An explainable-by-design ensemble learning system to detect unknown network attacks

Céline Minh*†, Kevin Vermeulen†, Cédric Lefebvre*, Philippe Owezarski†, William Ritchie*
*Custocy, Toulouse, France. Email: {cminh, clefebvre, writchie}@custocy.com
†LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France. Email: {celine.minh, kevin.vermeulen, owe}@laas.fr

***Abstract*—Machine learning is a promising technology for network intrusion detection systems. There is a wide variety of machine learning algorithms whose results seem complementary, but determining which result is true is difficult because models lack explainability. Our system intends to reconstruct attack patterns from a set of unsupervised learning models' outputs, and show them to security analysts. Therefore, we introduce an explainable-by-design system to detect network attacks, and evaluated its accuracy on the CSE-CIC-IDS2018 dataset [1].**

I. Introduction

Machine learning (ML) is a promising technology for network intrusion detection systems (NIDSs) as it allows accurate attack detection on large amounts of traffic data. An issue with ML models is that some models will qualify an input as an attack, whereas others will qualify the same one as benign. Choosing which model to trust is difficult because ML models are often considered as black boxes and do not provide evidence to justify their results. The ensemble learning approach [2], [3] takes advantage of the complementarity of multiple models, called base-learners, by combining them (e.g., weighted majority voting). Another ensemble method is meta-learning, where a model, called meta-learner, learns from base-learners' outputs. We propose a new ensemble-based approach that systematically presents the results of each base-learner to security analysts in a way that facilitates informed decision-making. This decision would be made thanks to a combination of a concise visual representation of detections and a high level of explainability of each base-learner.

Explainability is the property of a system that makes its reasoning and results understandable by humans [4], [5]. In current NIDSs, security analysts take decisions to solve security issues based on the system analysis. Therefore, providing intelligible evidence of ML models results is crucial for analysts to trust the system. Explainability is not only important for improving the collaboration between security analysts and AI systems, but also for assisting engineers and researchers in the design of more accurate systems, by helping them understand why and where a model may fail.

We propose a method that simultaneously allows users to take advantage of ensemble learning from multiple base-learners to enhance detections and visualize the results of all the base-learners to gain insights into how these detections are made. We introduce a visual representation of unsupervised learning (UL) detections over time that intends to both help security analysts understand what is happening on the network and allow our meta-learner, which is a convolutional neural networks (CNN), to identify attack patterns [6], [7]. Our system is expected to preserve UL properties, including the detection of unknown attacks, because the layer that is supervised, the meta-learner, analyzes the outputs of base-learners, which is meta-data and not raw network traffic data.

We then introduce an explainable-by-design system that analyzes and combines a set of UL models to detect network attacks. Our contributions are:

1) A more transparent ML-based NIDS, whose results are explained,
2) A visual representation of network anomalies that users can interpret,
3) An ensemble learning method that uses a CNN as a meta-learner to combine base-learners.

II. System design

A. System overview

We designed an ML-based NIDS that combines a set of UL models to detect network attacks. To combine models with different algorithms, we intended to characterize their results without assuming knowledge about the algorithms, and so by processing only their inputs and outputs.

Regarding the inputs, we needed a data representation that was concise and semantically interesting for UL models to extract attack signatures (Section II-B). Similarly to UNADA [8], [9], we aggregated network flows by source and by destination address, and computed aggregate features (Table I).

To quantify the degree of abnormality of aggregates (Section II-C), we computed anomaly scores from anomaly detections on different subspaces of features [8], [9]. Our anomaly scores are model-agnostic, they do not depend on the decision function of the model.

To go further in the characterization of models' outputs, we introduced a representation (Section II-D) that highlights both a spatial aspect (e.g., do anomalies affect or come from the same IP address?) and a temporal aspect (e.g., how frequent are anomalies?). Then, our meta-learner, a CNN, analyzed our representation of network anomalies to identify attack patterns.

B. Aggregation of network flows

Our system computes the features described in Table I from aggregated network flows. We chose to analyze both

Feature	Aggregation key	Description
n_dst_ip	IPsrc	Number of destination IP addresses
n_src_ip	IPdst	Number of source IP addresses
n_dst_ports	IPsrc & IPdst	Number of destination ports
n_src_ports	IPsrc & IPdst	Number of source ports
n_fwd_pkts	IPsrc & IPdst	Number of forward packets
n_bwd_pkts	IPsrc & IPdst	Number of backward packets
sum_flx_dur	IPsrc & IPdst	Sum of flows duration
tot_flx	IPsrc & IPdst	Number of flows
sum_pkts_size	IPsrc & IPdst	Sum of packets size
std_pkt_size	IPsrc & IPdst	Standard deviation of packets size

TABLE I: Aggregates features

aggregates by source IP address and aggregates by destination IP address because some attacks (e.g., DDoS) involve IP addresses from many sources towards a single destination, and other anomalies (e.g., network scans) involve IP addresses from a single source towards multiple destinations. The aggregates perspective was designed to better identify attack patterns, and also, be more readable for security analysts, as aggregates have a reduced number of features and security analysts are familiar with IP addresses.

In addition, network traffic can evolve quickly and look different at different times (e.g., working hours, weekends). So we chose to perform clustering on aggregates during a same time interval Δ_t of 2 minutes, instead of comparing aggregates with aggregates from previous states of the traffic.

C. Unsupervised anomaly scoring

This component is composed of a set of anomaly detectors. It takes as input the features of aggregates (Table I), either by source or destination IP address, from a same time frame of duration Δ_T of 30 minutes and returns a set of matrices. Each matrix contains the anomaly scores of all the aggregates computed by an unsupervised anomaly detector.

To be able to detect unknown attacks, we used UL models: they detect anomalies, which are aggregates that statistically differ from others. We tested multiple models from the scikit-learn library [10] and PyOD [11] on aggregates from the CIC-CSE-IDS2018 dataset [1], and we observed that binary results were different from one UL model to another [12]. We selected a set of models composed of Local Outlier Factor, KNN, and COPOD because together (i.e., the boolean sum of their results) they detected the most attacks while raising the least false alarms (benign traffic mistaken for attack). This makes sense because these algorithms have different mechanisms to detect anomalies, and so are more likely to be complementary:

- *Local Outlier Factor* [13] is a density-based algorithm.
- *Unsupervised KNN* [14] is a distance-based algorithm.
- *COPOD* [15] is a probabilistic algorithm.

To characterize and compare results of different UL algorithms, we used a model-agnostic quantification of anomalies, previously defined in UNADA [8], [9]. We considered all the subspaces of $k = 2$ features among all the $n = 9$ aggregates' features and performed outlier detection, with the same algorithm, on each subspace. The anomaly score of an aggregate is the number of times the aggregate was classified as an outlier. Those scores range from 0, for a completely normal aggregate (according to the model), to 36, for a completely abnormal aggregate which is the number of combinations of $k = 2$ features among $n = 9$:
$\binom{n}{k} = \frac{n!}{k!(n-k)!}$

D. Attack patterns recognition

In the previous section, we selected a set of UL models to detect abnormal aggregates. To reinforce the system, we studied the sequences of anomalies detected by each base-learner during a time frame Δ_T of 30 minutes. We represented the outputs of each base-learner as a matrix of anomaly scores, where each line corresponds to a source or destination IP address, depending on the aggregation key, and each column to a time interval.

We can then analyze these representations using deep neural networks (DNNs) such as CNNs. Therefore, we used a CNN as a meta-learner to identify attack patterns on traffic representations. The CNN takes as input the two sets of matrices (representations of anomalies) generated from the outputs of the base-learners on aggregates by source and destination IP address during a time frame Δ_T. The CNN gives the final result of the system, meaning if the network was under attack during the time frame Δ_T.

The CNN requires a labeled dataset for training and validation. We chose to label a representation as *attack* (which means the network is under attack during the time frame) if it contained at least one attack aggregate, and *benign* otherwise.

To visualize the analysis of the base-learners, we assigned them a color channel (red for Local Outlier Factor, blue for KNN, and green for COPOD). The color intensity of a pixel is proportional to the aggregate anomaly score generated by the model. Thus, a white pixel means that all models diagnosed the aggregate as very abnormal, and a black pixel means that no model raised an alarm (Figure 1).

III. Results

We evaluated our system on the CIC-CSE-IDS2018 dataset [1]. Sharafaldin et al. [1] implemented a realistic company network, with 420 machines and 30 servers, and an attacking infrastructure with 50 machines. The dataset consists of 10 days of network traffic captures during working hours. The authors simulated multiple attacks scenarios, such as brute-force, distributed denial of service (DDoS), SQL injections, etc. For this study, we evaluated the detection of brute-force and DoS only.

Our system first aggregated the network flows by source and destination IP address (Section II-B). Then, it generated intermediate visual representations of traffic anomalies (Section II-C). Finally, we obtained a dataset of 1860 pairs of images for training and evaluating the attack pattern recognition module (Section II-D).

Table II shows the F-score and confusion matrix of our attack pattern recognition module, i.e., our CNN, which analyzes representations of aggregates by source IP address and

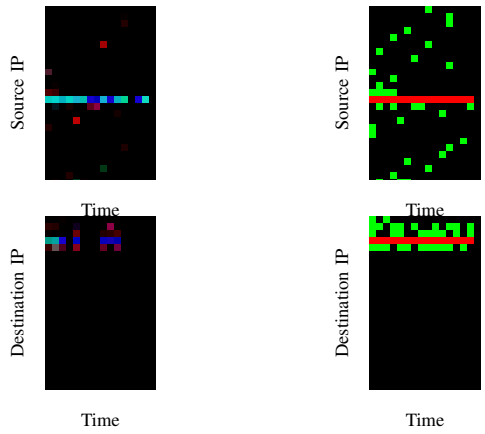

Fig. 1: Traffic representation. The images on the right represent the ground truth, where red pixels are attack aggregates and green pixels are benign aggregates. We observe a red horizontal line in both images, which means the attack comes from the same IP and the flows are close in time. The benign data points are more isolated. The images on the left are generated using the method described in Section II-C. On top, we observe that the set of UL models almost detected the entire attack line. At the bottom, the set of UL models only partially detected the attack line.

	F-score	Confusion matrix
Combined model	0.978	$\begin{bmatrix} 35 & 0 \\ 8 & 329 \end{bmatrix}$
Source IP model	0.961	$\begin{bmatrix} 32 & 3 \\ 11 & 326 \end{bmatrix}$
Destination IP model	0.963	$\begin{bmatrix} 38 & 9 \\ 5 & 320 \end{bmatrix}$

TABLE II: F-score and confusion matrix of the CNNs

by destination IP address (combined model). To determine if both representations are complementary for the model, we compared the combined model with a model that takes only images representing the aggregates by source IP address and with another model that takes only images representing the aggregates by destination IP address. The combined model has a better F-score than the two other models, it is free of false positives, however, it accurately detected fewer attacks than the destination IP model.

IV. Conclusion

We proposed an explainable-by-design system to detect network attacks. First, we used UL techniques to detect anomalies on aggregates by source and destination IP. The outputs remain human-readable because security analysts are familiar with IP addresses and the features are few in number. Second, we represented the traffic anomalies detected by an ensemble of UL models as images on which security analysts can see attack patterns. Third, we analyzed the traffic representations using CNN to detect attack patterns. To summarize, we proposed a more transparent system where security analysts can follow the analysis process. The evaluation on the CIC-CSE-IDS2018 dataset [1] showed that our system reached a decent accuracy, but more importantly that the anomaly representations made the remaining errors easy to identify.

References

[1] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116.

[2] J. Vanerio and P. Casas, "Ensemble-learning Approaches for Network Security and Anomaly Detection," in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. Los Angeles CA USA: ACM, Aug. 2017, pp. 1–6.

[3] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," *IEEE Access*, vol. 7, pp. 82 512–82 521, 2019.

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144.

[5] D. Han, Z. Wang, W. Chen, Y. Zhong, S. Wang, H. Zhang, J. Yang, X. Shi, and X. Yin, "DeepAID: Interpreting and Improving Deep Learning-based Anomaly Detection in Security Applications," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 3197–3217.

[6] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F. J. Aparicio-Navarro, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Generation Computer Systems*, vol. 89, pp. 349–359, Dec. 2018.

[7] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," Jan. 2017, pp. 712–717.

[8] P. Casas, J. Mazel, and P. Owezarski, "UNADA: Unsupervised Network Anomaly Detection Using Sub-space Outliers Ranking," in *10th IFIP Networking Conference (NETWORKING)*, vol. LNCS-6640. Springer, May 2011, pp. 40–51.

[9] J. Dromard, G. Roudière, and P. Owezarski, "Online and Scalable Unsupervised Network Anomaly Detection Method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, Mar. 2017.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau, "Scikit-learn: Machine Learning in Python," *MACHINE LEARNING IN PYTHON*, p. 6, 2011.

[11] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python Toolbox for Scalable Outlier Detection," p. 7, 2019.

[12] C. Minh, K. Vermeulen, C. Lefebvre, P. Owezarski, and W. Ritchie, "An explainable-by-design ensemble learning system to detect unknown network attacks," Cyblex Technologies ; LAAS-CNRS, Tech. Rep., Nov. 2022. [Online]. Available: https://hal.laas.fr/hal-03868401

[13] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00. New York, NY, USA: Association for Computing Machinery, May 2000, pp. 93–104.

[14] F. Angiulli and C. Pizzuti, "Fast Outlier Detection in High Dimensional Spaces," in *Principles of Data Mining and Knowledge Discovery*, ser. Lecture Notes in Computer Science, T. Elomaa, H. Mannila, and H. Toivonen, Eds. Berlin, Heidelberg: Springer, 2002, pp. 15–27.

[15] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "COPOD: Copula-Based Outlier Detection," *arXiv:2009.09463 [cs, stat]*, Sep. 2020.