Automated and Improved Detection of Cyber Attacks via an Industrial IDS Probe.

Almamy Touré, Youcef Imine, Thierry Delot, Antoine Gallais, Alexis Semnont, Robin Giraudo

Automated and Improved Detection of Cyber Attacks via an Industrial IDS Probe

Almamy Touré[1,2], Youcef Imine[1], Thierry Delot[1], Antoine Gallais[1], Alexis Semnont[2] and Robin Giraudo[2]

[1] Univ. Polytechnique Hauts-de-France, INSA Hauts-de-France, LAMIH, CNRS, UMR 8201, F-59313 Valenciennes, France.
[2]IBM Security France.

Abstract

Network flow classification allows to distinguish normal flows from deviant behaviors. However, given the diversity of the approaches proposed for intrusion detection via IDS probes, an adequate fundamental solution is required. Indeed, most of existing solutions address a specific context which does not allow to assess the efficiency of the proposed models on a different context. Therefore, we propose in this paper an approach for malicious flow detection based on One Dimensional Convolutional Neural Networks (1D-CNN). Our solution extracts features based on the definition of network flows. Thus, it can be common to any network flow classification model. This feature engineering phase is coupled to CNN's feature detector in order to provide an efficient classification approach. To evaluate its performance, our solution has been evaluated on two different datasets (a recent dataset extracted from a real IBM industrial context and the NSL-KDD dataset that is widely used in the literature). Moreover, a comparison with existing solutions has been provided to NSL-KDD dataset. Attacks in both datasets have been defined using the globally-accessible knowledge base of adversary tactics and techniques MITRE framework. The evaluation results have shown that our proposed solution allows an efficient and accurate classification in both datasets (with an accuracy rate of 94% at least). Moreover, it outperforms existing solutions in terms of classification metrics and execution time as well.

Keywords: Intrusion Detection System, Deep Learning, MITRE ATT&CK, Features Engineering, Cyber attack.

1 Introduction

Every day, new applications are connected to the Internet which increasingly complicates the management of the network in an information system traffic. Indeed, this growing number of applications diversifies the attack techniques targeting information systems [1]. Collecting and analyzing network flows facilitates traffic management, ensures better network trending and helps to identify known intrusions and new ones [2]. Intrusion detection via IDS probes are mainly based on static correlation rules that are manually updated. Moreover, it usually does not take into account neither recent attack scenarios nor the detection of zero-day vulnerability exploitation attempts [3]. Consequently, organizations must now rely on dynamic security solutions to adapt to the changing nature of current attacks. The diversity of applications used and obfuscation techniques also limits the detection capabilities of IDS probes.

Although, the use of dynamic attack detection approaches with Machine Learning (ML) and Deep Learning (DL) algorithms has allowed to improve the identification and classification on these attacks [4]. However, the proposed solutions are not generalized because of the engineering of the features manner used for the detection and classification of attacks. Indeed, it is clearly noted in the literature that these characteristics vary from one solution to another and therefore unlikely to be reproduced elsewhere. In addition, the datasets used for the testing phases do not have a benchmark to better evaluate the accuracy of the proposed solutions.

In view of the mentioned improvement axes and weaknesses observed in the state of the art, in this paper we present as part of the improvement of an existing IDS solution at IBM, a new solution for malicious network flows detection and classification in which:

- we propose a standard and producible engineering of relevant and necessary feature classes that can be extracted and used in any network flow classifier,
- we propose a deep learning model that combines our feature extractor and the CNN's feature detector to achieve an efficient classification model,
- we perform an evaluation of our model on two datasets from different contexts while validating their content using MITRE ATT&CK framework,
- we propose a comparison between our model, machine learning approaches and deep learning solutions defined in the state of the art.

The rest of the paper is organized as follows. In section 2, we discuss the recent intrusion detection works in the literature. We describe our proposed solution in Section 3 and present its performance evaluation results in Section 4. Finally, we conclude in Section 5.

2 Related Work

Intrusion Detection Systems (IDS) aim to detect malicious activities that could compromise a Host (HIDS) or a Network (NIDS) [5]. Obfuscation techniques

such as encryption, steganography, tunneling, anonymization, mutation, morphing, physical obfuscation allow for better data protection of information system and harden the traditional methods of incident detection via network flow collectors [6]. Thus, the application of automatic and deep learning techniques allows better detection of intrusions.

Different research works have considered the use of ML techniques for intrusion detection including, e.g., Decision Trees [7] , Random Forest [8] , Naives Bayes (NB) [9] and Hidden Naïve Bayes (HNB) [10]. These different approaches aim to detect intrusions, by classifying network flows by scenario, via ML algorithms.

Deep Learning extends ML principles, relies on hidden layers to learn in depth and process information and allows to process large volumes of data. In [11], a sequential classifier for decreasing the false positive rate in this large amount of data to process is proposed. They used Artificial Neural Network (ANN) to reduce false positives and negatives. The accuracy was measured over the KDD99 dataset and varies according to the number of ANN classifiers chosen. In [12], an approach using several layers with hierarchy for complex feature extraction is defined to improve the effectiveness of Artificial Neural Networks (ANN) in NIDS. The solution can be evaluated by comparing traditional supervised ML classifiers and ANN techniques. However, the measured accuracy of the model varies highly upon application to two distinct datasets (KDDCup 99 and UNSW-NB15), hence the need to investigate more generic solutions able to homogeneously perform on a large variety of datasets.

Other approaches focus on the use of Recurrent Neural Networks (RNN), models adapted to sequential data. In [4], they propose an RNN model that aims to improve both the accuracy of intrusion detection and the ability to recognize the type of intrusion. The reduction of the learning time and the decrease of the overlearning, the optimization of the search for hyper-parameters can be posed as axes of improvement of the proposed approach, in order to improve the proposed algorithm and the accuracy rate.

Binary classification and multiclass classification, allow to outperform other deep learning (ANN) and ML approaches (J48, Random Forest, SVM), by reducing the learning time and the overlearning. Using such approach imposes to finely determine hyper-parameters in order to reach a good accuracy.

Deep neural networks based on a Bidirectional Long Short-Term Memory (BLSTM) can increase this accuracy by circulating the input data in both directions. In [13], the most optimal hyper-parameters are identified and tested on the CICIDS2017 dataset, while Random Forest and Principal Component Analysis (PCA) algorithms allow to select features. Feature selection is indeed a matter of prime importance. In [14], a two-layer approach achieves both a spatial and temporal feature extraction from raw data with Convolutional Neural Networks (CNN) and LSTM respectively. However, the accuracy rate decreases for unbalanced data.

A similar combination of CNN and LSTM model has been used to serialize TCP/IP packets in a predetermined time range as a traffic model in [15].

Normal and abnormal network traffic is categorized and labeled for supervised learning in 1D-CNN and applied to the UNSW-NB15 IDS dataset, using CUDA GPU acceleration to reduce time consumption. Combinations of methods can be effective, in the classification.Deep Learning has also been combined with binary algorithms (e.g., Binary Algorithm, Binary Genetic Algorithm, Binary Gravitational Search Algorithm, Binary Bat Algorithm) as optimizers in order to increase the accuracy of detection and reduce the error rate [16]. Although inputs of the proposed hybrid IDS anomaly classification method are defined with eighty flow features from the CICIDS2017 dataset, the choice of these features remains specific to each situation.

Overall, our literature review emphasizes the lack of generic solutions to perform classification. Existing approaches remain specific to each dataset and so do the accuracy rate and the rate of false positives. The extraction and definition of features (feature extraction algorithms, extraction of spatial, temporal or vector features by deep learning) used by the proposed algorithms also highly depend on the targeted application case.

However, as observed, the solutions designed for a given dataset can hardly be transposed to others, especially given the constantly increasing diversity and complexity of cyber attacks. The MITRE framework allows to categorize the attack according to the Technical Tactics and Procedures (TTPs) their use [17]. This globally recognized knowledge base provides broad visibility into the detection perimeter. It also allows the evaluation of the dataset used and provides guidance for future research based on expanding IDS coverage. For all these reasons, we propose to use this framework to enhance the network flows we collected for further classification, thus easing the application of our approach in other situations.

3 Our Proposed Approach

In this section, we present our supervised learning approach for network intrusion detection. Network events in our approach are classified based on the features of network flows. Our approach uses CNN to achieve a fast and efficient classification and can be applied in any network context.

Thus, as we can see in the Fig.1 the proposed approach can be broken down into three steps: the first part will deal with feature engineering which will be developed in 3.1 and will focus on the operation of convolution neural networks, in particular the feature detector and feature map proposed in this algorithm; the last step will highlight the classification phase of the flows for the identification of the adequate classes to which the flows are associated.

3.1 Our Feature Engineering

As shown in Fig. 2, a flow represents a communication session between two assets during a given time interval. Flow characteristics (e.g., IP addresses, ports, communication protocols) can be found in any network activity. We propose feature engineering based on the definition of a network flow. This

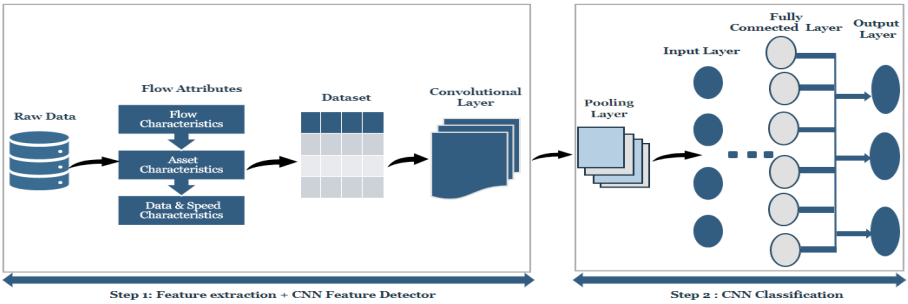

Fig. 1 Our proposed model.

aims at extracting from the raw data, relevant features useful for any network flow classification activity, reducing the processing time of a model to feed a network activity classification algorithm.

Our feature engineering aims at finding a set of universal minimalist features and facilitating several network flow analysis tasks. Thus, we believe it can be used to classify network activity. As shown in table 1, the features of any flow can be ranged in three main categories, namely:

- **Asset characteristics** that gather features related to Network, Transport and Application layers of the OSI model;
- **Flow characteristics** regroup features related to layer 5 (session) of the OSI model, including the type of session established, the direction and duration of the communication;
- **Exchanged data and rate characteristics** in which we find features related to exchanged contents (e.g., amount of communicating data).

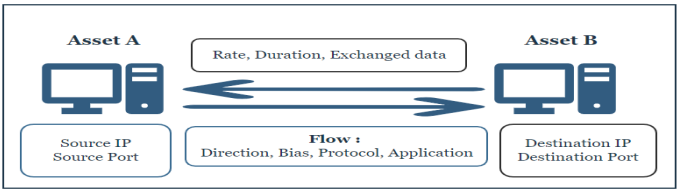

Fig. 2 Flow definition.

3.2 Our Model

Once the features are selected, the data of these features are injected into a classification algorithm. For this purpose, there are different ML and DL approaches and methodologies, as we have seen in the state of the art, most of which aim to improve the accuracy of the model. We have a part of the attack categories with similar feature va* lues, which may decrease the models based on the ML algorithm and target an application of our model to an industrial context with high data volumes. Therefore, we propose a classification based on deep learning where some features are learned during the learning phase.

We choose CNN among other DL approaches because of its unique feature detector on the convolutional layer. The feature detector detects the features that will match the highest values of the input data in the feature map. A convolution operation is performed between the input data (dataset) and the feature detector in order to determine this feature map that is fairly representative of the relevant data.

Our model aims at coupling this feature map formation process with the feature engineering proposed in 3.1 in order to define a robust method for classifying the network flows.

Flow Characteristics	
Flow-ID	Flow Type
Flow-Aggregation-Count	Flow Direction
Flow-Bias	Flow Duration
Asset Characteristics	
Source-IP	Source-Port
Destination-IP	Destination-Port
Protocol	Category
Application	Application-Group
Data and Speed Characteristics	
Source-Bytes	Destination-Bytes
Bit-Per-Second	Total Bytes

Table 1 Features family.

Thus, we propose a One-Dimensional CNN (1D-CNN) approach with parameters of 64 filters to extract 64 different features on the first convolution layer of the network and a kernel-size of 3 (3x3 matrix). This size corresponds to the size of the feature detector that allows to determine the number of feature maps created in our convolution layer. In this layer, we use the ReLu function to add non-linearity in our model. The max-pooling ensures a spatial invariance property and decreases the risk of overlearning by removing the unimportant information and keeping only the most relevant ones to generalize the model. Thus, we have chosen a value of two for the pool-size [18]. And, we add a pooling layer with two pool-size, followed by a flattening phase applied in all previously defined pool features maps. The fully connecter layer is added with 128 units corresponding to the average number of neurons in the input and output layers, while maintaining the ReLu function. For the output layer, we give as value to the parameters eight output neurons that correspond to the classes of attacks present in our model. We rely on the cross-entropy cost function for this network flow classification problem, with an `Adam` optimizer, and accuracy to measure the performance of our model. The activation function `Softmax` evaluates the output probabilities of each class.

Parameter	Value
Convolution Layer / Max-pooling	3 layers / 2 layers
Dropout	0.3
Fully Connected	2 layers
Output Layer / Optimizer	Softmax / Adam
Activation Function	ReLu and Softmax
Epoch / Batch Size	200 / 3

Table 2 Our CNN parameters.

In order to optimize the model, we will make it deeper while taking into account that the complexity of the model increases the computation time. Thus, we can add layers at the convolution layer or the fully connected layer or both layers. We define a loop that adds three convolution layers, and do max-pooling on these new layers while maintaining the same parameters as in the first convolution layer. However, to minimize the over-training, we will use dropout to disable some neurons. Thus, a probability of 0.3 is considered in our model to disable the neurons. Table 2 summarizes the different settings of our proposed convolution neural network model.

4 Evaluation - Results

In this section, we will evaluate the accuracy of our model in network flow classification. To do so, we first apply our feature engineering coupled with our CNN-based model on an IBM dataset extracted from a real industrial context. We compare then our solution to ML algorithms in terms of binary and multi-class classification. Since we aim to propose a general solution for network flow classification, we show, in the second part of this section, the effectiveness of our solution against existing solutions while considering, this time, the well known NSL-KDD dataset that is widely used in the literature.

4.1 IBM Dataset

IBM QRadar is a security appliance that is built on Linux. QRadar allows to collect, store and correlate logs for the detection of security incidents. To prove the effectiveness of our approach, we extracted raw data from a real-time industrial context, with an IBM proprietary IDS probe (QRadar Network Insight, QNI), which extends QRadar by providing a detailed view of real-time network communications [19].

Our dataset includes data collected in 2021, and thus recent attack chains (e.g., exploit log4Shell vulnerability). The flows used in our classification have been extracted during an interval of one week. This extraction contains legitimate flows and non-legitimate ones that we categorize using the framework MITRE ATT&CK [20]. This content is presented in table 3.

Flow-class	Target-ID	Tactics	Techniques	Size (#rows)
Normal	0	-	-	10.000
Large-Leakage	2	TA0010	T1567	10.000
Stealthy-leakage	1	TA0010	T1030	10.000
Web-Exploit	7	TA0002	T1204	5.000
DOS-Attack	3	TA0040	T1498	150
Indicator of Compromise Inbound	4	TA0001	T1189	500
Indicator of Compromise Outbound	5	TA0011	T1102	300
Malicious-Website	6	TA0011	T1102	15

Table 3 IBM Dataset content validated with MITRE ATT&CK.

4.2 Preprocessing

In this phase, we make transformations on our raw data to allow its processing by the learning algorithms that we used. The raw data has an average of 150 features by default. However, using all these features for classification is not an

optimal approach. Therefore, in our solution, we reduce the number of features to 14 (at most) according to our feature engineering approach (Section 3.1) in addition to one target that describes the flow class.

After this extraction, all decimal entries (e.g., rate, amount of data, source/destination port numbers) remain unchanged. On the other hand, each byte in a field related to an IP address is converted to its hexadecimal value (excluding the dots that separate the bytes). After that, these values are concatenated which gives us a unique value that will be converted to decimal. The fields related to "flow direction" has at most four possible values in the raw data. Therefore, these values are mapped into integers in the set $\{1, .., 4\}$. The fields related to "flow bias" has at most five possible values in the raw data. Therefore, these values are mapped into integers in the set $\{1, .., 5\}$. Finally, the type of the communication protocol used is replaced by its corresponding number assigned by the Internet Assigned Numbers Authority (IANA) (i.e., 6 and 17 for TCP and UDP respectively).

4.3 Results

To determine the quality of our classification model and to evaluate its performance against various existing learning algorithms, we used the following four popular evaluation metrics:

$$Accuracy = \frac{(TP+TN)}{(FP+FN+TP+TN)}, \; Precision = \frac{TP}{TP+FP}, \; Recall = \frac{TP}{TP+FN},$$
$$ScoreF1 = 2 * \frac{(Precision*Recall)}{(Precision+Recall)}$$

Where, for a given flow class C: True Positive (TP) represents the number of flows correctly classified in the given class C; True Negative (TN) represents the number of flows correctly classified outside the class C; False Positive (FP) represents the number of flows wrongly classified in the class C; and finally False Negative (FN) represents the number of flows wrongly classified outside the class C.

4.3.1 The evaluation of binary and multi-class classification using two-feature families

As a first step to validate our approach, we started with a binary classification which should allow us to distinguish legitimate flows from illegitimate ones. To do so, we first assign all legitimate flows (Normal flows in table 3) to the target 0 while all illegitimate flows will be assigned to the target 1. After that, we select ten features in each entry in our dataset. These features correspond to two features families among the three highlighted in section 3.1, namely, Flow features (the number of records in a flow, the type of flow, the direction of the asset that initiated the communication, the duration of the communication, the data transfer bias) and Asset features (the source IP address, the destination IP address, the source and destination ports, and the protocol used). This data has then been injected into our model and the results have been compared in table 4 with those of four ML algorithms (K-Nearest Neighbors (KNN), Naive Bayesian (NB), Random Forest (RF), Decision Tree (DT)).

T	KNN			NB			DT			RF			Our model		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
0	1	1	1	1	0.99	0.99	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Acc	100%			99%			100%			100%			100%		

Table 4 A comparison table in terms of binary classification using 2 families of features.

As we can see in the results of table 4, our solution can indeed distinguish legitimate network traffic from an abnormal one, where we notice of a 100% accuracy rate for our model. The other ML algorithms also achieve good results except for Naive Bayesian that has 99% accuracy and which is more likely due to its probabilistic nature. However, in a real industrial context, it is also important to distinguish the nature of the attack flows in order to take proper counter-measurements. Therefore, we present in table 5 a comparison of the results of a multi-class classification of illegitimate flows between our model and the above ML algorithms.

T	KNN			NB			DT			RF			Our model		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.74	0.76	0.75	0.94	0.15	0.26	0.70	0.69	0.69	0.71	0.75	0.73	1	0.96	0.97
2	0.75	0.74	0.75	0.55	0.79	0.71	0.68	0.69	0.68	0.72	0.68	0.70	1	0.92	0.94
3	1	1	1	1	1	1	1	1	1	1	1	1	1	0.95	0.92
4	0.83	0.31	0.45	0.40	0.67	0.50	0.93	1	0.96	0.93	1	0.96	0.91	0.91	0.94
5	1	1	1	1	0.45	0.62	1	0.86	0.93	1	0.91	0.94	1	0.96	0.96
6	1	1	1	1	1	1	1	1	1	1	1	1	0.85	0.89	0.92
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Acc	88%			79%			85%			86%			92%		

Table 5 Multiclassification with 2 families of features.

As shown in table 5, we notice that our solution offers a better accuracy and a very good precision, recall and F1 scores in all attack targets. As a reminder, the F1 score is an arithmetic average between precision and recall, so it will allow us to measure the ability of the learning algorithms to better associate flows with legitimate traffic classes or corresponding target attacks.

The first general observation is a confusion between targets 1 and 2 in almost all ML algorithms and a better classification for targets 3, 6 and 7 compared to the remaining targets. We note that targets 1 and 2 correspond to data leak flows including data leaks in short time intervals, and data leaks in much longer ones. For these flow types, the same IP address can be spotted in both scenarios, which may cause the confusion between the data in these two classes in ML algorithms. In KNN, the F1 score is 0.75 for target 1 and 2, and quite low, especially for target 4. In the NB, we notice the lowest accuracy among all ML algorithms that we tested, which strongly impacts the F1 score which is 0.26 for target 1. A significant confusion was also found in the NB between targets 4 and 5 (corresponding to the inbound and outbound compromise indicator data) with F1 scores of 0.5 and 0.62 respectively. This is explained by the fact that the flow duration in these two attack classes is relatively close, due to the probabilistic operation of NB. The DT and RF improve these F1 scores at target 4 and 5 since when an attribute has the same approximate value for two given classes in an internal node, these two algorithms dissociate the given targets in the following nodes by comparing other attributes.

Our solution, which combines our feature extraction approach with the feature detector proposed in CNN, solves the confusion observed in ML algorithms. Indeed, a neural layer overlay, including the layer filtering principle, is well adapted to our context of multi-class classification of confused output layers. Convolutional neural networks learn the values of the weights in the same way that they learn the filters of the convolution layer, which can allow it to distinguish between confused classes 1 and 2, and other classes (like target 4 and 5) with medium or low F1 scores which explains the improvement of accuracy highlighted in Table 5.

4.3.2 The evaluation of multi-class classification using three-feature families

In order to investigate the impact of the third family of features (highlighted in section 3.1) on the accuracy of our model, we select the minimalist features mentioned above, the characteristics related to "Data and Speed" in each entry considered in our dataset (e.g., the number of bytes sent by the source, the number of bytes received, etc.). Therefore, the number of characteristics increases from 10 to 14 features that we use in both our model and ML algorithms to classify malicious network flows. The results of this evaluation are presented in Table 6.

As we can observe in table 6, an improvement of the F1 score of all targets is noticed compared to the table5 in both our model and ML algorithms, with an exception for NB. In this model, the F1 scores are more or less the same as the ones in table 5 for all targets, except for target 4 (incoming traffic related to indicators of compromise) where a sharp decline is observed. This decreases the accuracy of this model from 79% to 73%.

Overall, we can clearly deduce that the additional information provided by the third family of features gives much more context and allows us to dissociate certain types of attacks. Our solution appears as the best approach with an average F1-score of 0.96 and an overall accuracy of 94.84% which is better than the ones presented in table 5.

T	KNN			NB			DT			RF			**Our model**		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	0.80	0.78	0.79	0.93	0.17	0.28	0.86	0.85	0.85	0.86	0.91	0.88	1	0.95	0.96
2	0.78	0.81	0.80	0.48	0.74	0.58	0.84	0.85	0.85	0.90	0.94	0.87	0.98	0.98	0.95
3	1	1	1	1	1	1	1	1	1	1	1	1	1	0.91	0.97
4	1	0.60	0.75	0.02	0.82	0.03	1	0.95	0.97	96	0.95	0.96	0.95	0.90	0.89
5	1	1	1	1	0.44	0.62	1	0.73	0.84	0.95	0.82	0.88	1	1	1
6	1	0.67	0.80	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Acc	90%			73%			91%			92%			94.84%		

Table 6 Multiclassification with of all families of features.

4.3.3 The evaluation of Epochs according to accuracy and loss values

The convergence of a model is determined by the analysis of the error rate and accuracy curves. Thus, we will focus on determining the optimal number of epochs in order to find a trade-off between the execution time of our CNN

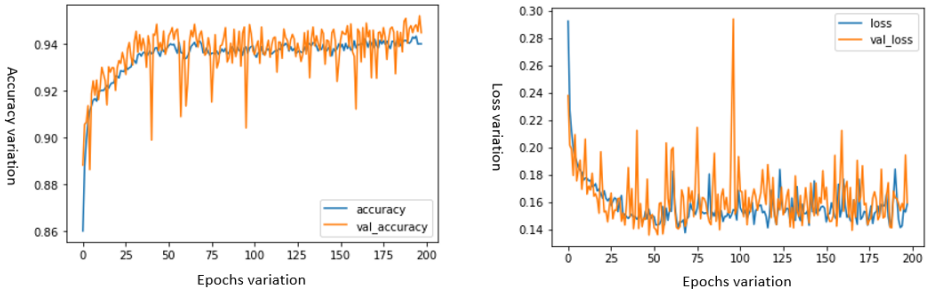

Fig. 3 Number of epoch with accuracy and loss

model and its overall accuracy because one of the drawbacks of deep learning based solutions is the execution time. In table 2, we have initially defined a baseline value of two-hundred epochs which we believe is more than enough for our model to converge. However, we went through the `earlyStopping` technique to explore the possibility of stopping the learning, as soon as the model starts to overlearn and thus to reduce the execution time. To do so, we present in Fig. 3, the accuracy and loss in the training data (blue curve) and in the test data (orange curve) according to the number of epochs used in our model.

As we can see in Fig. 3, from the first to the fiftieth epoch, the accuracy increases with the number of epochs and the loss decreases accordingly, which indicates that the model is still continuing to learn and converges. Between the fiftieth and seventieth epochs, both metrics start to stagnate. This stagnation continues until the 200^{th} epoch, with some fluctuations observed in some epochs due to confusions between some neurons.

4.4 A comparative analysis using the benchmmarking dataset NSL-KDD

There have been many intrusion detection approaches proposed in the literature. However, these approaches are evaluated in different datasets. Therefore, we tested our model on the NSL-KDD dataset (that has been used in several works of the literature) in order to check its efficiency and establish a fair comparison with existing solutions that adopted the same dataset.

The NSL-KDD dataset [21] is an enhanced version of the KDDCup99 dataset [22] which is a standard dataset consisting of a wide variety of simulated intrusions in a military network environment, defined in 1998 by "MIT Lincoln Labs" of the U.S. DARPA agency. NSL KDD has a KDDTrain+ set (125,973 records) divided into 22 types of attacks and normal traffic and a KDDTest+ test set (22,544 records) which contains 14 additional attacks not included in the KDDTrain+. The set of attack classes is divided into four main families: Denial of Service (DOS), Remote To Local (R2L), User To Root (U2R), Probe. The content of NSL-KDD is presented in the Table 7.

NSL-KDD dataset provides 41 features grouped into 3 families: basic features, traffic features to the same host and traffic features to the same

Flow-class	Tactics	Techniques	Train(#rows)	Test(#rows)
Normal	-	-	67 343	9 711
DOS	TA0040	T1498	45 927	7 458
R2L	TA0001	T1133/T1199	995	2 754
U2R	TA0004	T1548	52	200
Probe	TA0043	T1589/T1590	11 656	2 421

Table 7 The NSL-KDD dataset content.

service. However, the application of our feature engineering (proposed in section 3.1) will allow us to reduce the number of features used for training to the following 12 features (excluding labels): `duration`, `protocol-type`, `service`, `land`, `src-bytes`, `dst-bytes`, `count`, `srv-count`, `same-srv-rate`, `srv-diff-host-rate`, `wrong-fragment` and `flag`.
Moreover, to reduce the impact of inconsistencies between the training and test data in terms of targets, we chose to concatenate the two by keeping only 22 targets common to both sets, then splitting this concatenated set at 80% for the training sets and 20% for the test data. Then, we proceeded to a preprocessing and normalization of the extracted data, in which all numerical values are kept unchanged while attributes with categorical values like services and protocol-type have been mapped to decimal values, as it has been done with IBM dataset. Finally, targets have been labeled as follows: normal:0, DOS:1, Probe:2, R2L:3, U2R:4; and the resulting data is injected into our model. The results of the tests of our model on the NSL-KDD are presented in the Table 8.

Target	**Our model (NSL-KDD)**		
	Precision	Recall	F1-score
Normal	0.99	0.99	0.99
DOS	0.99	0.99	0.99
Probe	0.97	0.97	0.97
R2L	0.94	0.79	0.86
U2R	0.50	0.13	0.21
Accuracy	98, 7%		

Table 8 The evaluation results of our Model on NSL-KDD dataset

Overall, as we can see in Table 8, our model achieves a global accuracy of 98%. We also notice a high precision and a good F1-Score for the targets "Normal", "Probe" as well as "DOS" and to a lesser extent "R2L". However, the target "U2R" has a low score compared to the remaining targets. Actually, the "U2R" target relates to an attempt to elevate privileges which does not correspond to the definition of network stream. Thus, its classification cannot rely solely on the characteristics of a network flow, which explains in addition to the low number of inputs of that target the gap with the other targets that are in fact network flows and have many more entries in the dataset.

Moreover, based on the NSL-KDD dataset, we compare in Table 9 our model with other solutions proposed in the state of the art according to the confusion matrix (precision, recall, F1-score and accuracy). Note that the chosen solutions use the following algorithms on NSL-KDD dataset: ANN ([12]), Deep learning with ANN (Auto-encoder) ([23]) and RNN+LSTM, CNN ([24]).

As we can see in Table 9, our solution significantly enhances the results of the existing solutions especially in terms of classification accuracy. In addition, it reduces the preprocessing time applied to the raw data compared to

Model	Accuracy	Precision	Recall	F1-score
Our model	0.98	0.88	0.78	0.80
R. Vinayakumar et al. (2019) [12]	0.785	0.810	0.785	0.765
C. Zhang et al. (2019) [23]	0,7974	0,8222	0,97974	0,7647
Z. Tauscher et al. (2021) [25]	0,8047	N/A	N/A	0,7839
L. Liu et al. (2020) [24]	0,7824	0,7838	0,7823	0,7503

Table 9 Comparison with the state of art based on NSL-KDD dataset

some existing solutions. For instance, in [23], the authors run a deep learning algorithm to extract the most relevant features for classification, which is a time consuming task compared to our preprocessing phase that yet allows to achieve better classification results. Moreover, our solution also reduces the model execution time compared to the remaining solutions, since it uses less features for the training (12 in ours, 122 in [23], 41 in [12], [24] and [25]).

5 Conclusion

In this paper, we propose an intrusion detection solution for network flow collectors using 1D Convolutional Neural Networks. First, our solution presents a feature engineering for the extraction of data features according to the definition of a network flow. The proposed engineering can be used in any network flow classification, since it is based on criteria that can be identified in any network event. Our feature extraction is then associated with a feature detector functionality defined in CNN, which guarantees an accurate determination of normal network flows and a robust multi-class classification of malicious ones. Our model has been evaluated on two datasets, the first one has been extracted from a real IBM industrial context while the second one is the public dataset NSL-KDD. Both sets were validated using the MITRE ATT&CK framework, and the results have proved that our model distinguishes normal behaviors from deviant ones, and efficiently identifies the attack classes. Moreover, our model significantly improves the accuracy of the classification process when it is compared to other existing solutions of the state of the art. It also reduces the number of extracted features and thus the execution time of the global model compared to existing solutions. In the future, we intend to focus on zero-day vulnerability exploitation attacks. We also plan to work on a data generator composed of recent attack families, classified by the MITRE ATT&CK framework, given the obsolescence of some data sets.

References

[1] Lin, P., et al. A Novel Multimodal Deep Learning Framework for Encrypted Traffic Classification. IEEE/ACM Transactions on Networking, 1–16 (2022).

[2] Zhu, X., et al. Machine-learning-assisted Traffic Classification of User Activities at Programmable Data Plane. In: 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS) (2022).

[3] Xin, S.: Research of Intrusion Detection System. In: International Conference on Computational and Information Sciences, pp. 1460–1462 (2013).

[4] Yin, C., et al., A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access **5**, 21954–21961 (2017).

[5] Kılıç, H., et al., Evasion Techniques Efficiency Over The IPS/IDS Technology. In: 4th International Conference on Computer Science and Engineering (UBMK), pp. 542–547 (2019).

[6] Salman, O., et al., A review on machine learning-based approaches for Internet traffic classification. Annals of Telecommunications **75**(11), 673–710 (2020)

[7] Jabbar, M.A., et al., intelligent network intrusion detection using alternating decision trees. In: International Conference on Circuits, Controls, Communications and Computing (I4C)

[8] Sharmila, B.S., et al., Intrusion Detection System using Naive Bayes algorithm. In: IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE) (2019).

[9] Meena, G., et al., A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. In: International Conference on Computer, Communications and Electronics (Comptelix), pp. 553–558 (2017).

[10] Koc, L., et al., Network Intrusion Detection Using a HNB Binary Classifier. In: 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim), pp. 81–85 (2015).

[11] Varanasi, V., et al., Network Intrusion Detection using Machine Learning, Deep Learning - A Review. In: 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 1618–1624 (2022).

[12] Vinayakumar, R., et al., Deep Learning Approach for Intelligent Intrusion Detection System. IEEE Access **7**, 41525–41550 (2019).

[13] Sivamohan, S., et al., An Effective Recurrent Neural Network (RNN) based Intrusion Detection via Bi-directional Long Short-Term Memory. In: International Conference on Intelligent Technologies (CONIT) (2021).

[14] Wang, W., et al., HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection. IEEE Access **6**, 1792–1806 (2018).

[15] Azizjon, M., et al., 1D CNN based network intrusion detection with normalization on imbalanced data. In: International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp. 218–224 (2020).

[16] Atefi, K., et al., A Hybrid Anomaly Classification with Deep Learning (DL) and Binary Algorithms (BA) as Optimizer in the Intrusion Detection System (IDS). In: 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), pp. 29–34 (2020).

[17] Rajesh, P., et al., Analysis Of Cyber Threat Detection And Emulation Using MITRE Attack Framework. In: International Conference on Intelligent Data Science Technologies and Applications (IDSTA), pp. 4–12 (2022).

[18] Zheng, W.-F.: Intrusion Detection Based on Convolutional Neural Network. In: International Conference on Computer Engineering and Application (ICCEA), pp. 273–277 (2020).

[19] Sekharan, S.S., et al., Profiling SIEM tools and correlation engines for security analytics. In: International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 717–721 (2017).

[20] MITRE ATTA&CK. https://attack.mitre.org/

[21] Tavallaee, M., et al., A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009).

[22] Shah, B., et al., Reducing features of kdd cup 1999 dataset for anomaly detection using back propagation neural network. In: 2015 Fifth International Conference on Advanced Computing Communication Technologies, pp. 247–251 (2015).

[23] Zhang, C., et al., A deep learning approach for network intrusion detection based on nsl-kdd dataset. In: 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), pp. 41–45 (2019).

[24] Liu, L., et al., Intrusion detection of imbalanced network traffic based on machine learning and deep learning. IEEE Access **9**, 7550–7563 (2021).

[25] Tauscher, Z., et al., Learning to detect: A data-driven approach for network intrusion detection. In: 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), pp. 1–6 (2021).