Improving MQTT Security Through the Generation of Malicious Test Cases

Camilla Cespi Polisiani[1], Maria Carla Calzarossa[1], Marco Zuppelli[2], Luca Caviglione[2,*] and Massimo Guarascio[3]

[1]*University of Pavia, Pavia, Italy*
[2]*Institute of Applied Mathematics and Information Technologies, Genova, Italy*
[3]*Institute for High Performance Computing and Networking, Rende, Italy*

Abstract
The pervasive deployment of IoT technologies accounts for a variety of hazards often requiring a cross-layer approach. For example, the security posture of brokers responsible for handling the Message Queuing Telemetry Transport (MQTT) protocol has to be assessed at different functional layers, thus it is important to generate test cases ranging from network traffic conditions to application-specific patterns. Alas, this is a time consuming and poorly-generalizable process. Therefore, this paper proposes two frameworks for improving IoT security. The first is a suite for creating traffic flows starting from real traces or arbitrary configurations. The second is a Small Language Model that can produce realistic MQTT topics. To demonstrate their effectiveness, we showcase how they can be used to mitigate covert communications targeting IoT ecosystems. Results indicate that our tools can provide realistic test conditions for advancing IoT security, especially to better comprehend attacks targeting the MQTT protocol.

Keywords
covert communications, IoT security, test cases

1. Introduction

The rapid diffusion of Internet of Things (IoT) technologies is responsible for the transformation of various domains ranging from smart home environments to industrial control systems and digital urban infrastructures [1]. At the same time, their ubiquitous adoption opens the way to several security hazards that might also damage physical assets or harm individuals. In fact, the mix of hardware, software and network protocols results in a vast attack surface difficult to control. For instance, many modern IoT deployments are plagued by data breaches, unauthorized access, and exfiltration attempts. Therefore, enforcing IoT security is crucial for not endangering people, homes, industries, cities or the whole Internet [2].

Several architectural blueprints and network protocols have been introduced to support the rapid evolution of IoT applications also with the goal of improving their robustness. Among the others, deployments based on the Message Queuing Telemetry Transport (MQTT) are demonstrating their effectiveness, since they allow the organization of IoT nodes and data in a hierarchical structure, thus offering the possibility of segmenting the network to prevent bottlenecks [3]. Accordingly, many IoT ecosystems rely on an MQTT broker, which serves as a central communication hub that receives and dispatches messages across clients via a publish-subscribe paradigm. Given their importance, brokers are attractive targets for attackers as they might be affected by vulnerabilities that can compromise the entire IoT deployment [4, 5].

Owing to the critical nature of IoT technologies, efforts to counteract attacks have intensified but spawned an "arm race" leading to malware endowed with mechanisms to "obscure" data within network

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT
*Corresponding author.
camilla.cespipolisiani01@universitadipavia.it (C. C. Polisiani); mcc@unipv.it (M. C. Calzarossa); marco.zuppelli@cnr.it (M. Zuppelli); luca.caviglione@cnr.it (L. Caviglione); massimo.guarascio@cnr.it (M. Guarascio)
0000-0003-1015-3142 (M. C. Calzarossa); 0000-0001-6932-3199 (M. Zuppelli); 0000-0001-6466-3354 (L. Caviglione); 0000-0001-7711-9833 (M. Guarascio)

protocols or mimic normal application behaviors, just to mention some [6, 7]. A recent offensive trend is to deploy covert channels, which establish hidden communication paths within legitimate traffic flows to exfiltrate sensitive data, evade signature-based detection mechanisms, or orchestrate nodes of a botnet [7, 8]. Alas, improving security requirements of IoT ecosystems is often a complex task because of the dynamic, heterogeneous, and resource-constrained nature of IoT nodes. Among the various issues, enforcing segmentation through traffic engineering and the ability of performing automatic, reproducible, and consistent tests are major concerns (see [9] and the references therein). Therefore, this work showcases two mechanisms for the generation of test conditions to evaluate IoT security. Specifically, the first approach entails a framework for assessing the network part of an IoT ecosystem via the creation of traffic flows starting from real traces or arbitrary configurations. The second leverages a Small Language Model (SLM) to produce realistic MQTT topics used to quantify the permeability of brokers to covert communication attempts.

Summing up, the contribution of this work is threefold: *i*) it introduces two frameworks for testing the security properties of IoT ecosystems at different functional layers, i.e., traffic and topic levels; *ii*) it showcases the effectiveness of SLMs to automatically generate test cases for improving the robustness of IoT deployments against covert communications; *iii*) it considers a threat model leveraging information hiding, which is often neglected in the literature dealing with IoT/Cyber-Physical Systems (CPS).

The rest of the paper is structured as follows. Section 2 reviews past works on the assessment of IoT security, while Section 3 introduces the proposed test mechanisms. Section 4 showcases numerical results and Section 5 concludes the paper and hints at future research directions.

2. Related Work

Owing to their diffusion, the literature abounds of works dealing with security of IoT ecosystems. For instance, a recent survey highlights the major challenges that should be addressed in the near future, such as the lack of effective encryption schemes at the transport layer, insufficient authentication/authorization mechanisms and insecure cloud interfaces [2]. Moreover, the need of orchestrating a vast array of hardware entities (e.g., sensors, system on a chip frameworks, and resource constrained platforms) accounts for major software hazards [10]. In fact, nodes and appliances are often plagued with backdoors, firmware inheriting unpatched CVEs due to lack of control of the used codebase, and hazards arising from prioritizing performance over security [11]. Another important aspect concerns the ability of facing threats that can virtually target all the functional layers of IoT technologies, e.g., from bare metal to the application. Hence, being able to conduct tests is mandatory, especially to capture corner cases or the complex interplay of different hardware, software and vendors characterizing real-world deployments.

Concerning the generation of network traffic to test IoT infrastructures, [12] introduces a tool for supporting IoT network simulations, e.g., for evaluating security countermeasures. However, this tool is affected by some limitations: it cannot accurately simulate event-driven IoT devices, and it imposes a periodic publication pattern instead of more realistic time-varying behaviors. The literature offers many works dealing with the development of synthetic traffic models. For example, [13] showcases how the Scapy Python library can be used to produce traffic patterns characterizing IoT nodes deployed in smart home scenarios, whereas [14] discusses a workaround to the scarcity of datasets needed to drive models and obtain accurate results. Specifically, it suggests to deploy generative adversarial networks to create better traffic models, e.g., capable of taking into account also location information. A more refined approach is presented in [15].

Despite the used mechanisms, traffic generation schemes share some limitations. The first is the lack of comprehensive datasets, especially for the case of the MQTT protocol. A major exception is [16], which proposes a collection of IoT traffic traces capturing various network attacks. However, background traffic conditions are generated using the tool described in [12], thus the obtained dataset contains IoT nodes with the same "duty cycle". The second limitation observed in the literature is that some tools are not publicly available or have been created to investigate very narrow deployments, e.g., smart homes. For analyzing the impact of covert communications, [17] showcases a framework to produce traffic conditions representative of a variety of covert channels, which may also be suitable for IoT ecosystems. Moreover, [18] deals with a tool for cloaking information within `.pcap` traces that also offers a prime support to threats hiding data within MQTT message headers, such as the `Keep-Alive` and `Client ID`.

The mitigation of covert communications can surely benefit from the availability of mechanisms for generating test cases. In fact, identifying covert communications *a-posteriori* is a hard task, especially in IoT scenarios, where data could be hidden in multiple places, e.g., measurements of sensors or traffic traits. This is even more evident for the case of cyber-physical deployments, where the physical behavior of sensors/actuators or the timing of protocol data units could be exploited to conceal information [19]. To this extent, the literature proposes three main paradigms. The first acts in the early design stage of protocols, mainly to eliminate functional ambiguities, imperfect isolation issues, or optional/unused fields that can be abused as containers for the secret data [20]. The second takes advantage of some form of AI to develop models capable of replicating traffic conditions or "challenge" detection frameworks for improving their robustness [21]. In both cases, a core requirement is the ability to conduct a vast array of real trials, e.g., to gather traffic traces that can be used to drive simulations or train models. A third alternative approach deploys network-level fuzzers, proven to be effective to produce a multitude of test conditions in an automatic manner, especially to reveal coding errors, bugs, or security issues [22]. Even if pseudo-fuzzing techniques based on random permutation are effective to assess the susceptibility of HTTP headers against covert communication attempts [23], their systematic adoption is still vastly unexplored. At the same time, the use of AI for creating network fuzzers appears a very promising approach to investigate a wide range of security hazards or implementation issues [24].

Summing up, the limitation of available tools and the lack of comprehensive MQTT traffic datasets underscore the need for automated approaches able to generate realistic network traffic conditions or ad-hoc test cases for capturing specific traits of IoT deployments.

3. Mechanisms for Testing IoT Ecosystem Security

In this section, we present two mechanisms for assessing the security of IoT ecosystems. First, Section 3.1 describes a framework able to produce different traffic conditions to test IoT devices at the network level. Then, Section 3.2 showcases how SLMs can be used to tune mechanisms to reveal manipulations of MQTT topics.

3.1. Threat-driven Traffic Generation

To address the lack of realistic MQTT traffic datasets for engineering and research purposes, we created a framework able to generate both benign and threat-driven MQTT traffic conditions implementing a controlled yet realistic test environment. The tool enables the simulation of diverse network threats, including malicious software endowed with different types of covert channels and Denial of Service (DoS) attacks.

From an architectural viewpoint, the generator is organized into three functional layers. The initialization layer processes input configurations, establishing the parameter settings for the simulation scenario. The traffic simulation layer generates the MQTT traffic by considering clients publishing and subscribing to topics, and reproducing both benign and malicious behaviors. Finally, the execution and control layer manages the overall operation of the generator, coordinating the network traffic creation phase, ensuring the startup and shutdown of components, and enabling the traffic capture in `.pcap` format for further analysis.

The tool also supports a detailed modeling of IoT devices, including periodic sensors that transmit data at a fixed rate, and event-driven sensors that publish messages at instants of time sampled from a specified probability distribution, such as exponential or uniform. The flexibility in customizing the timings using different distributions makes the tool particularly suitable for simulating diverse IoT behaviors, such as CPS or large-scale urban deployments [2].

The tool has two built-in attack templates. The first allows the simulation of DoS attacks targeting the MQTT broker by flooding it with `PUBLISH` messages with large payloads from a set of tampered IoT nodes, while legitimate IoT devices, such as environmental sensors, periodically transmit telemetry data. These mixed traffic patterns facilitate the analysis of MQTT performance and security under realistic attack conditions. Being able to control the number of both legitimate and illicit IoT nodes enables a precise control over the simulated attack, especially in terms of duration and frequency. The second template considers a malicious actor cloaking information in MQTT traffic, specifically in MQTT topic names, either through `case manipulation` or `ID modulation` [8]. The tool allows the configuration of multiple parameters, such as topic names, Quality of Service (QoS) for message publishing, message payload, and to customize the IoT nodes, either publishers or subscribers, according to the MQTT interaction pattern.

The proposed traffic generation framework is implemented in Python, and leverages libraries such as Pandas, Numpy, and the Eclipse Paho MQTT client[1] for efficient data handling and network communications. The generator provides two modes of operation, namely, a manual configuration mode using a `.csv` configuration file for synthetic traffic generation, and an empirical distribution mode that replays previously captured traffic traces from `.pcap` files. This adaptability makes our tool a valuable asset for testing detection systems, refining anomaly detection algorithms, and advancing security protocols within MQTT-based IoT networks, under conditions that closely resemble real-world threats.

3.2. SLMs for Automatic Generation of Test Cases

Manipulation of MQTT topic names is used by attackers willing to establish a covert communication between two (or more) clients sharing the same MQTT broker. By taking advantage of the global "visibility" of the MQTT topics to all the connected devices, a malicious IoT device could encode secret data by altering the case sensitivity of a topic name: publishing a message on a topic composed of only lowercase letters signals the bit `1`, whereas sending a message on a topic containing an uppercase letter signals the bit `0`. We point out that such an attack model requires a suitable "visibility" over topics handled by the MQTT broker. In fact, properly configured brokers or as-a-Service deployments (e.g., based on AWS) that enforce access control policies might limit the number of topics that can be used to encode the secret data. Nevertheless, weak credentials or poor configuration choices often plagues MQTT brokers, which are exposed over the Internet without a proper security degree [25].

In this context, to simulate an attacker modifying the topic list of a targeted MQTT broker, we developed an AI framework based on the Bidirectional Encoder Representations from Transformers (BERT) [26]. In essence, this framework employs a transformer-based architecture that comprehensively understands word context within a sentence by analyzing both preceding and subsequent words. In more detail, the BERT consists of a stack of transformer encoder layers, each comprising multiple self-attention "heads". This bidirectional approach captures linguistic nuances more effectively than traditional unidirectional models.

The pre-training process includes two main steps, i.e., Word Masking and Next Sentence Prediction. In the masking step, a certain percentage of words within a sentence is either masked or randomly substituted. The BERT model is subsequently trained to predict these masked words by analyzing the surrounding context, which includes the words that precede and follow the masked word. This task is designed to help the model grasp the contextual relationships between words in a sentence. In the prediction step, the BERT model undergoes fine-tuning to identify the relationships between two consecutive sentences. This involves generating negative examples by replacing the second sentence with a random one. The model is then trained to differentiate between positive pairs (authentic consecutive sentences) and negative pairs (where the second sentence has been replaced).

For generating synthetic MQTT topics allowing the emulation of an attacker cloaking information in malicious/counterfeit topics, we employed a compact pre-trained variant of BERT[2] tailored for historical

[1] https://pandas.pydata.org, https://numpy.org, https://pypi.org/project/paho-mqtt/
[2] Hugging Face Model Hub: https://huggingface.co/dbmdz/bert-tiny-historic-multilingual-cased [Last Accessed: December 2024]

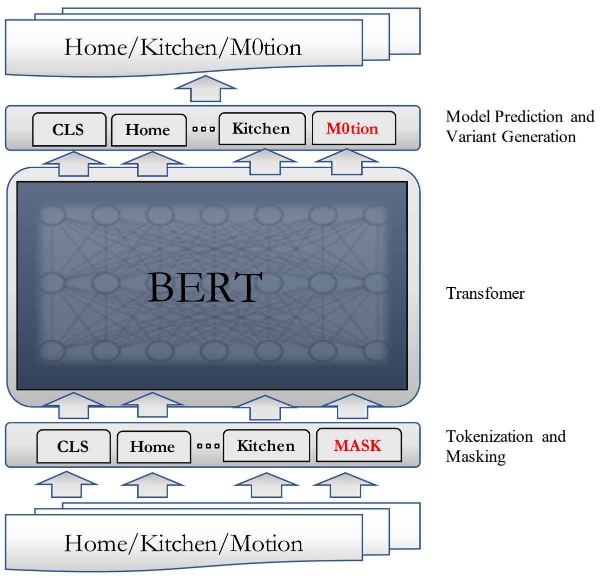


Figure 1: Learning/Application flow on an MQTT topic example.

and multilingual text processing. This model is optimized for handling multilingual data, making it especially effective at producing diverse and meaningful topic variations across different languages while maintaining computational efficiency.

The MQTT topic generator is based on a masked language modeling technique. The main steps involved in this process are:

1. **Tokenization**: the input MQTT topic is tokenized by the BERT tokenizer. The text is then converted into a sequence of tokens, which are mapped to their corresponding token IDs, e.g., from `Home/Kitchen/Motion` to `Home`, `Kitchen`, and `Motion`.
2. **Masking Tokens**: some tokens in the sequence are randomly selected and replaced with a special `[MASK]` token. The masking probability is set to 15%, as discussed in [27].
3. **Model Prediction**: the masked sequence feeds the BERT model, which predicts the original tokens that were replaced by the `[MASK]` tokens. These predictions are influenced by surrounding tokens creating a "context".
4. **Variant Generation**: the predicted tokens replace masked ones to generate a variant of the legitimate topic.

Figure 1 shows the components implemented for generating a test MQTT topic via the BERT framework. As depicted, the model takes as input a legitimate topic, which is properly tokenized. In the example, we consider the generation of a variant targeting the last level of the topic hierarchy. Masking is performed only during the learning phase, while variant generation is performed during the application phase. The `CLS` token is used as a delimiter. We point out that the generation process has been designed to create MQTT topics very similar to the ones provided by the training distribution. In fact, not to make the covert communication attempt obvious, the malicious actor is expected to use as the carrier new topics (or variations of a pre-existent entry), which are very similar or highly-correlated with the names handled by the targeted MQTT broker (see, e.g., [28]).

4. Numerical Results

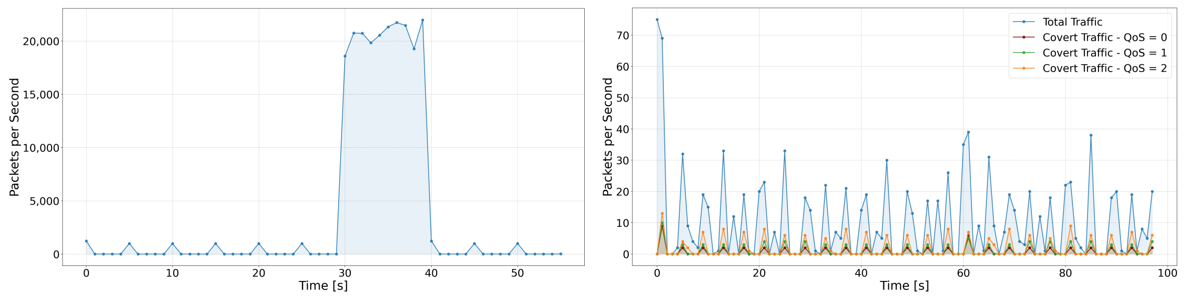
In this section, we present the experimental results. Specifically, Section 4.1 demonstrates the effectiveness of the approach to generate two different types of malicious network conditions, whereas Section 4.2 presents how pseudo-fuzz via SLMs can support the mitigation of covert channels targeting MQTT topics.

4.1. Generation of Network Threats

To demonstrate the versatility of our traffic generation approach, we setup two distinct network threats, i.e., a small DoS attack that tries to overwhelm an MQTT broker by flooding a specific topic name, and a covert communication channel implemented through topic manipulation.

The first experiment refers to a smart room environment that includes a broker that receives messages by two types of sensors: a temperature sensor that publishes a legitimate message every 5 seconds and 500 compromised humidity sensors, each publishing a message every 0.05 seconds with a QoS level set to 2. For implementing this offensive scenario, we deploy a Mosquitto broker version 2.0.18 on a virtualized Raspberry Pi with an ARM1176 processor and 256 MB RAM that models a setup commonly used in IoT home appliances. Instead, the traffic generators run on a workstation with an Intel Core i5-2500, 8 GB RAM, and 1 TB RAID 1 HDDs.

Figure 2(a) illustrates the throughput of the network, expressed in number of packets per second, during the simulated DoS attack. As shown, the high-frequency packet flooding, sustained over a



(a) DoS attack

(b) Covert channels

Figure 2: Network throughput measured during a small DoS attack implemented by flooding an MQTT broker using a specific topic, and during a covert communication implemented by manipulating MQTT topics.

10-second interval, causes a rapid increase in the network throughput, whose peaks exceed $20,000$ packets/second, that is, approximately 13 Mbps. At the end of the attack, the load goes back to its normal levels. Even though the attack is not able to saturate the network bandwidth, it affects the performance of the packets. For example, the mean latency of packet delivery raises sharply from 2.88 seconds under normal traffic conditions to 51.09 seconds during the attack, clearly highlighting the vulnerability of MQTT-based IoT networks to flooding attacks.

For the second experiment, i.e., the covert communication channel, we consider a smart home environment consisting of ten legitimate sensors (i.e., two subscribers and eight publishers) and three compromised IoT devices. These devices establish covert communication channels by exploiting the MQTT topic name field, each publishing messages every 4 seconds with a different QoS level (i.e., 0, 1, and 2). The use of various QoS levels allows the evaluation of the impact of the attack also in terms of the additional traffic being generated. We recall that higher QoS levels require MQTT to produce acknowledgments, which contribute to an increased packet volume and resource consumption.

Figure 2(b) depicts the network throughput over a 100-second interval, namely, the total traffic in blue, alongside the traffic from each compromised device, represented in red, green, and orange. During the observation interval, the peak throughput for covert traffic alone reaches approximately 8 Kbps, compared to a peak of approximately 48 Kbps for the total traffic. These results illustrate how higher QoS levels lead to increased packet exchanges due to the different acknowledgment schemes of the MQTT protocol, thus impacting both the overall throughput and the detectability of the covert channels.

Figure 3 further details the behavior of one of the compromised devices of the scenario previously described, namely, the device using a QoS level equal to 1. The flow shows the exchanges between the device and the MQTT broker and clearly illustrates the process that involves a single PUBACK acknowledgment per message, typical of this QoS level. Notably, the covert communication mechanism embeds the secret information within the last level of the topic name (e.g., `Home/Kitchen/Humidity`)

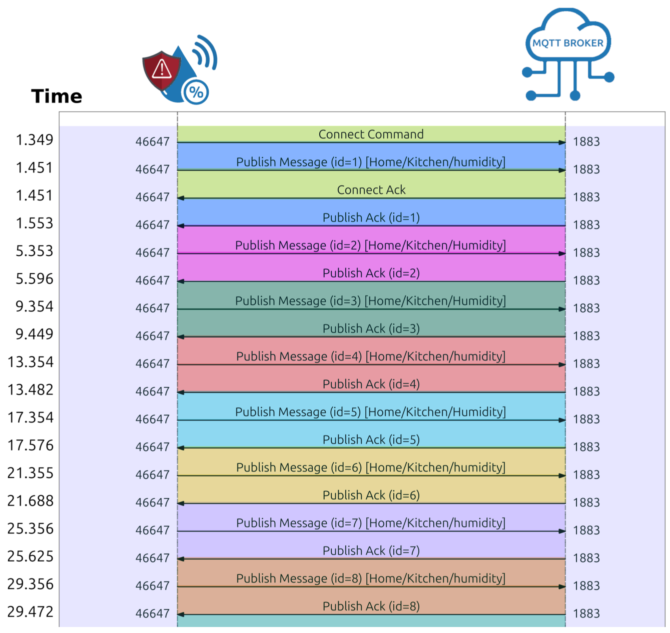

Time
MQTT BROKER
1.349 46647 Connect Command 1883
1.451 46647 Publish Message (id=1) [Home/Kitchen/humidity] 1883
1.451 46647 Connect Ack 1883
1.553 46647 Publish Ack (id=1) 1883
5.353 46647 Publish Message (id=2) [Home/Kitchen/Humidity] 1883
5.596 46647 Publish Ack (id=2) 1883
9.354 46647 Publish Message (id=3) [Home/Kitchen/Humidity] 1883
9.449 46647 Publish Ack (id=3) 1883
13.354 46647 Publish Message (id=4) [Home/Kitchen/humidity] 1883
13.482 46647 Publish Ack (id=4) 1883
17.354 46647 Publish Message (id=5) [Home/Kitchen/Humidity] 1883
17.576 46647 Publish Ack (id=5) 1883
21.355 46647 Publish Message (id=6) [Home/Kitchen/humidity] 1883
21.688 46647 Publish Ack (id=6) 1883
25.356 46647 Publish Message (id=7) [Home/Kitchen/humidity] 1883
25.625 46647 Publish Ack (id=7) 1883
29.356 46647 Publish Message (id=8) [Home/Kitchen/humidity] 1883
29.472 46647 Publish Ack (id=8) 1883

Figure 3: Flow diagram of the exchanges between a tampered node establishing a covert channel via the topic name case pattern and the MQTT broker. The device uses a QoS level equal to 1. Time is measured in seconds.

by utilizing the case pattern of the first letter as the container for the hidden data.

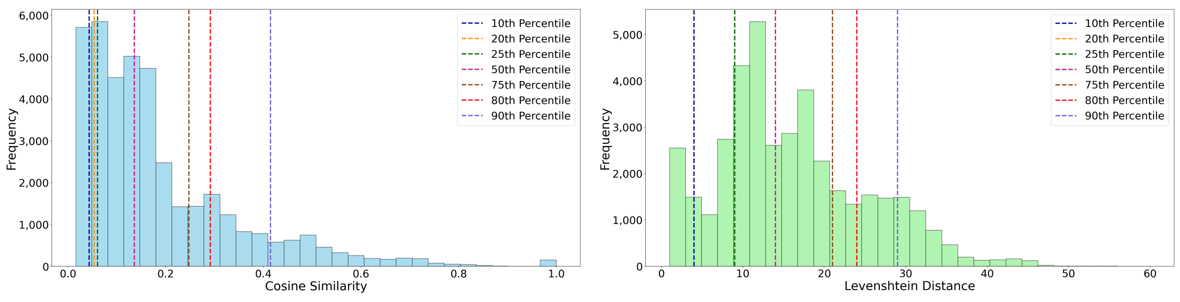
4.2. Generation of MQTT Topics for Covert Communications

As discussed in Section 3.2, an attacker could abuse MQTT topics to conceal information and build some form of covert communications. In this vein, being able to tune ad-hoc detection metrics or anticipate possible offensive hiding strategies are core tasks. To illustrate the effectiveness of SLMs for generating test cases, we consider a set of 98 distinct broker configurations, each containing 30 unique topic names representative of diverse, legitimate scenarios across multiple languages, including English, Spanish, and German. These topic names are obtained by querying two publicly accessible MQTT test servers[3] hosted by the Eclipse Foundation. The data collection process leverages a custom Python script built with the Paho MQTT client library, enabling the connection to the test servers and the subscription to all available topics using the # multi-level wildcard. From the total number of $4,918$ retrieved topics, a subset of $1,978$ entries is randomly extracted to train the SLM used for generating the topic variations.

To assess the SLM, we setup $5,880$ test cases equally distributed between real topics and SLM-generated topics that simulate an attacker crafting a covert, malicious variant closely resembling a legitimate topic. For each of the 98 brokers used in the experiments, we sequentially exclude one of its 30 topics at a time and we assess these topics with respect to the legitimate topic as well as a counterfeit variant generated by the SLM. Because of the string-based nature of MQTT topic names, these assessments are based on metrics commonly employed for text analysis, e.g., entropy, compressibility, Levenshtein distance, and cosine similarity [29, 30]. Moreover, such metrics demonstrated to be a prime effective mechanism to identify the presence of a threat actor trying to hide data through the manipulations of topics, even if they require further tweaking [27]. To classify topic names as legitimate or counterfeit, each metric is assigned a threshold value. These thresholds are determined based on a given percentile derived from the distribution of the metric, computed across all topic pairs for the considered brokers.

Figure 4 shows the distributions of the cosine similarity and Levenshtein distance. As can be seen from Figure 4(a), the distribution of the cosine similarity is highly skewed towards small values. For example, its median is equal to 0.14, whereas the first and third quartiles are equal to 0.06 and 0.25,

[3]https://test.mosquitto.org/, https://mqtt.eclipseprojects.io/



(a) Cosine similarity distribution

(b) Levenshtein distance distribution

Figure 4: Distribution of the cosine similarity scores (a) and of the Levenshtein distances (b) across all topic pairs on all brokers. The $10th$, $20th$, $25th$, $50th$, $75th$, $80th$, and $90th$ percentiles are highlighted.

respectively. We recall that cosine similarity quantifies the similarity of two vectors by calculating the cosine of the angle between them, thus small values (close to 0) indicate high dissimilarity, suggesting potential covert or counterfeit topics. On the contrary, high values (close to 1) indicate an alignment with legitimate topics. In addition, the densely clustered lower percentiles suggest that thresholds based on these values might effectively detect counterfeit topic names by filtering out topics with minimal similarity with respect to the baseline. In contrast, higher percentiles exhibit greater sparsity, indicating strongly aligned, legitimate topics. Instead, the distribution of Levenshtein distances (depicted in Figure 4(b)) is rather evenly spread. The distances are in the range $[1–60]$, and the first and third quartiles are equal to 9 and 21, respectively. We recall that the Levenshtein distance measures the minimum number of single-character edits needed to transform one string into another, thus it helps to identify related or similar topics: smaller values indicate an alignment with expected patterns of the considered broker, whereas higher values might signal counterfeit topics with significant deviations. In this case, higher percentiles potentially serve as effective thresholds for detecting that topics have been altered to conceal data, given that larger values represent greater divergence from legitimate topic names. Results have shown that the cosine similarity achieves the best performance independently of the value of the percentile, leading to the proposal of a detection method solely based on cosine similarity. Specifically, according to the observed distribution, four percentiles, i.e., $10th$, $20th$, $25th$, and $50th$, are selected to explore the balance between precision and accuracy. Each threshold percentile is used to classify topic names, labeling a topic as counterfeit if its cosine similarity falls below the given threshold.

Table 1 presents the classification performance assessed, as a function of the percentile, in terms of accuracy, precision, recall, and F1 score. Results indicate that lower thresholds enhance performance by

Percentile	**Accuracy**	**Precision**	**Recall**	**F1 Score**
$50th$	0.869	0.794	0.997	0.884
$25th$	0.961	0.936	0.989	0.962
$20th$	0.967	0.949	0.988	0.968
$10th$	0.985	0.983	0.987	0.985

Table 1
Performance achieved by the proposed implementation of the cosine similarity calculated across all the brokers with different percentile thresholds.

applying stricter criteria for identifying dissimilar topics, thereby reducing false negatives. At the $50th$ percentile threshold, accuracy is 0.869, with high recall (0.997) but lower precision (0.794), suggesting that more legitimate topics are misclassified. As the threshold decreases to $20th$ and $10th$, precision and F1 scores peak at 0.968 and 0.985, respectively. This demonstrates that lower thresholds improve the ability of the model to accurately classify legitimate topics, reducing false positives and achieving more precise detection of covert communications.

This percentile-based approach is particularly effective because it enables adaptable thresholding, leveraging the distribution of similarity scores on each broker to capture variations between legitimate and covert topics across diverse broker scenarios. Detection accuracy can be further improved by combining cosine similarity with other text analysis metrics. In this case, the complementary percentile is applied as a threshold, with higher metric values indicating counterfeit topics, while lower values confirm legitimacy. Although these metrics can contribute to detection improvements in specific cases, results indicate that their overall impact on accuracy is limited. For example, combining cosine similarity with Levenshtein distance yields a peak accuracy of 0.978 reached at the $90th$ percentile. Moreover, the low performance of compressibility, Levenshtein distance, and entropy in our model leads to an overall accuracy of 0.569. These results further support the advantages of relying solely on cosine similarity, which achieves robust performance without the complexity of implementing additional metrics.

To address a limitation of the proposed cosine similarity-based detection method, i.e., the potential misclassification of valid topics containing synonyms inadvertently introduced by legitimate users, we propose an enhanced approach. We still evaluate the cosine similarity of incoming topics against the existing topics on the broker, but we incorporate a semantic equivalence analysis leveraging Word2Vec embeddings. This integration accounts for word-level variations within incoming topics prior to classification, thereby reducing false positives by effectively recognizing synonyms. Results demonstrate that such an approach ensures robustness across diverse broker scenarios while maintaining a high level of detection accuracy. For instance, at the $20th$ percentile threshold, accuracy is equal to 0.951, that is, only slightly smaller than the value obtained without introducing synonyms. Similarly, precision and recall are equal to 0.942 and 0.959, respectively.

5. Conclusions and Future Work

In this paper, we presented an approach to improve the security of IoT ecosystems through the automatic generation of test data. To this aim, we introduced a tool for the creation of traffic conditions based on real-world threat templates, e.g., DoS and covert channels hidden within the MQTT protocol. We also discussed a framework taking advantage of an SLM to generate realistic MQTT topics especially to assess and improve possible countermeasures. As shown, both approaches can be effectively used to conduct a wide array of investigations, such as implementing pseudo-fuzzing approaches against detection metrics.

Future works aim at refining the proposed tools, e.g., by improving the predefined offensive templates. For instance, we are working towards hiding mechanisms based on topic wildcard to implement covert communications schemes among multiple endpoints as well as channels with an increased stealthiness. Another relevant part of our ongoing research concerns the creation of an integrated framework able to operate simultaneously at different layers of the protocol stack. For instance, this framework could be used to create test scenarios for assessing the robustness against advanced attack schemes, e.g., multi-stage loading mechanisms using both the application messages and protocol data units to exchange information with a remote command & control facility.

Acknowledgments

This research was partially funded by Project RAISE – Robotics and AI for Socio-economic Empowerment (ECS00000035), Project SERICS – SEcurity and RIghts In the CyberSpace (PE00000014), and by Project STRIVE/URAN – Advanced Approaches for Transitions in Urban Environments.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] G. Mylonas, A. Kalogeras, G. Kalogeras, C. Anagnostopoulos, C. Alexakos, L. Muñoz, Digital Twins From Smart Manufacturing to Smart Cities: A Survey, IEEE Access 9 (2021) 143222–143249. doi:10.1109/ACCESS.2021.3120843.

[2] A. E. Omolara, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, W. H. Alshoura, H. Arshad, The Internet of Things Security: A Survey Encompassing Unexplored Areas and New Insights, Computers & Security 112 (2022) 102494.

[3] B. B. Gupta, M. Quamara, An Overview of Internet of Things (IoT): Architectural Aspects, Challenges, and Protocols, Concurrency and Computation: Practice and Experience 32 (2020) e4946.

[4] A. J. Hintaw, S. Manickam, M. F. Aboalmaaly, S. Karuppayah, MQTT Vulnerabilities, Attack Vectors and Solutions in the Internet of Things (IoT), IETE Journal of Research 69 (2023) 3368–3397.

[5] M. M. Raikar, S. Meena, Vulnerability Assessment of MQTT Protocol in Internet of Things (IoT), in: Proceedings of the 2nd International Conference on Secure Cyber Computing and Communications, ICSCCC, IEEE, 2021, pp. 535–540.

[6] L. Caviglione, M. Choraś, I. Corona, A. Janicki, W. Mazurczyk, M. Pawlicki, K. Wasielewska, Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection, IEEE Access 9 (2020) 5371–5396.

[7] F. Strachanski, D. Petrov, T. Schmidbauer, S. Wendzel, A Comprehensive Pattern-based Overview of Stegomalware, in: Proceedings of the 19th International Conference on Availability, Reliability and Security, ARES ’24, Association for Computing Machinery, 2024.

[8] A. Mileva, A. Velinov, L. Hartmann, S. Wendzel, W. Mazurczyk, Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels, Computers & Security 104 (2021) 102207.

[9] H. Kim, A. Ahmad, J. Hwang, H. Baqa, F. Le Gall, M. A. R. Ortega, J. Song, IoT-TaaS: Towards a Prospective IoT Testing Framework, IEEE Access 6 (2018) 15480–15493.

[10] A. N. Duc, R. Jabangwe, P. Paul, P. Abrahamsson, Security Challenges in IoT Development: a Software Engineering Perspective, in: Proceedings of the XP2017 Scientific Workshops, XP’17, Association for Computing Machinery, 2017.

[11] I. Nadir, H. Mahmood, G. Asadullah, A Taxonomy of IoT Firmware Security and Principal Firmware Analysis Techniques, International Journal of Critical Infrastructure Protection 38 (2022) 100552.

[12] S. Ghazanfar, F. Hussain, A. Ur Rehman, U. Fayyaz, F. Shahzad, G. Shah, IoT-Flock: An Open-source Framework for IoT Traffic Generation, in: Proceedings of the 2020 International Conference on Emerging Trends in Smart Technologies, ICETST, 2020.

[13] H. Nguyen-An, T. Silverston, T. Yamazaki, T. Miyoshi, Generating IoT Traffic in Smart Home Environment, in: Proceedings of the IEEE 17th Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2020.

[14] S. Hui, H. Wang, Z. Wang, X. Yang, Z. Liu, D. Jin, Y. Li, Knowledge Enhanced GAN for IoT Traffic Generation, in: Proceedings of the ACM Web Conference, 2022, pp. 3336–3346.

[15] R. Li, Q. Li, Q. Zou, D. Zhao, X. Zeng, Y. Huang, Y. Jiang, F. Lyu, G. Ormazabal, A. Singh, et al., IoTGemini: Modeling IoT Network Behaviors for Synthetic Traffic Generation, IEEE Transactions on Mobile Computing 23 (2024) 13240–13257.

[16] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, E. Cambiaso, MQTTset, a New Dataset for Machine Learning Techniques on MQTT, Sensors 20 (2020) 6578.

[17] F. Iglesias, F. Meghdouri, R. Annessi, T. Zseby, CCgen: Injecting Covert Channels into Network Traffic, Security and Communication Networks 2022 (2022) 2254959.

[18] M. Zuppelli, L. Caviglione, pcapStego: A Tool for Generating Traffic Traces for Experimenting with Network Covert Channels, in: Proceedings of the 16th International Conference on Availability, Reliability and Security, ARES ’21, Association for Computing Machinery, 2021.

[19] K. Lamshöft, T. Neubert, C. Krätzer, C. Vielhauer, J. Dittmann, Information Hiding in Cyber Physical Systems: Challenges for Embedding, Retrieval and Detection Using Sensor Data of the SWAT Dataset, in: Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia

Security, IH&MMSec, 2021, pp. 113–124.

[20] L. Caviglione, W. Mazurczyk, You Can't Do That on Protocols Anymore: Analysis of Covert Channels in IETF Standards, IEEE Network 38 (2024) 255–263. doi:10.1109/MNET.2024.3352411.

[21] M. A. Elsadig, A. Gafar, Covert Channel Detection: Machine Learning Approaches, IEEE Access 10 (2022) 38391–38405.

[22] X. Zhu, S. Wen, S. Camtepe, Y. Xiang, Fuzzing: a Survey for Roadmap, ACM Computing Surveys 54 (2022).

[23] K. Hölk, W. Mazurczyk, M. Zuppelli, L. Caviglione, Investigating HTTP Covert Channels Through Fuzz Testing, in: Proceedings of the 19th International Conference on Availability, Reliability and Security, ARES '24, Association for Computing Machinery, 2024.

[24] Y. Wang, P. Jia, L. Liu, C. Huang, Z. Liu, A Systematic Review of Fuzzing Based on Machine Learning Techniques, PloS one 15 (2020) e0237749.

[25] Rachit, S. Bhatt, P. R. Ragiri, Security Trends in Internet of Things: A Survey, SN Applied Sciences 3 (2021) 1–14.

[26] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics, volume 1 of *NAACL-HLT*, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[27] C. Cespi Polisiani, M. Zuppelli, M. C. Calzarossa, L. Caviglione, M. Guarascio, Mitigation of Covert Communications in MQTT Topics Through Small Language Models, in: Proceedings of the 32nd International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication System, MASCOTS, IEEE, 2024.

[28] A. Velinov, A. Mileva, S. Wendzel, W. Mazurczyk, Covert Channels in the MQTT-based Internet of Things, IEEE Access 7 (2019) 161899–161915.

[29] C. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[30] W. Stallings, Cryptography and Network Security Principles and Practice, 8th Edition, Pearson Education, 2023.