Detecting anomalies in BACnet network data

Jernej Tonejc[1], Jaspreet Kaur[2], Alexandra Kobekova[3]

Abstract: Over the last few years, the volume of data in the Building Automation System (BAS) networks has increased exponentially. Nowadays, it is possible to obtain several kinds of data from building networks such as data based on individual service type, specific building location and even specific time of the day. As a consequence, large volumes of data with more variables have to be considered when performing the data analysis. This means that there is a need to identify the most important variables for analysis. In this paper, we introduce a framework which allows the characterization of BACnet network traffic data by means of machine learning techniques. This framework is based on unsupervised machine learning methods, specifically, Principal Components Analysis and Clustering. Such methods are used because of the large volume of data that needs to be taken into consideration, preventing the manual labeling of the data which is required for supervised learning methods. We show the efficiency and effectiveness of the framework in detecting anomalies by performing experiments on different BACnet network traffic data, captured by Wireshark, together with synthetically generated data.

Keywords: BACnet, data analysis, machine learning, flow mapping, unsupervised learning.

1 Introduction

Building Automation System (BAS) performs the monitoring and controlling of all the electrical and mechanical equipment of a building, from lighting to HVAC to security systems. The structure of BAS in a building is composed of an array of mechanical and electrical devices. All the devices are connected to a central control station, usually a computer, where BAS administrators can get an insight into the whole network. Different kinds of data are sent across the BAS network, including the values when a piece of equipment is turned on and off, change in value when some movement is detected in particular areas of a building, change of temperature within the building spaces, change of temperature of air or water within the mechanical systems, valve positions, etc.

Due to the continuous growth in BAS networks, huge amount of data is transferred through such networks every day. Thus, it is becoming an issue for the BAS administrators to perform efficient data analysis on such a big amount of data. It is hard to identify correlations and detect anomalies on such data traces and sometimes most of the anomalies go undetected by the BAS administrators.

To this end, new efficient techniques capable of dealing with huge BAS network traffic data need to be introduced. This work presents a framework which performs BAS traffic

[1] Fraunhofer FKIE, Cyber Security, Friedrich-Ebert-Allee 144, 53113 Bonn, jernej.tonejc@fkie.fraunhofer.de
[2] Fraunhofer FKIE, Cyber Security, jaspreet.kaur@fkie.fraunhofer.de
[3] Fraunhofer FKIE, Cyber Security, alexandra.kobekova@fkie.fraunhofer.de

analysis by means of machine learning techniques. The framework is graphically depicted in Fig. 1.

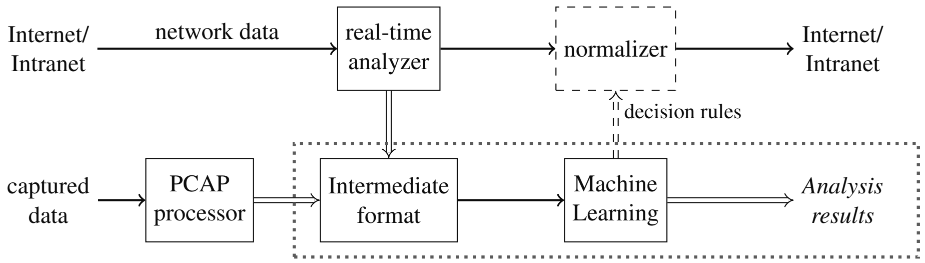

Fig. 1: BAS network analysis framework. The main components of this framework are the *real-time analyzer* and the *normalizer*. Both real-time network data and the captured data are converted into an intermediate form (described in Sect. 3), which is then used to generate decision rules for normalizer with the help of the machine learning methods (described in Sect. 4). This paper focuses on the components within the dotted rectangle.

The main goal of this framework is to characterize the traffic data and detect anomalies. Analysis is done on the BAS traffic captured by `Wireshark`, to differentiate between the relevant traffic and the anomalies. Unsupervised machine learning techniques are used because they are quite efficient in performing analysis on the large volume of network data (and continuously expanding data due to addition of new devices in the network). We consider the following methods: Principal Components Analysis (PCA), and Clustering. These methods have proven to be quite useful in detecting anomalies and discovering patterns while analyzing the network data from other network protocols [DD11].

The network traffic analysis is done on BACnet traffic data. BACnet protocol is one of the data communication protocols used in BAS [ISO12] and has been implemented in products by more than 800 vendors worldwide. The framework supports analyzing prerecorded data in `pcap` format, as well as live capture and on-the-fly analysis using a dedicated analyzer that outputs the live data in the intermediate format.

The rest of the paper is organized as follows. The related work is discussed in Sect. 2. Section 3 explains the structure of BACnet/IP and the machine learning methods used in this study. Section 4 describes the implementation details of machine learning methods along with the flow classifiers. The obtained results are discussed in Sect. 5. The conclusions and future work are presented in Sect. 6.

2 Related Work

Enormous amount of work has been devoted to the application of machine learning and data mining techniques to solve the problem of network traffic analysis. In [NA08], authors present a survey on the application of machine learning techniques for internet traffic classification. Several surveys of anomaly detection methods exist, for example, [C^+09]. Researchers have experimented with both supervised learning and unsupervised learning methods because network traffic analysis is rather a broad domain. Some examples of supervised learning methods to classify network traffic data include Naive Bayesian classifiers [MZ05], and Bayesian neural networks [A$^+$07]. These methods are capable of performing the analysis on a very large set of data but require previous knowledge of the application domain.

Instead, we use unsupervised machine learning methods in our work which can easily identify new anomalies. Unlike supervised learning methods, these techniques do not depend on labelled training data. These approaches are quite simple and effective in supporting the network traffic data analysis. Our analysis mechanism for BAS network traffic data describes the network behavior in a way that enables an extraction of a set of rules. The rules support the administrator to efficiently identify relevant and irrelevant communication among the BAS traffic data. The unsupervised learning methods also have some drawbacks compared to the supervised methods, for example, they generally produce more false positives, and can limit the performance of some specific tasks because of the use of general notions. Some examples of unsupervised learning methods for network traffic data analysis include Clustering algorithms [E$^+$06], Random Forests [DD11], One-Class SVM [DD11] etc.

In another work [M$^+$14], the authors present a comparison of different machine learning methods like Conditional Restricted Boltzmann Machine (CRBM), Artificial Neural Networks (ANNs), and Hidden Markov Models (HMMs) for estimating energy consumption in the buildings. They showed the evaluation of performance of CRBM in the context of building automation systems and proved that CRBM outperforms ANNs and HMMs.

In [P$^+$14], authors propose an anomaly-based intrusion detection system for BACnet/IP networks using Repeated Incremental Pruning to Produce Error Reduction (RIPPER) as a learning method. The features used for learning are similar to the features we use in our methods, however, the algorithm is a supervised learning algorithm and requires the input data to be tagged, and as such, it is less capable of detecting new or unknown types of anomalies.

3 Structure of BACnet/IP data

This section briefly describes the structure of the BACnet/IP network data and introduces the machine learning methods that we use to analyze the data.

3.1 Data selection and processing

In the case of BACnet/IP data over Ethernet, the data is encapsulated in User Datagram Protocol (UDP) layer. Each packet contains communication and control values. An important feature of BACnet networks is the requirement that there exists a unique message path between any two nodes on an interconnected BACnet network [ISO12]. One can therefore speak of flows between BACnet devices, and each flow is defined by the BACnet addresses of the communicating devices. As shown in Fig. 1, the data can either come from real-time capture or from a prerecorded data. The prerecorded data is captured using `Wireshark` and stored in a raw `pcap` format. In order to convert the raw data into the intermediate format, a fair amount of preprocessing is needed. In case of real-time capture and processing, a dedicated traffic analyzer outputs the data directly in the intermediate format. Figure 2 shows the structure of a typical BACnet/IP packet.

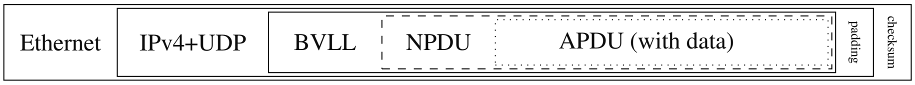

Fig. 2: BACnet/IP data is contained within the UDP datagram as a BACnet Virtual Link Layer (BVLL), which contains the Network layer Protocol Data Unit (NPDU) and the Application layer Protocol Data Unit (APDU) with data.

For a detailed explanation of the structure of NPDU and APDU, we refer the reader to [K+15]. For our analysis we chose the same fields as were suggested in [T+15]. For the sake of completeness, we list the fields in Fig. 3, grouped by the layer in which they appear. The fields that are present in every valid BACnet/IP packet are shown in bold.

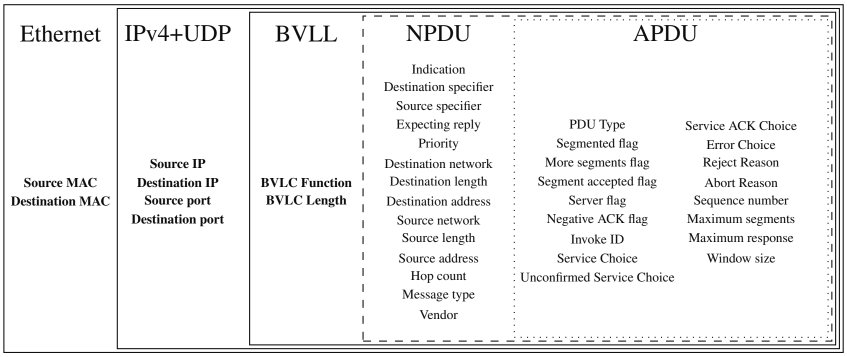

Fig. 3: The fields as features, organized by the layers in which they are present. In addition to the above fields, a timestamp is recorded for each packet (both in recorded `pcap` data as well as during the real-time analysis), bringing the total number of features per packet to 40.

BACnet packets do not have a fixed structure, as the presence of many fields depends on the values of various control bits and fields. For example, the APDU fields Service Choice, Unconfirmed Service Choice, Error Choice, Reject Reason and Abort Reason are never present at the same time, as each exists only for a specific value of the PDU Type field. We simply use the value zero for the missing fields.

The selected fields contain all the essential features of each BACnet/IP packet. Before we can use the network data in machine learning methods, we need to process it. The purpose of processing the network data is twofold:

1. to prepare the data for subsequent machine learning analysis, and
2. to aggregate and export the data in a form suitable for visualizing the flows and creating the model for the flow-based anomaly detection.

All of the fields in BACnet/IP data have a natural representation as unsigned integers. We perform only the following processing. We split the MAC addresses into the Organizationally Unique Identifier part and the vendor-specific identifier, each using three bytes of the MAC address, and encoded as 32-bit integers. The IP addresses are encoded as four 8-bit integers to reflect the natural structure of IP-based networks. The BACnet source and destination addresses could be up to 7 bytes long, so we encode them using 64-bit integers. In case of a malformed packet with address longer than 8 bytes we truncate the value to 64 bits. There is no loss of information for valid packets and the anomalies in addresses can still be detected as the value of the address length field falls outside the valid range. These preprocessing steps convert the 40 features into 48 numerical columns.

Due to completely different scales of data in various fields, e.g., the values for valid BACnet addresses could vary from 0 to 72057594037927935, for Hop count from 0 to 255 and for Indication from 0 to 1, we need to normalize the values, as some of the machine learning methods rely on Euclidean distance between the data points. In particular, PCA and distance-based clustering methods are affected by disproportionate scale for different features.

3.2 Machine learning methods

In order to detect new and unknown traffic anomalies, we need to use unsupervised machine learning methods, as the sheer volume of data prevents any manual tagging of individual packets. There are various machine learning methods that are well-suited for anomaly detection in the network traffic [C$^+$09, DD11]. In this paper, we focused on the following:

1. *Clustering methods*. We focus on the distance-based clustering methods, in particular, k-means and expectation maximization (EM). In k-means, the data is partitioned into k clusters, such that a point in a given cluster is closer to that cluster's centroid than to any other cluster's centroid. For EM, one first needs a probabilistic model, for which the parameters need to be estimated. The parameters are then determined in such a way as to maximize the expectation of the log likelihood.
2. *Principal Component Analysis*. Each packet is treated as a point in a 48-dimensional space. The objective is to project the data on a lower dimensional space, retaining as much variance as possible.

The principal component analysis was used as a preprocessing step before the clustering. The clustering methods were used as a classifier, where the small enough clusters were marked as anomalous.

3.3 Network flows

The communication in BACnet can either be a direct communication between two devices or a broadcast message (local, remote or global). The broadcast messages can be viewed as communication with a special device and are simply treated the same way as other communication in our analysis. The flows can be defined in terms of the source and destination MAC addresses, source and destination IP addresses or source and destination BACnet addresses. Different underlying structures of the BACnet network are revealed when using different flow definitions. For each such flow, a multidimensional statistical model is constructed to characterize the types of messages and their probabilities of occurrence, as well as to determine the expected packet sizes. Due to the mostly static nature of the building automation networks, such flow maps enable efficient anomaly detection. For the flow-based anomaly detection to work we need to make sure we can map the network when no anomalies are present, to obtain the base state of the network. The anomalies can later be detected using analytic methods, where we directly compare the current flow map with the original one, as well as the statistical properties of the individual flows, such as the message type probability distribution, and the packet frequency based on the flow and message type.

4 Implementation Details

In this section, we describe the details of the implementation, together with the data sources that were used to train and test our methods.

4.1 Data sources

As mentioned in Sect. 3, the `pcap` data needs to be processed to make it accessible to the machine learning methods. The preprocessing was implemented in C for efficiency reasons, following the BACnet standard [ISO12]. The result of the preprocessing is the intermediate format, which is simply a CSV-file with 48 columns. The same format is used when analyzing the packets in real time. For certain machine learning methods, namely k-means, EM and PCA, we also scale the data in each columns so as to make the values in that column lie between 0 and 1.

We use two types of sources for training and evaluating our methods. The first source is our BAS lab, which contains several BACnet devices. The traffic is recorded using Wireshark on a PC which is connected to the mirror port of a local switch. We recorded two days worth of traffic. On the first day, only the normal behavior of the lab was present, i.e., no deliberate anomalies were introduced. On the second day, we performed several scans of the BACnet network using a free demo version of a tool called BACeye. The two pcap files contain about 1.2 million BACnet packets each. The preprocessing step took about 6 seconds in total. The amount and variety of data is limited due to the small size of our BAS lab. For comparison, the number of packets that we observed in a large university network is about 16 million per day, or about ten times more than in our lab setup, so the lab setup is a relatively good approximation of an actual building automation network.

The second source of data is a set of artificially generated intermediate format files. As a basis we take the traffic recordings from our lab for a different day, then add various levels of synthetically generated anomalous traffic. The proportion of the anomalous traffic is 0.1%, 0.5%, 1% and 2%, with anomalous records either bunched up or roughly uniformly dispersed throughout the files (this affects the timing information of those packets). We analyzed the attacks in [K$^+$15] and based our anomalies on the following possible attacks:

- Covert channels using different message types to convey information (CCM)
- Covert channels using reserved bits (CCR)
- Using APDU service choices for writing properties or modifying objects (WP)
- Using incorrect bit combinations and strange values for BACnet address lengths (IB)

We generated 100,000 records for each combination, for a total of 33 (=$1+2\times4\times4$) files, including the file without modifications.

4.2 Implementation of machine learning methods and validation

The analysis of the network data was done after the preprocessing step. For the machine learning methods, we extracted all 40 listed features from each BACnet/IP packet, performed the preprocessing steps as indicated in Sect. 3 and obtained a 48-dimensional vector. We then used Weka [H$^+$09] to perform the selected machine learning algorithms. We performed the algorithms on large enough traffic recordings to capture the typical network behavior. Later, we check for each additional packet how well it fits into the model that was learned. This allows us to detect anomalies in the traffic as they do not fit the parameters that were extracted from the sample data. For the learning to be successful, the training data really needs to be representative, otherwise too many false positives are detected.

The data was split into a training set and a testing set. For the lab data we split it into 100,000-line chunks, then trained the algorithms on a randomly selected 66% of the data in one chunk, performed validation on the remaining 34% of the same chunk and then tested the remaining chunks. For the synthetically generated data, we performed the same 66%–34% training/validation split on the file with 1% of anomalous data, then tested the files with different proportions of anomalous data. In addition, we combined the 1% files with all the different types of anomalies and trained the methods on that (with the same training/validation split), then tested all the individual files to evaluate the ability of the so-trained methods to detect various anomalies.

We performed the following machine learning algorithms:

- k-means with $k=5$ (km5) and $k=10$ (km10) clusters
- EM clustering with $k=5$ (EM5) and $k=10$ (EM10) clusters
- PCA, followed by k-means with $k=5$ (Pkm5) and $k=10$ (Pkm10) clusters
- PCA, followed by EM clustering with $k=5$ (PEM5) and $k=10$ (PEM10) clusters

As stated in Sect. 3, we used the clustering methods as classifier, where we marked every cluster smaller than 2% of the total data as anomalous. This was based on the assumption that at most 2% of the data is anomalous. This can potentially lead to many false positive results if the number of clusters is too large, since the non-anomalous clusters also get smaller as the total number of clusters increases.

To evaluate the quality of the chosen machine learning methods, we used the standard quality measures, such as sensitivity, specificity, precision, accuracy and the F_1-score, derived from the *confusion matrix*

	Packet is anomalous	Packet is normal
Packet predicted anomalous	T_P	F_P
Packet predicted normal	F_N	T_N

where T_P denotes the true positives, T_N true negatives, F_P false positives, and F_N false negatives. The sum of all four quantities equals the number of the analyzed packets.

The F_1 score is the harmonic mean of the sensitivity and precision and given the expected nature of the anomalous data, we expect that the accuracy does not say much about the quality of the machine learning methods. Therefore we use the specificity (SPC) and the F_1 score, together with the Matthews correlation coefficient, which is considered one of the best measures for two-class classification [Pow11]. The three measures can can be computed directly from the above confusion matrix as

$$\mathrm{SPC} = \frac{T_N}{F_P + T_N},\ F_1 = \frac{2T_P}{2T_P + F_P + F_N},\ M_{CC} = \frac{T_P T_N - F_N F_P}{\sqrt{(T_P + F_N)(T_P + F_P)(T_N + F_N)(T_N + F_P)}}.$$

4.3 Analyzing the flow data

As mentioned in Sect. 3, when analyzing the flows, the packets can be grouped based on their source and destination addresses. Additionally, the packets can be split into two sets: packets without APDU (these are network layer messages) and packets with APDU. These two sets give rise to the network-layer flows and application-layer flows. Each device that sent or received a message represents a node in a directed weighted *flow graph*. The nodes can be identified using their MAC addresses, their IP addresses or their BACnet addresses. The directed edges of the graph are the flows. Different choices of the node identifiers and types of flows give rise to several different flow graphs for the same packet recording. Additionally, we can assign weights to the edges based on various features, for example, normalized total number of messages in that particular flow, total number of bytes exchanged, the inverse of the average inter-packet arrival time, etc. The graph data is exported as *Graph Exchange XML Format* (GEXF), as well as Javascript Object Notation (JSON) format.

We can learn the normal state of the BAS network by analyzing the data for which we know there are no anomalies. Later, we can perform anomaly detection by comparing the current flow graphs with the saved ones. We visualize the flow graphs and the differences using the Javascript framework D3.js. In addition to comparing the differences between the flow graphs, we also run a community detection algorithm on each graph. The algorithm used is from [B$^+$08]. It attempts to maximize the modularity Q of the graph, defined as

$$Q = \frac{1}{M} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{M} \right] \delta(c_i, c_j),$$

where A_{ij} is the weight of the edge between nodes i and j, k_i is the sum of the weights on the outgoing edges at node i, c_i is the community to which node i is assigned, M is the sum of all weights of all edges and δ is the standard Kronecker δ-function. The algorithm starts with each node in its own community and proceeds iteratively by assigning a node to the community that maximizes the increase of the modularity at each step. When assigning the weights to the edges it is important to normalize them to take into account the specifics of BAS, as there is usually more activity during business hours as compared to the night or weekend.

5 Results

In this section, we present the results of the various analyses that were performed on the data.

5.1 Evaluation of machine learning methods

As stated in Sect. 4, we use the standard quality measures for machine learning methods, together with the Matthews correlation coefficient. The results for various methods are presented in Tab. 1. Due to the space constraints, only a subset of the results is shown. The results for 10 clusters were not significantly better in any case and in some cases (specifically, PEM10), the results were actually worse since some small clusters with non-anomalous data were classified as anomalous, increasing the number of false positives. When training with one type of an attack, only the results of testing against the data with 0.1% of anomalies are shown. The results for 0.5%, 1% and 2% are generally better. When training against all the attacks at once, the results for 0.1% and 2% are given.

Since we know what kind of anomalies were introduced into the data, we are able to compute the values of the confusion matrix. We expect that the methods perform similarly when applied to the data for which the anomalies are truly unknown. For all the data sets, between 1 and 3 clusters were marked as anomalous, with 2 clusters (out of 5) in majority of the cases. For k-means, the training step took less than 3 seconds in all cases, for both $k = 5$ and $k = 10$. For EM, the training took about 40 seconds for $k = 5$ and about 80 seconds for $k = 10$, for each dataset. When combined with the PCA, the training for EM took 3, respectively 12 seconds. Testing was in all cases fast. When using PCA, we always took the first three principal directions, covering about 70–80% of the variance. From the table it is clear that by far the best method is k-means with $k = 5$, preceded by PCA. The

Algorithm	**Train set**	**Test set**	M_{CC}	**Specificity**	F_1 **score**
km5	IB	IB-0.1%	1.0	1.0	1.0
	WP	WP-0.1%	0.9284	0.9998	0.9259
	CCM	CCM-0.1%	0.8908	0.9997	0.885
	CCR	CCR-0.1%	1.0	1.0	1.0
	AA	IB-0.1%	0.9284	0.9998	0.9259
		IB-2%	0.9959	0.9998	0.996
		WP-0.1%	0.9284	0.9998	0.9259
		WP-2%	0.9962	0.9998	0.9963
		CCM-0.1%	0.9284	0.9998	0.9259
		CCM-2%	0.9959	0.9998	0.996
		CCR-0.1%	0.9284	0.9998	0.9259
		CCR-2%	0.9959	0.9998	0.996
	BACeye	BACeye	0.9912	0.9998	0.9912
		no-BACeye	—	0.9998	0.0
EM5	IB	IB-0.1%	0.245	0.9985	0.2357
	WP	WP-0.1%	0.6265	0.9998	0.6215
	CCM	CCM-0.1%	0.8511	0.9996	0.8403
	CCR	CCR-0.1%	0.8511	0.9996	0.8403
	AA	IB-0.1%	0.8157	0.9998	0.8155
		IB-2%	0.8884	0.9998	0.8856
		WP-0.1%	0.8999	0.9998	0.8959
		WP-2%	0.9818	0.9998	0.9821
		CCM-0.1%	0.533	0.9998	0.5212
		CCM-2%	0.7078	0.9998	0.6763
		CCR-0.1%	0.6042	0.9998	0.5977
		CCR-2%	0.7	0.9998	0.6664
	BACeye	BACeye	0.9225	0.9985	0.9202
		no-BACeye	—	0.9986	0.0
Pkm5	IB	IB-0.1%	0.9284	0.9998	0.9259
	WP	WP-0.1%	0.9284	0.9998	0.9259
	CCM	CCM-0.1%	0.9284	0.9998	0.9259
	CCR	CCR-0.1%	0.9284	0.9998	0.9259
	AA	IB-0.1%	1.0	1.0	1.0
		IB-2%	1.0	1.0	1.0
		WP-0.1%	1.0	1.0	1.0
		WP-2%	1.0	1.0	1.0
		CCM-0.1%	1.0	1.0	1.0
		CCM-2%	1.0	1.0	1.0
		CCR-0.1%	1.0	1.0	1.0
		CCR-2%	1.0	1.0	1.0
	BACeye	BACeye	0.9933	0.9999	0.9934
		no-BACeye	—	0.9999	0.0
PEM5	IB	IB-0.1%	0.8511	0.9996	0.8403
	WP	WP-0.1%	0.8908	0.9997	0.885
	CCM	CCM-0.1%	0.8908	0.9997	0.885
	CCR	CCR-0.1%	0.8908	0.9997	0.885
	AA	IB-0.1%	0.8573	0.9996	0.8475
		IB-2%	0.9909	0.9996	0.9911
		WP-0.1%	0.8573	0.9996	0.8475
		WP-2%	0.9912	0.9996	0.9913
		CCM-0.1%	0.8573	0.9996	0.8475
		CCM-2%	0.9914	0.9997	0.9916
		CCR-0.1%	0.8573	0.9996	0.8475
		CCR-2%	0.9917	0.9997	0.9918
	BACeye	BACeye	0.9825	0.9997	0.9825
		no-BACeye	—	0.9996	0.0

Tab. 1: The quality measures for the various machine learning algorithms. The methods and data sets are defined in Sect. 4. AA stands for All Attack types. Since no-BACeye test data contains no true positives or false negatives, the Matthews coefficient cannot be computed in this case.

best results were obtained when the classifier was trained on all the attack types at once. The cluster centers that are obtained in the training step can later be used for real-time analysis of each incoming packet.

5.2 Visualization of the flow data

In addition to detecting anomalies using machine learning methods, we can detect anomalies by constructing directed flow graphs and running the community detection algorithm on the graph, in order to obtain the large-scale structure of the graph. Figure 4 shows the difference between the flow graph of a network without anomalies (left) and the flow graph when the program BACeye is run at some point in time (right). The anomalies are detected and marked automatically by comparing the current flow graph with the original one. We compare the nodes, the edges, and the statistical properties of each edge that appears in both flow graphs.

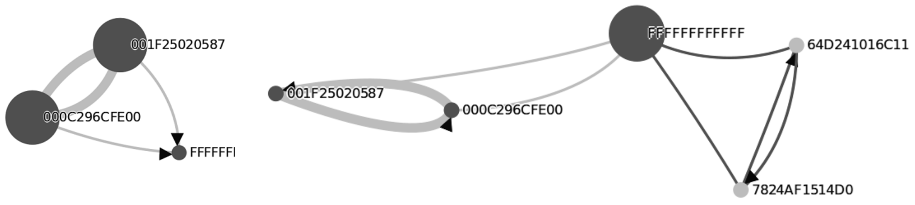

Fig. 4: Flow graph of a network without anomalies (left) and with anomalies introduced by running the program BACeye. Clearly visible are the new nodes, new flows, and the change in size of the existing nodes.

6 Conclusions and Future Work

We have shown that machine learning methods can be effective in detecting BAS network traffic anomalies. In addition, the communication flows have natural structure that lends itself to an efficient probabilistic description, which simplifies the detection of anomalies. Combined with dynamic visualization techniques, it enables the operators of BAS to detect anomalies early and act accordingly.

We focused on the following unsupervised machine learning methods: *k*-means and expectation maximization, both with or without a preceding principal component analysis. There are other algorithms that could also be used, for example one-class support vector machine, random forests, density-based clustering algorithms like CLIQUE and MAFIA, self-organizing maps using artificial neural networks, as well as a whole class of supervised and semi-supervised methods, for example *k*-nearest neighbor, Kalman filters, and Bayesian networks among others.

The selected features mostly come from the header values of the packets in our analysis, i.e., they primarily reflect the structure of the network and the kinds of communication and not so much the actual application data. We intend to include more features from the application level in the future, in order to detect anomalies in the *performance* of the hardware controlled by the BAS and not just the communication anomalies. This is important when trying to detect and thwart more sophisticated cyberattacks, where the attackers try to affect the BAS hardware.

Our plan is to test the developed methods on real-life traffic recordings, and to expand our test setup to include more devices with a more diverse set of anomalies. We also plan to implement the decision rule extraction in a way that can be directly used in a real-time traffic normalizer. Both flow-based rules and machine learning-based rules will be extracted.

Acknowledgement. This work was partially supported by the German Federal Ministry of Education and Research (BMBF) through project BARNI, project number 16KIS0148.

References

[A+07] T. Auld et al. Bayesian Neural Networks for Internet Traffic Classification. *IEEE Transactions on Neural Networks*, 18(1):223–239, Jan 2007.

[B+08] V.D. Blondel et al. Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008(10):P10008, 2008.

[C+09] V. Chandola et al. Anomaly Detection: A Survey. *ACM Comp. Surv.*, 41(3):15:1–58, 2009.

[DD11] S. Dua and X. Du. *Data Mining and Machine Learning in Cybersecurity*. CRC, 2011.

[E+06] J. Erman et al. Traffic Classification Using Clustering Algorithms. In *Proc. 2006 SIGCOMM MineNet '06*, pages 281–286, New York, NY, USA, 2006. ACM.

[H+09] M. Hall et al. The WEKA Data Mining Software. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[ISO12] ISO. Building automation and control systems – Part 5: Data communication protocol. ISO 16484-5:2012, International Organization for Standardization, 2012.

[K+15] J. Kaur et al. Securing BACnet's Pitfalls. In *Proc. 30. IFIP SEC, Hamburg*, volume 455, pages 616–629. Springer, 2015.

[M+14] E. Mocanu et al. Comparison of machine learning methods for estimating energy consumption in buildings. In *PMAPS 2014*, pages 1–6, 2014.

[MZ05] A.W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *Proc. 2005 SIGMETRICS '05*, pages 50–60, New York, 2005. ACM.

[NA08] T.T.T. Nguyen and G. Armitage. A Survey of Techniques for Internet Traffic Classification Using Machine Learning. *Commun. Surveys Tuts.*, 10(4):56–76, October 2008.

[P+14] Z. Pan et al. Anomaly based intrusion detection for Building Automation and Control networks. In *Proc. 11th AICCSA*, pages 72–77, 2014.

[Pow11] D.M.W. Powers. Evaluation: From Precision, Recall And F-Measure To ROC, Informedness, Markedness & Correlation. *J. Mach. Learning Tech.*, 2(1):37–63, 2011.

[T+15] J. Tonejc et al. Visualizing BACnet Data to Facilitate Humans in Building-Security Decision-Making. In *Proc. 3rd Int. Conf. HCI-HAS, Los Angeles*, pages 693–704, 2015.