Attack Detection in Smart Home IoT Networks using CluStream and Page-Hinkley Test

Fernando H. Y. Nakagawa
Dept. of Computer Science
State University of Londrina
Londrina, Brazil
fernando.nakagawa@uel.br

Sylvio Barbon Junior
Dept. of Computer Science
State University of Londrina
Londrina, Brazil
barbon@uel.br

Bruno Bogaz Zarpelão
Dept. of Computer Science
State University of Londrina
Londrina, Brazil
brunozarpelao@uel.br

***Abstract*—The expansion of IoT device networks increases the demand for security systems that detect attacks against these new targets. These devices have simple hardware, limited memory and processing power, and often are required to have low energy consumption. Batch supervised learning algorithms have been employed to address this issue, but they present some limitations. These algorithms demand benign and malicious labeled samples to be trained, which can be hard to obtain in real networks. Also, once they are trained, it is hard to update the learning model with newly found behaviors. In this work, we propose an online and unsupervised scheme to detect attacks in smart home IoT networks. This scheme is based on the combination of two algorithms: CluStream and Page-Hinkley Test. It does not require labeled samples to be trained and learns incrementally as it is used. Tests were performed over data obtained from publicly available datasets consisting of multiple smart home devices and the results are encouraging. Different types of attacks were detected with an overall detection rate around 97%, while the precision stayed above 87%.**

***Index Terms*—Internet of Things, stream learning, unsupervised algorithm, attack detection**

I. Introduction

The increasing number and importance of IoT devices in home environments ease people's everyday lives, e.g., by making daily tasks more efficient, increasing home security, and enabling health monitoring. However, while these devices bring seamless experiences of ubiquitous connectivity, they may lead to privacy and security problems. Most people trust these home appliances, but they do not know what information these devices transmit and store. Also, as these devices have high connection availability and low security, they become targets for attackers. Malicious users may use these opportunities to create botnets, spread malware, and break into people's privacy. This current scenario creates a demand for security systems that are able to detect attacks against these new targets [1]–[4].

IoT systems have simple hardware, including limited memory and processing power. Some security technologies are still challenging to be deployed in these devices, such as cryptography [2] or intrusion detection systems (IDS). Several approaches proposed to detect attacks in IoT systems are based on batch machine learning algorithms. Also, they usually consist of supervised learning schemes, which require labeled samples to be trained and create a learning model [1], [5].

Two main issues arise when it comes to batch supervised algorithms in this context. Firstly, batch algorithms cannot learn incrementally. These algorithms approach their problems as they were static: a finite amount of training data is gathered, a static model is trained, and then it is applied to classify future observations. When the data source behavior changes, updating the learned model may be challenging. Secondly, labeled samples are hard to obtain in real scenarios, especially malicious ones. Even if the user had these labeled samples, it would be necessary to update the model every time the network behavior changes. Most domestic users are not technology experts to manage their network security [2], [3]. Therefore, it would be difficult for them to use an attack detection system based on a batch supervised learning algorithm.

An alternative to batch learning algorithms is to use stream learning. Stream learning algorithms consider that input stream elements arrive online, and these data streams are potentially unbound in size. For this reason, stream learning algorithms do not store the data elements in memory, only statistical information about them. Also, they can learn incrementally, updating their models as new data arrives. Like batch learning algorithms, stream learning schemes can be supervised or unsupervised. By using an unsupervised process, they do not require labeled samples to be trained [6], [7].

In this study, we propose an online and unsupervised scheme to detect attacks in smart home IoT networks. The proposed approach starts by dividing the incoming packets into three streams: ICMP, UDP, and TCP streams. Each of these streams is input to a CluStream instance. This algorithm partitions the incoming packets online, creating, updating, and destroying micro-clusters as new packets arrive. Then, the scheme monitors the distance among the micro-clusters to detect new and sudden changes that may indicate attacks. The continuous collection of this indicator feeds a Page-Hinkley Test, which detects sudden changes automatically. Experiments were carried out over data collected from a publicly available dataset. The results show that the proposed approach can detect different types of attacks and has a low false-positive rate in most of the observed situations.

This paper is organized as follows: Section II presents the related work, while Section III introduces the CluStream framework and Page-Hinkley Test. In Section IV, we explain our proposed approach. Section V presents the experimental results, illustrated by two examples. Lastly, Section VI presents the conclusion about this study.

II. Related Work

Different studies have been carried out in recent years to tackle attacks against IoT home networks. Supervised machine learning algorithms are frequently found as their central component. Anthi et al. [1] proposed an intrusion detection system for smart home IoT devices that makes use of supervised learning in three different phases. First, supervised algorithms classify the traffic collected in the home router according to the source/destination IoT device. Then, traffic packets belonging to each device are classified as malicious or benign by another supervised model. Lastly, a third supervised model is employed to determine the attack type if a packet is classified as malicious. The proposal was evaluated in a testbed with different types of attacks such as scanning, denial of service (DoS), and man in the middle (MITM).

Moustafa et al. [5] also employed supervised algorithms, but, in their case, the proposal relies on ensemble learning to reach better results. The proposed approach makes use of Adaboost to combine three base classifiers: decision tree, naive Bayes, and artificial neural network. The approach analyzes the collected traffic to detect botnet attacks against HTTP, DNS, and MQTT protocols. Instead of exploring ensemble learning techniques, Brun et al. [8] apply deep learning, which has been widely discussed in the intrusion detection area. A model based on dense random neural networks was developed to detect DoS attacks through statistics collected from network traffic in their work.

Detecting unknown attacks can be challenging for supervised algorithms. To address this shortcoming, Wan et al. [9] proposed combining supervised and one-class classification algorithms. Instead of requiring malicious and benign observations for training, a one-class algorithm only needs samples of benign behavior. Thus, it detects attacks by spotting observations that are not similar to the benign ones used for training. In the approach by Wan et al., the network traffic analysis is divided into two steps. Firstly, a supervised algorithm is used to detect known attacks. Then, if the observation is classified as benign, it is input to a one-class algorithm for further analysis.

Meidan et al. [10] used only a one-class classification algorithm in their detection approach. More specifically, they applied deep auto-encoders to detect botnets in smart home network traffic. Bezerra et al. [11] also used exclusively one-class classification algorithms to detect botnets. Unlike Meidan et al., they proposed a host-based solution, which monitors the usage of resources like CPU and memory in each device. Multiple algorithms such as isolation forest, one-class SVM (Support Vector Machine), and LOF (Local Outlier Factor) were tested. Lastly, in [12], the authors proposed to use a one-class classification algorithm coupled with a reinforcement learning solution. They aimed to address a typical challenge in attack detection: devices' normal behavior can change, and the models must be updated to reflect these movements.

Overall, the reviewed solutions are based on batch learning algorithms, which means they are not designed to be frequently updated. Also, they assume that training data, no matter its volume, is integrally stored to build or update the learning model. This can be a burden in smart home environments, since they may not be designed to store and process all this data. Another challenge is the demand for samples for extensive training phases, which may be hard to meet in practical scenarios. In this work, we propose a detection approach based on a stream clustering algorithm, which does not require storing observations for training and can learn incrementally, being continuously updated.

III. Fundamental Background

A. CluStream

CluStream is a continuous data stream clustering framework proposed by [6]. The CluStream online process does not store the data instances but maintains a statistical summary in a micro-cluster structure. Let's consider a data stream consisting of a set of multi-dimensional data instances $\overline{X}_1, ..., \overline{X}_k$ arriving at time stamps $T_1, ..., T_k$. Each $\overline{X}_i$ contains d dimensions denoted by $\overline{X}_i = (x_i^1 ... x_i^d)$. The structure of a micro-cluster is defined as the tuple: $(\overline{CF2^x}, \overline{CF1^x}, CF2^t, CF1^t, n)$, where:

- $\overline{CF2^x}$: sum of the squares of the data values for each dimension;
- $\overline{CF1^x}$: sum of the data values for each dimension;
- $CF2^t$: sum of the squares of *time stamps*;
- $CF1^t$: sum of the *time stamps*;
- n: number of data points in the micro-cluster;

To start the online phase, it is necessary to determine the initial clusters. It can be achieved through a clustering method, such as K-Means. Then a maximum of q micro-clusters are maintained in the memory, assigning a unique identification number to each one. When a new data instance arrives, the algorithm checks whether it is within the *maximum boundary* of a micro-cluster. If so, it is added to the micro-cluster, updating the micro-cluster's tuple through the additive property. Otherwise, a new micro-cluster with only one data instance must be created. To respect the limit of q micro-clusters, an old one must be eliminated or two must be merged when the creation of a new cluster makes this limit be violated. To choose between these options, firstly it is verified if any micro-cluster has its *relevance stamp* below the value set in the hyperparameter δ_c. If true, it can be deleted. The relevance stamp stands for the age of the micro-cluster based on time properties. Finally, if no micro-cluster's relevance stamps are below δ_c, the two nearest micro-clusters are merged.

B. Page-Hinkley Test

The Page-Hinkley Test [13] detects changes based on the mean of an univariate stream. Upon a test, it considers the cumulative sum of the observed values and their mean up to the current moment. At first, the algorithm needs four

hyperparameters: $minimum\_instances$, δ_{ph}, $threshold$, and α. For each iteration i, the Page-Hinkley algorithm receives the x_i value from the stream. Then a mean M_i is calculated by:

$$M_i = M_i + \frac{(x_i - M_i)}{i} \tag{1}$$

Afterward, the cumulative sum of positive changes σ_i^+ is calculated:

$$\sigma_i^+ = max(0, \alpha \cdot \sigma_i^+ + (x_i - M_i - \delta_{\text{ph}})) \tag{2}$$

To detect changes in two-sized mode, the cumulative sum of descent values σ_i^- is given by:

$$\sigma_i^- = max(0, \alpha \cdot \sigma_i^- + (M_i - x_i - \delta_{\text{ph}})) \tag{3}$$

If a minimum of data instances ($i \geq minimum\_instances$) were analyzed and the conditions ($\sigma_i^+ > threshold$) or ($\sigma_i^- > threshold$) hold true, an alert is raised. Also, σ_i^+ or σ_i^- are reset.

IV. Proposed Approach

In this section, we present our approach to detect attacks in smart home IoT networks. This approach is based on the hypothesis that attacks against an IoT device will generate a detectable disturbance in the observed traffic. To detect anomalous behaviors, we employ online algorithms, which can learn incrementally and keep a low memory footprint. An overview of the approach is presented in Figure 1. Each packet coming into the device is classified according to its protocol (TCP, UDP, or ICMP). Then, the CluStream unsupervised algorithm assigns the incoming packet to a micro-cluster, and the mean of the maximum distances among the clusters' centroids is calculated. Having this result, the Page-Hinkley Test is applied to check whether there is an anomalous change in the micro-clusters' movement. On detection, the network administrator is alerted that there might be anomalous traffic occurring.

The first step consists of breaking the incoming stream into three different streams. Each packet is classified according to the protocol on top of the IP header in the protocol stack. For this study, we considered TCP, UDP, and ICMP protocols. The CluStream algorithm will be applied to each packet individually. As TCP, UDP, and ICMP headers have different fields, we decided to separate these packets into three different streams and dedicate an exclusive CluStream instance for each of them. Before inputting these packets to the CluStream instances, it is necessary to extract the features that will be analyzed, which are presented in Table I and Table II. The features *len, ip.flags.df, ip.flags.mf, ip.ttl, ip.frag_offset* are extracted in the three streams, once they are related to the IP header and the whole packet length. The features *tcp.dstport, tcp.flags.syn, tcp.flags.ack, tcp.flags.push* are exclusive to TCP. Also, it is considered only the *udp.dstport* for UDP and *icmp.code* for ICMP. All the features are normalized using min-max normalization. To this purpose, we consider the

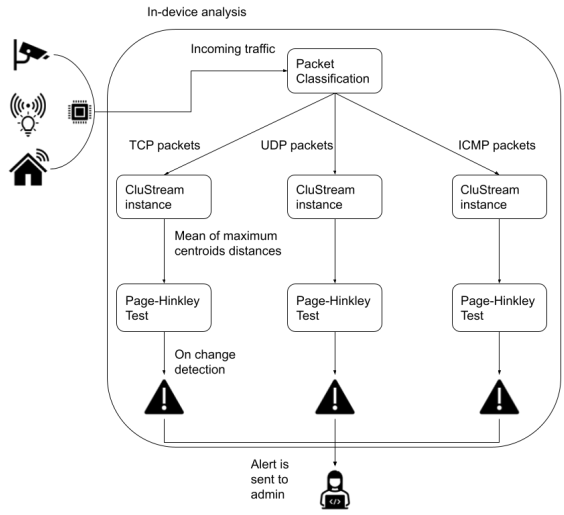

Fig. 1. Proposed Model to detect attacks in Smart Home IoT Networks.

protocol specification limits for each header field. For example, let's consider the feature *tcp.dstport*: its minimum value $x_{\min}$ is 0 and its maximum value $x_{\max}$ is 65535. To normalize this feature into the $[0, 1]$ interval, given a feature value x , the normalized value x_{norm} is calculated as in (4).

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{4}$$

TABLE I
Common features used in TCP, UDP and ICMP streams.

Common features	Description
len	Length of the datagram in bytes
ip.flags.df	Indicates if the datagram can be fragmented.
ip.flags.mf	Indicates if the datagram is the last fragment or there are more fragments.
ip.ttl	Indicates the datagram's time to live. Once it reaches 0, it must be destroyed.
ip.frag_offset	Position of the fragment in the entire datagram.

To run a CluStream instance, the hyperparameters q, $InitNumber$, and h need to be set, where:

- q is the maximum number of micro-clusters;
- $InitNumber$ is the number of initial data instances used to create the initial micro-clusters;
- h is a time horizon considered to calculate the mean of the maximum centroids' distances.

The other hyperparameters are left with their default values ($t = 2, r = 2, m = 10, \delta_c = 10$), where:

- t: cluster's maximum boundary factor, which is a factor of the root mean square (RMS) deviation of the data points in the cluster from the centroid;

TABLE II
EXCLUSIVE FEATURES USED IN TCP, UDP AND ICMP STREAMS.

TCP features	Description
tcp.dstport	Destination port of TCP datagram
tcp.flags.syn	Synchronization flag to start the synchronization of the sequence numbers.
tcp.flags.ack	Acknowledgement flag indicating that a connection is established.
tcp.flags.push	Push flag asking that data should be pushed to the application.
UDP features	**Description**
udp.dstport	Destination port of UDP datagram
ICMP features	**Description**
icmp.code	Control message code of ICMP.

- r: heuristic maximum boundary factor. For a micro-cluster with only 1 point, the maximum boundary is r times the maximum boundary of the next closest cluster;
- m: cluster's last data points. They are used to approximate the average time-stamp;
- δ_c: a threshold to eliminate a cluster. If the least relevance stamp of a cluster is below δ_c, the cluster can be erased, and a new cluster can be created.

To start using the CluStream instance, a set of initial data instances, as determined by $InitNumber$, is input to the K-Means algorithm to create the initial q clusters. After that, as new data instances arrive, the CluStream algorithm will maintain the micro-clusters adding each data instance to a cluster within the maximum boundary factor, eliminating, creating, or merging clusters.

Every time the CluStream instance clusters a new packet, the mean of the maximum distances among the cluster centroids is calculated as follows. Only the q' micro-clusters with new data instances assigned to them in the time horizon h are considered. First, the Euclidean distance for every pair of these micro-clusters centroids is calculated. The centroid of a micro-cluster i is calculated by $\overline{CF1}^x{}_i/n_i$. The calculated distances are stored in a matrix D with dimensions $q' \times q'$, where each position $d_{i,j}$ denotes the distance between the centroids of the micro-clusters i and j. For each row i in D, we pick the maximum distance, denoted as $mdist_i$. It represents the maximum distance of the centroid of the micro-cluster i to any other micro-cluster's centroid in D. Finally, we calculate the mean of all the maximum distances $\overline{mdist}$ as in (5). The computed mean will feed the respective instance of the Page Hinkley test.

$$\overline{mdist} = \frac{\sum_{i=1}^{q'} mdist_i}{q'} \tag{5}$$

Our hypothesis is that when a device is attacked, the involved packets will be clearly distinct from the other ones. When CluStream clusters a malicious packet, either a new micro-cluster is created, or this packet is assigned to an existing cluster. In the former possibility, the new micro-cluster will likely be distant from the other micro-clusters since it represents a new and anomalous behavior. In the latter, the micro-cluster that received the malicious packet will have its centroid significantly changed. In both cases, the mean of the maximum distances between the micro-cluster centroids will be affected. By monitoring this indicator, we can detect when it presents a sudden and significant change.

The calculated means of the maximum distances make up a series, which feeds an instance of the Page-Hinkley Test. When it detects a change, the system administrator is alerted. The hyperparameters $minimum\_instances$, δ_{ph}, $threshold$, and α for the Page Hinkley Test must be set and are crucial to determine if an alert will be raised or not.

V. EVALUATION

In this section, the proposed approach is evaluated. We start by presenting the experimental design containing information about the dataset, implementation and evaluation metrics. Next, we show the results related to the proposal's efficacy. Finally, two examples are discussed to provide more details about the approach's operation.

A. Experimental Design

To evaluate the proposed approach, we used the dataset created by [1]. This dataset consists of packets captured in multiple smart home devices such as smart plugs, cameras, and device hubs. Different attacks such as scanning, DoS, and MITM were launched against these devices, with the packets being labeled accordingly. To use the dataset in our evaluation, we first organized it by device and protocol (TCP, UDP, and ICMP). Then, we observed that the tests that originated the dataset followed a common pattern. For each device, the traffic starts with a long sequence of normal traffic, followed by a sequence of attacks. There are no situations that intersperse periods of normal traffic and attacks. As our proposed approach is based on incremental learning, it is important to observe its behavior when the traffic changes from normal to attack and vice-versa.

To address this issue, we augmented the original dataset. By following the dataset labels, the moments when attacks started and ended were identified. Also, we found the moments when the monitoring was turned off or restarted. This was possible through the analysis of the packet's timestamps. In some points, there are gaps between packets that clearly indicate a pause in the monitoring, which is expected in experimental environments. Using these moments as references, we extracted self-contained periods of normal and attack traffic, making a collection of them. Then, we could make new scenarios for each device, combining periods picked from this collection. As a result, we made 11 scenarios, where each scenario is a stream characterized by device + protocol, as Table III presents. This table also provides details about the amount of packets (length) and the normal:attack ratio for each scenario.

The data presented in this study are available on request by contacting the authors.

TABLE III
DESCRIPTION OF THE EXPERIMENTAL SCENARIOS.

Device name	Protocol	Attack Type	Length	Unbalance Ratio
Hive Hub	TCP	DoS, MITM, Scanning	193919	4.6:1
	UDP	MITM, Scanning	4464	10.2:1
	ICMP	DoS, MITM	36248	51:1
Samsung Smart Things	TCP	DoS, MITM, Scanning	122833	3.5:1
	UDP	MITM, Scanning	10612	30.2:1
Lifx Smart Lamp	TCP	DoS, MITM	64199	5.4:1
	ICMP	MITM	89073	164:1
TP-Link NC200 Camera	TCP	DoS, MITM, Scanning	39848	2:1
	UDP	DoS, MITM, Scanning	8549	3.4:1
TP-Link SmartPlug	TCP	DoS	8896	12.6:1
	UDP	DoS, IoT-Toolkit	12477	0.34:1

After data preparation, we implemented the proposed approach in Python. To implement the CluStream module, the code from a GitHub repository[1] was used as starting point. Some changes were added to this code, including the calculus of the mean of maximum centroid distances. As for the Page-Hinkley Test, we made use of the River's Page-Hinkley implementation [14], with modifications to detect changes in two-sided mode.

Finally, the metrics used to evaluate the proposed approach are:

- **Detection Rate**: proportion of how many attacks were detected. If an attack goes unnoticed by the system, this value gets lower;
- **Precision**: $TP/(TP+FP)$. The percentage of alerts that were true. TP stands for true positives and FP stands for false positives.

B. Results

To run our approach over the 11 scenarios, we set *Init-Number* to 10%, representing the data used by K-Means to generate the initial q micro-clusters. The value of q was set to 12. As for the Page-Hinkley Test hyperparameters, firstly, we used the default values: $minimum\_instances = 30, \delta_{\text{ph}} = 0.005, threshold = 50, \alpha = 0.9999$. Then different values for $threshold$ and α were analysed to find the best setting in terms of precision and detection rate. The thresholds used were: 0.001, 0.5, 1, 5, 10, 20, 50, 100, 150,200 and the α values were: 1, 0.9999, 0.999, 0.99, 0.9, 0.8, 0.75, 0.5. In Table IV, we present the best results for each scenario with its hyperparameters values, detection rate, and precision.

TABLE IV
TOP RESULTS FOR EACH DEVICE AND PROTOCOL.

Device name	Protocol	Threshold	α value	Detection rate	Precision
Hive Hub	TCP	150	0.999	100%	86,6%
	UDP	5	0.99	100%	75%
	ICMP	1	0.8	100%	100%
Samsung S. Things	TCP	0.5	0.5	100%	100%
	UDP	5	0.999	100%	28.5%
Lifx S. Lamp	TCP	150	1	100%	100%
	ICMP	0.5	1	100%	100%
TP-Link Camera	TCP	5	0.9	100%	92.3%
	UDP	0.5	0.9	66.7%	75%
TP-Link S. Plug	TCP	150	0.999	100%	100%
	UDP	0.5	0.8	100%	100%

After analyzing a total of 591.118 packets in 11 scenarios, the overall detection rate is around 97%, and the precision is approximately 87%. In most cases, the proposed approach yield better results to TCP and ICMP streams than to UDP ones. UDP streams presented a very irregular behavior even in normal traffic periods, causing frequent changes in the mean of the maximum distances. Conversely, when considering TCP and ICMP streams, five out of seven scenarios reached 100% of detection rate and precision. For the other two scenarios related to TCP and ICMP, the detection rate was 100%, and the precision stayed above 85%. It is also noteworthy that our approach was able to detect all the different attack types: DoS, scanning, MITM, and IoT-Toolkit. The results show that CluStream and the mean of maximum distances among centroids provide a good indicator of attacks in the TCP and ICMP streams. Moreover, Page-Hinkley Test could detect these changes precisely.

C. Illustration Examples

To better illustrate how the proposed approach works, figures 2 and 3 present more details on how the attacks were detected in two scenarios. More specifically, the objective is to show through these examples how the mean of maximum distances reacts to different attacks and how the proposed approach uses this indicator to detect these attacks.

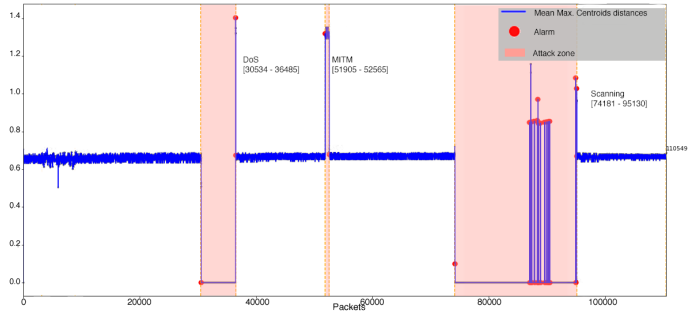

Fig. 2. TCP stream in the Samsung Smart Things Hub: mean of maximum centroid distances and generated alerts.

[1]https://github.com/FelixNeutatz/CluStream/blob/master/python

The scenario with TCP stream in the Samsung Smart Things Hub TCP is presented in Figure 2. Until packet #30533, the traffic is normal, and the mean of the maximum distances present small and regular variations. A DoS attack starts with packet #30534. Suddenly, the mean of maximum distances drops, signaling that there is a predominant behavior in this stream now, which makes the micro-clusters stay very close to each other as they are updated or created with DoS packets. As a consequence of the change, an alert is quickly raised. After packet #36485, the traffic goes back to normal, so the centroids move back to the previous pattern, thus generating two alarms that are not considered false positives since they are related to the attack. Next, a MITM attack occurs between packets #51905 and #52565, being properly detected by the approach. Finally, from packet #74180 to #95130, a scanning attack occurs, firing various alerts. It is noticeable that the varying nature of scanning attacks generated an irregular pattern in the mean of maximum centroid distances.

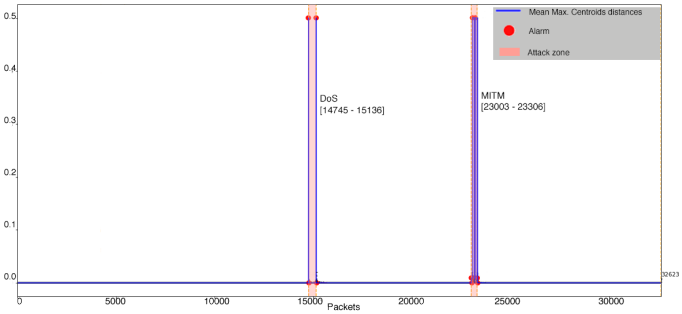

Fig. 3. ICMP stream in the Hive Hub: mean of maximum centroid distances and generated alerts.

Figure 3 depicts the scenario with ICMP stream in the Hive Hub. From the beginning to packet #14744, the traffic is normal, and we can observe that the mean of maximum distances is stable. When the DoS attack starts, there is a very high peak in the mean of maximum distances, signaling that a new micro-cluster was added or an existing one was updated and moved due to the new behavior brought by the attack. As time passes, the past micro-clusters regarding normal traffic become less relevant and are erased by CluStream. The DoS behavior predominates, making the mean of maximum distances drop. When the DoS attack stops (packet #15136), there is a sudden behavior change again, which reflects in the mean of maximum distances. Later in this scenario (packet #23003), a MITM attack starts, showing again that the proposed approach could detect it quickly, generating alerts.

VI. Conclusion

The increasing adoption of IoT home devices draws attention to their security. Attackers see new opportunities to break into these devices since they are more vulnerable and getting more numerous. This paper proposed a new approach for change detection in the network behavior through an unsupervised stream learning algorithm. After organizing the packets into three streams, CluStream instances process them, maintaining the packets partitioned in micro-clusters at low memory cost. Then, the Page-Hinkley Test is applied over the mean of maximum distances among micro-cluster centroids to detect changes in the device's incoming traffic. The proposed approach does not require labeled samples to be trained or build a learning model.

Experiments performed with data from five devices in eleven scenarios showed that the best results were mostly found for TCP and ICMP streams. A total of six out of eleven scenarios reached 100% of detection rate and precision, which is a great result. Also, despite the different changes that distinct attacks caused in the micro-clusters' behavior, the proposed approach could detect all experimented types of attacks.

References

[1] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet of Things Journal*, vol. 6, pp. 9042–9053, 2019.

[2] D. Pishva, "Internet of things: Security and privacy issues and possible solution," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, 2017, pp. 797–808.

[3] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster, "User perceptions of smart home IoT privacy," *Proc. ACM Hum.-Comput. Interact.*, vol. 2, no. CSCW, Nov. 2018.

[4] R. Chow, "The last mile for IoT privacy," *IEEE Security & Privacy*, vol. 15, no. 6, pp. 73–76, 2017.

[5] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, 2019.

[6] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proceedings 2003 VLDB conference*. Elsevier, 2003, pp. 81–92.

[7] J. Gama, *Knowledge discovery from data streams*. CRC Press, 2010.

[8] O. Brun, Y. Yin, E. Gelenbe, Y. M. Kadioglu, J. Augusto-Gonzalez, and M. Ramos, "Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments," in *Security in Computer and Information Sciences*, E. Gelenbe, P. Campegiani, T. Czachórski, S. K. Katsikas, I. Komnios, L. Romano, and D. Tzovaras, Eds. Cham: Springer International Publishing, 2018, pp. 79–89.

[9] Y. Wan, K. Xu, G. Xue, and F. Wang, "IoTArgos: A multi-layer security monitoring system for internet-of-things in smart homes," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 874–883.

[10] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[11] V. H. Bezerra, V. G. T. da Costa, S. Barbon Junior, R. S. Miani, and B. B. Zarpelão, "IoTDS: A one-class classification approach to detect botnets in internet of things devices," *Sensors*, vol. 19, no. 14, 2019.

[12] R. Heartfield, G. Loukas, A. Bezemskij, and E. Panaousis, "Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1720–1735, 2021.

[13] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[14] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet, "River: machine learning for streaming data in Python," 2020.