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

Céline Minh, Kevin Vermeulen, Cédric Lefebvre, Philippe Owezarski, William Ritchie

Advancement Report

Collaboration and research contract
#247966

CNRS & Cyblex Technologies

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

Authors:
Céline Minh
Kevin Vermeulen
Cédric Lefebvre
Philippe Owezarski
William Ritchie

November 23, 2022

Abstract

Security analysts have to deal with a large volume of network traffic to identify and prevent cyber attacks daily. To assist them in this task, network intrusion detection systems (NIDSs) monitor the network and raise alarms when they identify suspicious events or anomalies. We investigate unsupervised learning techniques to analyze network traffic captures because they are more likely to detect unknown attacks. There is a wide variety of unsupervised learning algorithms, whose results seem complementary, but their lack of explainability makes it difficult to find out which one of their results is right. Our system intends to reconstruct attack patterns from a set of unsupervised anomaly detectors outputs, and show them to security analysts. Therefore, we introduce an explainable-by-design system to detect attacks on networks, and evaluated its accuracy on the CSE-CIC-IDS2018 dataset [17].

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

Céline Minh

November 23, 2022

Contents

Chapter 1

Introduction

Security analysts have to deal with a large volume of network traffic to identify and prevent cyber attacks daily. To assist them in this task, network intrusion detection systems (NIDSs) monitor the network and raise alarms when they identify suspicious events or anomalies [3, 4, 8]. Those alarms can correspond to real attacks (true positive) or the system can be mistaken and raise alarms when the network is not under attack (false positive). In the worst case, NIDSs can remain silent and not raise any alarm when the network is under attack (false negative).

Machine learning (ML) is a very promising technology for assisting security analysts as it allows accurate identification of attacks on large amounts of data without explicitly defining attack signatures, as opposed to rule-based systems. In particular, unsupervised learning (UL) identifies anomalies without requiring any training dataset (i.e., a set of labeled flows to learn attack signatures), as opposed to supervised learning techniques, which can struggle to detect attacks that are not part of the training dataset. Therefore, we will investigate unsupervised learning techniques because they are more likely to detect unknown attacks.

One of the problems with machine learning models is that a same network flow can be considered as an attack by some models but not by others, and their binary outputs do not provide enough information to evaluate the relevance of their results. In other words, we are aware that the models we use still make some errors, but we do not know what triggers these and we have thus no means of further analyzing them manually.

In addition, there is no single model which always takes the right decision: we observed that the accuracy of different models varies according to the given dataset or the ML algorithm; different models make different mistakes. So, one problem when using ML is choosing which model to use. This is a perfect case for using ensemble methods [18, 5, 1] where multiple models can contribute to a final decision. The *ensemble learning* approach [18, 5, 1] consists of combining multiple models, called base learners, to build more accurate systems. Indeed, some models can be better at detecting specific types of attack (e.g., brute-force, and DoS). A possible way to address the problem is to use a weighted-majority voting, where weights are assigned based on the base models' accuracy. But in some cases, we observed that a weak model, in terms of global accuracy, can accurately detect an attack, whereas more globally accurate models can

fail to detect it. Therefore, voting approaches do not seem to us to go in the right direction. To take full advantage of the complementarity of the models, we propose a new approach which is to systematically present the results of each base learner to the end user in a manner 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 [16, 8]. In current NIDSs, actions to solve security issues are made by security analysts, based on the system analysis. Therefore, providing evidence of ML models results in an intelligible way is crucial for analysts to trust the system. Explainability is not only important for improving the collaboration between security analysts and Artificial Intelligence (AI) systems, but also for engineers and researchers to design more accurate systems: understanding why and where a model may fail can help fix the issues, so that the model does not reproduce the same errors.

We propose a method that will simultaneously allow a user to take advantage of ensemble learning from multiple base models to enhance detections and visualize the results of all the base models to gain insights into how these detections are made. Our method analyzes base models by acting on their inputs and outputs only, so that we can compare models with different algorithms.

A requirement for our work was to provide visual representations of traffic anomalies, as the cybersecurity company *Cyblex Technologies* that coordinates the project thinks it will help security analysts understand what is happening on the network efficiently. We proposed to generate images based on all the unsupervised learning detections, and use convolutional neural networks (CNNs), a supervised layer, to analyze them and give the final result. Our system is expected to preserve properties of unsupervised systems, including the detection of new attacks, as its supervised layer is used to analyze the behavior of base models rather than pure network traffic data.

Our idea is to look at all the alarms raised by multiple detectors during a macroscopic time window and identify patterns of attacks [6, 19]. As an example, if we consider a brute-force attack scenario where the attacker intends to find a password by trying repeatedly to authenticate from one machine with all the words from a dictionary, we expect to observe a sequence of alarms from the same IP address. Our system intends to reconstruct attack patterns from a set of anomaly detectors outputs and show them to security analysts.

We introduce an explainable-by-design system that analyzes a set of unsupervised learning models to detect network attacks. Our contributions are as follows:

1. A more transparent ML-based NIDS, whose analysis is understandable,
2. A representation of network anomalies as an image that the user can interpret,
3. An ensemble learning method that uses CNN to combine models.

Chapter 2

Related Work

Our research investigates mechanisms to reconstruct attack patterns based on multiple unsupervised learning detection techniques. Our work addresses issues related to (1) ensemble learning, (2) attack patterns detection, and (3) explainable learning. This section will address these themes in this order, and position our work in relation to recent studies.

Regarding ensemble learning systems for network anomaly detection, Vanerio and Casas [18] compared multiple *meta-learners* for detecting attacks. A meta-learner combines outputs of a set of base learners to return a more accurate detection. As an example of meta-learner, the authors considered a weighted-majority voting algorithm where the weights were defined depending on the accuracy of the base learner. We have taken a different approach in which the meta-learner is another ML layer, a CNN, that takes as input a visual representation of the base learners' outputs and detects attack patterns on them. Mirsky et al. [13] proposed Kitsune, an ensemble of autoencoders for detecting network anomalies. The system relies on autoencoders, which are often considered as unsupervised learning techniques because they use unlabeled data, but autoencoders still require a training phase on normal data. Kitsune stacks autoencoders, by using another autoencoder as a meta-learner to process their anomaly scores. The approach to combine base learners is close to ours, except that we address heterogeneous algorithms and data representations.

On a different topic, Zhou et al. [22] proposed a system using LSTM to detect multi-stage attacks. Their model treats sequences of alerts generated by the NIDS Snort and addresses the problem of long-term dependency between the alerts. Ghafir et al.[7], proposed a system for the detection and prediction of advanced persistent threats (APT). The system uses the Hidden Markov Model (HMM) to detect the most probable APT scenario given the alarms. Then, it forecasts the next step of the ongoing APT. A significant difference with our work is that their system is trained on a given attack lifecycle [9, 12], whereas our system learns attack patterns directly from the data. Wang et al. [19] converted raw traffic data (pcap files) into images and also noticed attack patterns.

Wei et al. [20] observed that current general-purpose explanation methods, such as SHAP [11] and LIME [16], are not suitable for NIDSs because they do not handle dependencies of network flows' features. To overcome this issue, the authors proposed data-driven explanation methods for DL-based NIDSs, that are based on history inputs. Based on the extracted feature importance,

the system generates defense rules to block malicious activities. Han et al. [8] addressed ML explainability by proposing a system to interpret existing unsupervised, DL-based NIDS. The system analyzed a given model detection by providing the most important features and describing their meaning so that a security expert can understand them. Instead of relying on *ad hoc* interpretations, we designed a more transparent system and provided a representation of traffic anomalies to help the security analyst monitor the traffic and identify potential errors.

Chapter 3

Methods

3.1 Approach

In this paper, we investigate unsupervised learning (UL) techniques as they are more likely to detect new attacks. To combine multiple models with different algorithms, we intended to characterize their behavior without assuming knowledge about the algorithms, by playing on the inputs and outputs.

Regarding the inputs, we needed a data representation that was semantically interesting for UL models to extract attack signatures (Section 4.2). Similarly to UNADA [3, 4], we aggregated network flows by source IP address and by destination IP address to distinguish point-to-point (e.g., brute-force) from distributed attacks (e.g., DDoS). Related work on unsupervised anomaly detection also used aggregated flows and consider other aggregations keys.

To quantify the degree of abnormality (Section 3.4), we played on subspaces of features: we applied UL algorithms on different subspaces of features and summed their results. Our anomaly scores are again model-agnostic, unlike the ones provided by the scikit-learn library that depend on the ML algorithms. On the same problem, Mirsky et al. [13] tried to approximate the normal behavior of traffic using autoencoders, and used the root mean squared error (RMSE) between the expected behavior and the observed behavior as anomaly score. Our approach is different because we did not try to learn the behavior of the system based on its previous states: this is because traffic can change drastically in short time periods.

To go further in the characterization of models' outputs, we introduced a representation (Section 3.5) 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 the anomalies?). This representation was analyzed by another layer of ML, a CNN, to identify attack patterns. In this sense, Wang et al. [19] identified attack patterns using CNN on a visual representation of pcap files. Our approach is different because we used a CNN as a meta-learner, that analyzed a combination of anomaly detectors to detect attacks.

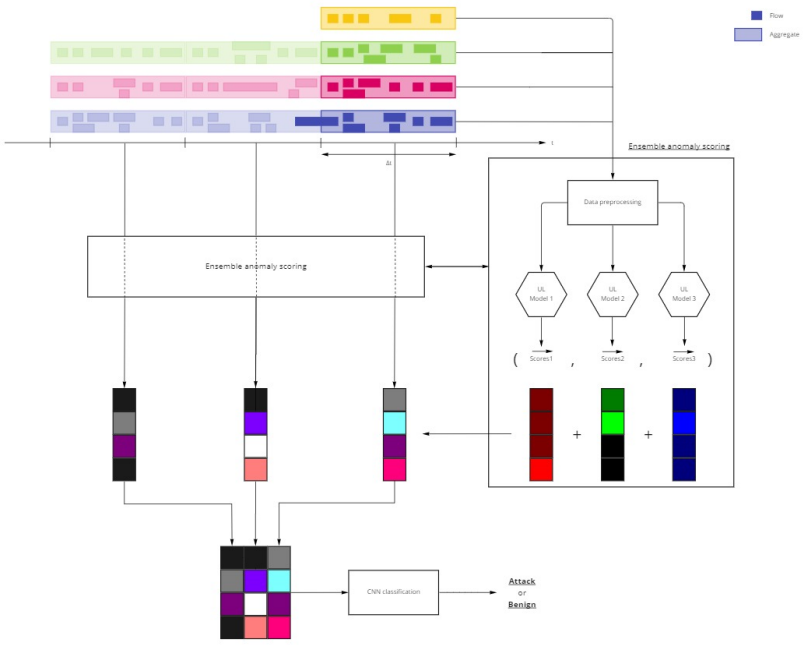

Figure 3.1: System overview

3.2 System overview

We designed a ML-based NIDS that uses a set of unsupervised learning models to detect network anomalies, and a supervised learning layer to combine them. The system analyzes a network traffic capture and raises an alarm if the network is under attack. The system, described in Figure 3.1, takes as input a network traffic capture (pcap files). A probe, that was not designed within this project, aggregates the packets into flows and extracts flow features such as the total number of forward packets of a TCP flow. Then, the system aggregates flows and computes aggregate features from flow features (Section 4.2). The aggregates are analyzed by a set of unsupervised learning (UL) models to detect anomalies and quantify their degree of abnormality (Section 3.4). The system generates images from the anomaly scores that represent traffic during a time window. Finally, convolutional neural networks (CNNs) analyze those images to recognize attack patterns (Section 3.5).

3.3 Aggregation of network flows

ML-based NIDSs usually perform on a given set of flow features, such as the flow duration, the number of packets in the forward direction, the average size of packets, etc. As an example, the CIC-CSE-IDS2018 dataset [17] proposes 83 flow features to evaluate ML-based NIDSs. As clustering algorithms perform poorly on high-dimensional data, we considered aggregates of flows, with fewer, statistical features.

The system uses two types of aggregates: aggregates by source IP address

and aggregates by destination IP address. In fact, some attacks such as Distributed Denial of Service (DDoS) involve IP addresses from many sources towards a single destination, and other anomalies such as network scans involve IP addresses from a single source towards multiple destinations. So, the aggregates perspective helps characterize the attacks and highlights attack patterns.

As network traffic can evolve very quickly, clustering algorithms were used to detect abnormal aggregates among all aggregates during the same time interval Δ_t. This was done to avoid detecting anomalies corresponding to a legitimate increase in network traffic. In addition, traffic can differ depending on the time of the day or whether it is during a working day or on holidays. By comparing the current traffic to a baseline, a system could raise an anomaly if an employee is working unusually late, for example.

To summarize, network flows that ended during a time interval Δ_t were aggregated by source IP address and aggregates by destination IP address. The aggregates perspective was designed to better identify attack patterns, and also, the resulting aggregates are more readable for the end user, who is the security analyst. In fact, there are fewer aggregate features than flow features, which makes the data more readable, and security analysts are familiar with IP addresses.

3.4 Unsupervised anomaly scoring

This component is composed of a set of anomaly detectors. It takes as input the aggregate features (Table 4.1), either by source IP address or by destination IP address, that ended during a same time frame of duration Δ_T and returns a set of matrices. Each matrix contains the anomaly scores of all the aggregates computed by an unsupervised anomaly detector.

Unlike unsupervised anomaly detectors, supervised learning models require a labeled training dataset to classify a flow. Recent learning techniques allow to accurately recognize attack flows that are statistically similar to the ones from the training dataset. However, those models can struggle to identify attacks that were not in the training dataset. To be able to detect new attacks (0-day attacks), we considered unsupervised learning techniques. The system used clustering techniques to detect anomalies, data samples that statistically differ from others, and did not require any knowledge dataset.

Clustering algorithms can give different outputs from the same input data, i.e., they may not detect the same aggregates as anomalies. Sometimes, a model can give the correct result where a more accurate model fails. To take advantage of the complementarity of the models' detections, we combined multiple clustering algorithms, referred to as base learners, to build a more accurate ensemble system. Instead of considering the binary output (normal or abnormal) of the base learners, we considered an anomaly score that quantifies the degree of abnormality of an aggregate according to a model.

Machine learning libraries, such as scikit-learn or PyOD provide an anomaly score function for most of the clustering algorithms implemented, but those anomaly scores depend on the decision function of the models. To compare the results using different clustering algorithms, we use a model-agnostic quantification of anomalies, previously defined in UNADA [3, 4].

To define anomaly scores, we considered all the subspaces of k features among

all the n aggregates' features and perform 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 the number of combinations of k features among n for a completely abnormal aggregate, which is:

$\binom{n}{k} = \frac{n!}{k!(n-k)!}$

To select a set of base learners among a list of algorithms, we considered all possible combinations of N algorithms and computed the anomaly scores of the aggregates by source IP address on the one hand and by destination IP address on the other hand on a training dataset. We sorted the combination using two criteria. First, the number of accurate alarms, as false negatives are considered more critical than false positives. Then, the number of false alarms. We selected the subset of N models that detected the most attacks with the fewest false alarms. The alarms raised by the chosen subset will be the input of a meta-learner (Section 3.5), a model that will make the final analysis, given the analysis of the base learners.

3.5 Attack patterns recognition

In the previous section, we selected a set of unsupervised learning models that detected anomalies on aggregates by source IP address and on aggregates by destination IP address. To reinforce the system, we studied the sequences of anomalies detected by each base learner during a time frame Δ_T, greater than the time intervals Δ_t. We proposed to represent the output 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. Then, the output of a set of base learners was a set of matrices.

To visualize the analysis of the base learners, we can assign them a color channel (red, blue, or green). The color intensity of a pixel was proportional to the anomaly score generated by the model for the aggregate. Thus, a white pixel means that all the models diagnosed the aggregate as very abnormal, and a black pixel means that none of the models raised an alarm (Figure 3.2 and Figure 3.3).

We can then analyze these images using deep neural networks (DNNs) such as convolutional neural networks (CNNs). Therefore, we used a CNN as a meta-learner to identify attack patterns on the traffic representations. The attack pattern recognition module takes two inputs:

1. The image generated from the outputs of the base learners on aggregates by source IP,

2. The image generated from the outputs of the base learners on aggregates by destination IP during the same time frame Δ_T,

The model required a knowledge database, a labeled dataset for training and validation. Each generated image was labeled *attack* if it contained at least one attack aggregate, and *benign* otherwise. The CNN gave the final result of the system, meaning if the network was under attack during the time frame Δ_T.

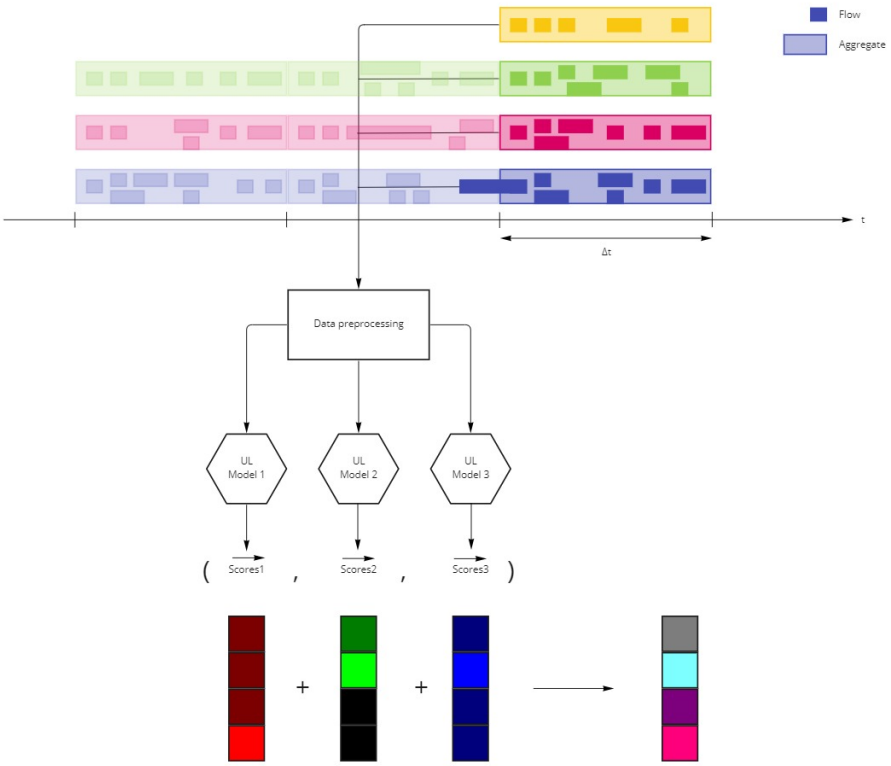

Figure 3.2: Representation of the analysis of 3 base learners during a time interval Δ_t. This analysis generates a vertical segment that is part of the image generated in Figure 3.3. Network flows during Δ_t are grouped into aggregates (4 in this example). Then, aggregates features are extracted and preprocessed to be analyzed by N UL models ($N = 3$ in this example). Each model scores the degree of abnormality of each aggregate. A color is assigned to each model so that an anomaly score vector was encoded into a monochrome segment. By superposing the monochrome segments of the N models, we obtain a colored segment that represents the anomalies detected by all the models during Δ_t.

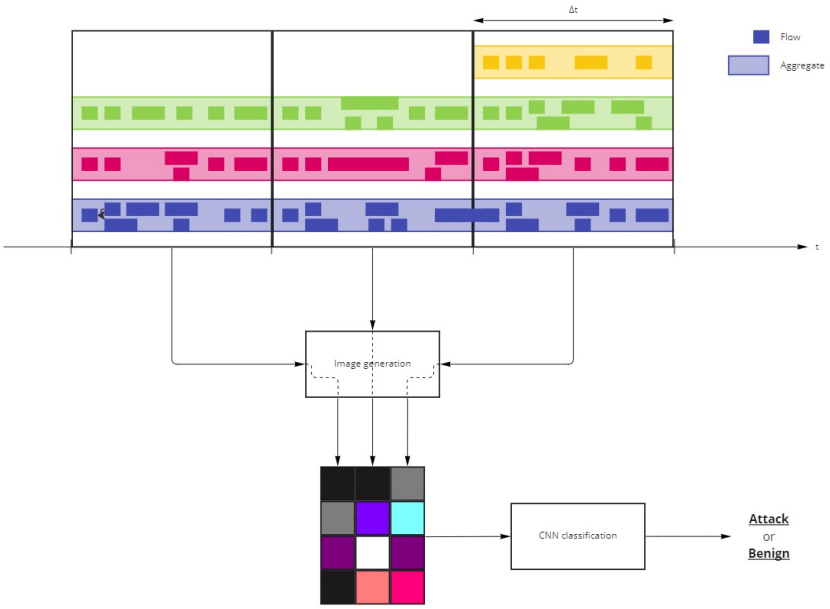

Figure 3.3: Representation of the analysis of 3 base learners during a timeframe Δ_T. The timeframe Δ_T consists in this example of 3 consecutive time intervals Δ_t. The flows that occurred during each time interval Δ_t are analyzed as in Figure 3.2. Each time interval Δ_t is represented by a colored segment. By putting the 3 segments side-by-side we obtain an image that represents the traffic anomalies detected by $N = 3$ models during the timeframe Δ_T.

Chapter 4

Experiments

4.1 Dataset

The proposed system was evaluated on the CIC-CSE-IDS2018 dataset [17]. Sharafaldin et al. [17] 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.

4.2 Aggregation of network flows

The brute-force, DoS, and benign network flows of the CIC-CSE-IDS2018 dataset [17] were grouped by a time interval Δ_t of 2 minutes. Then, the flows were aggregated by source IP address and by destination IP address. The aggregates features described in Table 4.1 are statistical features computed from the flows features. Table 4.2 and Table 4.3 show the proportion of aggregates by source IP address and by destination IP address respectively, by attack category (brute-force, DoS, and benign).

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 4.1: Aggregates features

Category type	Count	Proportion (%)
brute-force	95	1.457950
DoS	6	1.028238
Benign	6354	97.513812

Table 4.2: Proportion of aggregates by source IP address by attack category

Category type	Count	Proportion (%)
brute-force	95	2.725186
DoS	67	1.921974
Benign	3324	95.352840

Table 4.3: Proportion of aggregates by destination IP address by attack category

4.3 Unsupervised anomaly scoring

The aim of this section is to select a set of unsupervised learning algorithms to implement the proposed system.

For this study, we considered unsupervised algorithms with different mechanisms for anomaly detection. We used implementations from the scikit-learn library [15] and PyOD [21] libraries:

- *Isolation Forest* detects anomalies using isolation. The algorithm is constructed recursively: each step consists on randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The anomaly score is the number of splittings required to isolate a data sample.

- *Local Outlier Factor* detects anomalies by comparing the local density of a data sample with the local density of its neighbors.

- *DBSCAN* defines clusters by grouping the data samples that are closed according to a distance measure. If there are enough data samples in a group, they form a cluster, otherwise, they are outliers.

- *One-Class SVM* defines a separating hyperplane and detects anomalies based on a distance between the data samples and the hyperplane.

- *Unsupervised KNN* [2] is also a proximity-based model that uses the distance to the kth nearest neighbor as outlier score.

- *COPOD* [10] is a probabilistic model.

We then sought the best possible combination of three algorithms by iteratively testing them all on the aggregates defined in Section 4.2 grouped by a time interval of $\Delta_t = 2minutes$. In the following, we call *agreement zone* the samples where all models detected the same thing. We call *grey zone* the samples where the models' detections differ. For the aggregates by source IP address, there are 56 false negatives in the agreement zone, meaning that 56 attacks were not detected by any model. For the aggregates by destination IP address, there are 98 false negatives in the agreement zone.

Subset	Accurate attack detections	False alarms
(LocalOutlierFactor, KNN, COPOD)	84	2140
(IsolationForest, LocalOutlierFactor, KNN)	84	2154
(IsolationForest, LocalOutlierFactor, COPOD)	84	2159
(DBSCAN, KNN, COPOD)	84	2838
(IsolationForest, DBSCAN, KNN)	84	2851
(IsolationForest, DBSCAN, COPOD)	84	2852
(LocalOutlierFactor, DBSCAN, KNN)	84	3104
(LocalOutlierFactor, DBSCAN, COPOD)	84	3107
(IsolationForest, LocalOutlierFactor, DBSCAN)	84	3112
(IsolationForest, LocalOutlierFactor, OneClassSVM)	84	4138
(IsolationForest, DBSCAN, OneClassSVM)	84	4138
(IsolationForest, OneClassSVM, KNN)	84	4138
(IsolationForest, OneClassSVM, COPOD)	84	4138
(LocalOutlierFactor, DBSCAN, OneClassSVM)	84	4138
(LocalOutlierFactor, OneClassSVM, KNN)	84	4138
(LocalOutlierFactor, OneClassSVM, COPOD)	84	4138
(DBSCAN, OneClassSVM, KNN)	84	4138
(DBSCAN, OneClassSVM, COPOD)	84	4138
(OneClassSVM, KNN, COPOD)	84	4138
(IsolationForest, KNN, COPOD)	82	1063

Table 4.4: Detected attacks and false alarms of sets of unsupervised learning models on aggregates by source IP address

All the subsets of 3 models were evaluated on the *grey zone*. In the following, we refer as *accurate attack detections* the number of attack aggregates that were detected as an anomaly by at least one model of the subset. The number of *false alarms* for the subset is the number of benign samples detected as anomalies by at least one model. Table 4.4 and Table 4.5 show the number of accurately detected attacks and false alarms for each possible subset of 3 algorithms.

The combination that was chosen to analyze both aggregates by source IP address and aggregates by destination IP address was LocalOutlierFactor, KNN, COPOD. This makes sense because these algorithms have a different mechanism to detect anomalies, and so are more likely to be complementary.

4.4 Attack patterns recognition

This section describes the implementation and the results of the attack patterns detection module.

We built a dataset of images that represent the network traffic over a time frame of $\Delta_T = 30 minutes$, as described in Section 3.5. A color has been assigned to each model:

- Red for Local Outlier Factor
- Green for KNN
- Blue for COPOD

Subset	Accurate attack detections	False alarms
(IsolationForest, LocalOutlierFactor, KNN)	58	541
(LocalOutlierFactor, KNN, COPOD)	58	541
(IsolationForest, LocalOutlierFactor, COPOD)	58	546
(IsolationForest, LocalOutlierFactor, DBSCAN)	58	672
(IsolationForest, DBSCAN, KNN)	58	672
(IsolationForest, DBSCAN, COPOD)	58	672
(LocalOutlierFactor, DBSCAN, KNN)	58	672
(LocalOutlierFactor, DBSCAN, COPOD)	58	672
(DBSCAN, KNN, COPOD)	58	672
(IsolationForest, OneClassSVM, KNN)	58	675
(IsolationForest, OneClassSVM, COPOD)	58	675
(LocalOutlierFactor, OneClassSVM, KNN)	58	675
(OneClassSVM, KNN, COPOD)	58	675
(IsolationForest, LocalOutlierFactor, OneClass...	58	676
(IsolationForest, DBSCAN, OneClassSVM)	58	676
(LocalOutlierFactor, DBSCAN, OneClassSVM)	58	676
(LocalOutlierFactor, OneClassSVM, COPOD)	58	676
(DBSCAN, OneClassSVM, KNN)	58	676
(DBSCAN, OneClassSVM, COPOD)	58	676
(IsolationForest, KNN, COPOD)	56	317

Table 4.5: Detected attacks and false alarms of sets of unsupervised learning models on aggregates by destination IP address

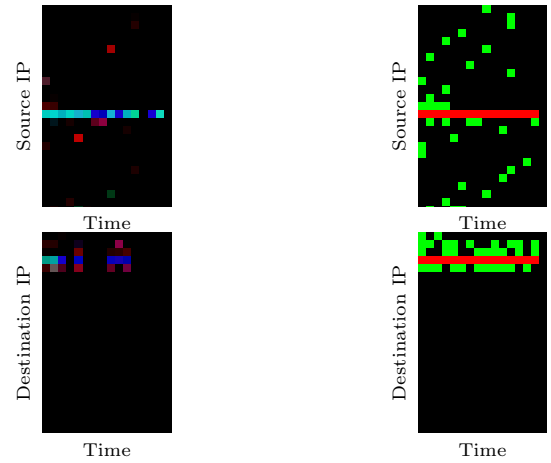

Figure 4.1: Traffic representation

Figure 4.1 are examples of images for an attack and a benign period. First, there are many red-green dots and a few light blue points, which is consistent with the distribution of anomaly scores.

In the attack image, we observe a distinctive horizontal line that corresponds to an attack. In the considered period of the CSE-CIC-IDS2018 dataset, the attack comes from the same IP and the flows are very close in time. The benign data points are more isolated.

We generated a dataset of 2000 images that represent periods of 30 minutes of CSE-CIC2018. An image that contains at least one attack aggregate is labeled as an attack. The images are padded to have the same dimensions. The images representing the aggregates by source IP address and the aggregates by destination IP address during the same time window were coupled. The resulting dataset was split into a training and a validation dataset. The models' hyperparameters were optimized using KerasTuner[14].

Table 4.6 shows the F-score and confusion matrix of the resulting model, and compares it 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. The combined model is free of false positives, however it accurately detected fewer attacks than the model taking only the images representing the aggregates by destination IP address.

The false positives are shown in Figure 4.2. Images 1, 2, 3, 4, and 8 are very dark: no base learner detected very abnormal aggregates. In images 6 and 7, there are a few light spots in the representations of aggregates by source IP address, but there are spread on the image. Image 5 corresponds to the end of a DoS attack where a single machine targets another one. There, the anomaly detectors detected a sequence of anomalies coming from the attacker, which forms a clear light line as we expected. Unfortunately, the CNN failed to detect an attack pattern on this image, so this error must be investigated.

	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 4.6: F-score and confusion matrix of the CNNs

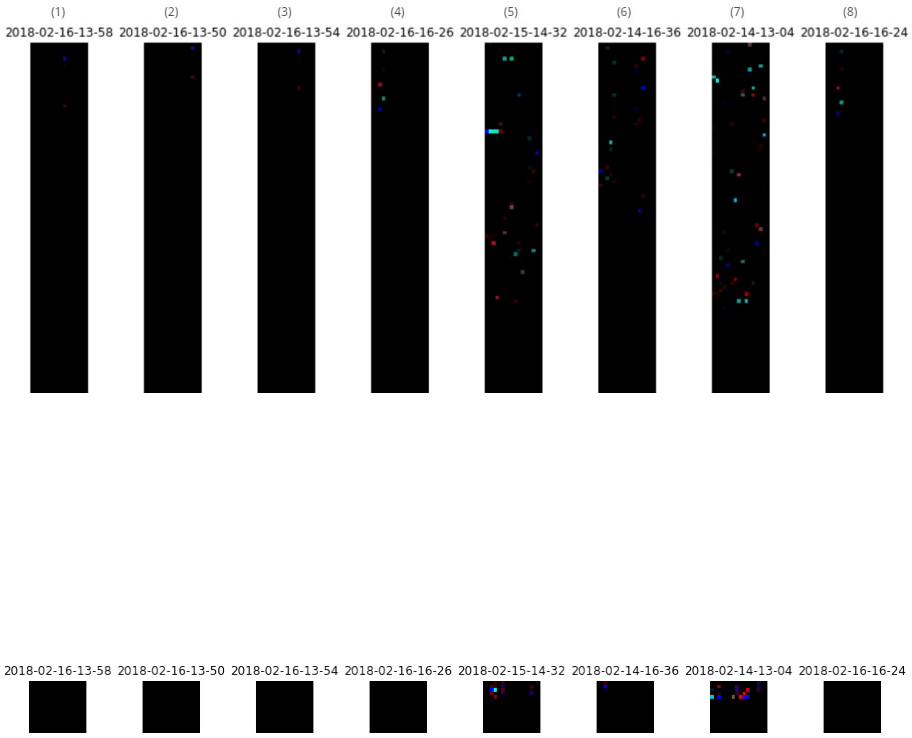

Figure 4.2: False negatives

Chapter 5

Discussion

We proposed a system that combines multiple anomaly-based detectors using CNN on a 2-D representation of the traffic. The combination allowed to improve significantly the performance of the system. However, the performance of the system is below other state-of-the-art systems using neural networks [19]. On this aspect, the intermediate representation of the traffic anomalies can help a security analyst identify errors.

The final system reached a decent accuracy, but not a competitive one. However, all the attacks were at least partially detected. To reinforce the system, we could consider using video processing techniques, where we could add a time series layer to our image analysis to detect pattern changes over time. In addition, the image dataset was too small for a CNN training. The observed attack patterns were also simple, with one victim and one attacker who was sending packets frequently. The attacker could change his/her IP address and space out his/her actions over time. This is why we will design a dataset with more complex attack patterns.

For this paper, we chose to study aggregates on a fixed time interval of two minutes. We may miss small but important parts of traffic that we could identify on a more fine-grained analysis. Thus, we should consider dynamically defining the time interval. In the same way, the set of models was chosen once. The selection of models could be dynamic, as some models may perform better depending on the context.

So far, we have only looked at aggregates by source IP. Using aggregates by destination IP is more relevant for identifying some sorts of attacks, such as DDoS. In addition, it is currently hard to distinguish attacks from some sorts of legitimate traffic, for example, DDoS from peer-to-peer. The sequences of flags, the evolution of their proportion, can help identify attacks. To prevent the curse of dimensionality, we will continue to perform macroscopic analyses. But, we will systematically analyze multiple, well-chosen dimensions, and combine those analyses to identify attacks.

In further work, we will evaluate our system on a larger dataset, and evaluate its capacity to detect new attacks.

Chapter 6

Conclusion

We proposed an explainable-by-design system to detect attacks on networks. First, we used unsupervised learning 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 unsupervised learning models as images on which a security analyst can see attack patterns. Third, we analyzed the traffic representations using CNN to detect attack patterns. To summarize, we proposed a more transparent system whose design allows a security analyst to follow the steps of the analyses. The evaluation on the CIC-CSE-IDS2018 dataset [17] showed that our system reached a decent accuracy, but more importantly that the anomaly representations made the remaining errors easy to identify.

Bibliography

[1] Majjed Al-Qatf, Yu Lasheng, Mohammed Al-Habib, and Kamal Al-Sabahi. 2018. Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection. *IEEE Access* 6 (2018), 52843–52856. https://doi.org/10.1109/ACCESS.2018.2869577

[2] Fabrizio Angiulli and Clara Pizzuti. 2002. Fast Outlier Detection in High Dimensional Spaces. In *Principles of Data Mining and Knowledge Discovery (Lecture Notes in Computer Science)*, Tapio Elomaa, Heikki Mannila, and Hannu Toivonen (Eds.). Springer, Berlin, Heidelberg, 15–27. https://doi.org/10.1007/3-540-45681-3_2

[3] Pedro Casas, Johan Mazel, and Philippe Owezarski. 2011. UNADA: Unsupervised Network Anomaly Detection Using Sub-space Outliers Ranking. In *10th IFIP Networking Conference (NETWORKING)*, Vol. LNCS-6640. Springer, 40–51. https://doi.org/10.1007/978-3-642-20757-0_4

[4] Juliette Dromard, Gilles Roudière, and Philippe Owezarski. 2017. Online and Scalable Unsupervised Network Anomaly Detection Method. *IEEE Transactions on Network and Service Management* 14, 1 (March 2017), 34–47. https://doi.org/10.1109/TNSM.2016.2627340

[5] Xianwei Gao, Chun Shan, Changzhen Hu, Zequn Niu, and Zhen Liu. 2019. An Adaptive Ensemble Machine Learning Model for Intrusion Detection. *IEEE Access* 7 (2019), 82512–82521. https://doi.org/10.1109/ACCESS.2019.2923640

[6] Ibrahim Ghafir, Mohammad Hammoudeh, Vaclav Prenosil, Liangxiu Han, Robert Hegarty, Khaled Rabie, and Francisco J. Aparicio-Navarro. 2018. Detection of Advanced Persistent Threat Using Machine-Learning Correlation Analysis. *Future Generation Computer Systems* 89 (Dec. 2018), 349–359. https://doi.org/10.1016/j.future.2018.06.055

[7] Ibrahim Ghafir, Konstantinos G. Kyriakopoulos, Sangarapillai Lambotharan, Francisco J. Aparicio-Navarro, Basil Assadhan, Hamad Binsalleeh, and Diab M. Diab. 2019. Hidden Markov Models and Alert Correlations for the Prediction of Advanced Persistent Threats. *IEEE Access* 7 (2019), 99508–99520. https://doi.org/10.1109/ACCESS.2019.2930200

[8] Dongqi Han, Zhiliang Wang, Wenqi Chen, Ying Zhong, Su Wang, Han Zhang, Jiahai Yang, Xingang Shi, and Xia Yin. 2021. 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. ACM, Virtual Event Republic of Korea, 3197–3217. https://doi.org/10.1145/3460120.3484589

[9] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. 2011. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. (2011), 14.

[10] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. 2020. COPOD: Copula-Based Outlier Detection. *arXiv:2009.09463 [cs, stat]* (Sept. 2020). arXiv:2009.09463 [cs, stat]

[11] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.

[12] Dan McWhorter. 2013. Mandiant Exposes APT1 — One of China's Cyber Espionage Units & Releases 3,000 Indicators. *Mandiant, February* (2013).

[13] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA. https://doi.org/10.14722/ndss.2018.23204

[14] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. 2019. KerasTuner.

[15] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. 2011. Scikit-Learn: Machine Learning in Python. *MACHINE LEARNING IN PYTHON* (2011), 6.

[16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "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 (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[17] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, 108–116. https://doi.org/10.5220/0006639801080116

[18] Juan Vanerio and Pedro Casas. 2017. 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*. ACM, Los Angeles CA USA, 1–6. https://doi.org/10.1145/3098593.3098594

[19] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Y. Sheng. 2017. Malware Traffic Classification Using Convolutional Neural Network for Representation Learning. 712–717. https://doi.org/10.1109/ICOIN.2017.7899588

[20] Feng Wei, Hongda Li, Ziming Zhao, and Hongxin Hu. [n. d.]. XNIDS: Explaining Deep Learning-based Network Intrusion Detection Systems for Active Intrusion Responses. ([n. d.]), 18.

[21] Yue Zhao, Zain Nasrullah, and Zheng Li. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. (2019), 7.

[22] Peng Zhou, Gongyan Zhou, Dakui Wu, and Minrui Fei. 2021. Detecting Multi-Stage Attacks Using Sequence-to-Sequence Model. *Computers & Security* 105 (June 2021), 102203. https://doi.org/10.1016/j.cose.2021.102203