Cascading Three Step Approach for Anomaly Detection in Unsupervised Communication Meta Data of IP-based Internet of Things Devices

Andreas Schäfer and Michael Massoth
Department of Computer Science
Hochschule Darmstadt (h_da) – University of Applied Sciences Darmstadt,
and Center for Research in Security and Privacy (CRISP), Darmstadt, Germany
E-mail: andreas.schaefer@stud.h-da.de; michael.massoth@h-da.de

***Abstract*— Internet of Things (IoT) are globally connected devices which are able to collect and exchange information. The increasing usage of IoT-devices in industrial and private environments result in the need for higher security and constant surveillance of such devices. Since 2016 novel botnets, consisting only of IoT-devices, where observed to execute major Distributed Denial of Service (DDoS) attacks. Due to the autonomous nature of these IoT devices, a compromised device might never be detected by system administrators. This creates the need for continuous monitoring of IoT network traffic. A possible solution for this problem is the permanent monitoring of anomalies within the network traffic of the IoT devices. Anomaly Detection Systems (ADS) monitor the behavior of a system and flag significant deviations from the normal activity as anomalies. This paper presents a new three step approach for anomaly detection in unsupervised communication meta data by cascading X-means clustering, decision tree, and statistical analysis, in order to monitor and protect IoT networks.**

Keywords-anomaly detection; internet of things; unsupervised machine learning; intrusion detection and prevention

I. INTRODUCTION

The Internet of Things (IoT) could be defined as a huge set of sensors and actuators, embedded in physical objects, which are linked through wired and/or wireless networks, often using the same Internet Protocol (IP), that connects the Internet [6]. The basic idea is that devices (things = sensors and actuators) perform tasks independently from human interaction and are connected to the Internet [7]. IoT devices work to a large extent with information from their immediate surroundings. This is to support people in their everyday life or in their work. As a rule, IoT devices behave as predictable as possible, since they are based on fixed implemented algorithms, and human influence on the devices is therefore minimal to non-existent.

The use of IoT is becoming increasingly widespread worldwide. It is estimated that by 2020 more than 30 billion devices of this kind will be in use around the world. The rapidly developing market and the high price pressure on manufacturers often result in insufficient investment in the security features of the IoT devices. Software is usually updated only rarely or not at all. This and the easy accessibility of the IoT devices via the Internet increases the risk of possible compromise by attackers. At the end of 2016, IoT botnets were first widely used in Distributed Denial of Service (DDos) attacks around the world. One of these attacks on the provider Dyn reached a bandwidth of 1.2 Terabits per second (TBps) and temporarily crippled platforms like Twitter, Amazon, CNN, PayPal and many other sites. Therefore, the vulnerability of these IoT devices has therefore often been criticized by IT security experts and researchers in the past. However, the lack of security of IoT devices is not only a danger for potential targets of botnets, the owners of the devices also have a great interest in anticipating a compromise. Especially in industrial environments, IoT devices are playing an increasingly important role in business processes, for example as production machines, part of the infrastructure or as sensors. These devices could also be sabotaged or misused by attackers for espionage purposes. As the number of IoT devices in use increases, so does the threat to businesses and the global threat posed by botnets. It is therefore essential for companies to monitor their IoT devices to detect possible compromises. Anomaly Detection Systems (ADS) [8] monitor the behavior of the IoT system and flag significant deviations from the normal activity as anomalies [11]. This paper presents a new approach for anomaly detection in unsupervised communication meta data of IP-based IoT devices by cascading X-means clustering, decision tree, statistical analysis, and the computation and monitoring of trust values for the monitored individual IoT devices.

The paper is structured as follows. In Section II, the focus of work, field of application and validity, as well as a short overview of the requirement specifications and the restrictions of the new anomaly detection approach is given. Section III presents some related work. In Section IV, the concept and used methods are discussed in more detail, including the following topics: Anomaly-based intrusion detection and prevention, unsupervised machine learning, the collection of meta data, the communication model and cascading three step approach, as well as computational trust. The overall conclusion of the new approach for anomaly detection in unsupervised communication meta data by cascading X-means clustering, decision tree, statistical analysis, is given in Section V. The paper ends with an outlook on future work, and points to an additional planned paper.

II. FOCUS OF WORK AND PREREQUISITES

Due to the wide variety of IoT devices and corresponding communications protocols, the focus on application and range of use of the new anomaly detection monitor for IoT devices will be specified in detail. In addition, all prerequisites and restrictions of the concept will be discussed. First of all it should be mentioned, that the new approach focuses on the

 ISBN: 978-1-61208-652-1

monitoring of internet protocol-based (IP-based) IoT devices only. Other network communication protocols will be neglected and are not taken into account.

A second very important prerequisite and restriction of the new approach is the focus on request/response application protocols only. The two most important request/response application protocols are the Hypertext Transfer Protocol (HTTP) and the Constrained Application Protocol (CoAP) [12]. Other application layer protocols, which use, e.g., the Publish/Subscribe principle, will be neglected and are not taken into account for the new approach.

The Constrained Application Protocol (CoAP) [12] is a specialized web transfer protocol, made for communication with constrained nodes and constrained networks in the Internet of Things. The CoAP is mainly designed for machine-to-machine (M2M) [5] applications, such as smart energy, intelligent buildings, home automation, smart grid, and smart factory applications. CoAP was developed as an Internet Standards Document, RFC 7252. CoAP is designed to use minimal resources, both on the device and on the network. Instead of a complex transport stack, CoAP use the User Datagram Protocol (UDP) over IP. Like HTTP, CoAP is based on the wildly successful Representational State Transfer (REST) model: Which means, servers make resources available under a Uniform Resource Locator (URL), and clients access these IoT resources using methods such as GET, PUT, POST, and DELETE. Since HTTP and CoAP share the REST model, both application protocols can easily be connected by using application-agnostic cross-protocol proxies. A Web client may not even notice that it just accessed a IP-based sensor resource. CoAP and HTTP can carry different types of payloads, and can identify which payload type is being used. CoAP and HTTP integrates with Extensible Markup Language (XML), JavaScript Object Notation (JSON), and many other data formats.

Additional assumptions: It will be assumed that the IP-based IoT devices show a normal communication behavior and behave predictably. Furthermore, it will be assumed that no tagged training meta data of the normal behavior of the individual IoT devices are available. In practice, it is very often the case, that no tagged trainings data are available for the particular IoT devices. The most important assumption is the following one: It will be assumed that the capable normal communication behavior of an individual IoT device can be observed and monitored, without compromising by malware, during a certain training time period.

As short overview, the new anomaly detection approach and concept has the following prerequisites and requirements:

- The IoT devices, to be monitored, communicate via the internet protocol (IP) and an IP network.
- The anomaly detection approach shall have a good performance and may require little storage space.
- The anomaly detection system should synchronize recognized connections with signatures.
- A compromise of an IoT device should be detected by anomaly detection.
- Payload of the data packets is ignored.
- The training data are not labeled.
- It is assumed that the monitored IoT devices work without human interaction and their behavior is largely predictable.
- For the monitoring and detection of anomalies, the metadata of the individual connections between end devices is considered only.
- Network traffic should be monitored both online and offline.
- A separate communication model is trained for each new IP-based IoT device type.
- Any anomalies that occur are saved and can be viewed by the system administrator.
- A computational trust value shall be displayed to the managing system administrator for each device, which is to serve as an indicator of the trustworthiness of this particular IP-based IoT device.

III. RELATED WORK

Gaddam et al. [1] developed a two-stage algorithm that uses K-means and Iterative Dichotomiser 3 (ID3) decision tree algorithm. For this, they used labeled training data where each data set is marked as attack or normal traffic. In the first phase, the test data is divided into k clusters. Based on the data vectors assigned to a cluster, a decision tree is then trained via ID3. This avoids two main disadvantages of K-means:

(1) forced assignment: If the value for k is smaller than there are real groups, dissimilar data vectors are assigned to the same cluster.
(2) class dominance occurs when a cluster contains a large part of the data vectors.

In theory, this two-step process should result in the trained tree recognizing more subclasses in the clusters. In order to decide whether a new connection represents an anomaly when applying the model, two rules are applied to the new data vector: First the nearest neighbor (cluster) is found and then the most similar rule of the corresponding ID3 tree is searched. Thereby, especially the decision tree algorithm ID3 needs labeled training data. The authors of this method state that an accuracy of 96.24 percent in detection of attacks was achieved with a network anomaly data test (Network Anomaly Data NAD-1998 dataset) with a false positive rate of 0.03 percent.

Meidan, Y. et. al. [14] presented how supervised machine learning can be applied to analyze network traffic data in order to detect unauthorized IoT devices by manual labeling.

The new cascading three step approach discussed in the following sections would like to overcome the restriction of supervised learning and labeled training data.

IV. CONCEPT AND USED METHODS

Intrusion detection is the monitoring of networks with the aim of detecting security-relevant events. These can be breaches of security rules or malware transmitted over the network. Such rules, also called security policy, are rules laid down by system administrators to which users of a network must adhere. They are designed to prevent users from unknowingly making their devices vulnerable to malware or trying to obtain rights that they are not entitled to.

 ISBN: 978-1-61208-652-1

An Intrusion Detection System (IDS) [3] automates the process of this intrusion detection by permanently monitoring network traffic for communication typical of such actions. An Intrusion Prevention System (IPS) has the same capabilities as an IDS and can still prevent detected events [4].

The main tasks of an IDS are:
- Recording information about discovered events.
- Inform system administrators about events.
- Creating reports.

While IDS focuses on detecting suspicious events, one of the tasks of an IPS is to additionally prevent certain events.

This can be done in various ways:
- Stopping the attack.
- Customize the security configuration.
- Manipulating the attack.

During anomaly-based intrusion detection, all network traffic is synchronized with a communication model that represents normal network traffic. If a significant deviation from the model is found, an intrusion report is triggered. This model can be configured for a network, user or computer. Statistical methods are often used for the comparison with the model. The advantage of Anomaly-Based Intrusion Detection is the ability to detect previously unknown attacks that could not be detected using a signature.

A model representing the normal traffic of a network, user or computer is configured either manually or automatically during a training period. Training times can be static or dynamic. A static model remains unchanged during its useful life and can only be replaced with a new model.

Dynamic models adjust their configuration during runtime. Both methods can cause problems over time, because networks change over time, and so the communication behavior of the network, computer or user can change. Static models therefore generate more false positive over time. Dynamic models do not have this problem, but are more susceptible to slow takeover by an attacker. This could only start with a small number of compromising network requests and increase them over time. A dynamic model would adapt to the behavior without triggering alarms.

A. Unsupervised Machine Learning

Unlike supervised machine learning, unsupervised machine learning does not require labeled training data. In the context of anomaly detection, these algorithms are based on two assumptions: First, that the entire network traffic is normal, and only a small proportion of traffic are attacks. Second, abnormal traffic differs from normal traffic based on statistical data. Widespread Unsupervised Machine Learning algorithms are, e.g., the K-Means, k-Nearest Neighbor (k-NN) algorithm.

Many unsupervised machine learning algorithms are based on clustering. During clustering, data vectors are grouped together based on their similarities. It is used in exploratory data mining, so it is an exploratory measure that examines and clustered the similarity of the data vectors. Depending on the algorithm used, outliers can also be detected. Outliers are individual data vectors that cannot be assigned to an existing cluster. Outliers could be anomalies in our context here.

K-means clustering is one of the most commonly used clustering algorithms. K-means divides all data vectors into k clusters and guarantees that data within a cluster are similar. Here, the center of a cluster is determined, a "centroid", to which the distances of the individual data vectors are calculated. The distance function is the Euclidean distance. This method determines related data vectors. K-Means groups N data vectors into k clusters, where $k < N$ must apply.

The standard k-Means algorithm can only be used with numerical data, but has better performance than other comparable clustering algorithms. However, the parameter k must be set manually. Pelleg and Morre [9] therefore developed the "X-Means" method, in which the k-means algorithm is tested with different k's and the resulting clusters are tested using a density function. The parameter k, which generates the clusters with the highest density, is then recommended for cluster training. With given training data, the optimal value for k is therefore found with this algorithm without having to be configured manually in advance. The X-Means algorithm will be used for the unsupervised machine learning and clustering of related data vectors into centroids.

B. Collection of Meta Data

Network metadata will be collected in the area of network administration for analysis purposes. For this purpose, the entire traffic of a network is observed and metadata about individual connections is recorded. In most cases, therefore, this is done at a central communication point, for example at a gateway.

For the collection of meta data the Bro Network Security Monitor (IDS) [13] will be used. Bro IDS [15] is an open source network monitoring framework based on Unix, which has been extended by some IDS-typical functions. Bro IDS processes observed network traffic in two steps:

(1) Event Engine: The Event Engine analyzes the observed network traffic live and detects events. The trigger for these events is defined in an associated enhancement. It is possible to collect further information about this event using these enhancements. Bro comes preinstalled with some extensions, such as signature and connection logs, but also application protocol events, like , e.g., Domain Name Service (DNS) and Dynamic Host Configuration Protocol (DHCP).

(2) Policy Script Engine: Uses events triggered by the Event Engine to perform custom actions. Here further analyses of the generated data can be carried out and dependent behavior can be implemented.

The Event Engine will be used to create connection logs. This information contains metadata for each detected connection. They consist of a mixture of numerical and categorical data, such as the number of transmitted packets, the contacted port, the IP addresses involved and much more.

 ISBN: 978-1-61208-652-1

In Bro IDS it is also possible to define signatures. If a recognized connection corresponds to the behavior of a defined signature, a corresponding event is triggered and a signature log is created.

Bro IDS is able to monitor a network live at a central communication point, for example as a gateway. However, there is also the option to analyze recorded network traffic. This would fulfill the requirement to be able to operate both online and offline.

C. Communication Model and Cascading 3 Step Approach

A communication model is the basis on which an ADS or IDS analyzes observed network traffic and detects anomalies. Based on the above requirements, a categorizing communication model is designed for the new anomaly detection approach. This means that all recognized connection metadata will be divided into categories. During the training phase, these categories are learned automatically using the procedure described below. If the assignment of new connection metadata to a category is not possible after the end of the training, during the application phase, the communication behavior is not known and therefore represents an anomaly. The new anomaly detection approach in unsupervised communication meta data of IP-based IoT devices is characterized by cascading X-means clustering, decision tree, and statistical analysis. The cascading three step approach works like this:

Step (1): X-means clustering (see Figure 1): In the first step, unsupervised training meta data will be collected and categorized based on the numerical data it contains about clustering. Therefore selected numerical data t_n from the connection metadata are used to categorize the training data into k clusters using k-means clustering. For the k-means algorithm, the parameter k, which determines the number of clusters to be formed, must be set. However, since the number of clusters is not known in advance, k must first be found. X-means clustering is used for this. Additionally, the system administrators should be given the option to set the k parameter manually.

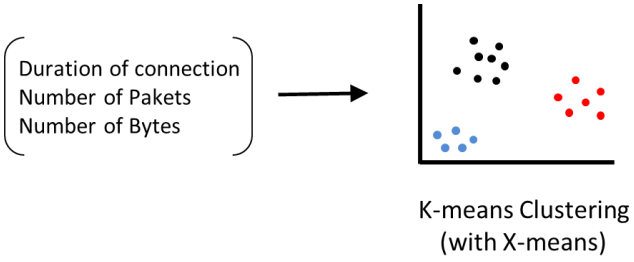

Figure 1. X-means Clustering

After the clustering of the training data has been completed, only the centroids of the clusters are stored persistently. For the later classification of metadata on during the application phase, only the k centroids are required to determine the affiliation of a data vector to a cluster. Since a trained model should not change, moving the centroids as in the k-means algorithm is no longer necessary. The t_n data vectors are therefore not needed for future use and can be deleted.

Step (2): Generation of decision trees (see Figure 2): For all data within a cluster, categorical values of the connection metadata will be stored in a decision tree structure to divide the observed connections into further subcategories. As a result, the recognized categories are further subdivided by generating trees from the categorical data. Proven in practice and used in the approach are the following decision tree structure (from root to leaf): Transmission Protocol => Service => Port Number => IP Address.

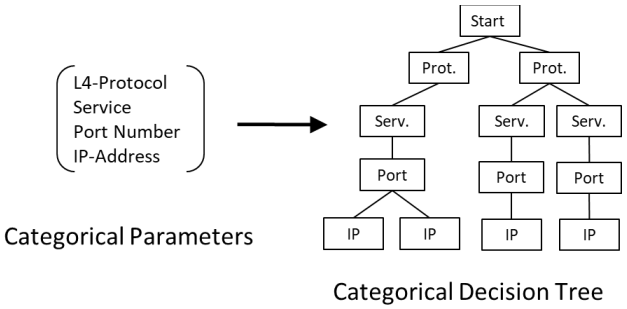

Figure 2. Generation of Decision Trees by Categorical Parameters

Step (3): Statistical evaluation (see Figure 3): The allocation of the connection metadata, collected during the training time to the individual categories, will be calculated proportionately in order to enable a statistical analysis later. As result of the statistical analysis, the distribution of the individual categories of the clusters and tree paths are calculated proportionately.

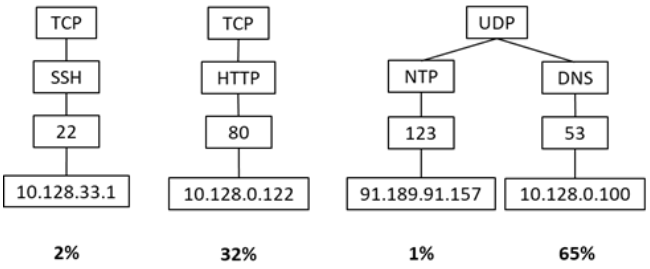

Figure 3. Statistical Distribution of the individual subcategories of the generated clusters and tree paths (an arbitrary example)

D. Training data, parameters, and analysis

The whole of all training data for a particular machine is referred to in this section as T_i, where i represents a particular machine. T_i consists of a series of data tuples t, each of which represents a connect log and contains all metadata of a connection. The objects in t consist of different data types: numeric (numbers) and categorical (characters and strings).

Not all metadata provided by logging frameworks are relevant for training communication models. Therefore, a series of data are selected from t, which are used for model creation. The numerical data is referred to as t_n and the categorical data as t_k. When the course is finished, the system

switches to the application phase. New metadata is recognized online or offline and applied to the communication model. Such a Conn log to be checked has the same design as $t \in T_i$ and is called a $\in A_i$. A_i is the set of recognized connection metadata during the application phase of a device identified by i. The communication model trained with the training data T_i is used to check the tuples $a \in A_i$. An subset of a contains the same selected numerical data as t_n, the same applies to a_k subset of a and t_k.

In order to guarantee the accuracy of the model, different methods suitable for the properties of the data types are used for learning the subcategories.

Numerical parameter: Numeric values are used to rank the scope of a connection. For this purpose, five elements from t are used, which are summarized in a new tuple t_n:
1. Duration: duration of the connection. 2. OrigPkts: Number of sent packets of the connection initiator. 3. RespPkts: Number of packets received by the connection initiator. 4. OrigBytes: number of bytes sent. 5. RespBytes: number of received bytes.

Categorical parameter: In contrast to numerical parameters, categorical parameters can only take a defined number of possible values. In this approach, the following four objects are transferred from t to t_k:
1. Transmission protocol: The used Layer 4 protocol (UDP or TCP). 2. Used Service: If detected, the application protocol used (like , e.g., DNS or DHCP) 3. Target port: The contacted target port. 4. Internet protocol destination address: The contacted IP address.

Since it can be assumed that IoT devices behave predictably, the observed categorical values can be regarded as complete in all t_k with sufficiently long training time. This means that all possible combinations of categorical values occur at least once during the training phase. Using the exploratory measure in the previous step, k categories for link metadata were found. Each cluster contains at least one $t \in T_{ix}$, where T_{ix} represents a certain subcategory. Whose categorical values can be learned further in order to find further subcategories within a cluster. These categorical values must now be learned in such a way that it can be efficiently checked whether there is a matching t_k for a given a_k where $a_k = t_k$. Tree structures are suitable for this. All t_k are grouped in a tree $b_j \in B_i$. B_i contains all the trees of a communication model, where i represents a specific device. Where j stands for the cluster for which this tree is trained. Each element of the tuple t_k corresponds to a node in tree b_j. The depth of the tree corresponds to $| t_k |$. Please note that the tree levels are selected in such a way that an optimal tree structure is generated.

For the categorical values selected above, this order is (from root to leaf): Transmission Protocol (TCP or UDP) => Service => Port Number => IP Address. The reason for this is that successive values must always be in a 1:n relationship.

A transmission protocol such as TCP or UDP can be used by several services, one service can use several ports and several IP addresses can be addressed via the same port. Saving the different combinations of categorical values in trees removes redundant elements and simplifies matching with new data sets. Trees are generated for each communication model. Each leaf of the trees represents its own communication category.

E. Computational Trust

The anomaly detection generates positive and negative experiences. These are summarized in a trust model and used to calculate a computational trust value for every individual IoT device in the network. This value allows system administrators to evaluate the behavior of an IoT device at a glance.

The concept of trust in a social context is well known from everyday life. Trust allows people to delegate tasks and assess whether information should be shared with another individual. Trust also enables us to evaluate information shared with us. Computational Trust raises the social construct of trust in the field of computer science to build trust or distrust between agents (devices or people). A practicable definition of trust has been specified by the sociologist Diego Gambetta [2]: “Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.” So trust is treated as something subjective, which is always associated with some form of prediction and expectation of behavior. Another important element of trust is reputation.

During the application phase, the communication model generates positive and negative experiences based on the observed connection metadata and the results of anomaly detection. If it is possible to assign the received Conn log (a) to a category, a positive experience is saved. If this is not possible or if a distribution anomaly is triggered, a negative experience is stored. These experiences are persistently stored and used by a computational trust model, to calculate a trust value for each device.

This value is the subjective probability value E, which indicates the probability with which future experiences with a device can be rated positive. The higher this value, the more trustworthy is an IoT device. This allows system administrators to control the behavior of an IoT device by checking a simple numerical value $x \in [0, \ldots 1]$ to evaluate. If the confidence value of a device decreases unexpectedly quickly, it can be assumed that the device will be compromised because the device triggers a high number of anomalies.

However, not all triggered anomalies actually indicate a compromise of the IP-based IoT device. The particularly rigid communication model can lead to an increase in false positives. However, by calculating trusted information based on experience, these individual false positives do not have a

 ISBN: 978-1-61208-652-1

strong impact on trustworthiness, which does not unnecessarily upset the system administrator. Only with a compromise of the IoT device can be expected with an increased number of anomalies, whereby the trust value decreases permanently. Computational Trust is therefore used to find and eliminate the false positive anomalies triggered by anomaly detection.

As trust model the proof-of-concept implementation will use Certain Trust, developed by Sebastian Ries [10], which allows to represent trust for ubiquitous computing and P2P systems in a way, which can be interpreted and updated by software agents as well as by users. A key feature of Certain Trust is that it is capable of expressing the certainty of a trust opinion, depending on the context of use.

V. CONCLUSION AND OUTLOOK

This paper has presented and discussed a new cascading three step approach for anomaly detection in unsupervised communication meta data of IP-based Internet of Things devices. The new approach cascades X-means clustering, decision tree, and statistical analysis (see Figure 4 below), in order to monitor and protect IoT networks. The new approach is restricted to IP-based IoT devices, and request/response application protocols. The cascading three step approach is designed for anomaly detection in unsupervised CoAP and HTTP communication meta data. Additionally a trust model has discussed in order to allow system administrators to control the behavior of an IoT device by simply checking the trust value of this particular IoT device.

It is planned to write an additional research paper on the real proof-of-concept implementation of the presented three step anomaly detection approach for the next International Conference on Cyber-Technologies and Cyber-Systems. Additionally, a detailed evaluation of CoAP and HTTP communication meta data of IP-based IoT devices, including a verification of the false positive rate, shall be done.

This work was supported by the Center for Research in Security and Privacy (CRISP), Darmstadt, Germany.

REFERENCES

[1] Shekhar R. Gaddam; Vir V. Phoha; and Kiran S. Balagani, "K-Means+ ID3: A novel method for supervised anomaly detection by cascading K-Means clustering and ID3 decision tree learning methods," In IEEE Transactions on Knowledge and Data Engineering, vol. 19 (2007), no. 3, pp. 345–354, 2007. http://dx.doi.org/10.1109/TKDE.2007.44. – DOI 10.1109/ TKDE. 2007.44

[2] Kapitel Can we Trust Trust? In: Diego Gambetta, "Trust: Making and Breaking Cooperative Relations," electronic edition, pp. 213–237, 2000.

[3] (CSD), NIST Computer Security D.: NIST SP 800-94, Guide to Intrusion Detection and Prevention Systems (IDPS), 2007.

[4] Ali A. Ghorbani, Wei Lu, and Mahbod Tavallaee, Chapter 4 "Theoretical Foundation of Detection" in "Network Intrusion Detection and Prevention," Advances in Information Security vol. 47, Springer Science, 2010.

[5] Christian Karasiewicz, "Why HTTP is not enough for the Internet of Things," https://www.ibm.com/developerworks/community/blogs/mobileblog/entry/why_http_is_not_enough_for_the_internet_of_things?lang=en. September 2013.

[6] Knud L. Lueth, "Why the Internet of Things is called Internet of Things: Definition, history, disambiguation", https://iot-analytics.com/internet-of-things-definition/. Dezember 2014

[7] Luigi Atzori, Antonio Iera, and Giacomo Morabi "The Internet of Things: A survey," In Computer Networks, Volume 54, Issue 15, pp. 2787-2805, 28 October 2010, http://dx.doi.org/10.1016/j.comnet.2010.05.010. – DOI 10.1016 j.comnet.2010.05.010

[8] Monowar H. Bhuyan ; D. K. Bhattacharyya ; and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," In IEEE Communications Surveys & Tutorials, Volume: 16, Issue: 1, pp. 303 - 336, 2014.

[9] Dau Pelleg, and Andrew Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," In Proceedings of the 17th International Conf. on Machine Learning, Morgan Kaufmann, pp. 727–734, 2000.

[10] Sebastian Ries, "Certain Trust: a trust model for users and agents," In Proceedings of the ACM symposium on Applied computing (SAC 2007), pp. 1599-1604, 2007. DOI=http://dx.doi.org/10.1145/1244002.1244342

[11] Salima Omar, Asri Ngadi, and Hamid H. Jebur, "Machine Learning Techniques for Anomaly Detection: An Overview," In: International Journal of Computer Applications (0975 – 8887), Volume 79, No. 2, October 2013.

[12] Zach Shelby; Klaus Hartke; and Carsten Bormann, "The constrained application protocol (CoAP)," https://tools.ietf.org/html/rfc7252. Version: 2014

[13] Robin Sommer, "Bro: An Open Source Network Intrusion Detection System," In: DFN Arbeitstagung über Kommunikationsnetze, pp. 273-288, 2003.

[14] Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N.O., Guarnizo, J.D., and Elovici, Y. „Detection of Unauthorized IoT Devices Using Machine Learning Techniques", 2017, https://arxiv.org/pdf/1709.04647.pdf

[15] The Bro Network Security Monitor - https://www.bro.org

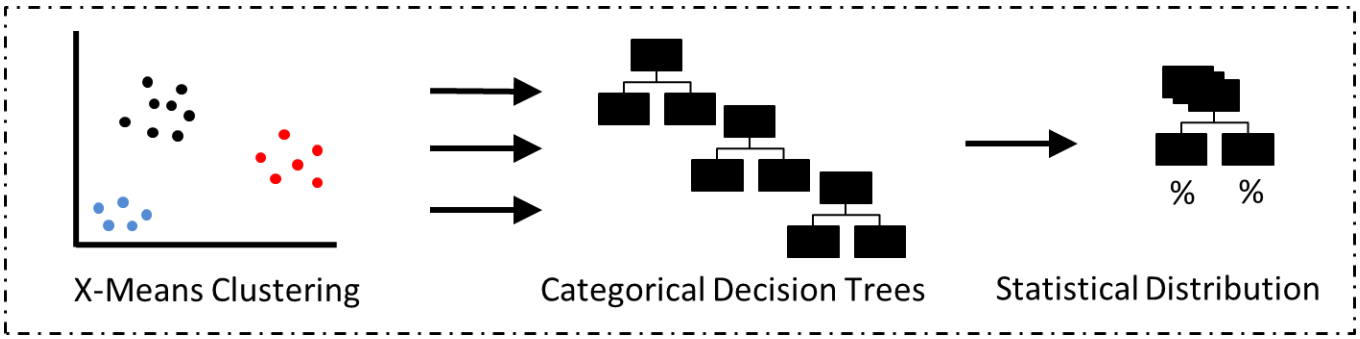

Figure 4. Cascading Three Step Approach for Anomaly Detection in Unsupervised Communication Meta Data of IP-based Internet of Things Devices

 ISBN: 978-1-61208-652-1