero-day attacks (i.e., new or unknown attacks), while 58% are based on known vulnerabilities. As with traditional detection systems, Machine Learning-based approaches have a difficult time detecting zero-day attacks because they rely on past and known attacks. Thus, it is fundamental to constantly retrain ML models to account for these attacks. However, retraining models from scratch is time-consuming and computationally intensive. Thus, incremental learning is essential, as it will allow updating trained models as new data is generated.
- Generate a corpus for web attacks detection: although DL-based web attacks detection problems are similar to Natural Language Processing (NLP) problems, there exists no corpus yet that can help to develop NLP-like models for web attacks detection problems.
- Define a standard and transparent research methodology: it is important to define a transparent research protocol that researchers should follow when proposing a DL-based method for the detection of web attacks. Such a methodology will improve the quality of research works and facilitate their comparison.
- Develop a common framework for comparing DL-based web attacks detection models: there is a large diversity in performance measures, datasets, and platforms used in reviewed studies, which makes a comparative analysis between research works difficult if not impossible. Therefore, it is fundamental to provide a standardization of datasets, performance metrics, environments, as well as a transparent research methodology that allows comparing the different approaches and evaluating the models' suitability for real-world web applications.

Author Contributions: R.L.A. and E.H.N. contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Technologies, P. Web Applications Vulnerabilities and Threats: Statistics for 2019. 2020. Available online: https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020/ (accessed on 20 February 2022).
2. Noman, M.; Iqbal, M.; Manzoor, A. A Survey on Detection and Prevention of Web Vulnerabilities. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 521–540. [CrossRef]
3. ASVS. Application Security Verification Standard. Available online: https://www.owasp.org/index.php/ASVS (accessed on 20 February 2022).
4. SAMMS. OWASP Software Assurance Maturity Model. Available online: https://www.owasp.org/index.php/SAMM (accessed on 20 February 2022).
5. Jovanovic, N.; Kruegel, C.; Kirda, E. Pixy: A static analysis tool for detecting web application vulnerabilities. In Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06), Berkeley/Oakland, CA, USA, 21–24 May 2006.
6. Medeiros, I.; Neves, N.; Correia, M. Detecting and removing web application vulnerabilities with static analysis and data mining. *IEEE Trans. Reliab.* **2015**, *65*, 54–69. [CrossRef]
7. Sun, F.; Xu, L.; Su, Z. Static Detection of Access Control Vulnerabilities in Web Applications. Available online: https://www.usenix.org/event/sec11/tech/full_papers/Sun.pdf (accessed on 20 February 2022).
8. Medeiros, I.; Neves, N.; Correia, M. DEKANT: A static analysis tool that learns to detect web application vulnerabilities. In Proceedings of the 25th International Symposium on Software Testing and Analysis, Saarbrücken, Germany, 18–22 July 2016; pp. 1–11.
9. Agosta, G.; Barenghi, A.; Parata, A.; Pelosi, G. Automated security analysis of dynamic web applications through symbolic code execution. In Proceedings of the 2012 Ninth International Conference on Information Technology-New Generations, Las Vegas, NV, USA, 16–18 April 2012; pp. 189–194.
10. Falana, O.J.; Ebo, I.O.; Tinubu, C.O.; Adejimi, O.A.; Ntuk, A. Detection of Cross-Site Scripting Attacks using Dynamic Analysis and Fuzzy Inference System. In Proceedings of the 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), Ayobo, Nigeria, 18–21 March 2020; pp. 1–6.
11. Wang, R.; Xu, G.; Zeng, X.; Li, X.; Feng, Z. TT-XSS: A novel taint tracking based dynamic detection framework for DOM Cross-Site Scripting. *J. Parallel Distrib. Comput.* **2018**, *118*, 100–106. [CrossRef]
12. Weissbacher, M.; Robertson, W.; Kirda, E.; Kruegel, C.; Vigna, G. Zigzag: Automatically Hardening Web Applications against Client-Side Validation Vulnerabilities. Available online: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-weissbacher.pdf (accessed on 20 February 2022).
13. Ruse, M.E.; Basu, S. Detecting cross-site scripting vulnerability using concolic testing. In Proceedings of the 2013 10th International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 15–17 April 2013; pp. 633–638.
14. Mouzarani, M.; Sadeghiyan, B.; Zolfaghari, M. Detecting injection vulnerabilities in executable codes with concolic execution. In Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 24–26 November 2017; pp. 50–57.
15. Duchene, F.; Rawat, S.; Richier, J.L.; Groz, R. KameleonFuzz: Evolutionary fuzzing for black-box XSS detection. In Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, San Antonio, TX, USA, 3–5 March 2014; pp. 37–48.
16. Deepa, G.; Thilagam, P.S.; Khan, F.A.; Praseed, A.; Pais, A.R.; Palsetia, N. Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications. *Int. J. Inf. Secur.* **2018**, *17*, 105–120. [CrossRef]
17. Pellegrino, G.; Balzarotti, D. Toward Black-Box Detection of Logic Flaws in Web Applications. Available online: https://s3.eurecom.fr/docs/ndss14_pellegrino.pdf (accessed on 20 February 2022).
18. Duchene, F.; Groz, R.; Rawat, S.; Richier, J.L. XSS vulnerability detection using model inference assisted evolutionary fuzzing. In Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, Montreal, QC, Canada, 17–21 April 2012; pp. 815–817.
19. Khalid, M.N.; Farooq, H.; Iqbal, M.; Alam, M.T.; Rasheed, K. Predicting web vulnerabilities in web applications based on machine learning. In Proceedings of the International Conference on Intelligent Technologies and Applications, Bahawalpur, Pakistan, 23–25 October 2018.
20. Anbiya, D.R.; Purwarianti, A.; Asnar, Y. Vulnerability Detection in PHP Web Application Using Lexical Analysis Approach with Machine Learning. In Proceedings of the 2018 5th International Conference on Data and Software Engineering (ICoDSE), Mataram, Indonesia, 7–8 November 2018; pp. 1–6.
21. Abunadi, I.; Alenezi, M. An Empirical Investigation of Security Vulnerabilities within Web Applications. *J. Univers. Comput. Sci.* **2016**, *22*, 537–551.

22. Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A survey of deep learning methods for cyber security. *Information* **2019**, *10*, 122. [CrossRef]
23. Torres, J.M.; Comesaña, C.I.; Garcia-Nieto, P.J. Machine learning techniques applied to cybersecurity. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 2823–2836. [CrossRef]
24. Sharma, A.; Singh, A.; Sharma, N.; Kaushik, I.; Bhushan, B. Security countermeasures in web based application. In Proceedings of the 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kannur, India, 5–6 July 2019; Volume 1, pp. 1236–1241.
25. Fredj, O.B.; Cheikhrouhou, O.; Krichen, M.; Hamam, H.; Derhab, A. An OWASP top ten driven survey on web application protection methods. In Proceedings of the International Conference on Risks and Security of Internet and Systems, Paris, France, 4–6 November 2020.
26. Mouli, V.R.; Jevitha, K. Web services attacks and security-a systematic literature review. *Procedia Comput. Sci.* **2016**, *93*, 870–877. [CrossRef]
27. Kaur, J.; Garg, U. A Detailed Survey on Recent XSS Web-Attacks Machine Learning Detection Techniques. In Proceedings of the 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 1–3 October 2021; pp. 1–6.
28. Kitchenham, B.; Charters, S.M. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Available online: https://www.researchgate.net/profile/Barbara-Kitchenham/publication/302924724_Guidelines_for_performing_Systematic_Literature_Reviews_in_Software_Engineering/links/61712932766c4a211c03a6f7/Guidelines-for-performing-Systematic-Literature-Reviews-in-Software-Engineering.pdf (accessed on 20 February 2022).
29. Kitchenham, B. Procedures for performing systematic reviews. *Keele UK Keele Univ.* **2004**, *33*, 1–26.
30. Luo, A.; Huang, W.; Fan, W. A CNN-based Approach to the Detection of SQL Injection Attacks. In Proceedings of the 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), Beijing, China, 17–19 June 2019; pp. 320–324. [CrossRef]
31. Yadav, S.; Subramanian, S. Detection of Application Layer DDoS attack by feature learning using Stacked AutoEncoder. In Proceedings of the 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), Beijing, China, 17–19 June 2019; pp. 361–366. [CrossRef]
32. Luo, C.; Su, S.; Sun, Y.; Tan, Q.; Han, M.; Tian, Z. A convolution-based system for malicious URLS detection. *Comput. Mater. Contin.* **2020**, *62*, 399–411. [CrossRef]
33. Tang, P.; Qiu, W.; Huang, Z.; Lian, H.; Liu, G. Detection of SQL injection based on artificial neural network. *Knowl.-Based Syst.* **2020**, *190*, 105528. [CrossRef]
34. Wang, Y.; Cai, W.D.; Wei, P.C. A deep learning approach for detecting malicious JavaScript code. *Secur. Commun. Netw.* **2016**, *9*, 1520–1534. [CrossRef]
35. Sheykhkanloo, N. Employing Neural Networks for the detection of SQL injection attack. In Proceedings of the 7th International Conference on Security of Information and Networks, Glasgow, Scotland, UK, 9–11 September 2014; pp. 318–323. [CrossRef]
36. Saxe, J.; Harang, R.; Wild, C.; Sanders, H. A deep learning approach to fast, format-agnostic detection of malicious web content. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24 May 2018; pp. 8–14. [CrossRef]
37. Gong, X.; Zhou, Y.; Bi, Y.; He, M.; Sheng, S.; Qiu, H.; He, R.; Lu, J. Estimating Web Attack Detection via Model Uncertainty from Inaccurate Annotation. In Proceedings of the 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), Paris, France, 21–23 June 2019; pp. 53–58. [CrossRef]
38. Zhang, M.; Xu, B.; Bai, S.; Lu, S.; Lin, Z. A Deep Learning Method to Detect Web Attacks Using a Specially Designed CNN. In Proceedings of the 24th International Conference on Neural Information Processing (ICONIP), Guangzhou, China, 14–18 November 2017; pp. 828–836. [CrossRef]
39. Wang, J.; Zhou, Z.; Chen, J. Evaluating CNN and LSTM for web attack detection. In Proceedings of the 10th International Conference on Machine Learning and Computing, Macau, China, 26–28 February 2018; pp. 283–287. [CrossRef]
40. Tian, Z.; Luo, C.; Qiu, J.; Du, X.; Guizani, M. A Distributed Deep Learning System for Web Attack Detection on Edge Devices. *IEEE Trans. Ind. Inform.* **2020**, *16*, 1963–1971. [CrossRef]
41. Saxe, J.; Berlin, K. eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. *arXiv* **2017**, arXiv:1702.08568.
42. Niu, Q.; Li, X. A High-performance Web Attack Detection Method based on CNN-GRU Model. In Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 12–14 June 2020; pp. 804–808. [CrossRef]
43. Kaur, S.; Singh, M. Hybrid intrusion detection and signature generation using Deep Recurrent Neural Networks. *Neural Comput. Appl.* **2020**, *32*, 7859–7877. [CrossRef]
44. Kadhim, R.; Gaata, M. A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack. *Indones. J. Electr. Eng. Comput. Sci.* **2020**, *21*, 1022–1029. [CrossRef]
45. Manimurugan, S.; Manimegalai, P.; Valsalan, P.; Krishnadas, J.; Narmatha, C. Intrusion detection in cloud environment using hybrid genetic algorithm and back propagation neural network. *Int. J. Commun. Syst.* **2020**. [CrossRef]
46. Smitha, R.; Hareesha, K.; Kundapur, P. A machine learning approach for web intrusion detection: MAMLS perspective. *Adv. Intell. Syst. Comput.* **2019**, *900*, 119–133. [CrossRef]

47. Moradi Vartouni, A.; Teshnehlab, M.; Sedighian Kashi, S. Leveraging deep neural networks for anomaly-based web application firewall. *IET Inf. Secur.* **2019**, *13*, 352–361. [CrossRef]
48. Zhang, K. A machine learning based approach to identify SQL injection vulnerabilities. In Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 11–15 November 2019; pp. 1286–1288. [CrossRef]
49. Liu, T.; Qi, Y.; Shi, L.; Yan, J. Locate-then-DetecT: Real-time web attack detection via attention-based deep neural networks. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macao, China, 10–16 August 2019; pp. 4725–4731. [CrossRef]
50. Tekerek, A. A novel architecture for web-based attack detection using convolutional neural network. *Comput. Secur.* **2021**, *100*, 102096. [CrossRef]
51. Song, X.; Chen, C.; Cui, B.; Fu, J. Malicious javascript detection based on bidirectional LSTM model. *Appl. Sci.* **2020**, *10*, 3440. [CrossRef]
52. Arshad, M.; Hussain, M. A real-time LAN/WAN and web attack prediction framework using hybrid machine learning model. *Int. J. Eng. Technol. (UAE)* **2018**, *7*, 1128–1136. [CrossRef]
53. Rong, W.; Zhang, B.; Lv, X. Malicious Web Request Detection Using Character-Level CNN. In *Machine Learning for Cyber Security*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 6–16. [CrossRef]
54. Nguyen, H.; Franke, K. Adaptive Intrusion Detection System via online machine learning. In Proceedings of the 2012 12th International Conference on Hybrid Intelligent Systems (HIS), Pune, India, 4–7 December 2012; pp. 271–277. [CrossRef]
55. Mokbal, F.; Dan, W.; Imran, A.; Jiuchuan, L.; Akhtar, F.; Xiaoxi, W. MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique. *IEEE Access* **2019**, *7*, 100567–100580. [CrossRef]
56. Kim, A.; Park, M.; Lee, D.H. AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection. *IEEE Access* **2020**, *8*, 70245–70261. [CrossRef]
57. Gong, X.; Lu, J.; Zhou, Y.; Qiu, H.; He, R. Model Uncertainty Based Annotation Error Fixing for Web Attack Detection. *J. Signal Process. Syst.* **2020**. [CrossRef]
58. Vartouni, A.; Kashi, S.; Teshnehlab, M. An anomaly detection method to detect web attacks using Stacked Auto-Encoder. In Proceedings of the 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Kerman, Iran, 28 Februay–2 March 2018; pp. 131–134. [CrossRef]
59. Deshpande, G.; Kulkarni, S. Modeling and Mitigation of XPath Injection Attacks for Web Services Using Modular Neural Networks. In Proceedings of the 5th International Conference on Advanced Computing, Networking, and Informatics (ICACNI), Goa, India, 1–3 June 2017. [CrossRef]
60. Liang, J.; Zhao, W.; Ye, W. Anomaly-based web attack detection: A deep learning approach. In Proceedings of the 2017 VI International Conference on Network, Communication and Computing, Kunming, China, 8–10 December 2017; pp. 80–85. [CrossRef]
61. Jin, X.; Cui, B.; Yang, J.; Cheng, Z. Payload-Based Web Attack Detection Using Deep Neural Network. In Proceedings of the 12th IEEE International Conference on Broadband Wireless Computing, Communicationand Applications (BWCCA), Barcelona, Spain, 8–10 November 2017. [CrossRef]
62. Althubiti, S.; Nick, W.; Mason, J.; Yuan, X.; Esterline, A. Applying Long Short-Term Memory Recurrent Neural Network for Intrusion Detection. *S. Afr. Comput. J.* **2018**, *56*, 136–154. [CrossRef]
63. Zhang, H.; Zhao, B.; Yuan, H.; Zhao, J.; Yan, X.; Li, F. SQL injection detection based on deep belief network. In Proceedings of the 3rd International Conference on Computer Science and Application Engineering, Sanya, China, 22–24 October 2019; pp. 1–6.
64. Jana, I.; Oprea, A. AppMine: Behavioral analytics for web application vulnerability detection. In Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, London, UK, 11 November 2019; pp. 69–80. [CrossRef]
65. Xie, X.; Ren, C.; Fu, Y.; Xu, J.; Guo, J. SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN. *IEEE Access* **2019**, *7*, 151475–151481. [CrossRef]
66. Qin, Z.Q.; Ma, X.K.; Wang, Y.J. Attentional Payload Anomaly Detector for Web Applications. In *Neural Information Processing*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 588–599. [CrossRef]
67. Sheykhkanloo, N. SQL-IDS: Evaluation of SQLi attack detection and classification based on machine learning techniques. In Proceedings of the 8th International Conference on Security of Information and Networks, Sochi, Russia, 8–10 September 2015. [CrossRef]
68. Hao, S.; Long, J.; Yang, Y. BL-IDS: Detecting Web Attacks Using Bi-LSTM Model Based on Deep Learning. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 551–563. [CrossRef]
69. Fidalgo, A.; Medeiros, I.; Antunes, P.; Neves, N. Towards a Deep Learning Model for Vulnerability Detection on Web Application Variants. In Proceedings of the 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Porto, Portugal, 24–28 October 2020; pp. 465–476. [CrossRef]
70. Gong, X.; Lu, J.; Wang, Y.; Qiu, H.; He, R.; Qiu, M. CECoR-Net: A Character-Level Neural Network Model for Web Attack Detection. In Proceedings of the 2019 IEEE International Conference on Smart Cloud (SmartCloud), Tokyo, Japan, 10–12 December 2019; pp. 98–103. [CrossRef]

71. Ito, M.; Iyatomi, H. Web application firewall using character-level convolutional neural network. In Proceedings of the 2018 IEEE 14th International Colloquium on Signal Processing Its Applications (CSPA), Penang, Malaysia, 9–10 March 2018; pp. 103–106. [CrossRef]
72. Zhao, J.; Wang, N.; Ma, Q.; Cheng, Z. Classifying Malicious URLs Using Gated Recurrent Neural Networks. In Proceedings of the 12th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Abertay University, Matsue, Japan, 4–6 July 2018. [CrossRef]
73. Tang, R.; Yang, Z.; Li, Z.; Meng, W.; Wang, H.; Li, Q.; Sun, Y.; Pei, D.; Wei, T.; Xu, Y.; et al. ZeroWall: Detecting Zero-Day Web Attacks through Encoder-Decoder Recurrent Neural Networks. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 2479–2488. [CrossRef]
74. Kuang, X.; Zhang, M.; Li, H.; Zhao, G.; Cao, H.; Wu, Z.; Wang, X. DeepWAF: Detecting Web Attacks Based on CNN and LSTM Models. In Proceedings of the International Symposium on Cyberspace Safety and Security, Guangzhou, China, 1–3 December 2019; pp. 121–136.
75. Luo, C.; Tan, Z.; Min, G.; Gan, J.; Shi, W.; Tian, Z. A Novel Web Attack Detection System for Internet of Things via Ensemble Classification. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5810–5818. [CrossRef]
76. Fang, Y.; Li, Y.; Liu, L.; Huang, C. DeepXSS: Cross Site Scripting Detection Based on Deep Learning. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence, Sanya, China, 21–23 December 2018; pp. 47–51. [CrossRef]
77. Mendonca, R.; Teodoro, A.; Rosa, R.; Saadi, M.; Melgarejo, D.; Nardelli, P.; Rodriguez, D. Intrusion Detection System Based on Fast Hierarchical Deep Convolutional Neural Network. *IEEE Access* **2021**, *9*, 61024–61034. [CrossRef]
78. Yang, W.; Zuo, W.; Cui, B. Detecting Malicious URLs via a Keyword-Based Convolutional Gated-Recurrent-Unit Neural Network. *IEEE Access* **2019**, *7*, 29891–29900. [CrossRef]
79. Jemal, I.; Haddar, M.; Cheikhrouhou, O.; Mahfoudhi, A. Malicious Http Request Detection Using Code-Level Convolutional Neural Network. In Proceedings of the International Conference on Risks and Security of Internet and Systems, Paris, France, 4–6 November 2020; Volume 12528, pp. 317–324. [CrossRef]
80. Yu, L.; Chen, L.; Dong, J.; Li, M.; Liu, L.; Zhao, B.; Zhang, C. Detecting Malicious Web Requests Using an Enhanced TextCNN. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 768–777. [CrossRef]
81. Jemal, I.; Haddar, M.A.; Cheikhrouhou, O.; Mahfoudhi, A. Performance evaluation of Convolutional Neural Network for web security. *Comput. Commun.* **2021**, *175*, 58–67. [CrossRef]
82. Tripathy, D.; Gohil, R.; Halabi, T. Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning. In Proceedings of the 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Baltimore, MD, USA, 25–27 May 2020; pp. 145–150. [CrossRef]
83. Melicher, W.; Fung, C.; Bauer, L.; Jia, L. Towards a Lightweight, Hybrid Approach for Detecting DOM XSS Vulnerabilities with Machine Learning. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 2684–2695. [CrossRef]
84. Pan, Y.; Sun, F.; Teng, Z.; White, J.; Schmidt, D.; Staples, J.; Krause, L. Detecting web attacks with end-to-end deep learning. *J. Internet Serv. Appl.* **2019**, *10*, 16. [CrossRef]
85. Manimurugan, S.; Al-Mutairi, S.; Aborokbah, M.M.; Chilamkurti, N.; Ganesan, S.; Patan, R. Effective attack detection in internet of medical things smart environment using a deep belief neural network. *IEEE Access* **2020**, *8*, 77396–77404. [CrossRef]
86. Stephan, J.J.; Mohammed, S.D.; Abbas, M.K. Neural network approach to web application protection. *Int. J. Inf. Educ. Technol.* **2015**, *5*, 150. [CrossRef]
87. Jemal, I.; Haddar, M.; Cheikhrouhou, O.; Mahfoudhi, A. ASCII Embedding: An Efficient Deep Learning Method for Web Attacks Detection. *Commun. Comput. Inf. Sci.* **2021**, *1322*, 286–297. [CrossRef]
88. Maurel, H.; Vidal, S.; Rezk, T. Statically Identifying XSS using Deep Learning. In Proceedings of the SECRYPT 2021-18th International Conference on Security and Cryptography, Online Streaming, 6–8 July 2021.
89. Karacan, H.; Sevri, M. A Novel Data Augmentation Technique and Deep Learning Model for Web Application Security. *IEEE Access* **2021**, *9*, 150781–150797. [CrossRef]
90. Chen, T.; Chen, Y.; Lv, M.; He, G.; Zhu, T.; Wang, T.; Weng, Z. A Payload Based Malicious HTTP Traffic Detection Method Using Transfer Semi-Supervised Learning. *Appl. Sci.* **2021**, *11*, 7188. [CrossRef]
91. Shahid, W.B.; Aslam, B.; Abbas, H.; Khalid, S.B.; Afzal, H. An enhanced deep learning based framework for web attacks detection, mitigation and attacker profiling. *J. Netw. Comput. Appl.* **2022**, *198*, 103270. [CrossRef]
92. Lin, Z.; Shi, Y.; Xue, Z. Idsgan: Generative adversarial networks for attack generation against intrusion detection. *arXiv* **2018**, arXiv:1809.02077.
93. Shahriar, M.H.; Haque, N.I.; Rahman, M.A.; Alonso, M. G-ids: Generative adversarial networks assisted intrusion detection system. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 376–385.
94. Farahnakian, F.; Heikkonen, J. A deep auto-encoder based approach for intrusion detection system. In Proceedings of the 2018 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon, Korea, 11–14 February 2018; pp. 178–183.

95. Gharib, M.; Mohammadi, B.; Dastgerdi, S.H.; Sabokrou, M. Autoids: Auto-encoder based method for intrusion detection system. *arXiv* **2019**, arXiv:1911.03306.
96. Alon, U.; Zilberstein, M.; Levy, O.; Yahav, E. code2vec: Learning distributed representations of code. *Proc. ACM Program. Lang.* **2019**, *3*, 1–29. [CrossRef]
97. Institute, P. Zero Day Attacks. Available online: https://cutt.ly/rYhswNo (accessed on 20 February 2022).