Use of Machine Learning Algorithms for Network Traffic Classification

Adrián Nieto Antelo, Diego Fernández Iglesias, and Francisco J. Nóvoa

Facultad de Informática, Universidade da Coruña, 15071, A Coruña, España
Centro de Investigación CITIC, Departamento de Ciencias de la Computación y Tecnologías de la Información, Universidade da Coruña, 15071, A Coruña, España
Correspondence: adrian.nieto1@udc.es

DOI: *https://doi.org/10.17979/spudc.000024.48*

Abstract: In recent years, the complexity of threats utilizing the network as an attack vector has significantly increased. Traditional attack prevention and detection systems (IPS/IDS) based on signatures do not provide an acceptable level of security for many organizations.
Furthermore, the volume of traffic on corporate networks has also grown exponentially, while quality of service requirements do not always allow for deep inspection (at the application layer) of packets.
The main objective of this work is to demonstrate that the application of machine learning techniques to the information of data flows circulating through the network allows for the satisfactory detection of malicious traffic. Specifically, this work is developed within an emerging network paradigm, such as software-defined networks.

1 Introduction

It is of vital importance to offer the highest possible protection to defend ourselves against all the current computer threats. Currently, our protection mainly relies on methods based on policies and rules. These methods have certain issues, such as human errors in rule configuration, response time to threats, and adaptability. That's why Artificial Intelligence, especially Machine Learning, is gaining more weight in cybersecurity.

This work aims to compare and understand different machine learning techniques used in the classification of network traffic on an SDN (Software-Defined Network). To achieve this, a methodology known as CRISP-DM (Cross-Industry Standard Process for Data Mining) has been employed Wirth and Hipp (2000). This article will be structured according to this methodology, with each section representing a different step in it. However, for this article, the business understanding section has been omitted, as it involves the theoretical explanation of the concepts that will be used throughout the work.

2 Understanding the data

This phase involves the collection, description, exploration, and verification of the data to be used.

The dataset used is one that already existed previously and was not generated specifically for this work, as creating a dataset is not one of the objectives of this project. In this case, the InSDN dataset Elsayed et al. (2020) is used.

This dataset presents various attack scenarios from different sources, both external and internal, that can affect an SDN network. It includes various attack scenarios, such as DoS attacks,

DDoS attacks, password guessing attacks, web application attacks, probe attacks, botnet attacks, and U2R attacks. The dataset is divided into three groups based on the type of traffic and target machines. The first group includes normal traffic. The second group includes attacks targeted at the Metasploitable2 server, which is a virtual machine server. The third group consists of attacks on the OVS machine, which is a switch.

After loading the dataset, exploration is conducted using a Pandas dataframe specifically created to work with this data. Among these explorations, the correlation matrix stands out, which will be used in the next step.

3 Data preparation

In this phase, the selection of features and samples to be used in the Machine Learning algorithm will be carried out. The quality of the data will be checked, formatted, modified, or new data will be created as necessary.

The data from the previous dataset are not in an appropriate format for training, so they need to be prepared for use in training.

The first step is to remove outliers that may be present in one of the dataset's features. The Isolation Forest algorithm, which is based on decision trees, was used for this task. After applying the algorithm, it was decided to remove values from four features in the dataset, namely Tot Fwd Pkts, TotLen Fwd Pkts, TotLen Bwd Pkts, and Bwd Header Len.

Next, modifications to existing features were made to provide more information. This includes the creation of the target class, which divides the data flows into normal and attack traffic. The Timestamp variable was also modified to obtain only the hour, and the source and destination ports of the flows were grouped into categories defined by IANA IANA (2023). In the case of the hour, as it contains a cyclical character, it is necessary to transform this variable in a way that algorithms can understand this cyclical nature. Therefore, a circular representation of the hour is used, where the value 0 appears immediately after 23. To achieve this, sine and cosine functions are used.

Afterward, the data was formatted because two of the algorithms used (SVM and Logistic Regression) require their features to be numeric values, both integers and floating-point values. Specifically, five object-type attributes were formatted. The source and destination IP addresses were encoded using Label Encoding, similar to what was done with the ports.

Following this, standardization was performed using the scikit-learn StandardScaler library. This process is necessary because algorithms like SVM are sensitive to the scale and magnitude of variance in the data, which can influence the model. This is because they cannot interpret the meaning of the data; they only see numbers. If the predictors are not scaled equally, features with a larger scale or more variance can have a greater influence on the final prediction.

The last step is data selection, as the database is extensive and optimization of performance and accuracy is needed. First, features that do not provide information were removed, whether they take only one value or have already been used to create other more relevant features. For this, the values from the correlation matrix, calculated in the previous step, were used to eliminate features that are correlated both directly and indirectly with others, removing the less important one after calculating its importance using Random Forest. Finally, from the resulting features, only a subset was chosen for model creation, which was done using the RFECV algorithm. To use this algorithm, a decision tree classifier was chosen as the estimator, Stratified K-fold as the cross-validation strategy, accuracy as the scoring metric, and a step size of one.

In the end, a dataset with 31 features was obtained.

4 Modeling

This phase involves selecting the algorithms to use, generating the design test, building the models, and evaluating them.

First, the methods to assess the quality and validity of the model will be established. In this case, two methods will be used: cross-validation and certain evaluation metrics, such as those obtained from the confusion matrix.

For designing the models, an approach based on Stratified K-fold Cross Validation was used. This was implemented through two nested cross-validation loops. The value of k was set to 5 for both the outer and inner loops. These values are standard and were chosen because they yield good results within a reasonable time frame.

The outer loop performs the initial partition, dedicating 20% of the dataset to the test set in each iteration. The inner loop takes the remaining 80% of the original dataset and divides it into five parts. One of these partitions is used for validation, while the rest are used for training. This means that 16% of the dataset is used for validation, and 64% for training.

For the inner loop, a scikit-learn class called GridSearchCV was used, allowing for hyperparameter tuning. GridSearchCV enables the comparison of different parameters to find the optimal ones. However, when using an outer loop, it provides 5 parameter combinations, which can be the same or different. At the end of all outer loop executions, the set of parameters that yielded the best results is obtained, but this value corresponds to a single execution of the outer loop. In other words, one combination may perform best in one run, while performing poorly in the other four. For this reason, a measurement has been devised to calculate the best parameters using information from all 5 iterations of the outer loop. This measurement takes into account both the mean and the standard deviation. A lower standard deviation indicates less variability, so the results tend to be more consistent. To use these two values and obtain the best possible metric, the decision was made to divide the mean by the standard deviation. This causes high means and low standard deviations to yield higher results. A higher final result indicates that the parameters generally yield better and more consistent results.

The following three algorithms were used: SVM, Random Forest, and Logistic Regression.

4.1 SVM

For this algorithm, the parameters C and Tol have been configured. C is the parameter that adjusts the hyperplane separating the two classes, and Tol specifies the tolerance for stopping criteria. In the end, the values chosen are 1 for the C parameter and 0.0001 for Tol.

4.2 Random Forest

For this algorithm, the parameters "n_estimators," which indicates the number of decision trees to be created in the forest, and "max_depth," which is the maximum depth a tree can have, have been configured. In this case, the chosen values are "max_depth" equal to 9 and "n_estimators" equal to 50.

4.3 Logistic Regression

For this algorithm, the parameters "Penalty," which specifies the penalty norm, "C," which is the inverse of the regularization strength, and "Tol," which specifies the tolerance for stopping criteria, have been configured. The chosen values are C=1, penalty=l2, and tol=0.0001.

5 Evaluation

In this phase, the results are evaluated, the process is reviewed, and the next steps to be taken are determined. To evaluate the results, cross-validation using a test set will be employed, and metrics derived from the obtained confusion matrices will be utilized. The results can be seen in Table 1.

As observed, Random Forest is the algorithm that yields the best results, and it will be the one we deploy.

Table 1: Métricas obtenidas a partir de las matrices de confusión.

	Precisión	Exactitud	Sensibilidad	F1-score
RL	0.893330	0.657411	0.968729	0.783270
RF	0.999952	0.999951	0.999805	0.999878
SVM	0.908813	0.707398	0.924062	0.801343

6 Deployment

Finally, in this phase, the model is implemented, maintenance and monitoring are conducted, a final report is produced, and the entire project is reviewed.

In this phase, the obtained model will be tested in an environment where a controller changes the behavior of a switch to block a flow if it has been classified as an attack by the model. Only the algorithm with the best results, which is Random Forest, will be used for this purpose.

For the deployment, two machines have been used, one running the network topology, while the other runs the network traffic controller.

The first step in this section is to create the environment in which the models will be tested. This will be an SDN that uses the OpenFlow protocol for communication between the controller and the switch. The tool used to create it is known as Mininet. Afterward, the controller to be used is chosen, which will be Ryu, and the application to be used within it. In this case, we have chosen the "simple_switch_13" template, which uses OpenFlow 1.3. This template allows defining the switch's behavior so that it can rewrite packet addresses, transfer packets from a specified port, transfer received packets to the controller, and transfer packets forwarded by the controller from a specified port Ryubook (2014).

When implementing the model, the first thing to consider is how to obtain the information corresponding to the attributes used by the model for classification. This is where our first problem arises. Obtaining this information in our test environment is not straightforward and would involve excessive complexity. Therefore, an alternative solution has been proposed. This solution involves retrieving this information from the dataset itself. This is possible because we are going to simulate the data flows that make up that database, as the .pcap files that allow simulation are available. The available traffic in a .pcap file will be injected into the Mininet network. After this, the FlowID of the flow will be obtained, which is the characteristic that identifies the flow, and it will be searched for in the dataset by this ID. Once found, the information corresponding to the attributes required for the model will be saved.

After implementing the template, it is monitored and its operation is displayed. To achieve this, the "simple_monitor_13" template has been modified to display information about the flow table on the screen.

However, this monitor has the problem that adjusting the time between messages is difficult, which can make the controller's screen unreadable at times. Additionally, it was not possible to display the output of the model because this information is not stored in the flow table. For this reason, the table will be printed after the processing of a flow, which leaves a cleaner interface and simplifies the code.

To observe the controller's behavior on a flow, two .pcap files belonging to the InSDN dataset will be injected, one corresponding to an attack flow and another containing normal traffic. To inject both files, tcpreplay has been used.

7 Conclusions

The main conclusions obtained is that it has been possible to deploy an application in the controller that is capable of classifying traffic and modifying the operation of an SDN switch depending on the response obtained, using machine learning. This machine learning-based security implementation would be possible to implement by varying certain steps performed in

this work, such as performing the deployment phase in a real environment instead of a test environment. In this environment, it would be possible to capture the traffic flowing over the network over a period of time, and then build the model using that more specific network data, rather than searching the dataset.

Acknowledgements

This research has been funded by the Spanish Ministry of Economy and Competitiveness and European Union ERDF funds (Project PID2019-111388GB-I00). CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Consellería de Cultura, Educación, Formación Profesional e Universidades and the Galician universities for the reinforcement of the research centres of the Galician University System (CIGUS).

Bibliography

M. S. Elsayed, N.-A. Le-Khac, and A. D.Jurcut. Insdn: A novel sdn intrusion dataset. *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9187858*, 2020. [Online; accessed 14-September-2023].

IANA. Service name and transport protocol port number registry. *https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml*, 2023. [Online; accessed 14-September-2023].

Ryubook. Switching hub. *https://osrg.github.io/ryu-book/en/html/switching_hub.html*, 2014. [Online; accessed 14-September-2023].

R. Wirth and J. Hipp. Crisp-dm: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 2000.