Design and Implementation of an Intelligent and Model-based Intrusion Detection System for IoT Networks

Peter Vogl, Sergei Weber, Julian Graf, Katrin Neubauer, Rudolf Hackenberg
Ostbayerische Technische Hochschule, Regensburg, Germany
Dept. Computer Science and Mathematics
email:{peter.vogl, sergei.weber, julian.graf, katrin1.neubauer, rudolf.hackenberg}@oth-regensburg.de

***Abstract*—The ongoing digitization and digitalization entails the increasing risk of privacy breaches through cyber attacks. Internet of Things (IoT) environments often contain devices monitoring sensitive data such as vital signs, movement or surveillance data. Unfortunately, many of these devices provide limited security features. The purpose of this paper is to investigate how artificial intelligence and static analysis can be implemented in practice-oriented intelligent Intrusion Detection Systems to monitor IoT networks. In addition, the question of how static and dynamic methods can be developed and combined to improve network attack detection is discussed. The implementation concept is based on a layer-based architecture with a modular deployment of classical security analysis and modern artificial intelligent methods. To extract important features from the IoT network data a time-based approach has been developed. Combined with network metadata these features enhance the performance of the artificial intelligence driven anomaly detection and attack classification. The paper demonstrates that artificial intelligence and static analysis methods can be combined in an intelligent Intrusion Detection System to improve the security of IoT environments.**

***Keywords*—Intrusion Detection; Artificial Intelligence; Machine Learning; Network Security; Internet of Things.**

I. Introduction

Demographic change is a particular challenge worldwide. One consequence of demographic change is an aging population. However, because of the now higher life expectancy, the risk of illness for each older person is also increasing [1]. For this reason, measures must be taken to enable the aging population to live more safely.

To imporve safetiness Ambient Assisted Living (AAL) is used. AAL refers to all concepts, products and services that have the goal of increasing the quality of life, especially in old age, through new technologies in everyday life [2]. The intelligent Intrusion Detection System (iIDS) described in this paper is part of a publicly funded research project Secure Gateway Service for Ambient Assisted Living (SEGAL). Within SEGAL, a lot of sensitive information such as heart rates, blood sugar or blood pressure are measured. These are needed so that people in need of care can live in their familiar environment for as long as possible. This is made possible by developing an AAL service for SEGAL. Within the AAL service, the recorded data from Internet of Things (IoT) devices are sent via the smart meter gateway to the AAL data management of the responsible control center from the AAL-Hub. The smart meter gateway is a secure communication channel, as a certificated communication path is used for the transmission of the recorded data [3]. However, the exchange of data between the IoT devices and the AAL hub is not necessarily to be considered secure and can be seen as a target for attacks. Therefore, there is the need to secure the communication between the IoT devices and the back end system, so that there is no theft and manipulation of the transmitted data. In this case, the iIDS is used to protect the sensitive recorded data, as it is intended to detect possible attacks. The individual layers of the iIDS are designed to be easily integrated into cloud structures. This allows cloud to take advantage of flexibility and efficiency to monitor network security optimally. Therefore, security services can be scaled depending on the circumstances [4]. In addition, the cloud offers the possibility to improve new innovative Artificial Intelligence (AI) security analytics and adapt them to the supervision of different networks.

This paper is a continuation of [5], in which the architecture of the iIDS has already been presented in detail. The implementation of the intelligent and model-based iIDS, including the explanation of attack detection methods is shown in this paper.

The structure of this paper is organized as follows: Section II describes the related work. Section III presents the architecture of the iIDS. In Section IV the rule-based modules of iIDS are described in detail. Section V deals with the explorative data analysis, while Section VI describes data preprocessing which is required for the AI modules. The used AI based modules of the iIDS are shown in Section VII, followed by a conclusion and an outlook on future work VIII.

II. Related Work

In recent years, AI methods have been increasingly used in many different sectors including the healthcare sector. The increasing number of IoT networks needs to be detected reliably and conscientiously from cyber attacks. Different approaches are used for the respective iIDS. Vinayakamur et. al [6] are using self-taught learning as a deep learning approach. Two steps are required to detect attacks. To begin with, the feature representation is learned from a large collection of

 ISBN: 978-1-61208-948-5

unlabelled data. Subsequently, this learned representation is applied to labelled data and thus used for the detection of attacks. Summerville et. al [7] use anomaly detection as their main detection method. The basis is a deep packet analysis approach, which uses a bit-pattern technique. The network payload is described as a sequence of bytes, also called bit-pattern. Feature extraction is done as an overlapping tuple of bytes, also known as n-grams. McDermott et. al [8], on the other hand, use a machine learning approach to detect botnets in IoT networks. They developed a model based on deep bidirectional Long Short Term Memory using a Recurrent Neural Network. Burn et. al [9] are using a deep learning approach for detecting attacks, in which they use a dense random neural network.

The approach for the iIDS differs in some aspects. On the one hand, we use common network analysing methods further described as static methods and on the other hand we use state of the art AI aproaches to detect anomalies and classify attacks. The previous mentioned approaches can detect anomalies, but none of them can classify attacks. It is also our goal to achieve a zero false positive rate by using AI algorithms and the static based models. As mentioned in our previous paper, we still do not know of a familiar combination that uses the same AI algorithms combined with the static based models.

Therefore two major research questions are to be answered in this paper:

- **RQ 1:** Can artificial intelligence and static analysis be sustainable implemented in practice-oriented intelligent Intrusion Detection System?
- **RQ 2:** How can static and dynamic methods be developed and combined to improve network attack detection?

The goal of this paper is to answer the identified research questions by presenting procedures and techniques for achieving advanced network monitoring.

III. Architecture

The architecture of the iIDS consists of 5 layers, with a Data Collection Layer (DCL) as it's foundation and a Reaction Layer as top layer. The organization of the individual layers and their connections are shown in detail in Figure 1.

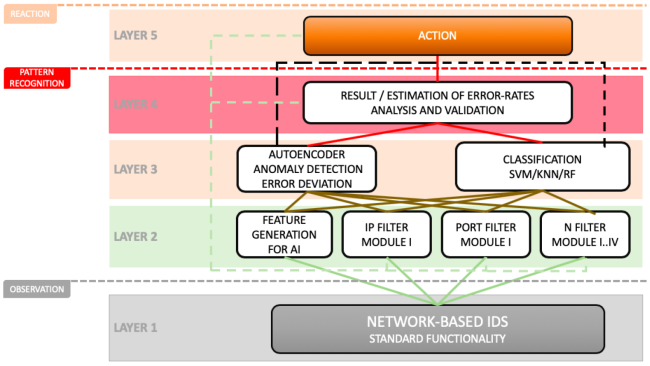

Figure 1. Architecture of the Intelligent Intrusion Detection System [5]

The DCL implements the capturing and conversion mechanism to monitor network traffic and to extract the required transmission information. All data is also stored in a MySQL database for later usage. On top of the DCL, several rule-based modules are implemented to analyse and filter probably malicious traffic with static network observation methods. Also, the data preparation for the upcoming machine learning-based modules is part of this section. The third layer locates the different AI modules used to detect intrusions and to classify the type of attack. For example, a neural network module for anomaly detection is realized at this point of the architecture. A deeper insight into these methods will be given in Section VII. All our modules, rule-based and AI-based, are designed to return an assessment over their predicted outcome. In the penultimate layer, all the return values are evaluated and the probability of an intrusion will be calculated. Based on this calculation and through additional information for example, from the classifier in the third layer the last layer can deploy dedicated security actions to prevent or limit damage to the system. Possible countermeasures could be notifications to an administrator, the shutdown of a connected device, or the interruption of the Internet connection as a final action.

To get a light weighted and expandable system, all major components, like the iIDS itself, the AI-based modules, or the MySQL database, are deployed in their own Docker containers and can be managed independently.

IV. Rule-based Modules

As mentioned in Section III, the rule-based modules are part of the second layer in the presented architecture. They act as a first security barrier and are capable to give roughly feedback about security issues based on the port and address information of Layer 2 and 3 of the ISO/OSI network model.

A. Analysing Port Information

Two different modules are implemented to analyse the network's port information. The first one allows monitoring the individual port usage. With an analysis of the network packages, the commonly used ports of the network participants can be discovered, which enables the ability to whitelist these ports. In reverse, packages which do not have at least a whitelisted source or destination port number will be treated as a possible malicious package and an intrusion assessment value for the subsequent evaluation will be addressed to the next layer. The second module is designed to discover port scan attacks. The purpose of a port scan is to evaluate the open ports of a target system which can be used to set up a connection. Despite a port scan is not an illegal action, at least in Germany, it is often used to get information about a target for later attacks. Because of this common intention and their easy execution with open-source software like Nmap [10], port scans will be treated as a threat indicator for the iIDS.

B. Analysing Address Information

Part of the captured data from the Data Link Layer and the Network Layer is the address information. Based on the unique

MAC-Address and the allocation of a static IP the trusted network members can be verified. To further enhance security also the state of the dynamic host configuration protocol is analysed for violations of thresholds, such as IP range limits. The obtained information is also used to support the AI-based modules and provides important indicators for the reaction layer to defend against attacks.

V. Explorative Data Analysis

Explorative Data Analysis (EDA) provides a statistic insight into a given data set, enables the recognition and visualisation of dependencies and anomalies, and forms the basis for further feature extractions [11].

A. Data Insights

The used data set for training and testing the AI-based modules is based on a laboratory replica of a smart home (SHLab) that delivers network data from common IoT devices. Table I shows the scope of the used data set based on different labels. Two-thirds of the data are packages from normal daily data traffic, one-third are attack packages. Most of the malicious data are Distributed Denial of Service (DDoS) attacks, divided into SYN-, PSH-ACK-, FIN-, ICMP- or UDP-floods, but also Wi-Fi-Deauthentication attacks are included.

TABLE I
Composition of the Used Data Set

Intrusion Class	Packages
Normal Data	**908355**
Wi-Fi-Deauthentication Attack	**32049**
DDoS Attacks	**468769**
— SYN-Flood	147849
— FIN-Flood	27408
— PSH-ACK-Flood	20971
— ICMP-Flood	185058
— UDP-Flood	87483
Combined Dataset	**1409173**

The focus of the iIDS is on the metadata analysis of the header information. The payload information is also only captured as metadata because it is often transmitted in encrypted form. Overall, 52 different data features are captured from the OSI layers 2, 3 and 4. This includes the address and port information mentioned in IV and furthermore data from the Transport Layer, for example, the TCP flags or checksums.

A correlation analysis allows a better insight into the correlations between important features. How well the features fit together is indicated by the correlation coefficient. The coefficient scales from – 1 to 1, whereby a value of 0 indicates no correlation between two features and -1 and 1 both indicate a strong linear correlation. Figure 2 shows a correlation matrix of the most important metadata.

One of these important features is the packet length. A comparison of normal data and attack data showed that attack packages have a significant lower packet length. Furthermore, the evaluation revealed that the attacks don't change the packet length over the attack time span. Another feature is the data offset of TCP packages, which is an indicator for the header

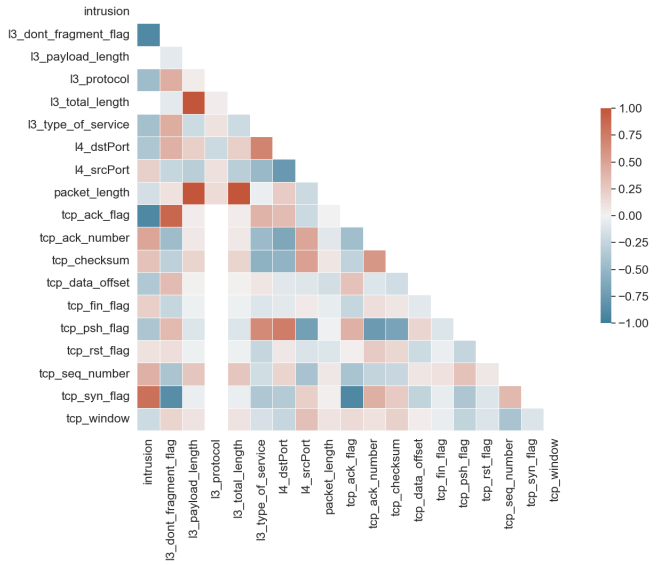

Figure 2. Feature Correlations

size because it contains the position of the payload in a packet. In addition to a shorter packet length, a more detailed insight showed that attack packages also have a shorter header length, which leads to a smaller data offset. DDoS attacks aim to flood their target with a large number of packages. To achieve this, it is useful to have the bare minimum of packet size. This contains a small payload size and the least amount of header options and as a result, a small data offset. These and several additional network characteristics, such as port, flag, protocol information, are examined in order to be able to derive sustainable input for the iIDS modules.

B. Feature Extratction

For the extraction of new features from the data set two different concepts were developed. Both approaches derive additional information from the temporal context of the network packets. However, a distinction is made here as to when or to what extent network packets are measured at the node. In both processing methods, the incoming network packets are aggregated into small blocks of different characteristics further called as block-based approach. Hereby the data is ether combined to a specific number of packets measured on the order of arrival on the node or measured on passing the node in specific time windows.

Quantity-Blocks: Combines the data captured by the observation level of the iIDS to equal sized blocks calculated on a specific count value.

Time-Window-Blocks: Combines the data captured by the observation level of the iIDS to equal sized blocks based on fixed time frames.

However, the best results have been achieved by combining both concepts together. To detected network attacks the incoming packets of each local network member is separated by destination IP or MAC addresses. These packets are then processed by both methods and combined to quantity- and

 ISBN: 978-1-61208-948-5

time-based blocks. This allows a device-specific analysis of the network traffic (Figure 3). Through this combined concept

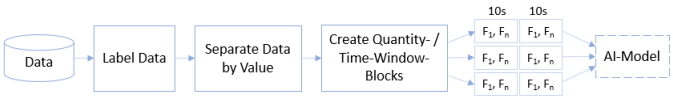

Figure 3. Feature Generation Process

time-based correlation can be used for each device to extract additional features to enhance AI-based attack pattern recognition. This modern, unconventional approach makes it possible to find new classification patterns and integrate them into the analyzes of the iIDS.

VI. Data Preprocessing

Preprocessing the data is also performed on the second layer alonge the rule-based modules. Data preprocessing consists of cleaning, labeling, encoding, normalization and standardisation of the caputred data.

A. Encoding

The encoding of the captured package data is an important step for later usage. Some network information like the MAC or IP addresses but also the different protocol types like TCP, UDP or ICMP are stored in a MySQL VARCHAR data format. Another already mentioned reason for encoding parts of the data is to make it accessible for the AI modules. The majority of ML models require specific data types.

There were two options to cope with that problem. The first one was to exclude this data from later usage. This is not a suitable option because of the importance of the information for the classification. To make the information usable for the AI-based modules a label encoding function is used. Label encoding replaces the distinct categories, i.e. the unique MAC addresses, with a numeric value. Through this, the specific value gets lost, but the overall correlation is still intact, which is important for the ML usage.

B. Cleaning

Missing data and NaN values are an additional problem for most ML models. Due to the huge variety of protocols used in network traffic the entries in the data set often contains empty fields. In example, an IPv4 package has no specific IPv6 information and vice versa. Features with more than 20% missing data entries are removed from the data set, because no statistical interpolation parameters can be calculated from the limited data stock, which can fill the missing gaps without errors. This doesn't apply for all network values like e.g. TCP flags. Therefore and for the other features we use different interpolation methods based on the specification of the feature to deal with missing data. This includes the use of mean and median interpolation for the empty data fields.

C. Normalization and Standardisation

Feature rescaling is an often-done step in the data preprocessing phase of most AI-based models. It is not an essential requirement and not all algorithms benefit in the same way from this process. However, it can lead to better learning performance because most AI-based models perform poorly if the input features have significant different scales. Features with a bigger scale have more weight in each iteration and subsequently dominate the training process. Through rescaling, this disparity in weighting between the features should be minimized.

There are two major ways to perform feature rescaling: normalization and standardisation. The normalization, also often called min-max-scaling, converts the original range of individual features to a general scale for all features. A common interval for this scale is [0, 1] [12, p. 44]. Figure 4 shows on the left side the original range of the port numbers with a scale of 0 to 65535. On the right site the port numbers have been normalized with a Min-Max-Scaler.

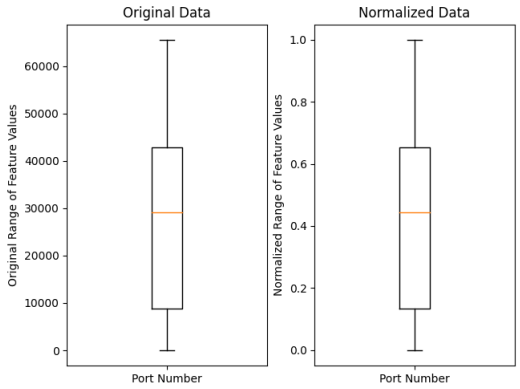

Figure 4. Comparison of Original and Normalized Data

The standardisation shapes the feature values in the proportion of a normal distribution. The mean value of the normal distribution is calculated over the elements of a feature. Unlike normalization, the interval limits are not given values. However, the standard deviation from the mean value is used to set the scale for the feature rescaling. Standardisation is often used for data with a natural standard distribution [12, p. 44].

D. Snorkel

Snorkel is a python package for labelling AI training data based on the work of Alex Ratner [13]. The proposed solution is, to enable the developer to implement labelling functions, which programmatically imply rules to label the data.

The implementation process starts by aggregating the data over 10 second time intervals. As already mentioned in Section V-B this is necessary to improve the performance of the AI-based modules and Snorkel is able to benefit from this procedure too. The aggregated data is used to generate specific indices for each intrusion class. Most flooding attacks don't change their parameters during an attack, therefore the

assumption is that the most common parameter subsets belong to flood packages. For a TCP flood, this leads to an index with the destination port and the packet length as parameters. This approach is also flexible enough to handle continuous new data from the SHLab without the need for changes. Trained on the data set mentioned in Section V-A Snorkel is able to classify all aggregated entries with an accuracy of 90-95%.

The difficulty is to find a classification model with two important properties. The first requirement is a good training result with the aggregated and labelled data delivered from Snorkel. The second requirement is a high accuracy on normal network data delivered from the SHLab. To test these requirements different classification models are used, like a Nearest Neighbour Model, a Support Vector Machine, a Logistic Regression Model, a Decision Tree and a Random Forest. All models accomplish the first requirement with an average accuracy of 90%. The second requirement is more difficult for some models. The Nearest Neighbour Model and the Support Vector Machine achieve 10% accuracy, followed by the Logistic Regression Model (75%) and the Decision Tree (85%) with noticeably better results. The Random Forest fulfilled both requirements with 90% accuracy.

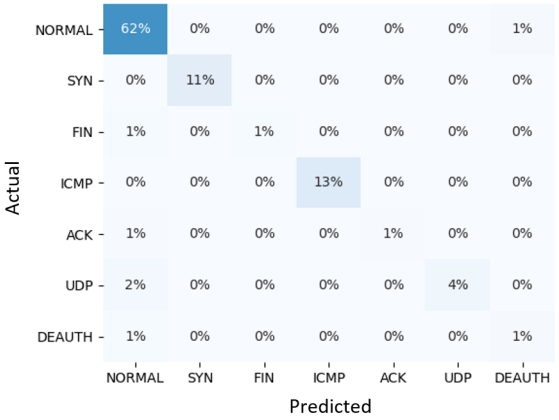

Figure 5. Confusion Matrix of the Random Forest Model

Figure 5 shows the detailed result of the Random Forest in form of a confusion matrix with the performed attacks and the distribution of the actual and predicted classes. The vertical axis represents the actual class of the input value. The horizontal axis represents the predicted class calculated by the classifier. The area percentages of a confusion matrix clarify how precise parts of the data set can be predicted. The evaluation reflects the already addressed good results but shows also some minor problems with the FIN-flood, SYN-ACK-flood and the Wi-Fi-deauthentication attack.

VII. AI-BASED MODULES

Artificial intelligence enables the consistent analysis of complex data through the use of special architectures and neural network techniques. In the following, the two architectures of the developed neural networks are presented. As shown in Figure 1, the AI-based modules are located in the third layer of the architecture. Two different models are developed. One is used to dectect anomalies through the use of an neural network, which is based on our previous publication where we described the theoretical approach. The other one is trained to classify attacks and is based on a pretrained VGG19 Convolutional Neural Network (CNN).

A. Anomaly Detection

To detect an attack there are two major ways, Signature Detection (SD) and Anomaly Detection (AD). The advantage of SD is, that known attacks can be detected very fast and with a high degree of precision. The downside is, that this method needs a well-maintained database with historical and actual attack signatures. This leads to a higher administrative burden and consequently, the system would be more costly. The AD avoids this disadvantage by monitoring the network and building a reference for the usual daily traffic. This allows the AD to recognize new and unknown attacks which would be overlooked by a SD-based system. But due to this characteristic, the AD is also prone to false-positive alarms because changes of the network traffic, for example, by bigger updates, can exceed the normal frame of reference.

1) Autoencoder-Anomaly-Detection-Model: To detect anomalies, we use a special neural network construction called Autoencoder. Their specific process logic allows the neural network to learn without any supervision. Autoencoder are useful tools for feature detection and can analyse unlabelled data with a large variety of different protocols. Autoencoder reduce a given input to the smaller dimensions. This has the consequence that the most important network information is elaborated. From this point on a reconstruction process is started to extrapolate the original input from the smallest reduced dimenson called bottelneck, as shown in Figure 6.

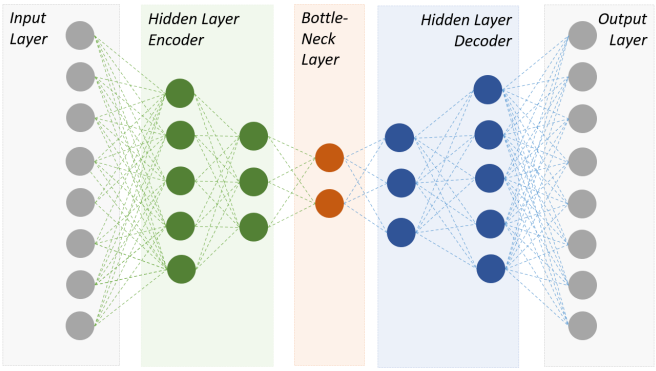

Figure 6. Basic Architecture of a Deep Autoencoder

After the training phase, the Autoencoder has learned to reconstruct the input information only based on the reduced information in the bottleneck. This means the reconstruction error of an extrapolated package compared to the input package is small. In reverse, the reconstruction of an attack package is not part of the trained behaviour of the neural network. Therefore the reconstruction error differs compared

to reconstruction error of normal network traffic. Based on the characteristics of the reconstruction error we can calculate the probability of an network anomaly. However, the Autoencoder cannot specify the specific kind of attack, this means classification models are good enhancements.

B. CNN-Classification-Model

Through a precise classification of threats, we gain additional information which can be used to deploy countermeasures in the reaction layer. The used classification model is based on the VGG19 Model, which was developed by the Visual Geometry Group of the University of Oxford [14].

1) CNN: The implementation of the classifier follows the assumption that the conversion of network packages into images can lead to better classification performance. To transform a packet into an RGB image all parameters had to be converted into a numerical value between 0 and 255. This range represents the 8 bits for each RGB colour channel. As described in Section VI-A all non-numeric values had to be transformed before rescaling. Based on this, a normalization with an Min-Max-Scaler was performed. Different to the procedure in Section VI-C the range for the Min-Max-Scaler was set to 0 – 255. To convert the rescaled data into an image, an array with 3072 elements was declared. The VGG19 algorithm requires 3 layers with a size of 32 by 32 pixels. Each layer represents the values of one RGB colour, i.e., red, green or blue. This leads to a bare minimum size of 3072 pixels for each image. Before the packet data is transferred, the array is initialised with 0, which can be seen as a representation of a fully black image. Thereafter the packet data is copied into the array and split into three parts, one for each layer. Every part is transformed into a 32 by 32 pixel layer and recombined with the other 2 layers to create a colour tensor that can be used by the VGG19 Model. First tests with this classification model delivered promising results. However, further tests with larger and more heterogeneous data sets are necessary to verify these results.

VIII. Conclusion and Future Work

The presented architecture provides the basis for the implementation of the static and AI-based methods for the iIDS. The conceptual design of iIDS is to combine different network monitoring methods in such a way that they can be operated both locally and in the cloud. The static methods are analyzing information of the recorded network traffic. In the same layer as the static analysis, EDA and data preparation are performed. The gathered information of EDA derives the most relevant features for the subsequent AI modules. Data preprocessing prepares the data set for the usage in static and AI modules. Different AI algorithms are implemented. The first module is used to detect anomalies using a special architecture of neural networks. By dimension reduction of the input space and subsequent extrapolation from the smaller dimension space, network anomalies can be detected by analyzing the reconstruction error expression. The second module is used for the classification of attacks. Here, data blocks are processed by time and number to create RBG image data. The convolutional neural network can then classify network attacks based on certain patterns within the image data. The data processing steps and data analysis methods described in the paper show that static and dynamic methods can be developed and combined in practice to provide better network monitoring.

In the future work, a more detailed evaluation layer is to be developed. In order to achieve the desired improvement, an algorithm will be developed to enhance the aggregation of the static and AI-based module results. Due to this changes the iIDS should be able to find even more appropriate countermeasures for detecting attacks.

References

[1] Robert Koch Institut, Demographischer Wandel, 30.07.2020, [online] Available at: https://www.rki.de/DE/Content/GesundAZ/D/Demographie_Wandel/Demographie_Wandel_node.html [retrieved: February, 2022]

[2] Ambient Assisted Living Deutschland - Technik die unser Leben vereinfacht, 2016 [online] Available at: http://www.aal-deutschland.de/ [retrieved: February, 2022]

[3] Bundesamt für Sicherheit in der Informationstechnik. 2022. Smart Meter Gateway Dreh- und Angelpunkt des intelligenten Messsystems. [Online] Available at: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Smart-metering/Smart-Meter-Gateway/smart-meter-gateway_node.html [retrieved: March, 2022].

[4] M. G. Avram, "Advantages and challenges of adopting cloud computing from an enterprise perspective", 2014, Procedia Technology Volume 12, p-529-534. [Online] Available at: https://www.sciencedirect.com/science/article/pii/S221201731300710X [retrieved: March, 2022].

[5] J. Graf, K. Neubauer, S. Fischer ,and R. Hackenberg, "Architecture of an intelligent Intrusion Detection System for Smart Home," 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2020, pp. 1-6

[6] R. Vinayakumar et al. "Deep Learning Approach for Intelligent Intrusion Detection System," in IEEE Access, vol. 7, pp. 41525-41550, 2019

[7] D. H. Summerville, K. M. Zach ,and Y. Chen, "Ultra-lightweight deep packet anomaly detection for Internet of Things devices," 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC), 2015, pp.1-8

[8] C. D. McDermott, F. Majdani ,and A. V. Petrovski, "Botnet Detection in the Internet of Things using Deep Learning Approaches," 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1-8,

[9] O. Brun et al. "Deep Learning with dense random neural network for detecting attacks against IoT-connected home environments", Communication in Computer and Information Science 821, vol.1, pp.79-89, February 2018

[10] M. Shah et al. "Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool," 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2019, pp. 1-6.

[11] S. K. Mukhiya, U. Ahmed, Hands-On Exploratory Data Analysis with Python, Birmingham: Packt Publishing, 2020

[12] A. Burkov, The hundred-page machine learining book, by Andriy Burkov, Quebec City, Canada, 2019

[13] A. Ratner et al. Snorkel: rapid training data creation with weak supervision, Proc. VLDB Endow. 11, 3 (November 2017), pp.269–282.

[14] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," presented at 3rd International Conference on Learning Representations, San Diego, CA, USA, May 7-9, 2015, pp.1-14.

 ISBN: 978-1-61208-948-5