OAuth based Secured authentication mechanism for IoT Applications

Sandeep Kumar Polu
PG Scholar
Master of Science (IT),
Acharya Nagarjuna University, Andhra Pradesh, India

***Abstract*— The Internet of Things (IoT) presents immense open doors for consumers and organizations, particularly in the regions of warehousing, transportation, and healthcare. Alongside this far-reaching adoption, programmers confront new difficulties to ensure that IoT applications are adequately secured on the grounds that these applications handle a considerable measure of sensitive information. Numerous security breaks have just been accounted for IoT applications, so programmers must spotlight on building security into their IoT applications when they plan and execute such solutions. This paper presents a solution-based approach to deal with limiting security hazards in IoT applications with OAuth based secured authentication mechanism in IOT applications.**

***Index Terms*—Internet of Things (IoT), OAuth, MQTT, Authentication.**

I. INTRODUCTION

Internet-based services have advanced fast in the most recent decade, empowering the production of new technologies and the improvement of existing ones. Moreover, enormous utilization of the Internet has brought about an expanded number of information generators and consumers, making it particularly imperative to address the difficulties of security and access control for this information.

The improvement rate of smart gadgets partner with the Internet is growing a result of the reliable focus towards fulfilling the vision of the Internet of Things (IoT). A standout among the foremost critical issue in IoT is security.

OAuth has been a generally utilized authentication/authorization protocol for HTTP. In here I am will dive into how we can execute OAuth with MQTT because MQTT Protocol is one of the cutting-edge protocols for IoT Devices.

IoT solutions encompass a complicated network of smart gadgets, for example, machines, vehicles or home appliances, that are mounted with software, hardware, sensors, and machine network, which empower these "things" to gather and modify information. The "things" in the IoT network allows programmers to supply a large scope of new services dependent on these cloud-empowered, associated physical smart gadgets. As IoT purposes gather increasingly more personal and sensitive statistics and allow get right of entry to distinct control works over the web, protection turns into a noteworthy challenge. In this manner, an IoT application must:

- **Support continuous monitoring.**
  Indeed, even the best-secured frameworks still leave numerous vulnerabilities. Likewise, the present best-secured solution (both software and hardware) probably won't be adequate to avoid attacks in the future. In this way, you should enhance your safety efforts with persistent observing and steady overhauling of the framework to secure against the most recent types of attacks.
- **Prevent system compromises or breaches.**
  Each tier of the IoT utility ought to enforce nice preventive measures to hold the hackers out. For example, you want to harden the gadget to make sure communication from the gadget to the cloud is secure.
- **Be versatile.**
  At last, if a break occurs, harm must be limited and the framework must recover as fast as could reasonably be expected.

II. DESIGN CHALLENGES

While the significance of IoT security is generally comprehended and settled upon, the actual design and usage of IoT security bring new difficulties and open doors for inventiveness. In the design of most any application, programmers dependably confront an exchange off among security and ease of use. For IoT applications, it turns out to be considerably more dangerous. IoT gadgets regularly have limited computing capacity and memory limit, making it hard to utilize complex cryptographic algorithms that require a larger number of assets than the smart devices give.

Another challenge is refreshing IoT gadgets with ordinary security fixes and updates. Taking off security patches to all IoT gadgets at once can be extremely troublesome in untrustworthy, low-bandwidth device networks, and many existing safety efforts, for example, internet browser security, probably won't be accessible to IoT applications.

What's more, security systems may be produced or improved for new conventions that are planned particularly for the Internet of Things, for example, Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP). In this way, it is particularly essential to factor in security contemplations from the earliest starting point when you design IoT applications.

III. RELATED TECHNOLOGIES

OAuth - OAuth is an open-standard authorization protocol or system that gives applications the capacity to "secure assigned access." For instance, you can disclose to Twitter that it's OK for flickr.com to get to your profile or post updates to your timeline without giving Flickr your Twitter password. This limits the risk significantly: In the occasion, Flickr endures a breach, your Twitter secret key stays safe.

OAuth doesn't share secret key information yet rather utilizes approval tokens to demonstrate an identity among consumers and service providers. OAuth is an authentication protocol that enables you to support one application interfacing with another for your benefit without giving endlessly your password.

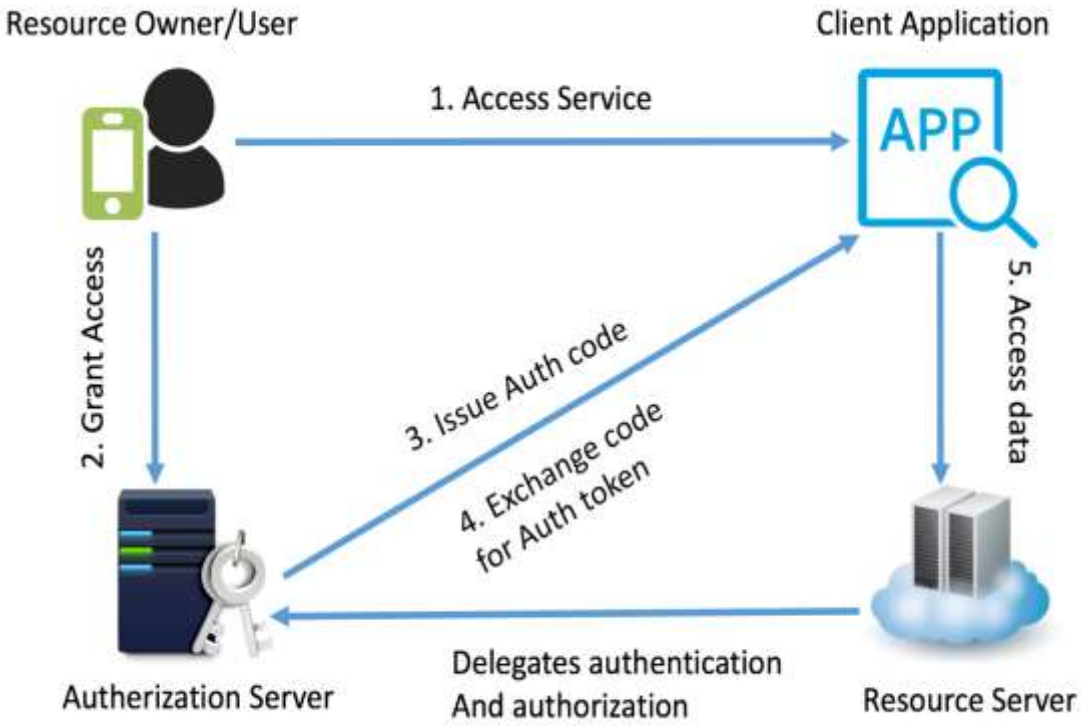

Fig.1: OAuth 2.0 Architecture

The steps in the design are described as follows:

1. You are utilizing a smartphone and are logged into the Canon picture printing site. You snap to print Flickr photographs. The Canon customer application diverts you to the Flickr OAuth approval server. You should as of now have a Flickr account.
2. You sign in to your Flickr account, and the Flickr approval server asks you "Would you like to permit the Canon printing application to get to your photographs?" You click Yes to approve.
3. When effective, the printing application gets an approval code at the callback URL that was preregistered out of the band.
4. The customer application gets the authorization code and should trade this authorization code for an access token. The customer application sends another request to the authorization server, saying it has a code that demonstrates the client has approved it to get to their photographs, and now issues the access token to be sent on to the API (asset/resource server). The authorization server confirms the approval(authorization) code and returns an access token.
5. The customer application sends the access token to the API (asset/resource server) and gets the photographs as requested.

MQTT - MQTT remains for Message Queuing Telemetry Transport. It is a publish/subscribe, to a great degree simple and lightweight messaging protocol, intended for constrained smart gadgets and low-bandwidth, inconsistent network systems. The design standards are to limit network bandwidth and asset requirements while likewise endeavoring to guarantee reliability and some level of confirmation of delivery. These standards likewise end up making the convention perfect of the developing "machine-to-machine" (M2M) or "Internet of Things" world of associated smart gadgets, and for mobile applications where data transfer capacity and battery control are at a premium.

The MQTT convention has Clients and a Broker. MQTT clients subscribe and publish on topics. The MQTT customers impart to each other through an MQTT Broker, which is basically in charge of getting all messages, separating them, choosing who is keen on it and after that sending the message to every single subscribed client.

A diagram demonstrating three clients and a broker is demonstrated as follows. The humidity sensor client distributes the present humidity on the "humidity" topic. The PC and smartphone clients get this humidity reading since they subscribed on the "humidity" subject. The Broker deals with the connections and message interactions.

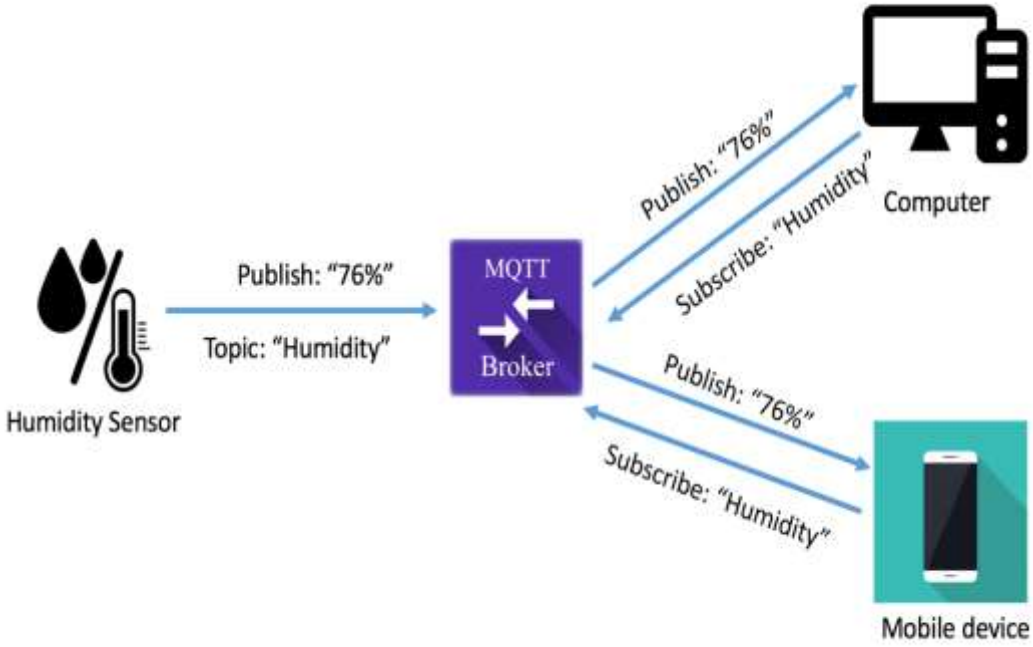

Fig.2: MQTT Publish/Subscribe model

The above diagram has MQTT clients conveying to each other through the MQTT Broker. Notwithstanding, one might need to store humidity readings in a database or maybe send a push notification message to a mobile application when the humidity falls below a specific value.

IV. PROPOSED SOLUTION

Developers have such a significant number of ways that they can apply IoT technologies to make IoT applications. They can make a basic home monitoring system that gives alarms to cell phones and smartwatches, or they can make complex healthcare applications that gather information and control a system of patient gadgets and numerous open doors for arrangements we can't yet envision.

Yet, associating objects like autos, homes, and machines expose a considerable measure of sensitive information, for example, the area of individuals in a building or medical record of patients. This information must be secured as per the key data security standards: classification, confidentiality, integrity, and accessibility.

Most IoT applications consist of three fundamental levels. IoT application components that keep running in every level need to join particular security measures to ensure against different vulnerabilities.

- Devices/Gateways level: Protect against a "fake" server that sends malignant directions, or ensures against a programmer that endeavors to tune in to private sensor information being sent from the gadgets.
- Network/Transport level: Protect against a "fake" gadget that sends false estimations that may corrupt the information that is being held on in the application.
- Applications level: Protect against a "fake" gadget that sends false estimations that may corrupt the information that is being held on in the application.

Application authorization with OAuth 2.0

In situations where enterprises need to utilize their centralized authorization mechanism for MQTT gadgets, an OAuth-based system can be utilized. OAuth 2.0 empowers detachment of the authorization server from a resource server, for example, an MQTT server. When you utilize OAuth 2.0, the client shows its credentials to the authorization server, which at that point plays out the verification check and returns an access token that enables consent to get to a resource.

The access token is then used to associate with the MQTT server. The MQTT server approves the access token, by communicating with the authorization server, at that point grants access to the asset. The flow is portrayed in the below diagram:

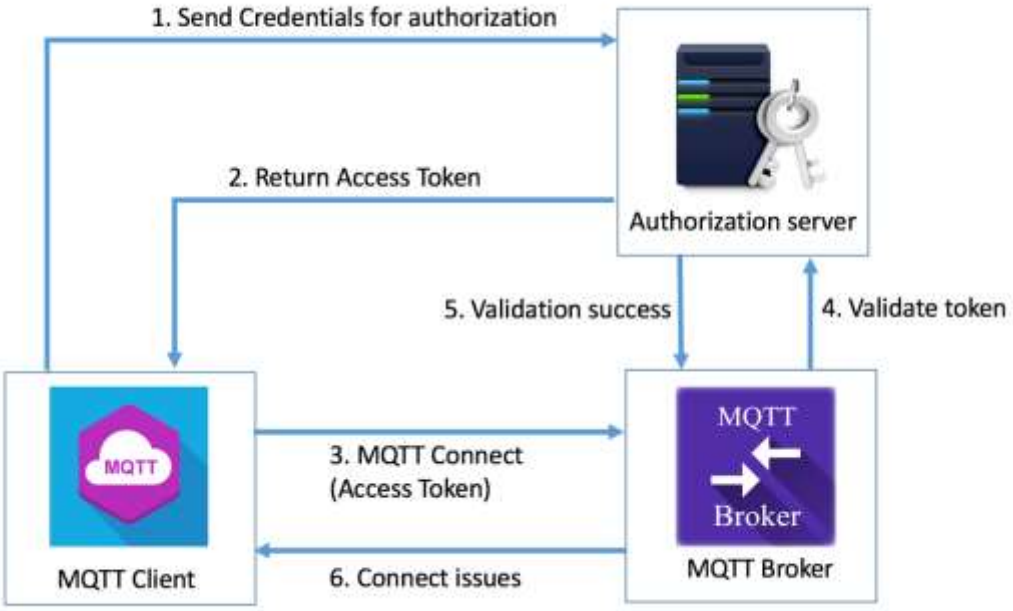

Fig.3: OAuth 2.0 authorization model for MQTT server

Application ID validation

Application ID approval is an additional level of security between the IoT application and the IoT gadget to guarantee that no fake application can send requests to the gadget. This mechanism can be utilized both as startup security and as a correspondence security system. By utilizing this, the gadget stores the unique ID of the IoT application and approves it when it forms the directions that are originating from the IoT application.

In the event that the IoT application sends an invalid unique ID with a request, the request is ignored by the gadget. In the event that the gadget has storage capability, the IoT application unique ID can be encrypted and saved. All things considered, the unique ID request isn't needed after each restart.

The flow is shown in the accompanying diagram:

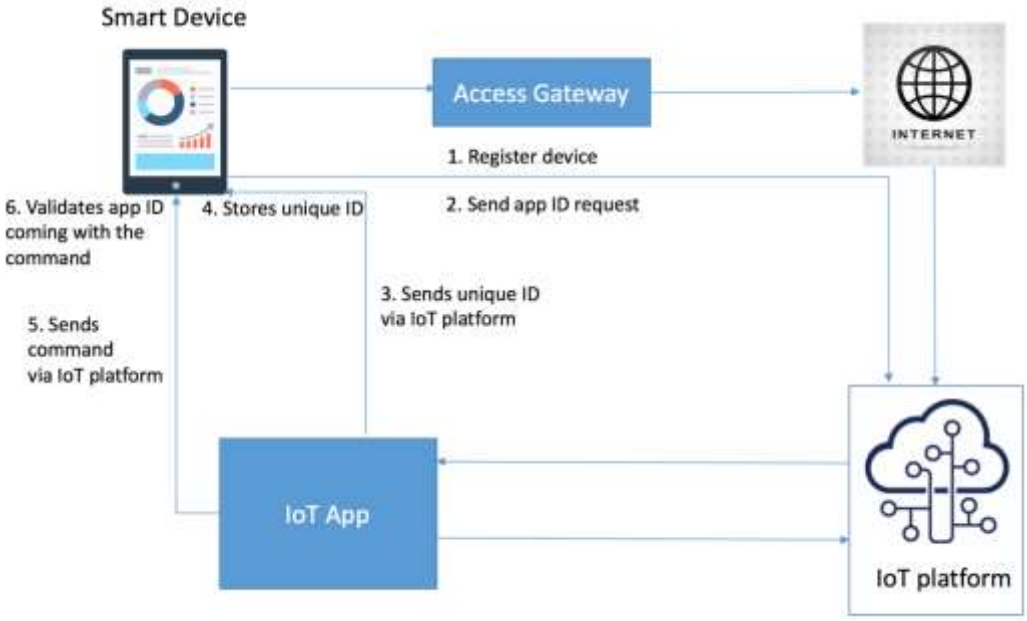

Fig.4: IoT application validation with OAuth

above diagram portrays the following aspects of the flow:

- The application ID approval can be turned on and off dependent on gadget ability.
- During restart, if application ID approval is empowered and unique application ID storage is empowered, the gadget attempts to reestablish the IoT application unique ID from an encrypted storage.

- If the gadget can't stack the unique application ID, it starts a request to get the application.
- Upon accepting the request, the IoT application sends the unique ID to the gadget.
- The gadget stores the unique ID in memory and in a secured file (in the event that it has storage ability).
- After that, the gadget expects a similar application ID in each request originating from the IoT application.
- If there is a mismatch with app ID, the request is disregarded by the gadget.

V. CONCLUSIONS AND FURTHER WORK

In this paper, I have laid out the challenges around applying the recently emerging area of OAuth 2.0 and MQTT to associated gadgets and the Internet of Things. I plot a model of enhancing secured authentication mechanism that focuses on IoT standards and exhibits how protocols, for example, MQTT can be incorporated into existing API Management models. Likewise, the model of the authentication mechanism that I have given OAuth 2.0 offers an extensive number of potential outcomes for giving integrity, availability, and confidentiality in IoT applications.

I have recognized various regions for future research. There is work on improving the IoT security with the Server unique ID validation; Message payload authentication; One-time secret password (OTP) confirmation;

REFERENCES

[1] Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Gener. Comput. Syst. 2013, 29, 1645–1660.

[2] The University of New England. (2017). SMART Farm. [Online]. Available:http://www.une.edu.au/research/research-centres- institutes/smart-farm.

[3] Maler, E.; Catalano, D.; Machulak, M.; Hardjono, T. User-Managed Access (UMA) Profile of OAuth 2.0; Kantara Initiative: Wakefield, MA, USA, 2016.

[4] Sensus. (2017). Smart WaterSmarter at Every Point. [Online]. Available:www.sensus.com/internet-of-things/smart-water

[5] Hammer-Lahav, E. The OAuth 1.0 Protocol; IETF: Fremont, CA, USA, 2010.

[6] H. El-Sayed and G. Thandavarayan, "Congestion detection and propagation in urban areas using histogram models," IEEE Internet Things J., to be published.

[7] Jara, A.J.; Martinez-Julia, P.; Skarmeta, A. Light-weight multicast DNS and DNS-SD (lmDNS-SD): IPv6-based resource and service discovery for the Web of Things. In Proceedings of the 2012 IEEE Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Palermo, Italy, 4–6 July 2012; pp. 731–738.

[8] K. M. Alam, M. Saini, and A. E. Saddik, "Toward social Internet of vehicles: Concept, architecture, and applications," IEEE Access, vol. 3, pp. 343357, Mar. 2015.

[9] C. A. Tokognon, B. Gao, G. Tian, and Y. Yan, "Structural health monitoring framework based on Internet of Things: A survey," IEEE Internet Things J., vol. 4, no. 3, pp. 619635, Jun. 2017.

[10] K. Ozcan and S. Velipasalar, "Wearable camera- and accelerometer- based fall detection on portable devices," IEEE Embedded Syst. Lett., vol. 8, no. 1, pp. 69, Mar. 2016.

[11] Emerson, S.; Choi, Y.K.; Hwang, D.Y.; Kim, K.S.; Kim, K.H. An OAuth based authentication mechanism for iot networks. In Proceedings of the 2015 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 28–30 October 2015; pp. 1072–1074.

[12] Amitranjan Gantait, Joy Patra, and Ayan Mukherjee; Securing IoT devices and gateways[Online] Available: https://www.ibm.com/developerworks/library/iot-trs-secure-iot-solutions1/index.html

[13] Seitz, L.; Selander, G.; Wahlstroem, E.; Erdtman, S.; Tschofenig, H. Authorization for the Internet of Things Using OAuth 2.0; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2015.

[14] Ouaddah, A.; Mousannif, H.; Elkalam, A.A.; Ouahman, A.A. Access control in the Internet of Things: Big challenges and new opportunities. Comput. Netw. 2017, 112, 237–262.

[15] Gupta, R.; Banks, A. MQTT Version 3.1.1 Protocol Specification; Oasis Standard; OASIS: Burlington, MA, USA, 2014.

[16] Cirani, S.; Picone, M.; Gonizzi, P.; Veltri, L.; Ferrari, G. Iot-oas: An oauth-based authorization service architecture for secure services in iot scenarios. IEEE Sens. J. 2015, 15, 1224–1234.