Dynamic Approach to IoT Security

A new security solution for IoT

Tobias Schläpfer
IoThentix
Rickenbach b. Will, Switzerland
tobias.schlaepfer@iothentix.ch

David Lorenz, Simon Künzli
Zurich University of Applied Science (ZHAW)
Institute of Embedded Systems (InES)
Winterthur, Switzerland
david.lorenz@zhaw.ch, simon.kuenzli@zhaw.ch

***Abstract*— Today's security solutions for IoT applications remain static in an ever-changing environment, making applications insecure and vulnerable to ever-growing cyber threats. The static solutions make credential life-cycle management difficult and lack proper protection of APIs. Overall, they weaken the security of not only the IoT devices but all components involved. A more dynamic approach to the security of IoT applications is required. IoThentix has developed a token-based approach to device authentication that not only allows IoT devices to access protected APIs according to state-of-the-art standards but also enables dynamic management of device credentials. As a result, it significantly improves the credential life-cycle management and the security of the entire IoT application.**

***Keywords*— IoT security, dynamic security, token-based authentication, OAuth 2.0 for IoT**

I. Introduction

Internet of Things (IoT), Industrial IoT (IIoT), and Operational Technology (OT) applications are omnipresent and add value to a wide variety of industries. Today, IoT applications are closely integrated into processes and infrastructures, with many different interfaces that form complex systems. Furthermore, IoT applications are often distributed over large areas and are composed of a large number of devices. On the one hand, being a crucial part of larger systems that generate high value naturally attracts the attention of cybercriminals. On the other hand, IoT applications often only provide weak security, not only weakening the security of the IoT applications themselves but also the systems they connect to. Especially, the interfaces, i.e., APIs, where IoT devices connect and interact with systems are at risk since IoT devices lack the capability to interact in the same secure and standardized manner IT systems are used to. The buzzword in that regard is "Zero Trust". The missing capabilities result from the use of static, outdated security practices, such as the use of certificate-based authentication solutions.

Certificate-based authentication solutions have several drawbacks, which are outlined in the following section. The paper then introduces the new dynamic, token-based authentication approach developed by IoThentix as a viable solution to address the drawbacks introduced by certificate-based solutions. Section four outlines in detail how the security of the token-based approach is guaranteed. The paper closes with appropriate conclusions.

II. Static Security Solutions

As outlined in the previous section, IoT applications today often use static security solutions that are not fit for purpose. The following drawbacks are highlighted in this section:

- Credential management
- Static device credentials
- Not built to scale
- No authorization

The weaknesses are explained using a static certificate-based authentication solution as a reference, but any security solution not able to address the outlined drawbacks exposes similar issues as a static security solution.

A. Credential management

Credential management is a cumbersome process in all life-cycle stages of any IoT device, especially for certificate-based solutions, as certificates need to be created, signed, and installed on each device individually. Since the certificates also expire at some point in time, the whole process eventually needs to be repeated.

B. Static device credentials

As a result of the cumbersome credentials management process, companies tend to create device certificates that have lifetimes of five years or even longer in order to avoid the process of renewing the device certificates over the lifetime of the IoT device. Long-living security credentials pose a significant threat to any application relying on them.

C. Not built to scale

IoT applications typically are composed of a large volume of devices that may be distributed over a wide area. For that reason, a complex or manual process is required to provision and update device credentials, resulting in a tremendous amount of effort and time, if it is possible at all.

D. No authorization

The sole purpose of certificates is to authenticate devices. As a result, certificate-based solutions do not provide any means of authorization. However, authorization is of the same importance as authentication, as system operators not only want to know which device is connecting to the API but also if the device is authorized to access the resource in question. To overcome this shortcoming, certificate-based solutions often introduce customized means of authorization that enable only a single use case which is not standardized and is inherently insecure.

III. DYNAMIC SECURITY SOLUTION

A more dynamic approach will significantly benefit all types of IoT applications as well as their application interfaces. One of the state-of-the-art standards used in IT to protect APIs is OAuth 2.0 [1]. OAuth 2.0 defines a token-based approach to protect application interfaces. The benefits of a token-based approach are:

- Tokens have a short lifetime
- Trust is established, not preconditioned
- Provide a framework for authorization

While OAuth 2.0 is primarily used in IT to authenticate[1] and authorizes human users, it can also be used in machine-to-machine applications, hence also for IoT applications.

IoThentix has adopted the token-based approach to fit the specific requirements of IoT applications, enabling IoT devices to utilize the benefits of token-based solutions and making their authentication and authorization process more dynamic and secure. Furthermore, the token-based approach simplifies credential management during the entire life cycle of the devices.

A. Token-based authentication & authorization

The core of a token-based solution is the identity provider (IdP). The primary function of the IdP is to identify and authenticate IoT devices. Upon successful authentication, the IdP issues a short-lived access token to the IoT device. The IoT device can then use this access token to authenticate itself to any protected API, e.g., to an IoT platform. The IoT platform then has to verify the access token with the IdP before excepting any data or request sent by the IoT device.

The process of utilizing the token-based approach for IoT applications is divided into four main steps. First, the devices have to be registered with the IdP. Once the devices are operational, they need to identify themselves with the IdP. As a result, the devices then get a short-lived token they can use to authenticate themselves to a target system, e.g., an IoT platform. Finally, the platform verifies the token with the IdP and grants the IoT device access to the authorized resources.

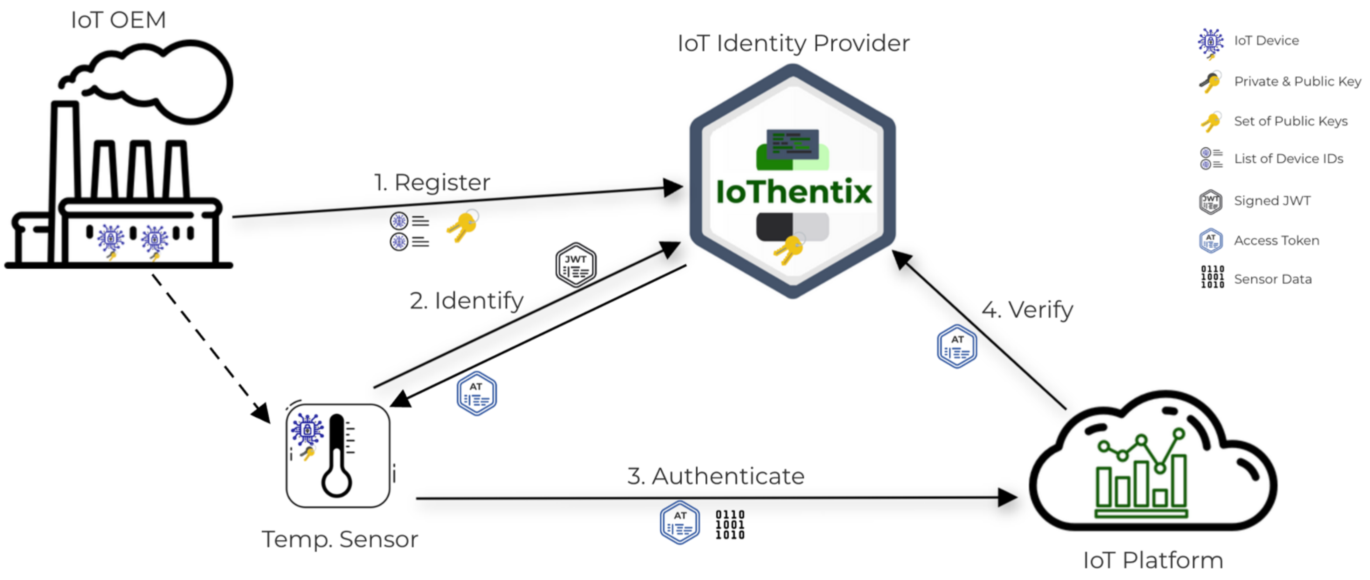

Figure 1: Overview token-based authentication flow

[1] OAuth 2.0 only provides a framework for authorization, to also provide authentication the OIDC [10] standard must be applied. The OIDC standard is built on top of the OAuth 2.0 standard.

1) Register: To register an IoT device, the device first has to generate a key pair. Then the public key, along with a unique device ID, is registered with the identity provider. There are two main observations to make here. First, the private key is generated on the IoT device and never leaves the device. Second, the information stored with the IdP is of public nature. Hence the IdP does not store any sensitive information about the IoT device.

2) Identify: For an IoT device to identify and authenticate itself, it must provide a so-called private key JSON Web Token (JWT) *[2]*. A private key JWT is a base64 encoded JSON object composed of three main parts, a header, a payload, and a signature. The header and the payload contain information (claims) to identify the device and protect the JWT from replay attacks. The signature is generated using the private key corresponding to the device's registered public key, ensuring the token's integrity, and allowing the IdP to authenticate the device. The following table displays an example of an encoded private key JWT and its decoded information.

Private key JWT	
Encoded	***Decoded***

eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImRlbW8uZGV2aWNlLjAxIn0.eyJzdWIiOiJkZW1vLmRldmljZS4wMSIsImp0aSI6IkRFNjVBNEIwMDhFN0IyNzg2MkFENUE0QjE2MzZDIn0.R7OWvQzNSbeEOhFRMmRUDBY9zLNUFf-i2Un5YLAP8QqTXGdUGBr8m14olyFmXy29VwUTnhLAhvGZnqxAyMWzqQ	{
	"alg": "ES256",
	"typ": "JWT",
	"kid": "demo.device.01"
	}

	{
	"sub": "demo.device.01",
	"jti": "DE65A4B008E7B27862AD5A4B1CB1636C"
	}

Table 1: Private key JWT example

Upon successful validation of the private key JWT, the IdP issues an access token (AT) to the IoT device. The access token is a so-called opaque token, which is a ~100-byte long random string that only the IdP that has issued the AT can verify. The following is an example of an opaque access token,

Iz2oXZIqaGGEtkWUBmtL77d0cS-_lbf1Zs02vNJBBtY.UzoqH2LuVk0tuZfceosaYfZ3VsSDYtAMGIJ8yIBhtJA

The AT contains several claims, such as the device ID, the requested scope, the issuer of the token, and the lifetime of the token. The lifetime of the AT can be defined according to the application requirements. The IdP may provide the information in the AT claims upon successful validation in step 4 to the IoT platform. The AT must be securely stored and may be used to authenticate the IoT device against any protected API.

3) Authenticate: Once the IoT device has received the access token from the identity provider, it can start sending its data/request to the IoT platform. The device establishes a secure connection to the IoT platform, authenticating the IoT platform through a (D)TLS connection. Then the device passes the data along with the access token to the IoT platform. The platform can now authenticate the IoT device based on the provided access token with the identity provider by calling the so-called introspection endpoint.

4) Verify: To verify the access token, the IoT platform must call the introspection endpoint of the IdP and pass along the provided access token of the IoT device. The IdP then validates the access token for its claims, such as its scope, expiration time, and issuer. Upon successful validation, the IdP confirms the token's authenticity to the IoT platform. The following table displays a standard response of the IdP when the introspection endpoint is called. The response confirms the validity as well as the claims associated with the AT.

Introspection endpoint response	
Successful	***Failed***

{
"active": true,
"sub": "demo.device.01",
"scope": "tenant.demo refresh.token temp",
"exp": 1675581071,
"iss": "https://iot-idp.demo.apps.iothentix.io/"
}
	{
	"active": false
	}

Table 2: Introspection endpoint response

Assuming the validation of the AT was successful, the IoT platform can authorize or deny the request based on the additional information of the AT, e.g., based on the scope values or the identity of the IoT device in the "*sub*" claim.

By following the outlined process, the IoT devices are able to use short-living, standardized access tokens allowing them to interact with any protected API. As a result, creating a highly dynamic and secure approach for IoT security. Furthermore, the new dynamic approach also simplifies device credential management, which is outlined in the next section.

B. Device credential management

Managing device credentials is a crucial part of the overall security of an IoT application. It should be dynamic yet simple (automated). Both requirements can be achieved with a token-based approach.

1) Initial creation: As outlined in the previous section, the initial creation of the credentials is straightforward and should be processed on the devices themselves. After that, an authorized process (manual or automated) is responsible for doing the initial registration.

2) Operation: Once the devices are operational, devices should constantly update their security credentials, following the principle "As short as possible, as long as necessary". During operation, there are two types of credentials the IoT devices will have to manage, the AT issued by the IdP and the device credentials to identify with the IdP.

a) Renew access tokens: The AT issued by the IdP will only be valid for a short period of time, e.g., one day. As the process of creating a JWT may be quite resource-draining for resource-constrained IoT devices, the IdP offers an API to receive a new access token based on a previously issued access token. The detailed process of how this is achieved in a secure and automated manner is outlined in Section 5. Renewing access tokens based on previously issued access tokens is undoubtedly less secure than through the whole identification process outlined in the "Identify" step of the previous section. However, depending on the requirements of the IoT devices, such a procedure offers a feasible compromise.

b) Update device credentials: The concept of short-living security credentials is at the heart of the token-based approach. This should also extend to the device credentials that are used to authenticate the IoT devices to the IdP. However, as this process is critical, the IoT devices must authenticate with the IdP in the same manner as when requesting an access token, with the addition that the integrity of the new credentials must also be ensured. The detailed process of how this is achieved in a secure and automated manner is outlined in Section 5.

3) End of life: Once IoT devices have reached their end of life, it is important that these devices are decommissioned along with their credentials. Again, the token-based approach is straightforward, as it can be achieved by simply removing the registered credentials from the IdP. As a result, this will prevent the IdP from issuing any new AT for decommissioned devices. Furthermore, already issued AT may either expire soon or can be revoked in the IdP.

IV. Proof of Concept Application

A proof of concept was implemented to prove that the suggested solution is viable for IoT. It consists of the IdP and its APIs, a small IoT network, and a simulated IoT platform. The network had the same topology as shown in Figure 1: OverviewFigure 1.
The IdP offers the following APIs:

- Token API to request and renew access tokens.
- IdP API to introspect existing tokens and verify their validity.
- Device API to update the device credentials.

For the IoT network, technologies and hardware had to be chosen that are used in real-life applications and have adequate resources to simulate a realistic scenario. For those reasons, we decided to build a small OpenThread [3] network with one end device and one border router. We used an nrf52840 development kit for the end device from Nordic Semiconductors [4]. On the MCU, we ran a simple application that could consume all the APIs of the IdP and send some dummy data to the simulated IoT platform. We built the application using the Zephyr-RTOS, which provides a module for OpenThread, and the mbedTLS crypto-library for all needed crypto operations. See Figure 2 [3] [5] for an overview of the layer stack of the application.

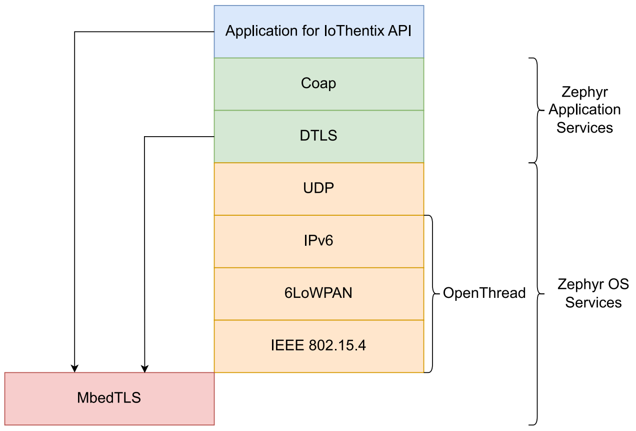

Figure 2: Layer stack of proof-of-concept application

The OpenThread border router was built on a Raspberry Pi using the standard image provided by OpenThread. Thread and therefore OpenThread are IPv6-based networking protocols. The IdP, on the other hand, is a cloud application running as an AWS cloud service and is IPv4 only as of now. Thus, the border router must provide a NAT64 and DNS64 to do network translations between the two IP versions. We used Tayga for the NAT64, which comes with the border-router image and only has to be configured and activated. More information on Tayga can be found in [6].

The IoT platform existed in two forms, running in the AWS cloud along with the IdP and as an independent entity running on a Raspberry Pi. It is a simple application which offers an online resource where IoT devices can store data. The OpenThread device sends the dummy data along with an access token to the IoT platform, which validates the token with the IdP, and stores the data if the validation is successful.

With this proof-of-concept implementation, we could show that an IoT device can authenticate itself securely to an IoT platform without the need for a device certificate. The implementation was realized with remarkably simple, off-the-shelf components, which are used extensively in the industry [7] [8]. Further, our solution provides not only secure and effortless device authentication but also the means for a manageable and standardized authorization mechanism.

V. Security Considerations

This section outlines how the solution developed by IoThentix ensures that only genuine IoT devices are able to consume the APIs of the identity provider (IdP). Hence, the measures further described below ensure the security of the new token-based approach. Furthermore, this section provides general recommendations regarding the management of device credentials on an IoT device.

A. Device registration

Before an IoT device can consume the identity provider's APIs, the device must be registered. It is important to note that the IdP does not store any sensitive device data.

1) Public key: The registered public key has to correspond to the private key on the IoT device that will be used to sign the JSON Web Token (JWT), which is used to identify and authenticate the IoT device with the identity provider.

2) Device ID: The device ID has to be unique among all registered devices of a particular client to ensure unique identification of the device by the combination of the device ID and public key.

B. Securely identify & authenticate IoT devices

The IoT device authenticates to the IdP using a JWT before receiving an access token. Hence, the IdP must be able to identify and authenticate eligible devices reliably. This is achieved by a thorough validation of the JWT provided by the IoT device. The validation includes the following.

1) Signature validation: The JWT signature is validated using the registered public key for the device ID provided in the "*kid*" header of the JWT.

2) Device ID validation: The JWT has to contain a "*sub*" claim with the device ID that is the same as the kid in the JWT header.

3) JWT identifier validation: Each JWT MUST contain a JWT ID ("*jti*") claim that represents a 16-byte random number encoded as a hexadecimal string that has never been used before by the same device. To prevent replay attacks, the IdP will register all "jti" claims provided by each IoT device.

4) JWT expiration: A JWT MAY contain an expiration ("*exp*") claim that indicates a time after which the JWT should be considered invalid. However, this requires the IoT device to be "time-aware", which is often not the case. Hence, the expiration claim is only optional, but if possible, the usage of the "*exp*" claim is strongly recommended.

C. Securely renew access tokens

The following requirements apply to ensure that the process of renewing an access token based on a previously issued access token is secure.

1) Access token validation: The access token that is used to request a new access token MUST not be expired.

2) Scopes validation: The access token that is used to request a new access token MUST have the scope "*refresh.token*". Furthermore, the renewed access token can only have the same scopes as the previously issued access token.

3) Limit consecutive token renewals: To ensure that devices have to authenticate using the JWT authentication process from time to time, a limit on consecutive token renewals is enforced. The number of consecutive renewals can be customized to fit the application requirements.

D. Securely update device credentials

To ensure the security and integrity of the device credential update process, the following parameters are evaluated before the device credentials are updated.

1) Signature validation: The JWT signature is validated using the registered public key for the device ID provided in the "*kid*" header of the JWT. Furthermore, the JWT has to contain a "*public_key*" claim that contains the new public key that should be registered with the IdP. As a result, the integrity of the new public key is ensured through the signature of the JWT.

2) Device ID validation
3) JWT identifier validation
4) JWT expiration: The validation regarding the device ID, the "*jti*" and "*exp*" claims is identical to the validation when an IoT device is requesting an access token.

5) Proof of possession: As the final validation, the device has to prove that it is in possession of the private key corresponding to the public key it tries to update. For this, the device has to provide a proof of Possession (PoP) object in the request body. The PoP object is a base64 encoded representation of the result of an ECDSA_SHA256 operation on the JWT provided for the authentication in the request. The IdP will use the public key provided in the "public_key" claim in the JWT to validate the PoP confirming that the device is in fact in possession of the corresponding private key.

E. Securely manage device credentials

For the security of the token-based approach, the secure management of the device's private key and the received access token is essential. Ultimately, it is the responsibility of the OEM to ensure the proper handling of the security credentials. However, the following best practices should be considered:

1) Security in transit: All communication with the IdP should be encrypted, and the server certificate provided by the IdP during the (D)TLS handshake SHOULD be validated.

2) Security at rest: To ensure the credentials are secure at rest, the private key and access token should be stored in dedicated secure storage, either ensured by software isolation techniques or dedicated secure storage.

3) Credentials integrity: In case the IoT devices support the use of Physical Unclonable Functions (PUF) credentials, the keys to authenticate the IoT device against the IdP should be derived using the PUF capabilities.

4) Lifetime: The lifetime of all security credentials should follow the principle: "As short as possible, as long as necessary".

VI. SUMMARY & OUTLOOK

In this paper, we presented a token-based authentication approach and compared it to state-of-the-art static credential-based security. The new dynamic approach adopts the widely used OAuth 2.0 standard and the recently published RFC 9200, Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework [9], to fit the special requirements of IoT applications. Its dynamic nature supports the regular exchange of IoT device credentials, enhancing the overall security of an IoT system. We showed the applicability of the approach by a proof-of-concept implementation and discussed several device security-related use cases, including device identification and authentication.

Following up on the promising results with the functionally correct prototype implementation, we will investigate the non-functional properties of the proposed approach. We plan to measure resource consumption on our proof-of-concept implementation and compare the results with a common static security approach. In addition, it will be interesting to prove the applicability and scalability of the proposed scenario for real applications.

ACKNOWLEDGMENT

The work presented in this paper has been partly funded by Innosuisse – Swiss Innovation Agency as Innocheque under grant number 61368.1 INNO-ICT.

VII. REFERENCES

[1] D. Hardt, *RFC 6749,* Internet Engineering Task Force (IETF), 2012.

[2] J. B. N. S. M. Jones, *RFC 7519,* Internet Engineering Task Force (IETF), 2015.

[3] Google, „OpenThread,“ Google, 2023. [Online]. Available: https://openthread.io/. [Accessed on February 2023].

[4] Nordic Semiconductor, "nordicsemi.com," 21 02 2023. [Online]. Available: https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk. [Accessed February 2023].

[5] Zephyr Project, „Zephyr overview 2023 slide 8,“ 2 01 2023. [Online]. Available: https://www.zephyrproject.org/wp-content/uploads/sites/38/2023/01/Zephyr-Overview-20230124.pdf. [Accessed February 2023].

[6] Google, „OpenThread/Tayga,“ 2023. [Online]. Available: https://github.com/openthread/tayga. [Accessed February 2023].

[7] Nordic Semiconductor, „Quarterly Reports 2022 Q4,“ Nordic Semiconductor, Trondheim Norway, 2022.

[8] N. Flaherty, „ee News Europe,“ 1 09 2021. [Online]. Available: https://www.eenewseurope.com/en/raspberry-pi-most-popular-sbc-in-industrial-and-iot-applications/. [Accessed February 2023].

[9] G. S. E. W. S. E. H. T. L. Seitz, *Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth),* Internet Engineering Task Force (IETF), 2022.

[10] J. B. M. J. B. d. M. C. M. N. Sakimura, „OpenID Connect 1.0,“ OpenID, November 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html#JWT. [Accessed February 2023].