D5.2 – Initial AAA layer for IoT cross platform models - Final

Deliverable ID	**D5.2**
Deliverable Title	**Initial AAA layer for IoT cross platform models**
Work Package	**WP5**
Dissemination Level	**PUBLIC**
Version	**FINAL**
Date	**2019-03-28**
Status	**FINAL**
Lead Editor	**AIRBUS**
Main Contributors	**Nicolas PABST (AIRBUS), Guillemette MASSOT (AIRBUS), Timothy WARD (PAREMUS)**

Published by the BRAIN-IoT Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780089.

Document History

Version	Date	Author(s)	Description
0.0	2019-01-21	TIMOTHY WARD (PAREMUS)	First Draft with TOC
0.1	2019-02-14	NICOLAS PABST (AIRBUS)	Include AIRBUS components
0.2	2019-03-01	NICOLAS PABST (AIRBUS)	Take into account and clear comments
0.3	2019-03-06	GUILLEMETTE MASSOT (AIRBUS)	Minor correction and updates

Review History

Version	Review Date	Reviewer	Summary of Comments
0.1	2019-03-20	Victor Sonora Pombo (Improving Metrics)	Approved with minor comments
0.2	2019-03-21	Mario Diaznava (ST)	Approved with minor comments

Table of Contents

1 Introduction

This document reports the activities performed in Task 5.2. This task aims at designing and implementing an Authentication, Authorization and Accounting (AAA) layer in order to provide security and trust for the Brain-IoT system.

A BRAIN-IoT environment is composed of a number of BRAIN-IoT Fabrics. These are in-turn composed of a number of BRAIN-IoT nodes (see the related document RD.1 in §1.2). A BRAIN-IoT environment may consist of arbitrary complex distributed interactions between dynamically deployed Smart Behaviours. These behaviours may migrate between runtime locations within each Fabric environment (see the related document RD.2 in §1.2).

To be discovered, and to participate as a member of a Fabric, each BRAIN-IoT node must have the appropriate X509 certificates. Certificates and TLS are both good defenses against external Man-in-the-middle (MITM) attacks, but are insufficient against internal MITM. So to guard against a Smart Behaviour erroneously (or maliciously) editing data for an event and sending them on, the project requires that each BRAIN-IoT message contains its own Authentication token.

Finally, BRAIN-IoT must ensure that Smart Behaviours only interact at runtime in the expected manner.

1.1 Scope

Security is an extremely broad subject within computer science, and it can quickly become difficult to describe the complete set of security actions that apply to a software system. In this deliverable the scope of security is limited to the following considerations:

- The registration of identity for users, devices and Smart Behaviours within the Brain-IoT system.
- The authentication of users, devices and Smart Behaviours.
- The validation of data integrity for messages passed through the Brain-IoT system.
- Permission management for users, devices and Smart Behaviours, limiting the permitted actions and data interactions.
- The runtime application of permissions to permit or deny access.

1.2 Related documents

ID	Title	Reference	Version	Date
[RD.1]	BRAIN-IoT_D4.1_Initial discovery, search, composition and orchestration enablers		1.0	Oct-2018
[RD.2]	BRAIN-IoT_D3.3 – Initial Enablers for dynamic distribution of IoT behaviour		1.0	Mar-2019

2 Scenarios

The following scenarios outline the necessary security flow required by Brain-IoT in some typical and illustrative situations:

2.1 Published data streams

A published data stream is a series of data events that may be triggered either by an action in the real world (for example, a car driving past a motion sensor) or by the passage of time (for example, a periodic reading from a temperature sensor).

The Brain-IoT Smart Utility use case includes the monitoring of water levels at a dam. This is a classic example of a periodic measurement, where the depth of the water will be reported at fixed intervals, potentially as frequently as once per minute. This data is then sent on to the Utility Company where it can be acted upon if needed, and also stored for long term analysis.

To ensure that erroneous and/or malicious events cannot be injected the data event published must be able to:

- Identify the sensor that produced the data.
- Confirm that the data has not been corrupted or changed.

2.2 Forwarded/Consumed/Enriched data streams

A typical data stream contains many data events, however most of those events are not interesting because no change has occurred or no action is needed. If all of these data events are sent to be processed centrally then it can cause a significant scaling problem. Therefore, it is typically useful, to perform a degree of filtering and aggregation at the edge of the system to reduce the number of events that reach the core of the system.

In Brain IoT, this edge filtering and enrichment is performed by Smart Behaviours. These components receive locally produced data events and then determine whether they should be discarded, forwarded, or used to trigger a different data event.

2.2.1 Simple Filtering

The simplest mode of operation for a stream processing behaviour is to filter out records that are deemed to be uninteresting. For example, in the case of water level readings from the dam a behaviour might apply the following rules:

1. Has an event been forwarded in the last 60 minutes? If no then forward the event to the cloud core.
2. Is the water level read higher or lower than 5 centimeters from the last forwarded reading? If yes, then forward the event to the cloud core.
3. Consume the event and prevent it being forwarded.

In this model, the data events are not being changed, however the System must be able to check that Smart Behaviour is permitted to:

- Read the data values from the publishing sensor.
- Consume an event and prevent it being forwarded.

2.2.2 Aggregating Filtering

Another mode of operation for a stream processing behaviour is to provide an average based on several readings potentially discarding readings that are believed to be outliers. For example in the case of water level readings from the dam there may be small fluctuations due to waves on the water surface. Behaviour might therefore forward the rolling average of the last 5 measurements, rather than the raw data.

In this model, the original data events are all consumed and new data events are created by the Smart Behaviour. The System must be able to check that Smart Behaviour is permitted to:

- Read the data values from the publishing sensor
- Consume the events and prevent them being forwarded
- Create and publish new events using the identity of the Smart Behaviour.

2.3 User driven manually actions

Users of the Brain-IoT system may manually interact with the sensors and actuators in the system, either requesting the value of a current sensor, or querying/changing the state of an actuator. In the Brain-IoT use cases this corresponds to scenarios such as:

- Manually closing the dam gates so that an engineer can inspect them.
- Instructing a robot in the factory to retrieve a stock cart.

User driven actions are probably the use case, which is most closely aligned with existing software applications. As such, the interaction model is similar, with the user requiring:

- Proof of their identity.
- Permission to query the sensor or perform the action on the actuator.

2.4 Automatic actions

Like user driven actions, automatic actions interact directly with the sensors and actuators in the system, either requesting the value of a current sensor, or querying/changing the state of an actuator. Unlike user driven actions, however, there is no human in the loop. Rather than the action being triggered by an external request the action is requested as a result of a trigger from a Smart Behaviour. In the Brain-IoT use cases this corresponds to scenarios such as:

- Automatically opening the dam gates when the water level rises above a certain level

- Switching off the water pump when the header feed tank is full.
- A robot in the factory requesting that a door be opened so that it can proceed.

As they are triggered by a Smart Behaviour, there is no user identity to associate with an automatic action. This use case therefore requires that Smart Behaviours be given their own identities which are then associated with the actuation request. From this point on, there is little difference between a user-triggered action and an automatic action.

2.5 Security Auditing

The Brain IoT system may monitor and control a large number of sensitive and/or safety-critical infrastructures. It is therefore vital to be able to look back and understand what instructions were given and by whom, either as part of a post-incident analysis or as part of a security review.

As described in D3.3 (see section 5.4) distributed control and monitoring in the Brain-IoT runtime is implemented via asynchronous events between the interacting parties over an EventBus. The Brain-IoT EventBus will provide a configurable facility to persist events of certain types.

The persisted events include information concerning the actor, the target, the action and the date/time. While control events from an Operator to a target Device, or between Smart Behaviours would usually always be persisted; discovery events possibly persisted, but monitoring events would not usually be persisted. The period events are retained is determined by the User Case regulatory requirements.

3 Brain-IoT Security Components

A Brain IoT fabric is constructed from a set of Brain-IoT nodes: Edge nodes that communicate directly with sensors and devices, and Core nodes which run services.

Smart Behaviours may run on any Brain-IoT node. Filtering and "fast reaction" Smart Behaviours will commonly run on the Edge node, with more complex distributed behaviours typically running in Core nodes. It is therefore necessary for all nodes to be able to interact with the security services.

The following Security Components are required within a BRAIN-IoT environment.

- Authentication Service
- Message Integrity Service
- Authorization Service
- Identity Administration Service
- Audit service

They will be described in the following subsections.

3.1 The Authentication Service

It is necessary for users, devices and Smart Behaviours to establish their identities using the authentication service. This action may be performed on behalf of the device or Smart Behaviour

based on the configuration of the Brain-IoT node. This service is linked to the Identity Administration Service.

To avoid having a single point of failure, the authentication service must be replicated, but it is not necessarily required to have an Authentication Service present on every node. Authentication Service instances are discovered through the OSGi service registry, although the backend authentication service may be deployed as a separate process. It is highly desirable that once authenticated by one Authentication Service the device/Smart Behaviour does not have to re-authenticate with another service, but it can instead reuse the same authentication token (SSO). This will help to significantly simplify the security service.

In addition, the Authentication Service will be used to authenticate operators and administrators of the Brain-IoT fabric. The Authentication Service must therefore have a suitable API that can be used to perform this authentication.

To do so, the project will integrate the CymID Cloud solution to the BRAIN-IoT Fabric.

CymID Cloud solution support main authentication mechanisms such as Kerberos, login / password, certificates, etc. It also supports Identity Federation such as SAMLv2 and OpenID Connect in order to rely on third party authentication services. It is packaged as a Docker.

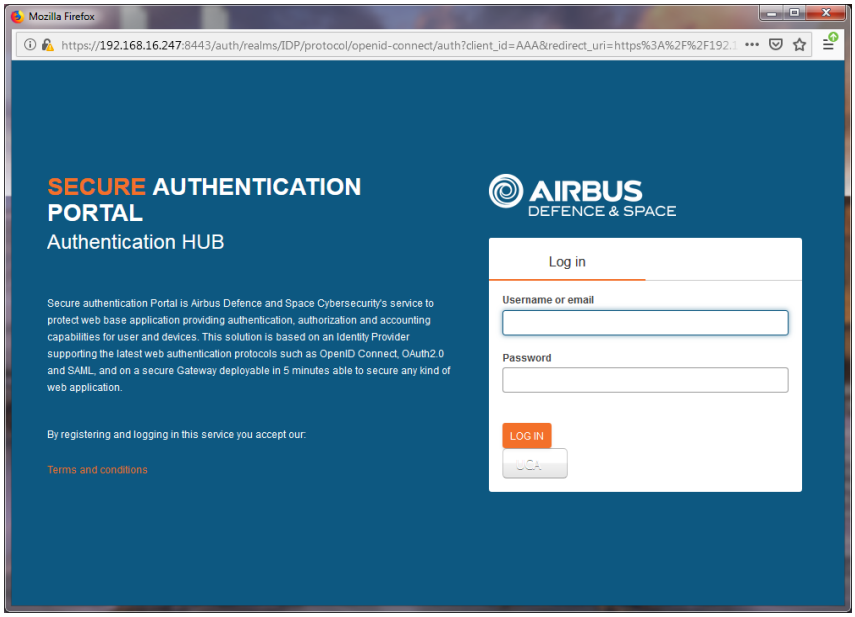

Figure 1: Authentication page of the Authentication Service

As proof of the identity, CymID Cloud uses Java Web Token (JWT). These JSON-based open standard (RFC 7519) tokens are a list of claims signed by the Authentication Service that delivers it.

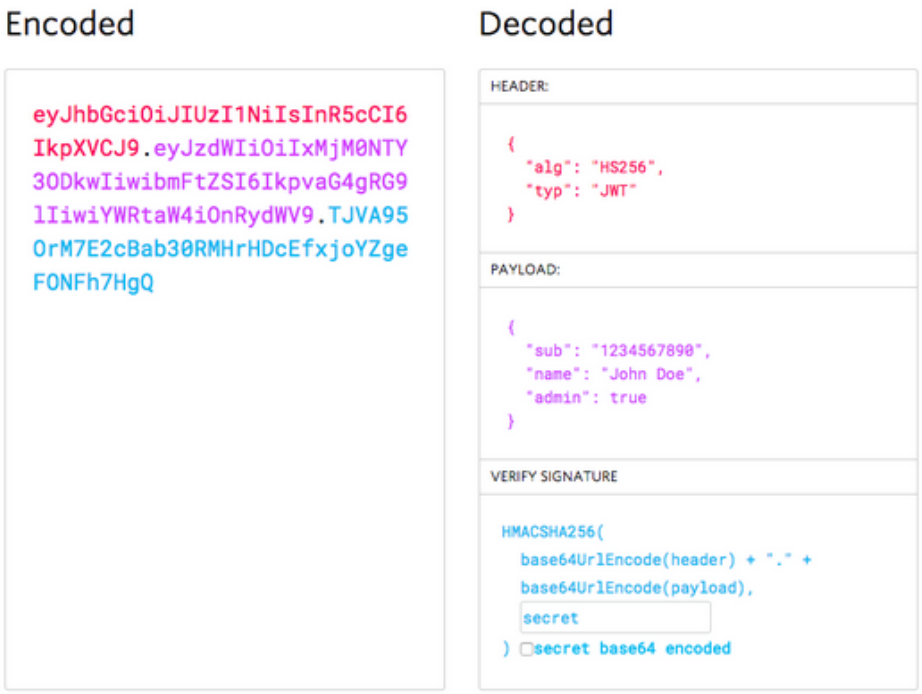

Figure 2: Java Web Token

These tokens will be verified by the Authorization Service to determine if a data, request, access should be allowed or not.

3.2 The Message Integrity Service

The Message Integrity Service provides two key functions:

1. The creation of publicly shareable cryptographically integrity tokens providing the name of the user/device/behaviour and a digest for the associated information: e.g. Related document 1: section 6.2 `Discovery Data Transfer Object (DTO)`.

2. A means of verifying the integrity of an event using the shared token.

The first function of the Message Integrity Service requires the user/device/behaviour to be authenticated, and so it may be collocated with the Authentication Service. The second function, is however in-band of the message flow; each event must be verified each time it is processed, even without an authenticated user. Ideally, this service needs to be embedded in each BRAIN-IoT node, or at least distributed across a number of nodes. Moreover, as both of these functions will be used regularly from Brain IoT code, it is necessary for them to be exposed as OSGi services.

To achieve key function 1, a new endpoint for this service will be developed for the Authentication provider. To achieve the second one, a new Message Token verification service is required; as this may need to be embedded in each participating BRAIN-IoT node and so should be implemented in OSGi / Java.

3.3 The Authorization Service

The Authorization Service is responsible for determining the permissions for a given authenticated id.

It will use the security token to verify users/devices/behaviours identity and establish whether or not the request is legitimate.

This may be used, for example, to:

- Establish whether a Smart Behaviour is permitted to receive a given data event produced by the named id
- Establish whether a device or Smart Behaviour is permitted to generate a specific data event
- Establish whether the incoming event is permitted to trigger the named action on a device.

Authorization is performed on each node before events are provided to application code or devices. In order to achieve sensible performance, it is expected that permissions will be cached locally, and that the Authorization Service Backend will push out invalidation notifications to each node when permissions change. To achieve this, the authorization engine/cache must be implemented as a Java / OSGi bundle and will be an integral part of each Brain-IoT node.

3.4 The Identity Administration Service

Identity administration is the process by which new identities are added to the Authentication Service and then associated with permissions in the Authorization Service.

Identity Administration is typically an out of band process performed by an operator when adding new users or devices, however some BRAIN-IoT scenarios requires Smart Behaviours to be dynamically deployed in response to environmental triggers from the BRAIN-IoT environment.

It is proposed that a Smart Behaviour should acquire an identity derived from its runtime host. In this way the permissions can be configured both:

- with respect to the host's node (and therefore apply to all behaviours on the node),
- if required also for individual Smart Behaviours.

Coordination is required between Identity Administration Service (IAS) and the BRAIN-IoT modelling environment to determine which types of Smart Behaviours should have which of the available runtime privileges. One approach may be for the full set of `potential` Privileges to be made available to the modelling environment from IAS; the generated Smart Behaviour OSGi Bundle artefacts stating - as Requirements - the subset of privileges that a BRAIN-IoT node must have in order that it is a viable host for the Smart Behaviour.

The Identity Administration Service will be provided by the CymID Manager developed by Airbus which allows to:

- Manage user and devices rights and permissions
- Declare and configure Smart Behaviour
- Manage Identity Broker service configuration

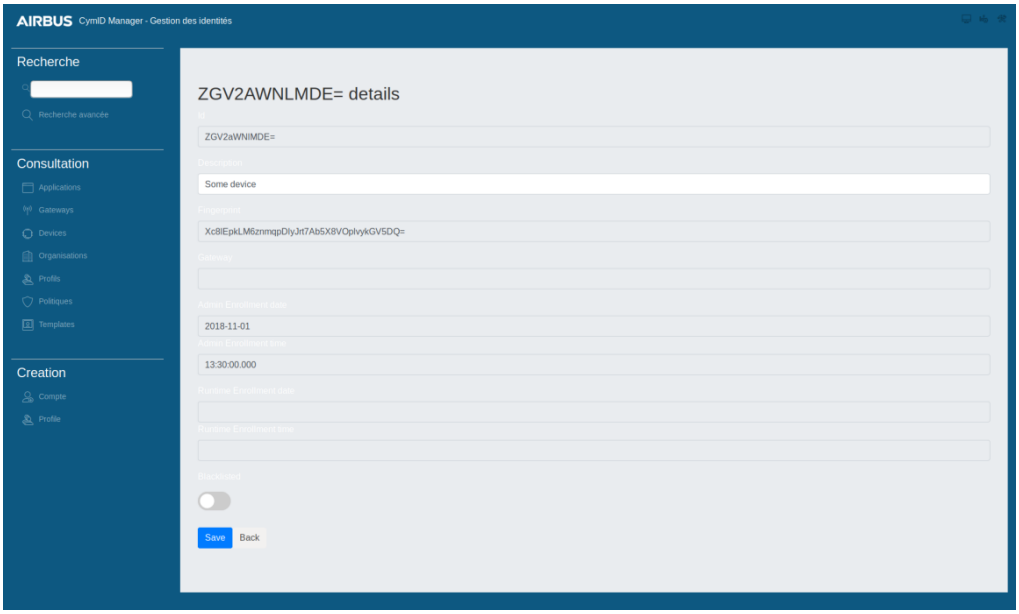

Figure 3: CymID Manager HMI

This application is available through 2 kinds of interfaces:

- Web REST API
- Web based user interface

CymID provides a REST based runtime service and is made available as a Docker Container Image artefact.

Developments are still on going on this component and will be finished for the next iteration.

The application also provides the enrolment of sensor device that is necessary for the use of the End-to-End data encryption solution developed by Airbus. This innovative solution based on Key derivation is integrated in a solution called MISNeT and will be demonstrated in Task 5.4 – End-to-end data security and provenance.

3.5 The auditing service

As each persisted event, include a security token, this generated from the data contained with the event (see Message Integrity Service - section 3.2), it is possible to associate all events with the user, device or Smart Behaviour that create them, and for the system to persistently log the actions that are taken.

Moreover, via this mechanism, it is possible to determine that the requested action or transmitted data has not been modified, nor its archived representation.

4 Future steps

This part describes the next actions to be performed to achieve the development and integration of the Brain-IoT security AAA layer.

- Development of the Message Integrity Service as a new Endpoint of the Authentication service.
- Develop a Message Token verification service that can be distributed across several, or embedded in all BRAIN-IoT nodes.
- Define local Authorization Cache API's and implement a simple `pass through` implementation. This will then allow the final local Authorization cache service to be developed in parallel with other activities; rather than blocking them.
- Integration of Airbus Docker components as Packager components: so components can be deployed and managed by each BRAIN-IoT Fabric.
- Define the relationship between BRAIN-IoT AAA runtime services and the modelling of the security aspects of BRAIN-IoT Smart Behaviours. This step can begin without completing the previous one.
- Proof of concept without Authorization cache: Integration of the AAA layer within the Brain-IoT Fabric.
- Determine the approach via which modelled privileges in the BRAIN-IoT modelling environment can be mapped into a production Environment; either via the Identification Service; or directly into the Authentication and Authorization services.
- Development and integration of Authorization service cache as a Java OSGi Bundle.

5 Conclusion

This document described the state of the development of the AAA layer of the Brain-IoT platform.

The next step is to finish ongoing developments and integrates different components with each other.

The result of this work will be described in the next iteration of the document: D5.6 – Final AAA layer for IoT cross platform models

Acronyms

Acronym	Explanation
AAA	Authentication, Authorization and Accounting
API	Application Programming Interface
HMI	Human Machine Interface
IAS	Identity Administration Service
JWT	Java Web Token
MITM	Man-in-the-middle
TLS	Transport Layer Security

List of figures