Man-in-the-Middle Attacks on MQTT-based IoT Using BERT Based Adversarial Message Generation

Henry Wong Tie Luo*
Department of Computer Science
Missouri University of Science and Technology, MO 65401, USA
E-mail: {henrycwong, tluo}@mst.edu

ABSTRACT

With IoT devices becoming more ingrained into everyday life and business, attacks on Internet-of-Things (IoT) systems can be costly and, in extreme cases, cause life-threatening situations and huge economic loss. Denial-of-Service (DoS) attacks have been well studied in cybersecurity, partly because of the Mirai malware which has shown extensive damage to an insecure network. However, Man-in-the-Middle (MITM) attacks have been largely overlooked especially in IoT networks. In this paper, we introduce a new scheme of a Man-in-the-Middle (MITM) attack on IoT devices that utilize the Message Queuing Telemetry Transport (MQTT) protocol for communications. This attack scheme consists of an MQTT Parser that is created to dissect and alter MQTT messages at the bit level, and a novel BERT-based adversarial model that generates malicious messages using an approach inspired by GAN. We present the design of this attack in order to show how a sophisticated attack could lead to serious damage that is difficult for typical security defense mechanisms to detect. We set up a test-bed using IoT hardware and software including Raspberry Pi, WiFi Pineapple, Mosquitto, etc. to conduct experiments. We show that our designed attack scheme successfully evades logistic regression, random forest, K-nearest neighbor, and support vector machine (SVM) based anomaly detection models. Multi-Layer Perceptron fares better against our model, but such use of deep neural networks on typical IoT devices is rather restricted due to the computation cost. In summary, the results show that the MITM attack is effective against a wide range of typical anomaly detection mechanisms.

KEYWORDS

Internet of things (IoT) security, man-in-the-middle attack, denial-of-service attack, anomaly detection, MQTT, BERT.

1 INTRODUCTION

Recent years have seen a sharp increase of the popularity of Internet-of-Things (IoT) devices due to their various benefits and low cost. The popularity, however, comes at a price with an large number of attacks targeting IoT devices, such as Mirai [2] and BrickerBot [9] as well as more conventional DDoS, spoofing, and Sybil attacks. Among these, direct hacking such as malware takes up 45% of all security breaches [3], which demonstrates that firewalls are often inadequate to protect networked systems.

Message Queuing Telemetry Transport (MQTT) is a lightweight application-layer protocol that uses a publish-subscribe model to enable communication among low-resource IoT devices. A *publisher* is typically a data-generating device (e.g., a sensor) that sends messages with a *topic* to a *broker*. Then a *subscriber* is typically a user device (e.g., smartphone) that receives messages from the broker on the topics the user device subscribes to. The elegant design and performance has gained MQTT tremendous support and wide use to the extent of becoming "the de facto standard of IoT" [23], surpassing other popular protocols including ZigBee, CoAP, and LoRa (regardless of layers in a protocol stack).

However, MQTT does not have a sophisticated security mechanism on its own, but relies on SSL/TLS (Secure Socket Layer/Transport Layer Security) to encrypt MQTT messages. Unfortunately, this is poorly enforced in IoT deployments. To demonstrate this, we run a Shodan query[1] of "port: 1883", the default port for MQTT, and the result shows over 416,618 ports on (unsecured) TCP connections; on the other hand, a query for "port: 8883", the default SSL/TSL port for MQTT, only shows 38 ports on (secured) TCP connections (as of May 16, 2020). This vast ratio of 12,625:1 presents a serious security vulnerability in IoT systems that use MQTT. Note that this vulnerability is not easy to fix, because of legacy reasons (an enormous number of existing deployments), practical reasons (users lack of technical know-how on SSL/TLS configuration), and technical reasons (SSL/TLS consumes extra resources which can be unfriendly to some IoT devices).

In this paper, we show that this vulnerability can indeed be exploited by adversarial users and such exploitation can be hard to detect. To this end, we design a Man-in-the-Middle (MITM) attack scheme. MITM attacks, also known as hijack attack, are a cyber attack where an attacker intercepts a network connection to alter the data being transferred between the two ends. For example, in Figure 1, an attacker hijacks the wireless connection between an aircraft and a control tower, in such a way that the aircraft deems the attacker as the control tower and the control tower deems the attacker as the aircraft; thus, the attacker serves as a malicious relay that can both eavesdrop and alter communications without being noticed.

The reason we choose MITM attacks is because it grants the attacker a large degree of freedom for manipulation, by having both read and write access (within a network of IoT devices) to sensitive or critical information without being noticed. In comparison, eavesdropping has no write access, and both jamming and Denial of Service (DoS) attacks can be easily detected. On the other hand, MITM attacks are technically challenging to design because the attacks need in-depth understanding of protocol and network details.

KDD'20 Workshops: the 3rd International Workshop on Artificial Intelligence of Things (AIoT), August 24, 2020, San Diego, CA.
*Corresponding author.

[1] Shodan is dubbed as the "Google of IoT" which allows users to search the Internet for various IoT devices.

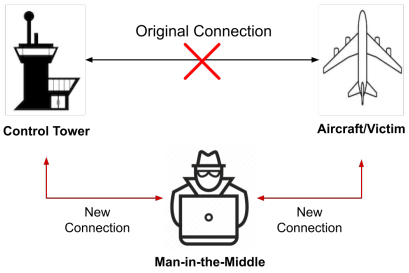

Figure 1: Man-in-the-Middle attack: an illustration.

To make our design sophisticated, in the sense that the attack can be automated and is hard to be detected (typically by anomaly detection mechanisms), we are inspired by generative adversarial networks (GAN) [7] and use the Bidirectional Encoder Representations from Transformers (BERT) model[6], an NLP pre-training technique, to construct a discriminator in our attack model. The design also involves the creation of a new MQTT Parser written in Python. We verify the effectiveness of our attack scheme against five representative anomaly detection algorithms using real IoT devices. In summary, this paper makes the following contributions:

- We design a Man-in-the-Middle attack to show the vulnerability of the "the de facto standard of IoT", MQTT, and how the vulnerability can be exploited.
- We develop *the first* open-source MQTT Parser[2] that can dissect and modify MQTT packets as per need.
- We are *the first* to apply an NLP technique (BERT) to the design of MITM attacks; in particular, we show how to automate the alteration of MQTT messages by constructing an adversarial model, which is inspired by GAN and based on BERT. We have also open-sourced this adversarial model.[3]
- We conduct experiments with real IoT devices and protocols (Raspberry Pi's, WiFi Pineapple, Eclipse Mosquitto, and Eclipse Paho). The results show that our designed MITM attack is elusive to detect, and evades most representative anomaly detection algorithms.

2 RELATED WORK

2.1 MQTT Security

With an increasing number of cyber attacks each year, the field of IoT security continues to expand, mostly towards Intrusion Detection Systems (IDS). The ARTEMIS Intrusion Detection System [5] uses multiple machine learning techniques to detect malicious MQTT messages with a promising accuracy of 0.9998 when using OneClassSVM. However, the tested attacks were generated with the malaria toolkit [18] which leaves it up to the user to create custom malicious messages, allowing the possibility of biased data, and the messages that were tested only contain numerical (temperature) values.

Besides IDS, authentication is another method to bolster security. A typical example is the use of OAuth 1.0, which has been explored for MQTT-based IoT devices [17]. This method allows low-powered devices that cannot use SSL/TLS to connect to an OAuth server. However, the configuration and implementation are technically challenging for most IoT users, leaving the method only useful to expert users. OAuth 1.0 specializes in protecting against Denial-of-Servie (DoS) and flooding attacks only, while MITM attacks can be launched if the attacker can replicate the authentication certificate.

Both of the above approaches, as well as most work done in the MQTT field, recommend using Secure Socket Layer and Transport Layer Security (SSL/TLS) to secure MQTT communications if possible. SSL and TLS are cryptographic protocols that allow communications to be encrypted, thereby making attacks like sniffing and MITM more difficult. However, as mentioned by Niruntasukrat et al. [17], some devices do not have the resources to support SSL/TLS. To this end, one can use a layer based approach [21] which offloads the processing (e.g., cryptographic) task to other devices that have enough resources. However, this approach specializes in preventing DoS and flooding attacks rather than Man-in-the-Middle attacks. There are other IoT security approaches such as JPAKE [10] (on device) and DIoT [16] (federated learning based), but they also prioritize protecting against DoS attacks.

2.2 BERT and Feature-Based Learning

BERT was introduced in 2018 by Google [6], which is an application of feature based learning to Natural Language Processing. It is a multi-layer bidirectional transformer encoder utilizing the transformer introduced by Vaswani el al. [24], and improves on top of left-to-right [15], right-to-left, and bidirectional feature models [19] in terms of speed and accuracy.

Since the introduction of BERT to the NLP community, many variations of the pre-training model has surfaced, including BioBert [12], SciBERT [4], and ClinicalBERT [1, 8]. There are also BERT variations in terms of size of the model, such as DistilBert [22] and Albert[11] which are smaller and faster versions of BERT that reduce the number of feature in BERT but as a result sacrifice performance. Other distinct variations of BERT with extended features include ERNIE [25] which incorporates a knowledge graph to pre-training, M-BERT [20] which is a model that pre-trains on 104 different languages, and ViLBERT [14] that uses both textual and visual inputs to train models. Models that reportedly claim to have to better results than Bert like RoBERTa [13] have surfaced. RoBERTa utilizes dynamic masking, Full-Sentence input (to achieve zero loss in Next-Sentence Prediction (NSP)), training in larger batches, Byte-Pair Text Encoding, and training on data sets ten times larger than the data BERT uses. However, RoBERTa requires much more resources, with an entire day of training on state-of-the-art GPUs, while BERT only requires a few hours.

3 DESIGN OF ATTACK

To facilitate understanding, and without loss of generality, suppose the victim is an aircraft engine or data center (hardware), which needs to be operated within a certain temperature range. The hardware is monitored and controlled by a thermostat over a wireless network via the MQTT protocol (see Figure 2). The goal of the attack is to disrupt this monitoring and control system such that malicious temperature can be injected to cause the aircraft engine or the data center servers to shutdown or explode, which can result in life or huge economic loss.

[2]https://github.com/HenryCWong/MQTTPython
[3]https://github.com/HenryCWong/adversarialBERTMessages

Figure 2: Victim system (an example): Temperature monitoring and control for an aircraft engine or a data center.

3.1 Overview

We consider a representative case where the victim network connection goes through a WiFi router. In fact, one of the main reasons that MQTT has gained such signi cant popularity is because it can operate over 802.11 or WiFi, which is ubiquitously available and o ers higher data transmission rate than other protocols such as Zigbee or Bluetooth. For attacking purposes, we use a WiFi Pineapple, which is a penetration testing tool created to allow cyber security professionals to test the vulnerability of a computer network. In our case, we use the WiFi Pineapple to de-authenticate and spoof the legitimate WiFi router (by spoo ng the WiFi's SSID) and leverage the Pineapple's capability of injecting raw packets to devices connected to it.

Figure 3 provides an overview of our MITM attack. The thermostat (publisher) connects to the aircraft engine or data center (subscriber) via the WiFi router, mediated by a MQTT broker.

(1) The WiFi Pineapple rst masquerades as the WiFi router by using the same SSID and waits for devices to connect to the Pineapple.
(2) Once the devices (are fooled to) connect to the WiFi Pineapple, the MQTT packets are dismantled and altered (Section 3.2) into malicious messages. Note that the WiFi Pineapple is still connected to a legitimate WiFi connection to provide connection for the rest of the network, so that the network still appears normal to the devices being attacked.
(3) The malicious messages dictate the temperature to be sent to the MQTT subscriber (climate control units of the aircraft engine or data center), where the temperature varies strategically based on an adversarial model (Section 4).
(4) The temperature is received by the victim (MQTT subscriber) which then regulates temperature in the attacker-desired manner and causes damage.

3.2 MQTT Parser

Despite MQTT being a popular IoT protocol, there does not exist a library that can parse MQTT packets, which is important because such a library is the foundation to enable protocol analysis for MQTT. This motivated us to write a MQTT parser in Python, which can dissect MQTT packets into bytes and modify them as per need, and we have open-sourced it.

Figure 3: Overview of the MITM attack.

Our MQTT parser functions as follows. First, WiFi Pineapple uses the TCPDump module to dump all the received TCP packets into .pcap les (a format originally created for TCPDump and an abbreviation for packet capture), each containing multiple (in our case 4000) packets. Next, each .pcap le is passed to the MQTT parser, which will create a MQTT Object, then call its internal functions to decode the le at the bit level to ASCII into MQTT Object Types (messageType, messageLength, topicLength, topicName, messageWords); see Figure 4. Following that, a function translates the hexadecimal message into ASCII to send to an adversarial model described in Section 4 to generate a malicious message. Another function then takes this malicious message and translates it back into hexadecimal format. Finally, the WiFi Pineapple injects this altered (and malicious) message to the subscriber (victim).

Figure 4: Structure of a MQTT packet.

4 ADVERSARIAL MESSAGE GENERATION

This section describes an adversarial model that can be used to not only automate the malicious message generation process, but also achieve the following objectives:

The malicious messages need to conform to standard protocol speci cation and the message content needs to make sense to the receiver (so that the device will take the attacker-intended actions to cause damage);
The malicious messages need to be prudently designed so as to bypass the incumbent anomaly detection systems.

In this paper, we cast the problem as an adversarial game, in which two models confront each other to create a malicious message. This idea is inspired by Generative Adversarial Networks (GAN), but we do not use GAN because its complexity is not necessary to handle our particular case of MQTT.

Resembling GAN, we construct one model to act as a generator to generate malicious messages, and construct the other model to act as a discriminator that separates real messages from malicious messages. Algorithm 1 outlines the overall process between the Generator and the Discriminator.

Algorithm 1: GAN-Inspired Adversarial Model

```
Input: - , a set of words from captured MQTT Messages
Output: W to be used by Generator
W  Arbitrary Large Number
d  a small number between 1 W 0
B2>A4  0
while optimal (lowest) B2>A4 is not found do
    -  Generator( , W)
    B2>A4  Discriminator( - , W, d)
    W  W  d
```

4.1 Generator

The generator uses two algorithms. For numerical values such as temperature (Algorithm 2), it takes a range of random numbers between one standard deviation ( ) below and above the mean, 1- º, scaled by a weight W. The value W is determined by Algorithm 4.

Algorithm 2: Generator for Numeric Values

```
Input: - : A set of numbers from captured MQTT Messages
       W: Adjusted Weight
Output: V: A malicious set of numbers
- <40=1- º
( -  BC3 º
V ;
for all - 8 2 - do
    V8  A0=3><1 1-  1 W ( - º•- , 1 W ( - ºº
```

For non-numeric (i.e., alphabetic) messages, we need a di erent approach. We choose a word (or multiple words depending on the size of the message) and nd the antonym of that word as outlined in Algorithm 3. In choosing the word, we prioritize adjectives because most MQTT use-cases are IoT devices that send messages to describe an event or object. The chosen word is determined by W, and hence each iteration will choose a di erent word. To identify adjectives for the experiment, the Natural Language Toolkit (NLTK) Library's Wordnet corpus for Python was used. The method of processing non-numeric messages is formulated in Algorithm 3.

Algorithm 3: Generator for Non-Numeric Values

```
Input: - : A set of words from captured MQTT Messages
       W: Adjusted Weight
Output: V: A malicious set of numbers
X  0
\  0B 1W  • hash() converts V into a unique integer
for all - 8 2 - do
    X  Number of Words mod \
    V8  Antonym of X-th word in the message 8
     (prioritizing adjectives)
```

4.2 Discriminator

The discriminator utilizes BERT and a Neural Network to di erentiate between malicious messages made by the Generator and real messages that were received from the original sender. BERT allows models to learn from all the words in a sentence without having to su er from long-term dependencies. It has a self-attention unit, which calculates the pair-wise correlation between all the words in each sentence in order to tune a weight on each word. BERT is bidirectional, in the sense that it can calculate from both left-to-right and right-to-left.

In this paper, we use BERT to pre-train the MQTT messages to check for bi-directional context which allows the model to nd correlations between words. In particular, we use DistilBERT [22] which is a lighter version of BERT: BERT pre-trains with 12 layers while DistilBert only uses 10 layers and removes other features such as token-type embeddings and pooling, while retaining 97% of BERT's performance. With marginal performance loss, DistilBert is faster (hence helpful in avoiding being detected) and is reported to perform better than ELMo [19].

To further expedite the process, we batch all the messages through padding and tokenization before running the messages through DistilBERT. After pre-training, a Multi-Layer Perceptron was used to classify whether the messages were malicious messages from the Generator or original messages from the captured MQTT packets. Algorithm 4 illustrates how the Discriminator works.

Algorithm 4: Discriminator

```
Input: - : A set of words from captured MQTT Messages
       W: Adjusted Weight
Output: B2>A4 Accuracy of MLPClassi er()
I  BERT( )
B2>A4  MLPClassi er( )  • Classi er returns the accuracy
```

5 EXPERIMENTS

5.1 Testbed Setup

Our testbed uses two Raspberry Pi 3B's, one as a Publisher and the other as a Subscriber. The testbed also consists of a WiFi Pineapple to perform Main-in-the-Middle attack, and a WiFi access point (AP). We use MQTT version 1.5.7 and Mosquitto version 1.6.9 to send and receive MQTT messages, where the control scripts are written in Python using the Eclipse Paho 3.1.1 library. The messages contain various sensory data including temperature, weather forecasts, wind direction, and humidity. To interact with the WiFi Pineapple remotely, we use Secure Shell (SSH) from a laptop terminal.

5.2 Results

Our primary goal was to evaluate the performance of our adversarial model, by testing a range of anomaly detection (AD) models against the malicious messages generated by the adversarial model (from 250 messages used to train). These messages were chosen from messages that were classified as false negatives in the Discriminator. We choose five typical AD models that use Logistic Regression, Random Forest, K-Nearest Neighbor (KNN), Multi-Layer Perceptron (MLP), and Support Vector Classification (SVC). In particular, random forest and SVC are also used in IDS systems like ARTEMIS [5]; MLP is relatively more resource-consuming and hence not always suitable for IoT devices, but we still include it to see how our model performs against such an upperbound.

In Table 1, the second column gives the results for our adversarial model with BERT removed, in the format of number of False Negatives (mis-detection) over the total number of malicious messages (False Negatives + True Positives). The third column (FN) gives the false negatives when BERT is used (the complete model). The fourth column represents the improvement of the third column over the second. We can see that our adversarial model fares well against Logistic Regression, Random Forest, and SVC. In the case of KNN, it is better to disable BERT. However, in either case (with or without BERT), our model is able to achieve a very high false negative rate (more than 63%), indicating that it is effective even in the case of KNN. In the case of MLP, using BERT is not desirable as there is a substantial decrease in FN rate. However, MLP as a deep neural network model does not fit most IoT devices due to the amount of required resource; furthermore, even in (rare) cases such deep models are used, we can simply remove BERT to still achieve a high (more than 50%) false negative rate, which is desired by the attacker. Finally, Table 1 demonstrates that BERT significantly improves the malicious messages against the SVC model with an improvement of 92.5%; in fact, it completely evades the AD model, resulting in a 100% mis-detection (FN) rate.

Figure 5 shows how the choice of W values affects accuracy and false negatives within the adversarial model. The accuracy and false negatives shown are results of the Discriminator using an MLP classifier. The figure shows the optimal W to be around 1.8 since the accuracy is low and the number of false negatives is high. Since even the Discriminator cannot classify these malicious messages well at these W values, it is unlikely for IoT devices with less resources than the attacker running the adversarial model to achieve a better accuracy or false negative rate.

Figures 5, 6, and 7 are based on information recorded while the adversarial model was running. Both Figures 6 and 7 are results of running the multiple classification models against messages that were output by the Generator. Both had no direct interaction with the Discriminator, unlike Figure 5 which records the values from the Discriminator. The results in Table 1 were collected after running the adversarial model and were tested with the malicious messages at optimal W values.

Classification Model	FN w/o BERT	FN	Improvement
Logistic Regression	58/102	81/90	58.3%
Random Forest	69/102	89/90	46.2%
KNN	69/102	57/90	-6.4%
MLP	99/102	12/90	-86.3%
SVC	53/102	90/90	92.5%

Table 1: Results of using different classification models to detect malicious messages generated from our adversarial model. The FN (third) column represents results for our adversarial model using BERT. The total numbers of malicious messages in columns two and three are different (102 vs 90) because the total number of malicious messages are only messages that fooled the Discriminator, which varies depending on whether BERT is used.

Figure 5: Effect of W on Accuracy and False Negatives on the Discriminator (the FN here is not to be confused with the FN in Table 1).

Figure 6 shows the number of false negatives of each AD model while the Adversarial model was running. All models except MLP and KNN show a slight inverse parabolic curve as the number of false negatives peaks at an approximate optimal W (around 1.8). An optimal W is further visualized in Figure 7 which describes the accuracy of the same situation as Figure 6. This Figure instead represents a slight parabolic curve (with the exception being MLP) with a similar optimal W. This proves that for the generated malicious messages, W is not just optimal for the Discriminator model, but also in multiple AD models.

6 CONCLUSION

In this paper, we present a well-designed MITM attack on MQTT-based IoT devices, due to the popularity and wide deployment of MQTT in IoT networks. This was intended to motivate follow-up research on such less studied attacks in the IoT context. Our MITM attack scheme consists a MQTT Parser created for dissecting and altering MQTT messages, and an adversarial model to generate