Analysis of Misconfigured IoT MQTT Deployments and a Lightweight Exposure Detection System

Seyed Ali Ghazi Asgar
Texas A&M University
alighazi@tamu.edu

Narasimha Reddy
Texas A&M University
reddy@tamu.edu

***Abstract*—The Internet of Things (IoT) is experiencing exponential growth, with projections estimating over 29 billion devices by 2027. These devices often have limited resources, necessitating the use of lightweight communication protocols. MQTT is a widely used protocol in the IoT domain, but defective security configurations can pose significant risks for the users. In this work, we classify the most commonly used open-source IoT applications that utilize MQTT as their primary communication protocol and evaluate the associated attack scenarios. Our analysis shows that home automation IoT applications have the highest number of exposed devices. In addition, our examination suggests that tracking applications are prone to higher risks as the normalized percentage of exposed devices for this category is 6.85% while only 2.91% of home automation devices are exposed. To tackle these issues, we developed a lightweight, open-source exposure detection system suitable for both computer-based clients and ESP32 microcontrollers. This system warns the users of compromised MQTT broker which enhances the overall security in IoT deployments without any significant overhead.**

I. Introduction

The Internet of Things (IoT) has varying definitions as it involves integrating devices into different facets of life, from cities, cars, roads to homes, and personal devices. These devices interact and exchange data, which is then processed to perform specific tasks. This ecosystem is expected to grow significantly and potentially surpass the mobile phone market [14], [1], [12]. According to [23], there are currently more than 16 billion active IoT devices, with projections indicating this number will exceed 29 billion by 2027.

IoT devices commonly encompass sensors, actuators, or have a portable form factor, often constrained by resource limitations such as power availability, memory constraints, and computational capacity [28], [25]. To address these challenges, the selection of an appropriate protocol to exchange information becomes important.

Extensible Messaging and Presence Protocol(XMPP), Hypertext Transfer Protocol(HTTP), Advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (CoAP), and Message Queuing Telemetry Transport (MQTT) are the most widely used protocols [3], [17]. HTTP and CoAP have a significant overhead because of their text-based structure and using RESTful approach (which requires constant polling). XMPP, AMQP, and MQTT rely on message-oriented broker/client architecture, enabling asynchronous communication, however, XMPP requires significant memory space due to XML data processing and AMQP is also more suitable for server-to-server communication. XMPP and AMQP both require authentication processes. On the other hand, MQTT is specially designed for constrained devices which require minimal resources of bandwidth, memory and processing power [17] and password authentication can be bypassed as well.

Securing MQTT protocol is really important as many IoT devices use these protocol for communicating. Username and password based authentication is optional for MQTT, so, many people might ignore providing an appropriate username and password for the communication leading to exposure of their messages to everyone on the network. In addition, even if an appropriate username and password authentication is used, since the messages are not encrypted and are transported over TCP protocol openly, eavesdropping, man-in-the middle (MITM) attacks, and password stealing are probable attack scenarios [9]. MQTT also supports TLS/SSL for extra security, but not all the IoT devices have the capabilities to use TLS/SSL. Moreover, managing certificates and keys between devices is another issue with this solution. In addition to that, SSL/TLS also suffers from attacks such as BEAST, and Heartbleed attack [29]. In [29], authors proposed a secure MQTT (SMQTT) protocol which provides a more robust security mechanism for IoT communications. By using Key/Ciphertext Policy-Attribute Based Encryption (CP/KP-ABE) with lightweight Elliptic Curve Cryptography (ECC). Researchers in [9] analyzed security vulnerabilities in the MQTT protocol using the Shodan API and an experimental environment using a Raspberry Pi as a MQTT broker, with Python clients acting as publishers and subscribers. They identified security issues at packet and topic level and implemented countermeasures including certificates, secure sockets, payload encryption, and ACLs. Moreover authors in [27] propose a security framework for MQTT, called AugMQTT, incorporating the AugPAKE [26] protocol to improve security without the need for certificate validation and revocation checks. AugMQTT uses AugPAKE to secure session keys and

Workshop on Security and Privacy in Standardized IoT (SDIoTSec) 2025
24 February 2025, San Diego, CA, USA
ISBN 979-8-9919276-6-6
https://dx.doi.org/10.14722/sdiotsec.2025.24xxx
www.ndss-symposium.org

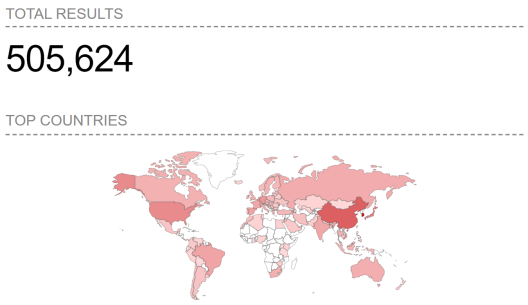

Fig. 1. There are more than 500,000 open devices on the internet using MQTT protocol without any password and username according to Shodan using the search term "port:1883 MQTT Connection Code: 0".

ensure message confidentiality by eliminating complex Public-Key Infrastructures (PKI) requirements.

Although researchers have come up with lightweight, scalable more secure encryption methods for MQTT [29], [20], [24], [27], these methods are not still widely used by the community. A search for the devices that use MQTT protocol on Shodan shows more than 500,000 devices over the internet with their MQTT port (1883) open without any security measures (see Fig. 1). Hence, it is necessary to assess the implications of insecure MQTT implementations and publicly exposed MQTT devices.

In this work we investigate the most commonly used open source applications over the MQTT protocol and evaluate the risk factors associated with each of them in case of an attack. To mitigate the public exposure of MQTT devices and messages, we will provide a framework for alerting the user of accidental exposure to the internet by the well-known security search engines. It this work we focus on no-authentication misconfiguration and other misconfigurations such as access control list, rate limits, and etc, can be explored in future works. The contributions of this paper are as follows:

- Investigating top open source repositories in the IoT domain that use MQTT as their communication protocol
- Assessing the number of misconfigured and exposed IoT devices over the internet.
- Classifying each of the repositories based on their applications.
- Evaluating attack scenarios and risk associated with each of the classes.
- Developing an exposure detection system based on search engines like Shodan.
- Developing the exposure detection system for both microcontrollers (ESP32) and personal computers (PC) using Python.
- Dockerizing the developed application for easy deployment and scalability.

The rest of the paper is organized as follows. In section I-A, we will provide a brief overview of MQTT protocol. Section II discusses our methodology for selecting and classifying open source repositories in IoT domain. Section III discusses the risk associated with each class of the applications. Then we will provide our proposed solution and a real-world experiment of our system in sections IV and V. Finally, in section VII we will summarize our work.

A. Structure of MQTT protocol

MQTT is a lightweight publish/subscribe protocol that is built on top of TCP/IP for machine to machine (M2M) communication [18]. The MQTT protocol consists of two main components, clients and a broker. Clients can interact with the broker through two main functions which are publishing and subscribing. Clients can send messages to the broker on a topic using publish command. In addition, whenever clients need to receive messages on a specific topic, they can subscribe to that topic through the broker [31].

Clients can use wildcards such as "+" and "#" for subscribing to the topics. Single-level wildcard "+" is used for exactly one level topic subscription. For instance, when a client subscribes to the "school/+/humidity" topic, it will also receive information from

- "school/office/humidity"
- "school/class1/humidity",
- and "school/lunchroom/humidity"

Multi-level wild card "#" is used for subscribing to a wide range of topics. As an example, when the client subscribes to the "school/#" it will receive messages from

- "school/office/temperature"
- "school/lunchroom/lights",
- "school/office/humidity",

and any topic that begins with "school/" [11].

When a client tries to connect to the broker, the broker will respond with a connection code. Connection code '0' implies that the client successfully connected to the server and connection codes from '1' to '5' are the results of unacceptable version, identifier rejected, server unavailable, bad username or password and not authorized error respectively [3], [19].

II. Classification Method

A. Selecting top open source repositories

In order to find the most commonly used open source applications that utilize MQTT protocol we followed the flowchart given in the Fig. 2. We first searched for the term MQTT in the GitHub and then sorted the results based on the stars of each repository. In the next step, we went through them one by one and eliminated the repositories that were offering MQTT brokers such as Eclipse, Mosquitto, HiveMQ, RabbitMQ, EMQX and etc since we are more interested in evaluating the applications of MQTT rather than the MQTT broker itself. In the next step we evaluated the repository and checked if it is used in IoT area or not. Afterward, we searched for the application on Shodan and reported the number of found records and ignored the repositories that had less than 900 stars because they were too customized and not widely used by the community. The threshold for the number of Shodan records was set to 10 and repositories with less than 10 records were eliminated.

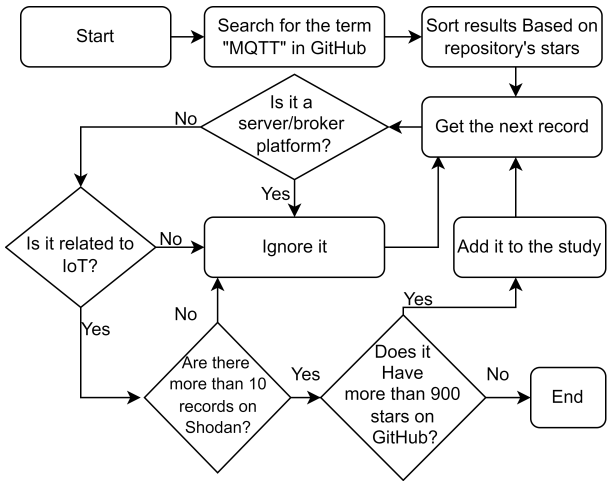

Fig. 2. The flowchart of the method used for selecting mostly used open source repositories that utilize MQTT protocol for IoT applications.

B. Classifying the repositories

After selecting the repositories, we classified each of them based on their applications. Based on our assessment, these repositories fall in the following groups: home automation (such as lights, heaters, air conditioners, air quality sensors, presence sensors, flow meters, etc), Energy monitoring (electrical vehicle(EV) charging, solar panels, or inverters), camera related, and tracking applications. Therefore, we classified and marked each of the repositories into corresponding groups as follows: HA for home automation, EM for energy monitoring, CM for camera related, and TR for device tracking repositories. Although some of theses libraries might reside in multiple different categories, we assigned them based on their primary application. We also provided our search terms on the Shodan search engine for querying the records in the Table I.

III. Investigating exposed IoT categories

A. Top repositories

Table II shows the top open source repositories that use MQTT protocol in different categories. We sort each of them based on the stars of each repository as mentioned in the Fig. 2. Afterward, we placed all of these 10 repositories into HA, CM, TR, and EM groups as we discussed in the previous section.

B. Attack scenarios

In this section we will analyze each class of the repositories and then will provide attack scenarios and discuss consequences of the exposure of them to the internet. We list the search terms utilized to query the records from Shodan in the Table I so it can be replicated in the future. We report on the number of records found on Shodan at the time of our work, but noticed that the number of records are increasing over time.

When a broker can be reached without authentication, an attacker can subscribe to the messages to compromise privacy. In addition, the attacker can send doctored messages with identical names to convey different/false information to subscribers.

TABLE I
Search term used for each repository to query the records from Shodan

Repository	search term
home-assistant	homeassistant port:1883
Tasmota	Tasmota port:1883
frigate	frigate
zigbee2mqtt	zigbee2mqtt port:1883
esphome	esphome port:1883
teslamate	teslamate port:1883
evcc	evcc port:1883
owntracks	owntracks port:1883
ESPresense	espresense port:1883
ahoy	ahoy port:1883

C. Home automation(HA) IoT repositories

A noticeable number of exposed devices are in the home automation category. IoT devices for home automation are widely used by the community and comprise a wide range of sensors such as smoke detectors, motion sensors, presence sensors, heat sensors, air conditioning system controllers, light sensors, leak sensors, window and door sensors [8], [16]. An attacker can create discomfort for the victim by toggling lights, or changing the air conditioner temperature to too hot or cold temperatures. In addition, some attacks can go beyond this point and even open the home's door through the home automation system [2], [21]. Another important point about Home automation system is that attackers can get the knowledge of the presence and the number of people currently inside the house and further exploit this information. For instance, ESPresense library is widely used for detecting the presence of a person inside the house which also can be used by malicious attacker to gain more knowledge about the victim. Home-assistant is a widely used IoT platform by

TABLE II
Top open source repositories that use MQTT protocol for IoT applications

Repository	Application	Stars	Shodan Records	Class
home-assistant	Home automation	69.3K	1,221	HA
Tasmota	IoT firmware for ESP MCU	21.6K	1,225	HA
frigate	Object Detection for IP cameras	15.2K	130	CM
zigbee2mqtt	A brdige to control Zigbee via MQTT	11.3K	690	HA
esphome	Home automation	7.9K	85	HA
teslamate	Data logger for Tesla car	5.4K	230	TR
evcc	Home and EV energy controller	2.7K	10	EM
owntracks	Mobile tracking application	1.3K	230	TR
ESPresense	Presence detection	1.3K	24	HA
ahoy	Logger for inverters	929	10	EM

the community. However, there are more than 1200 exposed devices on Shodan because of the misconfiguration by the users. Tasomato, and esphome are other repositories that use an ESP32 micro-controller as the MQTT broker and other sensors can exchange their data with this micro-controller over the MQTT protocol.

D. Camera related(CM) IoT repositories

Home security cameras are growing fast. Most cameras support live stream mode and motion detection mode [15], [7]. Frigate is an open source local network video recorder (NVR) library that is capable of real time object detection using image processing techniques and deep learning based image recognition. Another benefit of this library is that it can be merged with home-assistant easily. Based on the Shodan's records, there are more than 130 instances of exposed cameras using frigate application for monitoring their environment. As a result, people are openly publishing their cameras over the network which can lead to privacy and security issues.

E. Device tracking (TR) IoT repositories

Tracking smart devices such as tablets and mobile phones is beneficial when people lose their devices or when parents want to track their small children throughout the day for their safety. Owntracks is an open source application that can be used on both IOS and Android devices for tracking your device. However, there are more than 200 instances on Shodan where people are exposing their location over the internet without any need for authentication. Therefore, attackers can track victims using the exposed devices that are using MQTT protocol by transferring the information from the main broker. Teslamate is another open source library that can be installed on Tesla cars to log the information about the car's state such as mileage, doors and windows state, odometer, battery level, temperature, speed and elevation. Another interesting feature of Teslamate is that it can also log the car's location and send it to the server. Unfortunately, based on the Shodan's record there are more than 200 Tesla cars that do not have any authentication for their MQTT server and Teslamate is openly sending the information to the MQTT broker. An attacker can also subscribe to the MQTT broker easily and gather information about the location of the car which can lead to the leakage of private information such as work place, house, and favorite places [5], [22], [4]. We found that even some companies are publishing their data without any protection. For instance, during this work we encountered some records on Shodan related to a trucking company that was transmitting the locations of its trucks to the MQTT broker openly. Upon realizing this issue we contacted them and they were able to add authentication process to their system.

F. Energy management (EM) IoT repositories

Solar energy is a fast growing source of energy in the past decades [13] and photovoltaic (PV) technology advancements boosted the efficiency of solar panels. many governments also offer support to the solar energy users such as tax credits or tax exception which can decrease the cost of installing solar panels [6], [30]. Therefore, many people are attracted to PV panels due to a variety of reasons such as environmental and economic aspects. These solar panels are connected to an inverter which can turn the DC voltage into the AC voltage [10]. Most solar panel companies offer monitoring solutions for measuring voltage, current, wattage, and other smart metering parameters related to the panels or inverters.

Unfortunately, due to the security misconfigurations in the MQTT communication they are prone to cyber attacks. For instance 'evcc' and 'ahoy' repositories both offer logging system and smart metering for energy management, but because of the lack of authentication and security standards on their MQTT protocol they are exposed on Shodan which can lead to serious damage to their infrastructure. For instance, an attacker can change the charging limit of electrical devices or start/stop the charging process which can lead to discomfort for the owner.

G. Comparing different classes

To estimate the number of users for each application, we used the number of GitHub stars as a metric. As it is shown in the Fig. 3, home automation applications with 81% is placed at the top of the list. Camera based applications (11%) take second place, location tracking and energy management repositories are placed at the subsequent order, with 5% and 3% of the total share respectively.

Another interesting observation according to the Fig. 4 is that the percentage of the exposed TR applications are higher than other classes. The main reason for this exception can be explained by the fact that users must install the tracking applications on the moving devices such as smart phones or vehicles, therefore, the MQTT broker and the clients are not within the same network. As a result, if the MQTT broker is misconfigured and anonymous connections without username and password are allowed, the locations of the users are compromised. On the contrary, most of the HA, EM, and CM devices are stationary and located within the same network. In addition, it is not mandatory for the users to connect their

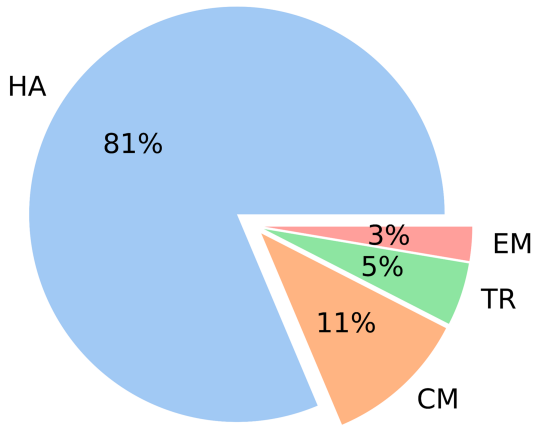

Fig. 3. Proportion of each repository class according to the number of GitHub stars.

MQTT broker to the internet. Hence, even if the MQTT connection itself is not secured these applications are prone to fewer cyber attacks as they are within a local network and not on the internet . However, for TR applications having internet connection is a must and cannot be skipped which results in higher percentage of compromised devices. Based on the given explanation and significant growth of electrical vehicles and smart cars, we can expect to have more cybersecurity threats for electrical vehicles specifically related to the third party and open source applications that can be installed by the users.

IV. Proposed Exposure Detection System(EDS)

Based on the evidence provided in the previous sections, it is clear that exposed IoT devices pose a great risk for users, especially when their IP addresses are available on cybersecurity search engines such as Shodan, Censys, Onyphe, Zoomeye, and Criminalip. To tackle this problem, we created an exposure detection system that can actively scan the search engines to check if the MQTT broker is exposed or not. For a proof of concept implementation of the method, we used Shodan search engine to automate the process using free APIs. As it is shown in the Fig. 5 the EDS constantly publishes its MAC address to the MQTT broker as a retained message. In MQTT protocol retained messages are stored inside the broker such that when a new client subscribes to the broker, broker sends the last available message to the new subscribers. Therefore, once a cybersecurity search engine such as Shodan connects to the MQTT broker and listens to all the topics, it will receive and store a topic named "MAC_ADDRESS" followed by the MAC address of the EDS. For instance, if the MAC address is "C8215DEE6D60", the search engine will save the "MAC_ADDRESS/C8215DEE6D60" as the record. In the next step, since EDS is constantly querying for its MAC address on the search engines, once it finds the record which matches its MAC address, it will send an alert topic to the

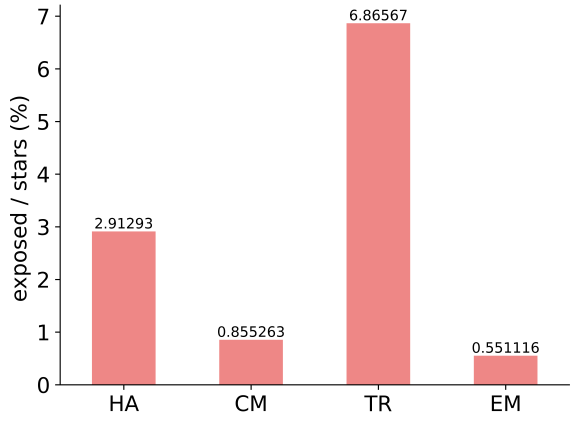

Fig. 4. The ratio of number of exposed devices to the number of stars for each class of IoT repositories. This metric indicates a normalized percentage of the exposed devices for each category.

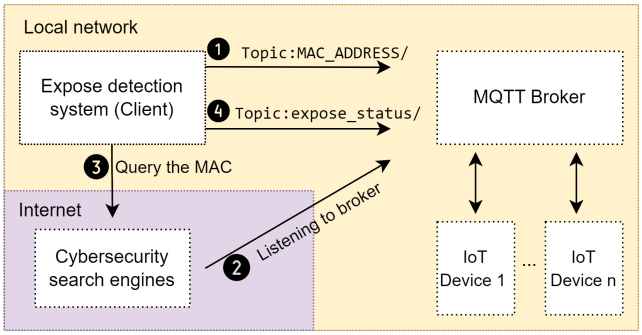

Fig. 5. The structure of exposure detection system. First the MAC address is sent to the MQTT broker using the "MAC_ADDRESS" topic. Second, if cybersecurity search engines connect to the local MQTT broker, they will get a copy of the "MAC_ADDRESS" topic and save it. Third, the exposure detection system queries for the MAC address periodically and once it finds a record, it will publish an alert topic to the MQTT broker in step four.

MQTT broker notifying the exposure status. In this work we implemented three different versions of this system, the first one is based on the Python language that is dockerized for easy deployment and scalability, and the second version is implemented for resource-constrained environments on top of an ESP32 micro-controller. In the third version, we unified the MQTT broker and the exposure detection system in a single docker-compose file such that once the broker is up and running, the EDS system is automatically deployed without any modification. We tested our EDS on both ESP32 and PC using different keywords on Shodan and it was capable of sending appropriate alerts to the main broker. We wrote 250 lines of C code for ESP32 and 230 lines of Python code for the computer-based EDS.

Our open source code and instructions to use the EDS are available at https://github.com/ali-ghazi78/mqtt-exposure-detection.

V. Experiment

To experiment the proposed method, we deployed a MQTT broker (Eclipse Mosquitto) on a server with the port 1883 open for remote connection. We also allowed any IP to connect to the server as many exposed servers turn on this feature. To save our record on Shodan and experiment our method, we used Shodan on-demand scanning tool. After using the Shodan's API we got our record on the Shodan's website. As it is shown in the Fig. 6, we published three main topics, the "/camera" topic which resembles an IoT camera data, the MAC address that we used as our unique identifier (UID) "C8215DEE6D60", and the last topic is the "expose_status" topic which sends the exposure status to the main broker. We also provided the logs of our EDS system in Fig. 7. Before exposing our device (Fig. 7(a)), the search result is returning "False" value and "expose_status" topic is also seeing "False" to the main Broker. However, Once the broker is exposed to the search engine (Fig. 7(b)), the "expose_status" starts sending "True" Value as well as the timestamp to the main broker.

Fig. 6. The shodan record that exposes our experimental MQTT server. Shodan also provides the retained topics that are used within the MQTT broker.

Therefore, the user can take appropriate actions and fix the vulnerabilities in the system.

This experiment validated our approach in detecting when an MQTT broker is exposed to the internet. This is a light-weight scheme that can be easily integrated into any MQTT application to provide exposure detection capability.

VI. Limitation

While this work investigated the IoT deployment that does not require any authentication, there could be other misconfiguration scenarios. For instance, we did not consider access control list, rate limit configuration, or other vulnerabilities. In addition, our EDS system focused on external exposure, the absence of authentication can also be a threat within the local network. Our EDS system is dependent on search engines and hence it might take some time before the misconfigurations are discovered add added to their databases.

VII. Conclusion

MQTT protocol is heavily used by the community for IoT applications. However, the surge in IoT devices causes security issues from lack of encryption and authentication. In this work we studied widely used open source repositories, classified them based on their applications and evaluated the risks associated with each of them. The number of exposed devices point to the urgent need for securing MQTT implementation to mitigate risks and cyber threats. To mitigate this problem, we developed an exposure detection system that scans these search engines to detect and alert if an MQTT broker is exposed. Our system is implemented in different versions with Python and ESP32 micro-controllers providing an automated exposure detection and alerting system. To further prove the effectiveness of our system, we also set up a Mosquitto MQTT broker and evaluated our EDS system in a real-world environment and showcased before and after exposure scenarios. Finally, our system proved to be effective in detecting a misconfigured MQTT broker and alerting the user.

Acknowledgment

This work is supported by part by the NSF Grant CCRI 2234972. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

References

[1] Antar Shaddad Abdul-Qawy, PJ Pramod, E Magesh, and T Srinivasulu. The internet of things (iot): An overview. *International Journal of Engineering Research and Applications*, 5(12):71–82, 2015.

[2] Cornelio Revelivan Aldawira, Handhika Wiratama Putra, Novita Hanafiah, Surya Surjarwo, Aswin Wibisurya, et al. Door security system for home monitoring based on esp32. *Procedia Computer Science*, 157:673–682, 2019.

[3] Syaiful Andy, Budi Rahardjo, and Bagus Hanindhito. Attack scenarios and security analysis of mqtt communication protocol in iot system. In *2017 4th international conference on electrical engineering, computer science and informatics (EECSI)*, pages 1–6. IEEE, 2017.

[4] Chih-Che Chang, Chia-Mei Chen, and Han-Wei Hsiao. Applying an iot analytics framework in east asia area. In *International Computer Symposium*, pages 421–433. Springer, 2022.

[5] David Colombo. How i got access to 25+ tesla's around the world. by accident. and curiosity, 2022.

[6] Christine Lasco Crago and Ilya Chernyakhovskiy. Are policy incentives for solar power effective? evidence from residential installations in the northeast. *Journal of Environmental Economics and Management*, 81:132–151, 2017.

[7] Saman Fatima, Naila Aiman Aslam, Iqra Tariq, and Nouman Ali. Home security and automation based on internet of things: a comprehensive review. In *IOP Conference Series: Materials Science and Engineering*, volume 899, page 012011. IOP Publishing, 2020.

[8] Ruben Figueiredo, Aníbal Alves, Vitor Monteiro, J Pinto, João Afonso, and José Afonso. Development and evaluation of smart home iot systems applied to hvac monitoring and control. *EAI Endorsed Transactions on Energy Web*, 8(34), 2020.

[9] MS Harsha, BM Bhavani, and KR Kundhavai. Analysis of vulnerabilities in mqtt security using shodan api and implementation of its countermeasures via authentication and acls. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2244–2250. IEEE, 2018.

[10] Ulrich Herrmann, Hans Georg Langer, and Heinz Van Der Broeck. Low cost dc to ac converter for photovoltaic power conversion in residential applications. In *Proceedings of IEEE Power Electronics Specialist Conference-PESC'93*, pages 588–594. IEEE, 1993.

[11] Ahmed J Hintaw, Selvakumar Manickam, Mohammed Faiz Aboalmaaly, and Shankar Karuppayah. Mqtt vulnerabilities, attack vectors and solutions in the internet of things (iot). *IETE Journal of Research*, 69(6):3368–3397, 2023.

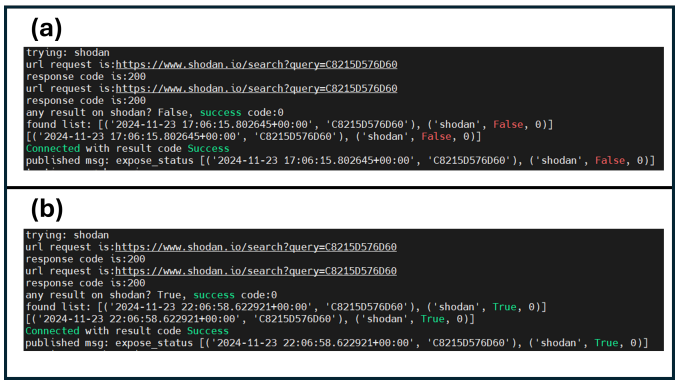

Fig. 7. (a) Shows the log of EDS system before being exposed. (b) Shows our alerting system after being exposed which sends the status of exposure to the main broker with the "True" Flag and timestamp.

[12] Falguni Jindal, Rishabh Jamar, and Prathamesh Churi. Future and challenges of internet of things. *Int. J. Comput. Sci. Inf. Technol*, 10(2):13–25, 2018.

[13] Daniel M Kammen. The rise of renewable energy. *Scientific American*, 295(3):84–93, 2006.

[14] Kaivan Karimi and Gary Atkinson. What the internet of things (iot) needs to become a reality. *White Paper, FreeScale and ARM*, pages 1–16, 2013.

[15] Jinyang Li, Zhenyu Li, Gareth Tyson, and Gaogang Xie. Your privilege gives your privacy away: An analysis of a home security camera service. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 387–396. IEEE, 2020.

[16] Bruno Mataloto, Joao C Ferreira, and Nuno Cruz. Lobems—iot for building and energy management systems. *Electronics*, 8(7):763, 2019.

[17] Aimaschana Niruntasukrat, Chavee Issariyapat, Panita Pongpaibool, Koonlachat Meesublak, Pramrudee Aiumsupucgul, and Anun Panya. Authorization mechanism for mqtt-based internet of things. In *2016 IEEE international conference on communications workshops (ICC)*, pages 290–295. IEEE, 2016.

[18] MQTT Organization. Mqtt: The standard for iot messaging. https: mqtt.org, 2022.

[19] Andrea Palmieri, Paolo Prem, Silvio Ranise, Umberto Morelli, and Tahir Ahmad. Mqttsa: A tool for automatically assisting the secure deployments of mqtt brokers. In *2019 IEEE World Congress on Services (SERVICES)*, volume 2642-939X, pages 47–53, 2019.

[20] Chang-Seop Park and Hye-Min Nam. Security architecture and protocols for secure mqtt-sn. *IEEE Access*, 8:226422–226436, 2020.

[21] Yong Tae Park, Pranesh Sthapit, and Jae-Young Pyun. Smart digital door lock for the home automation. In *TENCON 2009-2009 IEEE Region 10 Conference*, pages 1–6. IEEE, 2009.

[22] JA Quintana, JM Álvarez, J Jiménez Verde, and J Barruetabeña Pujana. A holistic approach on automotive cybersecurity for suppliers. *Proceedings of the VEHICULAR*, 2023.

[23] Sinha S. State of IoT 2023: Number of connected IoT devices growing 16 https://iot-analytics.com/number-connected-iot-devices/, 2023. [Accessed 30-05-2024].

[24] Ousmane Sadio, Ibrahima Ngom, and Claude Lishou. Lightweight security scheme for mqtt/mqtt-sn protocol. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 119–123. IEEE, 2019.

[25] Zhengguo Sheng, Hao Wang, Changchuan Yin, Xiping Hu, Shusen Yang, and Victor CM Leung. Lightweight management of resource-constrained sensor devices in internet of things. *IEEE internet of things journal*, 2(5):402–411, 2015.

[26] S Shin and Kazukuni Kobara. Efficient augmented password-only authentication and key exchange for ikev2. 2012.

[27] SeongHan Shin, Kazukuni Kobara, Chia-Chuan Chuang, and Weicheng Huang. A security framework for mqtt. In *2016 IEEE Conference on Communications and Network Security (CNS)*, pages 432–436. IEEE, 2016.

[28] Soraya Sinche, Duarte Raposo, Ngombo Armando, André Rodrigues, Fernando Boavida, Vasco Pereira, and Jorge Sá Silva. A survey of iot management protocols and frameworks. *IEEE Communications Surveys Tutorials*, 22(2):1168–1190, 2020.

[29] Meena Singh, MA Rajan, VL Shivraj, and Purushothaman Balamuralidhar. Secure mqtt for internet of things (iot). In *2015 fifth international conference on communication systems and network technologies*, pages 746–751. IEEE, 2015.

[30] Govinda R Timilsina, Lado Kurdgelashvili, and Patrick A Narbel. Solar energy: Markets, economics and policies. *Renewable and sustainable energy reviews*, 16(1):449–465, 2012.

[31] Muneer Bani Yassein, Mohammed Q. Shatnawi, Shadi Aljwarneh, and Razan Al-Hatmi. Internet of things: Survey and open issues of mqtt protocol. In *2017 International Conference on Engineering MIS (ICEMIS)*, pages 1–6, 2017.