Design and Implementation of a Model-based Intrusion Detection System for IoT Networks using AI

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, katrin.neubauer, rudolf.hackenberg}@oth-regensburg.de

***Abstract*—The rising digitalisation introduces many useful features, as a result of, which the vulnerability for cyber attacks increases. Internet of Things devices and networks can be used to monitor and process sensitive data but at the same time they often are not hardened against threats. This can subsequently put personal data at risk. In this paper the capabilities of an approach to combine artificial intelligence and static analysis in an intelligent Intrusion Detection System for Internet of Things networks are evaluated. The development of static and dynamic methods for attack detection in networks is additionally discussed. The architecture follows a layer-based concept. Methods of classic security analysis and artificial intelligence are therefore deployed in a modular manner. For the extraction of important features a block-based approach has been developed, in which the calculated entropy of the network traffic is used in the extraction process. Detailed insights into the methodologies to analyse port and address information as well as used tools like Snort and Snorkel are given respectively. The metadata of the network traffic and extracted features are then used in combination to further improve the performance of anomaly detection and attack classification. The various models and algorithms utilised in this process are also shown in detail. This approach demonstrates that the security of Internet of Things environments can be enhanced with the deployment of an intelligent Intrusion Detection System that uses combined methodologies of static analysis and artificial intelligence.**

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

I. Introduction

The methods and procedures presented are based on the publication "Design and Implementation of an Intelligent and Model-based Intrusion Detection System for IoT Networks" published at IARIA CLOUD COMPUTING 2022 [1] and are shown in more detail. The goal is to provide a deeper insight into the development of an intelligent Intrusion Detection System (iIDS) that goes beyond the original paper and thus also provides an extension to the latest research results.

Demographic change is a particular challenge worldwide. One consequence of demographic change is an ageing population. However, because of the now higher life expectancy, the risk of illness for each older person is also increasing [2]. For this reason, measures must be taken to enable the ageing population to live more safely.

Ambient Assisted Living (AAL) is used to improve the safety of people in need of assistance. 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 [3]. The iIDS described is part of the 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 and processed. This kind of sensitive data is required in order to enable people in need of care to live in their familiar environment for as long as possible. To address this problem an AAL service is to be developed within the research project SEGAL. The purpose of the AAL service is to process the recorded data from Internet of Things (IoT) devices and send it via the Smart Meter Gateway (SMGW) to the AAL data management of the responsible control center from the AAL-Hub. The SMGW is a secure communication channel, as a certificated communication path is used for the transmission of the recorded data [4]. However, the exchange of data between IoT devices and AAL hub is not necessarily to be considered secure and can be seen as a target for attacks. Therefore, it is necessary to secure the communication between IoT devices and the backend system to prevent manipulation of the transmitted data. In this case, the iIDS is used to protect sensitive recorded data, as it is intended to detect possible attacks.

The increasing need for security is not limited to the healthcare sector. All IoT networks can be targets for various attacks. The Federal Criminal Police Office of Germany states in one of their reports that these networks can be used by malicious actors to amplify Distributed Denial of Service (DDoS) attacks [5]. The following case is a prime example mentioned in this report. In September 2019 Wikipedia's server infrastructure has been targeted with a DDoS attack that has likely been conducted by IoT devices and was therefore unreachable for several hours. These devices are not only used to perform attacks but can also be the target. In 2018 the number of attacks against Symantec's IoT honeypot has averaged to 5200 per month targeting mainly routers and cameras [6]. Their report also shows that approximately half of the usernames and passwords used in attacks are present in the Top 10 ranking. This greatly increases the attack surface and may further bring

the attention of threat actors to IoT devices.

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 [7]. 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.

In [8] the architecture of the iIDS has been presented initially. The implementation of the intelligent and model-based iIDS, including the explanation of attack detection methods is further described in detail.

The structure is organised 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 (EDA), while Section VI describes data preprocessing, required for AI modules. The developed AI based modules are shown in Section VII, followed by a conclusion and an outlook on future work in Section VIII.

II. Related Work

In recent years, AI methods have been increasingly used in many different sectors including the healthcare sector. For the increasing number of IoT networks, possible cyber attacks needs to be detected reliably and conscientiously to guarantee security. Different approaches are used for the respective iIDS.

As Vinayakamur et al. [9] show deep learning approaches like self-taught learning can be an improvement for Intrusion Detection System (IDS). Also, Anomaly Detection (AD) approaches are commonly used. The usage of a bit-pattern technique for deep packet analysis is therefore one useful approach as Summerville et al. showed in [10]. McDermott et al. [11], on the other hand, use a deep 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. [12] are using a deep learning approach for detecting attacks, in which they use a dense random neural network.

The approach for the SEGAL iIDS differs in some aspects. On 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 approaches to detect anomalies and classify attacks. The previous mentioned approaches can detect anomalies, but none of them can classify attacks. Our goal is to achieve a false positive rate as small as possible by using AI algorithms and static based models.

Therefore, two major research questions are to be answered:

- **RQ 1:** Can AI and static analysis be sustainably implemented in practice-oriented iIDS in AAL environments?
- **RQ 2:** How can static and dynamic methods be developed and combined to improve network attack detection?

The goal is to answer the identified research questions by presenting procedures and techniques for achieving advanced network observation.

III. Architecture

The architecture of the iIDS consists of 5 layers, with an Observation Layer as its basis and an Action Layer as top layer. The organisation of the individual layers and their connections are shown in Figure 1.

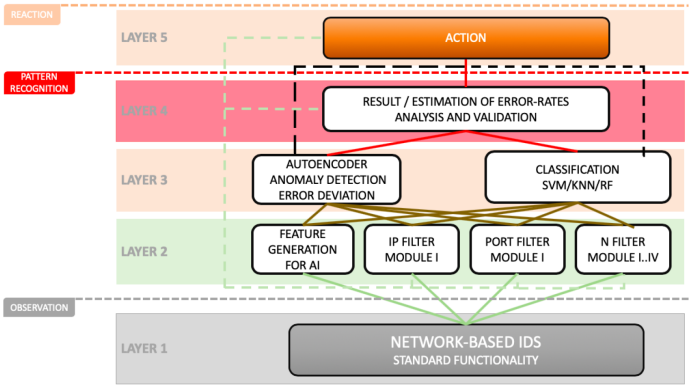

Figure 1. Architecture of the Intelligent Intrusion Detection System [8]

The Observation Layer, also called Data Collection Layer (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 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 AI-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. A deeper insight into these methods will be given in Section VII. All 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 lightweight and expandable system, all major components, like the iIDS itself, the AI-based modules, or the 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 feedback on security issues based on observed network metadata of different ISO/OSI network model layer.

A. Analysing Port Information

Two 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 may not be an illegal action, 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 [13], port scans can be treated in certain cases as threat indicators.

There exist multiple ways of conducting port scans. SYN scans, ACK scans and FIN scans use TCP packets with the according TCP flag set. Packets of NULL scans have no flags set while XMAS scans set the FIN, PSH and URG bits. Another possibility to determine open ports is to complete the three way handshake and close the connection immediately afterwards. There are also scans that use UDP and ICMP packets for reconnaissance [14].

One method to detect them is to monitor the network traffic and to define a threshold for the number of received packets from a source. Whenever within a specified time frame this threshold is exceeded by packets with the identically set TCP flags, it can be assumed that a port scan is conducted against the network. The source is determined by the source IP address of the packet. This threshold has to be adjusted to the regular network traffic in order to minimise false positives [14].

Another way of detecting port scans is proposed by Aniello, Lodi and Baldoni [15]. This approach combines three results of analysis into a rank. This rank is then compared to a defined threshold to determine if a scan is conducted. One step in this calculation is the determination of the entropy of failed connections from one source. For trustworthy sources this value will be near 0 while connection from scanners will be near 1. This is based on the way connections are established. A scanner will change either the destination IP address or the destination port with each request thus pushing the entropy value closer to 1. With this methodology the entropy value is calculated the following way where x = source IP address, y = destination IP address, p = destination port and $failures(x, y, p)$ = number of failed connections from x to y with destination port p [15].

$$N(x) = \sum_{y,p} failures(x, y, p) \tag{1}$$

$$stat(x, y, p) = \frac{failures(x, y, p)}{N(x)} \tag{2}$$

$$EN(x) = -\frac{\sum_{y,p}(stat(x, y, p)\log_2(stat(x, y, p)))}{\log_2(N(x))} \tag{3}$$

For the SEGAL iIDS both methods of these approaches are combined to get more consistent results while avoiding as many false positives as possible. The initial calculation of the entropy has been altered as described in equations (4) and (5) to fit the extended approach. First the number of packets from one source are categorised based on the layer 4 protocol of the ISO/OSI model and the set TCP flags. If the number of packets in one category exceeds the previously defined threshold, it is considered possible that the source is conducting a port scan.

Subsequently, the entropy value of the suspicious packets is determined with slightly adjusted calculations where $suspicious(x, y, p)$ denotes the number of suspicious packets. If this value is close enough to 1, the iIDS assigns these packets to a scan.

$$N(x) = \sum_{y,p} suspicious(x, y, p) \tag{4}$$

$$stat(x, y, p) = \frac{suspicious(x, y, p)}{N(x)} \tag{5}$$

$$EN(x) = -\frac{\sum_{y,p}(stat(x, y, p)\log_2(stat(x, y, p)))}{\log_2(N(x))} \tag{6}$$

B. Analysing Address Information

Part of the captured data from the Data Link Layer and the Network Layer is the address information. The data link layer and the network layer represent layers 2 and 3 in the ISO/OSI model and contain the necessary metadata to transmit network packets to a host in a destination-oriented manner. Based on the unique MAC-Address and the allocation of a static IP the trusted network members can be verified. A comparison of this information can be achieved by using whitelisting or blacklisting procedures. 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 Action Layer to defend against attacks.

C. Snort

Snort is a free network intrusion detection system (NIDS) and a network intrusion prevention system (NIPS) developed by Martin Roeasch. By using Snort it is possible to protocol IP packets and to analyse data traffic in real time [16]. The basis for pattern recognition is the Aho-Corasick algorithm [17].

Rules are the foundation of Snort's functionality. A distinction is made between two parts of the rule. These two parts are a general Rule Header and a more detailed specification by Rule Options. The header specifies the IP addresses and ports that are to be examined in more detail. In case of a detected signature, the Rule Header also defines the reaction to be performed. The Rule Options define further details of signatures and actions in case of detected intrusions. All rule options can be assigned to four different categories [18].

- **general:** In these options information about the rule can be found. However, these options have no effect on the detection.
- **payload:** These options are used to search for data within the payload. The options can also be linked with each other.
- **non-payload:** These options are used to search for non-payload data.
- **postdetection:** These options are rule-specific triggers. They are used after a rule was applied.

Snort is going to be used alongside the other static modules in the second layer of the SEGAL iIDS. To detect possible attacks, all recorded network traffic is compared against the configured rules. For the SEGAL iIDS, the community rules are used, which are a collection of rules submitted by members of the open source community or Snort integrators. Since the community rule sets are constantly maintained, Snort is able to quickly adapt to new attacks. This enables the SEGAL iIDS to react permanently and quickly to new threats. The rules are used to define malicious network activities. If a match is found against the rules, an alert is sent about the security issue. Snort can be considered one of the first security barriers for this reason.

V. Explorative Data Analysis

EDA provides a statistic insight into a given data set, enables the recognition and visualisation of dependencies, outliers and anomalies, and forms the basis for further feature extractions [19].

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 DDoS attacks, divided into SYN-, PSH-ACK-, FIN-, ICMP- or UDP-floods, but also WiFi-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 SEGAL iIDS focuses on the analysis of network metadata. Since it is not always possible to collect information about the payload due to cryptography, meta information such as payload size and others are captured. Overall, 192 different data features are processed. This includes the address and port information mentioned in Section IV and, furthermore, data from the Transport Layer, for example, the TCP flags or checksums.

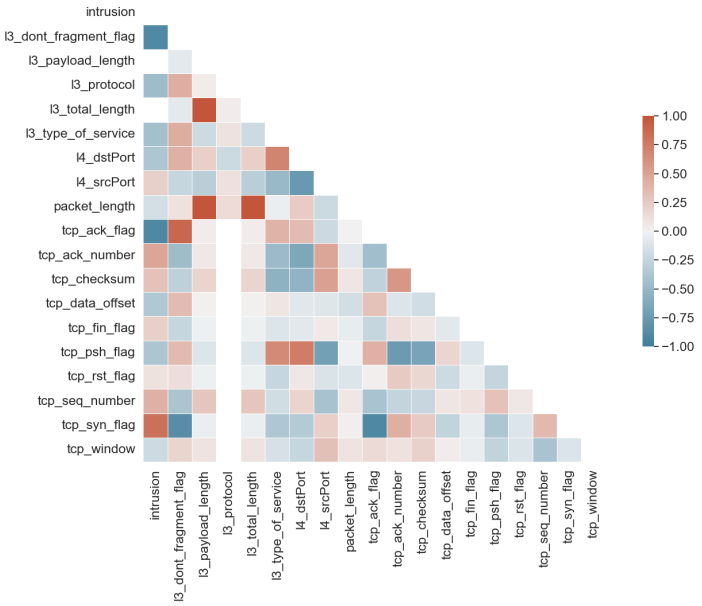

Figure 2. Feature Correlations

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 in average a significant lower packet length. Furthermore, the evaluation revealed that the trivial common DDoS 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 size containing 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, resulting in 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 Extraction

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 approaches. 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.

In principle, both the quantity-blocks and the time-window-blocks approach can be used for networks of all sizes. Depending on the type of system, both methods are differently suited and thus have both advantages and disadvantages. In a low-volume network, the quantity-block approach may be too slow to trigger a timely response from the iIDS. Due to time differences in the arrival of network packets, long waiting times may occur until the desired block size is reached. In this case, the time-window-block approach would allow a more continuous analysis and a faster reaction. However, it can generally be said that the best results were achieved by a combination of both approaches. In the example just given, the time-window-block approach is able to compensate time delays. By combining the two approaches, the time-window-block approach can be used for a preliminary analysis until there are enough packages for a quantity-block analysis.

To detect 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 time-window-blocks. This allows a device-specific analysis of the network traffic (Figure 3). Through this combined concept 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 analyses of the iIDS.

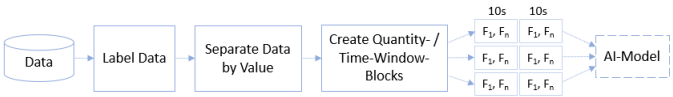

Figure 3. Feature Generation Process

These block-based classification features can be extracted through an analysis based on stochastic methods. Figure 4 shows for example the analysis of payload entropy based on the SHLab network device communication.

Entropy can be seen as a measure of uncertainty. The higher the distribution of values, the higher the value of entropy as a measure of this distribution. The highest value is reached when the dispersion of the values takes on an uniform distribution

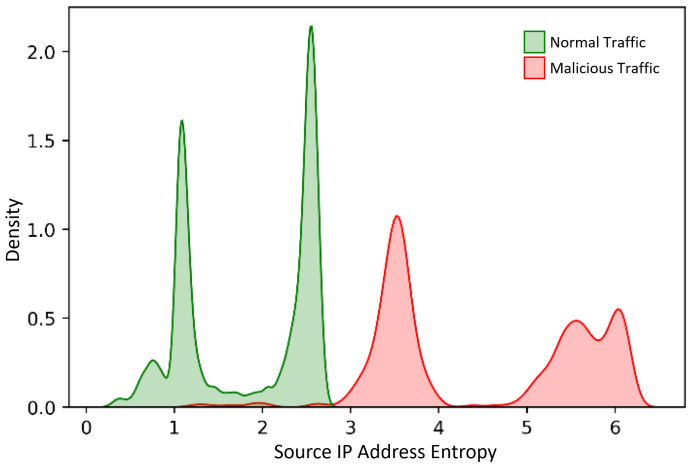

Figure 4. Source IP Address Entropy Comparison Between Normal and Malicious Network Traffic

over all possible outcomes [20]. The calculation of the entropy of a feature is based on the formula developed by Claude Shannon, also known as Shannon entropy [21]. The probability that a certain feature value occurs is specified by the parameter p_i. The individual calculated values are subsequently summed up. The parameter m equals to the number of packages in a quantity- or time-window-block. The result represents the entropy of a feature within a block.

$$H_1 = -\sum_{i=1}^{m} p_i \cdot \log_2 p_i \tag{7}$$

In the example shown in Figure 4, the entropy is the measure of the distribution of the source IP addresses. In more detail, the entropy indicates how scattered the source IP addresses of a block are. The possible outcome in this example is the scope of recorded source IP addresses. A high entropy value of a block indicates that the packets in a block originate from many different communication partners. Conversely, a low value indicates a limited variety. Figure 4 shows that in the case of an attack, the IP address entropy compared to the normal data traffic increases. To maximise the damaging effect, flooding attacks, such as the SYN flood shown here as an example, are usually carried out by different hosts simultaneously. During an attack, the source IP addresses are more evenly distributed over the entire recorded source IP addresses, and this leads consequently to an increased entropy.

VI. Data Preprocessing

Preprocessing the data is also performed on the second layer along the rule-based modules. Data preprocessing consists of cleaning, labelling, encoding, normalisation and standardisation of the captured data.

A. Encoding

The encoding of the captured package data is an important step for later usage. The captured network information like the MAC or IP addresses but also the different protocol types are

stored in a database. Another already mentioned reason for encoding parts of the data is to make it accessible for the AI modules. The majority of models require specific data types.

In order to ensure these requirements, two possible solutions were evaluated. 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 is lost, but the overall correlation is still valid, which is important for Machine Learning (ML) usage.

B. Cleaning

Missing data and NaN values are an additional problem for most AI 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 meaningful statistical interpolation parameters can be calculated from the limited data stock, which can fill the missing gaps without significant errors. This doesn't apply for all network values. Therefore, 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. Normalisation and Standardisation

Feature scaling 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.

There are two major ways to perform feature scaling: Normalisation and Standardisation. The normalisation, 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] [22]. Figure 5 shows an example of Min-Max-Scaling of ports.

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 normalisation, 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 [22].

D. Snorkel

Snorkel is a framework for labelling AI training data based on the work of Alex Ratner [23]. 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. Figure 6 illustrates how the size of the time interval affects both the runtime and the accuracy of Snorkel.

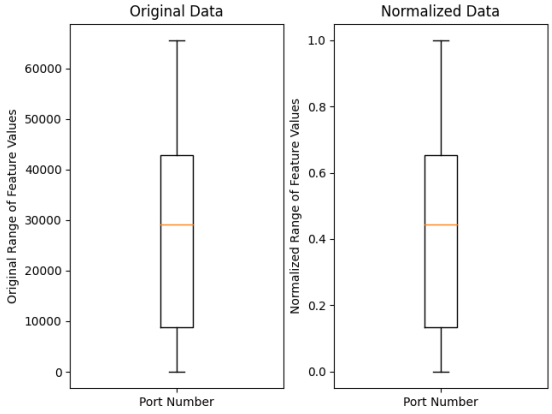

Figure 5. Comparison of Original and Normalised Data

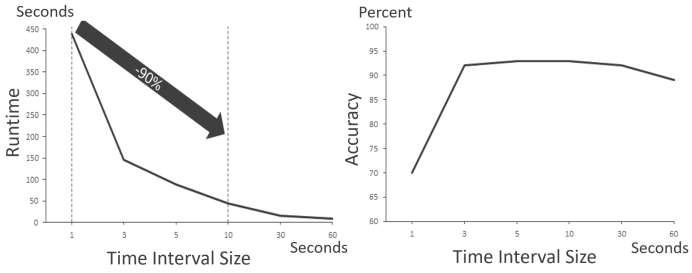

Figure 6. Impact of the Time Interval Size on the Runtime and Accuracy of the Snorkel Training Process

Figure 6 shows the average runtime of a training run over the entire data, when the recorded network packets are summarised over a 1 second intervals. Thereby, an average accuracy of approximately 70% is achieved. By increasing the time interval, the runtime of the Snorkel model can be reduced and the accuracy can be increased. With a time interval between 3 and 10 seconds, the accuracy of the Snorkel model stabilises at approximately 90%. However, longer intervals lead to a decrease in accuracy. Since the accuracy is almost the same with a time interval between 3 and 10 seconds, a 10-second interval is used in the further development. Another advantage of the 10-second interval is that the runtime of the Snorkel model can be reduced by approximately 90%.

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.

VII. AI-based Modules

AI enables the consistent analysis of complex data through the use of special architectures and deep learning 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. Three different modules are developed, whereby two are used to detect anomalies and one for attack classification. The first module for AD is implemented through the use of an neural network, which is based on our previous publication where we described the theoretical approach. The second module relies on the use of binary trees to isolate anomalies. The last module is trained to classify attacks and is based on a pretrained VGG19 [24] Convolutional Neural Network (CNN).

A. Anomaly Detection

To detect an attack there are two major ways, Signature Detection (SD) and 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 recognise 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, a special neural network architecture called Autoencoder is used. Their specific process logic allows the neural network to learn without any supervision. Autoencoder are useful tools for feature detection and dimensionality reduction. Autoencoder reduce a given input to a lower dimensional space. 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 so called bottleneck, as shown in Figure 7.

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

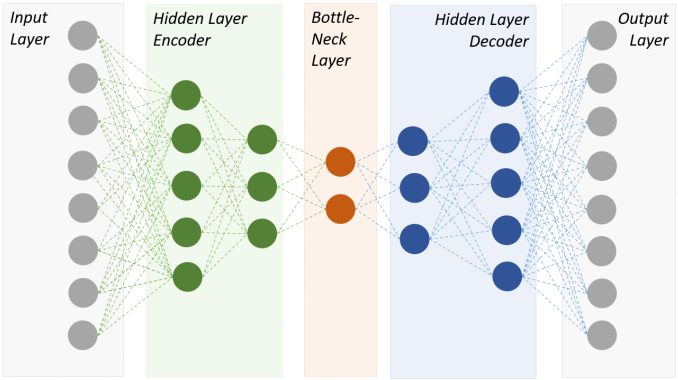

Figure 7. Basic Architecture of a Deep Autoencoder

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.

2) Isolation Forest: In addition to the Autoencoder, an Isolation Forest (IF) Model is currently under development to extend the existing iIDS. The IF is another AI-based method for the unsupervised detection of anomalies in the monitored network. Combined with the results of the Autoencoder, this allows for potentially higher accuracy in detecting anomalies in network traffic. To detect these anomalies, the IF relies on two basic properties that must be present. The anomalies should only make up a small part of the data. Furthermore, the values of the anomaly should differ substantially from those of the normal data. Based on these properties, the anomalies are isolated from the normal data by the IF algorithm. Similar to a random forest, the IF is based on a set of decision trees. These decision trees are also called isolation trees. During the training process of the model, random subsets from the captured network traffic are passed to the different isolation trees. The partitioning of these subsets are then carried out by the usage of a random feature selection based on the captured network metadata. The result of the training process is a set of differently trained isolation trees, which together form the IF. The probability of whether a packet is an anomaly or not is expressed by an anomaly score. Due to the aforementioned property of anomalies that they differ substantially from normal data, the anomaly is usually isolated near the root of the isolation tree, as shown in Figure 8. A short path from the root node to the decisive leaf, indicates an increased probability that the examined package is an anomaly. Conversely, a longer path is more likely to indicate normal network traffic. The anomaly score can thus be derived from the path length and is calculated by averaging all path lengths of the different isolation trees [26] [27].

An advantage of IF is the performance of the AD. Especially in networks with high network traffic, a fast processing of the packets is a decisive criterion in order to be able to initiate appropriate steps in a timely manner. However, like the Autoencoder mentioned above, the IF can only detect

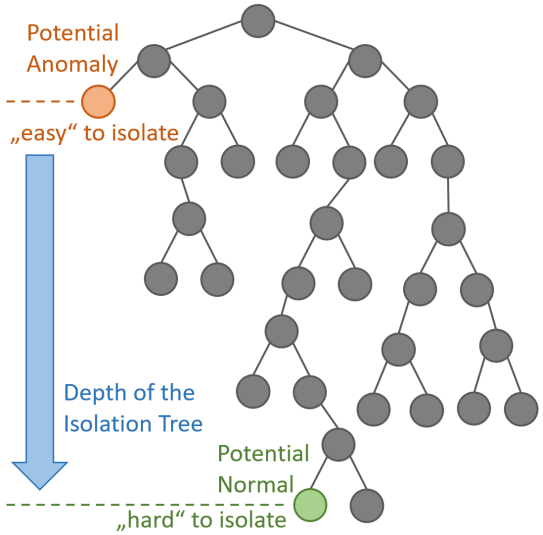

Figure 8. Architecture of an Isolation Tree [25]

malicious network traffic but not classify it in more detail. A more precise classification is therefore still necessary.

B. CNN-Classification-Model

Through a precise classification of threats, we gain additional information, which can be used to deploy countermeasures in the Action Layer. The used classification model is based on the VGG19 Model, which was developed by the Visual Geometry Group of the University of Oxford. A schematic illustration of the VGG19 architecture is shown in Figure 9.

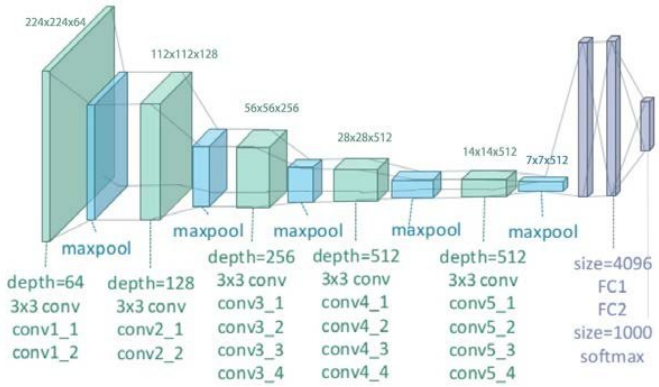

Figure 9. Architecture of the VGG19 Model [28]

The model consists of 16 convolutional layers with a filter size of 3x3 pixels. As shown in Figure 9, the convolutional layers are reduced by 5 maxpooling layers with a window size of 2x2 pixels. After the final reduction, the image data is further processed by 3 fully connected layers. The first two layers work with 4096 channels. For the third layer, the number of channels was reduced to 1000. The architecture of the CNN is completed by a final layer, which features a soft-max activation function [24].

1) CNN: The implementation of the classifier followes the assumption that the conversion of network packages into images can lead to better classification performance. Based on [29] there are 3 different approaches under development for the transformation process of the network data into RGB images. As Figure 10 illustrates, the transformation for all three approaches is based on the same data set. To obtain comparable results, regardless of the approach, the model is used for classification.

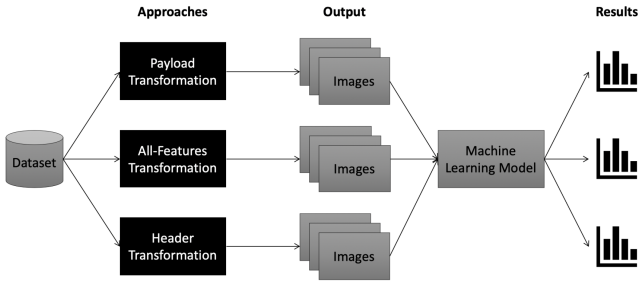

Figure 10. Overview of the RGB Image Transformation Process [29]

The first approach is based on the transformation of the payload data of the respective packet transmitted as an encrypted byte stream. The result of the transformation is a squared RGB image. For the transformation, the basic requirements of the VGG19 CNN for the image to be analysed must be observed. The first requirement is that the image must be in a square format. Also, all images need a minimum width of 32 pixels. Since the analysed picture is an RGB image with three separate colour channels, the minimum results is a 32x32x3 matrix with at least 3072 values. However, since payload data varies greatly in size, the transformation process would also result in images with different resolutions. Since this violates the requirements of the model, all fields of the matrix are initialised with 0. This can be seen as a representation of a completely black image with the necessary minimum dimensions. The individual colour channels, red, green and blue, are each represented by one byte for each pixel. Since the payload data is transmitted as a byte stream, these bytes can be written into the matrix without further processing to replace a portion of the fields previously initialised with 0. This matrix is then transformed into an RGB image that can be classified by the VGG19. Figure 11 shows three results of this transformation process.

Figure 11. RGB Images based on the Payload Transformation Approach [29]

In contrast to the first approach, the second approach in-

cludes not only the user data but also the corresponding header information of the packets in the transformation. The general procedure of the transformation does not change, but the header information must be converted in advance. The reason for this is that the string and integer values of most header information exceed the necessary value range of 0 to 255 for the transformation. As aforementioned, 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 normalisation 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. After rescaling the header information, all values are now within the necessary value range. The header information, each one byte in size, can now be written into the previously declared matrix together with the payload data using the same procedure as in the first approach. However, the images created after the transformation differ only minimally in the number of pixels from the RGB images shown in Figure 11.

The last approach for creating the RGB images for the CNN focuses exclusively on the use of the header information. As described in the second approach, the header information must first be rescaled. Afterwards, the header information can be transformed into an RGB image as also known from the first approach. In contrast to the transformation of the payload data, only a few pixels can be extracted from the header information for the RGB image. However, since the minimum dimensions still apply, the majority of the resulting image would be black. To counteract this problem, the initialised matrix is completely filled for each network packet by repeating the header information. Figure 12 shows three results of the third approach, which are significantly different from those of the payload transformation.

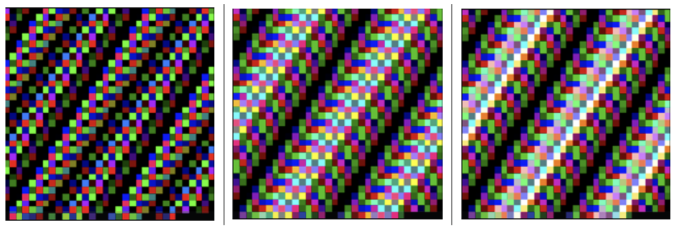

Figure 12. RGB Images based on the Header Information Transformation Approach [29]

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 SEGAL iIDS. The conceptual design of SEGAL 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 analysing information of the recorded network traffic. This allows the SEGAL iIDS to detect possible attacks in the network and alert about these security issues. The static methods also include the analysis of the port information. The analysis is done by categorising the number of packets. 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 analysing the reconstruction error expression. The development of an IF Model is expected to further improve AD. This approach is used to isolate anomalies from the normal data. Randomly sub-sampled network data is processed in a tree structure. Any samples that go deeper into the structure of the tree are less likely to be anomalies because multiple cuts are needed to isolate the samples. Otherwise, samples that end in shorter branches indicate anomalies because it was easier for the tree to separate them from other observations. The combination of these two AD methods empowers the SEGAL iIDS to detect anomalies in network traffic with increased accuracy. 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 CNN 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. The presented iIDS differs from conventional IDS by the modular structure and also by the outsourced preprocessing. Due to the planned module layers, the iIDS to be developed can be used in different application areas without problems. The outsourcing of preprocessing allows the iIDS to be used on the different systems without performance loss.

In the future, 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 these changes the SEGAL iIDS should be able to find even more appropriate countermeasures for detecting attacks. Furthermore, the already existing static and AI-based modules should be further expanded. In the case of the static modules, the detection logic is to be improved, whereby more attacks will be detected by the SEGAL iIDS. With regard to AI-based modules, the classification of the detected attacks should be improved. Thus, attacks detected by iIDS can be better classified. Deep package inspection is also to be used in the SEGAL iIDS allowing monitoring, analysing, filtering and marking of all data packets in the network.

References

[1] P. Vogl, S. Weber, J. Graf, K. Neubauer, and R. Hackenberg, ”Design and Implementation of an Intelligent and Model-based Intrusion Detection System for IoT Networks”, in Proc. CLOUD COMPUTING 2022 Special Track FAST-CSP, Barcelona, Spain, 2022, pp. 7-12.

[2] Robert Koch Institut, "Demographischer Wandel", 2020, [Online] Available at: https://www.rki.de/DE/Content/GesundAZ/D/Demographie_Wandel/Demographie_Wandel_node.html [retrieved: February, 2022].

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

[4] Bundesamt für Sicherheit in der Informationstechnik, "Smart Meter Gateway Dreh- und Angelpunkt des intelligenten Messsystems", 2022, [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].

[5] Bundeskriminalamt, "Cybercrime Bundeslagebild 2019", 2020, [Online] Available at: https://www.bka.de/SharedDocs/Downloads/DE/Publikationen/JahresberichteUndLagebilder/Cybercrime/cybercrimeBundeslagebild2019.html [retrieved: October, 2022].

[6] Symantec, "Internet Security Threat Report", Vol. 24, 2019, [Online] Available at: https://docs.broadcom.com/doc/istr-24-2019-en [retrieved: October, 2022].

[7] M. G. Avram, "Advantages and challenges of adopting cloud computing from an enterprise perspective", Procedia Technology, Vol. 12, 2014, p 529-534.

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

[9] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System", IEEE Access, Vol. 7, 2019, pp. 41525-41550.

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

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

[12] O. Brun, Y. Yin, E. Gelenbe, Y. M. Kadioglu, J. Augusto-Gonzalez, and M Ramos, "Deep Learning with dense random neural network for detecting attacks against IoT-connected home environments", in Proc. First International ISCIS Security Workshop, London, United Kingdom, 2018, pp. 79-89.

[13] M. Shah, S. Ahmed, H. Khan, "Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool", in Proc. 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2019, pp. 1-6.

[14] J. Gadge and A. A. Patil, "Port scan detection", in Proc. 16th IEEE International Conference on Networks, New Delhi, India, 2008, pp. 1–6.

[15] L. Aniello, G. Lodi, and R. Baldoni, "Inter-Domain Stealthy Port Scan Detection through Complex Event Processing", in Proc. 13th European Workshop on Dependable Computing, Pisa, Italy, 2011, pp. 67–72.

[16] snort.org, "SNORT - The Open Source Network Intrusion Detection System", [Online] Available at: http:www.snort.org [retrieved: October, 2022].

[17] B. Caswell, J. Baele, and A. Baker, "Snort Intrusion Detection and Prevention Toolkit (English Edition)", Amsterdam, Netherlands, 2007, p. 193.

[18] The Snort Project, "Snort User Manual", 2020, pp.182-185 [Online] Available at: https://docs.broadcom.com/doc/istr-24-2019-en [retrieved: October, 2022].

[19] S. K. Mukhiya and U. Ahmed, "Hands-On Exploratory Data Analysis with Python", Birmingham, United Kingdom, 2020.

[20] T. M. Cover and J. A. Thomas, "Elements of Information Theory", Hoboken, New Jersey, USA, 2005, pp. 29-30.

[21] C. E. Shannon, "A mathematical theory of communication", The Bell System Technical Journal, Vol. 27, 1948, pp. 379-423.

[22] A. Burkov, "The hundred-page machine learining book", Quebec City, Canada, 2019, p. 44.

[23] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Re "Snorkel: rapid training data creation with weak supervision", in Proc. 44th International Conference on Very Large Data Bases, Rio de Janeiro, Brazil, 2017, pp. 269–282.

[24] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", in Proc. 3rd International Conference on Learning Representations, San Diego, CA, USA, 2015, pp. 1-14.

[25] H. Rajeev and U. Devi, "Detection of Credit Card Fraud Using Isolation Forest Algorithm", In: G. Ranganathan, R. Bestak, R. Palanisamy, and A. Rocha, "Pervasive Computing and Social Networking", Lecture Notes in Networks and Systems, Vol. 317, Singapore, 2022.

[26] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation Forest", in Proc. Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2009, pp. 413–422.

[27] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation-Based Anomaly Detection", ACM Transactions on Knowledge Discovery from Data, Vol. 6, 2012, pp. 1-39.

[28] Y. Zheng, C. Yang, and A. Merkulov, "Breast Cancer Screening Using Convolutional Neural Network and Follow-up Digital Mammography", in Proc. SPIE Commercial + Scientific Sensing and Imaging, Orlando, Florida, United States, 2018, p. 8.

[29] J. Ostner, "Usage of Image Classification for Detecting Cyber Attacks in Smart Home Environments", in Proc. Regensburger Applied Research Conference 2020 (RARC 2020), Regensburg, Germany, 2020, pp. 37-43.