# Machine Learning Based Approach to Anomaly and Cyberattack Detection in Streamed Network Traffic Data

Mikołaj Komisarek[1,3], Marek Pawlicki[1,3]*, Rafał Kozik [1,3], and Michał Choraś [2,3]
[1]ITTI Sp. z o.o.,Poznań, Poland
[2]FernUniversität in Hagen, Germany
[3]UTP University Of Science And Technology, Bydgoszcz, Poland

Received: December 22, 2020; Accepted: February 18, 2021; Published: March 31, 2021

**Abstract**

In this paper, the performance of a solution providing stream processing is evaluated, and its accuracy in the classification of suspicious flows in simulated network traffic is investigated. The concept of the solution is fully disclosed along with its initial evaluation in a real-world environment. The proposition features Apache Kafka for efficient communication among different applications, along with Elasticsearch and Kibana as storage and visualisation solutions. At the heart of the engine are machine learning algorithms implemented using the TensorFlow library, providing the cutting edge in network intrusion detection. The tool allows easy definition of streams and implementation of any machine learning algorithm.

**Keywords**: machine learning, stream processing, intrusion detection

## 1 Introduction, Context and Rationale

The development of technology becomes more and more dynamic and the amount of processed and stored data is increasing dramatically. In many situations, data such as network traffic or telephone connections needs to be verified for suspicious patterns on an ongoing basis - to provide proper recognition and detection.

Increasingly more often, popular machine learning methods are used. One of the biggest challenges is the right amount of data, its proper preparation and analysis. Advances in machine learning and data mining techniques in the Big Data domain introduce new possibilities in recognising various types of patterns in cybersecurity [16], cybersecurity of web applications [7], in critical infrastructure protection [17] and many more.

This article proposes a solution for immediate data processing and passing data through machine learning algorithms and for visualizing data delivery in real-time. The proposed solution is tested with the use of a network intrusion detection dataset, since it is now applied in the cybersecurity domain (e.g., to detect attacks, malware etc.).

The proposed solution is an extension of the solution presented by the authors in [15] - it is flexible and prepared to detect threats in network traffic in many industrial sectors. One such example is in the SATIE project, which handles the problems of airport security. The proposed solution is one of the elements of the environment that realises threat detection. Since the terrorist attack on September 11th, 2001, greater precautions for security have been taken at airports. There has also been a growing emphasis on increasing the protection of the IT infrastructure, as it plays a key role in airport management [5].

*Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 12(1):3-19, Mar. 2021
DOI:*10.22667/JOWUA.2021.03.31.003*

*Corresponding author: ITTI Sp. z o.o., Rubież 46, 61-612 Poznań, Poland, Tel: +48-61-622-6985; Email: mpawlicki@itti.com.pl, Web: https://www.itti.com.pl

The European Aviation Safety Agency (EASA) estimates that about 1000 cyberattacks target aviation systems worldwide each month [23]. For a better presentation of the problem, the example from June 2015 can be used, where a Distributed Denial of Service (DDoS) attack on the Warsaw airport resulted in the cancellation of 22 flights and the grounding of about 1500 passengers [4].

The systems present at airports are not the only ones threatened by cyberattacks. The avionics systems are another area that needs cyberprotection. Some of the assets that can potentially be targeted by cyberattackers are, for example, the internal WiFi networks, systems that can be used to display false or alarming messages to passengers etc.

The main contribution of this work is the presentation of an innovative engine that allows to define the entire stream processing pipeline - from creating access to data via a stream (or loading a regular file) to the visualisation of the results of Machine Learning algorithms. The tool supports data scientists by allowing them to skip tedious and time-consuming activities, like the configuration of additional dependencies such as database connection or stream settings.

The article is structured as follows: In the first part, an overview of existing solutions on the market is presented. Then a novel solution, its proposed architecture, and the main assumptions and technical aspects of the engine are offered. Section 4 presents the results and assessment of the engine. The document ends with conclusions and plans for the future.

# 2 Related Work

## 2.1 Intrusion Detection Systems

Intrusion Detection Systems (IDSs) in the Critical Infrastructure (CI) networks are key services to protect resources and to alert against potential threats. The steady rise of new types of cyberattacks also caused the necessity for both continuous development of new defensive mechanisms and applying improvements to the existing ones. Multiple examples of defensive solutions for CI protection can be found in the literature. The authors of [30] use machine learning to classify and detect potential attacks. The authors adapted well-known algorithms such as: Decision Trees, Ripple Rule, Random Forest, and Bayesian networks, and split the system into smaller components. Data collection and preliminary preparation of data are executed in the sub-component called ”Preprocessor part”. Then, the pre-processed data directly feeds the classification and machine learning algorithms, while the final result is passed on to the last component called “Protection part”, where a decision on blocking a given IP address or port is made if an attack was detected. The system from [10] starts with a connection to the Ethernet and thanks to the packet sniffer library, network traffic is extracted from it. For the detection of the anomaly, the authors used the Weka engine, which is also written in Java and which is used for the machine learning process. The authors emphasise that they have used a small number of features. They used 12 features discussed in the article, which are effective at detecting and classifying 17 different attack types, and differentiating them from a normal traffic.

Another example of an IDS solution is the system presented in the article [1]. The authors assume that the purpose of their work is to build an IDS which can automatically classify and allow for blocking certain requests and activities by the online users. For this purpose, they use an unsupervised type of deep learning. They also indicate the use of an auto-encoder.

It should be emphasized that the authors of the article work on the NSL-KDD [26] dataset. The presented system has a processing and data preparation layer consisting of the following steps. The first one is to change the text parameters to the numerical format, to normalize all the features, and finally to change the number of labels from 39 to only four. After preparing the data, as mentioned, the authors apply deep teaching with multi-class classification based on auto-encoders. The model uses two layers of trained auto-encoders. In the model learning process, all 122 attributes are used. The

training process is done through batch processing. Finally, the authors show the obtained results. The lowest score was obtained by the J48 tree classifier algorithm and the highest by the Random Forest. The logistic regression gets an accuracy of 82.2% for training and 72.5% for testing. The proposed approach outperforms all the classical approaches with an accuracy of 99% for training and 91.28% for the testing phase.

Another example is in the publication [2], in which a secure framework for protection of remote healthcare systems is proposed. The goal of the authors was to propose a tool able to protect the data against common network attacks, including the Denial of Service (DoS) and U2R attacks. An intrusion detection system offered for this purpose uses a Support Vector Machine (SVM). According to the authors, the evaluation parameters of the layered architecture of the proposed IDS proved the efficiency of the framework. In [29], a hybrid model for an anomaly-based intrusion detection system using a machine learning approach is proposed for the Supervisory Control and Data Acquisition (SCADA) networks protection. The authors present a robust hybrid model for anomaly-based intrusion detection and a feature selection model with the goal to remove redundant and irrelevant features that can negatively impact the modelling power and reduce predictive accuracy. The experimental results show benefits in terms of reducing the time and computational complexity, as well as improved accuracy and detection rate.

In [24], a comprehensive analysis of machine and Deep Learning (DL) solutions for IDSs is presented in the context of the CI involving wireless sensor networks (WSNs). A Restricted Boltzmann machine-based clustered IDS (RBC-IDS) was examined as a deep learning-based IDS methodology for monitoring CI. The performance of the proposed RBC-IDS approach has been compared to IDS based on adaptive machine learning. On the other hand, the authors of [10] propose a framework for designing resilient distributed IDS tailored to large-scale CI ecosystems. The presented framework uses a risk assessment methodology to identify and assess the criticality of communication flows that have to be monitored. As a result, the authors propose an optimal distributed intrusion detection approach (including optimal positioning of detection devices) by delivering a K-resilient infrastructure in which each flow is resilient to a maximum of K path failures of communication links.

The focus of the solutions presented above is mainly on the detection of anomalies in the network traffic. However, for the effective detection of threats in real-world environments, a number of open issues remains. The effectiveness of the solution under the real-time heavy network traffic loads needs to be assessed. The issue of connecting one system to multiple servers simultaneously and collecting and analysing the traffic has to be addressed. Finally, the use in production environment, i.e., working with live data, should be demonstrated.

Those pitfalls are addressed in this work. To deal with those problems, the proposed solution performs most of its work on data streams.

The amount of processed and stored data is increasing dramatically. In many situations, data, such as network traffic or telephone connections, needs to be analysed for suspicious patterns in real-time. Advances in machine learning and data mining techniques, especially in the Big Data domain, introduce new opportunities for the recognition of various types of patterns. One of the biggest challenges in this area is the collection of a sufficient amount of relevant data. This step is then followed by proper pre-processing and analysis.

## 2.2 Working with streams of data

Working with streams of data creates a number of advantages. In some domains, data intrinsically has the form of a stream. Real-time processing allows to immediately respond to detected patterns. Processing data streams also poses a myriad of challenges to the current (ML) methods, making the difficulties of ML even more pronounced - especially in the disciplines where the learning examples are scarce, the data

distribution lacks adequate balance, where data is often missing, and many other challenges. Data Stream Processing Systems (DSPS) aim to fuse batch and stream processing. Exploitation of those capabilities allows to apply machine learning components trained on batch data to information flowing in the form of streams. A variety of data handling use-cases gave birth to a range of solutions. The solutions, in turn, differ from one to another at multiple levels:

- The way they ingest data, the way they process data.
- How the solution handles the storage of data.
- How it manages resources.
- What its output does. [13]

An in-depth analysis and comparison of the solutions such as Apache Spark 4, Storm 5, Flink 6 and others are provided in [22] [31] while a benchmarking method is provided in [6] and in [14].

Applying the machine learning algorithms allows for finding specific patterns and matching them to potential problems related to business activities. The Big Data market offers a variety of tools that can be used for such purposes. Solutions such as Apache Spark, Hadoop, Storm, Hive, Flink, Heron, Presto, Samza, Kudu and a range of other frameworks are worth mentioning. The choice of a particular tool depends primarily on its application and organisational context, due to the fact that each of those tools has its advantages and disadvantages. The main differences can be considered in terms of delays, performance and efficiency, support for multi-task approach, data storage and processing mechanisms, and many others. For example, Hadoop can be efficient for any large-scale batch processing that does not require immediacy. On the other hand, Spark, Storm or Samza have very low latency and can be considered as a reasonable choice for real-time processing.

## 2.3 Stream processing for IDS

A few existing implementations of stream processing architectures for IDS with the use of the Big Data approach can be found in the scientific publications. The combination of Apache Kafka [12] proposed for data ingestion, Spark for stream processing, and Spark UI used for visualisation is discussed in [27]. The tests are performed on the UNSW-NB15 dataset [20]. The main conclusion is that the interval batch size has an impact on the efficiency of Apache Spark. The same dataset was used in [3], where the authors analysed the performance of the four well-established machine learning algorithms implemented in Spark MLlib, and compared them based on accuracy, build time and prediction time measurements. The real-time processing was out of the scope in this work; however, the authors emphasised the advantages of Random Forest for this kind of task.

In [19], the authors apply a streaming version of the KMeans algorithm for anomaly detection. The tests are performed on the long-established KDD'99 dataset. However, the obtained results (about 76% of accuracy) are lower than the state-of-the-art batch methods on the same dataset.

In [21], the authors handle class imbalance by oversampling the training data in the context of Deep Learning for IDS in stream processing. The authors were focused on finding anomalies in streams by analysing the correlation between the time of the last event and the properties of the current event. The estimation was performed sequentially in an adaptive fashion and then augmented by anomaly detection. The experiments were conducted on the LANL7 NetFlow dataset [28]. The proposed method achieves results comparable to or better than the state-of-the-art anomaly detectors in a simulated environment. However, the authors do not disclose the specific Big Data frameworks utilised for this research.

# 3 The proposed solution for network pattern recognition and anomaly detection

## 3.1 General description

Work described in the literature often does not address the aspect of effectiveness under real-time heavy network traffic loads and the issue of connecting one system to multiple servers simultaneously, collecting traffic.

The proposed solution utilises Netflow protocols V5 and V9 for anomaly detection based on the delivered traffic. In addition, the focus of this work is also on the ability to spread the tasks over multiple servers and provide real-time streaming. Full integration with the Elasticsearch database and the Kibana tool is also provided. As such, the foreseen benefits of using the proposed solution are the following:

- Increased reliability of data delivery.
- Real-time intrusion detection
- Increased situational awareness, through integration with the Correlation Engine.
- Easy data browsing in the Elasticsearch database and visualization using Kibana.

While the benefits are clear, when using solutions such as the one proposed, one needs to be aware of the following potential difficulties:

- The model needs to be trained on network-specific data first to get the best results.
- Although the acquisition of labelled training datasets can be automated to an extent, achieving best results usually involves a certain level of human involvement in the labelling process.
- If large changes are introduced into the network traffic mix, e.g., by introducing new services, anomaly detection systems need to be treated with caution in order not to produce irrelevant alerts.

## 3.2 The choice of the appropriate network flow format

During the software development and testing phase, different network flow data formats were considered. Monitoring of network traffic is possible using the standards such as NetFlow or sFlow. These flow-based standards provide a metadata-based view of activity on the network. sFlow provides an outline of network traffic, while reducing the processing load on the network elements. This comes at the expense of its accuracy. On the other hand, NetFlow usually requires more processing power of the network elements and can by itself consume up to about 1% of the monitored traffic bandwidth. Still, NetFlow is capable of processing as much as 100% of the network packets to produce the final information on network flows. This makes NetFlow the preferred choice when considering the flow information for security and audit purposes. In contrast, sFlow by default analyses only 1% of the network traffic, making this protocol ineffective in relation to detecting anomalies in the network traffic. NetFlow data provides a more detailed view of the bandwidth usage and network traffic than other monitoring solutions (such as Simple Network Management Protocol - SNMP).

Following the above analysis, it was decided to base the process of anomaly detection on the NetFlow protocol. NetFlow can be defined as a schema for collecting, aggregating and logging the data about the network traffic. NetFlow collectors can be deployed as either hardware or software-based probes. The tool that is used to collect the traffic from the machines in the experimental environment is an open-source solution available for the Unix environment, called fProbe.

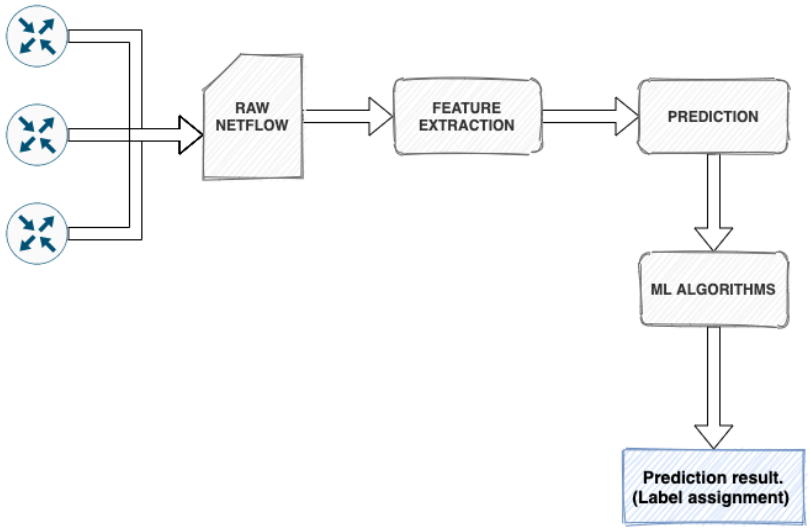



Figure 1: Outline drawing showing movement flow, feature extraction and prediction process

The process starts from the moment the NetFlow data is delivered from the probe to the collector – i.e., dedicated software or hardware that collects information from probes. A software-based collector is used, which can collect and export NetFlow flows generated by gateways, switches, border routers or any other device that can export flow data in NetFlow v5/v9. The collector is connected to the streaming tool, i.e., Apache Kafka. The collector then allows to manipulate both the input and output schema of the NetFlow frame. The redundant parameters that were not needed in the process of machine learning or prediction can be omitted immediately, resulting in less data passing through the stream and lowering the overall system load.

### 3.3 Technical description of the proposed solution

The proposed solution is a tool that offers a wide range of modular solutions to create a complex machine learning environment. The diagram of the proposed solution can be seen in Figure 4. The architecture of the proposed solution starts with data delivery. The main focus is on the network traffic analysis.

The collected network flow is directed directly to the collector, which provides frames on one of Kafka topics. This solution allows collecting network flow from multiple devices connecting to a specified network, and to analyse the flows in real-time without delay.

After implementing the mechanisms for collecting and transmitting NetFlow frames to a specific Kafka topic, the next important step is to prepare the data for feature extraction. The features are extracted from grouped raw packets. This is done each time when new data arrives directly at the Kafka stream. The data is grouped within a specified time frame, allowing to provide a feature vector for each, e.g., 1-minute window. This approach allows to extract more features and more data is processed in each batch.

During the development of the solution, one of the aspects that had the most importance was processing large amounts of data delivered by the stream. In order to avoid problems with performance and scalability of the module executing the reception and pre-processing of the incoming data, it is developed in the Scala language. It is based on one of the most popular open-source libraries - Apache Spark. This

approach allows for the utilisation of all of the features and extensions offered by the Spark platform.

**Algorithm 1:** Algorithm for anomaly detection process

```
Result: Result entire subprocesses
Read: ApacheKafka;
while ApacheKafka IS running do
    Read: stream;
    if stream NOT empty then
        broadcastToAllConnectedClients(netflowFrame);
        sendFrameToElasticsearch(netflowFrame);
    end
end
Read: NetflowCollector;
while NetflowCollector IS running do
    Read: netflowFrame;
    if netflowFrame != empty then
        sendFrameToTheKafkaTopic(netflowFrame);
    end
end
Read: SolutionSpark;
while SolutionSpark IS running do
    Read: streamKafka, windowTimeKafka;
    if streamKafka NOT empty AND windowTimeKafka > 60s then
        Read: All Netflow from last 60 seconds ;
        Group: Frames By Source IP Address And Port ;
        Count: all features (Packet In/Out Byte In/Out );
        Run: Prediction Model ;
        if prediction IS Normal then
            Set 0 in label field;
        else
            Set 1 in label field;
        end
        Result: Send to Kafka Topic Frame with label;
    end
end
Read: SolutionTensoflow;
while SolutionTensoflow IS running do
    Read: streamKafka;
    if streamKafka NOT empty then
        Read: New frame netflow from SolutionSpark Component ;
        Run: Neural Network to classification type attack
    end
end
```

This also allows to extend the capabilities of the component with additional new modules and with easy configuration of the entire environment.

This facilitates the scalability of the solution, which has the ability of utilising multiple spark workers and splitting tasks into different servers. Other benefits are the fault tolerance capabilities because of

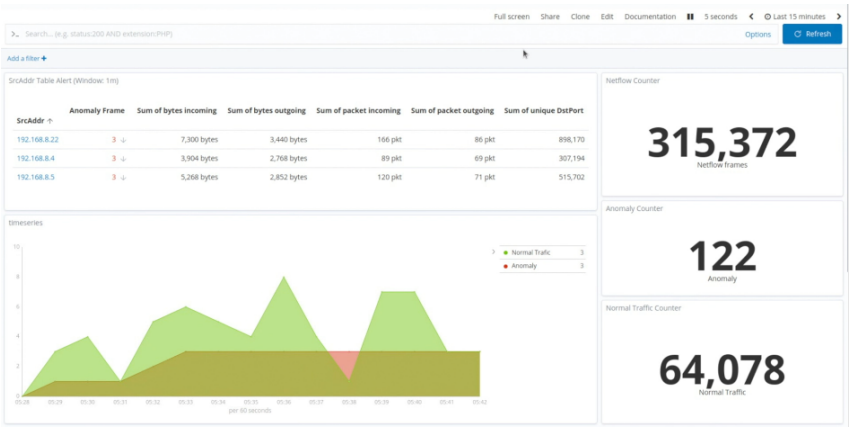



Figure 2: Graph visualizations in Kibana

immutable primary abstraction named Resilient Distributed Datasets (RDD).

This component is marked in Figure 4 as 'the solution' (SPARK). Within the whole solution, another important module is the one responsible for the process of feature extraction, machine learning and labelling. This module communicates directly with the data delivery module. Libraries such as Tensorflow 2 and Keras were utilised. The data delivery is realized by using a distributed streaming platform - Apache Kafka. This means that the software supports receiving, validating, transforming and sending messages between applications. The main advantages of Kafka are its reliability, efficiency and ability to work in a distributed environment. Apache Kafka has been integrated with the solution to deliver data from various sources to the preprocessors that process and prepare data for machine learning. After these operations, the system can send the output back to the Kafka theme or to the next task indicated in the start point configuration.

The whole environment is integrated into the Elasticsearch database and all the received and processed data goes directly to different indexes. Elasticsearch is a distributed, open-source search and analytics engine for all types of data, which allows for collecting large amounts of existing data from a database [11]. Using this solution opens up the possibility to visualise, easily search and browse the records using the Kibana tool [27]. The NetFlow data schema itself is extensive and contains a lot of information ranging from the source address to the number of bytes and packets transferred. Therefore, to provide a more human-readable format, the data contained in the indices is also visualised on a series of graphs. Figure 2 presents several of such graph visualisations in Kibana: e.g., the total number of the NetFlow frames (how many are in the system, number of anomalies detected) and the amount of normal uninfected network traffic.

Additionally, the time charts are grouped in 60 seconds' intervals and show how many attacks were detected in a specific interval along with a table of what IP addresses were involved in the network traffic for the last few minutes. Figure 3 presents the visualisations which present anomaly detection in the network traffic. Grouping is applied by source addresses. The number of bytes and packets that were sent as part of host communication and the total amount of normal and infected traffic from a given address are displayed, providing the operator a quick overview of what addresses are suspicious.

All these components comprise the basic environment of the proposed solution. The structure of the solution has been designed so that it is easy to add new modules or extend the existing ones. The whole example environment implementation is shown in Figure 4, the algorithm of the approach can be found in Algorithm 1.

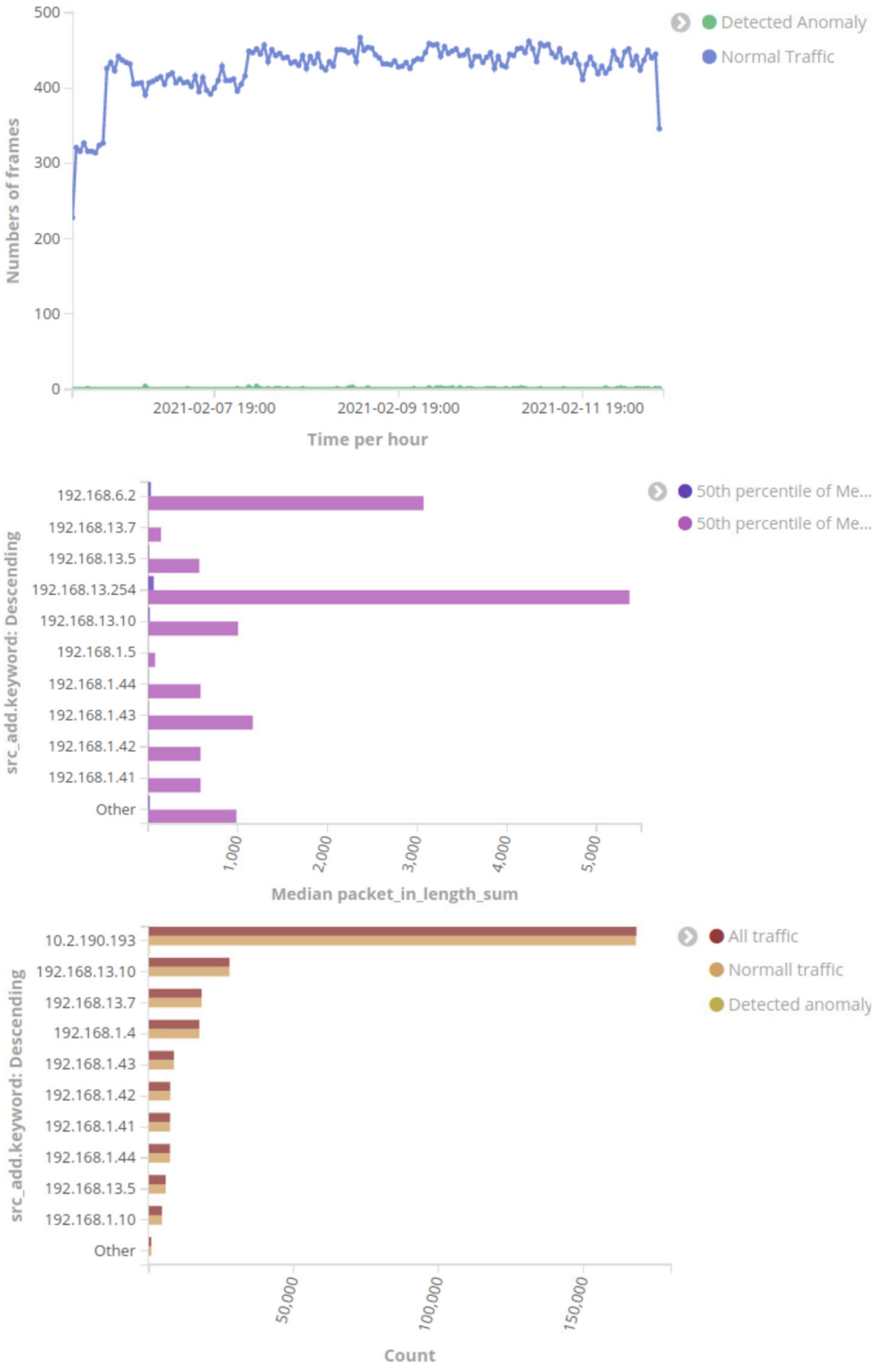



Figure 3: Visualisations of anomaly detection in the network traffic

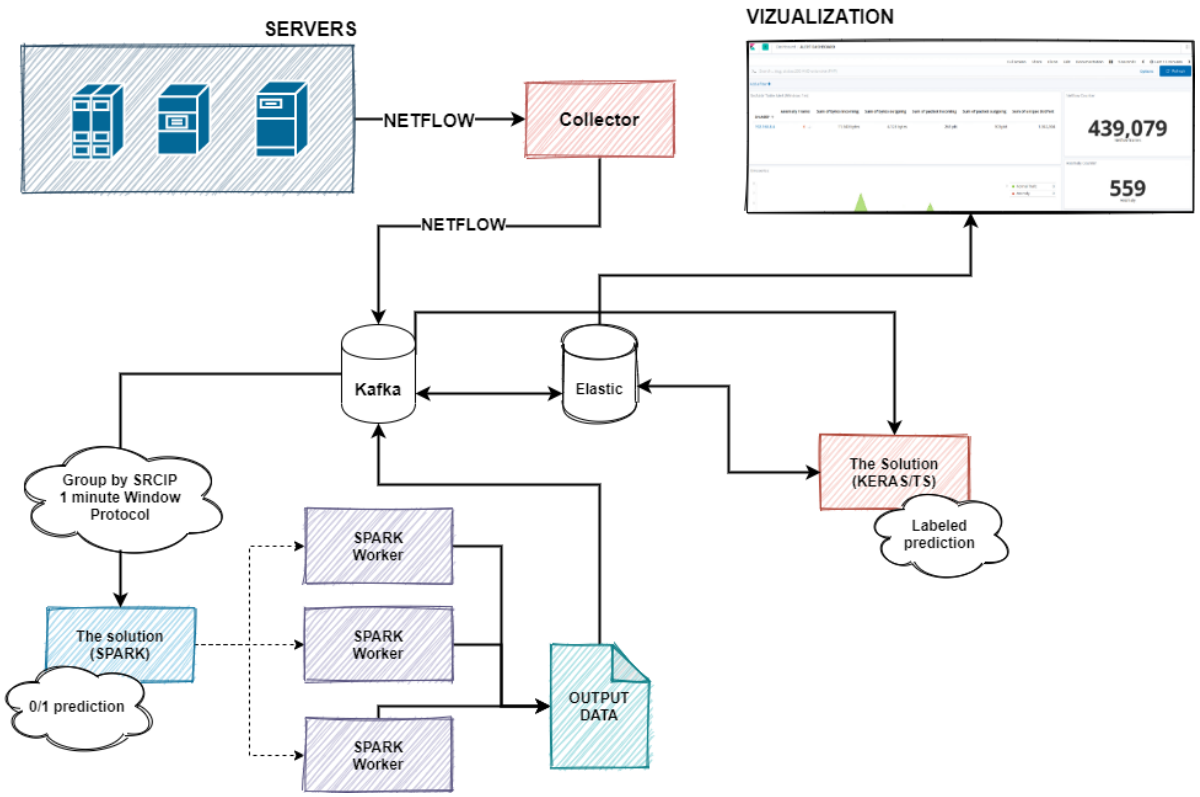



Figure 4: The technology stack and architectural view of the tool deployment

# 4 Experiments and results

## 4.1 Experimental setup

As part of the project activities, several tests were performed to present the capabilities and efficiency of the proposed solution, which will be explained in this section. Moreover, the use of the model with all the engineered features obtained during the solution's software tests is presented. The following tests are presented in this section:

- Test scenario 1: Detection of anomalies in the CTU-13 benchmark dataset [9]
- Test scenario 2: Detection of anomalies in the SATIE Network

During the verification process of our solution, the following factors were taken into account:

- The tests were conducted to confirm the reliability and operation readiness of the proposed solution.
- The environment in which the measurement was made consisted of three Kafka brokers and three Elasticsearch nodes with seven Apache Spark workers.
- For the experiment, the files were combined into one large dataset and split with the ratio of 70 percent training data, 30 percent testing data.

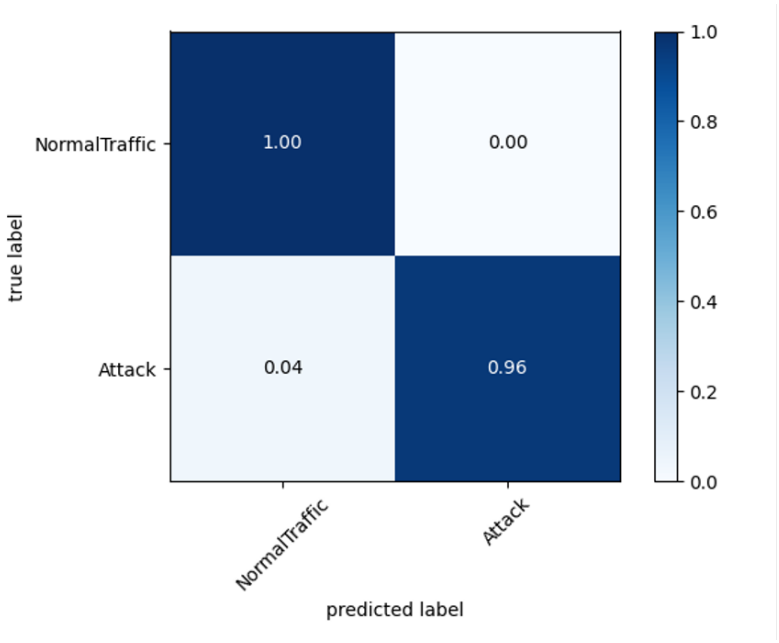



Figure 5: Confusion matrix for labelling prediction (Normal Attack or Anomaly, CTU-13 dataset)

### 4.2 Test scenario 1: Detection of anomalies in the CTU-13 Dataset

During the experiments, the accuracy and performance of detecting suspicious network activities using the Random Forest algorithm as part of the solution was verified. The machine learning algorithm was trained on the CTU-13 dataset, which contains botnet traffic captured at the CTU University, Czech Republic. The dataset is intended to have a large capture of real botnet traffic mixed with normal traffic and background traffic. The whole dataset consists of 13 files showing actual network traffic and contains about 20,643,076 frames.

The structure of the input data is presented and described in [9]. The process of subsequent preparation and processing of incoming data is based on aggregating frames into one-minute windows and additionally grouping them by port number, source address and protocol. The use of such a method enables engineering more features useful for machine learning.

After the model training process, the remaining files from the CTU-13 set were used to verify the anomaly detection capability. Figure 5 presents a confusion matrix with the results. It shows that 96% of the anomalies were detected correctly with no false positives.

### 4.3 Test scenario 2: Anomaly detection in the real-world SATIE Network

This scenario focused on checking the performance and reliability of an algorithm trained specifically for the SATIE Environment. The aim of this scenario was to validate the algorithm and increase its performance metrics. This scenario included a pre-run phase during which the network was observed, data was gathered, annotated, and used to create the model. This also included indexing clean traffic and the marking of suspicious packets.

In the pre-run phase, about 400 thousand frames of baseline traffic were collected. In the next step, about 100 thousand frames of mixed, i.e., infected and normal traffic, were gathered. Then, the data was labelled accordingly, and the ML algorithm was trained.

The execution of individual scenarios took place at different time intervals and individual attacks did

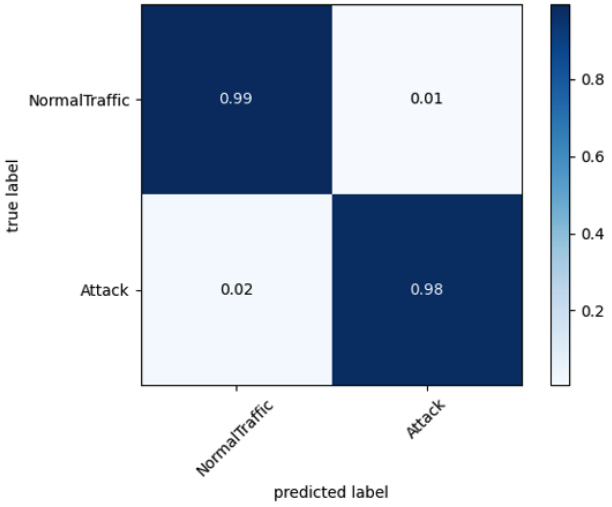



Figure 6: Confusion matrix for labelling prediction (Normal Traffic or Attack, SATIE dataset)

not overlap in time. The attacks were filtered and marked using the Kibana tool. When the collection process was completed, the created files were balanced. Figure 6 presents confusion matrix for labelling prediction (Normal Traffic or Attack). By training the model on data coming directly from the target network, performance comparable to the baseline from the test scenario 1 was achieved, i.e., 98 % of attacks were marked successfully, with around 1% or less of False Positives.

The model performed binary classification (normal/anomaly).

A new batch of experiments included a model capable of multinominal classification, using the same dataset with more detailed labels - the attack types that generated the anomaly. As always, while working with real-world data, the aspects of privacy are at stake [25]. The names of the particular attacks the CI security component has been tested for have been withheld due to security reasons.

This test scenario presents the results for this newly added function of the proposed solution.

The multinominal classification module itself is written in Python, while the data analysis and machine learning layer are implemented with Keras and TensorFlow. Using these industry standards allows the solution to implement model optimisation procedures like the ones described in [8]. The whole component is integrated with Kafka and the Elasticsearch database. The network traffic data was recorded using the Elasticsearch database, then marked and checked manually. Many ML algorithms experience problems when the distribution of classes in the dataset is not balanced [18]. In this case, the dataset is balanced using the Random Oversampling approach. Random oversampling involves randomly selecting examples from the minority class with replacement and adding them to the training dataset.

After the process of preparing and balancing the dataset, a model based on the Random Forest algorithm was created. In this scenario, the training set constituted 80% of the dataset, and the remaining 20% became the test set. The results of these experiments are shown in Figure 7 and Figure 8. Figure 7 presents the confusion matrix. The number of false positives is around or less than 1%. Figure 8 gives a wider picture of the results by adding the information such as precision, recall, f1-score. All the attacks were detected properly as anomalous traffic, with 98-99% accuracy in the prediction of the attack type.

Currently, this work is a part of the SATIE (Security of Air Transport Infrastructure of Europe) project, co-funded by the European Commission in Horizon 2020 research and innovation programme, where we work on extending the detection capabilities of our solution.

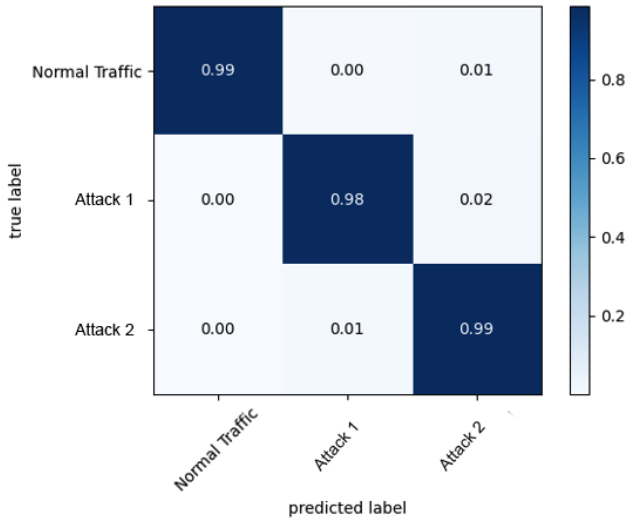



Figure 7: Confusion matrix for labelling predictions, SATIE dataset

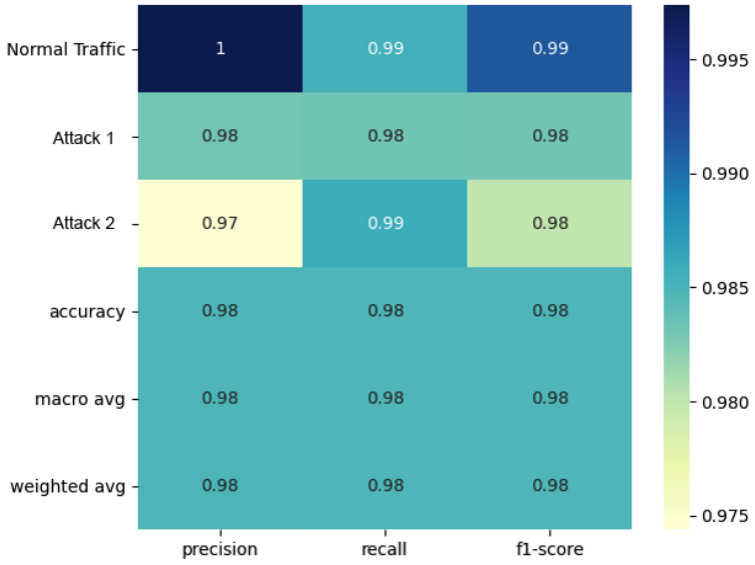



Figure 8: Classification report, SATIE dataset

## 5 Conclusion

This article presents a novel approach to the concept of a stream processing intrusion detection solution that allows the integration of a variety of components. It is an extension of our previous work [15]. The solution is capable of creating a stream in a Kafka topic and implementing one of many machine learning algorithms, providing suitable alerts or views/visualisation, and then transferring the results to a database. Using NetFlow-based features has a number of advantages, such as reducing processing load necessary to perform network intrusion detection. NetFlow also provides greater privacy than deep packet inspection. NetFlow also provides full coverage of the network packets. Utilising Apache Kafka

provides reliability, efficiency and opens up the ability to work in a distributed environment, facilitating communication among different applications. Storing the received data with the Elasticsearch solution enables immediate searchability and analysis of enormous volumes of data. Another advantage of Elasticsearch is the seamless integration with Kibana, which in turn is an amazing tool for visualisation and providing situational awareness to the cybersecurity operatives. The integration with modules written using the TensorFlow library allows to equip the solution with state-of-the-art advancements in machine learning, including deep learning. This provides the cutting edge in network intrusion detection. For future work, developments in the area of network training from live streams are foreseen. Moreover, further improvements towards integrating and expanding the concepts from the domain of machine learning are planned.

## Acknowledgements

This work is partially funded under SATIE project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 832969. This work has been also supported by 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.

## References

[1] B. Alsughayyir, A. M. Qamar, and R. Khan. Developing a network attack detection system using deep learning. In *Proc. of the 2019 International Conference on Computer and Information Sciences (ICCIS'19), Sakaka, Saudi Arabia*, pages 1–5. IEEE, April 2019.

[2] M. Begli, F. Derakhshan, and H. Karimipour. A layered intrusion detection system for critical infrastructure using machine learning. In *Proc. of the 2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE'19), Oshawa, Ontario, Canada*, pages 120–124. IEEE, August 2019.

[3] M. Belouch, S. El Hadaj, and M. Idhammad. Performance evaluation of intrusion detection based on machine learning using apache spark. *Procedia Computer Science*, 127:1–6, January 2018.

[4] C. Brook. Polish planes grounded after airline hit with ddos attack, June 2015. `https://threatpost.com/polish-planes-grounded-after-airline-hit-with-ddos-attack/113412/` [Online; accessed on March 22, 2021].

[5] C. Brook. How airport security has changed since 9/11, September 2016. `https://www.cntraveler.com/story/how-airport-security-has-changed-since-september-11` [Online; accessed on March 22, 2021].

[6] S. Chintapalli, D. Dagit, B. Evans, R. Farivar, T. Graves, M. Holderbaugh, Z. Liu, K. Nusbaum, K. Patil, B. Peng, and P. Poulosky. Benchmarking streaming computation engines: Storm, flink and spark streaming. In *Proc. of the 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPS Workshops'16), Chicago, Illinois, USA*, pages 1789–1792. IEEE, May 2016.

[7] M. Choraś and R. Kozik. Machine learning techniques applied to detect cyber attacks on web applications. *Logic Journal of the IGPL*, 23(1):45–56, February 2015.

[8] M. Choraś and M. Pawlicki. Intrusion detection approach based on optimised artificial neural network. *Neurocomputing*, December 2020.

[9] S. García, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123, September 2014.

[10] B. Genge, P. Haller, and I. Kiss. A framework for designing resilient distributed intrusion detection systems for critical infrastructures. *International Journal of Critical Infrastructure Protection*, 15:3–11, December 2016.

[11] S. Gupta and R. Rani. A comparative study of elasticsearch and couchdb document oriented databases. In *Proc. of the 2016 International Conference on Inventive Computation Technologies (ICICT'16), Coimbatore, India*, volume 1, pages 1–4. IEEE, August 2016.

[12] B. R. Hiraman, C. Viresh M., and K. Abhijeet C. A study of apache kafka in big data stream processing. In *Proc. of the 2018 International Conference on Information, Communication, Engineering and Technology (ICICET'18), Pune, India*, pages 1–3. IEEE, August 2018.

[13] H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine, and S. Khan. A survey of distributed data stream processing frameworks. *IEEE Access*, 7:154300–154316, October 2019.

[14] J. Karimov, T. Rabl, A. Katsifodimos, R. Samarev, H. Heiskanen, and V. Markl. Benchmarking distributed stream data processing systems. In *Proc. of the 34th IEEE International Conference on Data Engineering (ICDE'18), Paris, France*, pages 1507–1518. IEEE, April 2018.

[15] M. Komisarek, M. Choraś, 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'20), Virtual Event, Ireland*, pages 1–7. ACM, August 2020.

[16] R. Kozik and M. Choraś. Protecting the application layer in the public domain with machine learning methods. *Logic Journal of the IGPL*, 27(2):149–159, March 2019.

[17] R. Kozik, M. Choraś, A. Flizikowski, M. Theocharidou, V. Rosato, and E. Rome. Advanced services for critical infrastructures protection. *Journal of Ambient Intelligence and Humanized Computing*, 6(6):783–795, December 2015.

[18] P. Ksieniewicz and M. Woźniak. Imbalanced data classification based on feature selection techniques. In *Proc. of the 19th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'2018), Madrid, Spain*, volume 11315 of *Lecture Notes in Computer Science*, pages 296–303. Springer International Publishing, November 2018.

[19] S. N. Lighari and D. muhammed Akbar Hussain. The efficient way of detecting anomalies in large scale streaming data. *University of Sindh Journal of Information and Communication Technology*, 2(3):156–161, July 2018.

[20] N. Moustafa and J. Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *Proc. of the 2015 Military Communications and Information Systems Conference (MilCIS'15), Canberra, Australian Capital Territory, Australia*, pages 1–6. IEEE, November 2015.

[21] H. A. Najada, I. Mahgoub, and I. Mohammed. Cyber intrusion prediction and taxonomy system using deep learning and distributed big data processing. In *Proc. of the IEEE Symposium Series on Computational Intelligence (SSCI'18), Bangalore, India*, pages 631–638. IEEE, November 2018.

[22] H. Nasiri, S. Nasehi, and M. Goudarzi. Evaluation of distributed stream processing frameworks for iot applications in smart cities. *Journal of Big Data*, 6(1):1–24, December 2019.

[23] G. Olano. Aviation sector faces over 1,000 hacking attempts monthly, July 2016. `https://www.insurancebusinessmag.com/asia/news/breaking-news/aviation-sector-faces-over-1000-hacking-attempts-monthly-49541.aspx` [Online; accessed on March 22, 2021].

[24] S. Otoum, B. Kantarci, and H. T. Mouftah. On the feasibility of deep learning in sensor network intrusion detection. *IEEE Networking Letters*, 1(2):68–71, February 2019.

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

[26] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *Proc. of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, (CISDA'09), Ottawa, Ontario, Canada*, pages 1–6. IEEE, July 2009.

[27] M. T. Tun, D. E. Nyaung, and M. P. Phyu. Performance evaluation of intrusion detection streaming transactions using apache kafka and spark streaming. In *Proc. of the 2019 International Conference on Advanced Information Technologies (ICAIT'19), Yangon, Myanmar*, pages 25–30. IEEE, November 2019.

[28] M. J. M. Turcotte, A. D. Kent, and C. Hash. Unified host and network data set. In N. Heard, editor, *Data Science for Cyber-Security*, volume 3, chapter 1, pages 1–22. World Scientific, November 2018.

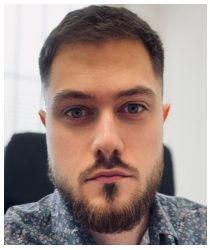
[29] I. Ullah and Q. H. Mahmoud. A hybrid model for anomaly-based intrusion detection in SCADA networks. In *Proc. of the 2017 IEEE International Conference on Big Data, (BigData'17), Boston, Massachusetts, USA*, pages 2160–2167. IEEE, December 2017.

[30] N. Wattanapongsakorn, S. Srakaew, E. Wonghirunsombat, C. Sribavonmongkol, T. Junhom, P. Jongsubsook, and C. Charnsripinyo. A practical network-based intrusion detection and prevention system. In *Proc. of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'12), Liverpool, UK*, pages 209–214. IEEE, June 2012.

[31] P. Zapletal. Comparison of apache stream processing frameworks: Part 1, 2018. `https://cloud.tencent.com/developer/article/1088152` [Online; accessed on March 22, 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. He is currently an Associate Professor with the UTP University of Science and Technology, Bydgoszcz. 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). He is the author of over 25 peer-reviewed scientific publications. His research interest includes the application of machine learning in several domains.

**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 professor position at the UTP University of Science and Technology in Bydgoszcz. 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 100 reviewed scientific publications.

**Michał Choraś** currently holds a professorship position with the UTP University of Science and Technology, Bydgoszcz, where he is the Head of the Teleinformatics Systems Division and the PATRAS Research Group. He is affiliated also with 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 the author of over 260 reviewed scientific publications. His research interests include data science, AI, and pattern recognition in several domains, e.g., cyber security, image processing, software engineering, prediction, anomaly detection, correlation, biometrics, and critical infrastructures protection. He has been involved in many EU projects (e.g., SocialTruth, CIPRNet, Q-Rapids, and InfraStress).