ECEN 404 - Electrical Design Lab 2

Semester: Fall 2020

Team 3 Final Report:

Anomaly detection in BACnet protocol systems

Team Members: Sofian Ghazali

Muhammed Zahid Kamil

Rahul Balamurugan

Project Mentors: Dr. Hussein Al Nuweiri

Dr. Ferdous Wahid

Mr. Salah Hessien

Submission Date: 26/04/2020

"An Aggie does not lie, cheat, or steal, or tolerate those who do."

TABLE OF CONTENTS

ABSTRACT

Building Automation Networks/Systems (BAN/BAS) are networks that control and monitor utilities and equipment in smart buildings. BACnet is a widely used device-to-device communication standard that enables building automation and allows for interoperability between devices from different vendors. The issue is that the most commonly used implementations of the BACnet standard, BACnet/IP (Internet Protocol) and BACnet/WS (Web Services) are entirely unencrypted. As more and more BACnet devices are being exposed to the internet, the risk of cyber-attacks is becoming increasingly prominent in addition to the always present risk of more physical attacks against building networks. Our solution is to create a continuously adapting robust anomaly detection mechanism to protect BACnet systems from cyber-physical threats. The project aims to understand and emulate a real local area BAN implementing the BACnet/IP standard, and create an anomaly detector for the same using machine learning. The detector will be trained on the emulated network itself and tested by injecting synthetic attacks into the system. This report details the literature review and customer needs analysis done to arrive at the initial design, performance and market analysis, system flowchart, a detailed discussion of the system design and an update on the progress of the project.

CHAPTER I

INTRODUCTION

With the advent of the smart grid and distributed energy resources, most office spaces and buildings in urban areas have adopted some measure of automation. The demand for devices to control and automate the control of building utilities has led to many competing vendors with KNX, Lon works and Modbus being a few of the more popular ones. In order to ensure interoperability, consumer convenience, and a fairer market, the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) developed the BACnet Standard in 1955. The BACnet Standard (ISO 16484-6, 2003) has evolved to be one of the most widely used communication protocols over the years due to the flexibility it affords to the management and application layers of the Building Automation Networks. Recently, there is a prevalence of users connecting their Building Networks to the Internet for remote access and monitoring, using the web services implementation of the BACnet Standard.

However, there are clear risks associated with increasing internet connectivity extended to building equipment. The current number of Building Automation Devices, another term extended to BAS systems, amounts to 48,112 [18]. This exposure is mainly since IP addresses of the building facilities using web services can be discovered via search engines such as censys.io and shodan.com that provide a list of such devices and relevant statistics. There are two types pertaining to BACnet protocol: BACnet/SC (secure connect) and BACnet/IP (Internet Protocol), of which the latter is the standard implementation used. BACnet secure ensures messages are encrypted and is a secure form of communication. However, it has interoperability issues with other devices []. On the other hand, BAC/IP allows for ease of interoperability but at a cost- the messages transmitted are in plaintext and completely unencrypted. This is made worse when these devices are exposed to the internet, making them more vulnerable to cyber-attacks.

The objective of this project is to introduce an automated, robust intrusion detection algorithm that detects the presence of anomalous data in BACnet traffic. The scope of our project does not extend towards securing the BACnet communication such as creating a firewall or secure encryption algorithm. Our task starts from the onset of a cyber-physical attack; We notify the management layer of an ensuing attack without exacerbating the situation. We aim to have our algorithm detect any anomalies in a responsive manner and absorb new datasets to become more intelligent. The focus is on reducing the labelling work associated with each data packet and improving accuracy by applying clustering techniques to create a unique program that detects novel or unusual data. To address our objectives, we gathered data necessary for the training phase of the algorithm and investigated the algorithms that can best model the data behaviour and to achieve an anomaly detector system. Moreover, we took inspiration from existing research into anomaly detection in building networks and conducted a customer needs survey to understand our project needs from the perspective of security experts and the potential users of the BANs.

Literature Review

Current solutions and research have been focused on using anomaly detection and Intrusion Detection Systems (IDS) to improve system resilience and device-level security management. Some existing anomaly detection methods are the timing-based detector for cyber-physical systems (CPS) by Zimmer et. al [14], and the finite states-based detector for Modbus networks by Goldberg et. al [15]. The major challenge faced by researchers in anomaly detection is to minimize the number of false alarms and improve their efficiency. In this regard, Zheng and Reddy, researchers at TAMU College Station, formulated a THE-driven anomaly detector for BACnet that uses frequency analysis [16].

Zheng et al. designed and developed this detector in 2017 based on a novel network traffic classification model they called the ‘THE-driven classification method’. In this technique, all communication traffic is classified into 1. Time-driven, 2. Human-driven, and 3. Event-driven categories. Time-driven traffic is

normally generated by scheduled control programs that trigger service requests according to different timers. Such traffic presents time regularity and is not affected by real-time events of the network. Human-driven traffic includes requests that are directly generated by humans or through control programs. Event-driven traffic includes service requests that are not generated by timers or humans, but as responses to certain programmed events. The classified network traffic, in this method, is then parsed for suspicious data packets and all such anomalies are flagged.

Although Zheng et. al's detector model is novel and effective, current rate of increase in data size of BACnet networks [17] warrant a need for an automated and unsupervised system. We planned to follow up on this research by using machine learning techniques instead of Fourier frequency analysis to capture normal data patterns and detect anomalous traffic. Previous research in the usage of machine learning to detect network anomalies is rather extensive. In this regard, we studied the work of Tonejc et. al [20] in characterizing BACnet network traffic data by means of unsupervised machine learning techniques. The model was trained from two different datasets. One of which is from two days' worth of network traffic from a BAS lab setup that contains about 20 different BACnet devices. The other was a dataset from artificially generated network traffic by obtaining network traffic from a different day and adding variations in the synthetically generated anomalous traffic. Clustering methods such as k-means are used to detect known attacks from the training data. Random Forests and Support Vector Machines (SVM) were used to detect new attacks.

Thus, our aim became to use the classification system proposed by Zheng et. al and the methods discussed by Tonejc et. al to create our anomaly detection system. Our project would fall under the category of un-supervised machine learning techniques and we decided to focus on temporal and human aspect of BACnet traffic. Unsupervised seems was an appropriate choice because we do not have labels to feed into our algorithm, which would in this case be anomaly or non-anomaly data. Therefore, clustering based approaches are very useful in unsupervised learning where it can learn on its own to discover hidden patterns in data [6]. Based on these objectives, our option was inclined towards using the K-Means algorithm as it is an efficient algorithm to work on datasets without labels and will find patterns for us. Our data will be split into numerical (frequency-based) and categorical (feature-based). In the case of numerical data, the K-means algorithm was feasible since this depended on numerical attributes to calculate the Euclidean distance between the values and find the average. For categorical datasets, we decided to work with K-modes algorithm, which is an alternative to k-means that works well with categorical data. The technical details of the K-means & K-modes algorithm and how it was implemented for our project will be explained further in the flowchart and system design section.

Motivation & Customer Needs Analysis (See Appendix for the survey questionnaire and results)

As novel cyber-physical attacks are on the rise, they pose a great risk to the integrity of BAS equipment. These zero-day attacks can only be detected by adaptive methods like Machine Learning. Hence, we need a novel detection algorithm using the concepts of ML to act as the first stage of a comprehensive security system in BAS systems implementing the BACnet protocol. With this motivation in mind, we started our analysis by first investigating customer needs to understand their needs and feedback for such a product in their building systems. To perform this, we implemented survey tools that will be discussed below.

The results of a survey elaborates on multiple foci - the need for security in unencrypted IP- based BAS, cyber-physical security features that customers believe they need/don't need, what information the audience believes could be obtained from a hacked BAS, how someone could possibly go about getting access to BAS controls, and finally, their opinions on the intrusion of privacy due to data collection for security purposes. The audience for the survey was multifaceted- including subject experts, Building Service operators, BACnet users (via User Forums), Building Automation companies, and the general public (potential users of BAS). From the survey, we wanted to get an idea of what attacks and devices we would need to model to emulate a real BACnet system, and what features our anomaly detection algorithm would need to have to appeal to a wider customer base. Additionally, we wanted to find out how feasible automated detection of

cyber-physical threats was in the eyes of experts in the field.

The methods used in the project are surveys aimed towards two different audiences. One for the general public assuming no/partial knowledge of security on Building Automation Systems. The other (Technical survey) is for the representatives of companies, professors or any individuals who have knowledge about Building Automation Systems and BACnet. The survey forms sent to the companies were sent to individuals with prior knowledge of BACnet security and their input will be needed on how to improve BACnet security. We also reached out to people in online BACnet community forums, specifically 'The BACnet Institute and the BACnet Protocol' Stack/Discussion on sourceforge.net.

From the responses of the general survey, we found out that the public realizes the importance of security in buildings and therefore supports the goal of our project to detect anomalies in network traffic. Most of the public also believe that the buildings they go to school, work or live have building automation systems. This shows the versatility of BACnet protocol and we aim to implement this model using 4 towers (raspberry pi stacks) which could represent university buildings, apartments, company buildings, hospitals etc. simply by varying the modelled sensors, services and traffic patterns. The responses from the general survey also showed that the public believe that BACnet devices need security. This shows that the public is aware of the danger in these devices being vulnerable to attackers and which in turn makes us develop BACnet devices to be secure and require authentication to access these devices. However, from the technical survey we received responses that they would rather not have too many authentication procedures as this could become too cumbersome, and to avoid collecting too much data. Thus, a middle ground solution would be to allow multiple levels of access with general users having only a single authentication upon connection and those with higher authority to access the control layer having to go through stricter authentication mechanisms. Therefore, any breach in the building can potentially be a breach of trust through one of the buildings' employees. We also wanted to know how the public would react to data being used for cyber security purposes. With data collection being a sensitive topic currently, almost 1/4th of the responses believe that it is a violation of their privacy. This result was useful is helping us decide the level of data collection we should undertake.

Regarding the veracity of the results themselves, it must be mentioned that this survey would have been served much better by larger numbers. It can be assumed that there is uncertainty in data measurement in the case of objective questions. However, since we based our observations and analysis on the general trends rather than concrete numbers, the probability of us making erroneous inferences should be much lower. If the number of responses had been in the range of 500-1000, this survey could have been a tool to assess the state of BAS security and public awareness regarding the same.

CHAPTER II

BENCHMARKING

In the course of envisioning our final product, we discuss in this section the existing solutions pertaining to our project and highlight some criteria that can be used to evaluate the performance of our project with that of existing solutions. Many advancements have already been made in the field of anomaly detection and we aim to explore how well other solutions perform and how we can benefit from it.

Performance Criteria

Public Safety and Privacy Protection

This is an essential criterion for our project since the main idea behind designing a predictive anomaly detection algorithm is to protect users from theft and hacking in building systems. BACnet protocol lacks in implementing effective security measures because hackers can access any building automation systems through a public domain. Our goal is to alert the building system of a potential breach without human intervention. Our metrics of comparison will be based on the level of protection each of the products offers a BAN overall. A score of 1 implies that the method is ornamental at best and a score of 5 means that the product is enough to defend against any threat to the system, be it a cyber-attack or a physical threat.

Responsiveness

Although anomaly detection methods might prove useful to secure BACnet systems, quick responsive approach to real-time attacks is valuable and of great challenge. The anomaly detection algorithm we are trying to implement must train with multiple sets of data to recognize anomalous data patterns at a fast-enough rate. A delay of just a few minutes might prove fatal to cyber-physical systems, resulting in financial loss. A score of 1 in this metric indicates that the method is too slow to deal with most threats to the system, and the mechanism leaves the system wide open to external access. On the other hand, a score of 5 implies that the system is fast enough to deal with all possible threats and prevent any and all information leaks. Note that since we were unable to test each of the mentioned products first-hand, this criterion is only a rough measure of overall responsiveness based off test results reported by the creators of the techniques.

Adaptability

Cyber threats are evolving at a very fast rate with many doing so in real time. Due to this, the capability of the security tool to adapt itself to the threats is very relevant in assessing the overall effectiveness of the method. A score of 1 in this implies that the product is fixed and cannot be updated to deal with future or evolving threats. A score of 5 means that the system can evolve with the threats perfectly, and updates very frequently. Similar to the above criterion, Adaptability cannot really be measured without exhaustive first-hand testing. However, this metric serves the objective of comparing the mentioned products loosely based on their descriptions.

Comprehensiveness

This criterion indicates the range of anomalies or threats detected by the technique as a function of the number of uncorrelated network parameters monitored by the detection mechanism. A score of 1 implies very poor range, with the tool only analysing a single network parameter. 5 indicates an all-encompassing range, with the tool analysing all possible parameters. Note that while the number of parameters measured may not accurately indicate the detection range of all devices, the majority of detection techniques can be successfully classified based on this criterion.

Measures for failure

This criterion is a measure of the product's capability to cope with system failures and to support the system in such situations. A score of 5 in this implies that the program has robust fallback and backup routines and can aid in system recovery. On the other hand, a score of 1 indicates the absence of any failsafe or recovery options in the program. A middling score in this regard indicates good failsafe options, but no recovery support.

Economics

The impact of the product on the economy is a major factor which we thought should be taken into consideration when designing our solution. The market for anomaly detection tools is not that lucrative for developers as of right now, mainly because of the low demand for such advanced means. However, security tools for BACnet are very much in demand due to the increased exposure of building networks to the internet. This way, we were able to rate the impact of each product on the market from 1 to 5, with 1 meaning it doesn't have much of one, while 5 means the product was able to dominate the market.

Cultural

Since our solution requires the periodic collection of large volumes of data (which may sometimes be personal), it was reasonable to expect that people from more private cultures could react adversely to the product. However, security tools in general do not have much of a cultural impact, and thus this criterion did not come into play during the comparison.

Global

BACnet is an international standard and is implemented in its varied forms worldwide. As a result, the impact of developing a security tool for BACnet on the global stage is quite high. Such methods if developed could reflect quite positively on the country as the research would be beneficial regardless of the place.

Environmental

As such, BACnet security does not have any direct impact on the environment. However, if the implementation of higher security measures leads to increased automation of buildings, and thus better energy efficiency across the board, this could lead to a net positive impact on the environment. Note that since this is purely speculation on our part, we did not use this as a criterion for comparison.

Product Comparison

Mentioned below are a few products like our proposed solution and how they compare:

- AC2000 BACnet Interface
- THE-driven anomaly detector by Zheng et al.
- Timing based detector for cyber-physical systems by Zimmer et al.
- Finite-state based detector for Modbus detectors by Goldberg et al.

AC2000 interface

This interface from Tyco Security Products enables alarms to be sent in BACnet protocol to third party systems including building management systems, HVAC, fire and any other systems that support BACnet communication. The interface is bi-directional, allowing for both the sending of AC2000 alarms and the receipt of third-party Change of Value (COV) BACnet messages, which can then be displayed on the AC2000 Security Hub alarm and event management application. As a security tool, this interface is quite versatile and can detect many common threats to BANs. However, from the parameters of data it monitors, the tool would be unable to do much in the face of a Denial of Service attack or a Man-in-the- Middle attack. It would also require an additional device The former depends exactly on the bandwidth being occupied by COV messages or alarms to break into BACnet systems, while the latter would in all probability go unidentified.

THE-Driven Anomaly Detector

As mentioned in the literature review, this detector was a major inspiration for the project. In terms of performance, the testing information for this as reported by Zheng et al. put its accuracy above 96% across all the tested attacks. It also purportedly has a near 100% anomaly capture rate with the exception of DoS attacks, which can be identified via other network volume-based detection techniques.

Timing-based detector

This detector works on the principle of utilizing information from static timing analysis to identify unauthorized instructions in real-time cyber-physical environments. In the case of building automation systems, the timing bounds in code segments are said to be easily available, thereby facilitating this method. The paper by Zimmer et al. [14] describes this mechanism which works by checking those bounds and implements the detection techniques either by itself in a self-checking manner, or through the operating system scheduler.

Finite-states based Detector

Goldenberg et al. [15] modelled a detector meant for the Modbus network protocol (and can be used in BACnet as well) based on identifying the unique deterministic finite automata “state” (DFA) of each individual channel between a HMI-PLC (Human-Machine Interface & Programmable Logic Controller) pair. The DFA is a finite-state machine that can accept or reject a given string of symbols and jumps from one state to another. It takes around 100 captured messages to identify the DFA for a given channel. This method purportedly is highly sensitive while having very low false positives. The paper mentions that Goldenberg et al. found zero false alarms over 111 hours of continuous operation.

Table 1 (below): General Comparison of each solution

1. General comparison:

	AC2000 Interface	**Timing-based detector**	**Finite-state detector**	**THE driven anomaly detector**	**Our Solution**
Technique used	Receive third- party Change- of-Value notifications and monitor all events via the security hub.	Statistical Attributes of data packets (Mean, Range)	Construct a deterministic finite automaton (DFA). Flag anomalies according to periodicity	Autocorrelation, THE Classification, and Fast Fourier Transform	THE-Driven traffic classification with semi- supervised ML techniques
Accuracy (%)	Information Not Available	>99%	>99%	~96%	-Can't be measured-
Advantages	User-friendly patented application to configure required BACnet alarm or event outputs.	Works both by itself and also through the program scheduler	Very high accuracy and highly sensitive. Able to access the deeper network layers	Detector can be used online.	Online, Adaptive Traffic Classification for better understanding of traffic source
Disadvantages	Not open to novel detection mechanisms.	Depends entirely on timing bounds.	Only two statistical parameters measure the anomalies.	Frequency analysis needs to be done manually; not accurate in more dynamic real life situations	Lack of data = less sophisticated algorithm
Limitations	Only limited to certain events or alarms that take place.	Only considers the timing bounds of a data packet. Data packets emerging from human driven activities will not be distinguished	Performance degrades for multi-period traffic patterns - slower traffic patterns increase false positive rate.	Impossible to have a 0% false positive rate. Classifies network traffic into only three different categories.	Impossible to have a 0% false positive rate Works well with more data fed into the system. Unable to introduce third party vendor detection.
Standards	Follows their own AC2000 standards therefore can be considered credible	Complies with most of ASHRAE's standards from NIST	Complies with most of ASHRAE's standards from NIST	Complies with most of ASHRAE standards by training the model to recognize most of the standard attacks.	The testing model should be able to recognize most of ASHRAE's security assessments defined from the NIST

Table 2 (below): Comparison of each solution according to Performance Criteria

2. Performance comparison (All scores are ranked from 1-5, with 1 being the least and 5 being best):

	AC2000 Interface	**Timing- based detector**	**Finite-state detector**	**THE driven Anomaly Detector**	**Our Solution**
Responsiveness	2- Only known threat patterns identified, needs to be manually updated	4- Real time response to threats are very good as this measure's differences in code time bounds	4 - Speed varies according to periodicity of traffic patterns. Multiple traffic patterns mean algorithm slows down.	4- Low latency for detection of most traffic anomalies.	Speed of algorithm depends mainly on the amount of data the algorithm has analyzed.
Adaptability	1- Not easily adaptable to new intrusion detections	3- Can adapt to multi-periodicity traffic patterns, but additional work needs to be done to achieve good performance. This includes testing on new data patterns.	4- Highly sensitive mode of detection means the algorithm can adapt to subtle changes in data packets.	4- Allows the detection of common types of attacks. Depends on labelled data.	5- Can adapt to new attacks due to unsupervised aspects and also better at identifying threats due to supervised learning methods
Comprehensiveness	2- Only known threat patterns identified, needs to be manually updated	3- Only applicable for time driven anomalies.	5- Highly sensitive to all changes in the interface-controller channel, analyzes deeply into the network	4-Trained to recognize different intrusion detection mechanisms through the BAS networks and synthetic generated attacks.	5- Should be able to detect any anomalous data packets due to comprehensive classification of normal traffic
Measures for failure	3- Alert management about successful attacks, infiltrators, and other network data red flags.	1-Difficult to check for failure as some breaches may occur undetected due to limited detection parameters	4- Any anomalies in the channel are found easily due to changes in the DFA of data packets	3 - Methods to alert user can be added to the mechanism. Built-in measures not available.	1- We have not planned any failsafe as of now

Table 3 (below): Comparison of Macro-Level parameters

3. Macro-level comparison:

	AC2000 Interface	**Timing-based detector**	**Finite-state detector**	**THE driven anomaly detector**	**Our Solution**
Economic	A pretty competitive product on the market. Similar products are some of the most widely used notification software in current BAS.	Generally applicable across BACnet and SCADA networks. Due to its scalability and high efficiency, this method had a high impact on the building networks market.	Designed for Modbus devices, this is nevertheless compatible with general networks. Due to its more focused development, this may have a higher impact in that market	The simplicity of the algorithm and high efficiency, with its ability to be used across BACnet systema, it is anticipated that this method if released into the market could have a high impact.	Could potentially be in demand depending on the trade-off between performance and data collection volume/frequency.
Global Compatibility	Compatible with most BAS systems	Compatible with SCADA systems	Compatible with SCADA systems	Compatible with SCADA and BACnet systems	Compatible with SCADA and BACnet systems

CHAPTER III

FUNCTIONAL MODELLING

We developed a functional structure of our project design using a flowchart model. It was a great tool for us to convey our ideas through a visual depiction of the processes that go into designing our project. It allowed us to elucidate the intricate details of our project and aid in team productivity. For a complicated project that involves huge data collection and running algorithms, it would've been difficult for us to comprehend the number of components that needs to be implemented to make this algorithm run successfully. Using graphical data representations techniques like flow-charts and black boxes, we can now use it as a reference point to review and understand the components throughout the duration of our senior design project. In addition, using flowcharts allowed us to consider troubleshooting issues that may occur during the project. This includes recognizing false positive results and how we can handle it by making decision boxes to determine flow of process. This flow leads to a possible solution and this assignment has allowed us to consider the possible solutions proactively, thereby allowing us to be productive in the future.

The general program for an unsupervised machine learning based anomaly detector involves training the detector to recognize normal data values and patterns. During implementation, the model can then classify new inputs as normal if they are like the training set or anomalous if they are dissimilar (beyond a certain threshold). Hence the inputs to any ML anomaly detection program are the training data and the testing data, and the outputs are the result of the comparison. Following a similar method, the black box functional model of the system is:

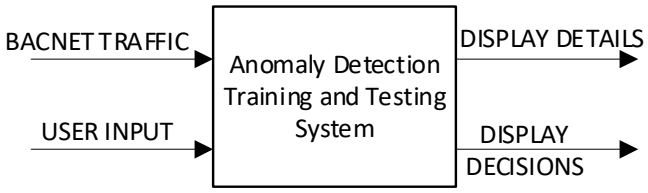


Figure 1: Black Box Design of our system

Input/output Design:

The inputs to the system are the normal training data and testing data. The training data for our model is the BACnet traffic and the testing set is a user created .csv file in the appropriate format for testing. In practice, the testing data should be network traffic data as well. However, as the model will not be tested real-time, the input is created by the user. Data collection for the training data set was first challenge we experienced. Usually for BACnet systems, there is a scarcity of data available due to building operators denying access to the BACnet traffic. This is mainly due to security purposes and so we had to think of ways to overcome this problem. The data set we trained on consisted of 2 hours of normal building in Qatar University, and was provided by our mentor. The output of the system is the decision for each data unit in the testing set, i.e., whether it is anomalous or normal. The details of the anomalies encountered are also output by the system. The output method is explained in chapter 5.

Elaborating functions:

Modelling the system according to our needs and adding the basic function blocks, the system model becomes:

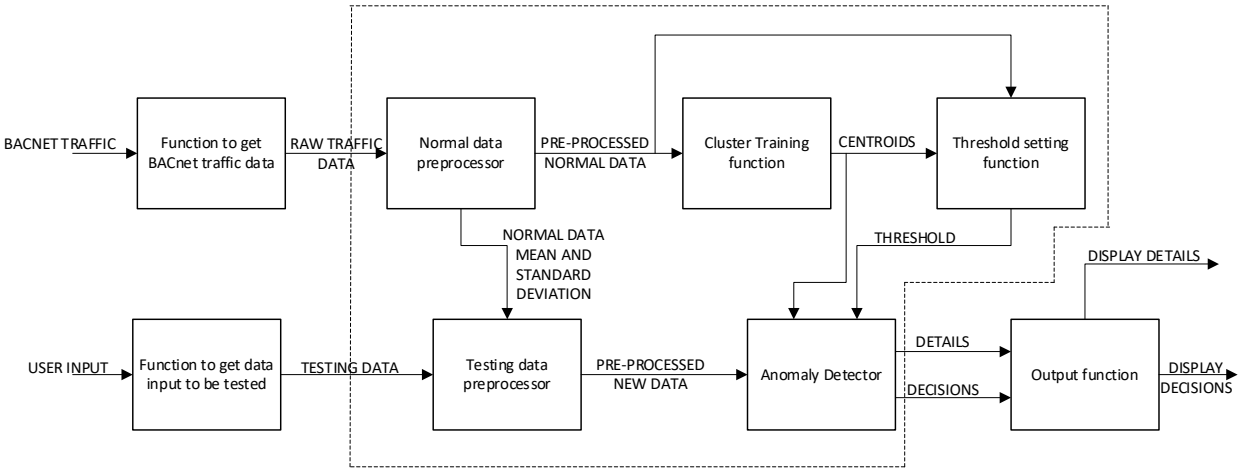


Figure 2: Detailed Information Flow in a Black Box

In the above figure, the basic data flow for the system is shown. The portion of the functional model enclosed inside the dashed boundary can be further decomposed further as shown in Fig.3. The main reason for the different treatment that can be observed from the figure below for message patterns and frequency patterns is the differing natures of the data. However, the two follow parallel steps and the results of the anomaly detectors are aggregated and are output as one.

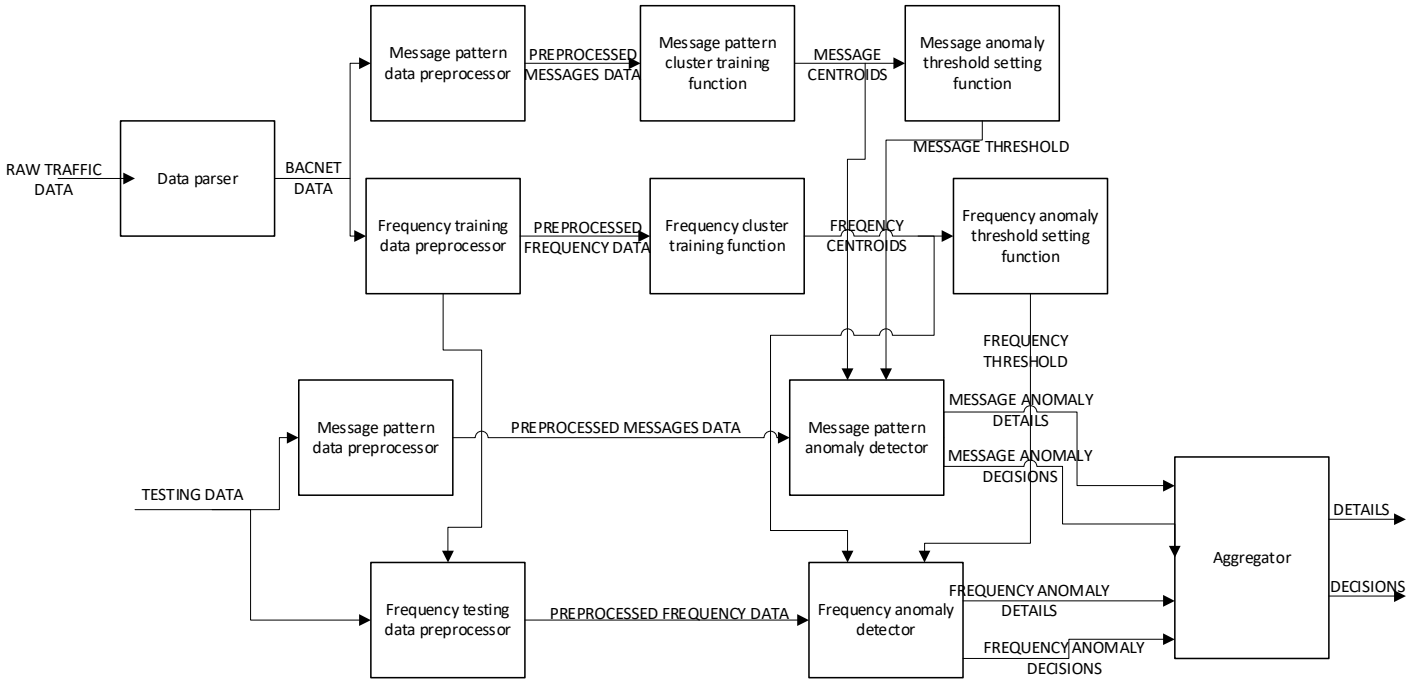


Figure 3: Expanded view of the portion of Figure 2 enclosed in dashed boundary. The data flow for the enclosed system is split into two distinct parts due to us treating the two flows as independent.

Explanation of the function blocks:

Pre-processing:

Once raw data is collected, we pre-process or clean up the data for the ML algorithm to make sense of. Pre-processing involves correcting raw data that is characterized by inconsistencies, lack of certain trends and outliers. Through parsing, the data can be made easier to process (less resource taxing) for the ML algorithm. To start parsing, we first import the dataset into a programming IDE (in our case, Jupyter Notebook) and categorize the dataset into a manageable format for the algorithm. In some cases, we can calculate statistical values such as mean, mode or median to replace missing values since this can be a good approximation of the missing dataset or delete portions of the data provided, we have enough of it in the first place.

Clustering:

The pre-processed data is passed through the chosen clustering algorithm to train and get representative cluster patterns for the normal BACnet traffic. The data is assigned into clusters based on how similar it is to a trained set of centroids, and which can represent the entire dataset . This enhances the performance of the anomaly detector later as the detector will simply compare the new inputs to the clusters rather than comparing to every single data unit in the normal training set.

Threshold Setting:

This function block finds the maximum distance of a data point in the normal data set to the centroid of the cluster it belongs to. The value of the threshold is passed to the anomaly detector.

Anomaly Detection:

Anomaly detection is comparing the input set to the training set and finding the dissimilarity score or distance between each unit of testing data and the training (or normal set). As we use clustering to represent the system, we compare the input set to the set of cluster centroid (representatives) and declare it as anomalous if the closest centroid to it is farther away than the distance score, or the dissimilarity reaches beyond the threshold.

CHAPTER IV

SYSTEM DESIGN

Reiterating the problem statement for our project design, it is that BACnet systems are vulnerable and this is more so when considering zero-day attacks. As the first step for any security measure is detection, our objectives for the system therefore were for it to be able to detect any anomalies in the BACnet traffic by understanding the 'normal' pattern of traffic. In our case, this was implemented by using a machine-learning based algorithm employing K-Means and K-Modes clustering methods, and the system was designed around the constraints we faced to achieve our objectives.

Constraints:

- Lack of data: Real building network data was difficult to get access to due to IT security concerns. An alternate solution to emulate our own BACnet system was considered and implemented. The hardware set-up though had to be abandoned due to the closure of labs following the COVID-19 outbreak. The lack of data became an inhibitor on the design due to the data hungry nature of many popular machine learning algorithms. Due to the amount of data we had on hand, we could only choose a method that does not demand a lot of data—K-modes and K-means.

- No real-time testing: The most effective way to test our system would be to implement the solution in a real BACnet and inject anomalies to get an estimate for the mean accuracy of the technique. However, due to us not having access to real BMS, nor any expertise in creating malicious attacks, we had limited success in creating a testing data set for the system. The methods we used to do this are discussed in the next chapter.

- Clustering: The choice of clustering using K-Modes or K-Means involves certain trade-offs and rigidity in pre-processing. For one, the K-Modes algorithm takes in categorical data, which indicates each of its features are Boolean, while K-Means takes in numerical data. In addition, the two ML training programs involve setting the number of clusters based on scores such as the Silhouette Index or the inertia-based Elbow method. The inability to confirm with full assurance the make-up of the centroid lists despite the methods leave only trial-and-error methods, albeit constrained within a range of values.

Process Design:

For the training part, we used K-modes and K-means clustering to represent two aspects of the QU training data set- message patterns and message frequency. Clustering is a popular way to represent the data and the K- methods are in general less data hungry than their counterparts. The need for separating the two aspects and use different models for training was because message pattern data was categorical while frequency had numerical data. As algorithms go, k-means is a well-known classifier for numerical data but fails in classifying categorical data while k-modes is the opposite. Thus, the combination of the two is better than standalones for our case. The overall process of training and testing is represented by the flowchart in Figure 4.

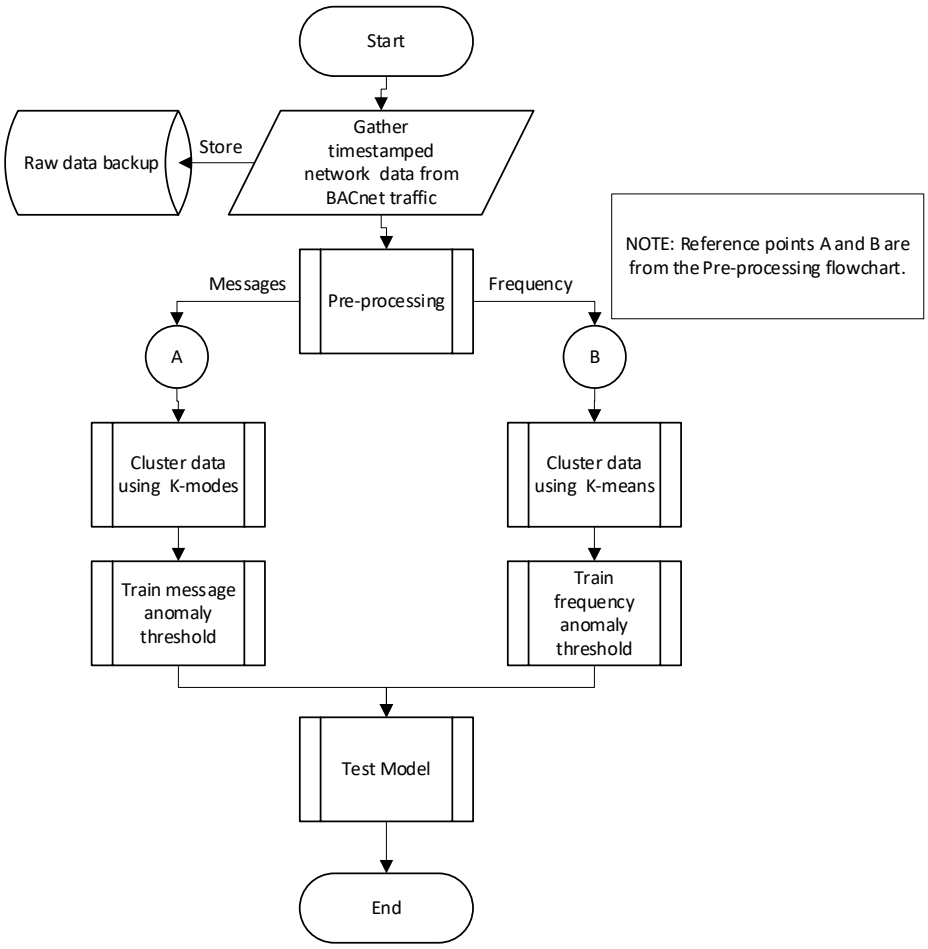


Figure 4: Overall Process Flowchart

The overall process is composed of pre-defined function blocks explained further below with elaborated discussion on the methods. The general structure is similar to the make-up of the functional diagram in chapter 3. We will first elaborate on the pre-processing techniques we implemented for the ML algorithms, how we implemented the ML for our project and how we determined the threshold for classifying anomaly messages. Testing the model, however, is a sub-unit that is of similar complexity to the full system and is explained in detail with the flowchart in the next chapter.

Pre-processing

Starting with the pre-processing, the BACnet messages are first filtered out from the captured QU data traffic packets. The initially filtered-out data is stored separately for backup in a .csv file. An example of this data is shown in Figure 5. This data is now split into two separate data sets- one with only time and message columns (frequency data set), and the other with Source IP, Destination IP, Protocol, and Message columns (Message pattern data set).

	Time_Stamp	Source	Destination	Protocol	Message_Type
0	0.000000	10.10.10.43	10.30.10.12	BACnet-APDU	Read-Property
1	0.022244	10.30.10.12	10.10.10.44	BACnet-APDU	Tell
2	0.036096	192.168.1.51	1.1.1.1	ARP	Read-Property

Figure 5: Parsed BACnet dataset from raw Data.

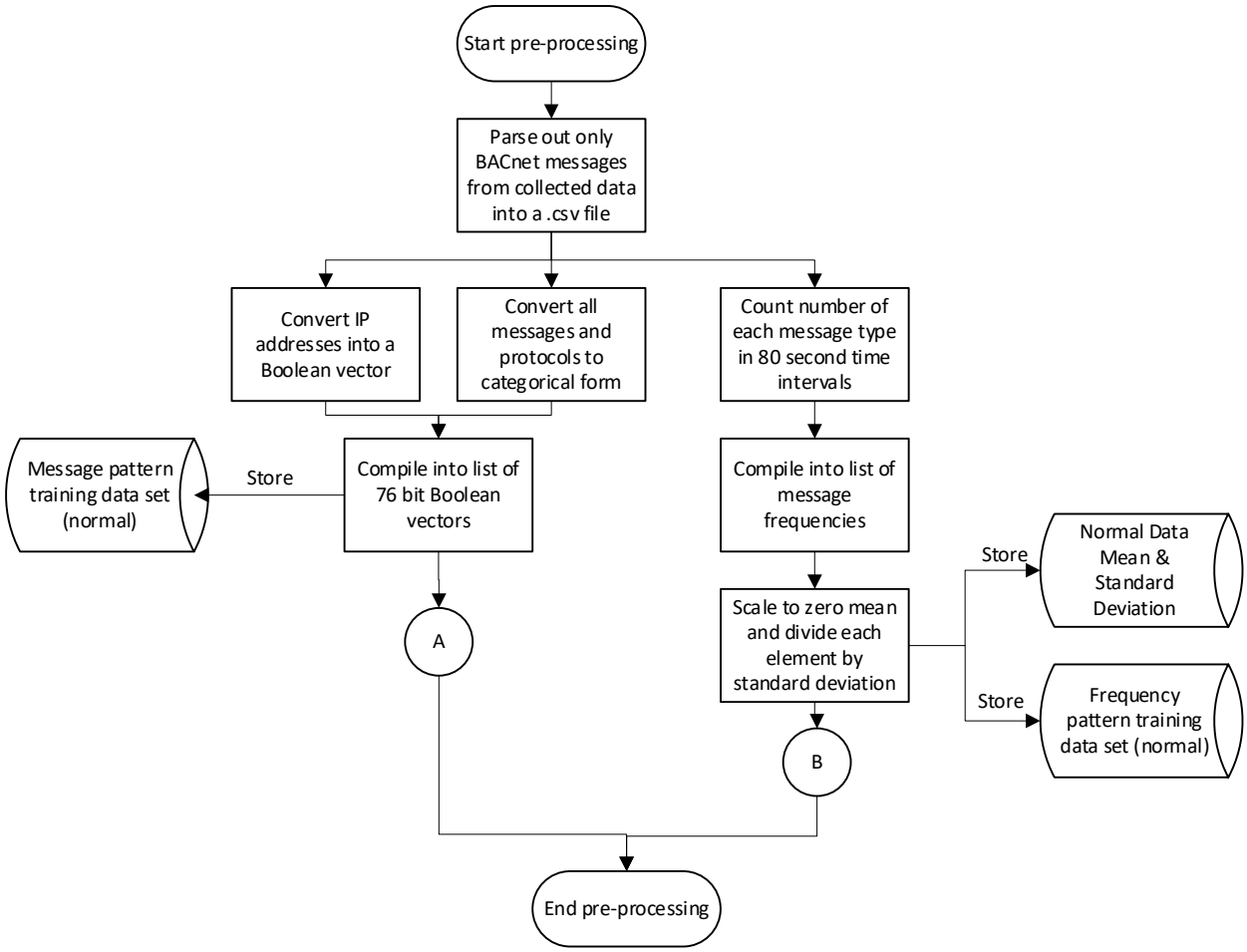


Figure 6: Pre-processing function block

The Message pattern data set is treated as shown in the flowchart in Figure 6, the unique IP addresses being separately stored in a. json format dictionary for archival. The source and destination IP addresses, the message types and protocols are converted into Boolean categories (76 of them). The frequency pattern data is separated into intervals of 10 seconds, after which the counts for each message type in the intervals are tabulated. This then becomes the frequency data set. Due to bias enabled by the different mean and standard deviation of the features, the entire data set is normalized (0 mean, one value of std) and the parameters stored for normalizing the testing dataset later.

The outputs of pre-processing A and B are passed to the overall flow as shown in the Figure 6.

To obtain output A, which corresponds to the dataset required for training K-modes, we performed the following steps:

Step 1: We converted the IP addresses into binary format of 32 bits.

Source	Destination
10.10.10.43	10.30.10.12
10.30.10.12	10.10.10.44
192.168.1.51	1.1.1.1

	bit1	bit2	bit3	bit4	bit5	bit6	bit7	bit8	bit9	bit10	...
0	0	0	0	0	1	0	1	0	0	0	...
1	0	0	0	0	1	0	1	0	0	0	...
2	1	1	0	0	0	0	0	0	1	0	...

Figure 7: Binary Dataset

This step was essential since the data was categorical in nature and we wanted our algorithm to study every feature attribute in order to train the model.

ip = ''.join([bin(int(x)+256)[3:] for x in i.split('.')]) → Converts IP address to a 32 bit binary number

Step 2: Convert the Protocols and Messages into One-hot encoded vectors

This step is important since we are converting many message types and protocols into binary representation. This will be very helpful for Machine Learning algorithms to learn the categorical features.

Protocol	Message_Type
BACnet-APDU	Read-Property
BACnet-APDU	Tell
ARP	Read-Property

Protocol_BACnet-NPDU	Messages_Ack-Message	Messages_At
0	0	0
0	0	0
0	0	0

Figure 8: Binary Message & Protocol

The overall pre-processed data looks like the figure shown below:

bit9	bit10	...	Protocol_BACnet-NPDU	Messages_Ack-Message	Messages_At	Messages_I-Am	Messages_Read-Property	Messages_Tell	Messa
0	0	...	0	0	0	0	1	0	
0	0	...	0	0	0	0	1	0	
1	0	...	0	0	0	0	0	1	
0	0	...	0	0	0	0	1	0	

Figure 9: Final Dataset for K-modes containing 76 Boolean variables

To obtain output B, which corresponds to the frequency dataset required for training K-means, we performed the following steps:

Step 1: Extracted the time stamp and corresponding messages

Time_Stamp	Message_Type
0.000000	Read-Property
0.022244	Tell
0.036096	Read-Property

Figure 10: The initial dataset for k-means

Step 2: Split the data into Training and Testing Set. This will be useful for validating the model

```
#Splitting Dataset into training and testing
train_dataset, test_dataset = train_test_split(Dataframe1, train_size=195234, test_size=48808, random_state=42)
```

```
test_dataset
```

	Time	Messages
189113	6080.007609	Unconfirmed-Transfer
27417	876.593662	Read-Property
168924	5437.786949	Read-Property
58435	1822.397862	Who-Router
52419	1631.677693	Read-Property
...	...	...
148241	4772.609094	Write-Property
124353	4057.954587	Who-Router
55728	1739.750261	Read-Property
138203	4459.285805	Unconfirmed-Transfer
182073	5865.041519	Unconfirmed-Transfer

48808 rows × 2 columns

```
train_dataset
```

	Time	Messages
109770	3587.797906	Read-Property
104305	3409.291461	Read-Property
11540	361.674274	Read-Property
53408	1664.723827	Read-Property
31033	992.000562	Who-Router
...	...	...
119879	3919.011194	Read-Property
103694	3389.528957	Read-Property
131932	4257.174674	Read-Property
146867	4730.404518	Tell
121958	3979.832025	Who-Router

195234 rows × 2 columns

Figure 11 & 12 : Testing dataset (left) & Training Dataset (right)

Step 3: Convert the dataset into 10 second time intervals

	Time	Messages
4	0.098198	Read-Property
6	0.222184	Read-Property
11	0.487171	Read-Property
12	0.505225	Read-Property
22	0.847124	Write-Property
...	...	...
244028	7870.868233	Who-Router
244029	7870.868403	Who-Router
244031	7870.868474	Who-Router
244032	7870.868497	Who-Router
244040	7871.043872	Read-Property

48808 rows × 2 columns

	Time	Messages
4	(0, 10]	Read-Property
6	(0, 10]	Read-Property
11	(0, 10]	Read-Property
12	(0, 10]	Read-Property
22	(0, 10]	Write-Property
...	...	...
244028	(7870, 7880]	Who-Router
244029	(7870, 7880]	Who-Router
244031	(7870, 7880]	Who-Router
244032	(7870, 7880]	Who-Router
244040	(7870, 7880]	Read-Property

48808 rows × 2 columns

Figure 13: Time Interval Conversion

This time interval was chosen because the spread between the frequency counts were balanced throughout and this will guarantee the algorithm will get an equal representation of data points to train.

Step 3: Make a frequency table

This is a two-way frequency that will group all the messages for each time interval

	Ack-Message	At	I-am	Read-Property	Tell	Unconfirmed-Transfer	Who-Is	Who-Router	Write-Property
(0, 10]	3	2	0	40	5	1	1	5	1
(10, 20]	1	3	3	35	2	5	2	6	0
(20, 30]	1	2	0	32	6	7	3	12	1
(30, 40]	4	7	1	49	0	5	1	9	61
(40, 50]	3	3	0	29	2	4	1	12	1

Figure 14: Two way frequency table

Step 4: Standardize the dataset

When we are dealing with an imbalanced dataset where the frequency count varies by a large amount, this can affect the machine learning process. The ML algorithm will be biased towards majority class and neglects the least occurring classes. We would want to avoid that to make sure algorithm learns every feature. Hence, standardizing the dataset is essential since all the values will be on the same scale and the ML algorithm won't be affected by very large values that can affect the learning process of the algorithm.

Below is the standardized data table where values range between [-1,1].

	Ack-Message	At	I-am	Read-Property	Tell	Unconfirmed-Transfer	Who-Is	Who-Router	Write-Property
0	0.383131	-0.420710	-0.589456	0.315202	0.999827	-1.458093	-0.685077	-0.301398	-0.133427
1	-0.926662	0.163980	2.049700	-0.264040	-0.657670	-0.084950	-0.008585	0.046845	-0.220105
2	-0.926662	-0.420710	-0.589456	-0.611586	1.552326	0.601621	0.667907	2.136300	-0.133427
3	1.038028	2.502741	0.290262	1.357838	-1.762668	-0.084950	-0.685077	1.091573	5.067254
4	0.383131	0.163980	-0.589456	-0.959131	-0.657670	-0.428236	-0.685077	2.136300	-0.133427

Figure 15: Standardized Data

Next, we will proceed towards the implementing our Machine Learning models with the datasets we have obtained. The Machine Learning models we chose were heavily inspired from undertaking the ECEN 489 course that relates to Machine Learning. We understood that our data was unsupervised since it did not contain any labels for anomaly data. In addition, we wanted to explore the patterns and behaviour of the dataset. Hence, we decided to go forward with a popular clustering technique called K-means & K-modes. The striking feature of these algorithm is that will study the normal data patterns well and will therefore be able to adapt to any novel anomalous data packets that deviate from the normal data. Below we will explain on how these algorithms work.

K-means:

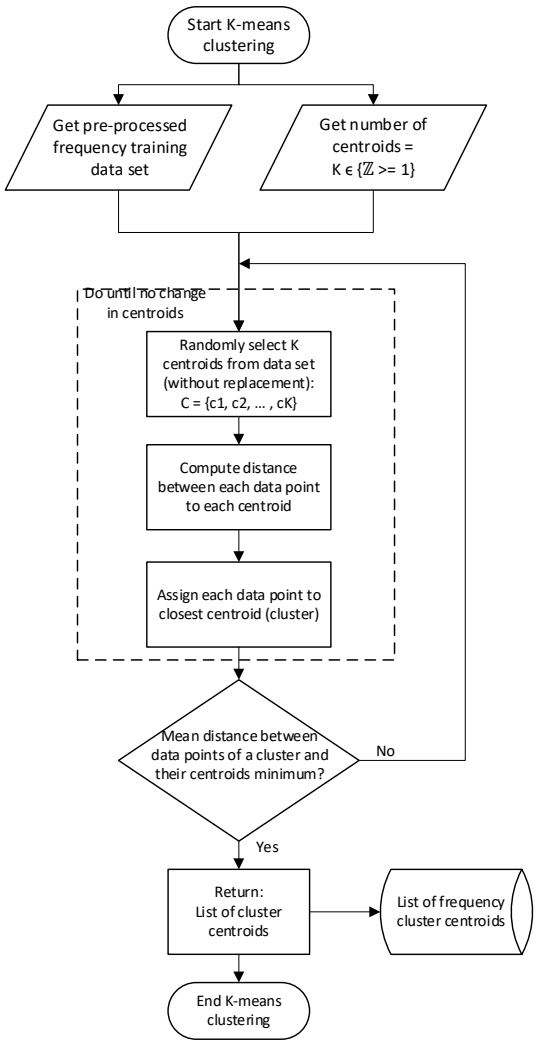


Figure 16: K-means algorithm flowchart

K-means algorithm was implemented with the goal of training our algorithm to detect DOS attacks, or frequency nature of data packets. The dataset we obtained had numerical attributes and hence K-means would be optimal since it can find out relationship between each feature. One main step that we needed to follow was in obtaining the parameter associated with finding the best number of clusters to represent the data. K-means does not give us the clusters automatically and therefore we need to feed the cluster parameter.

In order to determine the best number of clusters, we used two methods as shown below:

1. Elbow Method:

   This method involves plotting the variation in clusters as a function of the number of clusters. From this, we chose the number of clusters to improve the k-means model.

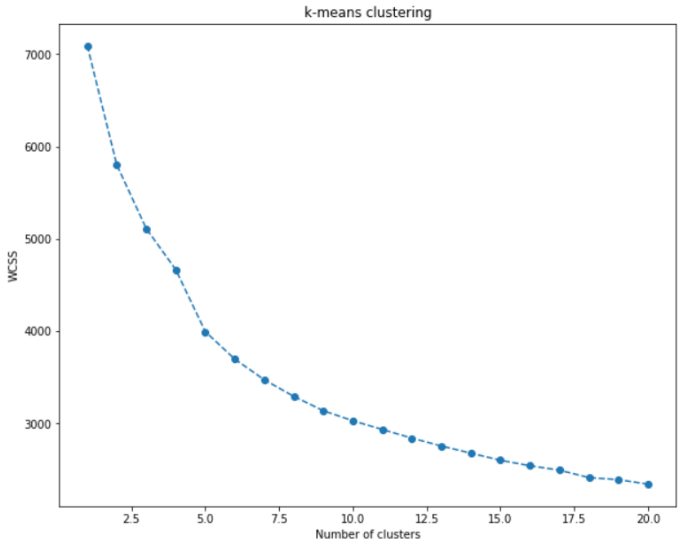


Figure 17: Elbow plot

   For the K-means we decided to go with the elbow method since it does not deal with categorical variables like Kmodes did and it has been shown to work better. As shown in figure 5, 5 clusters was chosen to be optimal since it had lower loss function and adding any more clusters than 5 would not really affect the performance of the model.

2. Silhouette Score

   This method is a measure of how similar an object is to its own cluster compared to other clusters (separation). The silhouette scores usually range from -1 to +1, where a high value tells us that objects is well matched to its own cluster and will be poorly matched to other clusters. This gives us preview of the performance of the k-means algorithm.

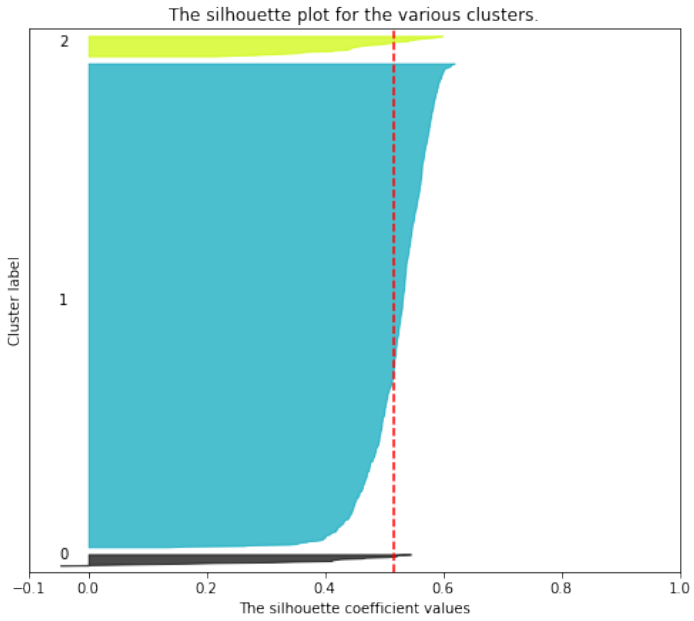


Figure 18: Silhouette Score for 3 centroids

The silhouette score was high for 3 clusters at about 0.53 compared to the clusters from elbow method. The large blue region tells us that most of the data points will belong to that cluster since they exhibit the same characteristics.

Now, the next step was to obtain the cluster centroids. Here we decided to go with 3 clusters because the silhouette score is above 0.2 and for all the other number of clusters, we found the silhouette score to be less than 0.2. The silhouette score is a measure of the dissimilarity of the clusters so there we wanted to have a high dissimilarity between centroids so that they would not represent similar data.

	Ack-Message	At	I-am	Read-Property	Tell	Unconfirmed-Transfer	Who-Is	Who-Router	Write-Property
0	-0.165400	0.800401	0.181164	0.192382	0.489187	-0.111949	0.232882	-0.165117	0.053874
1	-2.735486	-1.628621	-0.303063	-4.656611	1.974753	-0.752204	-0.982169	-1.156649	-0.172268
2	0.354351	-0.551646	-0.129990	0.201112	-0.569536	0.153831	-0.120735	0.230582	-0.032229

Figure 19: Cluster centroids used in this project

Obtaining the centroid values were only a 50% completion in designing our anomaly detector algorithm. We had to use these centroids to determine a suitable threshold value that will be classify any novel data into an anomalous or non-anomalous dataset.

The squared Euclidean distance is used to determine a distance-based anomaly score to detect outliers. The threshold value is calculated, and the highest distance score is taken. This high score indicates the distance at which the normal data point exists in the observable cluster. Anything above that distance will not belong to the cluster, and therefore will be an outlier.

The squared Euclidean distance is utilized mainly because it measures the similarities between the centroid values and the arbitrary data points by measuring the regular distances between them. Any distance that is unusually high indicates that there is a high degree of dissimilarity between the normal dataset supposed to exist and the novel data.

Below is the formula for square Euclidean distance between two points a and b in k dimensions:

$$d = \sum_{k=1}^{n} (a_k - b_k)^2$$

The flowchart explains how we find the threshold for the frequency anomaly detector with the help of k-means model.

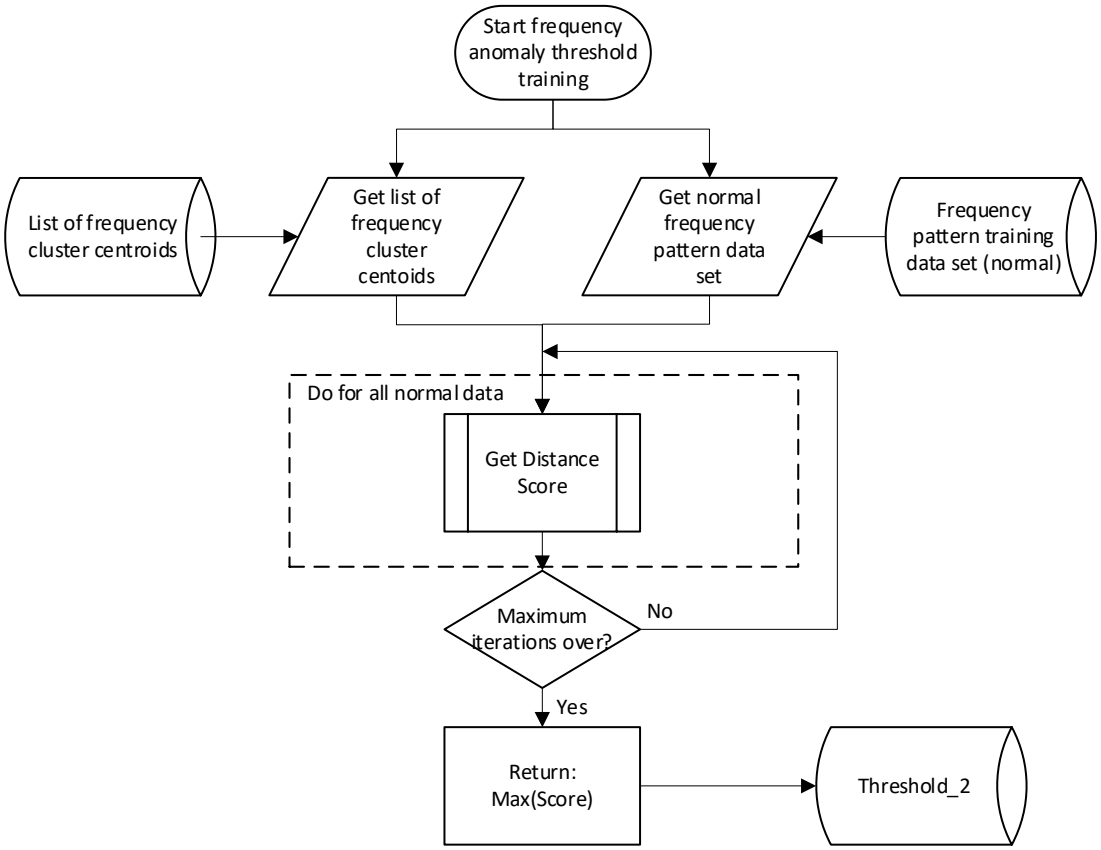


Figure 20: Function block for training the threshold for frequency anomaly detector

Below is the threshold value we found for 3 clusters. This value didn't change much from when we implemented 5 clusters. The value for 5 clusters was 25.95.

```
#Data: u = Normal dataset, v = Set of cluster centroids;
#Result: Distance score of each u from its respective nearest centroid
from scipy.spatial import distance
#class_weight
scaled_train_array = train_normalized.values
scaled_clusters3 = clusterInfo_3.values

def ThresholdCalculator(u,v,w):
    x = np.empty(shape = (len(v)-1),dtype='object')
    y = np.empty(shape = (len(u)-1),dtype='object')

    for i in range(len(u)-1):
        for j in range(len(v)-1):
            #Calculating dissimilarity or distance scores
            x[j] = distance.sqeuclidean(u[i],v[j],w)
        #minimum(distance of u[i] to each element in v):
        y[i] = np.amin(x)
    Threshold = max(y)
    return(Threshold)
```

```
Threshold =  25.21574972526369
```

Figure 21: Code to obtain the threshold value for any number of clusters

The threshold value found earlier is then checked on the testing data to see if the previous value found is useful in classifying most of the testing data as non-anomalous observations.

```
def AnomalyDetector(u,v,t,w):
    x = np.empty(shape = (len(v)-1),dtype='object')
    y = np.empty(shape = (len(u)-1),dtype='object')
    Status = y
    Threshold = float(t)
    for i in range(len(u)-1):
        for j in range(len(v)-1):
            #Calculating dissimilarity or distance scores
            x[j] = distance.sqeuclidean(u[i],v[j],w)
        #minimum(distance of u[i] to each element in v):
        y[i] = np.amin(x)

    for k in range(len(u)-1):
        if y[k] <= Threshold:
            Status[k] = 'Normal'
        else:
            Status[k] = 'Anomaly'

    return(Status)

#Calculate Threshold:
Threshold = ThresholdCalculator(scaled_train_array,scaled_clusters3, class_weight_norm)
print("Threshold = ", Threshold)

#Check to see all normal data is decided 'Normal'
Status = AnomalyDetector(scaled_train_array,scaled_clusters3,Threshold, class_weight_norm) #-Passed!
print("\nChecking Anomaly Detector:")
checker = 0
for i in Status:
    if i=='Anomaly':
        checker = 1
if checker == 0:
    print("   Anomaly Detector passed normal data check!")
else:
    print("   Check failed! Debug the function.")
```

```
Checking Anomaly Detector:
   Anomaly Detector passed normal data check!
```

Figure 22: Code to verify the threshold value obtained

K-modes:

K-modes algorithm is a slight variation of the K-means algorithm since it works very well for categorical datasets. It works based on finding mode or mutually occurring data patterns and counts the frequency data patterns occurring together. This is different from k-means since it relies completely on calculating the Euclidean distance and needs numerical values to find the distance. Below is a simplified version of k-modes and the resulting flowchart.

K-modes algorithm works in the following way:

i. Pick an observation at random and use that as cluster
ii. Compare each data point in the cluster centroid to each observation data point, and any elements that are not equal we add a counter of +1 and if they are equal, we keep it as '0'
iii. Assign each data point to each closest centroid based on the most least score made in step ii
iv. Update the centroids based on the most common values found from the observation data points.
v. Repeat steps ii-iv until the centroid values converge or doesn't update.

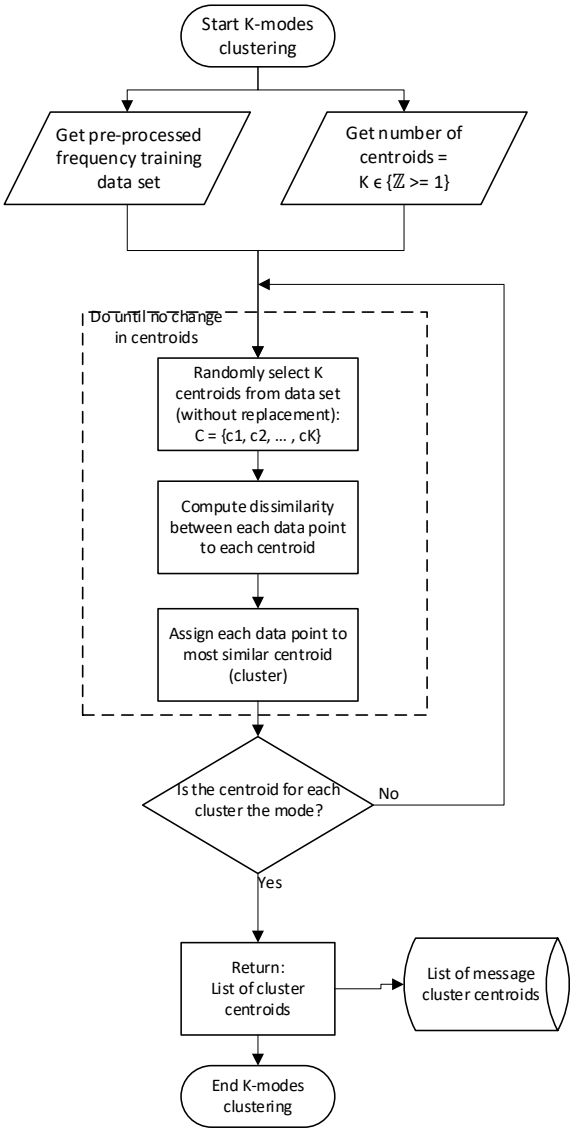


Figure 23: Flowchart for K-modes clustering algorithm

Now, we needed to determine the number of clusters that could represent the data that was collected. We first tried the elbow plot with the data for Kmodes. We did not get good results as shown below.

```
In [8]: def ElbowMethod(n, init_method,init_n,data):
            # Finding the best number of clusters by obtaining cost (elbow method)
            cost = []
            for num_clusters in list(range(1,n+1)):
                kmode = KModes(n_clusters=num_clusters, init = init_method , n_init = init_n, verbose=1)
                kmode.fit_predict(data)
                cost.append(kmode.cost_)

            y = np.array([i for i in range(1,n+1,1)])
            return y,cost
```

Figure 24: Elbow method code

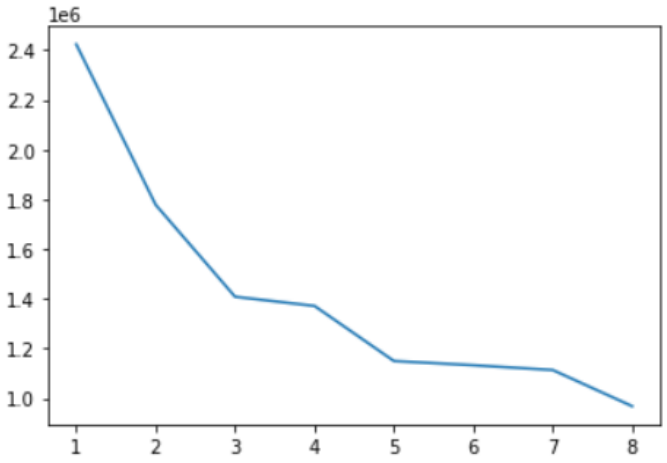


Figure 25: Elbow plot for K-modes

As you can see the elbow method shows the elbow occurring at for the number of clusters to be 3. This, we know is not fully representative of the dataset. Therefore, we move onto the other method for the silhouette plot (which was done in another python IDE called Visual Studio)

```
32  ### Code for the silhouette plot
33  for n_clusters in range(2,15):
34      # Create a subplot with 1 row and 2 columns
35      fig, (ax1, ax2) = plt.subplots(1, 2)
36      fig.set_size_inches(18, 7)
37
38      # The 1st subplot is the silhouette plot
39      # The silhouette coefficient can range from -1, 1 but in this example all
40      # lie within [-0.1, 1]
41      ax1.set_xlim([-0.1, 1])
42      # The (n_clusters+1)*10 is for inserting blank space between silhouette
43      # plots of individual clusters, to demarcate them clearly.
44      ax1.set_ylim([0, len(x) + (n_clusters + 1) * 10])
45
46      # Initialize the clusterer with n_clusters value and a random generator
47      # seed of 10 for reproducibility.
48      clusterer = KModes(n_clusters=n_clusters,init = 'Cao',random_state=10,verbose = 1)
49      cluster_labels = clusterer.fit_predict(x)
50
51      # The silhouette_score gives the average value for all the samples.
52      # This gives a perspective into the density and separation of the formed
53      # clusters
54      silhouette_avg = silhouette_score(x, cluster_labels)
55      print("For n_clusters =", n_clusters,
56            "The average silhouette_score is :", silhouette_avg)
```

Figure 26: Silhouette Score code

```
    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(x, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        # Aggregate the silhouette scores for samples belonging to
        # cluster i, and sort them
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        # Label the silhouette plots with their cluster numbers at the middle
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # Compute the new y_lower for next plot
        y_lower = y_upper + 10  # 10 for the 0 samples

    ax1.set_title("The silhouette plot for the various clusters.")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # The vertical line for average silhouette score of all the values
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 2nd Plot showing the actual clusters formed
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(x[:, 0], x[:, 1], marker='.', s=30, lw=0, alpha=0.7,
                c=colors, edgecolor='k')

    # Labeling the clusters
    centers = clusterer.cluster_centroids_
    # Draw white circles at cluster centers
    ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                c="white", alpha=1, s=200, edgecolor='k')

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                    s=50, edgecolor='k')

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")
    plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "
                  "with n_clusters = %d" % n_clusters),
                 fontsize=14, fontweight='bold')

plt.show()
```

Figure 27: Visual Studio code for the silhouette scores

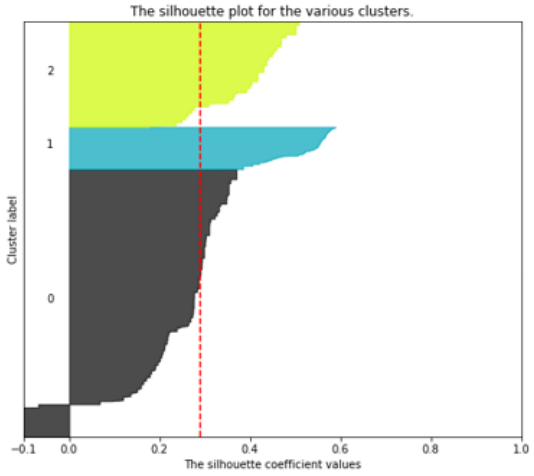


Figure 28: 3 cluster

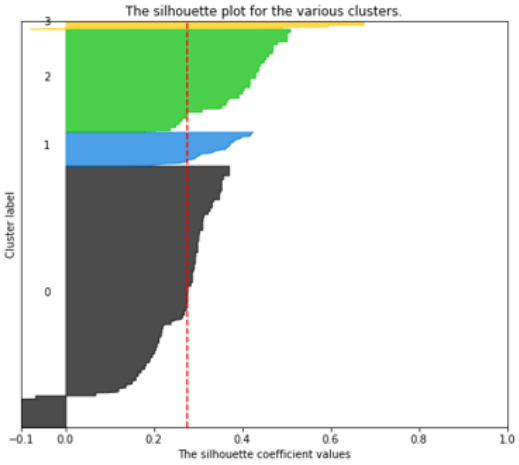


Figure 29: 4 clusters

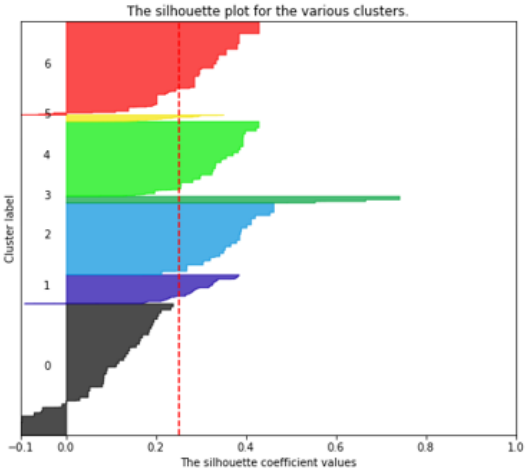


Figure 30: 7 clusters

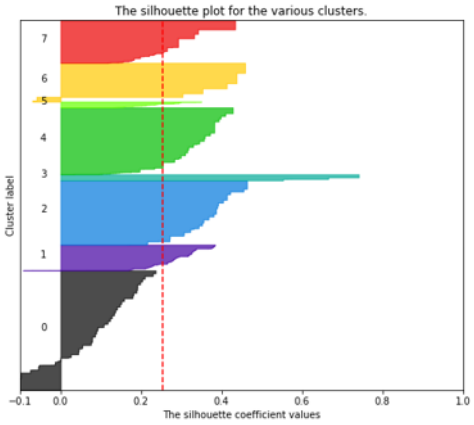


Figure 31: 8 clusters

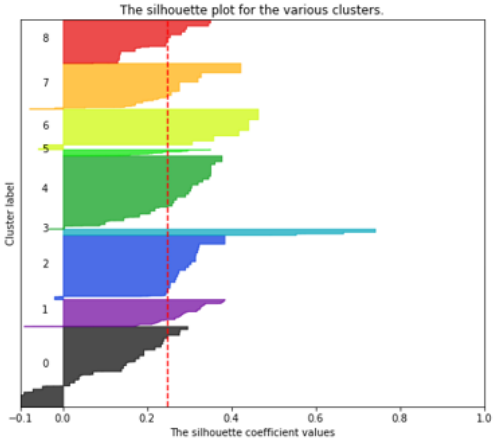


Figure 32: 9 clusters

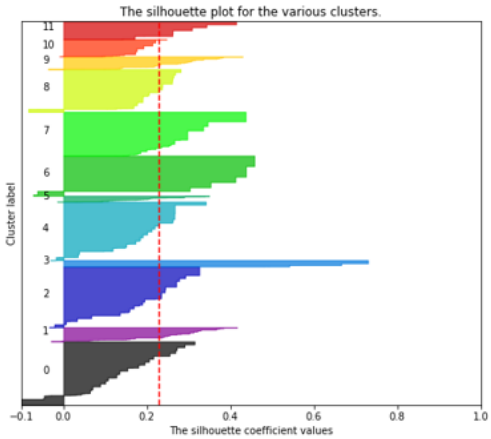


Figure 33: 12 clusters

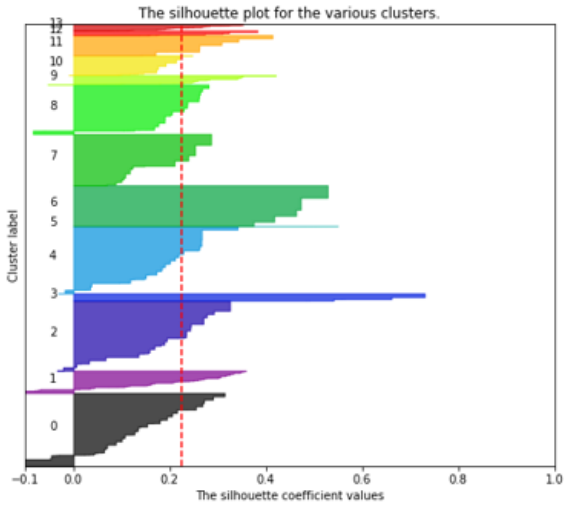


Figure 34: 14 clusters

As we can see from the figures above the widths of the cluster are imbalanced when the number of clusters are very small. As we increase the number of clusters, we then have a problem of how thin the clusters are. We chose 14 clusters because we believed that the cluster distribution looks good and that all of them cross the silhouette score (vertical dashed red line).

We had to determine the number of centroids for chosen number of clusters. For the kmodes detection algorithm we now know that we have to choose 14 clusters therefore the below code is a function defined to get the cluster centroids and we can see where each row/observation can be fitted for each cluster. The most important piece of information is the information of the cluster centroids. Each centroid would have a value for each feature. As you can see below from the dataset that each centroid from 0-13 (14 cluster centroids) have different values of features assigned to it.

```
In [16]: def makeClusters(Data, num_clusters,max_iter_no, init_method, init_no, random_state_no):

             km_cao = KModes(n_clusters= num_clusters, max_iter = max_iter_no, init = init_method, n_init = init_no,
                             verbose=1, random_state=random_state_no)
             data = Data.values # or Finalized Data not sure?
             fitClusters_cao = km_cao.fit_predict(data)

             clusterCentroidsDf = pd.DataFrame(km_cao.cluster_centroids_)
             clusForInfo = clusterCentroidsDf.copy()
             clusterInfo = clusForInfo
             clusterInfo.columns = Data.columns #clusterInfo shows the cluster information

             # Concatenating the data for visualization
             clustersDf = pd.DataFrame(fitClusters_cao)
             clustersDf.columns = ['cluster_predicted']
             combinedDf = pd.concat([Data, clustersDf], axis = 1).reset_index()

             return clusterInfo, clusterCentroidsDf, combinedDf, fitClusters_cao
```

	bit1	bit2	bit3	bit4	bit5	bit6	bit7	bit8	bit9	bit10	bit11	bit12	bit13	bit14	bit15	bit16	bit17	bit18	bit19	bit20	bit21	bit22	bit23	bit24	bit25	bit26
0	0	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	1	0	0	0
1	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0
3	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	1	0	0	0

Figure 35: Cluster centroids for kmodes

From here we can also see which clusters these different features would most likely belong to. This is more of just a visualizing aspect that we wanted to see for kmodes.

```
plt.subplots(figsize = (10,12)) #(x,y)
sns.countplot(x=combinedDf['Messages_Tell'],order=combinedDf['Messages_Tell'].value_counts().index,hue=combinedDf['clus
plt.show()
```

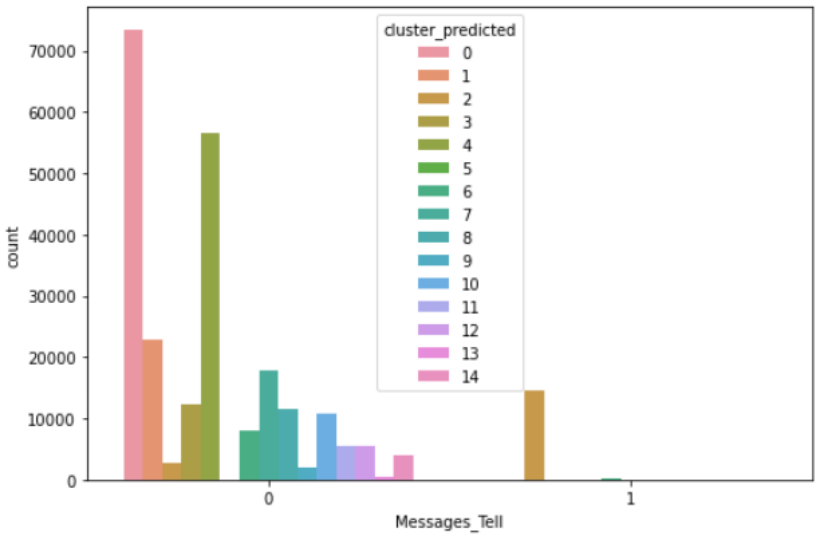


Figure 36: Cluster prediction for the BACnet Message Tell

Once the centroids are found we need to calculate an anomaly score. We first tried out different distance measurements that could be used for finding a threshold value. We tried Jaccard, Hamming, Rogers-Tanimoto and so on as you will see in the figure below. These scores were calculated by running a for loop of cluster centroids with each of the observations or rows in the datasets of either the collected dataset.

```
In [4]: #Data: u = data array to be tested, v = array of clusters; Result: Anomaly score for each element in u
def AnomalyScore(u,v):
    x = np.empty(shape = (7,len(v)-1),dtype='object')
    md = np.empty(shape = (7,len(u)-1),dtype='object')
    #Threshold for each measure:?
    for i in range(len(u)-1):
        for j in range(len(v)-1):
            #Calculating dissimilarity or distance scores
            x[0,j] = distance.jaccard(u[i],v[j])
            #x[1,j] = distance.kulsinski(u[i],v[j])
            x[1,j] = distance.dice(u[i],v[j])
            x[2,j] = distance.hamming(u[i],v[j])
            x[3,j] = distance.rogerstanimoto(u[i],v[j])
            #x[5,j] = distance.russellrao(u[i],v[j])
            x[4,j] = distance.sokalmichener(u[i],v[j])
            x[5,j] = distance.sokalsneath(u[i],v[j])
            x[6,j] = distance.yule(u[i],v[j])
        #minimum(distance of u[i] to each element in v)
        md[0,i] = np.amin(x[0,:])
        md[1,i] = np.amin(x[1,:])
        md[2,i] = np.amin(x[2,:])
        md[3,i] = np.amin(x[3,:])
        md[4,i] = np.amin(x[4,:])
        md[5,i] = np.amin(x[5,:])
        md[6,i] = np.amin(x[6,:])
        #md[7,i] = np.amin(x[7,:])
        #md[8,i] = np.amin(x[8,:])
    md = np.transpose(md)
    y = pd.DataFrame(md, columns = ["Jaccard","Dice","Hamming","Rogers-Tanimoto","Sokal-Michener","Sokal-Sneath","Yule"]
    #Anomaly condition
    #compare each element in Score_type column to the set threshold for that score_type. If above threshold, replace wi
    return(y)
```

Figure 37: Distance measure code to find threshold values

The flowchart explains how we find the threshold for the message/IP/Protocol anomaly detector with the help of k-modes model.

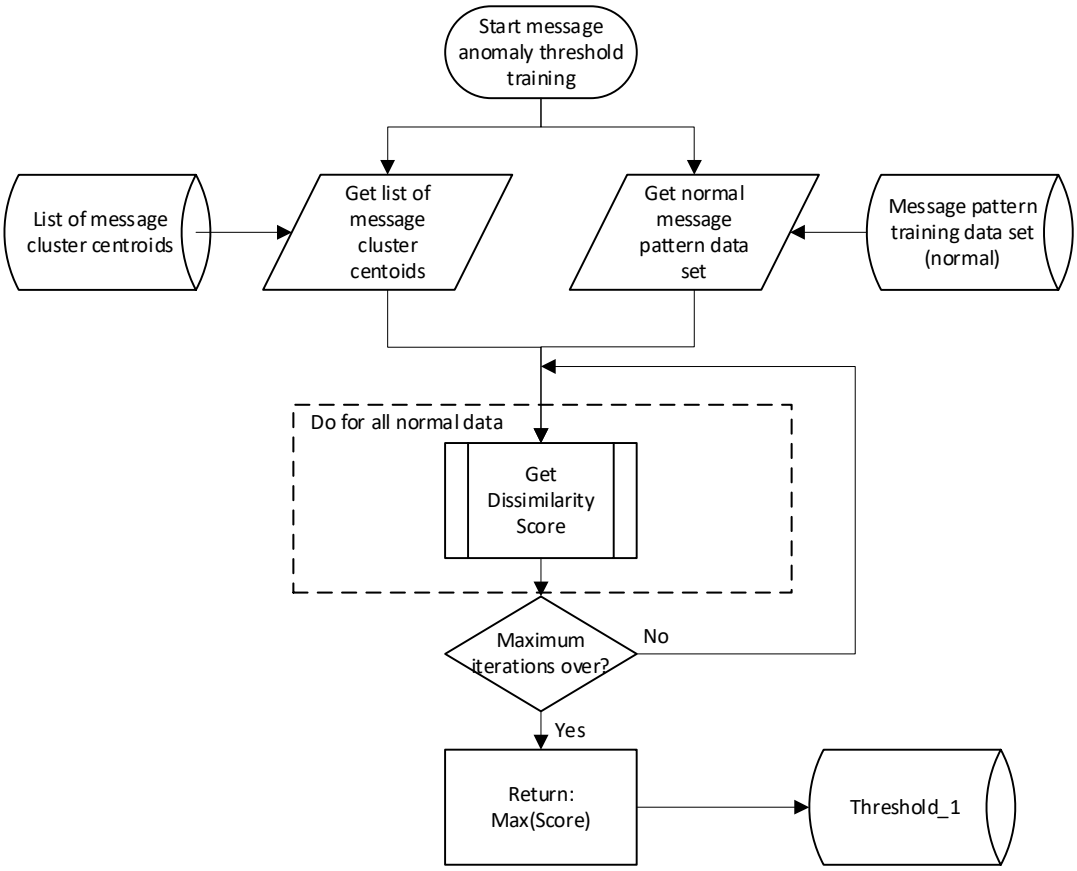


Figure 38: Training the threshold for message pattern anomaly detector

	Jaccard	Dice	Hamming	Rogers-Tanimoto	Sokal-Michener	Sokal-Sneath	Yule
0	0.166667	0.0909091	0.0526316	0.1	0.1	0.285714	0.00759013
1	0.125	0.0666667	0.0394737	0.0759494	0.0759494	0.222222	0
2	0.0588235	0.030303	0.0131579	0.025974	0.025974	0.111111	0
3	0.0454545	0.0232558	0.0131579	0.025974	0.025974	0.0869565	0
4	0.12	0.0638298	0.0394737	0.0759494	0.0759494	0.214286	0.00355872
...	...	...	...	...	...	...	...
248063	0.32	0.190476	0.105263	0.190476	0.190476	0.484848	0.0340136
248064	0.32	0.190476	0.105263	0.190476	0.190476	0.484848	0.0340136
248065	0.384615	0.238095	0.131579	0.232558	0.232558	0.555556	0.0534351
248066	0.25	0.142857	0.0789474	0.146341	0.146341	0.4	0.0169492
248067	0.32	0.190476	0.105263	0.190476	0.190476	0.484848	0.0340136

248068 rows × 7 columns

Figure 39: Different Scoring Methods

As shown above, the threshold values were calculated, and we chose the maximum of the Jaccard distance score which was 0.45. Therefore, anything above 0.45 would be considered as an anomaly.

Although the final systems design is robust and straightforward, there were several engineering decisions that needed to be taken, and much trial and error done to reach this state. Initially, this was in the choice of machine learning model— deep learning tools could find deeper and more complex relationships in the data, but this came at the cost of very high training data requirements. But the real-world difficulties we faced in the collection of building data and the inability to access the labs on account of COVID-19 made it hard, if not impossible, for us to choose such techniques. Hence, we went for the less data-hungry yet proven K-Means (Modes) clustering algorithm which could represent the basic clusters of our existing data set. The comparison between the conventional ML and Deep Learning was from the ECEN 489 Machine Learning course, which allowed us the capability to understand and implement the ML algorithms better.
In the visualization of data, which was a very important step towards building the system, we utilized methods from ECEN 303, Probability, to understand the optimization function of the clustering model and how the numbers of centroids K affects the distribution of the data. We also applied data smoothing functions, i.e., normalization to ensure equal bias for all the observations.

In the design of the system, we used many of the techniques taught in the ECEN 403 and 404 Senior Design classes as well, identifying the non-technical factors that affect our design. The market analysis we conducted, the results of which are included in Chapter 2, provided us with an idea of what features would be welcome for users and the general reception of the public and subject experts to data collection. The ethical implications of the data collection inherent to our system was not lost to us. Currently, the data may not be useful to a hacker, but we do realize that there is a need for better security of the collected data. Moving beyond the scope of the design as it is now, and to what it could be if improved and marketed, there may be positive impact on the environment- this is mainly assuming that more secure building systems would lead to greater usage of such systems which can drastically cut down on the net energy usage of the buildings they are applied to.

CHAPTER V

TESTING & FUNCTIONAL PROTOTYPING

In this section, we will show the testing results with K-means & K-modes algorithm and how we designed the GUI for our project. In order to test the robustness of our model, we needed to find an accuracy measure to check if our model works. However, we did not have any anomalous datasets due to fact that we lacked enough datasets and didn't have access to real-time BAS systems. Therefore, we decided to create our own anomalous datasets that we will discuss below. After we generate this dataset, we can run it against our model to check if our algorithm training came to any fruition. We will split this testing for the two algorithms since they had different techniques applied to the dataset.

K-modes Testing:

Firstly, we create an anomalous dataset before we apply our frequency anomaly detector to find accuracy. This step is important since we need to have dataset ready and pre-processed to run into our model. We take all the possible IP addresses that were not used in our collected data and then choose a similar number of ones and zeros for messages and the protocols to generate this anomalous dataset.

```
ii = []
jj = []
kk = []
qq = []
for blah in unique_ip.keys():
    vals = blah.split('.')
    ii.append(vals[0])
    jj.append(vals[1])
    kk.append(vals[2])
    qq.append(vals[3])
```

```
# creating ip addresses that were not used in the collected dataset
ip = []

for i in range(10,199,2):
    if str(i) in ii:
        continue
    for j in range(10,100,5):
        if str(j) in jj:
            continue
        for k in range(1,20):
            if str(k) in kk:
                continue
            for q in range(1,200,5):
                if str(q) in qq:
                    continue
                ip_str = "{I}.{J}.{K}.{Q}".format(I=i,J=j,K=k,Q=q)
                ip.append(ip_str)
```

Figure 40: Code for generating random IP values.

```
In [135]: for x in file_Protocol.columns:
              x = np.random.choice([0, 1], size=(531216,), p=[9./10, 1./10])
```

```
In [172]: # Creating random numpy arrrays with probability of 1s and 0s

          Protocol_ARP  = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Protocol_BACnet_APDU = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Protocol_BACnet_NPDU = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_Ack_Message = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_At = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_I_Am = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_Read_Property = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_Tell = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_Unconfirmed_Transfer = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_Who_Is = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_Who_Router = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
          Messages_Write_Property = pd.DataFrame(np.random.choice([0, 1], size=(len(ip),), p=[9./10, 1./10]).tolist())
```

Figure 41: Randomly shuffling the one hot encoded vector for message & protocol.

```
In [189]: Anomalous_Data = pd.concat([s,d,Protocol_ARP,
           Protocol_BACnet_APDU,
           Protocol_BACnet_NPDU,
           Messages_Ack_Message,
           Messages_At,
           Messages_I_Am,
           Messages_Read_Property,
           Messages_Tell,
           Messages_Unconfirmed_Transfer,
           Messages_Who_Is,
           Messages_Who_Router,
           Messages_Write_Property],axis=1)
```

Figure 42: Combining all the individual columns to make the anomalous dataset.

```
In [192]: Anomalous_Data
```

Out[192]:

	Source	Destination	Protocol_ARP	Protocol_BACnet-APDU	Protocol_BACnet-NPDU	Messages_Ack-Message	Messages_At	Messages_I-Am	Messages_Read-Property	Messages_
0	166.60.12.126	160.55.17.141	0	0	0	0	0	0	1	
1	180.35.5.76	26.15.13.171	0	0	0	0	0	0	0	

Figure 42: Final anomalous dataset snippet

From here we can evaluate the different distance metrices. We knew that we would be choosing Jaccard because it has been shown to work well with categorical data that was used in Kmodes. The flowchart below is how the scores are calculated for the all the different distance metrics. This flowchart below explains how the distance score is calculated for all the observations that exist in the actual dataset or the anomalous dataset. This is specifically for the k-modes algorithm to find anomalies in the source and destination IP addresses.

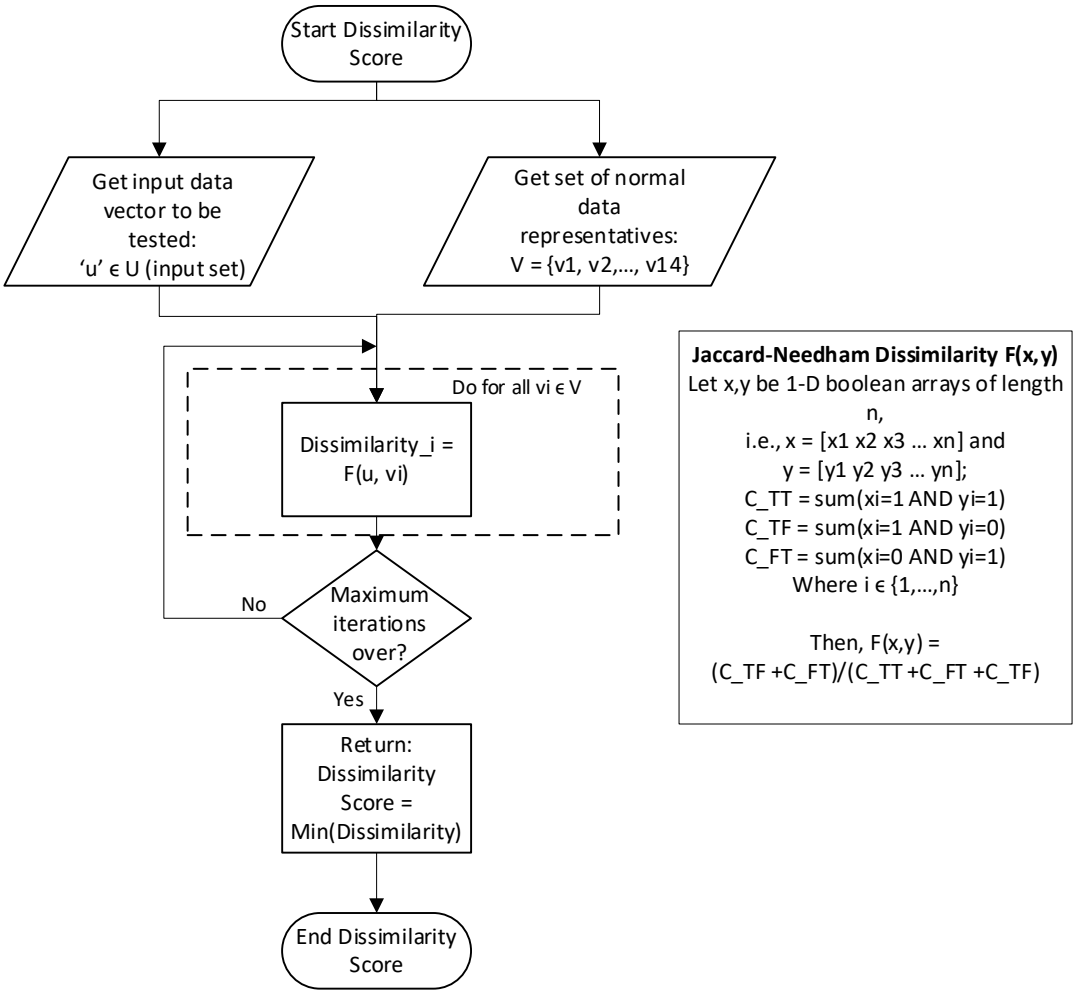


Figure 43: Dissimilarity Score sub-function block for K-modes

```
In [9]: AnomalyScore(Test_array1,Cluster_array)
```

Out[9]:

	Jaccard	Dice	Hamming	Rogers-Tanimoto	Sokal-Michener	Sokal-Sneath	Yule
0	0.08	0.0416667	0.0263158	0.0512821	0.0512821	0.148148	0
1	0.105263	0.0555556	0.0263158	0.0512821	0.0512821	0.190476	0.00206186
2	0.0869565	0.0454545	0.0263158	0.0512821	0.0512821	0.16	0.00179533
3	0	0	0	0	0	0	0
4	0.0952381	0.05	0.0263158	0.0512821	0.0512821	0.173913	0.00191205
5	0.0740741	0.0384615	0.0263158	0.0512821	0.0512821	0.137931	0.00163132
6	0.0689655	0.0357143	0.0263158	0.0512821	0.0512821	0.129032	0.0015748
7	0.0689655	0.0357143	0.0263158	0.0512821	0.0512821	0.129032	0.0015748
8	0.0769231	0.04	0.0263158	0.0512821	0.0512821	0.142857	0.00166528
9	0.0952381	0.05	0.0263158	0.0512821	0.0512821	0.173913	0.00191205
10	0.0869565	0.0454545	0.0263158	0.0512821	0.0512821	0.16	0.00179533
11	0.0869565	0.0454545	0.0263158	0.0512821	0.0512821	0.16	0.00179533
12	0.0689655	0.0357143	0.0263158	0.0512821	0.0512821	0.129032	0.0015748

Figure 44: Dissimilarity score with the Generated anomalous data

```
In [289]: test_acc = distance_metrics.iloc[:,0] #obtaining the jaccard score column

In [299]: # Counting the number of rows/observations higher than the threshold
          count = 0
          for t in test_acc:
              if t >= 0.5:
                  count = count + 1
              else:
                  continue

In [300]: count # The number of anomalous entries

Out[300]: 39880

In [301]: count/len(test_acc) # This is number of anomalous entries/total no. of observations

Out[301]: 0.9970249256231406
```

Figure 45: Code to determine the accuracy of the anomalous data

From here we can see that we obtained a 99.7% accuracy which is 'too good to be true'. However, the threshold value was decreased to 0.45 and a 98% accuracy was obtained. Either way the model needs to be fed more data since this model was trained with only 2 hours of BACnet data.

K-means Testing:

The testing accuracy is calculated to see if most of the normal data can be categorized as normal using the distance metric of the squared Euclidean distance. However, this is a bit of a problem because obtaining a 100% accuracy or no false positives supports the fact that this model needs to be fed more data. Testing the anomaly detection can be shown in the flowchart below. This flowchart explains how the anomaly detection would work in detecting any frequency anomaly such that a frequency of message from one IP address to another does not fit the norm. Here the distance measure of the squared Euclidean distance is used.

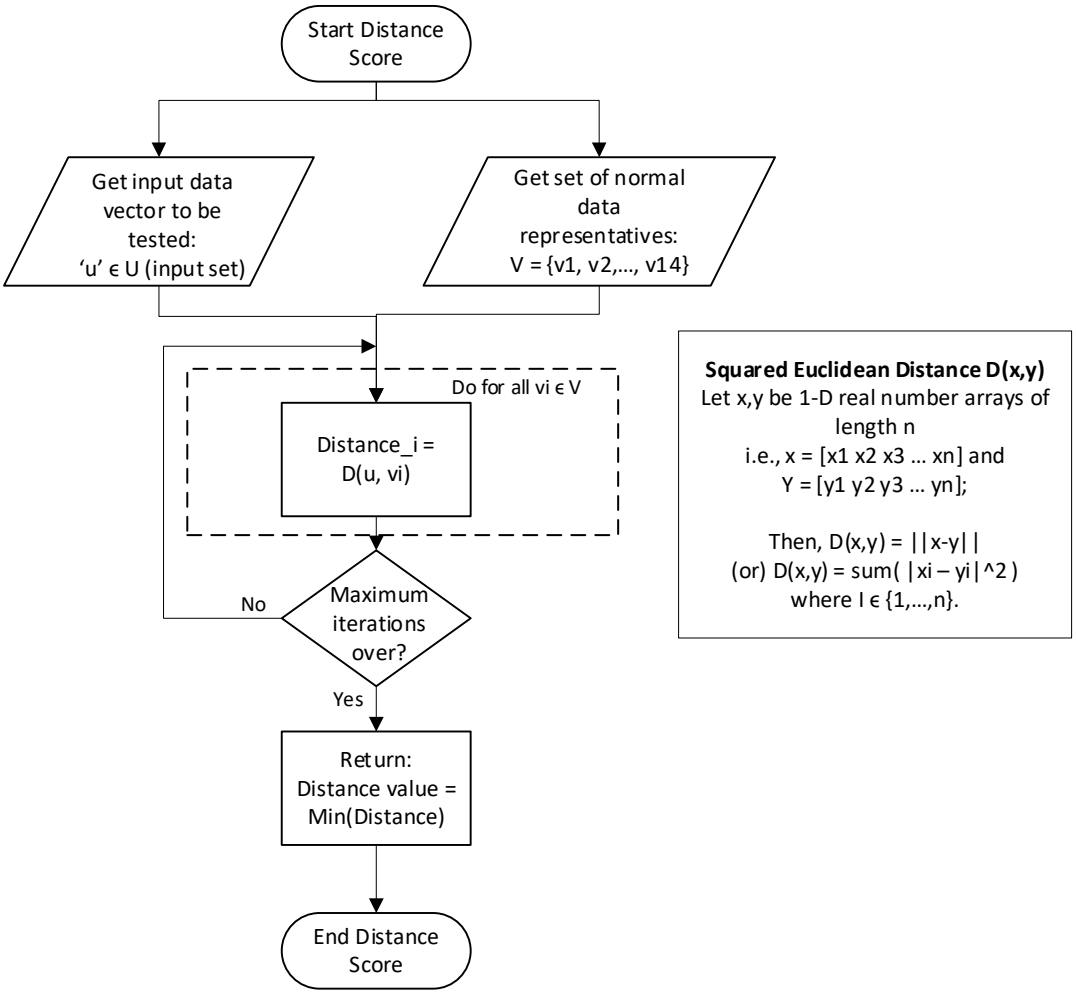


Figure 46: Distance metric sub-function block for K-Means

Here, we tested the false-positiveness of the model – the ability of our algorithm to classify normal messages as 'normal'

```
Test_array = test_normalized.values

Test_status = AnomalyDetector(Test_array,scaled_clusters3,25.2, class_weight_norm)
print(Test_status)
```

```
['Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal'
 'Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal'
 'Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal' 'Normal'
```

```
count = 0
for i in Test_status:
    if i == 'Anomaly':
        count = count + 1
print("false positive score = ",count/len(Test_status))
```

```
false positive score =  0.0
```

Figure 47: Code to calculate the testing accuracy

To test the accuracy of the model, we generated the anomalous dataset. The frequency anomalous dataset is created using the maximum value of each feature occurring in the time interval. In this scenario we assumed to multiply each of the features by 10-17 times of its value in order to generate the dataset.

```
In [45]: blah = blah.values

In [71]: values = [int(x) for x in blah[7]]
         val_dict = {}
         # val_dict.keys = freq_data.columns
         for index,val in enumerate(values):
             anom_val = []
             for mul in range(7):
                 anom_val.append(10*val+mul)
         #         val_dict = dict(zip(index,anom_val))
                 val_dict[index] = anom_val

In [72]: val_dict

Out[72]: {0: [250, 251, 252, 253, 254, 255, 256],
          1: [370, 371, 372, 373, 374, 375, 376],
          2: [160, 161, 162, 163, 164, 165, 166],
          3: [3490, 3491, 3492, 3493, 3494, 3495, 3496],
          4: [610, 611, 612, 613, 614, 615, 616],
          5: [700, 701, 702, 703, 704, 705, 706],
          6: [260, 261, 262, 263, 264, 265, 266],
          7: [780, 781, 782, 783, 784, 785, 786],
          8: [8770, 8771, 8772, 8773, 8774, 8775, 8776]}

In [104]: for row_val in range(652):
              for col_val in range(9):
                  anom_val = random.choice(val_dict[col_val])
                  if row_val+col_val == 652:
                      break
                  freq_blah[row_val+col_val,col_val] = anom_val

In [111]: columns = list(freq_data.columns[1:])

In [112]: columns

Out[112]: ['Ack-Message',
           'At',
           'I-am',
           'Read-Property',
           'Tell',
           'Unconfirmed-Transfer',
           'Who-Is',
           'Who-Router',
           'Write-Property']

In [115]: freq_blah.columns = columns
```

Out[116]:

	Ack-Message	At	I-am	Read-Property	Tell	Unconfirmed-Transfer	Who-Is	Who-Router	Write-Property
0	254	23	14	333	30	42	20	47	6
1	254	376	0	319	28	42	13	78	8
2	251	370	164	327	27	49	19	41	315
3	250	376	166	3490	30	49	20	71	6
4	251	371	166	3496	611	35	13	41	6
5	251	370	166	3496	612	702	19	54	5
6	251	375	160	3494	610	702	265	65	6
7	256	374	161	3491	613	702	263	780	6
8	253	373	161	3494	614	706	260	783	8772
9	250	371	162	3493	613	702	261	783	8771
10	254	372	166	3496	614	705	265	786	8770

Figure 48: Code for generating the frequency dataset

This data, as shown later will be useful for determining the anomalies. This is the same for the frequency detection as well. K means also uses cluster centroids to determine the anomaly score.

The next distance metric we used was for the frequency detection. We found that using the squared Euclidean distance can be seen as simple, but it is also very useful to determine the distance between an observation that belongs to the dataset and an observation that is an anomaly.

```
anomaly = anomaly_data.values
scaler = StandardScaler()
scaler.fit(message_class10.values)
Anom_data = scaler.transform(anomaly)

Test_status = AnomalyDetector(Anom_data,scaled_clusters3,25.2, class_weight_norm)
print(Test_status)

count = 0
for i in Test_status:
    if i == 'Anomaly':
        count = count + 1
print("accuracy score = ",count/len(Test_status)*100)
```

```
['Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly'
 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly'
 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly' 'Anomaly']
accuracy score =  100.0
```

Figure 49: Frequency detection for anomalous data

GUI Design (See Appendix for code):

From here we can detect the threshold and the accuracy to be used for the GUI. For the Graphical User Interface (GUI) we had to combine both frequency detection and the anomalous detection together in order to detect the anomaly. The GUI consisted of converting the user input into a format such that the model can calculate an anomaly score and determine if the user input is an anomaly or not. Below you will find the functions defined and the whole code is provided later in the appendix. The flowchart below shows how the GUI will work. This is the final model that combines the previous algorithms to determine from a user point of view whether his/her input into the GUI is an anomaly or not. This GUI replicated an intruder entering or tampering with a BMS system. Therefore, we wanted to ensure that sending messages to IP addresses that normally do not talk to each other and a brute force attack where sending a high number of messages in a short amount of time between two IP addresses can be anomalies. This shows how further intruder attacks could also be integrated with our existing model.

Figure 50: GUI design

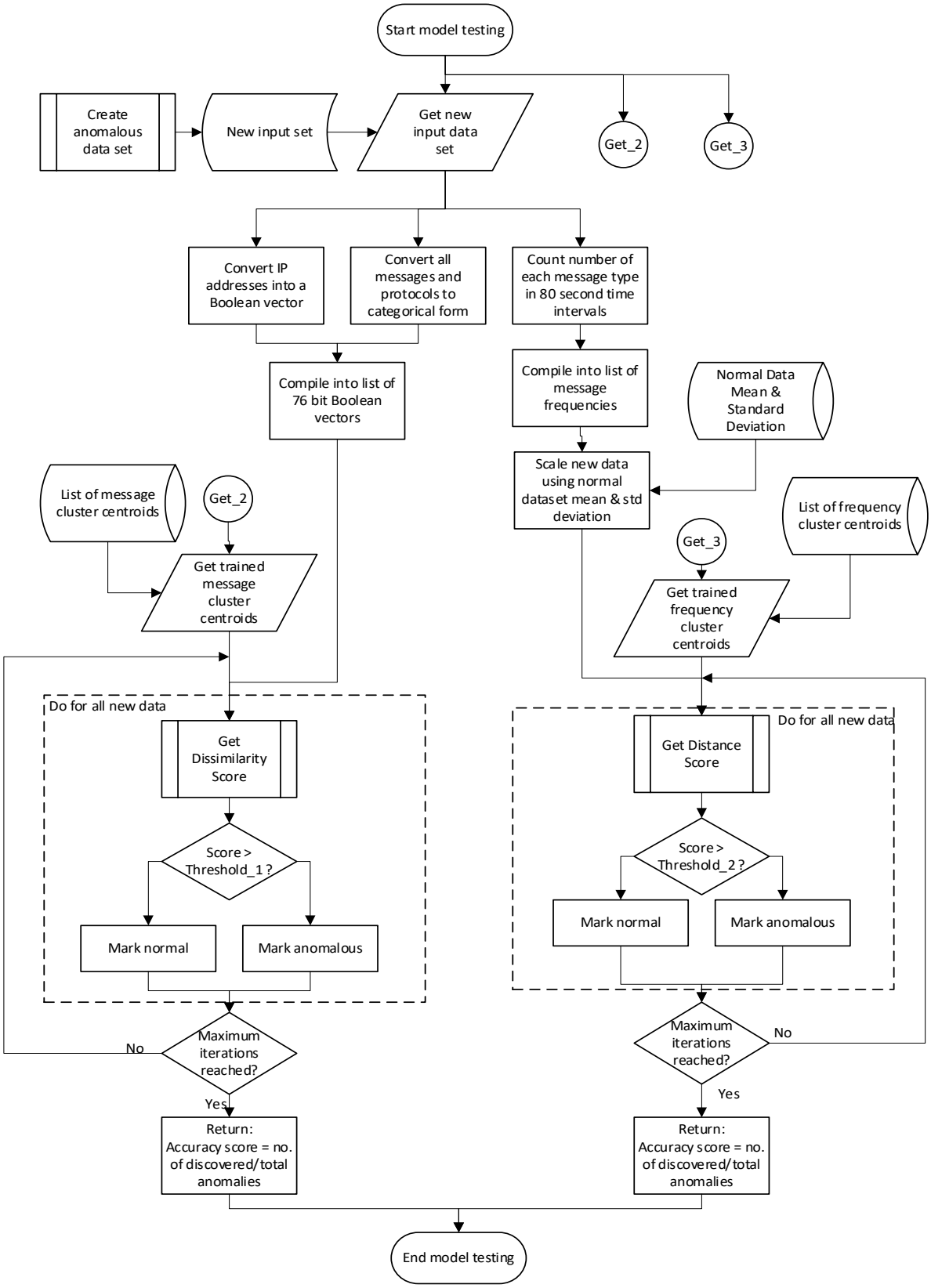


Figure 51: Final model flowchart for the GUI

CHAPTER IV

CONCLUSION

Despite the hiccups caused by social distancing, the project is close to completion. The focus of the project, however, changed from emulating a BACnet system using Raspberry Pis to training a machine learning model. Unsupervised clustering algorithms such as K-modes and K-means turned out to be very useful in determining anomalies in the data. The anomaly IP detection using K-modes and the anomaly frequency of input using K-means proved to be effective methods of finding any anomalies in the data. Using distance metrics such as the Jaccard distance and Squared Euclidean distances proved to be effective in determining anomalies in the user inputs. However, obtaining an accuracy almost as close to 100% proves to be 'too good to be true' such that the K-modes and K-means models need to be fed more data. In addition, a prototype GUI was created to allow users to track anomalies and generate BACnet protocol messages to test the model. This model worked very well in predicting any anomalies but there needs to be more work done in obtaining more data to optimize and improve the learning algorithm. Optimizing this machine learning algorithm could be done using K-prototypes which includes both categorical and normal data. Additionally, finding the most important features/columns using PCA could also improve the machine learning algorithm. Finally, our model would also be able to integrate and detect more forms of intruder attacks that will prove to be useful in detecting any anomalous data in a BMS system.

REFERENCES

[1] Ashrae.org. (2019). *Standard 135-2016, BACnet™ -- A Data Communication Protocol for Building Automation and Control Networks*. [online] Available at: https://www.ashrae.org/technical-resources/standards-and-guidelines/standards-addenda/standard-135-2016-bacnet-a-data-communication-protocol-for-building-automation-and-control-networks [Accessed 6 Dec. 2019].

[2] Standards.ieee.org. (2019). *IEEE 802.3-2018 - IEEE Standard for Ethernet*. [online] Available at: https://standards.ieee.org/standard/802_3-2018.html [Accessed 6 Dec. 2019].

[3] Chandrayan, P. (2019). *Deep Learning: Autoencoders Fundamentals and types*. [online] Medium. Available at: https://codeburst.io/deep-learning-types-and-autoencoders-a40ee6754663 [Accessed 6 Dec. 2019].

[4] OpenGenus IQ: Learn Computer Science. (2019). *Different types of Autoencoders*. [online] Available at: https://iq.opengenus.org/types-of-autoencoder/ [Accessed 6 Dec. 2019].

[5] Z. Zheng and A. Reddy, "Safeguarding Building Automation Networks: THE-Driven Anomaly Detector Based on Traffic Analysis," 2017. [Online]. Available: http://cesg.tamu.edu/wp-content/uploads/2018/01/ICCCN-Zhiyuan-Zheng-Paper-2017-July-1.pdf. [Accessed: 08- Sep- 2019].

[6] Cimetrics. (2019). *uBACstac - BACnet Protocol stack for small devices*. [online] Available at: https://www.cimetrics.com/products/products-bacnet-ubacstac [Accessed 8 Sep. 2019].

[7] Ferreira, H. (2019). *Confusion matrix and other metrics in machine learning*. [online] Medium. Available at: https://medium.com/hugo-ferreiras-blog/confusion-matrix-and-other-metrics-in-machine-learning-894688cb1c0a [Accessed 6 Dec. 2019].

[8] G, A. (2019). *abelusha/AutoEncoders-for-Anomaly-Detection*. [online] GitHub. Available at: https://github.com/abelusha/AutoEncoders-for-Anomaly-Detection [Accessed 6 Dec. 2019].

[9]"Hands-on Machine Learning on Google Cloud Platform", *Subscription.packtpub.com*, 2019. [Online]. Available:https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788393485/6/ch06lvl1sec39/supervised-and-unsupervised-machine-learning . [Accessed: 08- Sep- 2019].

[10] Dms.hvacpartners.com. (2019). *Bacnet Basics User's Guide*. [online] Available at: https://dms.hvacpartners.com/docs/1000/Public/04/11-808-417-01.pdf [Accessed 3 Sep. 2019].

[11] Bushby, S. (2003). *BACnet(trademark): A Standard Communication Infrastructure for Intelligent Buildings.*. [online] NIST. Available at: https://www.nist.gov/publications/bacnettrademark-standard-communication-infrastructure-intelligent-buildings [Accessed 6 Dec. 2019].

[12] Donges, N. (2019). *Basic Linear Algebra for Deep Learning*. [online] Medium. Available at: https://towardsdatascience.com/linear-algebra-for-deep-learning-f21d7e7d7f23 [Accessed 6 Dec. 2019].

[13] Brownlee, J. (2019). *5 Reasons to Learn Probability for Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/why-learn-probability-for-machine-learning/ [Accessed 6 Dec. 2019].

[14] C. Zimmer et. al, "Time-based intrusion detection in cyber-physical systems," in *ICCPS'10 Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems, Stockholm, Sweden, April 13-15, 2010*, ACM, New York, USA, 2010, pp. 109-118.

[15] N. Goldberg et. al, "Accurate modeling of Modbus/TCP forintrusion detection in SCADA systems," *International Journal of Critical Infrastructure Protection,* vol. 6, issue 2, June 2013, pp. 63-75. [online]. Available at: https://www.sciencedirect.com/science/article/pii/S1874548213000243. [Accessed 8 Sep 2019].

[16] Z. Zheng and A. Reddy, "Safeguarding Building Automation Networks: THE-Driven Anomaly Detector Based on Traffic Analysis," 2017. [Online]. Available: http://cesg.tamu.edu/wp-content/uploads/2018/01/ICCCN-Zhiyuan-Zheng-Paper-2017-July-1.pdf. [Accessed: 08- Sep- 2019].

[17] S. Lyons, "Current Status of Cyber Security in the BAS Industry," *Cimetrics February 2019 Newsletter*, Mar 1, 2019. [online]. Available at: https://www.cimetrics.com. [Accessed 8 Sep 2019].

[18] Metadata, https://censys.io/ipv4/metadata?q=bacnet&

[19] D. Bhattacharyya and J. Kalita, "Introduction," Network Anomaly Detection: A Machine Learning Perspective, New York: CRC Press, 2014, pp. 1-13.

[20] J. Tonejc et al, "Machine Learning Methods for Anomaly Detection in BACnet Networks," *Journal of Universal Computer Science*, vol. 22, no. 9 (2016), 1203-1224. [online] Available at: https://pdfs.semanticscholar.org/d823/6a08011ad5f33e5d5c8f20d87c85a08bf784.pdf [Accessed 8 Sep 2019]

[21] P. Liang and D. Klein, "Analyzing the Errors of Unsupervised Learning," CS Division, EECS Department, Univ. of Cali., Berkeley, CA 94720, USA, June 2008. [online] Available at: https://pdfs.semanticscholar.org/038a/fe82cc61215e9087e572d8aab9663c1bdb0f.pdf. [Accessed 8 Sep 2019]

[22] S. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica* 31, 2007, pp. 249-268. [Online]. Available at: http://www.informatica.si/index.php/informatica/article/viewFile/148/140. [Accessed 8 Sep 2019]

APPENDIX

BACnet Survey Form

Building Automation Systems (BAS) are softwares that can connect and automate relevant devices/sensors like a/c vents, electronic locks, lighting, supply lines and other utilities in a building. Of these systems, the BACnet (Building Automation Control network) is a widely used standard protocol for BAS systems. BACnet is preferred because it employs basic encryption methods as compared to most others which are completely in clear text, but it is still very vulnerable to cyber and physical attacks.

Our project is to identify/detect hackers who have gained access to the Building Automation Systems by implementing data collection and machine learning on the BACnet protocol.

Please help us understand the general opinion on BAS security by taking a few minutes of your time to complete this survey. Thank you.

* Required

1. What is your profession? *

_____

2. Does any building you regularly use have a Building Automation System?

Mark only one oval.

- ◯ Yes, one of them does
- ◯ Yes, more than one of them does
- ◯ No
- ◯ Maybe

3. Why do you think BACnet systems are important?

Check all that apply.

- ☐ To allow for interoperable (compatible communication without restrictions) communications between devices
- ☐ To detect hackers in a Building Management System
- ☐ To improve the automation capabilities of Building Management Systems

4. Do you need security in BACnet devices?

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: _____

5. Do you think authentication is required every time you connect to a Building Automation System (BAS) device?

Mark only one oval.

- ◯ Yes
- ◯ No

1/2

6. **Are automated detection of threats on BACnet systems a feasible option?**
Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: _____

7. **What information would you target if you get access to a BACnet device?**

_____

8. **What is your opinion on data collection for cyber-security measures in BAS?**
Mark only one oval.

- ◯ It is a violation of my privacy
- ◯ It is necessary, but I don't like it
- ◯ It is necessary and I don't really mind it
- ◯ It is unnecessary
- ◯ Other: _____

BACnet survey

Building Automation Systems (BAS) are softwares that can connect and automate relevant devices like a/c vents, electronic locks, lighting, supply lines and other utilities in a building. Of these systems, the BACnet (Building Automation Control network) is a widely used standard which provides a wide platform for network integration and supports a wide selection of devices. It is preferred because it uses some basic encryption methods as compared to most other BAS which do not, but the BACnet is still very vulnerable to cyber and physical attacks. This is especially the case as the number of BAS networks is increasing these days.

Our project is aimed at implementing data collection and machine learning to teach a software how to detect attacks or threats on a BACnet system.

Please help us understand the general opinion on BAS security by taking a few minutes of your time to complete this survey. Thank you.

1. What is your profession?

_____

2. Do you think authentication is required every time you connect to a Building Automation System (BAS) device?

Mark only one oval.

- ◯ Yes
- ◯ No

3. What information do you think can be obtained from hacking into Building Automation Systems? Check all that apply:

Check all that apply.

- ☐ Credit card/Debit card information
- ☐ Personal information about employees
- ☐ Individual/Team's working hours or schedule
- ☐ Information of critical services like gas line monitoring and air conditioning
- ☐ IP addresses of BAS devices
- ☐ Other: _____

4. What sensors do you think a BACnet connected building includes? Check all that apply:

Check all that apply.

- [ ] Motion Sensors
- [ ] Temperature sensors (control over the temperature of a room instead of one central Air Conditioner)
- [ ] Pressure sensors
- [ ] Fine dust sensors/Air quality sensors
- [ ] Water tank level and quality sensors
- [ ] Energy consumption monitors
- [ ] Other: _____

5. Which of these attacks do you think a BACnet is most likely to face?

Mark only one oval.

- ◯ Physical attacks on BAS devices
- ◯ Man in the middle (data interception and impersonation) attacks
- ◯ Denial of service (brute force spamming of signal traffic) attacks
- ◯ Password hacking
- ◯ Other: _____

6. What security features do you have in your BAS(if you use one)? Check all that apply:

Check all that apply.

- [ ] Automated regular data collection
- [ ] password protection to control access
- [ ] Data Encryption and distribution of trusted keys
- [ ] Hired security guards to monitor physical activity around BAS devices
- [ ] Access authentication using challenge-response mechanism
- [ ] Not applicable. I do not use/have a BAS.

7. Of the above, which features would you like to have(if you don't already)? Check all that apply:

Check all that apply.

- [ ] Automated regular data collection
- [ ] password protection to control access
- [ ] Data Encryption and distribution of trusted keys
- [ ] Hired security guards to monitor physical activity around BAS devices
- [ ] Access authentication using challenge-response mechanism
- [ ] Other: _____

8. **What would you like Building Automation Systems to notinclude?**

9. **Are automated detection of threats on BACnet systems a feasible option?**
Mark only one oval.

- Strongly disagree
- Disagree
- Neutral
- Strongly agree
- Agree

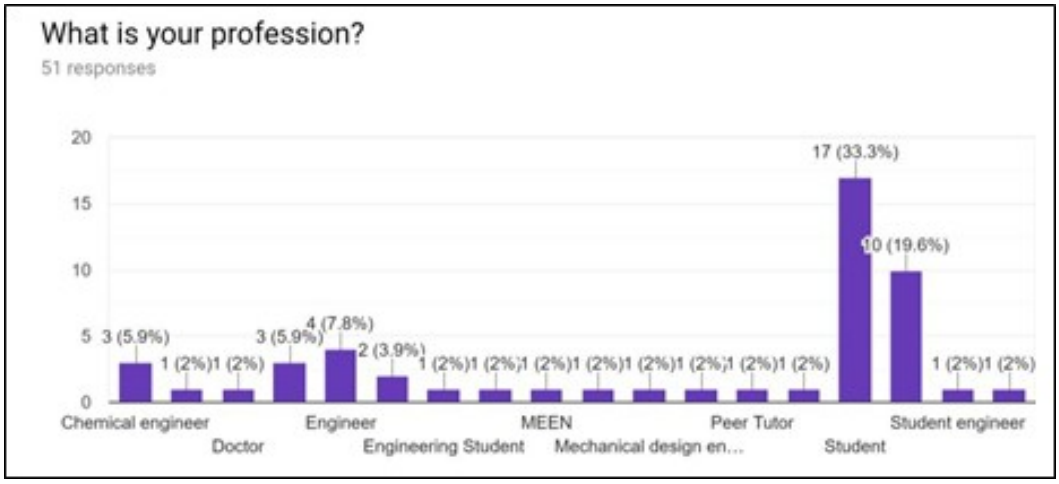
10. **What is your opinion on data collection to prevent cyber-physical attacks on BAS?**
Mark only one oval.

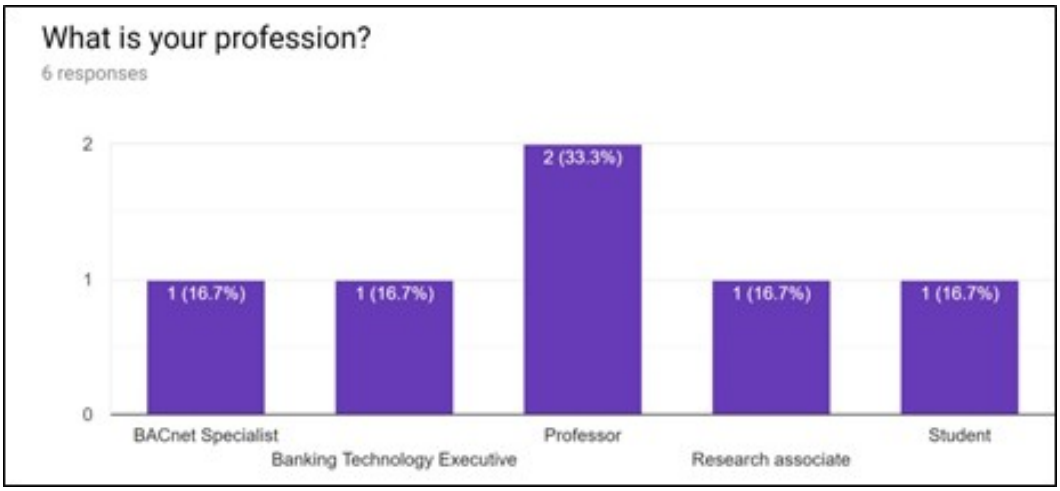
- It is a violation of my privacy
- It is necessary for security
- It is necessary but I don't like it
- It is necessary, but I would like to have minimal amounts of data collection in people's working spaces

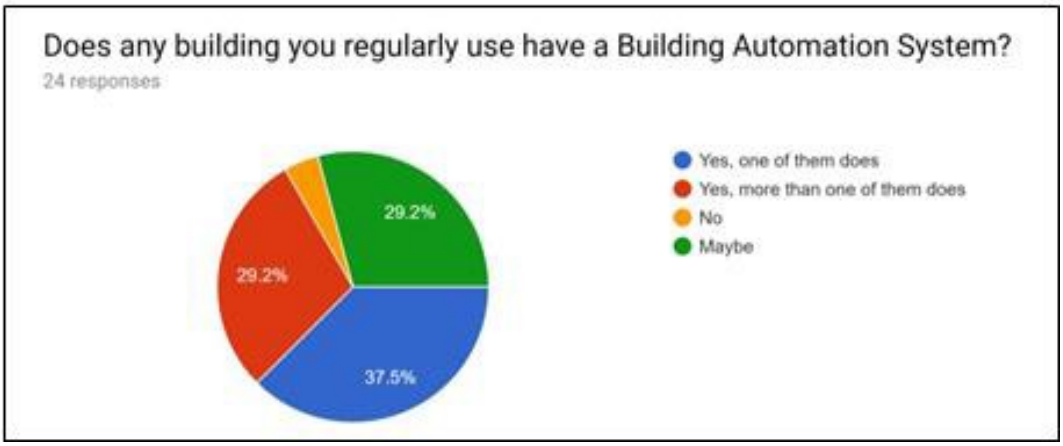
11. **On a scale of 1 to 10, how severe is your need for security measures in your building automation system?**
Mark only one oval.

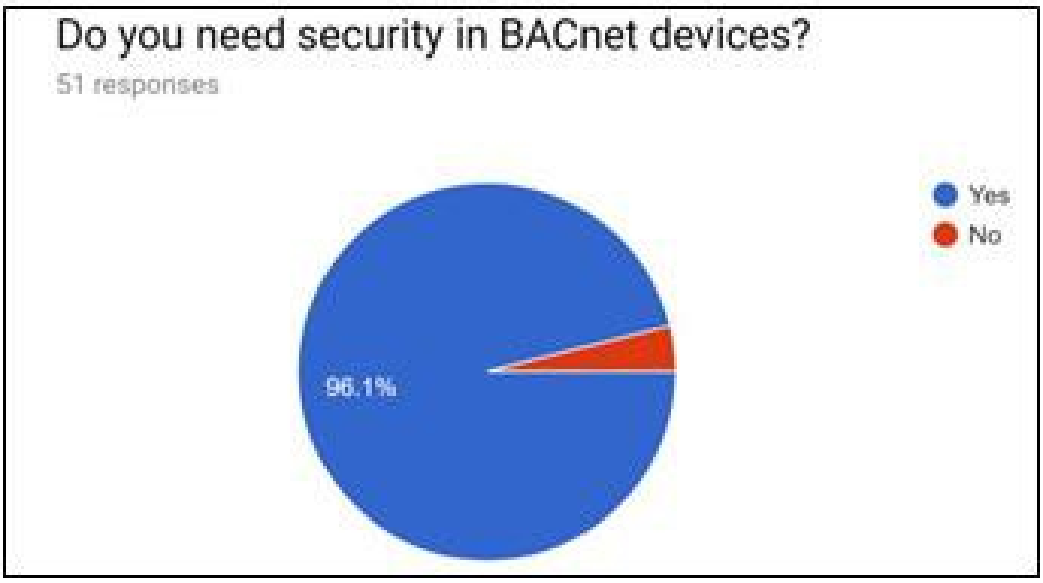
	1	2	3	4	5	6	7	8	9	10	
Not really necessary											We need it right now

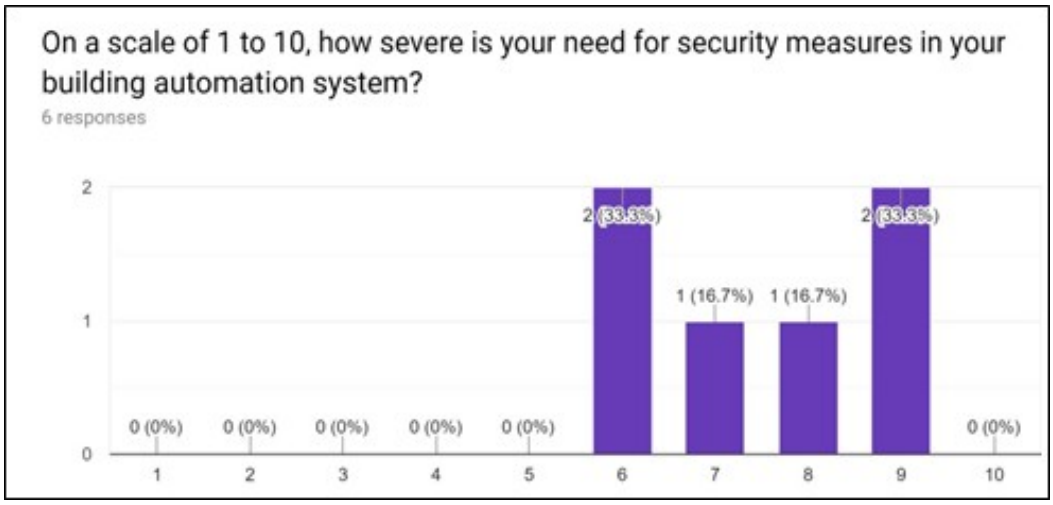
12. **Thanks for giving this survey your time. We appreciate your opinions and would like to hear your thoughts on this survey and/or on BACnet (BAS) security in general. Please do tell us here:**

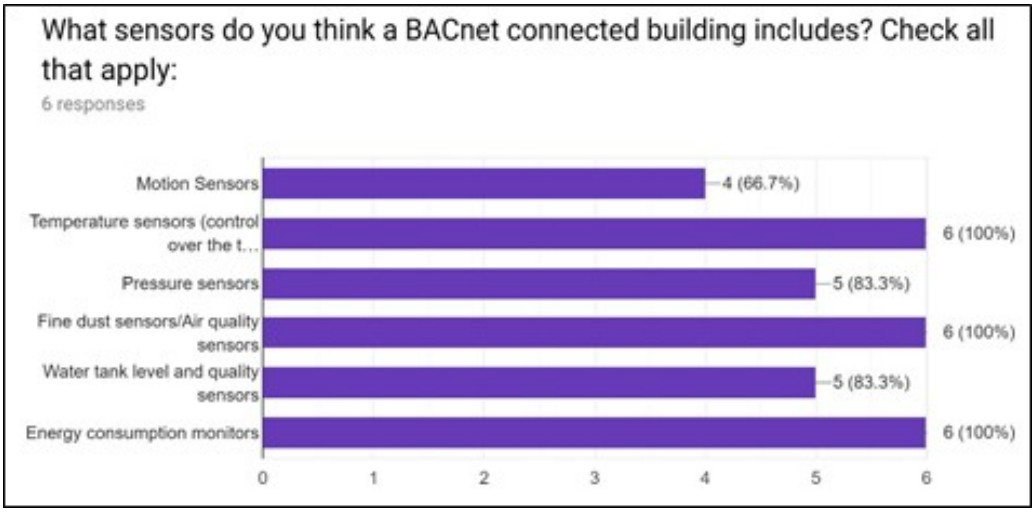

What is your profession?
51 responses
20
15
10
5
0
3 (5.9%)
1 (2%)
1 (2%)
3 (5.9%)
4 (7.8%)
2 (3.9%)
1 (2%)
1 (2%)
1 (2%)
1 (2%)
1 (2%)
1 (2%)
1 (2%)
1 (2%)
17 (33.3%)
10 (19.6%)
1 (2%)
1 (2%)
Chemical engineer
Doctor
Engineer
Engineering Student
MEEN
Mechanical design en…
Peer Tutor
Student
Student engineer

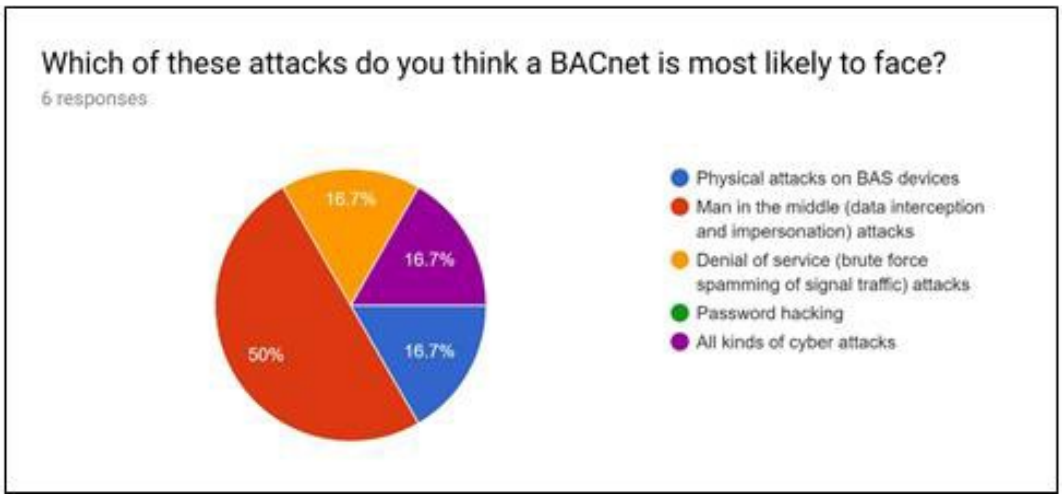

What is your profession?
6 responses
2
1
0
1 (16.7%)
1 (16.7%)
2 (33.3%)
1 (16.7%)
1 (16.7%)
BACnet Specialist
Banking Technology Executive
Professor
Research associate
Student

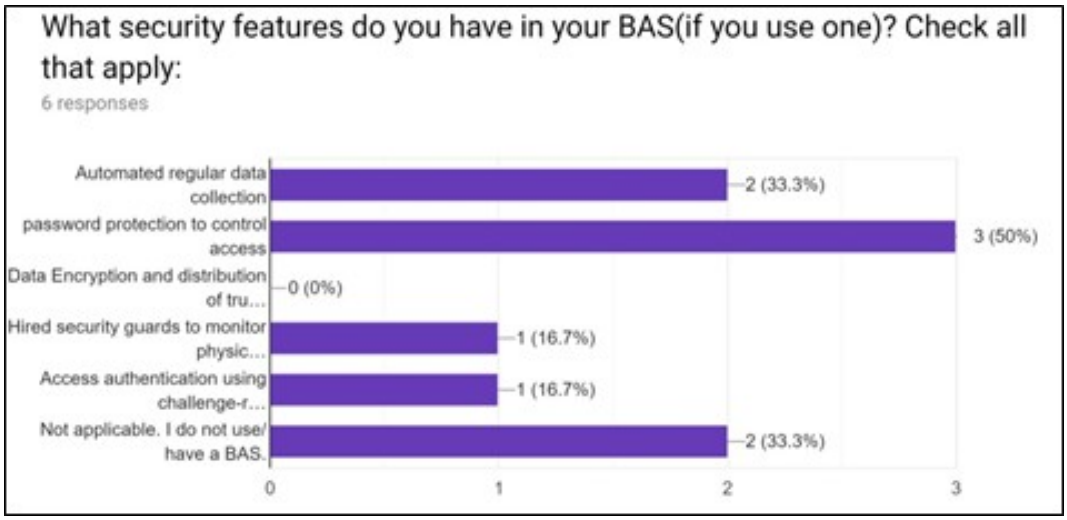

Does any building you regularly use have a Building Automation System?
24 responses
29.2%
29.2%
37.5%
Yes, one of them does
Yes, more than one of them does
No
Maybe

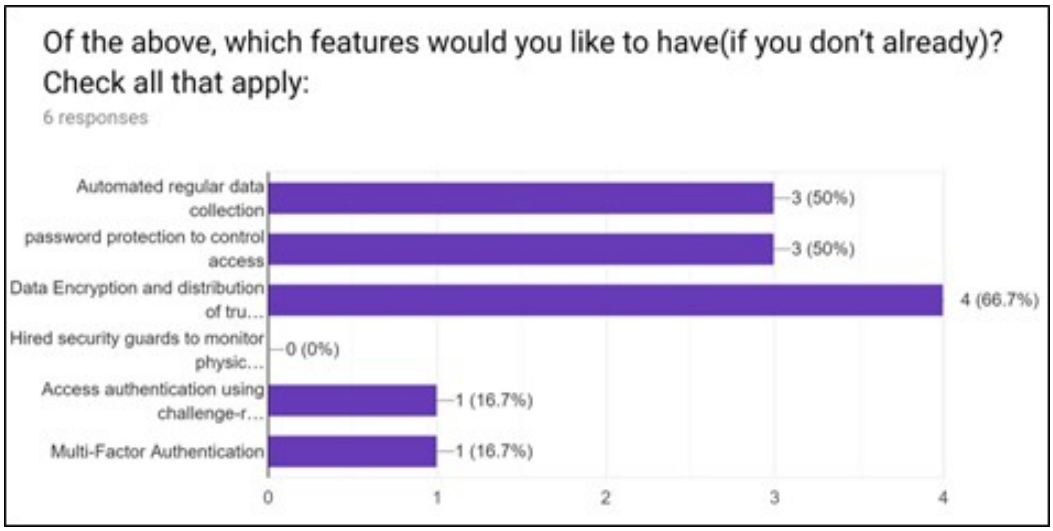

Do you need security in BACnet devices?
51 responses
Yes
No
96.1%


On a scale of 1 to 10, how severe is your need for security measures in your building automation system?
6 responses
2
1
0
0 (0%)
0 (0%)
0 (0%)
0 (0%)
0 (0%)
2 (33.3%)
1 (16.7%)
1 (16.7%)
2 (33.3%)
0 (0%)
1
2
3
4
5
6
7
8
9
10


What sensors do you think a BACnet connected building includes? Check all that apply:
6 responses
Motion Sensors
4 (66.7%)
Temperature sensors (control over the t...
6 (100%)
Pressure sensors
5 (83.3%)
Fine dust sensors/Air quality sensors
6 (100%)
Water tank level and quality sensors
5 (83.3%)
Energy consumption monitors
6 (100%)
0
1
2
3
4
5
6

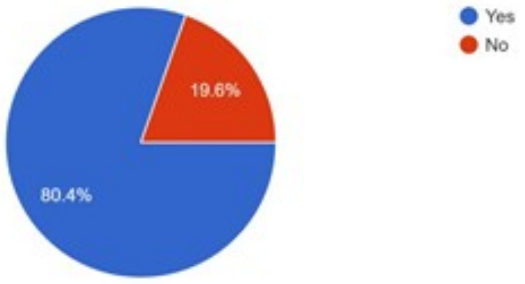

Which of these attacks do you think a BACnet is most likely to face?
6 responses
16.7%
16.7%
50%
16.7%
Physical attacks on BAS devices
Man in the middle (data interception and impersonation) attacks
Denial of service (brute force spamming of signal traffic) attacks
Password hacking
All kinds of cyber attacks


What security features do you have in your BAS(if you use one)? Check all that apply:
6 responses
Automated regular data collection
2 (33.3%)
password protection to control access
3 (50%)
Data Encryption and distribution of tru...
0 (0%)
Hired security guards to monitor physic...
1 (16.7%)
Access authentication using challenge-r...
1 (16.7%)
Not applicable. I do not use/ have a BAS.
2 (33.3%)
0
1
2
3


Of the above, which features would you like to have(if you don't already)? Check all that apply:
6 responses
Automated regular data collection
3 (50%)
password protection to control access
3 (50%)
Data Encryption and distribution of tru...
4 (66.7%)
Hired security guards to monitor physic...
0 (0%)
Access authentication using challenge-r...
1 (16.7%)
Multi-Factor Authentication
1 (16.7%)
0
1
2
3
4

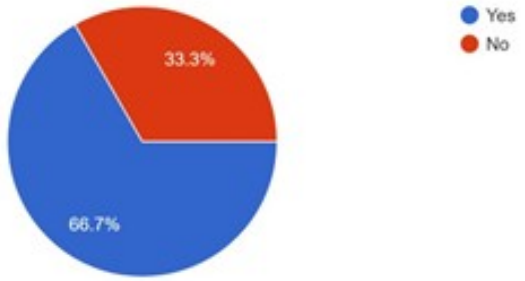
Do you think authentication is required every time you connect to a Building Automation System (BAS) device?

51 responses



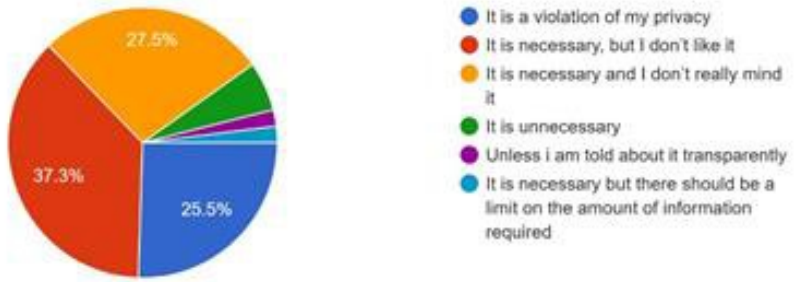
Do you think authentication is required every time you connect to a Building Automation System (BAS) device?

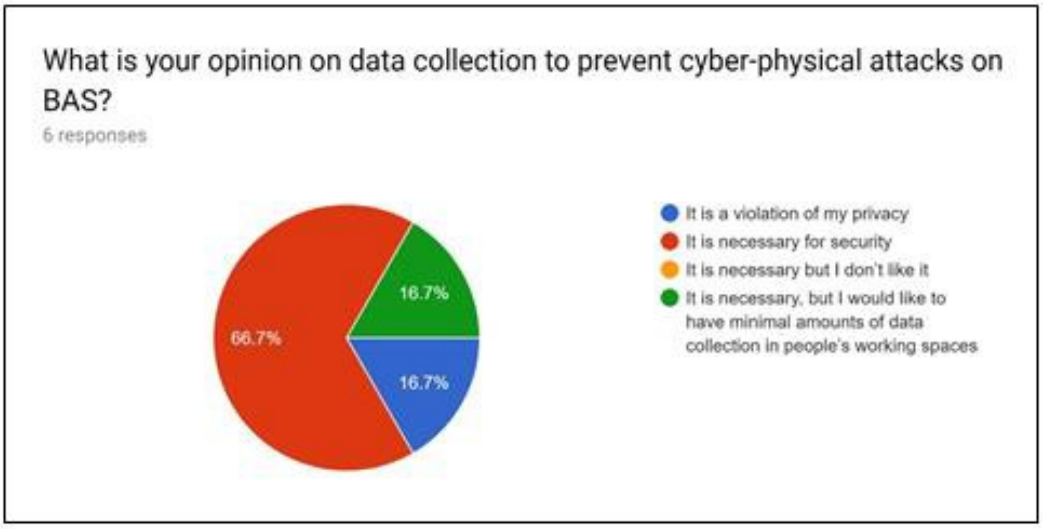
6 responses

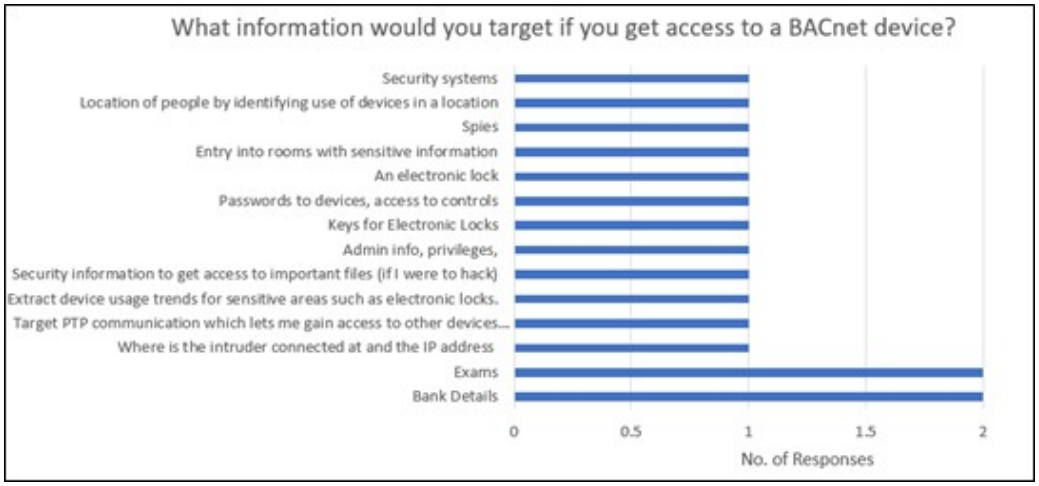


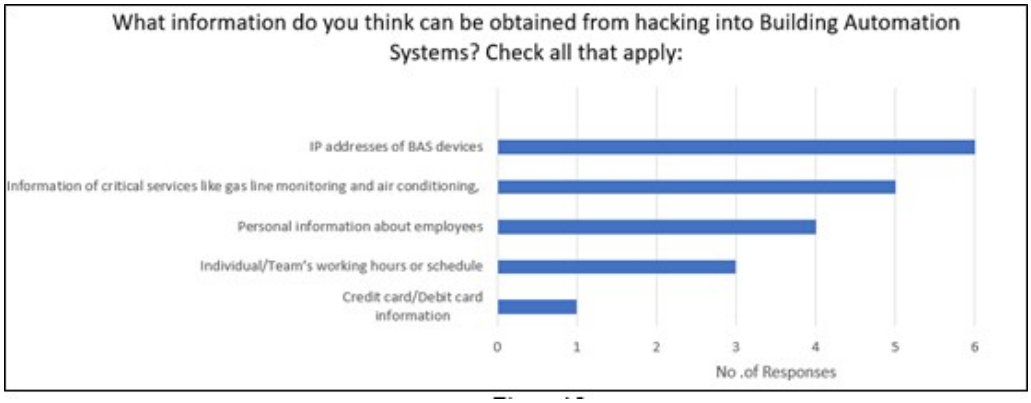
What is your opinion on data collection for cyber-security measures in BAS?

51 responses




What is your opinion on data collection to prevent cyber-physical attacks on BAS?
6 responses
16.7%
66.7%
16.7%
It is a violation of my privacy
It is necessary for security
It is necessary but I don't like it
It is necessary, but I would like to have minimal amounts of data collection in people's working spaces


What information would you target if you get access to a BACnet device?
Security systems
Location of people by identifying use of devices in a location
Spies
Entry into rooms with sensitive information
An electronic lock
Passwords to devices, access to controls
Keys for Electronic Locks
Admin info, privileges,
Security information to get access to important files (if I were to hack)
Extract device usage trends for sensitive areas such as electronic locks.
Target PTP communication which lets me gain access to other devices...
Where is the intruder connected at and the IP address
Exams
Bank Details
0
0.5
1
1.5
2
No. of Responses


What information do you think can be obtained from hacking into Building Automation Systems? Check all that apply:
IP addresses of BAS devices
Information of critical services like gas line monitoring and air conditioning,
Personal information about employees
Individual/Team's working hours or schedule
Credit card/Debit card information
0
1
2
3
4
5
6
No .of Responses

GUI DESIGN

```python
import tkinter as tk
from PIL import ImageTk, Image
from tkinter import font
import pandas as pd
from scipy.spatial import distance
import numpy as np
from sklearn.preprocessing import StandardScaler

Clusters = pd.read_csv('clusterInfo_14.csv')
Clusters = Clusters.loc[:, ~Clusters.columns.str.contains('unnamed', case=False)]
clusters_3 = pd.read_csv('3_clusters.csv')
HEIGHT = 600 #700 pixels
WIDTH  = 700

root = tk.Tk()
root.title('Anomaly Detector Demo')
root.iconbitmap('D:\\Python\\Ideas\\tamuq_ecen.png')
# root.geometry('800x800')

# root = tk.Tk() #root window to place everything into

# This function is to calculate the anomaly
def anomaly_values(user_input,inputs):

    scaler = StandardScaler()
    columns = list(clusters_3.columns)
    listofzeros = [0]*len(columns)
    msgs_df = dict(zip(columns,listofzeros))
    msgs_df = pd.DataFrame(msgs_df,index=[0])
    scaler.fit(clusters_3.values)
    Anom_data = scaler.transform(msgs_df.values)

    jaccard_distances = []
    Cluster_array = Clusters.values
    # comparing the jaccard distances of the input with each of the 14 centroids to see which cluster it could/could not belong to
    for i in range(len(Clusters)):
        anomaly_score = distance.jaccard(user_input.values,Cluster_array[i]) # computing the jaccard distances
        jaccard_distances.append(anomaly_score)

    anom_freq_score = []
    centroid_values = clusters_3.values
    Scaled_normal_array = Anom_data

    for j in range(clusters_3.shape[0]):
        score = distance.sqeuclidean(centroid_values[j],Scaled_normal_array)
        anom_freq_score.append(score)

    scores = np.asarray(jaccard_distances) # converting to numpy array
    # scores_freq = np.asarray(anom_freq_score)

    # We obtain the minimum vaue of jaccard distance to see which is the closest/nearest cluster it could belong to
    anomaly_val = np.amin(scores)
    freq_val = np.amin(anom_freq_score)
    return anomaly_val, freq_val
```

```
def decision(inputs):
    message_input = 'Messages_{}'.format(inputs[2])
    protocol_input = 'Protocol_{}'.format(inputs[3])# obtaining in list format
    IP_Value = inputs[0:2]
    IP_extraction = []

    # Obtaining the binary representation of the ip addresses
    for i in IP_Value:
        ip = ''.join([bin(int(x)+256)[3:] for x in i.split('.')])
        IP_extraction.append(ip)

    dict_IP_all = dict(zip(IP_Value, IP_extraction))

    #creating dataframe for messages and protocol
    columns = list(Clusters.columns[-12:,]) # obtaining the list of columns to be the keys
    listofzeros = [0]*len(columns) # assigning zeros to be the values for all the messages and protocols
    msgs_protocol = dict(zip(columns,listofzeros)) # creating a dictionary
    msgs_protocol[message_input] = 1 # changing the value of whatever message that was input to be 1
    msgs_protocol[protocol_input] = 1  # changing the value of whatever protocol that was input to be 1
    msgAndProtocol_df = pd.DataFrame(msgs_protocol,index = [0])

    user_val = pd.DataFrame([dict_IP_all[inputs[0]]]) # creating a dataframe for the source
    user_val2 = pd.DataFrame([dict_IP_all[inputs[1]]]) # creating a dataframe for the destination
    user_val = user_val.rename(columns={0: 'IP'})
    user_val2 = user_val2.rename(columns={0: 'dest'})

    # creating dataframe for source ip to be bit1,bit2....bit32
    user_val = user_val['IP'].apply(lambda x: pd.Series(list(x)))
    a1 = user_val.columns[np.arange(0,32)]
    a2 = ['bit' + str(i) for i in np.arange(1,len(a1)+1)]
    a3 = dict(zip(a1, a2))
    user_val = user_val.rename(columns=a3)
    user_val = user_val.astype(int)
```

```
    # creating dataframe for source ip to be bit33,bit32....bit64
    user_val2 = user_val2['dest'].apply(lambda y: pd.Series(list(y)))
    b1 = user_val2.columns[np.arange(0,32)]
    b2 = ['bit' + str(i) for i in np.arange(33,len(b1)+33)]
    b3 = dict(zip(b1, b2))
    user_val2 = user_val2.rename(columns=b3)
    user_val2 = user_val2.astype(int)

    user_input = pd.concat([user_val, user_val2,msgAndProtocol_df], axis=1)
    check_anomaly,freq_anomaly = anomaly_values(user_input,inputs)

    # check if the anomaly score is greater than the threshold
    # if it is then we assign the boolean to be true that the input was an anomaly
    bms_output = 0
    if check_anomaly > 0.45:
        bms_output = 1

    freq_output = 1
    if freq_anomaly>25.21:
        freq_output = 1

    return bms_output, check_anomaly,freq_anomaly, freq_output
```

```
# This function is to test the inputs
def test_function(entry):
    # myLabel = tk.Label(root, )
    inputs = []
    for x in entry:
        # print(x.get())
        inputs.append(str(x.get()))

    print(inputs[4],type(inputs[4]))

    output = 'Your inputs are: \n{0} | {1} | {2} | {3}\n'.format(inputs[0],inputs[1],inputs[2],inputs[3])
# if there is no frequency involved then it would undergo the kmodes detection algorithm
    checking_val = decision(inputs[0:4])
    if checking_val[0] == 1  or checking_val[3] == 1:
        test_output = 'an anomaly'
    else:
        test_output = 'Not an anomaly'


    label_o['text'] = '{0}\nYour input is {1} with a score of {2} \nfor a frequency input of {3} and score of {4}'.format(output, test_
```

```
canvas = tk.Canvas(root,height = HEIGHT, width = WIDTH) #create size of window
canvas.pack()


path = 'D:\\Python\\Ideas\\black_background.png'
background_image = tk.PhotoImage(file = path)
background_label = tk.Label(root, image = background_image)
background_label.place(x= 0,y=0, relwidth = 1, relheight = 1)

# creating frame for all the user inputs so that it can have an entry and a label
frame_S = tk.Frame(root, bg = '#80b3ff', bd = 4)
frame_S.place(relx = 0.5, rely = 0.02, relwidth = 0.9, relheight = 0.08,anchor = 'n')

label = tk.Label(frame_S, text = 'Source', font = ('Ink Free', 18), anchor = 'w')
#increasing the font size doesnt make much of a difference so importing font from tkinter
label.place(relx = 0, rely = 0, relwidth = 0.3, relheight = 1)
# label.pack(side = tk.LEFT)

entry1 = tk.Entry(frame_S,font = ('Ink Free', 20))
entry1.place(relx = 0.2, rely = 0, relwidth = 0.8, relheight = 1)
# entry1.pack(side = tk.RIGHT)


frame_D = tk.Frame(root, bg = '#80b3ff', bd = 4)
frame_D.place(relx = 0.5, rely = 0.12, relwidth = 0.9, relheight = 0.08,anchor = 'n')

label2 = tk.Label(frame_D, text = 'Destination', font = ('Ink Free', 18), anchor = 'w')
#increasing the font size doesnt make much of a difference so importing font from tkinter
label2.place(relx = 0, rely = 0, relwidth = 1, relheight = 1)

entry2 = tk.Entry(frame_D,font = ('Ink Free', 20))
entry2.place(relx = 0.2, rely = 0, relwidth = 0.8, relheight = 1)
```

```
frame_M = tk.Frame(root, bg = '#80b3ff', bd = 4)
frame_M.place(relx = 0.5, rely = 0.22, relwidth = 0.9, relheight = 0.08,anchor = 'n')

label3 = tk.Label(frame_M, text = 'Message', font = ('Ink Free', 18), anchor = 'w') #increasing the font size doesnt make much of a dif
label3.place(relx = 0, rely = 0, relwidth = 1, relheight = 1)

entry3 = tk.Entry(frame_M,font = ('Ink Free', 20))
entry3.place(relx = 0.2, rely = 0,relwidth = 0.8, relheight = 1)


frame_P = tk.Frame(root, bg = '#80b3ff', bd = 4)
frame_P.place(relx = 0.5, rely = 0.32, relwidth = 0.9, relheight = 0.08,anchor = 'n')

label3 = tk.Label(frame_P, text = 'Protocol', font = ('Ink Free', 18), anchor = 'w') #increasing the font size doesnt make much of a di
label3.place(relx = 0, rely = 0, relwidth = 1, relheight = 1)

entry4 = tk.Entry(frame_P,font = ('Ink Free', 20))
entry4.place(relx = 0.2, rely = 0,relwidth = 0.8, relheight = 1)


frame_F = tk.Frame(root, bg = '#80b3ff', bd = 4)
frame_F.place(relx = 0.5, rely = 0.43, relwidth = 0.9, relheight = 0.08,anchor = 'n')

label4 = tk.Label(frame_F, text = 'Frequency', font = ('Ink Free', 18), anchor = 'w') #increasing the font size doesnt make much of a d
label4.place(relx = 0, rely = 0, relwidth = 1, relheight = 1)

entry5 = tk.Entry(frame_F,font = ('Ink Free', 20))
entry5.place(relx = 0.2, rely = 0,relwidth = 0.8, relheight = 1)


entries = [entry1, entry2, entry3, entry4, entry5]

```