Hunting cyberattacks: experience from the real backbone network

Mikołaj Komisarek[1,2], Marek Pawlicki[1,2*], Mikołaj Kowalski[3], Adrian Marzecki[3], Rafał Kozik[1,2] and Michał Choraś[2,4]

[1]ITTI Sp. z o.o., Poznań, Poland
[2]Bydgoszcz University of Science and Technology, Bydgoszcz, Poland
[3]Orange, Warsaw, Poland
[4]FernUniversität in Hagen, Germany

Received: November 22, 2021; Accepted: May 18, 2022; Published: June 30, 2022

Abstract

Computer networks are exposed to attacks which have been increasingly more effective. To counter these emerging threats, researchers and security engineers work relentlessly to keep up with the arms race and offer improvements to intrusion detection systems as soon as possible. In the recent years, there has been an increase in the proliferation of systems employing deep learning and machine learning algorithms to detect suspicious patterns more effectively. To leverage AI effectively in a real-world scenario of intrusion detection, a scalable stream processing system to feed the detection algorithms with data samples in a timely and reliable manner has to be established. In this paper, two use cases of intrusion detection are presented. The first one shows a real-world example of data collected by one the largest telecom operators - ORANGE. The data was gathered for the SIMARGL project. The second use case presents the experiments and results of intrusion detection based on the Netflow scheme. The paper also proposes a scalable streaming architecture based on the Apache Spark and Apache Kafka technologies. The results of the evaluation of the effectiveness of detecting malicious behavior in network packets using several machine learning techniques in conjunction with the stream processing framework are presented.

Keywords: Machine learning, Stream processing, Intrusion detection, Network data

1 Introduction

The number and variety of Internet services continue to grow, and so does the number of network related threats. It is a strenuous task for software engineers and security operatives to provide impeccable security. Despite their efforts, data leaks, security breaches, malware attacks, denial of service attacks and many other are taking place repeatedly and consistently, leading to a whole range of dire consequences for the companies, the services and the citizens involved. Furthermore, the utilisation of ransomware by cybercriminals for financial gains is a widespread phenomenon. Critical infrastructure and industries such as oil and gas, finance, healthcare, food and beverage, and transportation are on the cross-hair [1]. The cybersecurity provider SonicWall reported a record 78.4 million ransomware attacks worldwide in 2021 [2]. One of the more momentous attacks in 2021 targeted a U.S. oil pipeline. The attack disrupted fuel delivery to most of the U.S. East Coast for several days. Another case study comes from March 2021 - Acer, the Taiwanese multinational hardware and electronics corporation, was attacked by

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 13(2):128-146, June 2022
DOI: *10.22667/JOWUA.2022.03.31.128*

*Corresponding author: ITTI Sp. z o.o., Rubież 46, 61-612 Poznań, Poland, Tel: +48 61/ 622 69 85, Web: https://www.itti.com.pl, Email: mpawlicki@itti.com.pl

the ransomware REvil. The criminals responsible for this attack demanded a ransom of $50 million. An example of a successful attack from the insurance sector was carried out against the CNA Financial Corp, a financial corporation based in Chicago. The company reportedly lost about 75,000 records of its customers' data. The lost data may have included vital, personal information like the names, health benefit information, and Social Security numbers of current and former company employees, contract workers, and their dependents.

In the attack on Kia Motors, a South Korean multinational automobile manufacturer, the so-called DopplePaymer gang, responsible for the ransomware attack, demanded a ransom of $20 million to decrypt company data and not to reveal the stolen company secrets. The victim later disclosed information about the affected services.

Criminals are increasingly trying to steal the data of ordinary people [3].

These contemporary examples provide an outline of the state of the cyberspace. Cybersecurity is crucial and the lack of adequate countermeasures can result in disastrous consequences. Every present-day cyberspace participant, regardless if it is a company, a critical infrastructure or a singular citizen should be as prepared as possible for the possibility of an attack. This is why it is so important to invent and improve techniques to prevent such events. The struggle between attackers and defenders is like an arms race, where cybersecurity experts are inventing new defensive techniques, and cybercriminals are attempting to circumvent them.

This paper will present an Intrusion Detection System (IDS) which leverages machine learning and stream processing to detects unwanted patterns in computer networks in real time. The data to train and verify the defense system was collected from real-world ORANGE traffic for the H2020 SIMARGL project. The main objective of the project is to counteract cyberthreats and invent effective defense techniques. This paper is an extended version of the work presented in [4].

The paper is structured as follows: Section two presents related work highlighting similar solutions. Section number 3 is then devoted to the datasets used in this research paper and a brief presentation of the use cases. Section 4 introduces the reader to the structure and architecture of our system. Section 5 is then devoted to describing the methodology and the metrics that describe the performance of each detector. The last two sections refer to the experimental results and the conclusion.

2 Related works

Sophisticated defenses are built for the purpose of stopping security breaches, among those a set of mechanisms called Intrusion Detection Systems (IDS)[5].

Multiple recent machine-learning-based network intrusion detection propositions can be found in the literature [6]. A large fragment of those research items is based on supervised learning. An important factor in the supervised learning pipeline is a proper, annotated dataset that contains a healthy mix of various examples of correctly labeled attacks.

In [7], the authors proposed a network intrusion detection system (NIDS) utilising a convolutional neural network (CNN) in a multi-classification model. The model was trained on the KDD-CUP'99 and NSL-KDD datasets. In the first step, the samples were transformed from a one-dimensional dataset to a two-dimensional one. Then, an optimized CNN was used. The authors proved in their method that the multi-classification model benefited the final results, improving accuracy and reducing false positives.

The authors of [8] devoted their attention to the topic of feature selection, noting its significance in IDS. The authors showed that by using the Krill herd (KH) algorithm in the experimental phase, it was possible to drastically reduce the number of features (down to seven) in the NSL-KDD dataset, while in the CICIDS2017 dataset, the number of necessary attributes was reduced to ten. It was also emphasized that a high detection accuracy was maintained and the time to obtain results was reduced.

Another example can be found in [9], where the authors used the Apache Spark platform to reduce the problem of time consumption during detection. They used a deep random forest to detect intrusions in computer networks. In the first stage, they used a sliding window to split the samples and train a model on it in the second stage. A multilevel cascaded random forest model was used. Finally, a voting system was selected to determine the result.

The authors of [10] considered using Flume in conjunction with Kafka for massive data handling. The traffic was fed to the well-known convolutional neural network architecture - AlexNet [11], the 2012 ImageNet competition winner. The network was trained with the KDD'99 dataset, which is one of the staples in NIDS research - a dataset available for over twenty years now. The reported detection rate was 94.32%.

In [12], a Distributed Intrusion Detection System (DIDS) has been proposed. The authors attempted to merge two novel technologies - blockchain and cloud computing - with IDS in an attempt to make it more robust and reliable. The presented work proved scalability of the platform.

The authors of [13] took an in-depth look into the data-side of network intrusion detection, taking a survey of the past decade of research. The work dissected some of the recently published datasets, including LITNET2020, and elaborated upon their pros and cons, including the problems with noisy data, redundant data, weakly correlated data, insufficient samples of certain attack types, the existence of the imbalance problem in the distribution of classes, etc. The authors found that one of the major challenges of the datasets available to researchers in the intrusion detection domain is the lack of real-world network data.

The authors of [14], provide a thorough analysis of network traffic. They conclude that most of the cyberthreats present in high-speed, wide-area networks can be identified from packet flows. In contrast, they identify challenges found in analyzing large amounts of IP packet flows and scalability issues in IDS solutions. They use network streaming as a form of network traffic delivery and analyze the effects of such a solution.

In [15], the authors present another Netflow traffic collection technique. They argue that the Netflow protocol alone does not provide enough information needed to analyze network access behavior. Here, they propose an alternative version of traffic collection that combines packet capture and flow technique to accurately identify applications.

In [16], the authors address a topic related to scalable traffic monitoring in a large network. They highlight the main challenge as efficiently collecting network traffic from hundreds of machines running in a data center. They point out that in typical networks, the gatherer configuration is centralized and such a solution is not able to cope with the challenge of large data flow from many machines, the solution may be to create a cluster of gatherers that would balance the network traffic and be able to accept large amounts of data. The authors propose a solution which is writing network traffic directly to the scalable Cassandra database and the collector used is a free open source ntop-ng software. The result of this research was to confirm the assumptions that the scalable Cassandra database can cope very well with storing large amounts of network traffic data.

The topic of NetFlow-based attack detection in networks is described in [17]; the authors focus on DDoS attacks. They emphasize at the beginning of the article the seriousness of DDoS attacks and the fact that these types of attacks are evolving both in terms of employed techniques and traffic volume. They also present how to detect DDoS attacks using NetFlow feature selection and machine learning. The detector built was based on the RandomForest algorithm and after training the model, the authors get results of 99% accuracy and a false positive of less than 0.5%.

The work presented in [18] highlights the problem of network security and the authors propose a method called SAWANT, which aims to detect malicious network traffic. The operation of this technique uses deep learning along with data extraction using the sliding window technique. The extracted attributes using this technique are transferred to a deep neural network to identify unwanted anoma-

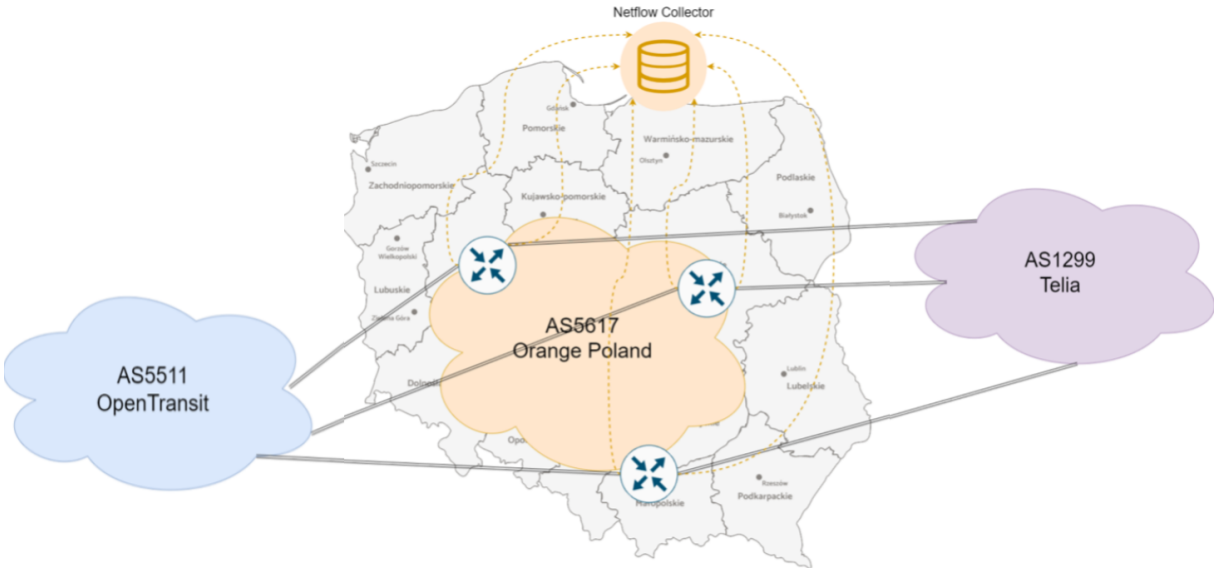


Figure 1: Structure of Orange network

lies. The authors based their study on the popular CTU-13 dataset and show that they have achieved an accuracy of 99% using a small amount of data.

In [19], the authors conduct a study related to identifying the effect of features on the detection of particular types of attacks. Histograms were used to identify the most important features and histograms from infected and uninfected samples were compared to each other.

In the research presented by the authors of [20], a set of studies dedicated to the analysis of data in the form of NetFlow and IPFIX in relation to network traffic monitoring is described. The authors introduce a comprehensive comparison of the two methods and explain their schemes. The entire research focuses on presenting an example of an ecosystem for flow monitoring. In addition, a comprehensive comparison of network traffic collection tools is presented.

Extreme Learning Machines (ELM), Distributed Random Forest, and Distributed Random Boosted-Trees were used in [21]; the authors focus on botnet attack detection. The system architecture is also presented which will enable the processing of Big Data. In this use case, network data based on the NetFlow scheme was used. The authors' research summary shows promising results against network anomalies.

In [22] and [23], the authors illustrate an IDS architecture approach designed to analyze network traffic in real time. To achieve this effect, the authors use data streams, and for this purpose they use Apache Kafka. Apache Kafka is ideal for scalable environments to transfer large amounts of data. To process large amounts of data from the stream, Apache Spark software was used along with the Elastic-search database. The authors group network traffic according to a pattern based on source IP address and sliding windows.

In [24] a set of guidelines for compliance of IDS with the European legislative framework is provided. The privacy and data protection issues with regard to modern intrusion detection systems is provided.

3 Datasets

This paper places emphasis on data sets based on the Netflow network schema. This type of data was introduced by Cisco in 1996. With this functionality, a network administrator is able to analyze traffic

and determine, among others, the source and destination of the traffic. Three components are needed to fully monitor flows using Netflow. The first is the exporter, which performs the aggregation of the flow packets on a given machine and exports them to one or more flow collectors. The second component is the gatherer, which accepts data from connected Netflow exporters. Data at this stage can be processed and stored. The final component in the ecosystem is the software that collects traffic from the gatherers and analyzes the traffic for purposes such as anomaly detection or network traffic profiling. It is worth mentioning, that the most popular Netflow versions are version 5 and 9, but because of the fact that version 9 contains much more important information, it is recommended to use this version for IDS purposes. A major challenge is to effectively analyze large amounts of network traffic in real time. In a wide area network where there are innumerable participants, the network flow can be of considerable size. This is why a scalable solution that can sustain exporting large amounts of data and post-processing it is so critical [16].

This paper will present two case studies. The first use case describes building a detector that can detect suspicious network traffic in data coming from real-world network traffic provided by ORANGE. In a description of the infrastructure and the process of collecting this data will be provided. The second scenario showcases the results obtained by the proposed solution on public benchmark network security datasets.

3.1 Overview of the ORANGE infrastructure and dataset

The TPNET (AS5617) is the Orange Poland backbone network. It consists of numerous routers and has several entry points (upstream links). As can be seen in Figure 1, two upstream service providers are providing connectivity with other autonomous systems. Total aggregated capacity of those uplinks is measured in the Tbps (terabit) scale. Although some Indicators of attack (IOA) - mostly originating from badly configured subscribers' equipment - can be seen inside of the network, we believe that the majority of threats is coming through inter-AS uplinks. This is the data source for analysis in this work. Currently, there are seven routers that are providing connectivity with the upstream. To allow acquisition, IPFIX/jFlow [25] data was generated from the production ingress traffic directly. The router was operating JunOS platform with a sample 1:2000 ratio on each interface. This data is fed to Apache Kafka. The structure of the network data collected with the collector is presented in Table 1. The table contains the names of thirteen fields that represent the network data along with their descriptions. The data allows for efficient network intrusion detection. Since the dataset was collected from live network traffic, the ratio of normal to anomalous traffic was 122:1. Data balancing techniques were used to counter the data imbalance problem, as described in Section 5.

3.2 Overview of the NIDS dataset

The work showcased in this paper utilises the NetFlow scheme.

As highlighted in the "Related Works" section, flow-like features allow to build effective Network Intrusion Detection methods. A comprehensive description of the Netflow V9 protocol can be found in [26].

In part of this research, the NF-UNSW-NB15 dataset will be used. The set is based on the well-known UNSW-NB15 dataset. The authors of NF-UNSW-NB15 present a comprehensive paper in which they convert several popular datasets into NetFlow format datasets [27]. These have been extended with additional features that are found in the NetFlow schema. In addition, the data is tagged with appropriate attack categories. The total dataset contains 2,390,275 flows, including 95,053 infected flows. Attacks included in this collection can be classified into 9 categories.

Table 1: The schema of a network sample

#	Field Name	Description
1	IN_BYTE	Incoming flow bytes
2	OUT_BYTE	Outgoing flow bytes
3	IN_PKT	Incoming flow packets
4	OUT_PKT	Outgoing flow packets
5	SRC_IP_ADDR	IPv4 source address
6	DST_IP_ADDR	IPv4 destination address
7	SRC_PORT	IPv4 source port
8	DST_PORT	IPv4 destination port
9	PROTO	Protocol
10	BYTES	Sum of bytes
11	FLOWS	Sum of flows
12	PACKETS	Sum of packets
13	DURATION	Flow duration

The second benchmark dataset used in this study is the ToN-IoT converted to the NetFlow form. Its origin is the IOT network; the total number of frames available in this set is exactly 16 940 496. 63.99% of this traffic is infected, which translates into 10 841 027 flows, and the remaining 36.01% is normal traffic which corresponds to 6 099 469 flows.

The third dataset used in this work is the NF-BoT-IoT dataset. Its structure is based on the NF-ToN-IoT dataset, although the authors generated more malicious traffic. Its size is 37,763,497 network frames, divided in the ratio 99.64% of network attacks to only 0.36% of the normal traffic.

The last public benchmark dataset used in this paper is the popular CSE-CIC-IDS2018 collection, which was converted from PCAP files to the Netflow schema. The total size of this set is 18,893,708 flows. The set contains 88.05% of normal traffic (16,635,567 flows), the rest are 2,258,141 attacks (11.95%).

A detailed description of the conversion process from PCAP files to the Netflow protocol and the creation of the aforementioned datasets is described in [27].

4 Practical IDS system deployment

As explained in the previous section, the data gathered by Orange comes from a production node. The data was used to formulate a model that is at the basis of the proposed IDS system. The IDS is based on a modular architecture and each module performs a separate task. The software leverages the Apache Kafka [28] platform for data streaming, which is optimal for transporting large amounts of data through the stream. This solution ensures stability and guarantees delivery of the sent data. It also guarantees high performance and ability to operate in a distributed environment. Apache Kafka was field-proven for message distribution by tech giants such as LinkedIn, Netflix or Spotify. In 2019, the total number of messages handled by Apache Kafka in LinkedIn exceeded 7 trillion per day [29]. This proves software stability and performance in big-data conditions.

Network frames that have passed through the proposed IDS and been classified are saved and stored in the Elasticsearch Database [30]. This solution was also chosen for its high search/indexing performance and seamless scalability, which increases the capabilities of the database and reduces threats such as service unavailability. Additionally, the database engine itself provides a REST API, thanks to which it is possible to retrieve data via HTTP queries. Additionally, the database can be equipped with a visual-

isation tool to increase situational awareness by serving customisable reports filled with actionable intel of choice. The module is called Kibana [31].

The software developed to protect against potential network attacks, code-named the "Big Data Engine (BDE)", uses the capabilities and extends the Apache Spark framework [32]. The use of this platform is ideal for processing large amounts of data and the dispersion of tasks to different servers. The proposed solution provides several modules that are able to run directly on the Apache Spark platform. For this purpose, languages such as Scala or Python were used.

The first module is a component which accepts data from a source; it can be a file, data stream or a database. On this data, all necessary processes are performed - cleaning, scaling or feature selection.

The process of checking and cleaning the data contained in a single frame involves discarding frames that are incomplete. Due to the fact that using network data collector ensures complete data extraction without any loss of information, this stage is only triggered to ensure 100% complete data in prediction or machine learning stage.

Some ML algorithms are sensitive to feature scale, so as part of the data preprocessing step feature scaling mechanisms are employed. The flows were subjected to the StandardScaler method, which performs feature standardisation removing the mean and scaling to unit variance.

Having completed the preprocessing steps, feature selection was performed.

Irrelevant or noisy feature values in the data can reduce the accuracy of ML models. The use of the feature selection process aims to reduce the computational time of the model and increase the metrics achieved by anomaly detection. The feature selection method used in this study is the chi square method [33]. It is based on calculating the chi square between each feature and the target value.

$$chi^2 = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \tag{1}$$

In Equation (1) N is the observed value of w, and E the expected value. e_t takes the value of 1 if the sample contains the term t, and 0 otherwise. e_c takes the value 1 if the sample belongs to class c, and 0 otherwise.

Then, the pre-processed data is saved or goes back to the stream, where it can be forwarded to the ML-based classifier module. The classifier module can be trained on the obtained data, or it can perform the classification task on the basis of a defined model.

The module that performs the function of training a particular classifier is always run when one needs to train a new model or retrain the existing one. This task is performed in the offline mode, and annotated data must be provided from a file or the stream [34].

The module that performs the task of classification of a given frame reads the provided model (e.g., artificial neural network) and on its basis, it assigns a specific label. Typically, the result and a particular sample are saved to the database, or returned again to the stream. This design provides the possibility of formulating and using infinitely many different models and processes. The entire solution is shown with all the components in Figure 2.

5 Methods

Detecting anomalies in the network traffic is a challenging endeavour, as more and more threats reach modern networks every day [35]. To counter novel abuses, software is being developed that effectively detects the transgressions and allows the security operators to thwart the attacks. Thanks to the data provided by Orange, research could be accomplished on detecting anomalies in real-world network traffic. Using the presented platform, several stages of testing were performed with the goal of providing domain-standard measures of the effectiveness of the trained classifiers.

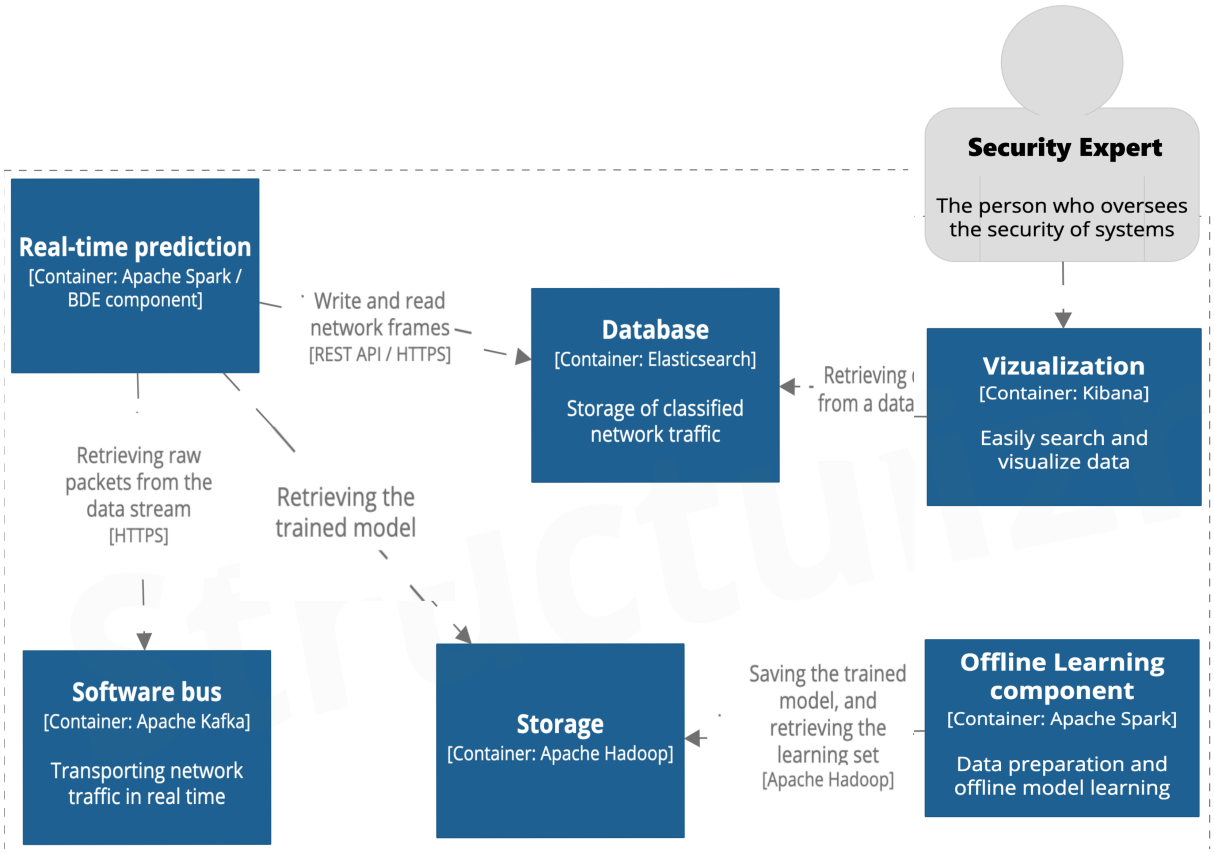


Figure 2: Big data engine - Architecture

The first stage was to load the provided network traffic frames from CSV files. At this stage of research, the individual samples that did not contain complete data were removed - in future work imputation methods will also be tested on this data. The next step was to use a feature selection technique, by which the final shape of the input field schema was established.

There are many ways of approaching netflow-based feature extraction and dimesionality reduction [**?**], [**?**]. In this work, a set of IPFIX fields was exported. The SelectKBest algorithm from the scikit-learn [36] package was then used to choose the features which contribute to the result. The method selects features according to the k highest scores. By changing the parameter 'score_func', we can apply the method to both classification and regression data, making it suitable for all types of datasets. Using this feature in machine learning eliminates the less relevant part of the data and reduces the training time.

In addition, multidimensionality was reduced using principal component analysis (PCA). After the data preparation was completed, the last step before training the model was to tackle the problem of unbalanced distribution of classes in the dataset. Data imbalance may lead to the deterioration of the model effectiveness metrics. For this purpose, the number of samples belonging to minority classes was increased using the Synthetic Minority Oversampling Technique (SMOTE) [37] algorithm.

In this research work, a range of classifiers has been used to build anomalous network traffic detectors. In the first use case, the following models were used: Random Forest, Xboost, AdaBoost and ANN. In the second use case, the LightGBM algorithm was also added.

5.1 Description of used machine learning models

The choice of the particular ML methods and their implementations is based on the effectiveness achieved in our previous works [22], [23] and [4], and initial experiments.

The first classifier included in this study is the Random Forest algorithm. This method belongs to the group based on the ensemble learning technique. The benefit of using this classifier is the simplicity of application. The algorithm does not require extensive hyperparameter tuning, fits to the data relatively quickly and achieves satisfactory metrics on NIDS data. The working principle of this algorithm is based on creating multiple decision trees at the training stage and using majority voting to determine the classification. By using the majority voting method, the problem of overfitting the training data is reduced. Additionally, in the training stage, the model uses bagging. This technique involves repeatedly selecting a random sample from the training set, and matching trees to these samples.

The main advantage of the LightGBM [38] classifier is its high performance and distributability. Working principle of the algorithm is based on boosted decision trees. The difference that sets it apart from other boosting algorithms is its partitioning method, as it grows the trees leaf-wise with starting with the tree having the most impact on the decrease of the gradient, while other boosted trees algorithms grow the trees level by level.

The Adaptive Boosting classifier (AdaBoost) [39] is a boosted decision trees variation with the main advantage of being less prone to overfitting, as the input parameters are not optimized together. However, when using this classifier, one needs to use a good quality set and avoid noisy and outlier data. The technique involves adding an element of boosting and adaptively adjusting the weights on misclassification. This results in weak weights being converted to strong weights.

Artificial neural network (ANN) - The working principle of the artificial neural network is is based on layers of computational nodes called neurons connected by weights adjustable by backpropagation of error. In this research paper, the network construction starts with a hidden layer having 64 neurons, and a second layer of 32 neurons. In both cases, the activation fucntion is set for Rectified Linear Unit (RELu). The final layer has the exact number of neurons as the number of classes in the researched dataset (number of anomaly types). The neural network loss function in the classification case was ”categorical_crossentropy” , while the selected optimization algorithm was Adaptive Momentum (ADAM).

The employment of neural networks in this work is was carried out due to multiple advantages ANNs offer in comparison to traditional machine learning techniques. First and foremost, with large volumes of data ANNs can continue to learn when traditional ML techniques saturate. Another benefit is the ability to freeze parts of the network and retrain the remaining part, which is useful from the perspective of adapting the network to new, but similar data, for example in the occurance of concept drift - which is out of scope of this paper, but is part of the general research direction in the real-world deployment of ML-based IDS.

XGBoost algorithm, otherwise known as Extreme Gradient Boosting is considered a classifier belonging to the family of boosting algorithms. Its operation is based on the use of gradient boosting (GBM); however, instead of using gradient descent in the function space, XGBoost relies on Newton-Raphson with the use of a second order taylor’s approximation. The boosting technique operates on the principle of ensemble, combining a set of weak learners and providing better prediction accuracy.

5.2 Metrics for model evaluation

To be able to compare the performance of ML algorithms, a number of metrics is calculated. The evaluation metrics are computed on the basis of the the confusion matrix.

In the confusion matrix for IDS, four main values are presented - the first value describes cases in which a sample was correctly classified as an attack. This value is referred to a true positive (TP). The

second type of metric refers to the correct classification that the sample is benign. This metric is referred to as true negatives (TN). The next two metrics relate to misclassification. The first, referred to as false positives (FP), represents the fact that the algorithm classified the occurrence of an event where in fact it did not occur. The last value is the number of false negatives (FN), which aggregates the instances where the algorithm classifies that an event did not occur where, according to the data it did, in fact. All the variables presented above are then used to calculate a set of informative metrics.

In this paper, the following measures are used to evaluate the algorithms: Accuracy (ACC - equation: 2), Precision (Pr - equation: 3), Recall (Re - equation: 4), F1-Score (equation: 5), Balanced accuracy (BCC - equation: 6) and Matthews correlation coefficient (MCC - equation: 7).

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{2}$$

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{5}$$

$$BCC = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2} \tag{6}$$

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{7}$$

6 Experiments and Results

This section is divided into two use cases. The first scenario is the real-world example of network intrusion detection on the traffic coming from the ORANGE network. The second scenario presents the results of intrusion detection using the public NetFlow benchmark datasets.

6.1 Usecase - Orange

In the previous section, the proposed methodology was presented. The results of the research based on the data provided by Orange will be described in this section. The feature contribution to the classification was estimated using the SelectKBest algorithm. The result of this procedure is presented in Figure 3.

After data pre-processing, the performance of four algorithms in terms of their ability to correctly distinguish anomalous network traffic from the normal one was tested. The notion of proper hyperparameter setup can influence the final results in ML [40]. The first classifier is Random Forest. The number of estimators was configured to be 100, the maximum tree depth was set to 10, and the remaining variables were left as defaults. The results for this classifier are presented in Table 2.

Gradient Boosted Trees is the second algorithm tested for detection of intrusions in the provided network traffic. The obtained results can be found in Table 3.

Another tested algorithm is LightGBM. It complements and extends the research that was conducted in the original paper. The results obtained can be found in the table 4.

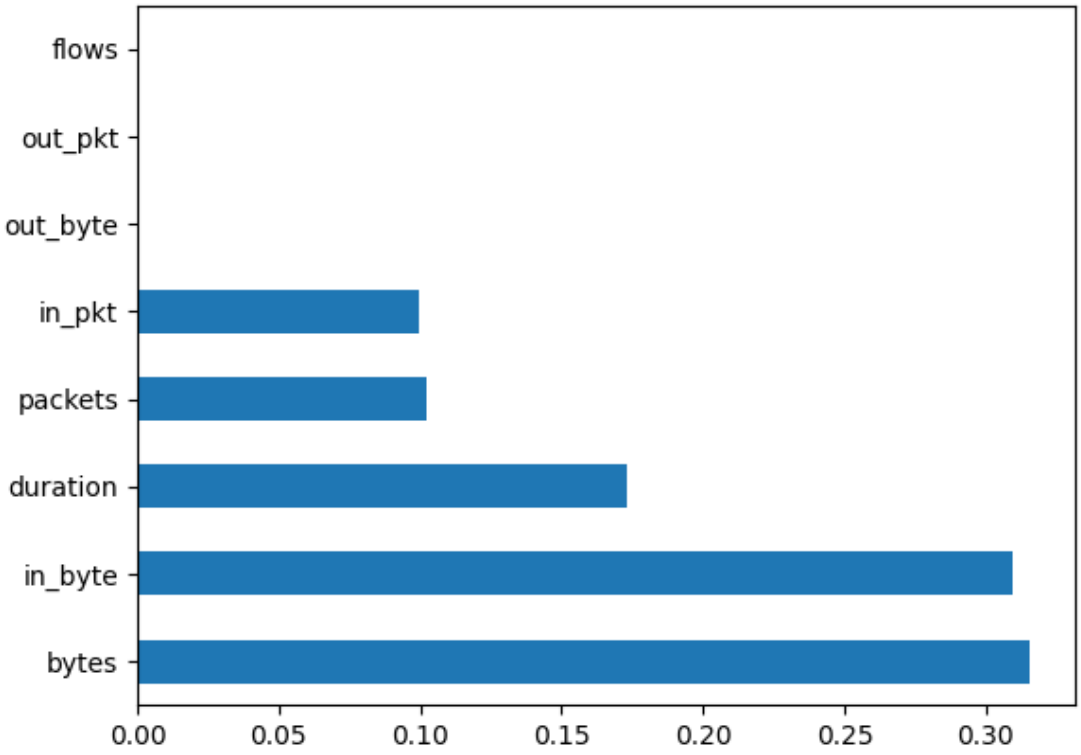


Figure 3: Result of feature importance

Table 2: Summary of the results for the Random Forest Classifier

#	Accuracy(AVG)	Precision (AVG)	Recall (AVG)	F1 (AVG)	BACC	MCC
1	0.91	0.91	0.91	0.91	0.9135	0.8272
2	0.91	0.91	0.92	0.91	0.9140	0.8280
3	0.91	0.91	0.91	0.91	0.9240	0.8210
4	0.91	0.91	0.91	0.91	0.9139	0.8279
5	0.92	0.92	0.92	0.92	0.9155	0.8311
6	0.91	0.91	0.91	0.91	0.9148	0.8298
7	0.91	0.91	0.91	0.92	0.9137	0.8275
8	0.92	0.92	0.92	0.92	0.9154	0.8310
9	0.91	0.91	0.91	0.91	0.9147	0.8295
10	0.91	0.91	0.91	0.91	0.9147	0.8296

The penultimate tested classifier is AdaBoost and its results can be found in Table 5. When building this classifier, the main estimator was set to DecisionTreeClassifier, initializing it with a depth of one. The maximum number of estimators after which learning is terminated was 150.

The weight applied to each classifier in each boosting iteration was 1.

The final model to conclude the study was an artificial neural network. The network structure is as follows: The used activation function was the Rectified Linear Unit (ReLU), the ANN had just one

Table 3: Summary of the results for the Gradient Boosted Trees

#	Accuracy(AVG)	Precision(AVG)	Recall(AVG)	F1(AVG)	BACC	MCC
1	0.88	0.88	0.88	0.88	0.8757	0.7515
2	0.88	0.88	0.88	0.88	0.8760	0.7529
3	0.88	0.88	0.88	0.88	0.8761	0.7532
4	0.88	0.88	0.88	0.88	0.8772	0.7546
5	0.88	0.88	0.88	0.88	0.8772	0.7555
6	0.88	0.88	0.88	0.88	0.8770	0.7541
7	0.88	0.88	0.88	0.88	0.8768	0.7537
8	0.88	0.88	0.88	0.88	0.8767	0.7536
9	0.88	0.88	0.88	0.88	0.8775	0.7561
10	0.88	0.88	0.88	0.88	0.8772	0.7556

Table 4: Summary of the results for the LightGBM

#	Accuracy(AVG)	Precision(AVG)	Recall(AVG)	F1(AVG)	BACC	MCC
1	0.89	0.89	0.89	0.89	0.8757	0.7515
2	0.90	0.90	0.90	0.90	0.8760	0.7529
3	0.88	0.88	0.88	0.88	0.8761	0.7532
4	0.89	0.89	0.89	0.89	0.8772	0.7546
5	0.90	0.90	0.90	0.90	0.8772	0.7555
6	0.89	0.89	0.89	0.89	0.8770	0.7541
7	0.90	0.90	0.90	0.90	0.8768	0.7537
8	0.89	0.89	0.89	0.89	0.8767	0.7536
9	0.88	0.88	0.88	0.88	0.8775	0.7561
10	0.89	0.89	0.88	0.88	0.8772	0.7556

Table 5: Summary of the results for the AdaBoost Classifier

#	Accuracy(AVG)	Precision(AVG)	Recall(AVG)	F1(AVG)	BACC	MCC
1	0.86	0.86	0.86	0.86	0.8556	0.7132
2	0.85	0.86	0.85	0.85	0.8535	0.7087
3	0.85	0.86	0.85	0.85	0.8549	0.7119
4	0.86	0.86	0.86	0.86	0.8560	0.7142
5	0.86	0.86	0.86	0.86	0.8553	0.7127
6	0.86	0.86	0.86	0.86	0.8560	0.7141
7	0.86	0.86	0.86	0.86	0.8557	0.7133
8	0.85	0.86	0.85	0.85	0.8519	0.7101
9	0.85	0.86	0.85	0.85	0.8549	0.7119
10	0.86	0.86	0.86	0.86	0.8558	0.7138

Table 6: Summary of the results for the deep neural network

#	Accuracy(AVG)	Precision(AVG)	Recall(AVG)	F1(AVG)	BACC	MCC
1	0.88	0.88	0.88	0.88	0.8792	0.7591
2	0.88	0.88	0.88	0.88	0.8791	0.7595
3	0.88	0.88	0.88	0.88	0.8762	0.7537
4	0.88	0.88	0.88	0.88	0.8755	0.7531
5	0.87	0.87	0.87	0.87	0.8719	0.7450
6	0.86	0.86	0.86	0.86	0.8569	0.7143
7	0.86	0.86	0.86	0.86	0.8571	0.7144
8	0.87	0.87	0.87	0.87	0.8746	0.7493
9	0.88	0.88	0.88	0.88	0.8770	0.7550
10	0.88	0.88	0.88	0.88	0.8787	0.7586

Figure 4: The flow-based real-time machine learning network intrusion detection data processing pipeline

hidden layer to ensure fast processing.

The results generated by this approach can be found in Table 6

6.2 Usecase - NIDS

The following paragraphs showcase the experiments and the results of NIDS based on NetFlow protocol. With the use of machine learning algorithms: Random Forest, LightGBM, XBoost and ANN, network intrusion detection algorithms were built with the following datasets: NF-UNSW-NB15, NF-BoT-IoT, NF-CSE-CIC-IDS2018, and NF-ToN-IoT.

The data preparation process was constant for each dataset. The datasets have a unified feature vector rooted in the Netflow schema. The full process can be seen in Fig. 4.

Tuning and finding the appropriate parameters for each of the classifiers used involved applying a process of finding the ideal model architecture called hyperparameter tuning. The ”Grid Search” method was used to find the most optimal parameters. The method works by building models for each possible combination of all provided values of hyperparameters and then evaluating the results that these combinations bring in order to choose the architecture that provides the best metrics. In Table 7, the final hyperparameter configuration for each classifier is shown. The structure of the table includes the name of the algorithm, the name of the hyperparameter and the value used in this research.

The results of the study are presented in the summary in Table 8. A number of metrics were used to determine the performance of individual detectors. The higher the value in a particular metric, the better the indicated effectiveness of the particular classification algorithm is.

All the used classifiers achieved an accuracy above 99%, with high metrics across the board.

Comparing the two case studies with results displayed in Tables 2, 4, 3, 5, 6, and 8 shows that in both cases the effectiveness of using machine learning is more than 91% with respect to detecting attack

Table 7: The final result of tuning hyperparameters by using the GridSearch technique.

Model	Parameter	Value
Random Forest	n_estimators	200
	max_features	auto
	max_depth	10
	criterion	entropy
LightGBM	max_depth	2
	feature_fraction	0.5
XBoost	var_smoothing	1e-9
ANN	epochs	16
	batch size	20
	loss function	categorical_crossentropy

Table 8: A table summarizing the results of all tests for the four datasets and the Random Forest, XBoost, LightGBM, AdaBoost and ANN algorithms

NF-UNSW-NB15						
Classificator	**Accuracy**	**Precision**	**Recall**	**F1**	**Bcc**	**Mcc**
Random Forest	**1**	**0.99**	**0.99**	**0.99**	**0.97**	**0.96**
XBoost	1	0.99	0.97	0.98	0.97	0.95
LightGBM	1	0.99	0.97	0.98	0.95	0.95
AdaBoost	1	0.99	0.97	0.98	0.95	0.95
ANN	1	0.98	0.97	0.99	0.96	0.97
NF-BoT-IoT						
Classificator	**Accuracy**	**Precision**	**Recall**	**F1**	**Bcc**	**Mcc**
Random Forest	1	1	0.98	0.99	0.98	0.98
XBoost	**1**	**1**	**0.99**	**0.99**	**0.98**	**0.98**
LightGBM	1	1	0.98	0.98	0.97	0.98
AdaBoost	1	1	0.98	0.98	0.97	0.98
ANN	1	0.98	0.97	0.97	0.94	0.95
NF-ToN-IoT						
Classificator	**Accuracy**	**Precision**	**Recall**	**F1**	**Bcc**	**Mcc**
Random Forest	**1**	**1**	**0.99**	**0.99**	**0.98**	**0.98**
XBoost	1	1	0.98	0.98	0.96	0.98
LightGBM	1	1	0.99	0.98	0.98	0.98
AdaBoost	1	1	0.98	0.98	0.97	0.98
ANN	0.99	0.97	0.99	0.99	0.99	0.96
NF-CSE-CIC-IDS2018						
Classificator	**Accuracy**	**Precision**	**Recall**	**F1**	**Bcc**	**Mcc**
Random Forest	**1**	**0.98**	**1**	**0.99**	**0.99**	**0.97**
XBoost	1	0.95	1	0.97	0.98	0.97
LightGBM	**1**	**0.98**	**1**	**0.99**	**0.99**	**0.97**
AdaBoost	1	1	0.98	0.98	0.97	0.98
ANN	0.99	0.97	0.99	0.99	0.99	0.96

pattern behaviour in network traffic based on NetFlow data. Of the tested algorithms the highest results in the first case study was achieved using the Random Forest algorithm, which achieved accuracy of 91% on data coming from the real network as provided by Orange. When tested on netflow NIDS benchmark datasets datasets, the lowest metric was detection results were comparable on all algorithms used, with Random Forest having a slight lead on three out of four benchmarks.

Tables 2-6 contain the results of the first use case for each of the employed algorithms. The Tables report results of the 10 folds of Cross-Validation. Each case reports six validation metrics (Accuracy, Precision, Recall, F1, Balanced Accuracy - BACC and Matthiews Correlation Coefficient - MCC). The way the metrics are calculated is presented in Section 5.2. As can be observed from those Tables, the 10 folds of Cross-Validation present stable results each of the tested algorithms.

The results summarizing the experiments constituting the second use case are included in Table 8.

The algorithm that gets the highest metrics is Random Forest, closely followed by other tree-based algorithms - Xboost and LightGBM. In both use cases it can be concluded that the use of ML results in high detection of anomalies in the network.

7 Conclusion

The research presented in this paper is an extension of the work we have presented in [4], which completely describes the use case related to detection of anomalous traffic in the real-world ORANGE network. The extension of the research provided in this paper presents intrusion detection based on the NetFlow schema. To validate the effectiveness of the proposed solution, which allows for using algorithms such RandomForest, XBoost, ANN or LightGBM, four NIDS datasets were used.

In comparison to our original paper [4], the work showcased in this paper extends the research of detection of attacks on a dataset collected from a real-world computer network managed by Orange. The extension presents the use of the detection pipeline presented in the original paper on novel datasets and with more algorithms. Thus, the added value comes from increased reproducibility due to four large NIDS Netflow open benchmark datasets form. The research has also increased the number of covered machine learning algorithms with the following: LightGBM and Xboost. Within this paper the issue of scalable architecture of the solution which is able to process real network traffic was also presented.

The research in this paper carried out under the H2020 SIMARGL project and serves the creation and real-world deployment of a machine-learning-based network intrusion detection solution. Some of the AI solutions used in this work were also researched in the H2020 SPARTA project. As of the time of writing this paper, the ML-based NIDS is a fully-fledged engine, which enables the utilisation of the entire ML pipeline with the use of industry-standard libraries like sci-kit and TensorFlow. The prepared models can be employed to perform real-time classification by ingesting data from the Kafka stream. The engine, its principles, testing and how it pushes the envelope for ML-based NIDS was the subject of a series of scientific publications [41], [42], [43], [44]. The solution is being continuously developed and integrated, with the problems found in deployment giving inspiration for new research directions and scientific work.

In the long term, the research concerning the NIDS is to be extended to include elements related to the domains of life-long learning, concept drift detection and transfer learning / domain adaptation. In addition, future work is devoted to further improvements towards integrating more machine learning concepts and algorithms, spearheading further scientific developments in the utilisation of ML for network intrusion detection.

Acknowledgements

This work is cofunded under the SIMARGL Project – Secure Intelligent Methods for Advanced RecoGnition of malware and stegomalware, with the support of the European Commission and the Horizon 2020 Program, under Grant Agreement No. 833042. This work is cofunded under the SPARTA Project – Secure Intelligent Methods for Advanced RecoGnition of malware and stegomalware, with the support of the European Commission and the Horizon 2020 Program, under Grant Agreement No. 830892.

References

[1] D. Biswas. 5 ransomware attacks of 2021 that blew the internet, August 2021. https://analyticsindiamag.com/5-ransomware-attacks-of-2021-that-blew-the-internet/ [Online; Accessed on June 15, 2022].

[2] A. Siddiqui. Ransomware attacks hit record 300 mn in 1st half of 2021: Report, August 2021. https://thenewsmotion.com/ransomware-attacks-hit-record-300-mn-in-1st-half-of-2021-report/ [Online; Accessed on June 15, 2022].

[3] ENISA. Enisa threat landscape 2020 - list of top 15 threats, October 2020. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-list-of-top-15-threats [Online; Accessed on June 15, 2022].

[4] M. Komisarek, M. Pawlicki, M. Kowalski, A. Marzecki, R. Kozik, and M. Choras. Network intrusion detection in the wild - the orange use case in the simargl project. In *Proc. of The 16th International Conference on Availability, Reliability and Security (ARES'21), Vienna, Austria*, pages 1–7. ACM, August 2021.

[5] H. E. Poston. A brief taxonomy of intrusion detection strategies. In *Proc. of 2012 IEEE National Aerospace and Electronics Conference (NAECON'12), Dayton, OH, USA*, pages 255–263. IEEE, July 2012.

[6] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti. A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems. *IEEE Systems Journal*, 15(2):1717–1731, May 2020.

[7] G. Liu and J. Zhang. Cnid: Research of network intrusion detection based on convolutional neural network. *Discrete Dynamics in Nature and Society*, 2020(2020):1–11, May 2020.

[8] X. Li, P. Yi, W. Wei, Y. Jiang, and L. Tian. Lnnls-kh: A feature selection method for network intrusion detection. *Security and Communication Networks*, 2021:1–22, January 2021.

[9] Z. Liu, N. Su, Y. Qin, J.Lu, and X. Li. A deep random forest model on spark for network intrusion detection. *Mobile Information Systems*, 2020:1–16, December 2020.

[10] D. Yuansheng, R. Wang, and H. Juan. Real-time network intrusion detection system based on deep learning. In *Proc. of the 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS'19), Beijing, China*, pages 1–4. IEEE, March 2019.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of the 25th International Conference on Neural Information Processing Systems (NIPS'12), Lake Tahoe, Nevada, United States*, pages 197–1105. ACM, December 2012.

[12] M. Kumar and S. A. Kumar. Distributed intrusion detection system using blockchain and cloud computing infrastructure. In *Proc. of the 2020 4th International Conference on Trends in Electronics and Informatics (ICEI'20), Tirunelveli, India*, pages 248–252. IEEE, July 2020.

[13] D. Chou and M. Jiang. Data-driven network intrusion detection: A taxonomy of challenges and methods. *arXiv*, abs/2009.07352, September 2020.

[14] T. Jirsik, M. Cermak, D. Tovarnak, and P. Celeda. Toward stream-based ip flow analysis. *IEEE Communications Magazine*, 55(7):70–76, July 2017.

[15] R. Zou, T. Xu, and H. Hou. An enhanced netflow data collection system. In *Proc. of the 2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC'12), Harbin, China*, pages 508–511. IEEE, December 2012.

[16] C. Balantrapu, A. Potluri, and N. Das. A novel approach to netflow monitoring in data center networks. In *Proc. of the 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS'14), Bangalore, India*, pages 1–4. IEEE, January 2014.

[17] J. Hou, P. Fu, Z. Cao, and A. Xu. Machine learning based ddos detection through netflow analysis. In *Proc. of the 2018 IEEE Military Communications Conference (MILCOM'18), Los Angeles, CA, United States*, pages 1–6. IEEE, October 2018.

[18] M. H. Haghighat, Z. A. Foroushani, and J. Li. Sawant: Smart window based anomaly detection using netflow traffic. In *Proc. of the 2019 IEEE 19th International Conference on Communication Technology (ICCT'19), Xi'an, China*, pages 1396–1402. IEEE, October 2019.

[19] K. M. Straub, A. Sengupta, J. M. Ernst, R. W. McGwier, M. Watchorn, R. Tilley, and R. Marchany. Malware propagation in fully connected networks: A netflow-based analysis. In *Proc. of the 2016 IEEE Military Communications Conference (MILCOM'16), Baltimore, MD, USA*, pages 497–502. IEEE, November 2016.

[20] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*, 16(4):20370–2064, May 2014.

[21] R. Kozik, M. Pawlicki, and M. Choras. Cost-sensitive distributed machine learning for netflow-based botnet activity detection. *Security and Communication Networks*, 2018:8753870, December 2018.

[22] M. Komisarek, M. Pawlicki, R. Kozik, and M. Choras. Machine learning based approach to anomaly and cyberattack detection in streamed network traffic data. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 12(1):3–19, March 2021.

[23] M. Komisarek, M. Choras, R. Kozik, and M. Pawlicki. Real-time stream processing tool for detecting suspicious network patterns using machine learning. In *Proc. of The 15th International Conference on Availability, Reliability and Security (ARES 2020), Virtual Event, Ireland*, pages 1–7. ACM, August 2020.

[24] A. Pawlicka, D. Jaroszewska-Choras, M. Choras, and M. Pawlicki. Guidelines for stego/malware detection tools: Achieving gdpr compliance. *IEEE Technology and Society Magazine*, 39(4):60–70, December 2020.

[25] A. Pras, R. Sadre, A. Sperotto, T. Fioreze, D. Hausheer, and J. Schönwälder. Using netflow/ipfix for network management. *Journal of Network and Systems Management*, 17(4):482–487, December 2009.

[26] M. S. Rohmad, F. Azmat, M. Manaf, and J. A. Manan. Enhanced netflow version 9 (e-netflow v9) for network mediation: Structure, experiment and analysis. In *Proc. of the 2008 International Symposium on Information Technology (ITSim'08), Kuala Lumpur, Malaysia*, pages 1–6. IEEE, August 2008.

[27] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann. Netflow datasets for machine learning-based network intrusion detection systems. *CoRR*, 371:117–135, Nov 2020.

[28] H. Wu, S. Zhihao, and K. Wolter. Performance prediction for the apache kafka messaging system. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS'19), Zhangjiajie, China*, pages 154–161. IEEE, October 2019.

[29] J. Lee and W. Wu. How linkedin customizes apache kafka for 7 trillion messages per day, October 2019. `https://engineering.linkedin.com/blog/2019/apache-kafka-trillion-messages` [Online; Accessed on June 15, 2022].

[30] K. Darshita and M. Darshita. Paper on searching and indexing using elasticsearch. *International Journal Of Engineering And Computer Science*, 6(6):21824–21829, June 2017.

[31] N. Shah, D. Willick, and V. Mago. A framework for social media data analytics using elasticsearch and kibana. *Wireless Networks*, 28(3):1–9, December 2018.

[32] S. Salloum, R. Dautov, X. Chen, and P. X. Pengand J. Z. Huang. Big data analytics on apache spark. *International Journal of Data Science and Analytics*, 1(3):145–164, November 2016.

[33] N. Sölpük. Karl pearsons chi-square tests. *Educational Research and Reviews*, 15(9):575–580, September 2020.

[34] B. Hiraman, M. Chapte, and C. Abhijeet. A study of apache kafka in big data stream processing. In *2018 International Conference on Information , Communication, Engineering and Technology (ICICET'18), PUNE, India*, pages 1–3. IEEE, November 2018.

[35] R. T. Adek and M. Ula. A survey on the accuracy of machine learning techniques for intrusion and anomaly

detection on public data sets. In *Proc. of the 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA'20), Medan, Indonesia*, pages 19–27. IEEE, September 2020.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, November 2011.

[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16(1):321—-357, June 2002.

[38] Z. Dongyang and G. Yicheng. The comparison of lightgbm and xgboost coupling factor analysis and prediagnosis of acute liver failure. *IEEE Access*, 8:220990–221003, December 2020.

[39] A. Shahraki, M. Abbasi, and Ø. Haugen. Boosting algorithms for network intrusion detection: A comparative evaluation of real adaboost, gentle adaboost and modest adaboost. *Engineering Applications of Artificial Intelligence*, 94:103770, September 2020.

[40] M. Chora´s and M. Pawlicki. Intrusion detection approach based on optimised artificial neural network. *Neurocomputing*, 452(5):705–715, September 2021.

[41] M. Komisarek, M. Pawlicki, P. Sobonski, A. Pawlicka, R. Kozik, and M. Choras. Extending machine learning-based intrusion detection with the imputation method. In *Proc. of the in Image Processing, Pattern Recognition and Communication Systems (CORES, IP&C, ACS'21), Bydgoszcz, Poland*, volume 255 of *Lecture Notes in Networks and Systems*, pages 284–292. Springer, August 2021.

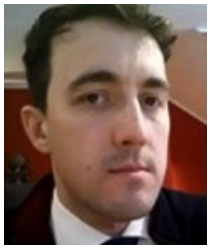
[42] M. Komisarek, M. Pawlicki, M. Kowalski, A. Marzecki, R. Kozik, and M. Choraś. Network intrusion detection in the wild-the orange use case in the simargl project. In *Proc. of The 16th International Conference on Availability, Reliability and Security (ARES'21), Vienna, Austria*, pages 1–7. ACM, August 2021.

[43] M. E. Mihailescu, D. Mihai, M. Carabas, M. Komisarek, M. Pawlicki, W. Holubowicz, and R. Kozik. The proposition and evaluation of the roedunet-simargl2021 network intrusion detection dataset. *Sensors*, 21(13):4319, June 2021.

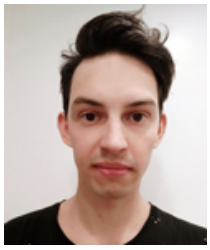
[44] M. Komisarek, M. Pawlicki, R. Kozik, W. Holubowicz, and M. Choras. How to effectively collect and process network data for intrusion detection? *Entropy*, 23(11):1532, November 2021.

Author Biography

Mikołaj Komisarek graduated with a Master's Degree in Computer Science in 2020 from the University of Technology and Life Sciences in Bydgoszcz, Poland and obtained an Engineering Degree in 2019. He currently works as a programmer and develops software in the education sector and works as a consultant in artificial intelligence at ITTI. His main interest is machine learning and deep neural networks. In addition, he is currently working on lifelong learning. He has been involved in many cybersecurity, critical infrastructures protection, and software quality projects such as H2020 SIMARGL, H2020 Infrastress, H2020 SPARTA, Q-Rapids.



Marek Pawlicki received the Ph.D. degree in computer science in 2020. He is currently an Associate Professor with the Bydgoszcz University of Science and Technology. His ongoing research investigates the novel ways of employing machine learning, data science, and granular computing in cybersecurity and anomaly detection. He has been involved in a number of international projects related to cybersecurity, critical infrastructures protection, and software quality (e.g., H2020 SIMARGL, H2020 Infrastress, and H2020 SocialTruth, H2020 SPARTA). He is an author of over 40 peer-reviewed scientific publications. His research interest includes the application of machine learning in several domains.



Mikołaj Kowalski graduated with a Master's Degree in Telecommunication in 2017, currently a PhD student in Cybersecurity on the Warsaw University of Technology. While working as a security solution architect for Orange Poland he has been involved in H2020 SIMARGL project. Main area of interest and research is network security - routing resilience and Denial of Service attacks.

Adrian Marzecki received his Ph.D. degree from the Faculty of Electronics and Information Technology of the Warsaw University of Technology in 2003. He is also a graduate of the International MBA of Warsaw University of Technology Business School with cooperation with HEC School of Management Paris, London Business School and Norwegian School of Economics and Business Administration. He is the author of over 30 publications in the fields of cybersecurity and IT/Telecommunications security management. His research interests include cybersecurity (H2020 SIMARGL project), AI and critical infrastructures protection.

Rafał Kozik Ph.D., D.Sc., Eng. obtained his Doctor of Science degree in computer science from West Pomeranian University of Technology in Szczecin in 2019. He holds a professor position at the Bydgoszcz University of Science and Technology. In 2013 he received his Ph.D. in telecommunications from University of Science and Technology (UTP) in Bydgoszcz. Since 2009, he has been involved in a number of international and national research projects related to cybersecurity, critical infrastructures protection, software quality, and data privacy (e.g. FP7 INTERSECTION, FP7 INSPIRE, FP7 CAMINO, FP7 CIPRNet, SOPAS, SECOR, H2020 Q-Rapids). He is an author of over 130 reviewed scientific publications.

Michał Choraś is a full professor of computer science. He is at the Bydgoszcz University of Science and Technology, where he is the Head of the Teleinformatics Systems Division and the PATRAS Research Group. He is also affiliated with the FernUniversität in Hagen, Germany, where he is a Project Coordinator for H2020 SIMARGL (secure intelligent methods for advanced recognition of malware and stegomalware). He is an author of over 280 reviewed scientific publications. His research interests include data science, AI, and pattern recognition in several domains, e.g., cyber security, fake news detection, software engineering, anomaly detection, correlation, biometrics, and critical infrastructures protection. He has been involved in many EU projects (e.g., SPARTA, SocialTruth, CIPRNet, Q-Rapids, and InfraStress).