Towards understanding alerts raised by unsupervised network intrusion detection systems

Maxime Lanvin, Pierre-François Gimenez, Yufei Han, Frédéric Majorczyk, Ludovic Mé, Eric Totel

Towards Understanding Alerts raised by Unsupervised Network Intrusion Detection Systems

Maxime Lanvin
CentraleSupélec, Univ. Rennes, IRISA
France
maxime.lanvin@centralesupelec.fr

Pierre-François Gimenez
CentraleSupélec, Univ. Rennes, IRISA
France
pierre-francois.gimenez@centralesupelec.fr

Yufei Han
Inria, Univ. Rennes, IRISA
France
yufei.han@inria.fr

Frédéric Majorczyk
DGA-MI, Univ. Rennes, IRISA
France
frederic.majorczyk@intradef.gouv.fr

Ludovic Mé
Inria, Univ. Rennes, IRISA
France
ludovic.me@inria.fr

Eric Totel
Samovar, Télécom SudParis, Institut Polytechnique de Paris
France
eric.totel@telecom-sudparis.eu

ABSTRACT

The use of Machine Learning for anomaly detection in cyber security-critical applications, such as intrusion detection systems, has been hindered by the lack of explainability. Without understanding the reason behind anomaly alerts, it is too expensive or impossible for human analysts to verify and identify cyber-attacks. Our research addresses this challenge and focuses on unsupervised network intrusion detection, where only benign network traffic is available for training the detection model. We propose a novel post-hoc explanation method, called *AE-pvalues*, which is based on the p-values of the reconstruction errors produced by an Auto-Encoder-based anomaly detection method. Our work identifies the most informative network traffic features associated with an anomaly alert, providing interpretations for the generated alerts. We conduct an empirical study using a large-scale network intrusion dataset, CICIDS2017, to compare the proposed *AE-pvalues* method with two state-of-the-art baselines applied in the unsupervised anomaly detection task. Our experimental results show that the *AE-pvalues* method accurately identifies abnormal influential network traffic features. Furthermore, our study demonstrates that the explanation outputs can help identify different types of network attacks in the detected anomalies, enabling human security analysts to understand the root cause of the anomalies and take prompt action to strengthen security measures.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; • **Computing methodologies → Machine learning**.

KEYWORDS

intrusion detection, machine learning, explainable AI (XAI)

1 INTRODUCTION

Recent years have witnessed the flourishing of the deployment of Machine Learning (ML)-based Network Intrusion Detection Systems (NIDS) [3]. ML techniques, especially end-to-end deep learning methods, can conduct automated feature engineering over the attributes of network traffics, such as application protocol, TCP flags, or payload size. Furthermore, ML-assisted NIDS can obtain prompt detection results and offer flexible detection covering various attacks, which help security operation teams (SOCs) reach fast responses to emerging new security incidents during day-to-day security practices.

Despite its advantages, ML-driven Network Intrusion Detection Systems (NIDS) are susceptible to producing a high rate of false positives in their detection output. In light of the increasing volume of network traffic in IT assets, a high false positive rate can lead to prohibitively expensive inspection efforts by human security analysts in Security Operations Centers (SOCs), who are tasked with reviewing the raised alerts. This can result in overwhelmed analysts, leading to delayed responses to potential threats, commonly known as "alert fatigue" [12]. The root cause of this issue stems from the black-box nature of ML-based detection methods. Since these methods lack human-understandable interpretations of the detection results, it becomes difficult for analysts to verify and monitor incident alarms triggered by the ML-driven model. Therefore, improving the transparency of the decision logic underlying the detection output of ML-driven NIDS is necessary.

Our research echoes the challenge of developing eXplainable Artificial Intelligence (XAI)-based solutions for Network Intrusion Detection Systems (NIDS). Specifically, our study focuses on unsupervised anomaly detection, as explored in previous works such as [10, 16], where no labeled attacks are available for training the detection model, and only benign traffic data is used to capture the profiles of normal network activities. The core methodology of anomaly detection involves identifying traffic with significantly deviated profiles from normal traffic as abnormal activity. For instance, Leichtnam et al. [16] utilized an Auto-Encoder (AE)-based deep neural network to reconstruct the network traffic data, with any traffic that produced a large reconstruction error being considered an anomaly. Similarly, Ede et al. [10] employed a recurrent

neural network model to predict the next system logs based on previously observed logs, using the prediction result to determine whether the observed log sequence contained abnormal activities.

In recent years, XAI methods, such as LIME [21] and SHAP [18], have been developed to provide post-hoc explanations of classification/detection outputs by identifying the most important features. As post-hoc methods, they are used on top of existing models and can be combined with any Machine Learning model architectures, which offers great flexibility in practices of XAI techniques. However, these methods face practical challenges when applied to security analysis. Firstly, SHAP is computationally expensive and suffers from inclusion of unrealistic data instances when features are correlated. Secondly, although LIME is computationally efficient, its linear surrogate model cannot accurately approximate complex model architectures. Additionally, neither SHAP nor LIME is adapted to a unsupervised context where only benign data is available for training. While DeepCase and ROULETTE [4] have been proposed as alternatives, they have limitations. DeepCase assumes that sequential causality exists between logs of normal system behaviors, which may not hold true in complex and environment-dependent correlations between network traffic attributes. ROULETTE is a supervised approach and uses actual classes of data for the learning task, which is a stronger assumption than our work that only uses benign traffic for learning. Therefore, in our work, we propose a novel XAI-based solution to NIDS that is free from such assumptions and specifically designed for unsupervised anomaly detection.

Our contribution can be summarised in the following perspectives:

- We propose a new XAI approach, namely *AE-pvalues*[1], which is designed to not only identify the important features responsible for unsupervised anomaly detection but also help categorize the detected anomalies into specific attack types. We instantiate our study with a state-of-the-art unsupervised Auto-Encoder-based NIDS method [16].
- We organize a comprehensive experimental study to evaluate quantitatively the usefulness of the explanation results produced by *AE-pvalues* and various state-of-the-art XAI methods on the CICIDS2017 dataset. We demonstrate that our method can offer significantly more accurate explanation results to the identified security incidents.
- We organize a use-case study which is divided into two parts. First, we show that the explanations provided can be practically used to categorize the detected security incidents into the corresponding attack types with high precision. In practice, the explanation results from our method can facilitate human experts to understand the campaigns of the detected attack behaviors. Second, we conducted a manual inspection over the explanation results generated by our method on the network attack behaviors in CICIDS2017. We show the explanations are highly consistent with the behavior of the associated attack type.

The remainder of this paper is organized as follows. Section 2 is a presentation of Sec2graph which is the detection method we used to obtain the alerts. The CICIDS2017 intrusion detection dataset is also presented. Section 3 presents related work on XAI approaches used in the context of intrusion detection. Section 4 presents the new XAI technique we propose, and Section 5 contains a benchmark to compare its performance to other XAI techniques. Finally, Section 6 shows how the explanations we produce can be used in practice on alerts generated from the CICIDS2017 intrusion detection dataset. Section 7 concludes the paper.

[1]We provide an implementation here https://gitlab.inria.fr/mlanvin/ae-pvalues

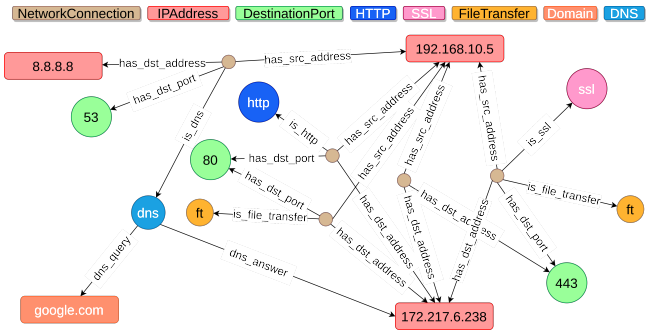

Figure 1: Example of security object graph in *Sec2graph*

2 BACKGROUND

For this work, we used Sec2graph as the network intrusion detection system to explain. The first subsection presents this approach. We then present the CICIDS2017 dataset we use in our experiments.

2.1 Sec2graph

Sec2graph [16] is one of many unsupervised approaches [6, 19] relying on the reconstruction error of an Auto-Encoder to detect anomalies. This approach can be decomposed into three steps: the graph building, its encoding, and finally, how the detection is performed.

2.1.1 Building a graph of security objects. The raw network captures (.pcap files) are analyzed using Zeek to extract logs regarding different network protocols. Then a "security objects graph" is built from these logs to reveal the structure in the data and connect elements of different kinds. The Figure 1 shows an example of a security objects graph constructed from network logs. In this example, a client requested the IP address of the domain `google.com` via the DNS protocol, then contacted the `google.com` server with an HTTP request to obtain an SSL certificate before initiating an HTTPS connection with the same server. Since the graph is constructed from logs of these various protocols, the graph's nodes correspond to the network information in this scenario (client, DNS server, HTTPS server, network ports, network connections, SSL certificate, etc.). These nodes are directly linked when the information they represent appears in the same network event. Due to the diversity of nodes and edges, the graph is heterogeneous. The whole security objects graph model is available in the appendix A.

2.1.2 Encoding a graph of security objects. The second step of the approach is to encode a graph in the form of vectors usable by the Auto-Encoder. In this encoding, each edge of the graph is processed independently and is vectorized. This vector encodes the triplet *(source node, edge type, destination node)*. The encoding of the edge

type, the source, and destination nodes attributes are concatenated to create the vector of each triplet. Remark that there is only one Auto-Encoder but several types of nodes and edges. For this reason, the vectors are sparse: every dimension of the vector related to a type not present in an edge is set to zero.

Sec2graph uses one-hot encoding, a classical technique to encode categorical variables. For each categorical variable, it creates a vector whose size is the number of categories. Each feature of this vector contains a 0, except the feature associated with the variable category that receives a 1. Discrete and categorical attributes are processed differently. For discrete attributes (such as port numbers or IP addresses), the number of possible categories is limited by keeping the most frequent categories and merging the rare categories into a single one. One can observe an example of the categorical encoding on the Figure 3 where the vectors represent the protocol encoding. For continuous attributes (such as flow duration or the number of packets exchanged), a clustering for each attribute is performed with a Gaussian Mixture Model (GMM). The resulting cluster membership labels are used as the categories in the one-hot representation. In the end, both categorical and continuous attributes are treated as categorical attributes.

2.1.3 Anomaly detection. In the theoretical study of machine learning, anomaly detection can be formulated as a problem of one-class learning, where only the benign data are labeled. Any testing samples deviating significantly from the benign training samples are considered as anomalies [9]. One-class learning is a branch of the family of semi-supervised learning techniques, which try to recover the classification boundary despite the lack of fully tagged training data. In our work, we follow the spirit of one-class learning methods. We train the AE-based detection model using only benign traffic. The model is supposed to cover the profile of benign traffics as much as possible. On the contrary, we do not use any attack data tagged by human experts or signature-based IDS for training. From the perspective of security practices, this is an unsupervised learning scenario, as no supervision information regarding the attack events (labeled attack events) is provided. We hence attribute our method to unsupervised detection hereafter.

For this one-class classification problem, Sec2graph uses an Auto-Encoder to learn how to accurately reconstruct the normal network traffic. Then, during the detection phase, the Auto-Encoder produces a reconstruction error of an input network traffic record. The reconstruction error is the binary cross-entropy between the input and the output of the Auto-Encoder. A lower (resp. higher) reconstruction error indicates that the corresponding input is less (resp. more) likely to represent an anomaly.

In the detection phase, the reconstruction error for each edge of the security objects graph is computed. The average reconstruction error for all the edges relative to a single network connection is then compared to a predefined threshold. Any network connection with an anomaly score above the threshold is considered as an anomaly.

2.2 CICIDS2017 dataset

CICIDS2017 [23] is a popular dataset for evaluating network intrusion detection systems. This dataset includes the network traffic of an IT infrastructure of a dozen of machines. The traffic was recorded over five days, including one day, the first one, without attack and four days with attacks. The first day is typically used for learning by one-class detection models, which only use the benign class to learn the normal behavior. Although this dataset is generally considered to have a good quality, recently, some studies [15, 17] found several mistakes, including labelling issue, in the dataset. Thus, we use the fixed version of the labels proposed by these articles. Different types of network attacks and the number of network connections belonging to each attack type are shown in the Table 1.

Attack types	Counts	Attack types	Counts
benign	1614317	heartbleed	1
botnet	736	infiltration	7
ddos	95146	infiltration - portscan	60867
dos goldeneye	7650	portscan	159043
dos hulk	162113	ssh-patator	2960
dos slowhttptest	1780	web - brute force	73
dos slowloris	2232	web - sql injection	13
ftp-patator	3972	web - xss	18

Table 1: Number of network connections per attack type and for the benign traffic in the CICIDS2017 dataset.

3 RELATED WORK

According to the taxonomy presented in [1], we can categorize XAI approaches into several main branches.

Intrinsically explainable models.

These types of XAI methods encompass simple and interpretable ML models such as linear or logistic regressions, decision trees, naive Bayes, and K-Nearest Neighbours. These models possess straightforward structures that can identify important features that trigger the decision of a given input. For instance, decision trees can rank feature importance by assessing the information gain obtained by selecting a specific feature in the decision chain. Linear and logistic regression models can evaluate the impact of different features on the decision output by utilizing sparsity-inducing constraints such as L1 regularization in Lasso/Elastic net regression methods and using the magnitudes of the coefficients. However, these models may underfit complex associations in the training data, leading to a loss of utility due to their simple model architecture. In our case, we propose producing post-hoc explanations directly from the output of the AE model. By doing so, the AE-based detection model can capture the underlying nonlinear relations between input attributes, which maintains utility for anomaly detection. Moreover, we demonstrate that the output from the AE-based reconstruction provides sufficient information to produce indicative explanations of the contribution of each network traffic attribute in the detection task.

Model-agnostic explanation methods. These methods provide post-hoc explanations regardless of the architecture of the target ML models. Global and local surrogate function based techniques [21] learn to approximate the global classification boundary of the target ML model or local classification boundary around the testing input. The surrogate function is intrinsically explainable,

e.g., a linear or decision tree-like classifier. By fitting the surrogate function to the true classification boundary, we can estimate the features' contribution to the classification output. A widely applied method in this branch is LIME [21], which can be categorized as a local surrogate function based method. Using a linear model, it approximates the target model locally around a given testing input. The magnitudes of the linear coefficients are considered as the base for the explanation. Similarly, SHAP [18] leverages the Shapley values to compute each input feature's contribution and then provide explanations.

There are also other XAI methods beyond the two major categories. For example, counterfactual analysis [11] and the influence function-based method [14] are developed to find influential data points close to the classification boundary of the target ML model. These influential data points are used to understand the importance of different features in triggering a classification output. However, most of the off-the-shelf XAI methods [8] are adopted for supervised learning tasks. They require fully labeled training data to approximate the true classification boundary and measure the usefulness of different feature dimensions. For this reason, they are not compatible with the scenario of unsupervised anomaly detection in our work, where only benign training samples are provided.

It is worth noting that there are a few XAI methods coping with the anomaly detection task [2, 4, 13, 19, 24]. [4] proposed ROULETTE to reformulate anomaly detection as a binary classification problem differentiating benign and outlier data. Then it adopts a self-attention module in the classification model to provide explanations. [13, 19] used the gradient of an Auto-Encoder-based anomaly detection model to deliver axiomatic feature importance evaluation. However, these methods consider continuous attributes as inputs to the detection model. In our study, the network traffic attributes are mostly categorical features, such as the user agent string or TCP flags. It is thus infeasible to use gradients as the indicator to feature-wise contribution to the detection output. In contrast, our proposed explanation methods is developed to adapt to unsupervised anomaly detection and handle both categorical and numerical inputs.

4 THE ALGORITHM DESIGN OF AE-PVALUES

4.1 Notations

Let b denote one categorical network traffic feature in our study. To facilitate the training of the model, we transform b into a one-hot encoded vector. The categorical feature b is unfolded into a k-dimensional binary vector $\{b_j\}$ $(j = 1, 2, 3, ..., k)$, where b has k possible category values. For example, the feature *protocol* has three possible category values *tcp*, *udp*, and *icmp*. Therefore the protocol feature is encoded into a 3-bit binary vector, one bit per category value. The bit is valued as 1 if the protocol feature carries the corresponding category value. In the end, the one-hot encoded vectors of each network traffic feature are concatenated into a high-dimensional feature vector x as input to the Sec2Graph-based detection model. Each x_i denotes the binary status of one category value of a network traffic feature. The corresponding reconstruction output from the Sec2Graph model is denoted correspondingly as $\hat{x}$.

4.2 Explanation Algorithm Design

Following the design of Sec2Graph, we can derive the dimension-wise reconstruction error regarding each feature dimension x_i. As we observe, the empirical distributions of the reconstruction errors per feature dimension x_i vary significantly. There exist feature dimensions that exhibit significantly higher variance of reconstruction errors, even in benign network traffic. Such feature dimensions are inherently more challenging to distinguish from truly abnormal feature values based solely on the absolute values of dimension-wise reconstruction errors. Specifically, in cases where feature dimensions exhibit large variances of reconstruction error, the magnitude of the dimension-wise reconstruction error may not necessarily imply the presence of abnormal activities.

As an echo, we consider using the p-value of the empirical distribution of the dimension-wise reconstruction error to flag abnormal feature values. Let H_0 and H_1 be the null and alternative hypotheses. H_0 and H_1 assume the reconstruction error of x_i is within the normal range or not respectively. We use r_i to denote the reconstruction error variable corresponding to x_i in theory. e_i denotes the empirically observed reconstruction error for x_i given an input x to Sec2Graph. e_i is given by the difference between the reconstructed variable $\hat{x}_i$ and x_i its input value. The p-value is then defined as the probability $p_i = \mathbb{P}(r_i > e_i)$. In our study, we take the empirical distribution $\hat{\mathbb{P}}$ of the reconstruction error of x_i as the reference distribution in H_0. We then compare the observed e_i to $\hat{\mathbb{P}}$ to compute the p-value of e_i with respect to $\hat{\mathbb{P}}$. A smaller p-value means that the H_0 hypothesis is more likely to be rejected, i.e. x_i has a large reconstruction error with respect to $\hat{\mathbb{P}}$ and vice versa. Given the feature dimension-wise p-value, we can rank the dimensions based on their probability of respecting the null hypothesis and obtain the explanation list.

Using the p-value to flag abnormal feature values alleviates the bottleneck of the L1-distance-based criterion for tagging anomalies. Computing p-values takes the variance of the per-dimension empirical reconstruction error $\hat{\mathbb{P}}$ into consideration. Therefore, whether or not the variance of $\hat{\mathbb{P}}$ is large does not change the derived p-value. In practice, the dimension-wise $\hat{\mathbb{P}}$ is a discrete empirical distribution, as the values of reconstruction error observations are discrete. As a result, we adopt the following computation in Eq 1 to obtain the p-value and produce the explanation results, which gives:

$$p_i = \frac{\#\{r_i \geq e_i\}}{\#\{r_i\}} \tag{1}$$

Improving the sensitivity of explanation. For the feature dimensions contributing extremely large reconstruction error, the derived p-value can become arbitrarily small, approaching to 0. In that case, the p-value is not differentiable enough to compare the usefulness of the features, i.e., these feature dimensions have equally small vanishing p-values. To increase the sensitivity of our explanation, we complement the proposed *AE-pvalues* method in this extreme situation by inspecting the difference between the empirical quantiles of the dimension-wise reconstruction error, which gives as below:

$$\alpha_i = \frac{e_i - m_i}{p_i^{99} - p_i^{1}} \tag{2}$$

where m_i is the median value (the 50-th percentile) of the empirical reconstruction error. p_i^{99} and p_i^{1} are the 99-th and 1-st percentiles

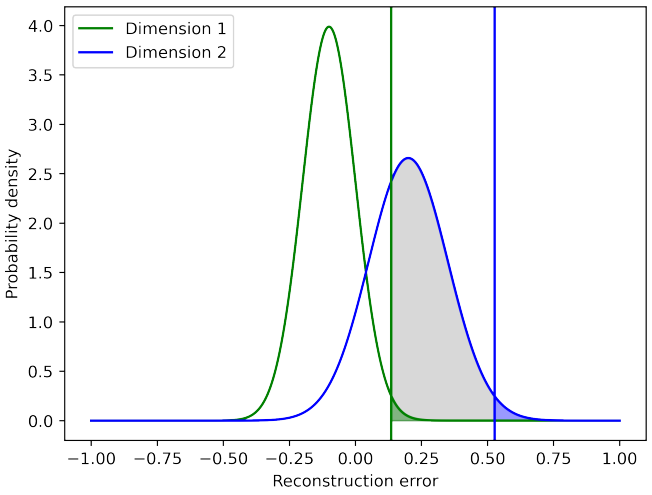

Figure 2: The reconstruction error distributions of two feature dimensions.

of the empirical reconstruction error. According to Eq 2, we take the difference between the 1-st and 99-th percentiles of the empirical reconstruction error of x_i. This difference is considered as the estimate of the variance range of the reconstruction error. After that, we perform the normalization of e_i by centering it with m_i and scaling the error by dividing the difference between percentiles to further remove the influence of the variance. The derived normalized reconstruction error value, noted as α_i, is particularly used to evaluate the anomaly level of the corresponding feature x_i in the case when e_i is extremely large, and p_i is equal to 0. More generally, the α_i are useful for ranking the feature dimensions when several share the same p-values score.

We raise a toy example in Figure 2 to demonstrate the benefit of using p-value instead of L1 distance. We create two toy variables following Gaussian distributions. The mean values and variances of the two Gaussian distributions are -0.1 and 0.1, and 0.2 and 0.15 respectively. These two variables correspond to the distributions of reconstruction error of the two toy features (feature dimensions 1 and 2, respectively, in the figure). The variance of the reconstruction error of feature dimension 1 is less dispersed than that of feature dimension 2. The two vertical lines denote the p-value-determined threshold to trigger anomaly alerts for the two features. Without loss of generality, we set $p = 0.25$ in this example. The p-values are represented by the green and the blue areas in the Figure 2. As seen, no unique L1 distance-based threshold value can be applied to both features 1 and 2. For example, the threshold used for feature dimension 1 may produce a high false positive rate for feature dimension 2, represented by the grey area on the Figure 2. Instead, we can use a fixed p-value in this example to adaptively set the threshold to detect anomalies for both features, regardless of the reconstruction error variance.

Besides, we involve two explanation baselines to organize the comparative study. The first method noted as *AE-abs*, is a variant of the proposed *AE-pvalues*. It ranks feature dimensions x_i by the L1 distance-based reconstruction error per x_i, namely by the $|\hat{x}_i - x_i|$ scores. We introduce *AE-abs* to show the benefit of using p-values instead of the magnitude of the reconstruction error, given the varying variance of the per-dimension reconstruction error.

The second method adopts SHAP for Auto-Encoders [5], noted as *SHAP_AE*. The core idea of *SHAP_AE* is to use Shapley values to measure the contribution of each input dimension x_i to the highest dimension-wise L2 distance-based reconstruction errors among Sec2Graph's output. We then rank the Shapley values of each x_i to prioritize the feature dimensions contributing the most to the reconstruction error.

5 EXPERIMENTAL EVALUATIONS

We first explain the empirical protocol used to set up the comparison between different explanation techniques in Section 5.1. Then, we present the results of the comparative evaluation on the CICIDS2017 intrusion detection dataset in Section 5.2.

5.1 Experimental protocol

5.1.1 Problem statement. The aim of the explaining methods is to identify correctly where the anomaly is located in the vectors. Each **vector** represents an edge in the graph as reminded in the Subsection 2.1.2. The vector is a representation of all the **network traffic features** of the nodes that are associated with the edge. These network traffic features are, for instance, the protocol, the HTTP method, and the browser. A comprehensive list of them is available in the appendix A which is the Sec2graph model description. Each of these network traffic features has several **category values**. For example, (icmp, udp and tcp) are the category values for the network feature protocol, (GET, POST, PUT, DELETE, other) would be those of HTTP method. In the rest of the article, we named **feature dimensions** the whole list of the category values.

5.1.2 Protocol. **Evaluating the accuracy of the explanation results:**

Our approach involves evaluating different explanation methods by measuring their accuracy in identifying both the network traffic feature and the perturbed category values responsible for producing anomalies. We evaluate the identification at two levels, and our evaluation methodology is based on the widely used one-factor-at-a-time sensitivity analysis method [7]. To ensure that no anomalies are present, we use vectors from days without attacks. For each network traffic feature such as *protocol*, *conn_state* or *file-transfer_depth* and the others , we select 100 feature vectors without attacks and introduce artificial perturbations to the chosen feature of each vector. In each of the 100 vectors we create a perturbation in the chosen network feature by changing the position of the selected feature dimension as shown in Figure 3. To perturb categorical network attributes, such as the public/private status of an IP address, we replace the original category value with a randomly selected value. For binary features, we simply flip the original values. Next, we evaluate whether the explaining methods can identify the perturbed category values in each feature vector where artificial noise is injected. Each explanation method ranks the feature dimensions corresponding to all category values according to their anomaly scores and identifies the top-K feature dimensions as the most likely to present abnormal values. We refer to this list as the top-K list.

As explained in Section 2.1.1, different kind of vectors exists in the graph because of the heterogeneous nature of the edges in the

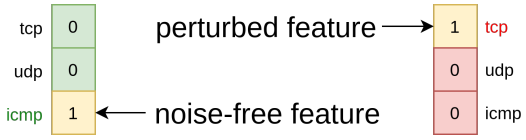

Figure 3: Example of noise insertion in a network feature (protocol). This network feature has three category values (tcp, udp, icmp), thus encoded to a 3-dimensional one-hot encoded representation. The one-hot dimension containing originally a 1 is named "noise-free feature", and the dimension containing, in the end, the 1 is called "perturbed feature"

Sec2graph representation. The use of certain features may vary depending on the type of vectors that are selected. To guarantee the effectiveness of the evaluation, we decided to only inject noise into vectors for which the feature-to-perturb was already used. This prevents the introduction of artificial embeddings that do not exist in the data because all the network traffic features are not used for all of the edge types. These kinds of anomalies are too easy to detect and unrealistic. The explanations are edge-specific. Indeed, each vector corresponds to an edge in the graph and has a type. We only keep the explanations that are meaningful with the edge type. For instance, when assessing the explanations regarding a vector that corresponds to an edge that is linking an HTTP object and a NetworkConnection object, we do not take into account the feature dimensions that are associated with DNS, SSH, or any other non-related objects. Thus all the dimensions non-related to the two objects of the edge are discarded.

In the comparative study, we involve *AE-pvalues* and *AE-Abs* explaining methods. A *Random* strategy is also compared with the other methods and consists in ranking the feature dimensions randomly. In addition to these methods, we also include *SHAP_AE* [5] in the comparative study. *SHAP_AE* uses the model-agnostic approximation of SHAP values, a.k.a. Kernel SHAP [21]. It needs a background set to approximate the model locally with another linear model. This background set is made up with 200 vectors from the training set of the Auto-Encoder used in Sec2graph. Similarly, *AE-pvalues* also needs a background set to estimate the normal distributions of the reconstruction errors for each feature dimension. We provided the whole train set of the Auto-Encoder to have as much diversity as possible.

To set up the ground truth to measure the accuracy of the explanation results, we include the feature dimensions manually perturbed in the ground truth of the explanation evaluation. Beyond that, we also include the other feature dimensions statistically significantly correlated to the manually perturbed ones (with a significance level of less than 0.05). For example, if the perturbation is injected into the *http_code_200* category value and the first returned explanation is the category value associated with the *http_msg_OK* attribute, which is confirmed to have the same physical significance with *http_code_200*. Therefore, if the feature dimension manually perturbed or any other feature dimensions highly correlated with the perturbed ones exist in the top-K list, the explanation results are considered to be effective. In the rest of the paper, when the "_corr" suffix is added to any method name, it denotes that we match not only the perturbed dimensions but also the significantly correlated ones to the top-K features ranked by the explanation output. We define two metrics by finding the matches feature dimensions in the top-K list.

The mean rank (noted as Rank): we first compute the ranking index of the matched features in the top-K list produced by each explanation method. We then compute the mean rank of the matched features among all the vectors selected in the test. A lower value of the rank denotes a more accurate identification of the features containing anomalies.

The top-K accuracy (Accuracy – K): we compute the fraction of the matched features in the top-K list. The higher the accuracy score is obtained, the more accurate the explanation method is in identifying features containing anomalies.

5.2 Experimental results

Results of the mean rank. In Figure 4, we present the average rank of both the noise-free feature and the perturbed feature, along with the rank of the network feature itself. In the example of the Figure 3, *icmp* is the noise-free feature, *tcp* is the perturbed feature, and *protocol* is the network feature. The Table 2 lists the mean ranks provided by different explaining methods. All the involved explanation methods are evaluated with and without taking into account the correlated network traffic features. As shown, *AE-pvalues* provides consistently lower mean rank values compared to the other methods, no matter whether the correlated features are taken into consideration. The results indicate that *AE-pvalues* can better prioritise the features that contain anomalies in the explanation results than the other opponents.

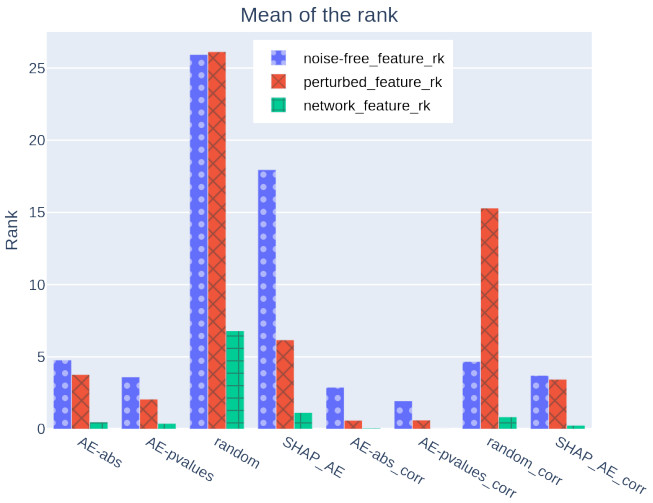

Figure 4: Mean rank comparison between the different explaining methods.

Results of top-K accuracy. Figure 5 illustrates the top-K accuracy obtained by different explanation methods. We conduct the analysis in two steps. We first compare the explanation methods by taking into account the correlations. Then we revisit the comparison between different methods without taking into account the correlations. In the first case, where correlations are taken into

explaining method	Mean rank of the noise-free feature dimension	Mean rank of the perturbed feature dimension	Mean rank of the network feature
AE-pvalues_corr	**1.96**	**0.63**	**0.02**
AE-abs_corr	2.89	0.61	0.07
SHAP_AE_corr	3.71	3.44	0.26
AE-pvalues	**3.61**	**2.07**	**0.39**
AE-abs	4.78	3.78	0.49
Random_corr	4.68	15.3	0.85
SHAP_AE	17.96	6.18	1.15
Random	25.93	26.13	6.8

Table 2: Table of mean ranks of the noise-free feature, the perturbed feature, and the network feature where the noise is inserted. See Figure 3 for more precise column name definitions.

account, we can find that *AE-pvalues* obtains the highest accuracy. In contrast to the other two methods, *SHAP_AE* (evaluated with or without considering the correlations) is always less accurate than the others except for $K = 1$. Even increasing the number of explanations, *SHAP_AE* still have more difficulties to identify the relevant feature than the other approaches. When comparing the different methods without taking into account the correlations, we can observe that *AE-abs* and *SHAP_AE* perform better for the top-1 accuracy. *AE-pvalues* certainly provides a correlated attribute in the first position more often than the two other methods. However, when we take the top-K with $K > 2$, *AE-pvalues* becomes the most accurate method. That is we are more likely to obtain the abnormal feature in the provided top-K explanations. As we can find, *SHAP_AE* delivers less accurate explanations compared to *AE-pvalues* and *AE-abs*. We further demonstrate the superior accuracy of the explanation results produced by *AE-pvalues* in Figure 6. We first aggregate the variance of reconstruction error per feature dimension to obtain the averaged variance of reconstruction error for each network traffic feature. We then split all the network traffic features into two separate subsets. One set contains the features with an averaged variance larger than 10^{-6} (noted as the large variance subset), and the other set contains those with an averaged variance less than 10^{-6} (noted as the small variance subset). Figure 6 illustrates the top-K accuracy of all the involved explanation methods over the larger and small variance feature subsets. As shown in the right side plot of Figure 6, the proposed *AE-pvalues* gives significantly higher top-K accuracy than *AE-abs* and *SHAP_AE*, especially with K less than 4. By comparison, the top-K accuracy of *AE-pvalues*, *AE-abs* and *SHAP_AE* are close when the average variance of network traffic features is small according to the left side plot. The observation echos the motivation of using p-values instead of *AE-abs* and *SHAP_AE*. The limit of *AE-abs* and *SHAP_AE* is rooted in the use of L1 distance based reconstruction error as the measurement to evaluate feature informativeness. The L1 distance only indicates the magnitude of the reconstruction error. It does not consider the baseline variance of the reconstruction error under the noise-free scenario. In case the variance of the reconstruction error is large in the noise-free case, a large magnitude of L1 distance-based reconstruction error does not necessarily indicate the existence of anomalies. By incorporating p-value selection, the explanation output of *AE-pvalues* is more robust to variations in the reconstruction error's variance. Consistently, the average ranks of the good explaining features and category values are better using *AE-pvalues* method as shown in the Table 2.

5.3 Performance considerations

Beyond the relevance of the explanations brought by the different explaining methods, we also measured the scalability of the different methods. In the Table 3, we present the time to process the explanations for a single sample. The measurements were done on 100 vectors (samples) to explain, and the values represent the average processing time. Given the table, on the one hand, we clearly see that *SHAP_AE* does not scale well since the duration is far too long to process many vectors. On the other hand, *AE-pvalues* method is much faster and thus can be used to process many vectors. It is important to mention that *AE-pvalues* method takes approximately 150 additional seconds as a pre-processing step to compute the distributions associated with the "normal" behaviors. This duration only depends on the size of one vector. Here the size of each vector is 400 dimensions. It is only computed once for all and does not depend on the number of samples to explain. Moreover, the implementation of *AE-pvalues* can be easily parallelized and thus provide even much more efficient performances for many samples to explain.

Method	Processing time per sample
SHAP_AE	28 s
AE-pvalues	1.9 ms
AE-abs	1.0 ms

Table 3: Processing time for one sample for each explaining method

6 EXPLANATION IN THE CONTEXT OF ANOMALY DETECTION

Through Section 5, we analyzed different explaining methods. From the benchmark, we know *AE-pvalues* is the most accurate explaining method. Moreover, *AE-pvalues* is scalable. For all these reasons, the explanations obtained hereafter will be only based on *AE-pvalues*. In the following, we demonstrate how human analysts can benefit from these explanations to analyze alerts raised by IDS.

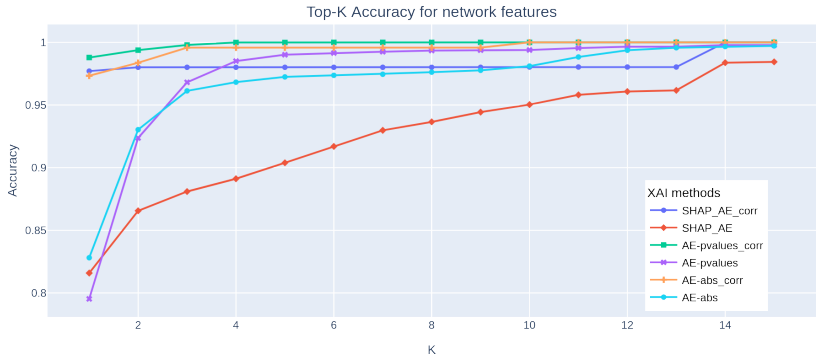

Figure 5: Comparison of the top-K accuracy of the explanations of the different XAI methods

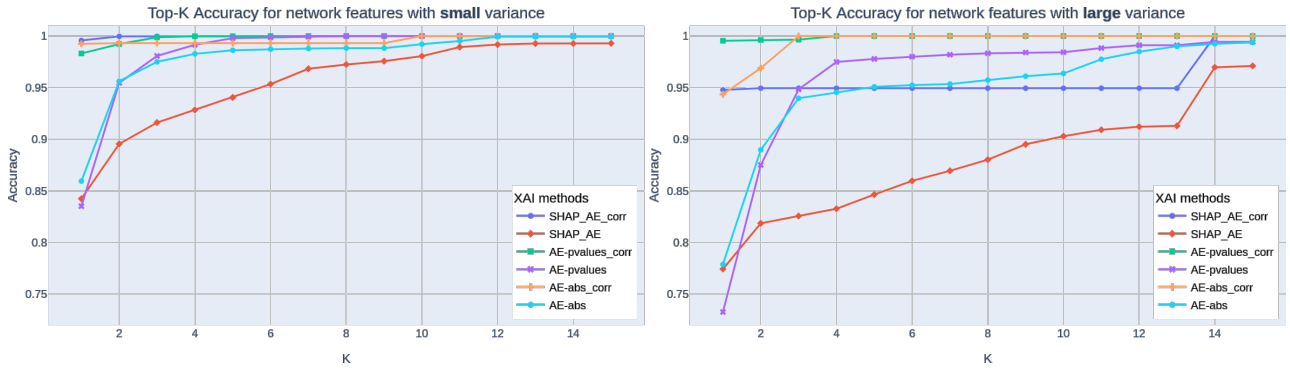

Figure 6: Comparison of the top-K accuracy of the explanations of the different XAI methods. The left Figure is for small variance network features, and the right Figure is for the high variance ones.

In Section 5, we focused on the explanations for a single vector that represents a single edge in the graph, see Section 2.1.2 for details. In practice, the Sec2graph method represents a single network connection with several vectors (several edges). We hence define how to combine explanations produced from different vectors. The goal is to achieve the categorization of different attack types in the network connections.

Using the anomaly scores aggregated per network connection either of each category value or of each network feature, we show that we can perform clustering on them and group by attack type the different network connections that are identified as positives by the detector.

6.1 Clustering explanation outputs to identify network attack types

We perform clustering over the explanation outputs from *AE-pvalues*. We aim to unveil different attack types in the detected anomalies. The clustering results can help demonstrate the expressiveness of the explanations to capture differences among attack events. Thus, when a new security alert is generated, the security operator can obtain insights into the type of attack the alert belongs to by leveraging the previously labeled alerts and associated clusters. To this end, we constructed vectors by extracting the explanations and their corresponding anomaly scores from a network connection, as detailed in Section 6.2. These vectors have dimensions equal to the number of possible features, and their values represent the anomaly scores of each feature. We experiment with both the category value scores and the network feature scores. Each vector thus captures the abnormality scores of either the category values or the network features for a given network connection. To ensure sufficient representation of each attack type, we sampled 1000 network connections of each type. We used all the available attack types existing in the CICIDS2017. As presented in Table 1, some types of attacks, such as heartbleed, infiltration, and web attacks, are significantly more rarely witnessed than others. For the attack types with less than 1000 network connections, we used all the available network connections relative to this attack.

We employ an unsupervised hierarchical clustering algorithm to detect clusters among the vectors, where we specify the desired number of clusters to be formed. In this study, we aim to have 7 clusters corresponding to the major classes of attacks, including patator, Denial Of Service (DoS), heartbleed, portscan, web, infiltration, and botnet attacks. The algorithm then attempts to group together vectors that share similarities in the explanations of each network connection. The method constructs a hierarchical tree of clustering outcomes, starting from a single cluster encompassing all the vectors and then repeatedly partitioning it until the desired number of clusters is obtained. The complexity of the hierarchical algorithm is $O(n^3)$, where n is the number of network connections

involved in the clustering. The clustering analysis was performed on a Linux machine with an Intel(R) Core(TM) i7 CPU processor (2.70GHz, 12 cores), and 32Gb of RAM. The execution took 17 seconds to cluster 2×9848 data points with respective dimensions $(389, 109)$ for (feature dimensions, network features) .

We proceed to calculate the Normalized Mutual Information (NMI) between the predicted clusters and the actual labels. In the case of a random strategy, the NMI obtained is 0.0011. However, when using the clustering method based on the anomaly scores of network features obtained from *AE-pvalues*, the NMI value of the clustering results significantly increases to 0.64. This is much better than the random guess baseline, as shown in Table 4.

The attack types captured by each of the seven clusters are illustrated in Figure 7. It can be observed that the SSH-patator attack and port scans are effectively captured in a single cluster, namely clusters 4 and 5, respectively. The network connections associated with the SlowHttpTest attack, which belong to cluster 5 are rejected network connection attempts that behave similarly to port scans. This explains why they are grouped together in the same cluster. However, for DoS attacks, they are divided among clusters 0, 2, and 6. The ftp-patator and botnet attacks are contained in cluster 1. Detailed information regarding the composition of each cluster is available in the appendix B.

	AE-pvalues NMI scores	**Random**
feature dimensions	**0.638**	0.0011
network features	0.554	

Table 4: Comparison of the Normalized Mutual Information scores using *AE-pvalues* to obtain explaining scores per feature dimension and per network feature and a *Random* strategy.

The results demonstrate the feasibility of categorizing network connections based on their associated attack types. The methods used to calculate the anomaly scores at the network connection level will be elaborated in Section 6.2.

6.2 Aggregating explanations of each network connection

We utilize the p-values obtained from the explanation list of each network feature vector to rank the feature dimensions and network features. The p-values indicate the abnormality of each feature dimension and their ranking order. Our proposed ranking scheme takes into account the global reconstruction score of each vector j, which is computed using BinaryCrossEntropy (BCE) as bce_j. We assign a higher weight to vectors with higher reconstruction error (higher bce). The p-value p_i of the i-th dimension indicates its abnormality; smaller p-values indicate a higher abnormality. As the variation of the bce_j score is the opposite of the variation of the p-value p_i, we use $(1 - p_{ij})$ which we call β scores to make the two quantities evolve in the same way. The score of the i-th dimension is the average of all the β scores, weighted by their respective bce, for all vectors related to the network connection. The formula for computing the score of the i-th dimension is given by Eq 3.

$$score\_dimension_i = \frac{1}{nb\_edges_i} \sum_{j=1}^{nb\_edges_i} bce'_j \times (1 - p_{ij}) \quad (3)$$

We also aim to aggregate the explanations at the network feature level and propose Eq 4 to achieve this. The basic principle is similar to the per feature dimension case described in Eq 3. However, since we have multiple category values for a single network feature, we needed to devise a strategy to combine the abnormality scores of the different category values into a single score. We decided to use the highest abnormality score among all the category values of the network feature. Therefore, we use the max function over the β scores for each category value that belongs to the network feature.

$$score\_feature_i = \frac{1}{nb\_edges_i} \sum_{j=1}^{nb\_edges_i} bce'_j \times \max_{l \in dim\_feat_i} (1 - p_{lj}) \quad (4)$$

In both Eq 3 and Eq 4, nb_edges_i is the number of edges that are compatible with the i-th network feature or category value and bce'_j is the bce_j score normalized as expressed in Eq 5.

$$bce'_j = \frac{bce_j}{\sum_{k=1}^{nb\_edges} bce_k} \quad (5)$$

6.3 Explaining alerts raised by a ML-based NIDS

To figure out if the explanations were relevant for each attack type, we took all the True Positive (TP) samples obtained with the Sec2graph approach. Like in the clustering section, we analyzed the explanations of 1000 network connections of each attack type. These network connections were obtained using the following sampling scheme: we took the first 300 network connections and the last 300 network connections, and then we sampled 400 network connections uniformly in the rest of the available network connections. We chose this approach to get connections related to the beginning, the middle, and the end of each attack. Indeed, the behavior of the attack may change over time, as shown afterward. We include a manual inspection of several attacks in CICIDS2017 and verify whether the explanations produced by *AE-pvalues* are consistent with the unveiled attack mechanisms in the network traffic data. In addition, we also performed the manual investigation for some False Positive (FP) samples.

In practice, a security operator is likely to be able to handle between three and five explanations per network connection. If we provide more than five explanations, the operator might be unable to analyze them thoroughly. If we provide less than three explanations, the explanations might not be discriminating enough to identify the attack type. In this section we decided to always work with the top-5 feature dimensions or network features.

The Figure 8 proposes a general overview by summarizing the features contributing by at least 10% in an attack's explanation. To compute these values, we collected the first five explaining network features for each network connection and measured the appearance frequency of these features for each attack type. The darker the

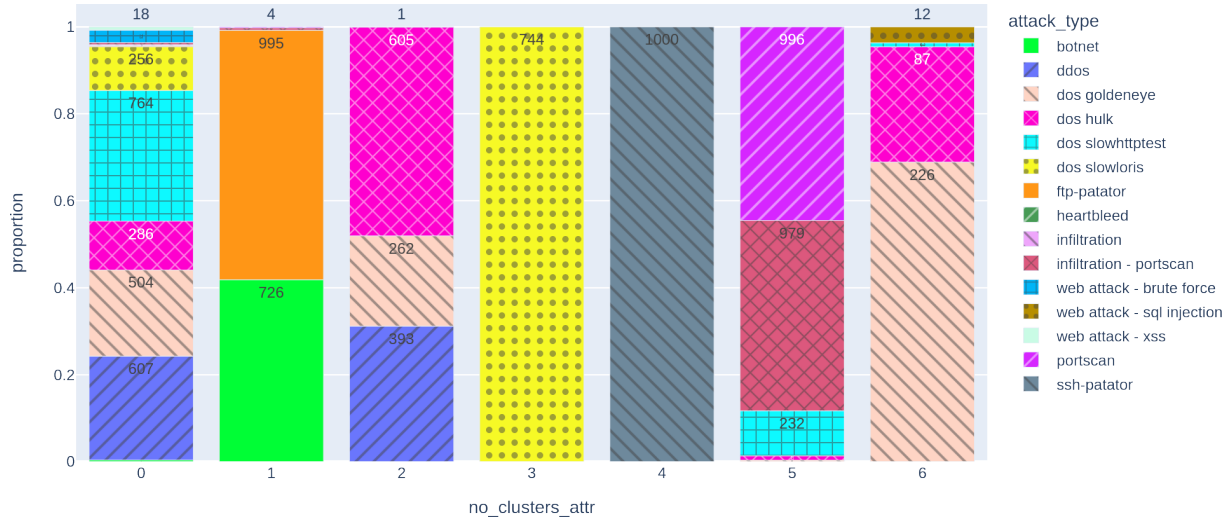

Figure 7: Proportion of each attack type in the different clusters

color, the higher frequency has the feature for the corresponding attack. The maximum score per network feature is 20%, which means the network feature always occurs among the top-5 explanations. We can observe some common patterns based on the different attack types. For instance, web attacks (highlighted in green) are characterized by the user agent (browser and operating system). Several attacks are characterized by the *port_value*, which is involved in attacks such as the port scan and botnet attacks. SSH-related features are specific to the ssh-patator attack. From this high-level perspective, the explanations seem reasonable and consistent with the class of attack. In this section, we present a manual analysis of the DoS Slowloris, the brute force belonging to the web attacks, the SSH-patator, the botnet attack, and the DoS GoldenEye.

In the following, we use both network features and feature dimension explanations. Each feature dimension can be described as a tuple (feature, category value). For clarity, we write such a tuple as *feature/category*. For example, the explanation *ua_browser/Firefox* means that the value *Firefox* of the feature *ua_browser* is important to explain the abnormality of the network connection. As seen in Section 2.1.2, continuous features are categorized using GMM. Thus the name of these features ends with *_gmm*, and their associated category value is the number of the Gaussian in which the value falls. Again for clarity's sake, some very long category values are shortened with "[...]".

DoS Slowloris With this attack, the attacker seeks to consume all the processes or threads allocated by the web server to process queries. They do so by maintaining the TCP connections open for a very long time. To maintain these connections open, the tool does not end the HTTP header part of the request and regularly sends (before the server timeout) a new HTTP header. The most frequent explanations provided at the category values level are: *http_status_code/0*, *http_info_code/0*, *http_status_msg/other*, *http_info_msg/0*, and *http_status_code/other*. They represent 55% of the samples, i.e., of 1000 network connections of this attack type. These explanations describe the situation when the server is down. Thus, it is not responding anymore. Therefore we can observe the missing status code and message since the server never replied before the network connection was stopped.

Then 40% of the network connections were explained by the following feature dimensions *ua_browser/Firefox*, *ua_os/Mac OS X*, *ua_browser/other*, *filetransfer_mime_type/image/gif*, and *filetransfer_mime_type/application/xml*. These explanations are related to the abnormal user-agent of the attacker. Indeed, the user-agent seen during this attack has never been used on the training day (the day without attacks). The Auto-Encoder detects this anomaly, and the explanation method correctly highlights the user-agent (containing the browser and the operating system).

Brute Force (web) With this attack, the attacker makes many authentication attempts on a web page with a machine under the Kali Linux operating system. Thus the attacker has a very discriminating user agent, and the detection model uses it to detect this attack. We find it again in the explanations. 96% of the 73 network connections associated with this attack are explained at the feature dimensions level by the following attributes: *ua_browser/other*, *ua_browser/Firefox*, *ua_os/Mac OS X*, *ua_browser/Safari* and *http_status_msg/Found*. The HTTP message "Found" is consistent with the fact that the attacker requests the `login.php` page to perform the brute force, and for each query, the server returns the code 302 and the message "Found". The explanations at the network feature level are *ua_browser*, *ua_os*, *http_status_msg*, *http_status_code*, *http_trans_depth*, are also valuable for the expert since they relate the fact that there are a large number of HTTP requests within the same network connection as shown in the *http_trans_depth* category. At the network feature level, we find the user-agent and the HTTP status code related features again.

SSH-Patator The attack consists in brute forcing an SSH server to try to obtain access to the server. The attacker uses a tool named Patator, developed in Python. That tool uses the paramiko library to establish the ssh connection and attempts some login password couples. Zeek logs four authentication attempts for each network connection. None of the attempts are successful.

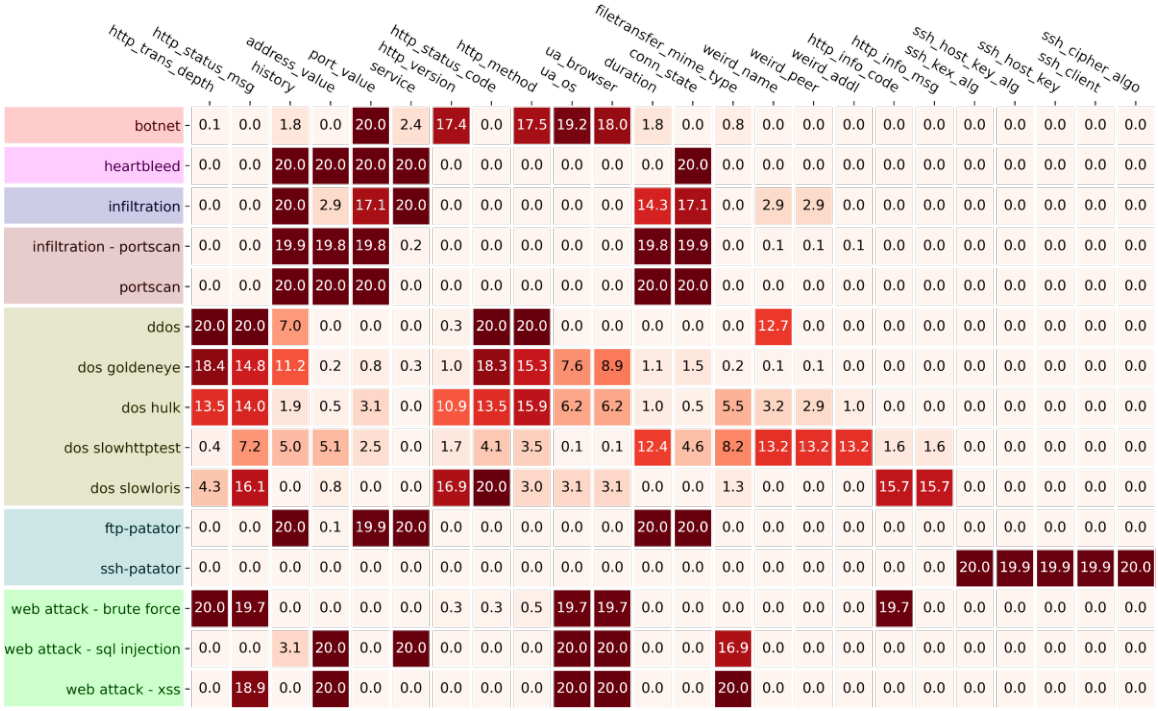

Figure 8: Visualisation of the explanations per attack type. The scores are the percentages indicating how often the features occur in the top-5 explanations for the respective attack type.

99.6% of the alerts are explained by two groups of explanations. 77.5% of the alerts linked to SSH-Patator are explained by the following feature dimensions: *ssh_host_key/b5:61:[…], ssh_mac_alg/hmac-sha1, ssh_server/SSH-2.0-OpenSSH[…], ssh_compression_alg/none, ssh_host_key_alg/ssh-rsa*. Their corresponding network features are: *ssh_host_key, ssh_mac_alg, ssh_server, ssh_compression_alg, ssh_host_key_alg*. Then the rest (22.1%) of the alerts are explained by the following feature dimensions: *ssh_host_key/b5:61:[…], ssh_mac_alg/hmac-sha1, ssh_server/SSH-2.0-OpenSSH[…], ssh_host_key_alg/ssh-rsa* and *ssh_compression_alg/none*, and the associated network features explanations are *ssh_host_key, ssh_mac_alg, ssh_server, ssh_host_key_alg, and ssh_compression_alg*.

In fact, on the training day, only one SSH server is accessed, and the clients used to access the SSH server are all the same. It implies that all the attributes linked to the SSH security object are correlated. As the attacker uses the Patator tool, which uses the paramiko library, only two network features are different in the SSH security objects related to the attack: the *ssh_client* (linked to paramiko) and the *ssh_mac_alg* (certainly depending on the use of the paramiko library). The explanations support the fact that, for the detection model, the network features of the SSH security object are inconsistent.

Botnet. The bots are certainly preinstalled on five Windows machines of the experimental setup of CICIDS2017. The bots installed are related to an open-source botnet developed in Python named Ares[2]. The bots communicate with the C2 server by HTTP on the port 8080. They send regular beacons to the C2 and sometimes receive commands from the server, such as taking a screenshot of the desktop or listing a directory.

86.5% of the alerts are explained by the following feature dimensions: *duration_gmm/2, history/w, conn_state/RSTO, conn_state/SF, proto/udp*. The network features linked to those alerts are: *duration, history, conn_state, proto, orig_ip_bytes*.

The HTTP connections related to these alerts are linked to the beaconing, and they last less than 10ms. That is due to the fact that the C2 server responds to those beacons only with HTTP headers (no HTML page is sent by the server) and to the geolocation of the C2 server which is in the same setup as the other machines (only one router is between the computers infected by the botnet and the C2 server). The explanations point to the fact that the duration value should be higher and that, for very short-lived connections, it is abnormal to have a full HTTP connection on tcp (conn_state/RSTO, conn_state/SF, proto/udp). So the explanations are corroborated by the actual analysis.

DoS GoldenEye. The GoldenEye tool[3] is a DoS tool that aims to take down a server by preventing the cache mechanism, maintaining connections with the keep-alive header (timeout option) and sending a large number of requests. In order to prevent easy filtering of the requests, all requests have a random argument with a random value; the referrer is also random, and the user-agent is taken randomly from a list of user agents. It must be noted that the Apache server attacked by that DoS does not care about the

[2]https://github.com/sweetsoftware/Ares

[3]A version of the tool can be found here: https://github.com/jseidl/GoldenEye

timeout value specified in the request and sets it to five seconds. After those five seconds, if the client sends no packets, the server closes the TCP connection.

For this attack, we observed two cases. For the first case, 25.9% of the alerts are explained by the following feature dimensions: *http_method/other, http_trans_depth_gmm/2, http_status_msg/other, http_status_code/304, http_trans_depth_gmm/1*. The network feature-level explanations related to those alerts are the following: *http_method, http_trans_depth, http_status_msg, http_status_code, history*.

Secondly, another case includes 16.5% of the alerts and the alerts are explained by the following feature dimensions: *http_status_msg/other, http_trans_depth_gmm/3, http_status_msg/Found, history/R, history/t*. The explanations at the network feature level related to those alerts are the following: *http_status_msg, http_trans_depth, history, conn_state, duration*.

The network feature *http_trans_depth* represents the number of HTTP requests sent in one TCP connection (thanks to the keep-alive mechanism). For all those alerts, only one request is sent, but the connection duration is around five seconds in the first case and around five minutes in the second case (due to a RST packet sent by the attacker five minutes after the FIN packet sent by the server). For those kinds of duration, more HTTP requests should have been sent in the TCP connection. The network feature *history* is also interesting because the network connections related to the attack are closed in an abnormal way (the attacker does not answer with a FIN/ACK packet to the FIN packet sent by the server and, in the second case, responds with a RST packet five minutes later). The explanations are, therefore, consistent with the expert analysis.

False Positives analysis. The Sec2Graph NIDS raised about 17,000 false positive alerts on the four days with attacks (containing almost 1,700,000 connections on the whole). Given the results of Table 4, the explanations for each dimension of every network feature produce a more fine-grained and informative understanding of the detection result. Therefore, we adopt this level of explanation to investigate the FP examples. The alerts can be grouped into 3,000 different explanation tuples. Each tuple corresponds to a particular case to be investigated. One such tuple is composed of the top 5 feature dimensions ranked by the proposed *AE-pvalues* method, like in the TP analysis, and they are considered to contribute the most to the corresponding alerts. We provide the analysis of three different FP alerts with the highest anomaly scores. For these samples, there is no ground truth regarding the explanations. We can only rely on the understanding of the network captures.

The following 5 feature dimensions are considered to explain the first FP example:*dcerpc_endpoint/other, dcerpc_operation/other, dcerpc_named_pipe/other, dcerpc_endpoint/lsarpc, dcerpc_named_pipe/135*. Given the explanations, this network connection was very likely to be related to using the DCE RPC protocol. In the corresponding network capture, we indeed observed the use of the DCE RPC protocol for the remote administration of the machine. We also note that, as suggested by the explanations, the endpoint, the named pipe, and some operations used are never seen in the benign training data of the AE. Moreover, this protocol is rarely used in the benign network traffics and represents only 0.05% of the network connections. The scarcity of the protocol and the unseen values may explain why this connection triggers the FP. A network administrator should know the mechanisms used for the administration of its network. In such a situation, he or she should know whether DCE RPC is legitimately used. Therefore from the provided explanations, he or she should be able to eliminate this FP alert quickly.

Another interesting FP sample among the most abnormal ones is a network connection with the following explanations: *port_value/other, port_value/80, port_value/443, history/S, conn_state/RSTR*. In this example, we can expect a problem related to the value of the port used. The history of the network connection seemed to be abnormal, as suggested by the SYN flag (*history/S*), and finally, the connection state flags also seemed to indicate an issue (RSTR: Responder sent a RST). When we looked at the network capture, the port value did not seem abnormal (the port value is 80). However, the two other explanations were very indicative. The network connection was observed at the beginning of Wednesday's capture. In fact, the beginning of the connection was missing, so it missed the SYN packet of the emitter at the beginning of the connection. At the end of this connection, the server ended the connection via the issuance of a RST packet. This connection corresponded to an update of a Windows client's malware signatures on Wednesday morning in the dataset. It was, therefore, not abnormal. But it was also reasonable to return an alert regarding an initiated connection without a complete TCP handshake and then aborted by the server. For this FP, even though the explanations well indicated how the low-level network event occurs, we can find that a manual inspection would be required to verify the true cause. The explanations give initial clues about the anomaly to the analyst and should accelerate the analysis.

For the third FP sample, the explanations include *ssh_version/other, ssh_server/other, ssh_kex_alg/other, ssh_cipher_algo/other, ssh_host_key/other*. We observe many ssh-related explanations. We can thus expect an issue with this ssh protocol. The network connection was initiated by a Windows client with the IP address 192.168.10.5 toward the Ubuntu web server with the IP address 192.168.10.50. The client did not launch any attack against the web server. However, in parallel to this connection, an SSH-Patator attack was ongoing on the web server. When we checked the fields of network features highlighted by the explanations, we observed that all of them except the *ssh_version* were left empty, which is never witnessed in benign traffics. These fields were empty because when the client initiated the connection, the server had no more available resources due to the attack. The server thus sent a FIN packet right after the TCP handshake. The client continued to issue data, but the server terminated the connection by issuing a RST packet. No information about the versions of cipher and key exchange algorithms was exchanged. This network connection was not malicious but had unusual behavior due to the ongoing attack. In the above paragraph on the SSH-Patator attack, we saw that the explanations on the true positive alerts allowed the analyst to understand that the paramiko tool was used, revealing an ongoing ssh-related attack. Similarly, the explanations of this alert also highlighted ssh-related issues, so the analyst could quickly understand that it was a consequence of the attack.

Discussion. The explanations are very precise at the network level, pointing to the feature dimensions and the network features

that differentiate the network connections of the attacks from normal network connections. However, they are generally too low-level to give an immediate insight to the analyst so that he can classify the alert without further analysis. The clustering of the alerts is a way to give this insight and ease the classification (see Section 6.1). However, since the clustering is unsupervised, clusters are not labeled and thus require analysts to have previously labeled some samples of the different clusters to infer which attack type it is.

The lack of high-level explanations is mainly due to four reasons: high correlations between some feature dimensions or some network features, many biases in the CICIDS2017 dataset, the fact that the detection model is only based on a single network connection and the fact that the explanations only point out a feature dimension or a network feature without telling whether the value should be present or absent. This last issue can be illustrated as follow. If an explanation is *history_R*, the analyst does not know if the detection model expected that the client sent an RST packet, but this event did not happen, or if it is the opposite and the explanation highlights the presence of such a history flag. We could easily overcome this issue by confronting the explanations at the feature dimension level with the actual category value existing in the vector. This information is more difficult to obtain at the network features level.

About the biases of the CICIDS2017 dataset, we discovered many of them by analyzing the explanations given by our method. Several works already tackled CICIDS2017 dataset quality and performed sometimes a manual analysis of the network captures such as [17], [22] or [15] however, they highlighted labeling issues, flow extraction issues, traffic capture issues, but none of them is highlighting the biases that exist in the dataset. To the best of our knowledge, the biases we identified are novel and were not previously reported. For example, we can see in Figure 8 that the alerts for the web attacks and the botnet attack are mainly explained by network features related to the user-agent. In fact, in the CICIDS2017 dataset, the user agents are the default ones specified in the tools used to implement those attacks. Script-kiddies would certainly keep those values, while a more advanced attacker would change them. Thanks to the explanations, we realized that for those attacks, the detection is more related to an artifact of the attacker's tool than the network behavior of the attack type. The experimental setup used to generate the dataset also affects network features such as the duration of connections (for the botnet attack, the C2 server is in the experimental setup while all the other web servers are on the Internet). In the false positive analysis, we even saw how cutting network captures can artificially make benign connections have abnormal behavior. These examples point out an issue shared by several security datasets: the evaluated detectors can learn to correctly identify attacks by relying on experimental artifacts, leading to a potential overestimation of the detection performances of these detectors on such datasets without being relevant in practice.

7 CONCLUSION

In this article, we proposed *AE-pvalues*, a new unsupervised method based on p-values to provide explanations that could be used on top of any AE-based anomaly detection method. We compared this new method to existing ones and showed that it performs better at finding the right feature responsible for the anomaly. It also obtained better performances when it comes to clustering the network connections by attack type based on the explanations. The method is scalable which was not the case for *SHAP_AE*. Besides, we used this p-value-based method to analyze practical attacks of the CICIDS2017 dataset and show its relevance on real data. On this occasion, we could observe the limitation of the dataset regarding the quality of the attacks. They are often not stealthy enough, and thanks to the explanations, we were able to highlight that the detector often relies on easy clues to detect them and not especially on the "real" malicious behavior. This is not an issue due to the learning of the ML model but a problem related to data quality. The explanations are an excellent tool to improve detection by checking the proper functioning of the (N)IDS. Lastly, we also used the anomaly scores associated with the explanations provided by our explaining method to cluster the network connections by the attack type they belong to and obtained good clustering results.

In our study, we did not discuss the role of evading techniques in attacker behaviors in detecting anomalies. Our explanation technique is positioned as a post-hoc method. Therefore, it will act as a lens to help human experts understand the logic triggering the detection. It nevertheless does not change the results of detection. That said, the explanations produced by our method could be used by adversaries to evade the ML-driven IDS because they highlight the network characteristics on which the IDS is relying for the detection. The adversaries may change these attributes to avoid detection. Beyond that, the defenders/practitioners of ML-driven IDS may use the explanations to predict potential evading efforts and take proactive actions to harden the detection system.

All these contributions are aimed at facilitating the forensic investigation and the alert analysis done by security operators. As a future work, we plan to leverage the concept of p-values not only to provide explanations but also for the detection step. In addition, it would be interesting to investigate how the explanations could automatically help reduce FP. We believe that the explanations provided by our AE-pvalue method can be used to triage the different alerts and identify FP. First of all, we have demonstrated in Section 6.1 that the explanations can be used to perform clustering of different attack behaviors and offer a good clustering accuracy. Second, we can combine the clustering results and a few labels of attack types provided by the human analysts' manual investigations. We can then propagate the attack class memberships across the clustering results based on semi-supervised learning techniques, e.g., label propagation [20] or even supervised learning methods, which provide the classification confidence of each network traffic data. The estimated confidence can help differentiate TP from FP. For example, FP tend to locate in the ambiguous area close to the classification boundary separating benign and attack data. Therefore, FP may have low classification confidence in the label propagation process, which could be used as an indicator to identify them.

REFERENCES

[1] Amina Adadi and Mohammed Berrada. 2018. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

[2] Diana Laura Aguilar, Miguel Angel Medina-Pérez, Octavio Loyola-Gonzalez, Kim-Kwang Raymond Choo, and Edoardo Bucheli-Susarrey. 2022. Towards an interpretable autoencoder: A decision-tree-based autoencoder and its application in anomaly detection. *IEEE transactions on dependable and secure computing* 20, 2 (2022), 1048–1059.

[3] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. 2021. Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches. *Trans. Emerg. Telecommun. Technol.* 32, 1 (jan 2021), 30 pages. https://doi.org/10.1002/ett.4150

[4] Giuseppina Andresini, Annalisa Appice, Francesco Paolo Caforio, Donato Malerba, and Gennaro Vessio. 2022. ROULETTE: A neural attention multi-output model for explainable network intrusion detection. *Expert Systems with Applications* 201 (2022), 117144.

[5] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. 2022. Explaining Anomalies Detected by Autoencoders Using Shapley Additive Explanations. *Expert Syst. Appl.* 186, C (dec 2022), 14 pages. https://doi.org/10.1016/j.eswa.2021.115736

[6] Ons Aouedi, Kandaraj Piamrat, and Dhruvjyoti Bagadthey. 2020. A Semi-supervised Stacked Autoencoder Approach for Network Traffic Classification. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. 1–6. https://doi.org/10.1109/ICNP49622.2020.9259390

[7] J. E. Campbell, G. R. Carmichael, T. Chai, M. Mena-Carrasco, Y. Tang, D. R. Blake, N. J. Blake, S. A. Vay, G. J. Collatz, I. Baker, J. A. Berry, S. A. Montzka, C. Sweeney, J. L. Schnoor, and C. O. Stanier. 2008. Photosynthetic Control of Atmospheric Carbonyl Sulfide during the Growing Season. *Science* 322, 5904 (2008), 1085–1088. http://www.jstor.org/stable/20145274

[8] Fabien Charmet, Harry Chandra Tanuwidjaja, Solayman Ayoubi, Pierre-François Gimenez, Yufei Han, Houda Jmila, Grégory Blanc, Takeshi Takahashi, and Zonghua Zhang. 2022. Explainable artificial intelligence for cybersecurity: a literature survey. *Annals of Telecommunications* 77 (2022), 789 – 812.

[9] Koby Crammer and Gal Chechik. 2004. A Needle in a Haystack: Local One-Class Optimization. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04)*. Association for Computing Machinery, New York, NY, USA, 26. https://doi.org/10.1145/1015330.1015399

[10] Thijs van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. 2022. DEEPCASE: Semi-Supervised Contextual Analysis of Security Events. In *2022 IEEE Symposium on Security and Privacy (SP)*. 522–539. https://doi.org/10.1109/SP46214.2022.9833671

[11] Swastik Haldar, Philips George John, and Diptikalyan Saha. 2021. Reliable counterfactual explanations for autoencoder based anomalies. In *Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD)*. 83–91.

[12] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. 2019. NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. *Proceedings 2019 Network and Distributed System Security Symposium* (2019).

[13] Nguyen Xuan Hoang, Nguyen Viet Hoang, Nguyen Huu Du, Truong Thu Huong, Kim Phuc Tran, et al. 2022. Explainable anomaly detection for industrial control system cybersecurity. *IFAC-PapersOnLine* 55, 10 (2022), 1183–1188.

[14] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1885–1894. https://proceedings.mlr.press/v70/koh17a.html

[15] Maxime Lanvin, Pierre-François Gimenez, Yufei Han, Frédéric Majorczyk, Ludovic Mé, and Éric Totel. 2023. Errors in the CICIDS2017 Dataset and the Significant Differences in Detection Performances It Makes. In *Risks and Security of Internet and Systems*, Slim Kallel, Mohamed Jmaiel, Mohammad Zulkernine, Ahmed Hadj Kacem, Frédéric Cuppens, and Nora Cuppens (Eds.). Springer Nature Switzerland, Cham, 18–33.

[16] Laetitia Leichtnam, Eric Totel, Nicolas Prigent, and Ludovic Mé. 2020. Sec2graph: Network Attack Detection Based on Novelty Detection on Graph Structured Data. In *DIMVA*. 238–258.

[17] Lisa Liu, Gints Engelen, Timothy Lynar, Daryl Essam, and Wouter Joosen. 2022. Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In *2022 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 254–262.

[18] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *NIPS*.

[19] Quoc Phong Nguyen, Kar Wai Lim, Dinil Mon Divakaran, Kian Hsiang Low, and Mun Choon Chan. 2019. GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection. In *2019 IEEE Conference on Communications and Network Security (CNS)*. 91–99. https://doi.org/10.1109/CNS.2019.8802833

[20] Rattana Pukdee, Dylan Sam, Maria-Florina Balcan, and Pradeep Ravikumar. 2023. Label Propagation with Weak Supervision. arXiv:2210.03594 [cs.LG]

[21] 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* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[22] Arnaud Rosay, Eloïse Cheval, Florent Carlier, and Pascal Leroux. 2022. Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017. In *International Conference on Information Systems Security and Privacy*.

[23] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *ICISSP*.

[24] Véronne Yepmo, Grégory Smits, and Olivier Pivert. 2022. Anomaly explanation: A review. *Data & Knowledge Engineering* 137 (2022), 101946.

A SEC2GRAPH MODEL

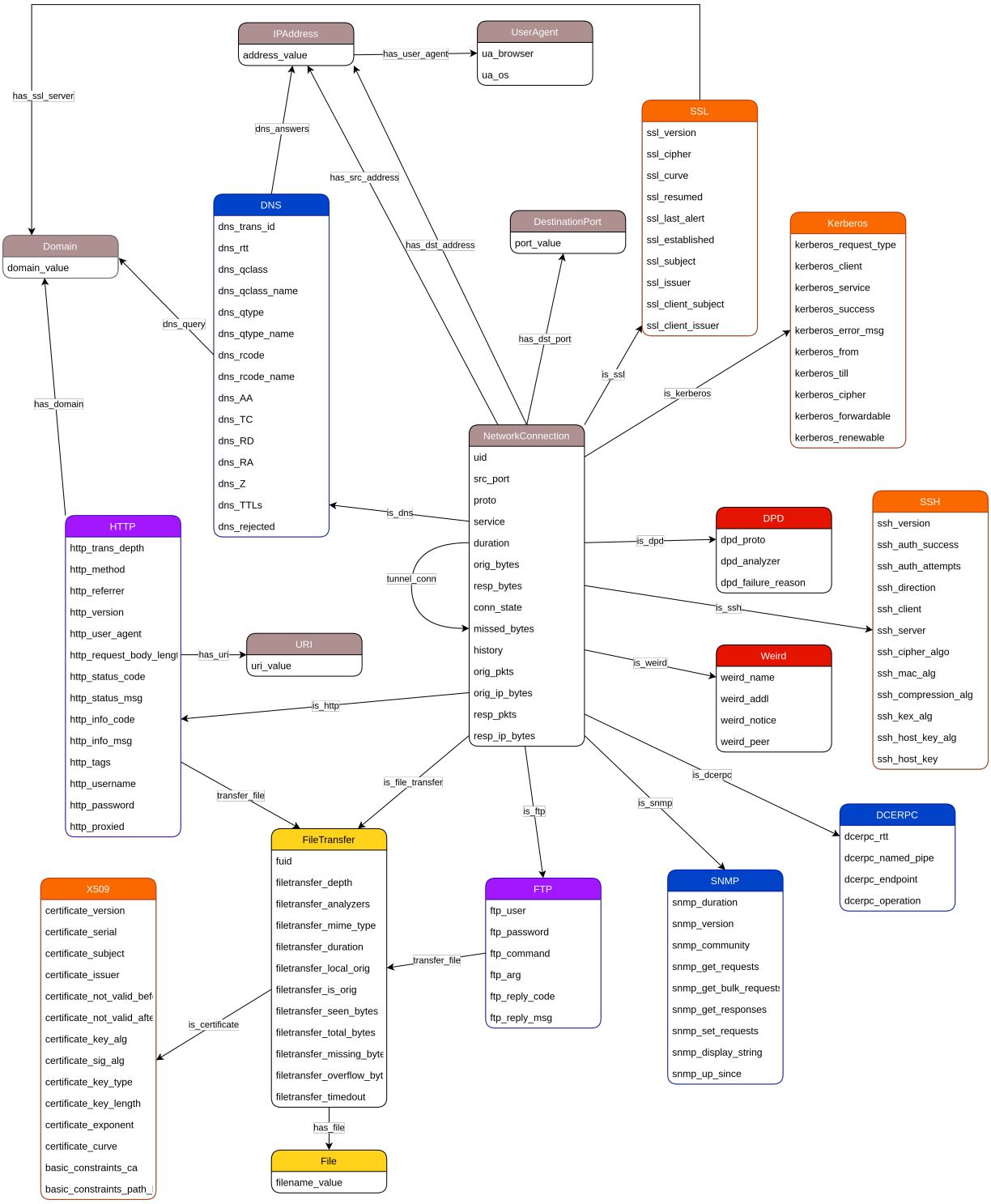

Figure 9: Sec2graph: security object graph definition

B CLUSTERING RESULTS DETAILS

no_clusters_attr	attack_type	counts	cluster_proportion (%)
0	dos slowhttptest	764	30.0
0	ddos	607	23.9
0	dos goldeneye	504	19.8
0	dos hulk	286	11.2
0	dos slowloris	256	10.1
0	web attack - brute force	73	2.9
0	web attack - xss	18	0.7
0	infiltration - portscan	12	0.5
0	botnet	10	0.4
0	ftp-patator	5	0.2
0	infiltration	6	0.2
0	heartbleed	1	0.0
0	web attack - sql injection	1	0.0
1	ftp-patator	995	57.3
1	botnet	726	41.8
1	infiltration - portscan	9	0.5
1	portscan	4	0.2
1	infiltration	1	0.1
2	dos hulk	605	48.0
2	ddos	393	31.2
2	dos goldeneye	262	20.8
2	dos slowhttptest	1	0.1
3	dos slowloris	744	100.0
4	ssh-patator	1000	100.0
5	portscan	996	44.5
5	infiltration - portscan	979	43.8
5	dos slowhttptest	232	10.4
5	dos hulk	22	1.0
5	dos goldeneye	8	0.4
6	dos goldeneye	226	68.9
6	dos hulk	87	26.5
6	web attack - sql injection	12	3.7
6	dos slowhttptest	3	0.9

Table 5: Details of the cluster content.