Journal of Universal Computer Science, vol. 22, no. 9 (2016), 1203-1224
submitted: 8/7/16, accepted: 30/8/16, appeared: 1/9/16 © J.UCS

Machine Learning Methods for Anomaly Detection in BACnet Networks

Jernej Tonejc
(Fraunhofer FKIE, Bonn, Germany
jernej.tonejc@fkie.fraunhofer.de)

Sabrina Güttes[1]
(Bundeskartellamt, Bonn, Germany
sabrina@guettes.de)

Alexandra Kobekova
(Fraunhofer FKIE, Bonn, Germany
alexandra.kobekova@fkie.fraunhofer.de)

Jaspreet Kaur
(Fraunhofer FKIE, Bonn, Germany
jaspreet.kaur@fkie.fraunhofer.de)

Abstract: In recent years, the volume and the complexity of data in Building Automation System networks have increased exponentially. As a result, a manual analysis of network traffic data has become nearly impossible. Even automated but supervised methods are problematic in practice since the large amount of data makes manual labeling, required to train the algorithms to differentiate between normal traffic and anomalies, impractical.

This paper introduces a framework which allows the characterization of BACnet network traffic data by means of unsupervised machine learning techniques. Specifically, we use clustering, random forests, one-class support vector machines and support vector classifier, after a pre-processing step that includes principal components analysis for dimensionality reduction. We compare the effectiveness of the methods in detecting anomalies by performing experiments on BACnet network traffic data from various sources. We describe which of these unsupervised methods work best in specific scenarios since each method has its distinct advantages and disadvantages. In particular, we discuss which method is best suited to detect new types of anomalies (novelty detection), or which method most reliably and efficiently finds new attacks of a type that has been captured in the data previously.

Key Words: BACnet, anomaly detection, data analysis, machine learning, flow mapping, unsupervised learning.
Category: C.2.3, I.2.6

[1] **Disclaimer**: The opinions expressed in this article are the author's own and do not reflect the view of the Bundeskartellamt.

1 Introduction

A 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 a 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 monitor the whole network. Different kinds of data are sent across the BAS network, including the information about when a piece of equipment is turned on and off, changes in value when some movement is detected in particular areas of a building, changes of temperature within the building spaces, changes of temperature of air or water within the mechanical systems, valve positions, etc.

Due to the continuous growth in BAS networks, a huge amount of data is transferred through such networks every day, up to several gigabytes for moderately sized networks. This order of magnitude makes it impossible for BAS administrators to search for anomalies and identify correlations without efficient algorithms that scale well and work largely unsupervised. In the absence of such methods, most of the anomalies remain undetected and the security of the system is compromised.

This work presents a framework which performs BAS traffic analysis by means of machine learning (ML) techniques. The framework is graphically depicted in Fig. 1.

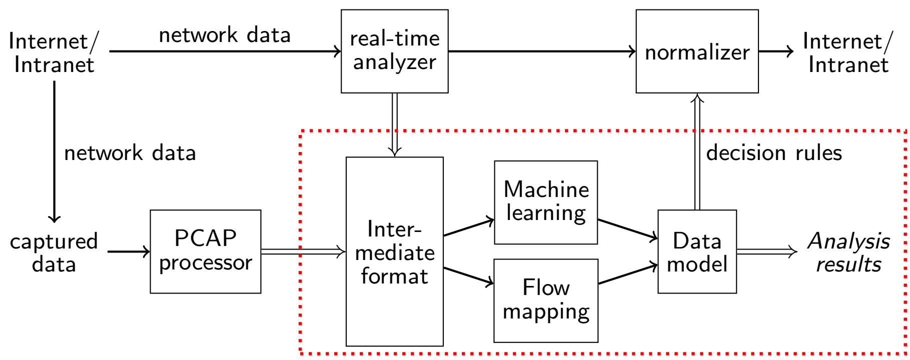

Figure 1: BAS network analysis framework. The main components of this framework are the *real-time analyzer* and the *normalizer*. Both real-time network data and the captured data are converted into an intermediate format (described in Sect. 3), which is then used to generate a set of decision rules for the 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` (version 1.12.7). Unsupervised machine learning techniques are used because they are quite efficient in performing analysis on a large volume of network data (and continuously expanding data due to addition of new devices to the network), both in terms of performance and quality. First, we preprocess the data by normalizing it and using principal component analysis (PCA) for dimensionality reduction. Then, we implement and test four of the most established machine learning methods, namely clustering, random forests, one-class support vector machines (SVM) and support vector classifier (SVC). These methods are known to be robust in general and have proven to be very useful in detecting anomalies and discovering patterns while analyzing the network data from other network protocols [Dua and Du(2011)].

The BACnet protocol is one of the data communication protocols used in BAS [ISO(2012)] and has been implemented in various products by more than 900 vendors worldwide. The framework supports the analysis of the 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 analysis in the paper was performed on BACnet/IP data, however, the methods can be applied with minor changes to any BACnet data, since only a few features come from the IP portion of the packet data, as can be seen from Fig. 3.

This paper is an extension of [Tonejc et al.(2016)]. The most significant changes are the increased number of features that were extracted from the packet fields, improving the performance of the machine learning, new machine learning methods (random forests, SVM and SVC) and more detailed performance and quality results, together with the expanded results from the flow analysis.

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

2 Related Work

A significant amount of work has been devoted to the application of machine learning and data mining techniques to solve the problem of network traffic analysis. Fortunately, several surveys exist that can serve as introduction to the topic. For example, [Chandola et al.(2009)] provides a rather extensive introduction to anomaly detection methods together with a range of possible applications. We also recommend the survey by [Nguyen and Armitage(2008)], with a slightly different focus on classification techniques, as well as [Omar et al.(2013)], which gives a great summary of techniques including a discussion of their various drawbacks and advantages.

As far as supervised learning methods to classify network traffic data are concerned, we mention for example the use of Naive Bayesian classifiers [Moore and Zuev(2005)], and Bayesian neural networks [Auld et al.(2007)]. In [Pedro Casas(2016)], the authors compare a variety of supervised methods, in particular two statistical-based detection approaches and five standard supervised ML-based approaches including random forests.

As can be seen from these references, supervised methods are often capable of performing the analysis on a very large set of data but require previous knowledge of the application domain.

Recently, learning on streams [Gaber et al.(2007)] and deep learning methods [Oliveira et al.(2016)] have been employed for anomaly detection and network traffic analysis and classification, however, more research is needed to fully evaluate the applicability of these methods to building automation network data.

In [Pan et al.(2014)], the 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, once again 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.

Unlike supervised learning methods, unsupervised techniques do not depend on labelled training data and are better at detecting new kinds of anomalies. Of course, the unsupervised learning methods also have some drawbacks compared to the supervised methods as they generally produce more false positives, have lower accuracy and can fail if the anomalies represent a significant proportion of the traffic. Some examples of unsupervised learning methods for network traffic data analysis include Clustering algorithms [Erman et al.(2006)], random forests [Dua and Du(2011)], one-class SVM [Dua and Du(2011)], and SVC, as well as maximum entropy approaches [Gu et al.(2005)].

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

In the case of BACnet/IP data over Ethernet, the data is encapsulated in a User Datagram Protocol (UDP) layer. Each packet contains communication and control values. As shown in Fig. 1, the data can either come from real-time capture

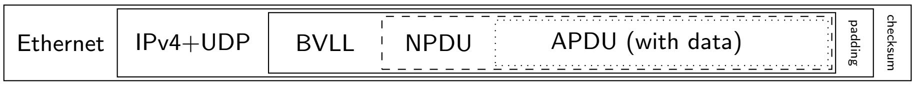

Figure 2: BACnet/IP data is contained within the UDP datagram as a BACnet Virtual Link Layer (BVLL), which contains the Network layer Protocol Data Unit (NPDU) and the Application layer Protocol Data Unit (APDU) with data. For a detailed explanation of the structure of NPDU and APDU, we refer the reader to [Kaur et al.(2015)].

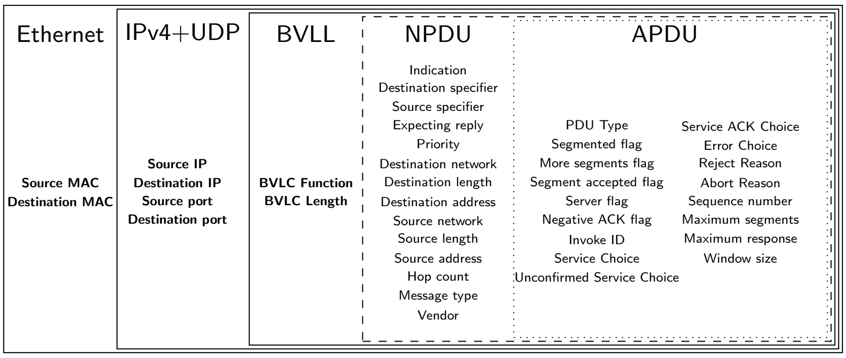

Figure 3: The fields as features, organized by the layers in which they are present. In addition to the above fields, up to 6 reserved bits are present in each packet, as well as a timestamp (saved for each packet, both in recorded `pcap` data as well as during the real-time analysis), bringing the total number of features per packet to 46.

or from 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.

For our analysis we chose the fields as suggested in [Tonejc et al.(2015)], with the addition of several fields from the Application Layer Protocol Data Unit (APDU) and the reserved fields ([Tonejc et al.(2015)] only considered the PDU Type field). From these fields we extracted the features that were used for the machine learning methods. 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.

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 all 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.

3.2 Preprocessing: normalization and dimensionality reduction

The selected fields contain all the essential features of each BACnet/IP packet, however, the network data needs to be processed before it can be used in the machine learning methods. 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. The following processing is done: the MAC addresses are split into the organizationally unique identifier 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 they are encoded using 64-bit integers. In case of a malformed packet with address longer than 8 bytes, the value is truncated 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 46 features into 54 numerical values.

The next step is the normalization, as the values in various fields have completely different scales, 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. Since some of the machine learning methods rely on Euclidean distance between the data points, we need to normalize the values. In particular, PCA and distance-based clustering methods are affected by disproportionate scale for different features. When performing the normalization, the values in each field can be mapped to the interval $[0, 1]$ using the minimum and maximum possible values for the particular field, or different weights can be assigned to different fields. For example, a higher weight can be assigned to the reserved fields as these should always be zero and if not, that should be considered very significant. The output of the preprocessing and normalization steps is the intermediate format, which is a CSV-file with 54 columns, with each packet represented as a vector in 54-dimensional space.

The running time and the performance of the machine learning methods depend significantly on the number of features. In order to reduce the number of features, we perform dimensionality reduction using PCA. This method finds a set of orthogonal directions, such that the variance along the first one is maximal and each subsequent direction has the maximum variance possible under the constraint of being orthogonal to all the preceding directions. The data is then projected onto these directions, obtaining a representation where each new variable is linearly uncorrelated with others. There is a trade-off between how much of the total variance is preserved and how much the number of dimensions gets reduced. If more principal directions are retained, more of the variance is captured and if fewer are retained, the dimension is more reduced. In practice one needs to find a balance between the two by deciding whether the running time or the accuracy is more important.

An additional issue with PCA is that components which contain no variation in values, for example, reserved bits of normal data, get discarded by PCA, as they do not contribute to the variance. This can be a problem if we later want to detect the traffic with anomalies in reserved bits, as these will no longer be present after the dimensionality reduction step. There are some solutions around this problem and they are described in more detail in Sect. 4.

3.3 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 [Chandola et al.(2009), Dua and Du(2011)]. 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 clustering, a probability distribution is assigned to each sample, indicating the probability of the sample belonging to a specific cluster. We assumed an underlying Gaussian mixture model for which the parameters need to be estimated. The parameters are then iteratively determined using Expectation steps and Maximization steps in such a way as to maximize the expectation of the log likelihood.

2. *One-class Support Vector Machine* (SVM). We focus on two variants of this machine learning method, namely Support Vector Classifiers (SVC) [Boser et al.(1992), Cortes and Vapnik(1995)], and One-class SVM [Schölkopf et al.(1999)]. In both cases, the data is first mapped to a very high-dimensional feature space using a non-linear kernel, typically a radial bump function of the form
$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|}{\gamma}}, \quad \gamma > 0.$$

In the case of One-class SVM, the algorithm tries to find a hyperplane at a maximum distance from the origin that has all the data points lying on the side away from the origin. When testing new samples the method essentially performs outlier detection.

3. *Support Vector Classifier* (SVC). In the case of SVC, after applying the same radial basis function as in the SVM approach, we train the model to obtain an optimal hyperplane that separates normal from anomalous data as much as possible.

4. *Random Forests.* A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement. To avoid overfitting, when constructing individual decision trees the random forest algorithm selects a random subset of features at each splitting. By varying the number of trees, a trade-off between running time and prediction accuracy is achieved.

There is a significant difference between the two clustering methods and the other three methods. In order for the clustering methods to detect the anomalies, the training data has to already contain the kinds of anomalies that should be detected. Random forests and the two SVM methods, on the other hand, are used for novelty detection, i.e., they allow detecting arbitrary new attacks that have not been encountered before. The random forest or the SVC/SVM can be trained on a data set that optimally contains no attacks.

For SVC and random forests in particular we need to augment the dataset (this is not needed for one-class SVM): we generate random data (by sampling from a uniform distribution independently for each coordinate) and train the algorithm to distinguish between the original training data that describes normal traffic and the synthetic data. If the trained random forest or the SVM is then evaluated on new data, anything that looks significantly different from the original training data will be flagged as anomalous.

In summary, this means that the clustering methods are optimally used to monitor attacks that were already present in the training data, whereas the other methods will detect new attacks, and new attacks only - if an attack was already present in the training data it will not be flagged in the future. Hence, optimally one should use a clustering method in combination with a method from the other set.

3.4 Network flows

Network flows have long been used for intrusion detection in IP networks, e.g., [Sperotto et al.(2010)]. 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 [ISO(2012)]. One can therefore speak of flows between BACnet devices, and each flow is defined by the BACnet addresses of the communicating devices. 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, packet size 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 [ISO(2012)] to parse the network packets. The result of the preprocessing is the intermediate format, which is simply a CSV-file with 54 columns. The same format is used when analyzing the packets in real time. The machine learning methods that were used depend on the Euclidean distance between the data points, therefore the data in each column was scaled, so as to make the values in each column lie between 0 and 1. For that, the theoretical minimum and maximum values were used for each column and not the minimum and the maximum of the actual values present.

We used two types of sources for training and evaluating our methods. The first source is our BAS lab, which contains about 20 different BACnet devices.

The traffic was recorded using Wireshark on a PC which was 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. Although the amount and variety of data is limited due to the small size of our BAS lab, it is nevertheless a relatively good approximation of an actual building automation network. 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. The day-to-day variation in traffic characteristics is in both cases low.

The second source of data is a set of artificially generated intermediate format files. As a basis we took the traffic recordings from our lab for a different day, then added various levels of synthetically generated anomalous traffic. The proportion of the anomalous traffic was 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 [Kaur et al.(2015)] and based our anomalies on the following possible attacks:

- Covert channels using different message types to convey information, specifically, *Who-Is-Router-to-network* was used to indicate bit 0 and *I-Could-Be-Router-to-network* to indicate bit 1 (CCM),
- Covert channels using reserved bits, with reserved bits in the NPDU control octet indicating whether a covert message is being transmitted and four bits of the message in the reserved bits of APDU (CCR),
- Using APDU service choices for writing properties or modifying objects (WP), and
- Using incorrect bit combinations and strange values for BACnet address lengths (IB).

The two covert channel attacks were inspired by [Wendzel et al.(2012)] and both involve only the storage channels. The timing channels were not simulated and would not be detected by the machine learning methods, however, they would most likely be detected by the flow analysis methods as they change the statistical properties of the affected flows. The DoS and router spoofing attacks were not simulated, although both should be detectable using our methods.

We generated 100,000 records for each combination, for a total of 33 ($=1 + 2 \times 4 \times 4$) 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 46 listed features from each BACnet/IP packet, performed the preprocessing steps as indicated in Sect. 3 and obtained a 54-dimensional vector. We then used *Weka* [Hall et al.(2009)] and *scikit-learn* [Pedregosa et al.(2011)] to perform the selected machine learning algorithms, as well as the dimensionality reduction. We executed the algorithms on large enough traffic recordings to capture the typical network behavior. Later, we checked for each additional packet how well it fit into the model that was learned. This allowed us to detect anomalies in the traffic as they did 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. Due to the fundamentally different modes of operation for the clustering methods (outlier detection) and the other three machine learning methods (novelty detection), the training sets were generated in different ways.

4.2.1 PCA implementation details

As mentioned in Sect. 3, one needs to be careful when performing PCA, as the features that contain no variation, for example, reserved bits in normal data, get discarded. This is especially important for the novelty detection methods, where the training is performed on normal data only. Computing PCA only on the training data and applying the same mapping to the test data is not an option, since then it is possible that the attack changes only such features of the original data which PCA essentially discards, thus making it impossible to detect the attack. The two outlier detection methods are less affected, because the training data already contains anomalous packets and no additional gain in variance is expected when looking at the test data.

In our approach for novelty detection, we perform PCA jointly on the training set and the data to be tested, then train the algorithms on the transformed training set first, followed by testing the transformed test set. Obviously this increases the time needed to do the anomaly detection since PCA and the learning phase have to be performed each time we want to test the new data.

When doing PCA it is important to include enough directions to cover a significant enough proportion of the total variance, otherwise very infrequent attacks will be overlooked as they simply will not contribute enough to the variance. We tested the quality of the anomaly detection when using a different number of principal components, specifically, we evaluated the quality when using 3, 5, 7, 10 and all components.

4.2.2 Training sets for clustering

The clustering methods can only detect anomalies that are already present in the data, so only the data with anomalies can be used for training. The data was split into a training set and a testing set as follows. 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 a chunk with anomalies, 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 attacks 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.

4.2.3 Training sets for RF and SVM

The training is performed as follows. We assume that the normal traffic has a certain structure that distinguishes it from traffic produced in an attack. We represent every network packet as a vector in $\mathbb{R}^{54}$. We then perform PCA jointly on the datasets with training data and the data to be checked for anomalies. For each coordinate of the projected training data set we find the corresponding possible range of values, which can be an interval or a set of numbers, for example $\{0, 1\}$. Next, additional data is generated by sampling each coordinate independently from the other coordinates using a uniform distribution on the respective range. Because we draw each coordinate independently, the newly generated data does not have the same structure as the original training data. This means we can now use a random forest or a SVM to recognize the structure of normal traffic by training it to distinguish between the original and the randomly generated data in the same space.

4.2.4 Implementation of ML methods

We ran the following machine learning algorithms, each with or without the preceding dimensionality reduction using PCA:

- k-means with $k = 5$ and $k = 10$ clusters
- EM with $k = 5$ and $k = 10$ clusters
- Random forests with 15 trees
- Support vector classifier (SVC)

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

Figure 4: The confusion matrix. Here 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 total number of the analyzed packets.

As stated in Sect. 3, we used the clustering methods as a 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.

The k-means and EM clustering was performed using *Weka*. For the random forests and SVC we used *scikit*, which internally uses *libsvm* [Chang and Lin(2011)] for the support vector machine calculations.

4.2.5 Validation of ML methods

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*, shown in Fig. 4.

The F_1 score is the harmonic mean of the sensitivity and precision. Given the expected nature of the anomalous data, we conclude 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 [Powers(2011)]. 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 [Blondel et al.(2008)]. 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.

4.4 Decision rule extraction

The results of the machine learning methods and the flow data analysis lead directly to decision rules that can be used to classify the traffic in real time.

Different methods lead to different types of rules. For k-means, we use the cluster centroids to automatically decide for each incoming packet whether it is anomalous or not. Similarly, we use the learned underlying model from EM to assign the probability of being anomalous. If there are no changes in the BAS, the models can be used for long periods of time without updating. When changes are made to the network structure (new devices added, services changed, ...), the models need to be updated or re-learned. Figure 5 shows an example of a rule based on k-means clustering.

1. Project the packet data using the following equations:

$$\mathbf{x} = -0.245 \cdot \mathsf{dst\_MAC\_id} + 0.245 \cdot \mathsf{NPDU.dlen} + 0.245 \cdot \mathsf{NPDU.dest.spec} + \dots$$
$$\mathbf{y} = 0.449 \cdot \mathsf{dst\_MAC\_v} + 0.449 \cdot \mathsf{IP.dst.2} + 0.449 \cdot \mathsf{BVLC.function} + \dots$$
$$\mathbf{z} = -0.674 \cdot \mathsf{src\_MAC\_v} - 0.541 \cdot \mathsf{IP.src.3} - 0.475 \cdot \mathsf{BVLC.length} + \dots$$

2. If the point (x, y, z) is closer to the point $(0.73, -0.09, -9.83)$ than to any of the points $(-4.74, 0.80, -1.42)$, $(-4.08, -0.11, 0.33)$, $(4.06, 0.07, 0.22)$, $(-3.96, -0.11, 0.07)$, then the packet is anomalous.

Figure 5: An example of a rule based on k-means with 3-dimensional PCA as the preprocessing step and $k = 5$ clusters. The field values were normalized before applying the PCA, as described in Sect. 3.2.

Extraction of useful decision rules is harder for the random forests and SVM, as the PCA step needs to be performed each time with the new data sets, followed by the learning step, which makes a direct rule extraction impossible. However, random forests performed well even without the PCA, so the learned decision trees can be stored and later directly used as decision rules, without the need to re-train.

The rules from the flow data analysis can be divided into two types: immediate exclusion rules and alert-only. We consider new flows or never before used commands for immediate exclusion. The alert-only rules are based on the learned statistical model and are triggered by any unlikely event, for example, unlikely interpacket arrival time or unlikely packet length.

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 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, EM), 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 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 with the highest and lowest MCC are shown. One-class SVM yielded no

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
		WP-2%	0.9962	0.9998	0.9963
	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	WP-2%	0.9818	0.9998	0.9821
		CCR-0.1%	0.6042	0.9998	0.5977
	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
	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
		CCR-2%	0.9917	0.9997	0.9918
	BACeye	BACeye	0.9825	0.9997	0.9825
		no-BACeye	—	0.9996	0.0
PRF	no-BACeye	IB-0.1%	1.0	1.0	1.0
		WP-0.1%	1.0	1.0	1.0
		CCM-0.1%	1.0	1.0	1.0
		CCR-0.1%	1.0	1.0	1.0
		BACeye	0.9926	1.0	0.9999
PSVC	no-BACeye	IB-0.1%	0.7702	0.6636	0.9978
		WP-0.1%	0.6174	0.4486	0.9966
		CCM-0.1%	0.9826	1.0	0.9996
		CCR-0.1%	0.985	1.0	0.9997

Table 1: The quality measures for the various machine learning algorithms, denoted as follows: km5 is k-means with 5 clusters, EM5 is expectation maximization with 5 clusters, PRF is random forest and PSVC is the support vector classifier. A leading P indicates that PCA was performed first. The data sets are defined in Sect. 4.1, 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 these cases.

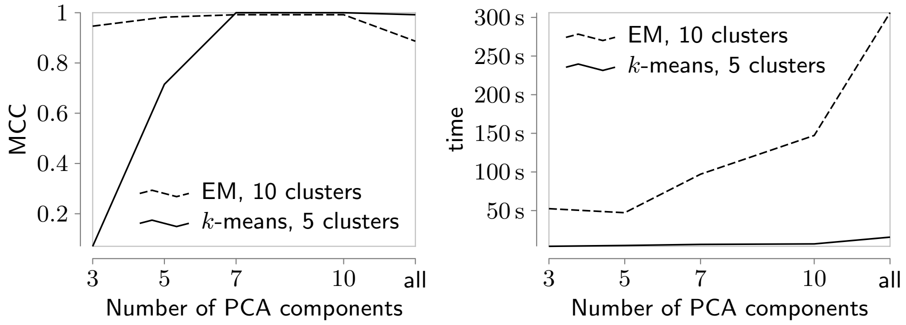

Figure 6: The Matthews correlation coefficient (left) and the running time (right) as a function of the number of PCA components. In both cases, the results are for EM with 10 clusters, using all attack types for training and testing against the data with 1% of packets with incorrect bits. The algorithm were executed on a desktop computer with Intel i7-3770 CPU and 32 GB RAM.

useful results, hence these results are not included in the table. The table clearly shows that k-means and random forests perform best. These two methods are best used simultaneously, as they are complementary in nature.

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. The duration of the training step depended most on the chosen method. Random forests and k-means were both relatively fast, lasting less than 20 seconds in all cases. For EM and SVC, the speed depended a lot on the number of PCA components. Figure 6 shows the timing results as a function of the number of PCA components, as well as the quality of the learning. For SVC we also had to reduce the training set to just 10.000 samples, as the algorithm failed to complete in a reasonable time with more samples.

Testing was in all cases fast. When using PCA, there is always a tradeoff between the performance and quality. In most cases, 5 components provided the optimal balance. From the table it is clear that by far the best methods are k-means with $k = 5$, preceded by PCA and random forests. The best results for clustering methods were obtained when the classifier was trained on all the attack types at once. The cluster centroids that were obtained in the training step were later used for real-time analysis of each incoming packet as part of the decision rules.

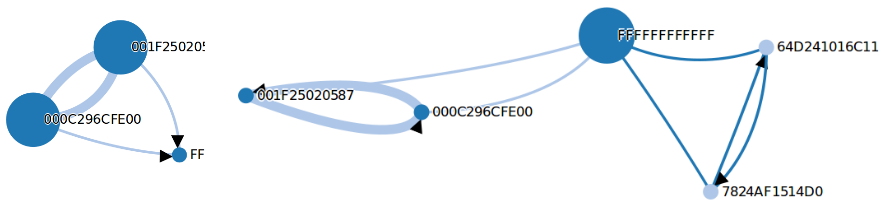

Figure 7: Flow graph of a network without anomalies (left) and with anomalies introduced by running the program BACeye (right). Clearly visible are the new nodes, new flows, and the change in size of the existing nodes. Taken from [Tonejc et al.(2016)].

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 7 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 (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.

Figure 8 shows a different network, with a star topology. The statistical properties of the flows for the three marked nodes are shown in Fig. 9. In particular, the interpacket arrival time and the packet length are analyzed.

It is clear from the figures that in order for the attacker to remain undetected, a very limited choice of packet lengths and interpacket timings is available.

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, expectation maximization, random forests and one-class support vector machines, all with or without a preceding dimensionality reduction using the principal component analysis. There are other algorithms that could also be used,

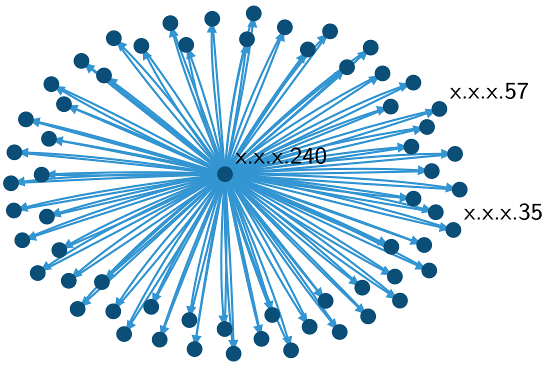

Figure 8: Another example of a flow graph for a simple network where all devices communicate with one central node. The details of the network traffic for the named nodes are presented in Fig. 9. This is a fairly typical arrangement when devices are being managed from a central node.

for example 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 network traffic comes naturally as a stream of data, so methods such as CluStream, ClusTree and StreamKM++, designed for learning on streams, could also be used. In our future work we plan to test these methods on the network data streams (after the preprocessing step) with existing tool MOA [Bifet et al.(2010)]. Additionally, a combination of several machine learning methods could yield better results, this is also one of the things we plan to investigate in the future.

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.

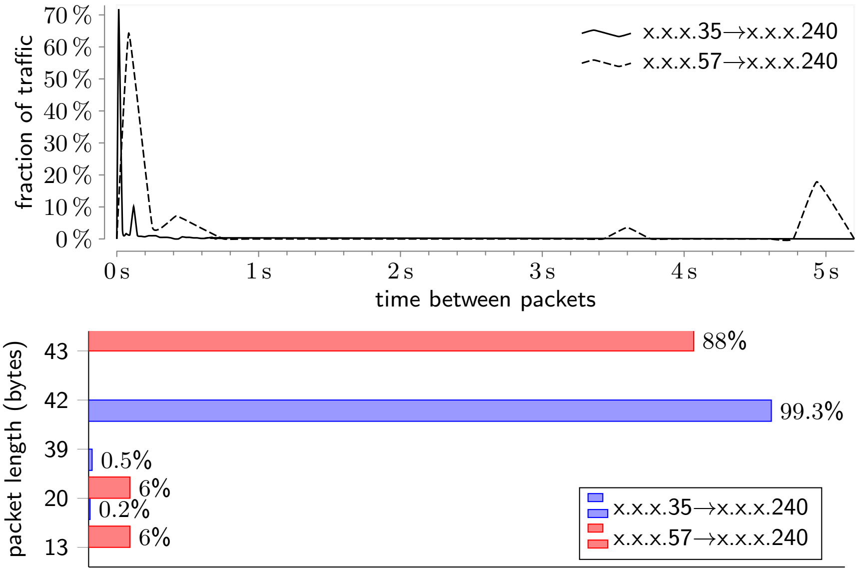

Figure 9: The distribution of the interpacket arrival times for the flows x.x.x.35→x.x.x.240 and x.x.x.240→x.x.x.57 (above) and packet length distribution for the same flows (below). The packet lengths are fairly representative for BACnet networks. The interpacket arrival times depend a lot on the configuration of the devices in a given network, specifically, the timings are more predictable when a central node polls the data as opposed to the devices reporting the values by using the Change-Of-Value subscriptions.

Acknowledgements

Parts of the paper were initially published in the Proceedings of Sicherheit 2016, under the title “Detecting anomalies in BACnet network data”.

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

References

[Auld et al.(2007)] T. Auld, A. Moore, and S. Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on Neural Networks*, 18(1):223–239, 2007.

[Bifet et al.(2010)] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

[Blondel et al.(2008)] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008(10):P10008, 2008.

[Boser et al.(1992)] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT ’92, pages 144–152, New York, NY, USA, 1992. ACM.

[Chandola et al.(2009)] V. Chandola et al. Anomaly detection: A survey. *ACM Comp. Surv.*, 41(3):15:1–58, 2009.

[Chang and Lin(2011)] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2: 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Cortes and Vapnik(1995)] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

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

[Erman et al.(2006)] J. Erman et al. Traffic classification using clustering algorithms. In *Proc. 2006 SIGCOMM MineNet '06*, pages 281–286, New York, NY, USA, 2006. ACM.

[Gaber et al.(2007)] M. M. Gaber, A. B. Zaslavsky, and S. Krishnaswamy. A survey of classification methods in data streams. In C. C. Aggarwal, editor, *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*, pages 39–59. Springer, 2007.

[Gu et al.(2005)] Y. Gu, A. McCallum, and D. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, IMC '05, pages 32–32, Berkeley, CA, USA, 2005. USENIX Association.

[Hall et al.(2009)] M. Hall et al. The weka data mining software. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

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

[Kaur et al.(2015)] J. Kaur, J. Tonejc, S. Wendzel, and M. Meier. Securing BACnet's pitfalls. In *Proc. 30. IFIP SEC, Hamburg*, volume 455, pages 616–629. Springer, 2015.

[Moore and Zuev(2005)] A. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proc. 2005 SIGMETRICS '05*, pages 50–60, New York, 2005. ACM.

[Nguyen and Armitage(2008)] T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Commun. Surveys Tuts.*, 10(4):56–76, 2008.

[Oliveira et al.(2016)] T. P. Oliveira, J. S. Barbar, and A. S. Soares. Computer network traffic prediction: a comparison between traditional and deep learning neural networks. *International Journal of Big Data Intelligence*, 3(1):28–37, 2016.

[Omar et al.(2013)] S. Omar, A. Ngadi, and H. H. Jebur. Article: Machine learning techniques for anomaly detection: An overview. *International Journal of Computer Applications*, 79(2):33–41, 2013.

[Pan et al.(2014)] Z. Pan et al. Anomaly based intrusion detection for building automation and control networks. In *Proc. 11th AICCSA*, pages 72–77, 2014.

[Pedregosa et al.(2011)] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Pedro Casas(2016)] A. D. Pedro Casas, Pierdomenico Fiadino. Machine-learning based approaches for anomaly detection and classification in cellular networks. In *Proceedings of the 8th International Workshop on Traffic Monitoring and Analysis*, TMA 2016, pages 1–8. IFIP Open Digital Library, 2016.

[Powers(2011)] D. Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *J. Mach. Learning Tech.*, 2(1):37–63, 2011.

[Schölkopf et al.(1999)] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, et al. Support vector method for novelty detection. *NIPS*, 12:582–588, 1999.

[Sperotto et al.(2010)] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An overview of IP flow-based intrusion detection. *IEEE Communications Surveys and Tutorials*, 12(3):343–356, 2010.

[Tonejc et al.(2015)] J. Tonejc, J. Kaur, A. Karsten, and S. Wendzel. Visualizing BACnet data to facilitate humans in building-security decision-making. In *Proc. 3rd Int. Conf. HCI-HAS, Los Angeles*, pages 693–704, 2015.

[Tonejc et al.(2016)] J. Tonejc, J. Kaur, and A. Kobekova. Detecting anomalies in BACnet network data. In M. Meier, D. Reinhardt, and S. Wendzel, editors, *Sicherheit 2016: Sicherheit, Schutz und Zuverlässigkeit, Beiträge der 8. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 5.-7. April 2016, Bonn*, volume 256 of *LNI*, pages 253–264. GI, 2016.

[Wendzel et al.(2012)] S. Wendzel, B. Kahler, and T. Rist. Covert channels and their prevention in building automation protocols: A prototype exemplified using BACnet. In *2012 IEEE International Conference on Green Computing and Communications, Conference on Internet of Things, and Conference on Cyber, Physical and Social Computing, GreenCom/iThings/CPSCom 2012, Besancon, France, November 20-23, 2012*, pages 731–736. IEEE Computer Society, 2012. ISBN 978-1-4673-5146-1.