ARTEMIS: An Intrusion Detection System for MQTT Attacks in Internet of Things

Ege Ciklabakkal
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
ciklabakkal.ege@metu.edu.tr

Ataberk Donmez
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
ataberk.donmez@metu.edu.tr

Mert Erdemir
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
mert.erdemir@metu.edu.tr

Emre Suren
Informatics Institute
Middle East Technical University
Ankara, Turkey
emre.suren@metu.edu.tr

Mert Kaan Yilmaz
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
kaan.yilmaz@metu.edu.tr

Pelin Angin
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
pangin@ceng.metu.edu.tr

***Abstract*—The Internet of Things (IoT) is now being used increasingly in transportation, healthcare, agriculture, smart home and city systems. IoT devices, the number of which is expected to reach 25 billion all over the world by 2021, are required to be deployed very fast, taking into account commercial pressures. This results in a very important layer, i.e. security, being either completely neglected or having significant shortcomings. Since IoT has a heterogeneous structure, there is a need for intrusion detection systems (IDSs) that take into account the specifics of an IoT system architecture, including the computing power limitations, variety of protocols and prevalence of zero-day attacks. In this paper, we describe ARTEMIS, an IDS for IoT, which processes data from IoT devices using machine learning to detect deviations from the normal behavior of the system and generates alerts in case of anomalies. We have implemented a prototype of the system using IoT devices subscribed to topics at an MQTT broker and provide experimental evaluation of the system under MQTT-related attacks.**

***Index Terms*—IoT, Intrusion Detection, MQTT**

I. INTRODUCTION

The advances in and wide availability of networking infrastructures and smart devices in the last decade have given rise to The Internet of Things (IoT) phenomenon, enabling the connectivity of physical and virtual objects to create smart environments. Although IoT systems are relatively new, IoT-enabled devices have already created a large attack surface for hackers to exploit. Notorious security incidents include a massive distributed denial of service (DDoS) attack[1] launched by hacking into thousands of security cameras, hackers remotely taking control of a Jeep Cherokee[2], the Stuxnet virus destroying a fifth of Iran's nuclear centrifuges[3], among others. Many of the current IoT devices lack basic security mechanisms, and there is a lack of standardization for IoT standards and protocols, which creates security loopholes.

This paper aims to contribute to the design of effective intrusion detection approaches for IoT systems, where devices communicate using the MQTT protocol. We propose an intrusion detection system (IDS) performing anomaly-based intrusion detection with machine learning (ML) algorithms to create alerts when the observed behavior of the system deviates significantly from its normal behavior learnt by the algorithms.

The main contributions of this paper are as follows:

- We describe a lightweight anomaly-based IDS for MQTT-based IoT networks.
- We provide a comprehensive experimental evaluation of the anomaly detection performances of six ML algorithms for detection of simulated MQTT attacks using data collected by the prototype IoT system developed.
- We provide a dataset consisting of packet captures generated by simulating attacks on the implemented IoT network. The unavailability of such IoT-specific datasets in the literature is a major problem for security researchers.

II. RELATED WORK

Intrusion detection in IoT has been a popular area of research for the past few years, owing to the significant adverse effects of cyber attacks on IoT systems. Kasinathan et al. [1] adapted Suricata, a signature-based IDS to detect DoS attacks in 6LoWPAN networks. Their system analyzes the IDS alerts of channel interference rate and packet dropping rate to confirm the attack along with reducing the false alarm rate. Liu et al. [2] proposed a signature-based IDS that utilizes Artificial Immune System (AIS) techniques. This approach is not a suitable deployment for IoT networks containing low capacity nodes due to the cost of attack signature storage and running algorithms. Cho et al. [3] proposed an anomaly-based detection scheme for botnets in 6LoWPAN sensor networks. The solution monitors the network traffic and notifies when unexpected changes in the computed averages for packet length and number of connections are observed for any node.

[1] https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

[2] https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[3] https://www.businessinsider.com/stuxnet-was-far-more-dangerous-than-previous-thought-2013-11

Lee et al. [4] leveraged the regular energy consumption as a parameter to detect anomaly behavior in low capacity 6LoWPAN networks. Summerville et al. [5] designed a deep-packet inspection method for anomalies that is capable of running on resource constrained IoT devices. They experimented with two Internet-enabled devices and the false positive rates for the worm propagation, tunneling, SQL code injection, and directory traversal attack types were shown to be low. Pongle and Chavan [6] designed three algorithms to detect wormhole attacks in IoT networks. Although the system is suitable for low resource IoT devices, the authors did not report the false positive rates.

The main shortcoming of previous work is that they were mostly not evaluated with datasets specific to IoT and/or IoT-specific communication protocols and attacks. Closest to our work is that of Alaiz-Moreton et al. [7], which utilized three methods, Extreme Gradient Boosting (XGBoost), GRU Recurrent Neural Networks, and LSTM Recurrent Neural Networks, for detecting 3 types of attacks, DoS, man-in-the-middle, and an MQTT-specific intrusion. However, we consider different ML algorithms and focus on detection of anomalies rather than multi-class attack classification.

III. Methodology

A. System Architecture

The architecture of the IoT network and IDS we model in this work is as shown in Fig. 1.

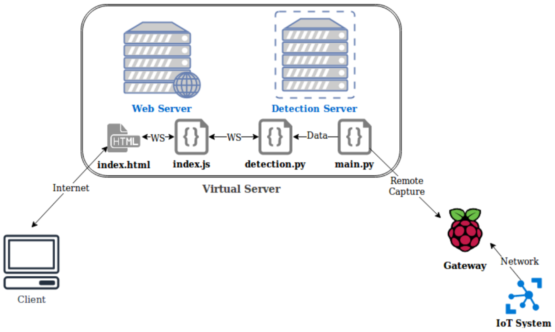

Fig. 1. System Architecture

The IoT system consists of various devices that share data such as sensor readings using MQTT. There is a single gateway device (a Raspberry Pi in the prototype) to which the sensors connect and every message published is communicated over this gateway. From the remote server, we capture packets using tcpdump and ssh. By directly transferring the packets to the server and completing the computationally expensive tasks such as training and prediction here, we minimize the processing done on the gateway device. We also present users with a simple UI to monitor the system.

We train ML models (using Python Outlier Detection Library - PyOD [8]) to make predictions on each packet. For generating alerts, we use the predict_proba[4] method of the prediction model. We use the formula below, in order to take advantage of the outlier probabilities of the previous batches as well, where *mov_avg* is the moving average outlier probability, *prev_mov_avg* is the moving average outlier probability for the previous batch, *avg* is the outlier probability of the current batch, and *W0* is the weight. *W0* essentially indicates how much we value past data. In our experiments, we set it to 0.4.

$$mov\_avg = W0 * prev\_mov\_avg + (1 - W0) * avg \quad (1)$$

Alerts are generated based on the difference of the current moving average and the previous moving average.

B. Data Collection

In the system prototype developed, we have a DHT11 sensor connected to a Raspberry Pi, which sends temperature and humidity data out. For diversity, we subscribe to some topics in public MQTT test brokers such as test.mosquitto.org and iot.eclipse.org. The Node-RED[5] IoT programming tool is used to set up connections between publish and subscribe nodes.

C. Feature Design

We use the information in the TCP, MQTT and IP layers of packets. In addition to these features we also calculate, for each packet, the average time between the 20 preceding packets. A complete set of the 31 features used can be seen in Table I.

TABLE I
Feature Set and Descriptions

No.	Name	Layer	Description
1	packet_length	-	Total packet length
2	src_ip	IP	IP address of the source
3	dst_ip	IP	IP address of the destination
4	ip_len	IP	Length of the IP layer
5	ip_df_flag	IP	Don't fragment flag
6	ip_mf_flag	IP	More fragment flag
7	ip_rb_flag	IP	Reserved bit flag
8	ttl	IP	Time to live
9	tcp_w_size	TCP	TCP window size
10	tcp_len	TCP	Length of the TCP layer
11	tcp_pdu_size	TCP	PDU size
12	tcp_ack_flag	TCP	Acknowledgement flag
13	tcp_cwr_flag	TCP	Congestion Window Reduced flag
14	tcp_ecn_flag	TCP	ECN-Echo flag
15	tcp_fin_flag	TCP	Fin flag
16	tcp_ns_flag	TCP	NS flag
17	tcp_push_flag	TCP	Push flag
18	tcp_res_flag	TCP	Reserved flag
19	tcp_reset_flag	TCP	Reset flag
20	tcp_syn_flag	TCP	Synchronize flag
21	tcp_urg_flag	TCP	Urgent flag
22	tcp_src_port	TCP	Port number of the source
23	tcp_dst_port	TCP	Port number of the destination
24	tcp_tdelta	TCP	Time elapsed since the last pacekt
25	l20_avg	TCP	Average tcp_delta of last 20 packets
26	mqtt_header	MQTT	MQTT header flags
27	mqtt_msg	MQTT	MQTT message (payload)
28	mqtt_len	MQTT	Length of the MQTT message
29	mqtt_topic_len	MQTT	Length of the MQTT topic
30	mqtt_msg_type	MQTT	Type of the message
31	mqtt_qos_lvl	MQTT	MQTT Quality of Servic Level

IV. Evaluation

Using the developed system prototype, we performed experiments to evaluate the performance of the following ML algorithms: Autoencoder, Single-Objective Generative Adversarial Active Learning (SO_GAAL), Random Forest, Isolation

[4] PyOD library provides this method

[5] https://nodered.org/

Forest, One-Class Support Vector Machines (OCSVM), and K-means Clustering. In the captures that contain an attack, the MQTT malaria tool[6] was used to send messages containing fuzzy payloads as fast as the tool allows. In the clean version of the dataset, there are around 180k packets of which 100k are MQTT packets and the rest are mostly TCP packets of the related MQTT packets.[7] While we used the autoencoder, SO_GAAL, Random Forest and k-means methods to train a model with both benign and attack data, OCSVM and Isolation Forest were used to train a model with only benign data.

For autoencoder, we did not change the default parameters of the library. Random forest already achieved good results with the default parameters, therefore only the *n_estimators* parameter was changed to 100. We used the PyOD implementation of OCSVM and the *SelectKBest* method was used to select the best 24 features among the 31. We created our clustering models with the k-means and Brich methods from the Python sklearn library[8]. We have 2 clusters, inlier and outlier. For Brich, the branching threshold was set to 0.2.

Both the training and test sets of the attack dataset contain normal behaviour (benign) and fuzzing attack packets. Furthermore, JSON objects require special handling as they are not a primitive data type. Thus, we created filtered versions of the mentioned datasets, which do not include packets with JSON objects in their payloads. The distribution of the number of packets in the datasets is summarized in Table II.

TABLE II
PACKET DISTRIBUTIONS IN THE DATASETS

	Benign Without JSON Dataset	**Attack Without JSON Dataset**	**Benign With JSON Dataset**	**Attack With JSON Dataset**
# of Benign Packets	58748	232941	102738	286985
# of Attack Packets	0	135123	0	135123
Total # of Packets	58748	368064	102738	422108

We evaluated the performances of the methods with ROC AUC Scores (Table III) and Accuracy Scores (Table IV). The experiments were performed in four ways. The first and the second experiments use 80% of the benign dataset as the training set and 20% of the attack dataset as the test set for the models. While the second set of experiments used the packets with JSON objects in the payloads, the first did not use them. These experiments involved one-class algorithms (OCSVM and Isolation Forest). The third and the fourth experiments used 80% of the attack dataset as the training set and 20% of the attack dataset as the test set for the models. While the fourth set of experiments used the packets with JSON objects in the payloads, the third set of experiments did not use them.The third and fourth sets of experiments involved multi-class algorithms (Autoencoder, SO_GAAL, Random Forest and k-means). For the Tables III and IV, the columns represent these stages, where '-' indicates that the corresponding test was not applied for that method.

TABLE III
ROC AUC SCORES OF THE ML METHODS

	Benign Train - Attack Test Without JSON	**Benign Train - Attack Test With JSON**	**Attack Train - Attack Test Without JSON**	**Attack Train - Attack Test With JSON**
Autoencoder	-	-	0.4106	0.3788
SO_GAAL	-	-	0.8924	0.8728
Random Forest	-	-	1.0	0.8816
K-Means	-	-	1.0	0.8816
OCSVM	0.9998	0.9998	-	-
Isolation Forest	0.8326	0.5	-	-

TABLE IV
ACCURACY SCORES OF THE ML METHODS

	Benign Train - Attack Test Without JSON	**Benign Train - Attack Test With JSON**	**Attack Train - Attack Test Without JSON**	**Attack Train - Attack Test With JSON**
Autoencoder	-	-	0.5945	0.6378
SO_GAAL	-	-	0.8415	0.7860
Random Forest	-	-	1.0	0.8007
K-Means	-	-	1.0	0.8007
OCSVM	0.9998	0.9998	-	-
Isolation Forest	0.7534	0.8417	-	-

V. CONCLUSION

In this paper, we described the design and implementation of a lightweight anomaly-based IDS for IoT networks, focusing on attacks on MQTT. We generated a dataset that contains attacks for MQTT and provided it for the use of the security community. Our IDS integrates various ML techniques to classify IoT network behavior as normal or anomalous. We provided a comparative analysis of the performances of the k-means, SO_GAAL, OCSVM, Random Forest, Isolation Forest and autoencoder models for the anomaly detection task. The experiment results suggest that ML-based intrusion detection in MQTT-based IoT networks can achieve impressive results even when not trained with previously known attacks.

REFERENCES

[1] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits, "Denial-of-service detection in 6lowpan based internet of things," in *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*, pp. 600–607, IEEE, 2013.

[2] C. Liu, J. Yang, R. Chen, Y. Zhang, and J. Zeng, "Research on immunity-based intrusion detection technology for the internet of things," in *2011 Seventh International Conference on Natural Computation*, vol. 1, pp. 212–216, IEEE, 2011.

[3] E. J. Cho, J. H. Kim, and C. S. Hong, "Attack model and detection scheme for botnet on 6lowpan," in *Asia-Pacific Network Operations and Management Symposium*, pp. 515–518, Springer, 2009.

[4] T.-H. Lee, C.-H. Wen, L.-H. Chang, H.-S. Chiang, and M.-C. Hsieh, "A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, pp. 1205–1213, Springer, 2014.

[5] D. H. Summerville, K. M. Zach, and Y. Chen, "Ultra-lightweight deep packet anomaly detection for internet of things devices," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–8, IEEE, 2015.

[6] P. Pongle and G. Chavan, "Real time intrusion and wormhole attack detection in internet of things," *International Journal of Computer Applications*, vol. 121, no. 9, 2015.

[7] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Muñoz-Castañeda, I. García, and C. Benavides, "Multiclass classification procedure for detecting attacks on mqtt-iot protocol," *Complexity*, vol. 2019, pp. 1–12, 2019.

[8] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *arXiv preprint arXiv:1901.01588*, 2019.

[6] https://github.com/etactica/mqtt-malaria

[7] The full dataset containing raw packet captures is available at https://drive.google.com/open?id=1bNj1lNjU0Q3YxzhutMFCpSytPX8jFxC-

[8] https://scikit-learn.org/stable/modules/clustering.html