Computers & Security 120 (2022) 102788

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

TC 11 Briefing Papers

A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques

Tomás Sureda Riera [a,*], Juan-Ramón Bermejo Higuera [b], Javier Bermejo Higuera [b], José-Javier Martínez Herraiz [a], Juan-Antonio Sicilia Montalvo [b]

[a] Computer Science Department, University of Alcalá, Ctra.Madrid-Barcelona, Km. 33.600 28805 Alcalá de Henares (Madrid), Spain
[b] Escuela Superior de Ingeniería y Tecnología (ESIT), Universidad Internacional de la Rioja (UNIR), Av. de la Paz, 137, Logroño (La Rioja) 26006, Spain

ARTICLE INFO

Article history:
Received 9 February 2022
Revised 23 April 2022
Accepted 3 June 2022
Available online 5 June 2022

Keywords:
Multi-label classification
Dataset
LightGBM
CatBoost
Machine learning

ABSTRACT

Context: There are many datasets for training and evaluating models to detect web attacks, labeling each request as normal or attack. Web attack protection tools must provide additional information on the type of attack detected, in a clear and simple way.

Objectives: This paper presents a new multi-label dataset for classifying web attacks based on CAPEC classification, a new way of features extraction based on ASCII values, and the evaluation of several combinations of models and algorithms.

Methods: Using a new way to extract features by computing the average of the sum of the ASCII values of each of the characters in each field that compose a web request, several combinations of algorithms (LightGBM and CatBoost) and multi-label classification models are evaluated, to provide a complete CAPEC classification of the web attacks that a system is suffering. The training and test data used for training and evaluating the models come from the new SR-BH 2020 multi-label dataset.

Results: Calculating the average of the sum of the ASCII values of the different characters that make up a web request shows its usefulness for numeric encoding and feature extraction. The new SR-BH 2020 multi-label dataset allows the training and evaluation of multi-label classification models, also allowing the CAPEC classification of the various attacks that a web system is undergoing. The combination of the two-phase model with the MultiOutputClassifier module of the scikit-learn library, together with the CatBoost algorithm shows its superiority in classifying attacks in the different criticality scenarios.

Conclusion: Experimental results indicate that the combination of machine learning algorithms and multi-phase models leads to improved prediction of web attacks. Also, the use of a multi-label dataset is suitable for training learning models that provide information about the type of attack.

1. Introduction & motivation

Every year there are significant increases in the number of attacks against web servers and applications; e-commerce platforms, financial and government institutions, large corporations, etc. are targeted by web attacks for economic or ideological reasons. According to Cisco (Cisco, 2018), 14.5 million DDoS attacks are expected in 2022. Also, SQL Injection (SQLi) and Cross-site Scripting (XSS) attacks are easy and powerful methods for attacking a web site (Johari and Sharma, 2012). The impact of cyber-attacks suffered by companies threatened their viability in 17% of cases, reported specialist insurer Hiscox(Hiscox, 2021), with their website becoming the first point of entry in 29% of cases.

Several technologies and systems exist to prevent and detect attacks on web servers and applications: *misuse detection systems* with large rules and vulnerability signature databases that must be continuously updated, *anomaly detection systems* that interpret deviations based on expected patterns of user and application behavior, taking this behavior as evidence of malicious activity. Recently, there has been a significant increase in scien-

* Corresponding author.
E-mail addresses: tomas.sureda@uah.es (T.S. Riera), juanramon.bermejo@unir.net (J.-R.B. Higuera), javier.bermejo@unir.net (J.B. Higuera), josej.martinez@uah.es (J.-J.M. Herraiz), juanantonio.sicilia@unir.net (J.-A.S. Montalvo).

https://doi.org/10.1016/j.cose.2022.102788

tific interest in anomaly detection techniques applied to web intruder detection (Sureda Riera et al., 2020). To successfully train and evaluate models based on anomaly detection techniques, specific web traffic datasets are needed; a major drawback in the study of web attack prevention and detection is the lack of public datasets to audit and validate the studies performed in this field (Sureda Riera et al., 2020); the DARPA dataset and those belonging to the KDD family have been widely criticized in different studies (Brugger, 2007; Mahoney and Chan, 2003; McHugh, 2000; Tavallaee et al., 2009). The CSIC-2010 dataset has become one of the most popular in recent years for testing protection systems against web attacks. This dataset is generated by synthetic traffic and contains 36,000 requests labeled as normal and more than 25,000 labeled as anomalous (Torrano-Gimenez et al., 2009).

Most datasets are composed of artificially generated traffic and, to our best knowledge, all datasets available for training and/or evaluation of machine learning models provide only labeling of the request in terms of normality or attack, without specifying in any case what type(s) of attack(s) is/are being suffered. The authors strongly believe in the need for datasets that collect this type of additional information and that allow the training of machine learning models that classify attacks based on internationally accepted classification criteria, such as CAPEC. In this way, by providing the incident response team with the CAPEC classification of the attack, response times and effectiveness will be improved, as the specific attack pattern and possible mitigations can be queried.

For this reason, one of the main achievements of this work is the generation of a new dataset (the SR-BH 2020 dataset) that collects different types of attacks, coming from real traffic data (generated by collecting real traffic in a honeypot exposed to the Internet for 12 days), with multi-labels, that report the normality of the request, or the CAPEC classification of the type or types of attack that the web request represents. This dataset, to our knowledge, is the first one that allows the training and evaluation of multi-label machine learning models and algorithms, which can provide the CAPEC classification of the attack(s) that a web application is suffering.

One of the fundamental stages in any data science project is the preprocessing of the data that will be used to train the machine learning models; this stage includes the selection of the relevant characteristics of the dataset that allow a high level of model efficiency to be achieved when making the prediction. In the case of a dataset with web traffic data, it is necessary to numerically encode the various fields of each web request, so that it is possible to apply different statistical techniques to the resulting numerical values. In this study, we present a new form of numerical encoding consisting of calculating the average of the sum of the ASCII values of each of the characters that make up each field of a web request.

Several supervised machine learning algorithms and techniques applied to intrusion detection have been studied over the years, most notably algorithms using ensembles of decision trees in combination with gradient boosting (Tama et al., 2020; Vu et al., 2019); two popular algorithms that provide a gradient boosting framework are LigthGBM (Ke et al., 2017) and CatBoost (Dorogush et al., 2018). In this paper, we will evaluate (using different metrics, depending on the criticality levels of several scenarios) the performance of these algorithms in predicting web attacks, such that they report the normality of the request or the CAPEC classification(s) of the attack. As this is a multi-label classification task since a single request may contain more than one type of attack, the SR-BH 2020 dataset will be used and different multi-label classification models will be combined with the LightGBM and CatBoost algorithms.

This paper makes the following contributions:

- Construction of the SR-BH 2020[1] dataset, a new multi-label dataset for Web attack detection and prediction based on CAPEC attack patterns, suitable for training multi-label classification models.
- A new way of web request data encoding using the mean of the sum of the characters ASCII values.
- Design of one-phase, two-phase, and customized classification multi-label machine learning models.
- Study of the behavior of novel algorithms applying the designed classification multi-label models.
- A ranking is obtained of the different combinations of algorithms/models to be applied in the protection of different scenarios, according to their criticality levels, using different evaluation metrics.

The rest of this work is structured as follows: Section 2 describes background and related work. Section 3 shows in an overview the process followed in this work. In Section 4, the materials and methods used are provided including the dataset used for the experiments and method evaluation, the system followed for feature extraction from the dataset, and a description of the different models applied to algorithms. Section 5, shows the model results and considerations. The conclusions of the study are provided in Section 6.

2. Background and related works

2.1. Background

This section provides an overview of the WAF and RASP web protection tools and the similarities and differences between them, a presentation of the characteristics of the algorithms analyzed in this study, namely LightGBM and CatBoost, as well as a description of the metrics and scenarios of criticality to be used to evaluate the performance of the various models and algorithms.

2.1.1. Web application protection

One of the most widely used methods for Web application protection is the implementation of Web Application Firewall (WAF) tools. A WAF is deployed between the application and the requesting user, inspecting the incoming traffic at the application layer of the OSI model and looking for attack patterns, eventually blocking incoming malicious traffic. WAF devices work by checking that incoming traffic against a database of signatures or rules so that the update of that database is critical. They are independent of the programming language of the web application as they act before the malicious traffic gets to execute the code. Different ways of circumventing the protection provided by a WAF have been proposed (Ristic, 2022), OWASP: Normalization Method, Using HTTP Parameter Pollution (HPP), Using HTTP Parameter Fragmentation (HPF), Using logical requests AND / OR, replacing with their synonyms the SQL functions that get WAF signatures, using comments, case changing, triggering a Buffer Overflow / WAF Crashing. Most of these methods focus on protocol layer exploits that attempt to take advantage of small differences between how WAF, web servers, and backend applications see traffic.

Runtime Application Self-Protection (RASP) tools are defined by Gartner Gartner (2022) as "a security technology that is built or linked into an application or application runtime environment, and is capable of controlling application execution and detecting and preventing real-time attacks". These tools combine real-time contextual awareness of the factors that have led to the application

[1] Available at https://doi.org/10.7910/DVN/OGOIXX.

of current behavior (Dubey, 2016; Steiner et al., 2017). They work via "embedding" the security in the application server to be protected, intercepting all calls to the system to check they are secure so that they depend entirely on the programming language of the application to protect.

Both technologies work in the application layer filtering the HTTP protocol. WAF is a black box technology (it does not need access to the logic of the application; it intercepts the calls and responses to the logic of the application to be protected, performing a syntactic analysis to detect attacks). On the other hand, RASP is a white-box technology that works by installing an agent on the application server, examining how variables in the application's process and code stack get their values to, based on that information, predict whether an attack is taking place.

Several attempts have been made to improve the effectiveness of WAF solutions. The model proposed by Moosa (2010) is based on an Artificial Neural Network (ANN) that defends against SQL injection attacks. While in the training phase the ANN is exposed to a set of normal and harmful data, in the working phase, the trained ANN is embedded into the WAF, thus protecting the entire web server.

In an attempt to improve the effectiveness of WAF rule sets, Auxilia and Tamilselvan (2010) propose a negative security model, which monitors applications for security anomalies, uncommon behaviors, and common web application attacks.

Taint analysis is one of the techniques used in RASP solutions. Haldar et al. Haldar et al. (2005), propose the analysis of externals to the web application data, marking them as untrusted (tainted data). They identify, monitor, and prevent the inappropriate use of this type of data at runtime in the web application to protect, developing a heuristic that instruments the java.lang.String class, to propagate taintedness of strings, as well as to mark certain strings as untainted.

Halfond et al. (2008) propose the use of *positive tainting*, identifying and tracing trusted data at execution time, while conventional tainting is centered on untrusted data. In this way, False Negatives (FN) are completely eliminated, at the cost of producing an increase in False Positives (FP). This approach also makes use of syntax-aware evaluation, so that data usage regulation is applied at development time based on its syntax in the query string, requiring only the deployment of the web application using the MetaStrings library.

2.1.2. Algorithms

LightGBM LightGradient Boosting Machine, abbreviated as LightGBM, is an open-source library that provides a fast, decentralized, highly performant gradient boosting environment on the basis of a decision tree algorithm. Guolin Ke, et al.Ke et al. (2017), introduced two key concepts:

- Gradient-based One-Side Sampling (GOSS): A modified version of the gradient boosting method that uses only large gradient data instances. GOSS can get fairly accurate information gain estimates with a much smaller data size, which speeds up training and reduces the computational complexity of the method.
- Exclusive Feature Bundling (EFB): This approach groups dispersed (mostly null) features that are mutually exclusive from each other, thus becoming a feature selection method.

In standard decision tree algorithms, such as c4.5 (Ross Quinlan et al., 1994) and CART (Breiman et al., 1984), nodes are expanded in order of depth *(level-wise tree growth)* through of the "divide and conquer" strategy, using a prefixed order (usually from left to right).

LightGBM, on the other hand, is part of a so-called "best-first decision trees" *(leaf tree growth)* where nodes are expanded in the best order rather than in a fixed order. In this case, the best split node at each step is added to the tree. The best node is a node that is not classified as a terminal, that is, a node that minimizes the Gini impurity index among all nodes that can be split (Shi, 2007).

LightGBM uses a histogram-based algorithm. That is, group successive feature values into individual bins, build feature histograms during training, and speeds up this process and reduces memory usage. Following the leaf-wise based approach produces a much more complex tree than the hierarchical-based approach, which is a key factor in achieving higher accuracy. But sometimes this can lead to overfitting. LightGBM aims to reduce the complexity of histogram construction by using GOSS and EFB to reduce sampling data and features.

CatBoost

Yandex developed CatBoost, an open-source software library that provides an innovative categorical feature processing algorithm and a gradient boosting framework that implements *ordering boosting*, a permutation-based alternative to traditional algorithms. CatBoost uses one-time encoding to implement a symmetric tree that handles categorical features and helps reduce prediction times. LightGBM uses *GOSS* to reduce complexity, while the CatBoost algorithm introduces *Minimal Variance Sampling (MVS)*, a weighted version of stochastic gradient boosting sampling used to normalize boosting models. Using MVS reduces the number of examples required for each boosting iteration and greatly improves the quality of the model, making the model more general and less likely to overfit (Dorogush et al., 2018; Prokhorenkova et al., 2018).

Another important concept of CatBoost is the use of Oblivious Decision Trees (ODT's) in the process of building Decision Trees so that a set of ODT's is built. If an ODT has n levels, the set will have 2^n levels, since an ODT is a complete binary tree. The same splitting criterion will be applied to all nodes that are not leaves of the ODT (Hancock and Khoshgoftaar, 2020; Prokhorenkova et al., 2018). According to Prokhorenkova et al. ODTs are balanced, allow speeding up execution, and are less prone to overfitting (Prokhorenkova et al., 2018).

2.1.3. Evaluation metrics and scenarios

Most supervised classification algorithms focus on binary or multi-class classification; in this case, classical classification metrics are adequate, e.g. accuracy, F1 Score, precision, recall, etc. (Sureda Riera et al., 2020). However, when working with datasets in which there are several labels for each observation, it is necessary to complement the classical metrics since the notion of a partially correct prediction of the various labels that make up each observation is introduced (Cheng and Hüllermeier, 2009; Gouk et al., 2016; Read et al., 2011; Zhang and Zhou, 2014).

Also, according to the work of Antunes and Vieira (2015), it is necessary for the metrics to make sense of the model being evaluated and the scenario to which the model is applied; for this reason, their recommendations are followed and four different types of scenarios are defined in which each algorithm/model combination is evaluated, applying the recommended metric in each scenario.

In our case, we have chosen to perform several evaluations: first, we have calculated the Accuracy following the exact-match principle (exact prediction of all the labels); F measure, Recall, Precision, ROC AUC, Informedness, and Markedness of each of the model/algorithm combinations have been calculated on an overall basis. On the other hand, Accuracy and F measure have been evaluated for each of the labels individually, as it may be useful to determine how well the model predictions fit for each type of attack; additional metrics such as Hamming Loss, Hamming Score, and Jaccard Similarity have been introduced to evaluate the models taking into account possible partially correct model predictions.

Metrics

- **Accuracy (for each label):** This is the measure of accuracy that we will use to assess the model's efficiency in predicting each of the labels that make up an observation; in this case, we will apply the classical definition of accuracy:

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

- **Accuracy - Exact Match (EMR):** This is the measure of accuracy that we will use to evaluate the global model performance; in our case, the exact measure of the prediction for all the labels that compose an observation: We just ignore partially correct predictions (considering them incorrect) and extend the concept of Accuracy used for single label prediction to the multi-label case.

$$EMR = \frac{Number\ of\ records\ with\ exact\ label\ match}{Total\ number\ of\ records}$$

- **Precision:** In our case, to account for label imbalance, we computed the proportion between the correct predicted labels and the total labels averaged over all cases and weighted them by support (the total number of cases for each label).

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

$$Precision_{weighted} = \sum_{classes} weight\ of\ class\ x\ precision\ of\ class$$

- **Recall:** This is the true positive percentage correctly identified, averaged across all instances and support-weighted.

$$Recall = \frac{TP}{TP + FN}$$

$$Recall_{weighted} = \sum_{classes} weight\ of\ class\ x\ recall\ of\ class$$

- **F measure (for each label):** It is a measure of the accuracy of a test, taking into account both precision and recall. Also known as F1-Score or F-Score, this value is a balanced harmonic mean of two metrics: Precision (P) and Recall (R) Bermejo Higuera (2013); Díaz and Bermejo (2013); Van Rijsbergen (1979).

$$F\ measure = \frac{2\ x\ precision\ x\ recall}{precision + recall}$$

- **F measure (averaged):** F measure score averaged across all instances and support-weighted.

$$F\ measure_{weighted} = \sum_{classes} weight\ of\ class\ x\ F\ measure\ of\ class$$

- **Informedness:** Is a measure of how much information the system provides about positive and negative labels,i.e. how informed a predictor is for the desired condition, as opposed to chance, averaged across all instances and support-weighted.

$$Informedness = \frac{TP}{FN + TP} + \frac{TN}{FP + TN} - 1 = \frac{TP}{FN + TP} - \frac{FP}{FP + TN}$$

$$Informedness_{weighted} = \sum_{classes} weight\ of\ class\ x\ Informedness\ of\ class$$

- **Markedness:** A measure of the confidence in the system's positive and negative predictions, it quantifies how consistently the outcome includes a predictor variable as a marker, that is, how the labeled condition for a given predictor compares to chance, averaged across all instances and support-weighted.

$$Markedness = \frac{TP}{TP + FP} + \frac{TN}{FN + TN} - 1 = \frac{TP}{TP + FP} - \frac{FN}{FN + TN}$$

$$Markedness_{weighted} = \sum_{classes} weight\ of\ class\ x\ Markedness\ of\ class$$

- **ROC AUC:** Shows the global efficiency of a classification model at all classification levels, by plotting the true positive rate (TPR) versus the false positive rate (FPR). The AUC is a bidimensional metric of the area under the full ROC curve, which ranges from 0 (100% inaccurate predictions) to 1 (100% accurate predictions), reflecting the separability grades and showing the capacity of a particular model to distinguish among classes (Sureda Riera et al., 2020; Swets, 1996). In our case, ROC AUC is averaged across all instances and support-weighted.
- **Hammming Loss:** Is the proportion of incorrectly predicted labels over the total number of labels. In multi-label classification, the Hamming loss is calculated as the Hamming distance between the true and the predicted values. Its value ranges from 0 to 1. The lower the value, the better the performance of the model.Let D be a multi-label dataset, consisting of $|D|$ multi-label observations (x_i, Y_i), i = 1.|D|, $Y_i \subseteq L$. Let H be a multi-label classifier and $Z_i = H(x_i)$ be the set of labels predicted by H for observation x_i.

$$Hamming\ Loss\ (H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \Delta Z_i|}{|L|}$$

Where Δ represents the symmetric difference between the two sets (Schapire and Singer, 2000; Tsoumakas and Katakis, 2009).
- **Jaccard Similarity:** It measures the degree of similarity between two sets by examining the proportion of correctly predicted positive labels in a potentially positive set (expected positive and real positive).

$$\mathcal{J}(T, P) = \frac{T \cap P}{T \cup P}$$

where T and P are true labels and predicted labels respectively.

Scenarios

Four different scenarios are defined, based on the classification made by Antunes and Vieira (2015):

- **Very high critical scenario:** Represents the development and evaluation of critical applications that have very high-security requirements because they need to provide their customers with a reliable system. Examples of this scenario are internet bank websites, trade-in equity shares, or massive electronic commerce systems. The top priority in this scenario is the **elimination of the largest number of attacks**, thus assuming the investment of time and resources in remediating the occurrence of non-existent attacks (false positives). In this case, the metric of choice is **recall**, as it maximizes attack detection.
- **Heightened-critical scenario:** In this case, the goal is to achieve a **balance between the priority of detecting and eliminating the maximum number of attacks and preventing the excessive reporting of false positives**, since resources in this type of scenario must be properly managed. Security requirements are high, but lower than Very high critical scenario. Examples of this scenario are government portals, e-commerce web applications, etc. A metric of choice in this scenario could

be the F-measure, but the problem is that it assigns equal importance to precision and recall; **Informedness** appears to be a better alternative, as it is bias-free and does not have the disadvantages of harmonic averaging.

- **Medium-critical scenario:** This scenario features less exposed or less criticals applications, typically with a limited budget, so the resources available for remediation of reported attacks are limited. For this reason, **both finding and eliminating as many attacks as possible and saving resources on remediation of false positives have equal importance**. Examples of this type of scenario are web sites, where attacks result in lower financial losses, or intranet applications that are less prone to external attacks. The optimal solution for this case is to use **F-Measure**.
- **Low-critical scenario:** This scenario represents non-critical applications that are not very exposed to attacks. They are characterized by being implemented with small budgets, so the resources available are limited; the use of these resources should be focused on confirmed attacks. The goal is to **report as few false positives as possible**, increasing confidence in reported attacks. Examples of such scenarios are web portals for small and medium-sized companies. **Markedness** seems to be the best metric. Actually, it is able to give greater preponderance to the accuracy considerations, while at the same time being capable of considering the attacks that are left unnotified.

2.2. Related work

This section details the state of the art related to studies on dataset generation, pattern extraction and feature selection techniques, attack classification and deep learning models.

2.2.1. Datasets

Many datasets have been proposed for the study and evaluation of models and techniques to enable web attack prediction:

- DARPA: The Lincoln Laboratory of the Massachusetts Institute of Technology (MIT) created this dataset in 1998 and updated it in 1999. It consists of about 5 million connection records, which contain raw tcpdump data. It has 244 tagged cases of 58 attacks on four different OS (Lichman, 2000; Tavallaee et al., 2009).
- KDD Cup 99: This is a variant of the DARPA dataset with about 4,900,000 and 2,000,000 entries in the training and test datasets, each of which contains 41 features and is appropriate for training several machine learning algorithms. It may produce inconsistent results, due to the presence of duplicate records (Tavallaee et al., 2009).
- NSL-KDD: In an attempt to deal with one of the problems with the KDD Cup 99 dataset, a series of cleanup operations were performed on the duplicate records. From this processing, the new NSL-KDD dataset was created, which has 175,341 and 82,332 records in the training and test sets, respectively (Devi and Abualkibash, 2019).
- UNSW-NB15: This is a mixed dataset of real normal activities and synthetically generated behaviors of modern attacks, with 47 features and 2 class-labeled features (Moustafa and Slay, 2015).
- Kyoto 2006: This is an exhaustive and representative dataset generated from data obtained from real-time traffic. It has 24 features, 14 of which were retrieved from KDD Cup '99, plus 10 additional ones, but not including those features that contain duplicate records (Protić, 2018).
- ISCX: Emerged from the work of Shiravi et al. Shiravi et al. (2012), this dataset was built by generating simulated traffic for seven days. It contains 11 features and, in addition to the requests being labeled as normal or attack, provides a description of the network traffic.
- CSIC-2010: Based on the work of Torrano-Gimenez, Perez-Villegas and Alvarez Torrano-Gimenez et al. (2009), this dataset was produced at the Consejo Superior de Investigaciones Científicas (CSIC). Originated from simulated web requests to an e-commerce web application, this dataset is composed of 36,000 normal requests and over 25,000 anomalous ones, marked as either normal or anomalous.
- ECML/PKDD 2007: Generated from real traffic and replacing parameter values and names with random values in order to anonymize the data, this dataset includes 35,006 normal records and 15,110 records labeled as attacks (Raïssi et al., 2007).

Although the University of California, specifically the archival authority of the KDD Archive in Irvine discourages its use Brugger (2007); Tavallaee et al. (2009), as well as is considered inadequate and obsolete Mahoney and Chan (2003); McHugh (2000), the KDDD family (DARPA, KDD CUP 99, and NSL_KDD), is widely used in current intrusion detection system evaluation studies (Siddique et al., 2019). With the generation of the new SR-BH 2020 dataset, the state of the art is improved by providing the first dataset, derived from real web traffic data, specifically designed for the training of multi-label web attack prevention and detection models.

2.2.2. Pattern extraction and features selection

Krügel et al. (2002) use a model of distribution of characters to characterize the traffic genuinely generated to the web application. In this work, in contrast to the previous one, a numerical value to each field of a web request is assigned, based on the ASCII value corresponding to each character, in order to train models that can predict the normality or malignity of a web request, as well as the corresponding CAPEC key.

In Kruegel and Vigna (2003), Kruegel and Vigna introduce an anomaly detection system that uses web server log files as input to produce an anomaly score for every web request based on the length and character distribution of the attributes. In the present work, the anomaly score is calculated for each field of interest of a web request extracted from the ModSecurity log.

Kozik et al. (2015) detect anomalies in HTTP traffic, using a pattern extraction method derived from the distribution character model suggested by Kruegel, Toth, and Kirda as well as token detection of a web request, using text segmentation. Our work takes advantage of the log generated by ModSecurity and returns a numeric value for each field of interest.

Resende and Drummond (2018) select characteristics and profile parameters for intrusion detection methods based on anomaly, using an adaptational approach that relies on a genetic algorithm. In our work, features are selected by generating a histogram based on the mean ASCII value obtained by each field of a web request.

In Tan and Hoai (2021), Tan and Hoai propose the HQTN technique that transforms the HTTP request into numeric, focusing on attributes names and values, and query strings and using the CityHash hash function, testing their approach on the CSIC 2010 dataset. In our approach, we transform the full web request to a numeric value by calculating the mean value of the sum of the ASCII values of all the web request fields, which in principle is much easier and gives good results. In addition, they work with a binary label dataset (CSIC 2010) and we work with the SR-BH 2010 dataset which is specific for multi-label classification.

2.2.3. Attacks classification

Dang and François (2018) use relationship inference between various cybersecurity issue repositories: *CAPEC (Common Attack Pattern Enumeration and Classification), CWE (Common Weakness Enumeration), and CVE (Common Vulnerabilities and*

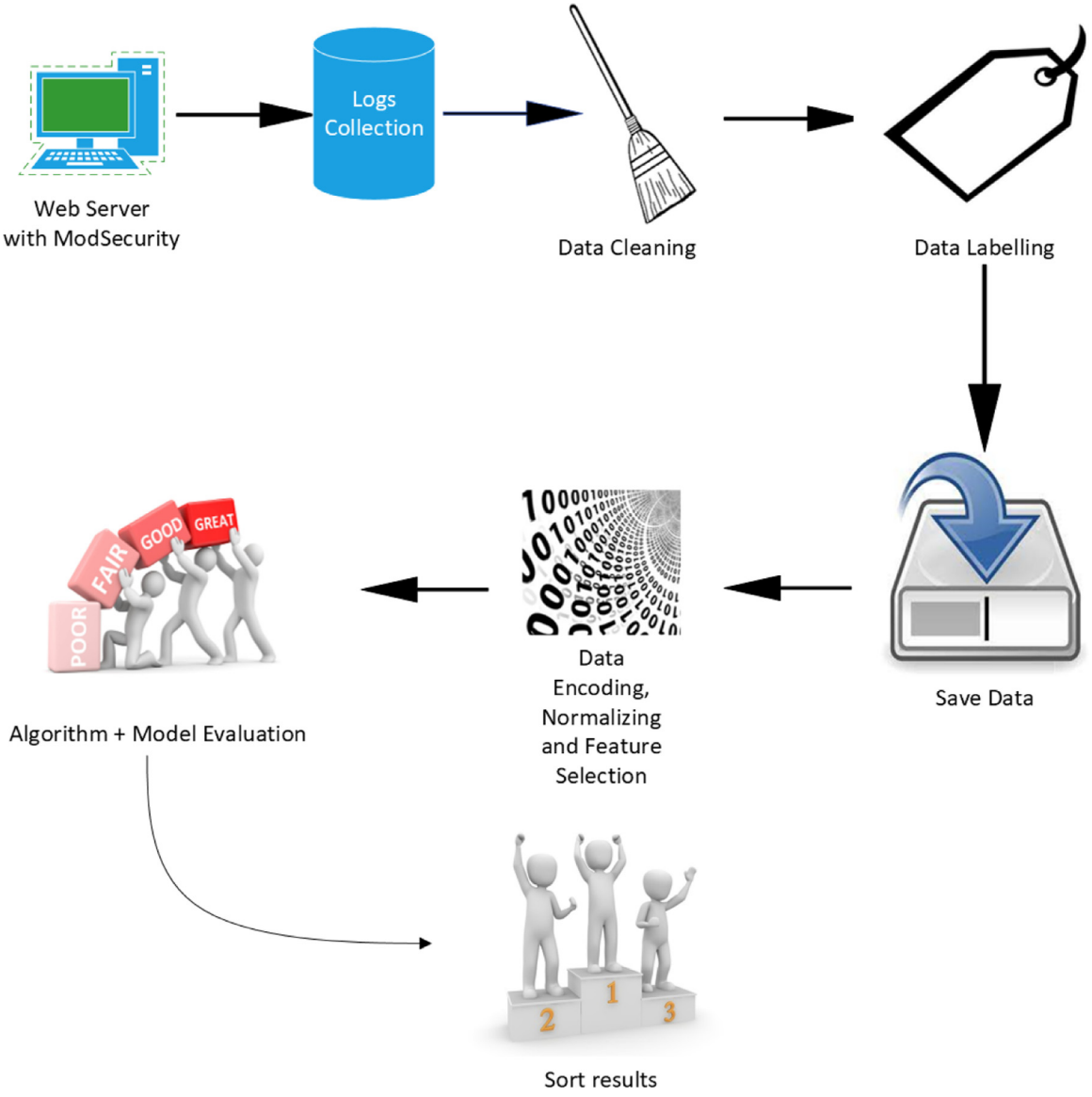

Fig. 1. Process overview.

Exposures) to detect patterns of attacks and weak points in the software associated with SDN/NFV software vulnerabilities. Kanakogi et al. (2021) use three different measures of similarity: TF-IDF, Universal Sentence Encoder (USE), and Sentence-BERT (SBERT) to track correlated CAPEC-IDs based on CVE-IDs. The algorithm/model combinations proposed in our work, return information to the security operator on the CAPEC classifications of the attack the system is undergoing.

2.2.4. Deep learning models

Mac et al. (2018) detect harmful patterns in the HTTP/HTTPS traffic, using an autoencoder. They worked on the raw web request, collecting the absolute path, the method, and the query parameters thanks to a preprocessing of the data by tokenizing the URL, replacing the characters with their corresponding ASCII code. In the present work, a numeric value based on the mean value of the sum of the ASCII score of all the characters that are part of a field in a web request is calculated.

Liang et al. (2017) propose the use of an Autoencoder and a Recurrent Neural Network (RNN) to detect anomalous requests on web servers, tokenizing the URLs to reduce their variability, while Jin et al. (2018) apply AutoEncoder and RNN to identify web attacks based on payloads. Instead of using neural networks, various combinations of models and novel algorithms (LightGBM and CatBoost) in the field of machine learning are evaluated.

Pan et al. (2019) detect runtime intrusions by mining call traces in web applications and learning the correct program execution model through a stacked denoising autoencoder; they call their model Robust Software Modeling Tool (RSMT). Truong et al. (2019) propose detecting anomalous HTTP queries using Sum Rule and Xgboost and combining the related results with several stacked denoising autoencoders (SDAE). In our work, we do not use deep learning techniques, but train models of different phases, using LightGBM and CatBoost.

Tama et al. propose an architecture of stacked ensembles where its base learners are other ensembles learners Tama et al. (2020). Tekerek proposes an anomaly-based Web attack detection architecture that relies on Convolution Neural Network (CNN) (Tekerek, 2021). Our work proposes a two-phase model architecture with CatBoost algorithm.

Montes et al. (2021) use deep learning techniques to enhance Modsecurity performance using a two-phase model: first, using

Table 1
Number of web requests by CAPEC classification.

CAPEC Classification	Number of web requests	% of total requests
000 - Normal	525,195	57.85%
272 - Protocol Manipulation	9153	1.00%
242 - Code Injection	15,827	1.74%
88 - OS Command Injection	7482	0.82%
126 - Path Traversal	20,992	2.31%
66 - SQL Injection	250,311	27.57%
16 - Dictionary-based Password Attack	1847	0.20%
310 - Scanning for vulnerable software	2718	0.30%
153 - Input Data Manipulation	2272	0.25%
274 - HTTP Verb Tampering	5437	0.60%
194 - Fake the source of data	56,145	6.18%
34 - HTTP Response Splitting	19,738	2.17%
33 - HTTP Request Smuggling	1059	0.12%
TOTAL	918,176	

deep learning technology, features are extracted; in a second phase, http requests are treated as raw text to train a one-class supervised model. Our work uses a numerical abnormality value of the fields of interest of a web request to train a two-phase model combined with the CatBoost algorithm.

Oliveira et al. (2021), propose a *multi-class* classification to inform the attack type, comparing a Random Forest (RF), a Multi-Layer Perceptron (ML), and a Long-Short Term Memory (LSTM) algorithm. Although their proposal obtains very good results, it is important to consider that the fact of working with a multi-class classification implies that a request can only be classified under a single type of attack. Our work allows us to properly classify web requests that involve more than one type of simultaneous attack.

Zhang and Zhou (2007), based on the K-nearest neighbor (KNN) algorithm, develop a lazy multi-label learning algorithm. Madjarov et al. (2012) used 11 benchmark data sets on which they experimentally compared 12 multi-label learning methods by using 16 metrics. Zhang and Zhou (2014) review different multi-label learning algorithms and detail different evaluation metrics. Read et al. (2011) perform multi-label classification using a chain of classifiers. Büyükçakir et al. (2018) propose an online stacked ensemble for multi-label stream classification. Wang et al. (2020) propose a collaboration-based multi-label model for e-commerce fraud detection.

3. Process overview

The process followed to complete this work can be summarized as follows:

- First, ModSecurity for Apache with Core Rule Set (CRS) version 3.3.0 is installed on a web server exposed to the Internet. ModSecurity is configured in "Detection Only" mode.
- During the exposure period, the logs generated by the ModSecurity activity are collected on a daily basis.
- The logs are reviewed and cleaned manually and semi-automatically. The correct labeling of each log made by ModSecurity is verified, thus ensuring a proper CAPEC classification.
- The reviewed logs are saved in CSV format, resulting in the SR-BH 2020 dataset.
- Each of the input fields of the dataset is numerically encoded by calculating the mean value of the sum of the ASCII code assignment of each of the characters in each field.
- The values obtained are normalized and a selection of the relevant features of the dataset is made.
- The performance of different combinations of algorithms and multi-label classification models in predicting the CAPEC classification is evaluated.

Table 2
Number of different CAPEC classifications assigned to a web request.

Number of different CAPEC classification	Number of web requests
1	898,576
2	8132
3	1088
4	18

- The results obtained are tabulated and sorted by different metrics according to the level of criticality of the different scenarios chosen.

A graphical overview of the process can be seen in Fig. 1.

4. Materials and methods

4.1. Dataset description

In this study, our new SR-BH 2020 dataset has been developed and implemented to experiment and evaluate the different algorithms and models. The dataset is composed of web requests collected during 12 days of July 2020 by a web server (Wordpress) installed on a virtual machine and exposed to Internet. On this server, Modsecurity version 2.9.2 for Apache, with Core Rule Set (CRS) version 3.3.0 was installed in *"Detection only"* mode, so that all requests (legitimate and malicious) were recorded in the log generated by ModSecurity, but without being blocked. Daily, the logs generated by ModSecurity were collected and the virtual machine was restored to a clean state.

Once the web server exposure period was over, the collected logs were manually and semi-automatically processed by one of the authors to review the web request tagging performed by Modsecurity, correcting where necessary the normal/attack assignment to the corresponding web request and ensuring an appropriate CAPEC classification assignment.

The final result is a multi-label dataset aimed especially at web attack detection and composed of 907,814 requests of which 525,195 are normal requests and 382,619 are anomalous requests, where each record has 24 different features and a set of 13 labels. See Table 1 for detailed information on the number of times a web request is classified under a given CAPEC heading. Note that the sum total of the number of CAPEC classifications in Table 1 is greater than the number of web requests present in the dataset, due to the fact that there are web requests in which more than one type of attack is combined. See Table 2 for details of the number of web requests with multiple CAPEC classifications.

In order to protect the personal data of users accessing the web server, the environment was configured so that all web re-

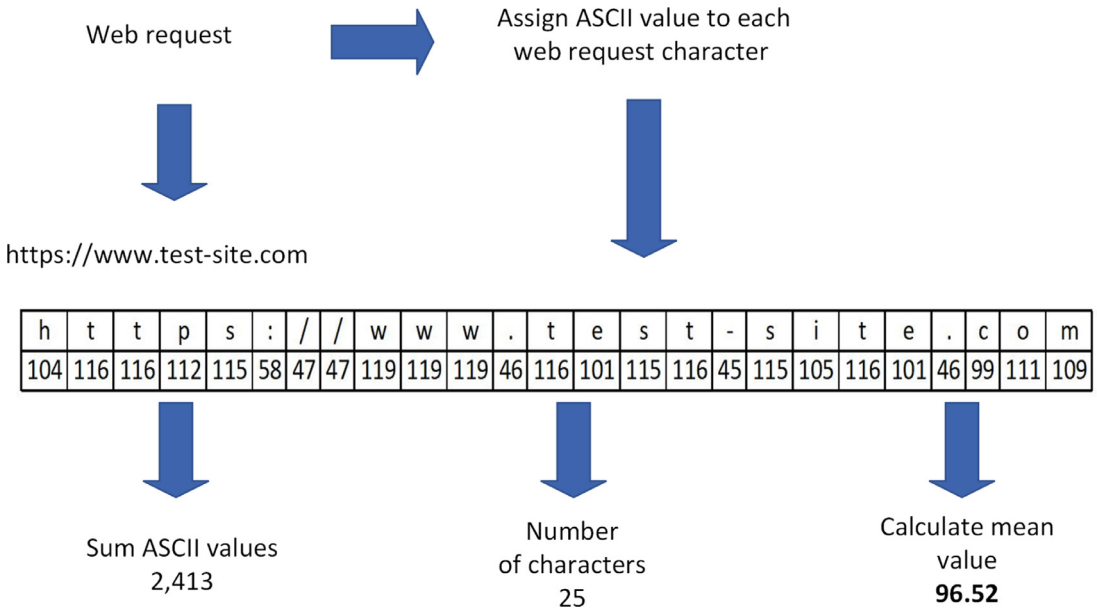

Fig. 2. Example of a web request numerical coding.

quests pass through a router interposed between the web server and the Internet connection: in this way, all requests received by the web server seem to be originated from the local IP address of the router.

4.2. Preprocessing the data, numerical transformation and features selection

Before performing the training of the different machine learning models, a review and preprocessing of the dataset data is necessary to avoid inconsistent and/or duplicated data, error correction and, at the same time, to adapt the data for numerical coding so that they are usable for the machine learning models.

The main objective of the data preprocessing and selection of relevant features of the dataset is to allow the different combinations of algorithms and models evaluated to reach the maximum level of performance and efficiency in their predictions, in addition to reducing the computational cost of modeling.

The numerical transformation of the dataset data was carried out using a procedure inspired by the work of Kozik et al. (2015), Krügel et al. (2002) and Kruegel and Vigna (2003). In our case, the mean of the sum of the ASCII values of the characters (applying a transformation to lowercase) of each of the fields of each web request is calculated: in this way, those fields with a high presence of anomalous characters (such as "./../.", present in a typical "path traversal" attack attempt) will obtain different mean ASCII values than those fields where the web request made is normal. The detailed procedure is provided by Algorithm 1. See Fig. 2 for an example of the proposed numerical coding.

In Table 3, the mean values of different fields of a normal web request and one labeled as a combination of "Protocol Manipulation" and "OS Command Injection" attacks are compared. In Table 4, shows the detail of the fields of the web request in which there are differences in the mean sum of their ASCII values.

Feature selection involves selecting those input variables that have the strongest relationship with the targets variables. Once the mean ASCII values of each feature have been calculated, a histogram is generated for each field in such a way that it is possible to determine and eliminate those characteristics that do not provide differential information. As can be seen in graph A of Fig. 3,

Table 3
Mean field ASCII values of normal and attack web request.

Field	Mean value normal request	Mean value attack request	Difference
method_value	106.667	113.5	Yes
http_request_value	97.1	94.7	Yes
protocol_value	79.875	79.875	No
referer_value	105.667	105.667	No
agent_value	73.9618	98.4444	Yes
host_value	99.6154	48.5556	Yes
origin_value	105.667	105.667	No
cookie_value	105.667	105.667	No
content_type_value	105.667	100.848	Yes
accept_value	43.6667	43.6667	No
accept_language_value	105.667	105.667	No
accept_encoding_value	95.6154	95.6154	No
do_not_track_value	0	0	No
connection_value	99.5	99.5	No
body_value	105.667	92.7102	Yes
response_http_protocol_value	79.875	79.875	No
http_status_code_value	200	404	Yes
http_status_message_value	109	101	Yes
response_content_length_value	51.6667	51.3333	Yes

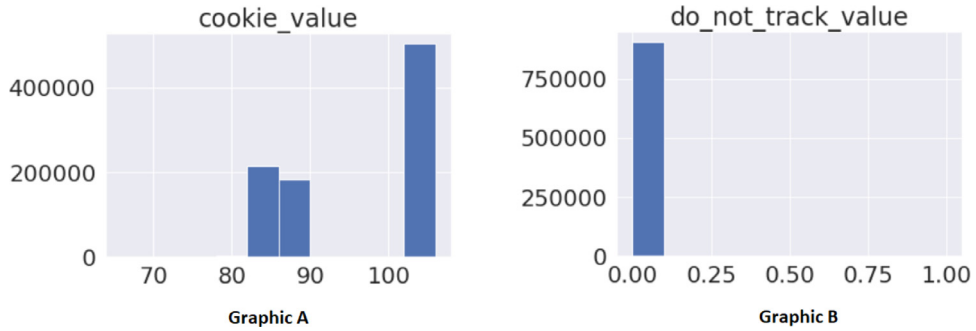

Fig. 3. Histogram of features informational levels.

Data: A field of a web request F
Result: Decimal value D
$L \leftarrow$ *length of* F;
if $L = 0$ **then**
  $D \leftarrow 0$;
else
  $v \leftarrow 0$;
  while *there is data to read in* F **do**
    $c \leftarrow$ read a character;
    $c \leftarrow$ convert c to lowercase;
    $c \leftarrow$ calculate c ASCII value;
    $v \leftarrow v + c$;
  end
  $D \leftarrow v/L$;
end

Algorithm 1: Calculation of the mean of the sum of character ASCII values in a field.

the cookie_value feature provides useful information to allow the differentiation of web requests, since its numerical values are distributed in the 80–90 and 100–110 ranges. However, in graphic B it can be seen that do_not_track_value feature does not provide any useful information since all web requests have the same value.

Once the features that do not provide useful information have been removed, an approximation to the normal distribution of each remaining feature is made by applying the natural logarithm to each value of the remaining features, subsequently standardizing their values using the StandardScaler class of the Scikit-learn library Pedregosa et al. (2011).

Finally, using the random forest classification algorithm and the RFECV class Guyon et al. (2002) of the scikit-learn library Pedregosa et al. (2011), we performed recursive feature removal using cross-validation and selected the final number of features.

When working with a dataset composed of real data, it is common to have imbalanced classes; this different percentage of representation of the classes in a dataset can affect the different evaluation metrics of the learning models. Although there are several methods to generate synthetic data that promote the equal representation of the different classes in a dataset, such as SMOTE Chawla et al. (2002) and MLSMOTE Charte et al. (2015), we have chosen to work only with real data and evaluate the algorithms and models with specific metrics that take in consideration the different percentage of representation of the classes in the dataset: Accuracy, Precision, Recall, F-Score, Hamming Loss, Hamming Score, Jaccard Similarity and ROC AUC. The selected features and labels are detailed in Table 5.

4.3. Models

The SR-BH 2020 dataset has a set of 13 labels. First label indicates whether the web request is considered normal or not, so it is assumed that if its value is 1 (normal request), the rest of the labels of the set should be 0. On the contrary, if the value of the first label is 0 (possible attack), there should be one or more of the labels of the remaining set with its value at 1.

This assumption allows us to establish a division of classifications by phases: Classifications can be established with a single-phase model, in which an attempt is made to predict the entire set of labels independently of the value obtained by the first label, i.e., even if the first label indicates that the request is normal, an attempt will be made to predict the remaining labels in the set. On the other hand, it is possible to generate two-phase prediction models in which the algorithm will only predict the rest of the tags

Table 4
Detail of fields with different mean ASCII values.

Field	Normal web request	Attack web request
method_value	GET	POST
http_request_value	/blog/xmlrpc.php?rsd	/cgi-bin/ViewLog.asp
agent_value	Mozilla/4.0 (compatible;...)	B4ckdoor-owned-you
host_value	test-site.com	127.0.0.1
content_type_value	nan	application/x-www-form-urlencoded
body_value	nan	remote_submit_Flag=1& remote_syslog_Flag=1& RemoteSyslogSupported=1&LogFlag=0& remote_host=%3bcd+/tmp;wget+ http://45.95.168.230/ taevimncorufglbzhwxqpdkjs/Meth.arm7 ;chmod+777+Meth
http_status_code_value	200	404
http_status_message_value	OK	Not Found
response_content_length_value	317	271

Table 5
Final set of selected features and labels.

Feature	Selected
method_value	Yes
http_request_value	Yes
protocol_value	**No**
referer_value	Yes
agent_value	Yes
host_value	**No**
origin_value	**No**
cookie_value	Yes
content_type_value	**No**
accept_value	Yes
accept_language_value	**No**
accept_encoding_value	**No**
do_not_track_value	**No**
connection_value	**No**
body_value	Yes
response_http_protocol_value	**No**
http_status_code_value	**No**
http_status_message_value	Yes
response_content_length_value	Yes
000 - Normal	Yes
272 - Protocol Manipulation	Yes
242 - Code Injection	Yes
88 - OS Command Injection	Yes
126 - Path Traversal	Yes
66 - SQL Injection	Yes
16 - Dictionary-based Password Attack	Yes
310 - Scanning for Vulnerable Software	Yes
153 - Input Data Manipulation	Yes
274 - HTTP Verb Tampering	Yes
194 - Fake the Source of Data	Yes
34 - HTTP Response Splitting	Yes
33 - HTTP Request Smuggling	Yes

if the value of the first tag is 0; if its value is 1 (normal request), it will automatically set all the remaining tags in the set to 0.

A customized model is also generated in which the best hyperparameters for the classification of each of the labels are calculated using GridSearchCV with the LightGBM and CatBoost algorithms. The prediction of each label will be performed with the corresponding algorithm adjusted with the calculated hyperparameters.

In order to obtain a model that provides the best possible results in predicting the normality of the web request, or the resulting CAPEC classification in the case of a web attack, the two algorithms (LightGBM and CatBoost) have been evaluated, using five different models:

i A single-phase model, using the *skmultilearn.problem_transform.BinaryRelevance* class of the *scikit-multilearn* library Szymański and Kajdanowicz (2017), transforming a multi-label classification problem with L labels into L individual binary classification problems with single labels. The prediction result is the union of all label classifiers. Binary Relevance ignores label correlation that exists in the training data, i.e. it assumes independence between labels; due to this loss of information, it would be possible for the prediction sets to contain combinations of labels that never occur in reality.

ii A single-phase model, using the *skmultilearn.problem_transform.ClassifierChain* class of the *scikit-multilearn* library Szymański and Kajdanowicz (2017). This class constructs a conditional sequence of Bayesian label classifiers, as described by Read Read et al. (2009), by generating label classifiers and sorting them into a sequence according to the *Bayesian Chain Rule*. In this case, a chain of binary classifiers is formed in which each classifier in the chain is responsible for learning and predicting the binary association of the label given the feature space, modified by all previous binary predictions on the labels in the chain.

iii A two-phase model in which, in the first phase, it detects the request is normal or not and, in the case of an anomalous request, it passes to the second phase of the model to obtain its CAPEC classification using the *BinaryRelevance* class of the *scikit-multilearn* library (Szymański and Kajdanowicz, 2017)

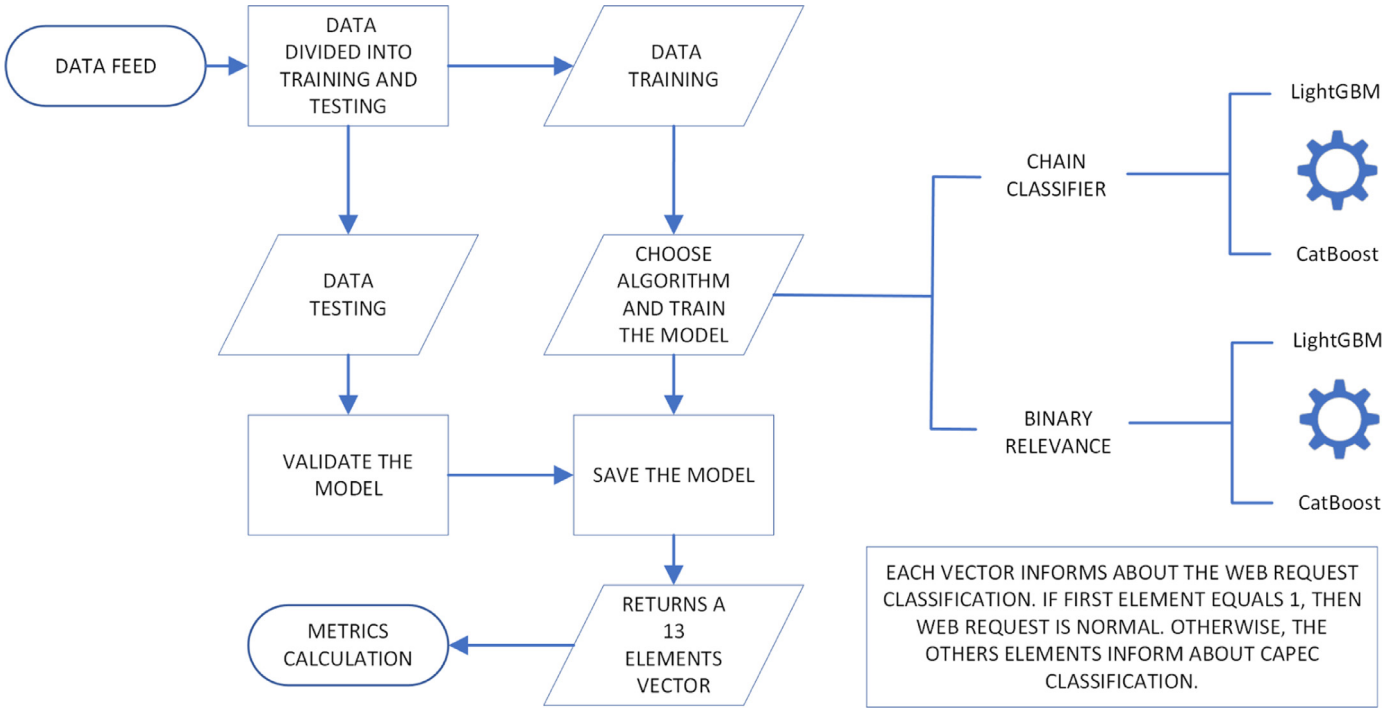

Fig. 4. Single-phase model flowchart.

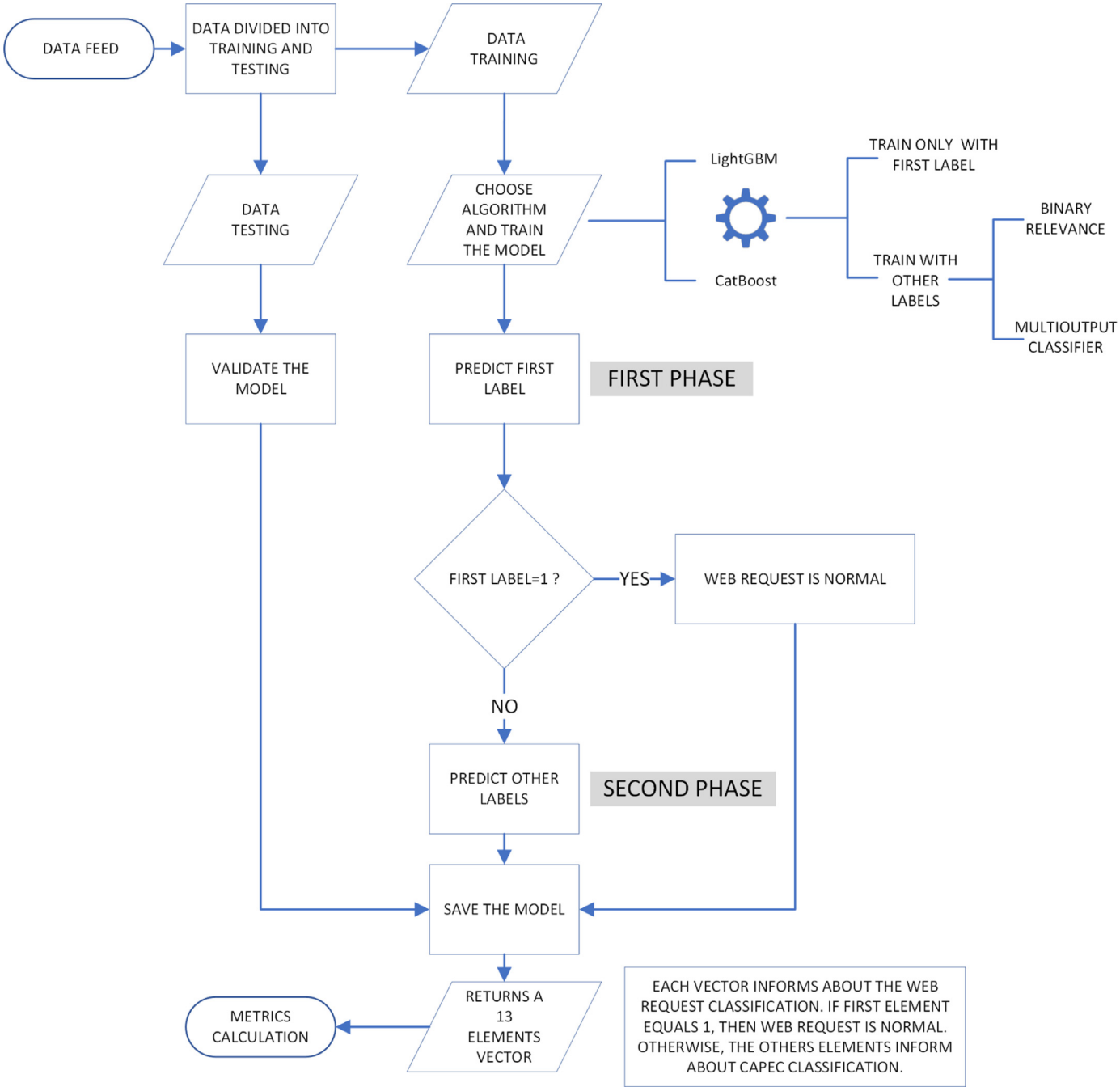

Fig. 5. Two-phase model flowchart.

iv A two-phase model in which, in the first phase, it detects the request is normal or not and, in the case of an anomalous request, it passes to the second phase of the model to obtain its CAPEC classification using the *MultiOutputClassifier* class of the *sklearn.multioutput* module of Scikit-learn library (Pedregosa et al., 2011).

v A customized model in which, the prediction is performed through the specific algorithm(LigthGBM or CatBoost) tuned with the hyperparameters calculated for each CAPEC label.

Figs. 4, 5, and 6 show the flowcharts for the single-phase, two-phase, and customized models respectively.

The combination of models designed and algorithms allows the presentation to the security specialist of information about the attack(s) the system is suffering, including the CAPEC classification(s), as shown in Fig. 7.

5. Results and discussion

The results of the algorithms and model combinations have been evaluated according to the metrics and scenarios discussed in Section 2.1.3. One of the main objectives of the present work is to propose the best combination of algorithms and models able to classify as accurately as possible the different types of attacks in each possible scenario, following the CAPEC classification. Since a web request can be simultaneously classified in more than one CAPEC key, it is necessary to work with multi-label models. The evaluation has been carried out with data from the SR-BH 2020

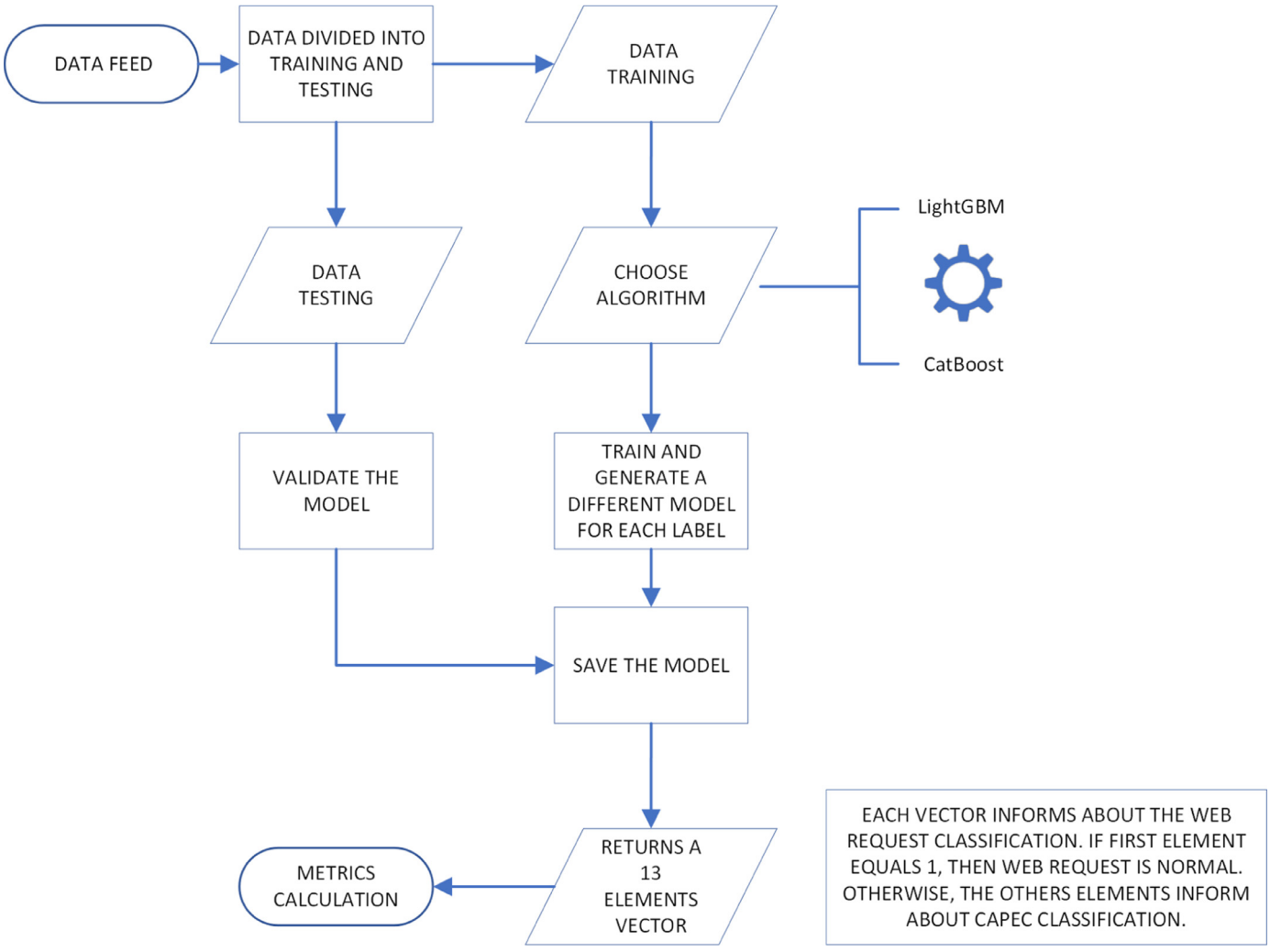

Fig. 6. Customized model flowchart.

```
In [18]: %%time
prediccion([[5.30460883087438,
 1.043082871047808,
 0.623192182505229,
 5.216494102681128,
 0.903506237388453,
 -1.2130669975644175,
 -5.68336728338642,
 -0.9974627487855351,
 -0.5428089419695353]])

CPU times: user 68.6 ms, sys: 0 ns, total: 68.6 ms
Wall time: 21 ms

Out[18]: ([0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 ['272 - Protocol Manipulation', '88 - OS Command Injection'])
```

Fig. 7. Information on attacks, received by the security operator.

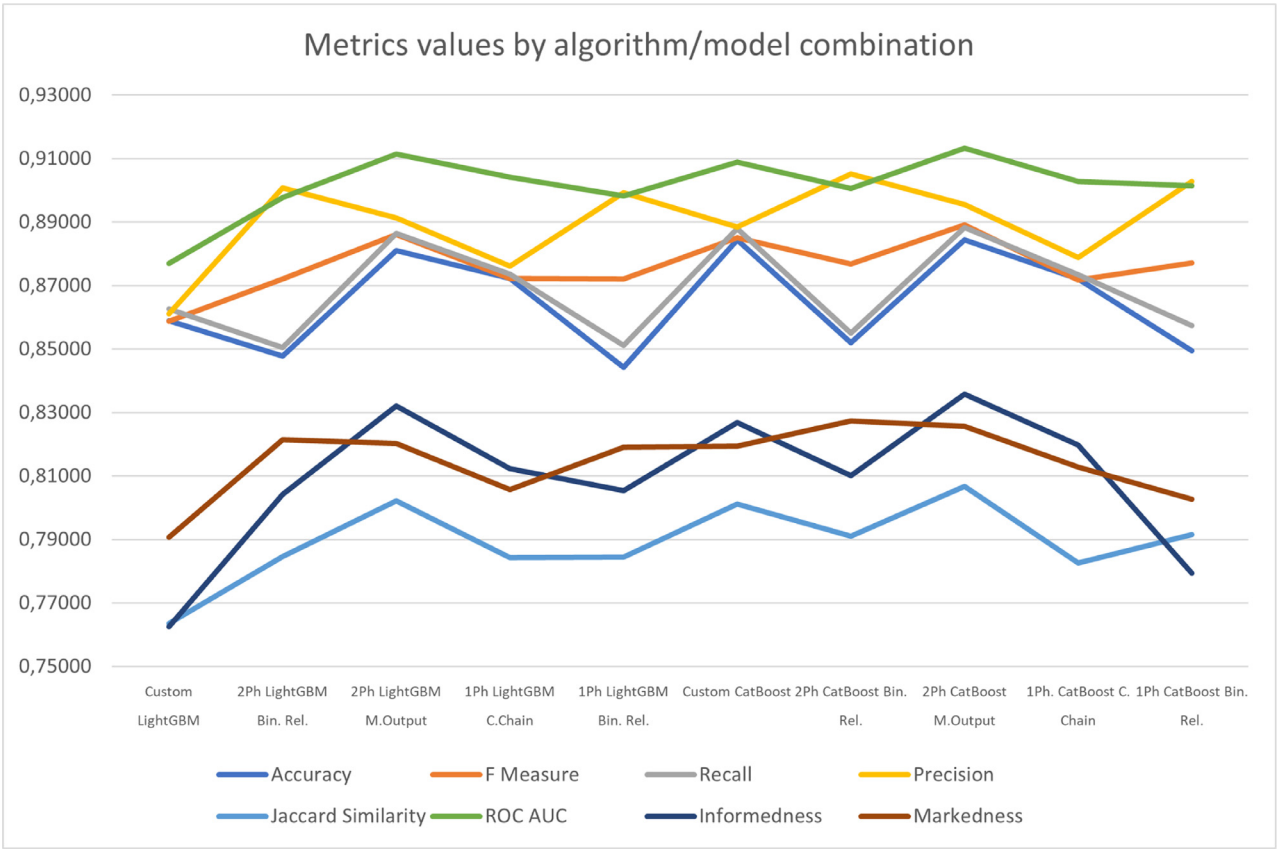

Fig. 8. Metrics by algorithm/model.

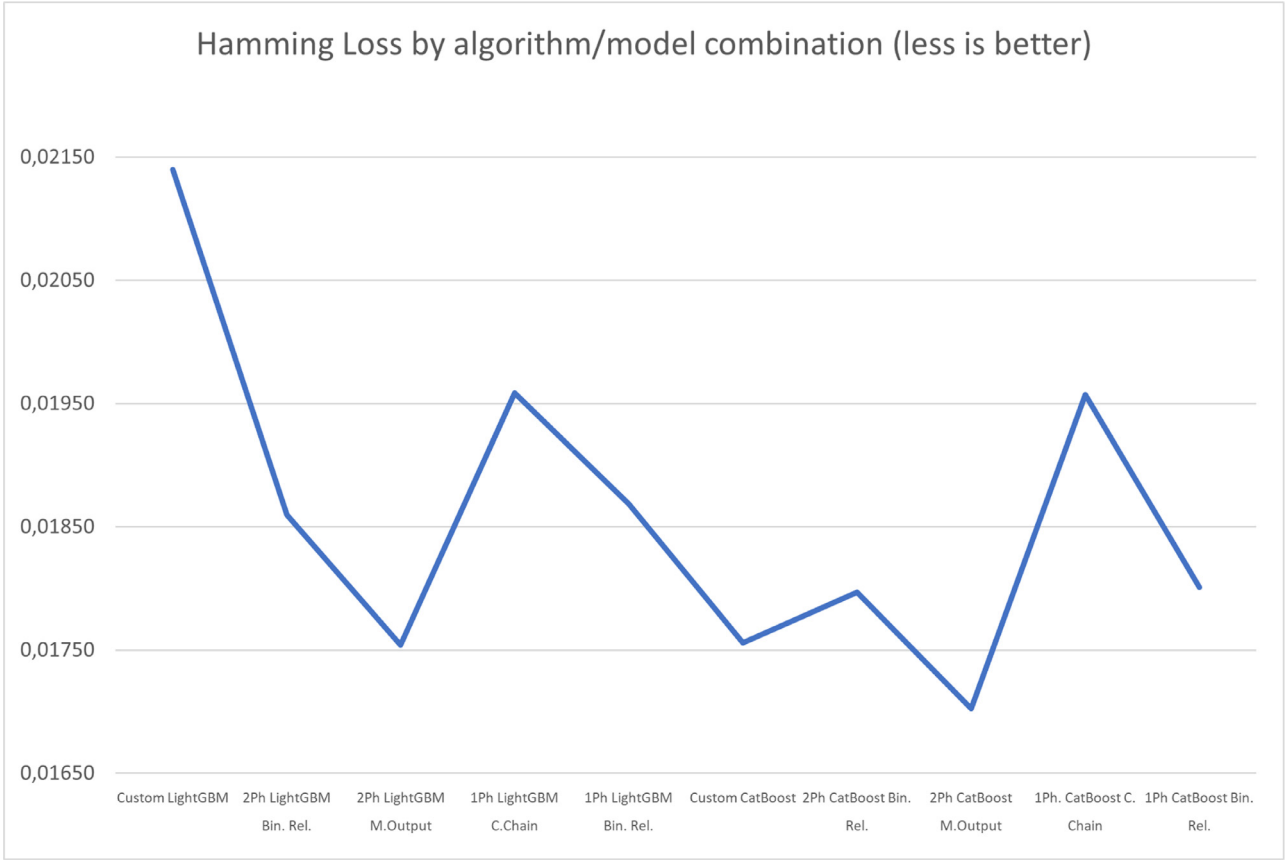

Fig. 9. Hamming Loss by algorithm/model.

Table 6
Summary of metrics by algorithm and model, ordered by recall score.

Model	Accuracy EMR	Precision	Recall	F meas. Avg.	ROC AUC	Hamming Loss	Jaccard Sim.	Inform.	Marked.
Two-phase MultiOutput CatBoost	**0.88445**	0.89557	**0.88829**	**0.88912**	**0.91322**	**0.01703**	**0.80672**	**0.83580**	0.82563
Customized model CatBoost	0.88436	0.88853	0.88790	0.88501	0.90887	0.01756	0.80115	0.82690	0.81941
Two-phase MultiOutput LightGBM	0.88095	0.89137	0.88641	0.88615	0.91139	0.01754	0.80214	0.83200	0.82028
Single-phase Clas.Chain LightGBM	0.87224	0.87610	0.87360	0.87227	0.90421	0.01958	0.78432	0.81238	0.80567
Single-phase Clas.Chain CatBoost	0.87213	0.87876	0.87343	0.87171	0.90281	0.01957	0.78264	0.81973	0.81282
Customized model LightGBM	0.85888	0.86108	0.86270	0.85874	0.87702	0.02140	0.76356	0.76248	0.79066
Single-phase Binary Relevance CatBoost	0.84939	0.90279	0.85734	0.87221	0.90139	0.01801	0.79153	0.77943	0.80261
Two-phase Binary Relevance CatBoost	0.85201	**0.90515**	0.85508	0.87680	0.90055	0.01797	0.79100	0.81007	**0.82741**
Single-phase Binary Relevance LightGBM	0.84419	0.89927	0.85112	0.87204	0.89820	0.01869	0.78439	0.80532	0.81899
Two-phase Binary Relevance LightGBM	0.84782	0.90075	0.85049	0.87216	0.89768	0.01860	0.78468	0.80426	0.82138

Table 7
F measure by CAPEC Classification.

Model	F meas. Avg.	000	272	242	88	126	66	16	310	153	274	194	34	33
Two-phase MultiOutput CatBoost	**0.889**	**0.928**	0.618	**0.862**	**0.884**	**0.855**	**0.855**	**0.999**	0.997	**0.903**	0.999	0.855	**0.509**	**0.905**
Two-phase MultiOutput LightGBM	0.886	0.927	0.600	0.858	0.858	0.852	0.850	**0.999**	**0.999**	0.857	**1.000**	0.849	0.508	0.897
Customized model CatBoost	0.885	0.925	**0.621**	0.860	0.877	0.852	0.853	**0.999**	0.997	0.895	0.999	**0.856**	0.431	0.897
Single-phase Binary Relevance CatBoost	0.877	0.928	0.559	0.790	0.718	0.804	0.843	0.997	0.997	0.842	0.999	0.841	0.359	0.718
Two-phase Binary Relevance CatBoost	0.877	0.928	0.561	0.788	0.716	0.803	0.843	0.997	0.996	0.849	0.998	0.841	0.354	0.718
Single-phase Clas.Chain LightGBM	0.872	0.927	0.560	0.760	0.685	0.790	0.838	0.997	0.996	0.636	**1.000**	0.822	0.362	0.723
Two-phase Binary Relevance LightGBM	0.872	0.927	0.548	0.761	0.677	0.797	0.837	0.998	0.999	0.807	**1.000**	0.827	0.325	0.726
Single-phase Binary Relevance LightGBM	0.872	0.926	0.546	0.762	0.680	0.796	0.838	0.998	0.999	0.806	**1.000**	0.828	0.327	0.726
Single-phase Clas.Chain CatBoost	0.872	0.923	0.549	0.792	0.729	0.797	0.835	**0.999**	0.997	0.834	**1,000**	0.833	0.364	0.724
Customized model LightGBM	0.859	0.907	0.553	0.759	0.671	0.797	0.840	0.996	0.012	0.802	**1.000**	0.825	0.331	0.732

000: Normal 272: Protocol Manipulation 242: Code Injection 88: OS Command Injection 126: Path Traversal 66: SQLi 16: Dictionary based Password Attack 310: Scanning for Vulnerable Software 153: Input Data Manipulation 274: HTTP Verb Tampering 194: Fake the Source of Data 34: HTTP Response Splitting 33 HTTP Request Smuggling.

Table 8
Accuracy by CAPEC Classification.

Model	Acc. global	000	272	242	88	126	66	16	310	153	274	194	34	33
Two-phase MultiOutput CatBoost	**0.884**	**0.919**	**0.994**	**0.995**	**0.998**	**0.994**	**0.914**	**1.000**	**1.000**	**1.000**	**1.000**	**0.982**	**0.983**	**1.000**
Customized model CatBoost	**0.884**	0.914	**0.994**	**0.995**	**0.998**	0.993	0.913	**1.000**	**1.000**	**1.000**	**1.000**	**0.982**	0.982	**1.000**
Two-phase MultiOutput LightGBM	0.881	0.917	0.993	0.995	**0.998**	0.993	0.911	**1.000**	**1.000**	0.999	**1.000**	**0.982**	**0.983**	**1.000**
Single-phase Clas.Chain LightGBM	0.872	0.917	0.993	0.992	0.995	0.990	0.903	**1.000**	**1.000**	0.998	**1.000**	0.978	0.979	0.999
Single-phase Clas.Chain CatBoost	0.872	0.913	0.913	0.993	0.996	0.991	0.900	**1.000**	0.990	0.999	**1.000**	0.980	0.980	0.999
Customized model LightGBM	0.859	0.891	0.993	0.992	0.995	0.991	0.911	**1.000**	0.990	0.999	**1.000**	0.979	0.981	0.999
Two-phase Binary Relevance CatBoost	0.852	**0.919**	0.993	0.993	0.996	0.992	0.913	**1.000**	**1.000**	0.999	**1.000**	0.981	0.981	0.999
Single-phase Binary Relevance CatBoost	0.849	**0.919**	0.993	0.993	0.996	0.992	0.913	**1.000**	**1.000**	0.999	**1.000**	0.981	0.981	0.999
Two-phase Binary Relevance LightGBM	0.848	0.917	0.993	0.992	0.995	0.991	0.910	**1.000**	**1.000**	0.999	**1.000**	0.980	0.981	0.999
Single-phase Binary Relevance LightGBM	0.844	0.917	0.993	0.992	0.995	0.991	0.910	**1.000**	**1.000**	0.999	**1.000**	0.980	0.980	0.999

000: Normal 272: Protocol Manipulation 242: Code Injection 88: OS Command Injection 126: Path Traversal 66: SQLi 16: Dictionary based Password Attack 310: Scanning for Vulnerable Software 153: Input Data Manipulation 274: HTTP Verb Tampering 194: Fake the Source of Data 34: HTTP Response Splitting 33 HTTP Request Smuggling.

Table 9
Best scores obtained for LightGBM and CatBoost Algorithms.

Metric	LightGBM	CatBoost
Accuracy EMR	0.88095	**0.88445**
Precision	0.90075	**0.90515**
Recall	0.88641	**0.88829**
F measure	0.88615	**0.88912**
ROC AUC	0.91139	**0.91322**
Hamming Loss	0.01754	**0.01703**
Jaccard Similarity	0.80214	**0.80672**
Informedness	0.83200	**0.83580**
Markedness	0.82138	**0.82741**

dataset, created specifically to allow this type of multi-label classification. 70% of web requests present in the dataset have been used to train the different algorithms, while the remaining 30% of web requests have been used to validate the models generated.

Table 6 presents a summary of the different metrics obtained for each algorithm/model combination, ordered by recall score.

Table 7 details the F measure obtained for each algorithm/model combination in the different CAPEC classifications.

Table 8 details the Accuracy obtained for each algorithm/model combination in the different CAPEC classifications.

As can be seen in tables 6, 7, 8, the combination of the CatBoost algorithm and a two-phase model in which the MultiOutputClassifier class of the Scikit-learn library is applied is the one that obtains the best results in practically all the metrics. This model obtains a strict Accuracy (EMR) of 0.8844, an average F measure of 0.8891, an AUC ROC of 0.9132, and a Hamming Loss of 0.01703.

The two-phase models with the MultiOutputClassifier class are clearly superior to the other models, regardless of the algorithm (LightGBM or CatBoost) included in them: using LightGBM, their EMR is 0.88095, their average F measure is 0.8862, their ROC AUC is 0.9114 and their Hamming Loss is 0.01754.

See Figs. 8, 9 for a detail of the metrics obtained by each algorithm/model combination.

Although all algorithm/model combinations obtain good F-Score and Accuracy scores in each CAPEC classification, it becomes evident that the combination of the CatBoost algorithm and the two-phase model with the MultiOutpoutClassifier class yields the best results, obtaining the highest Accuracy scores in all CAPEC classifications and the highest F-Score in 9 of the 13 CAPEC classifications.

The superiority of the CatBoost algorithm over LightGBM can also be clearly seen, regardless of the model with which they are combined. See Table 9 for a comparison of the best scores obtained by LightGBM and CatBoost in the various metrics evaluated. Note that in the case of the Hamming Loss metric, a lower score is indicative of better performance, as indicated in Section 2.1.3.1.

Following the work of Antunes and Vieira (2015), which recommends the most appropriate metrics according to the type of scenario, the best algorithm/model combination has been selected based on the recommended metric for each different scenario as detailed in 2.1.3.2. Table 10 shows the recommended metric for each of the four scenarios, the top three values obtained in this metric, as well as the algorithm/model combinations that obtained this value.

From the results in Table 10, it is concluded that the combination of the CatBoost algorithm and the two-phase model with the *MultiOutpoutClassifier* class, is the one that obtains the best performance with the recommended metrics in the three most critical scenarios analyzed (*Very high critical scenario, Heightened-critical scenario*, and *Medium-critical scenario*). Only in the *Low-critical scenario*, this algorithm/model combination is outperformed by the combination of CatBoost and the two-phase model with the *Bina-*

Table 10
Recommended metrics for scenario and top three algorithm/model combination.

Scenario	Metric	Best values	Algorithm / Model Combination
Business-critical	Recall	0.88829	Two-phase MultiOutput CatBoost
		0.88790	Customized model CatBoost
		0.88641	Two-phase MultiOutput LightGBM
Heightened-critical applications	Informedness	0.83580	Two-phase MultiOutput CatBoost
		0.83200	Two-phase MultiOutput LightGBM
		0.82690	Customized model CatBoost
Best effort	F-Measure	0.88912	Two-phase MultiOutput CatBoost
		0.88615	Two-phase MultiOutput LightGBM
		0.88501	Customized model CatBoost
Minimum effort	Markedness	0.82741	Two-phase Binary Relevance CatBoost
		0.82563	Two-phase MultiOutput CatBoost
		0.82138	Two-phase Binary Relevance LightGBM

ryRelevance class, obtaining the second-best score. From the review of the scores obtained in the different metrics, it can be concluded that the Two-phase MultiOutput CatBoost model is adequate for all the scenarios considered.

6. Conclusions

In this work we have presented the SR-BH 2020 multi-label dataset, which includes a set of 13 different labels, providing information about the normality of each web request and its possible classification into 12 different CAPEC categories.

A new way to give a numerical value to the alphanumeric strings and symbols that constitute the different fields that conform a web request, by calculating the average of the sum of the ASCII values of each of the characters in each field, is proposed; this numerical value, calculated easily and quickly, allows the extraction of features and the training of the different machine learning models.

We have also designed and evaluated different multi-label classification models, using modules and classes from the scikit-learn and scikit-multilearn libraries. Two leading algorithms in the field of machine learning have been tested with these models: LightGBM and CatBoost. The results obtained by our experiments show a clear superiority of the combination of the CatBoost algorithm and the two-phase model with the MultiOutputClassifier module of the scikit-learn library, in multi-label classification tasks. In future work, the possibility of executing automatic remediation actions, based on CAPEC attack patterns, could be considered.

Consideration should also be given to the possibility of generating a new dataset with data from traffic originating from communications between SOAP or REST web services or even data from web API communication. The labeling of the attacks performed on these services and their CAPEC classification would allow the application of the combinations of algorithms and models generated in this work to new datasets as well as to determine the possible generalization of results to other contexts.

In addition, due to the excellent results obtained in all the scenarios analyzed by the Two-phase MultiOutput CatBoost model, more effort and research could be devoted to the development and improvement of this model to achieve its integration into commercial or open-source web application protection tools that can be used in all types of scenarios with different levels of criticality.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Tomás Sureda Riera: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Juan-Ramón Bermejo Higuera:** Conceptualization, Methodology, Validation, Writing – review & editing, Visualization, Supervision. **Javier Bermejo Higuera:** Conceptualization, Methodology, Validation, Writing – review & editing, Visualization, Supervision. **José-Javier Martínez Herraiz:** Validation, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Juan-Antonio Sicilia Montalvo:** Conceptualization, Methodology, Validation, Writing – review & editing, Visualization, Supervision.

References

Antunes, N., Vieira, M., 2015. On the metrics for benchmarking vulnerability detection tools. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 505–516. doi:10.1109/DSN.2015.30.

Auxilia, M., Tamilselvan, D., 2010. Anomaly detection using negative security model in Web application. In: 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp. 481–486. doi:10.1109/CISIM.2010.5643461.

Bermejo Higuera, J.R., 2013. Metodología de evaluación de herramientas de análisis automático de seguridad de aplicaciones web para su adaptación en el ciclo de vida de desarrollo. Universidad Nacional Educación a Distancia (UNED). http://e-spacio.uned.es/fez/eserv/tesisuned:IngInd-Jrbermejo/BERMEJO_HIGUERA_Juan_Ramon_Tesis.pdf

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and regression trees.(the wadsworth statistics/probability series), belmont. CA: Wadsworth.

Brugger T.. KDD Cup '99 dataset (Network Intrusion) considered harmful (KDnuggets News 07:18, item 4, Features). 2007. https://www.kdnuggets.com/news/2007/n18/4i.html.

Büyükçakir, A., Bonab, H., Can, F., 2018. A novel online stacked ensemble for multi-label stream classification. In: International Conference on Information and Knowledge Management, Proceedings. Association for Computing Machinery, pp. 1063–1072. doi:10.1145/3269206.3271774.

Charte, F., Rivera, A.J., Del Jesus, M.J., Herrera, F., 2015. MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. Knowl Based Syst 89, 385–397. doi:10.1016/j.knosys.2015.07.019.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., et al., 2002. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 16, 321–357. doi:10.1613/jair.953.

Cheng, W., Hüllermeier, E., 2009. Combining instance-based learning and logistic regression for multilabel classification. Mach Learn 76 (2), 211–225. doi:10.1007/s10994-009-5127-5.

Cisco. Cisco Annual Internet Report (2018–2023). 2018. https://bit.ly/3a4a1H4.

Dang, Q.V., François, J., 2018. Utilizing attack enumerations to study SDN/NFV vulnerabilities. In: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pp. 356–361. doi:10.1109/NETSOFT.2018.8459961.

Devi, R., Abualkibash, M., 2019. Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets - a review paper. International Journal of Computer Science and Information Technology 11, 65–80. doi:10.5121/ijcsit.2019.11306.

Díaz, G., Bermejo, J.R., 2013. Static analysis of source code security: assessment of tools against SAMATE tests. Inf Softw Technol 55 (8), 1462–1476. doi:10.1016/j.infsof.2013.02.005.

Dorogush, A.V., Ershov, V., Gulin, A., 2018. Catboost: gradient boosting with categorical features support. arXiv preprint arXiv:181011363.

Dubey, S.K., 2016. An evaluation of java applications using security requirements. International Journal of Recent Trends in Engineering & Research Issue 02 (08). 2455–1457

Gartner. Runtime Application Self-Protection (RASP) - Gartner IT Glossary. 2022. http://www.gartner.com/it-glossary/runtime-application-self-protection-rasp.

Gouk, H., Pfahringer, B., Cree, M., 2016. Learning distance metrics for multi-label classification. In: Durrant, R.J., Kim, K.E. (Eds.), Proceedings of The 8th Asian Conference on Machine Learning. In: Proceedings of Machine Learning Research, volume 63. PMLR, The University of Waikato, Hamilton, New Zealand, pp. 318–333. https://proceedings.mlr.press/v63/Gouk8.html

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., et al., 2002. Gene selection for cancer classification using support vector machines. Mach Learn 46 (1–3), 389–422. doi:10.1023/A:1012487302797.

Haldar, V., Chandra, D., Franz, M., 2005. Dynamic taint propagation for Java. In: 21st Annual Computer Security Applications Conference (ACSAC'05), volume 2005. IEEE, Tucson, Arizona, pp. 303–311. doi:10.1109/CSAC.2005.21.

Halfond, W.G.J., Orso, A., Manolios, P., 2008. WASP: Protecting web applications using positive tainting and syntax-Aware evaluation. IEEE Trans. Software Eng. 34 (1), 65–81. doi:10.1109/TSE.2007.70748.

Hancock, J.T., Khoshgoftaar, T.M., 2020. Catboost for big data: an interdisciplinary review. J Big Data 7 (1), 94. doi:10.1186/s40537-020-00369-8.

Hiscox, 2021. Don't let cyber be a game of chance. Hiscox Cyber Readiness Report 2021. Technical Report. Hiscox. https://bit.ly/3AbiJ0t

Jin, X., Cui, B., Yang, J., Cheng, Z., 2018. Payload-based Web attack detection using deep neural network. In: Lecture Notes on Data Engineering and Communications Technologies, volume 12. Springer, Cham, pp. 482–488. doi:10.1007/978-3-319-69811-3_44.

Johari, R., Sharma, P., 2012. A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection. In: 2012 International Conference on Communication Systems and Network Technologies, pp. 453–458. doi:10.1109/CSNT.2012.104.

Kanakogi, K., Washizaki, H., Fukazawa, Y., Ogata, S., Okubo, T., Kato, T., Kanuka, H., Hazeyama, A., Yoshioka, N., 2021. Tracing CVE vulnerability information to CAPEC attack patterns using natural language processing techniques. Information 12 (8). doi:10.3390/info12080298.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y., 2017. LightGBM: a highly efficient gradient boosting decision tree. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp. 3149–3157.

Kozik, R., Choraś, M., Rafał, R., Witold, H., 2015. Patterns Extraction Method for Anomaly Detection in HTTP Traffic. In: Herrero, A., Baruque, B., Sedano, J., Quintián, H., Corchado, E. (Eds.), International Joint Conference. Springer International Publishing, Cham, pp. 227–236.

Kruegel, C., Vigna, G., 2003. Anomaly detection of Web-based attacks. In: Proceedings of the ACM Conference on Computer and Communication Security (CCS). ACM Press, Washington, DC, pp. 251–261.

Krügel, C., Toth, T., Kirda, E., 2002. Service specific anomaly detection for network intrusion detection. In: Proceedings of the 2002 ACM Symposium on Applied Computing. Association for Computing Machinery, New York, NY, USA, pp. 201–208. doi:10.1145/508791.508835.

Liang, J., Zhao, W., Ye, W., 2017. Anomaly-based Web attack detection: a deep learning approach. In: Proceedings of the 2017 VI International Conference on Network, Communication and Computing. Association for Computing Machinery, New York, NY, USA, pp. 80–85. doi:10.1145/3171592.3171594.

Lichman M.. 1999 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory. 2000. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset.

Mac, H., Truong, D., Nguyen, L., Nguyen, H., Tran, H.A., Tran, D., 2018. Detecting attacks on Web applications using autoencoder. In: Proceedings of the Ninth International Symposium on Information and Communication Technology. Association for Computing Machinery, New York, NY, USA, pp. 416–421. doi:10.1145/3287921.3287946.

Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S., 2012. An extensive experimental comparison of methods for multi-label learning. Pattern Recognit 45 (9), 3084–3104. doi:10.1016/J.PATCOG.2012.03.004.

Mahoney, M.V., Chan, P.K., 2003. An analysis of the 1999 DARPA/lincoln laboratory evaluation data for network anomaly detection. In: Vigna, G., Kruegel, C., Jonsson, E. (Eds.), Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 220–237.

McHugh, J., 2000. Testing intrusion detection systems: A Critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans Inf Syst Secur 3 (4), 262–294. doi:10.1145/382912.382923.

Montes, N., Betarte, G., Martínez, R., Pardo, A., 2021. Web Application Attacks Detection Using Deep Learning. In: Tavares, J.M.R.S., Papa, J.P., González Hidalgo, M. (Eds.), Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer International Publishing, Cham, pp. 227–236.

Moosa, A., 2010. Artificial neural network based web application firewall for SQL injection. International Journal of Information, Control and Computer Sciences 3.0 (4). doi:10.5281/zenodo.1329474.

Moustafa, N., Slay, J., 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6. doi:10.1109/MilCIS.2015.7348942.

Oliveira, N., Praça, I., Maia, E., Sousa, O., 2021. Intelligent cyber attack detection and classification for network-Based intrusion detection systems. Applied Sciences 11 (4). doi:10.3390/app11041674.

OWASP. SQL Injection Bypassing WAF - OWASP. https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF.

Pan, Y., Sun, F., Teng, Z., White, J., Schmidt, D.C., Staples, J., Krause, L., 2019. Detecting web attacks with end-to-end deep learning. Journal of Internet Services and Applications 2019 10:1 10 (1), 1–22. doi:10.1186/S13174-019-0115-X.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A., 2018. CatBoost: Unbiased Boosting with Categorical Features. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp. 6639–6649.

Protić, D., 2018. Review of KDD cup '99, NSL-KDD and Kyoto 2006+ datasets. Vojnotehnicki glasnik 66 (3), 580–596. doi:10.5937/vojtehg66-16670.

Raïssi, C., Brissaud, J., Dray, G., Poncelet, P., Roche, M., Teisseire, M., 2007. Web Analyzing Traffic Challenge: Description and Results. In: ECML/PKDD'2007 Discovery Challenge, France, p. 6. https://hal-lirmm.ccsd.cnrs.fr/lirmm-00168955

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2009. Classifier chains for multi-label classification. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, pp. 254–269.

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2011. Classifier chains for multi-label classification. Mach Learn 85 (3), 333–359. doi:10.1007/s10994-011-5256-5.

Resende, P.A.A., Drummond, A.C., 2018. Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. Security and Privacy 1 (4), e36. doi:10.1002/spy2.36.

Ristic I.. Protocol-Level Evasion of Web Application Firewalls – Network Security Blog | Qualys, Inc. 2022. https://blog.qualys.com/ssllabs/2012/07/25/protocol-level-evasion-of-web-application-firewalls.

Ross Quinlan, b.J., Kaufmann Publishers, M., Salzberg, S.L., 1994. C4.5: Programs for machine learning by J. Ross Quinlan. Morgan Kaufmann Publishers, inc., 1993. Machine Learning 1994 16:3 16 (3), 235–240. doi:10.1007/BF00993309.

Schapire, R.E., Singer, Y., 2000. Boostexter: A Boosting-based system for text categorization. Mach Learn 39 (2), 135–168. doi:10.1023/A:1007649029923.

Shi, H., 2007. Best-first Decision Tree Learning. The University of Waikato, Hamilton, New Zealand. https://hdl.handle.net/10289/2317

Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security 31 (3), 357–374. doi:10.1016/j.cose.2011.12.012.

Siddique, K., Akhtar, Z., Khan, F.A., Kim, Y., 2019. KDD Cup 99 data sets: a perspective on the role of data sets in network intrusion detection research. Computer (Long Beach Calif) 52 (2), 41–51. doi:10.1109/MC.2018.2888764.

Steiner, S., de Leon, D.C., Alves-Foss, J., 2017. A structured analysis of SQL injection runtime mitigation techniques. In: Proceedings of the Annual Hawaii International Conference on System Sciences, volume 2017-January, pp. 2887–2895. doi:10.24251/hicss.2017.349.

Sureda Riera, T., Bermejo Higuera, J.R., Bermejo Higuera, J., Martínez Herraiz, J.J., Sicilia Montalvo, J.A., 2020. Prevention and fighting against web attacks through anomaly detection technology. a systematic review. Sustainability 12 (12). doi:10.3390/su12124945.

Swets, J.A., 1996. Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US.

Szymański, P., Kajdanowicz, T., 2017. A scikit-based python environment for performing multi-label classification. ArXiv e-prints.

Tama, B.A., Nkenyereye, L., Islam, S.M.R., Kwak, K.S., 2020. An enhanced anomaly detection in web traffic using a stack of classifier ensemble. IEEE Access 8, 24120–24134. doi:10.1109/ACCESS.2020.2969428.

Tan, H.H., Hoai, T.V., 2021. Web Application Anomaly Detection Based On Converting HTTP Request Parameters To Numeric. In: 2021 15th International Conference on Advanced Computing and Applications (ACOMP), pp. 93–97. doi:10.1109/ACOMP53746.2021.00019.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A Detailed Analysis of the KDD CUP 99 Data Set. In: Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications. IEEE Press, pp. 53–58.

Tekerek, A., 2021. A novel architecture for web-based attack detection using convolutional neural network. Computers and Security 100, 102096. doi:10.1016/j.cose.2020.102096.

Torrano-Gimenez, C., Perez-Villegas, A., Alvarez, G., 2009. A Self-learning Anomaly-Based Web Application Firewall. In: Herrero, A., Gastaldo, P., Zunino, R., Corchado, E. (Eds.), Computational Intelligence in Security for Information Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 85–92.

Truong, D., Tran, D., Nguyen, L., Mac, H., Tran, H.A., Bui, T., 2019. Detecting Web attacks using stacked denoising autoencoder and ensemble learning methods. In: Proceedings of the Tenth International Symposium on Information and Communication Technology. Association for Computing Machinery, New York, NY, USA, pp. 267–272. doi:10.1145/3368926.3369715.

Tsoumakas, G., Katakis, I., 2009. Multi-Label classification: an overview. Int. J. Data Warehouse. Min. 3, 1–13. doi:10.4018/jdwm.2007070101.

Van Rijsbergen, C.J., 1979. Information Retrieval, 2nd Butterworth-Heinemann Newton, MA, USA.

Vu, Q.H., Ruta, D., Cen, L., 2019. Gradient boosting decision trees for cyber security threats detection based on network events logs. In: Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019. Institute of Electrical

and Electronics Engineers Inc., pp. 5921–5928. doi:10.1109/BigData47090.2019.9006061.
Wang, H., Li, Z., Huang, J., Hui, P., Liu, W., Hu, T., Chen, G., 2020. Collaboration based multi-label propagation for fraud detection. In: IJCAI International Joint Conference on Artificial Intelligence, volume 2021-Janua. International Joint Conferences on Artificial Intelligence, pp. 2477–2483. doi:10.24963/ijcai.2020/343.
Zhang, M.L., Zhou, Z.H., 2007. ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognit 40 (7), 2038–2048. doi:10.1016/j.patcog.2006.12.019.
Zhang, M.L., Zhou, Z.H., 2014. A review on multi-label learning algorithms. IEEE Trans Knowl Data Eng 26 (8), 1819–1837. doi:10.1109/TKDE.2013.39.

Tomás Sureda Riera holds a degree in psychology from Universidad Nacional de Educación a Distancia (UNED) and a master's degree in informatics security from the Universidad Internacional de la Rioja (UNIR), in 2002 and 2017, respectively. His master research thesis was qualified with honors and won an ISACA (Valencia Chapter) award for best master research thesis in 2017. His research interests are: security operations online, web and databases security, penetration testing and data science applied to cyber security.

Juan Ramón Bermejo Higuera received the M.Sc. degree in computer engineering and the Ph.D. degree from the Distance Education National University, Spain, in 1998 and 2014, respectively. He is currently the Chief of the Cybersecurity Unit, National Institute of Aerospace Techniques, Spain. He is also a Professor of applications security in the International University of La Rioja and also an Associate Professor of the Ciberdefence Master Science degree with the Alcala de Henares University. He has authored several publications. His research interests include cybersecurity and cyber defense.

Javier Bermejo Higuera received the B.S. degree from Alcala University and the Ph.D. degree from the Army Polytechnic School. He is currently a Professor with the Escuela Superior de Ingeniería y Tecnología, Universidad Internacional de La Rioja. He has authored several publications. His research interests include the fields of software security, cybersecurity, and malware analysis.

Jose Javier Martínez-Herráiz has been an Associate Professor with the Department of Computer Science (Artificial Intelligence Area), University of Alcalá, Alcalá de Henares, Spain (since 1994), where he is the Rector's Delegate for Electronic Administration and Security. He was with private telecommunication business companies (Spain and Italy) as a Software Analyst, Project Manager, and Consultant between 1988 and 1999. He was the Director of the Computer Science Department between 2008 and 2011. He has collaborated extensively with Spanish law enforcement agencies and cybersecurity companies, and his work was awarded with Order of Merit of the Police Forces (2011 and 2015). He has practical experience in software development, technology and modeling, methodologies for software projects, planning and management, software maintenance, e-learning, gamification technology, and cybersecurity. He received the degree in computer science from the Polytechnic University of Madrid, Madrid, Spain, and the Ph.D. degree from the University of Alcalá in 2004.

Juan Antonio Sicilia Montalvo received the B.S. and Ph.D. degrees from the Universidad de Zaragoza. He is currently a Professor with the Escuela Superior de Ingeniería y Tecnología, Universidad Internacional de La Rioja. His research interests include combinatorial optimization, computer security, software development based on mathematical algorithms, numerical methods, and heuristic techniques for solving engineering problems.