Multi-layered Security Technologies

for hyper-connected smart cities

D4.10: M-Sec overall end-to-end security

March 2021

Grant Agreement No. 814917

Multi-layered Security technologies to ensure hyper-connected smart cities with Blockchain, BigData, Cloud and IoT

Project acronym	M-Sec
Deliverable	D4.10 M-Sec overall end-to-end security
Work Package	WP4
Submission date	31 March 2021
Deliverable lead	Mathieu Gallissot (CEA) / Akira Tsuge (KEIO)
Authors	Radhouene Azzabi (CEA), Mathieu Gallissot (CEA), Akira Tsuge (KEIO), Xavier Cases (WLI), Aamir Bokhari (YNU), Kenji Tei (WU), Takafumi Komoto (NII)
Internal reviewer	Orfefs Voutyras (ICCS) / Kenji Tei (WU)
Dissemination Level	Public
Type of deliverable	DEM

The M-Sec project is jointly funded by the European Union's Horizon 2020 research and innovation programme (contract No 814917) and by the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN (contract No. 19501).

Version history

#	Date	Authors (Organization)	Changes
v0.1	22 February 2021	Mathieu Gallissot (CEA)	Full ToC and assignments
v0.2	10 March 2021	Mathieu Gallissot (CEA)	Section 1, 3 content provided
v0.3	23 March 2021	Mathieu Gallissot (CEA)	Section 2 updated
v0.4	25 March 2021	Kenji Tei (WU)	Internal Review
v0.5	26 March 2021	Orfefs Voutyras (ICCS)	Internal Review
v0.6	29 March	Mathieu Gallissot (CEA)	Integrating review comments
v1.0	31 March 2020	Mathieu Gallissot (CEA)	Version ready for submission

Table of Contents

List of Figures

Glossary

Acronym	Description	Acronym	Description
AAA	Authorization, Authentication and Accounting	OCST	Online Certificate Status Protocol
API	Application Programming Interface	OIDC	OpenID Connect
CA	Certification Authority	OT	Operational Technology
CLI	Command Line Interface	OTP	One Time Password
CoAP	Constrained Application Protocol	P2P	Peer-to-Peer
CSR	Certificate Signing Request	PKI	Public Key Infrastructure
DNS	Domain Name System	REST	Representational state transfer
Dx.y	Deliverable y of WP x	RPC	Remote procedure call
FG	Functional Group	SASL	Simple Authentication and Security Layer
FQDN	Fully Qualified Domain Name	SSH	Secure Shell
IETF	Internet Engineering Task Force	SSL	Secure Session Layer
IT	Information Technology	SSO	Single Sign On
JSON	Javascript Object Notation	SSSD	System Security Services Deamon
LDAP	Lightweight Directory Access Protocol	TLS	Transport Layer Security
MQTT	Message Queuing Telemetry Transport	Tx.y	Task y of WP x
NIST	National Institute of Standards and Technology	UC	Use Case
NTP	Network Time Protocol		

Executive Summary

The work described in this deliverable (D4.10) was carried out in the framework of WP4 – “Multi-layered Security Technologies”, and more specifically, in the framework of T4.5 – “Overall End-to-End Security”. The report presents the updated and final version of the document (the first version being D4.9), providing the technical details of the Functional Group and Functional Components related to the Task.

All technical partners involved in this task collaborated and developed the appropriate tools to meet the objectives set out in the project, especially with regard to novel Security aspects in IoT contexts. Every partner focuses on the individual modules that they are responsible for during the implementation phase of WP4 and supports the integration activities of WP2, while following the common Architecture framework set by WP3 in D3.4.

All of the updated versions of the WP4 technical deliverables (D4.2, D4.4, D4.6, D4.8, D4.10) follow the same approach and have the same structure. Section 1 provides an introduction to the scope of this document and its relation with other WPs and Tasks. Section2, which aggregates all the main outcomes of the Task, presents extensively the FG and the Functional Components covered by the Task, by providing an extensive description of the corresponding functionalities, and details related to the API of the FG and its interactions with other FGs of the M-Sec solution. Finally, Section 3 concludes the document.

Regarding the differences between ‘D4.9 M-Sec Overall end-to-end Security – first version’ and ‘D4.10 M-Sec Overall end-to-end Security – final version’:

- WP4 deliverables in their final version are re-organized to follow the newest version of the M-Sec architecture from WP3. In particular, this document it introduces the functional groups that are implemented by partner assets.
- M-Sec middleware, namely SOXFire and sensiNact, has been integrated in the “Secure City Data Access” functional group, which is detailed in deliverable 4.4.

All in all, the deliverable is considered to have provided all of the information required to expose the M-Sec technical solutions related to T4.5 as well as the results of the integration and demonstration related activities.

1. Introduction

1.1 Scope of the document

We present in this document the reference design demonstrating M-Sec end-to-end security as worked out in task 4.5. End-to-end Security has a particular approach in the M-Sec Project and WP4 as it is a combination of the output from the four other tasks (Task 4.1, 4.2, 4.3, and 4.4), completed by a security manager, ensuring a secured and smooth interoperation of each element of the architecture. The functions provided by this security manager interface with the other functional groups and assets to provide a consistent security backend.

1.2 Relationship to other work packages and tasks

The following figure summarises the relations of this deliverable (and the corresponding task) to other tasks and WPs.

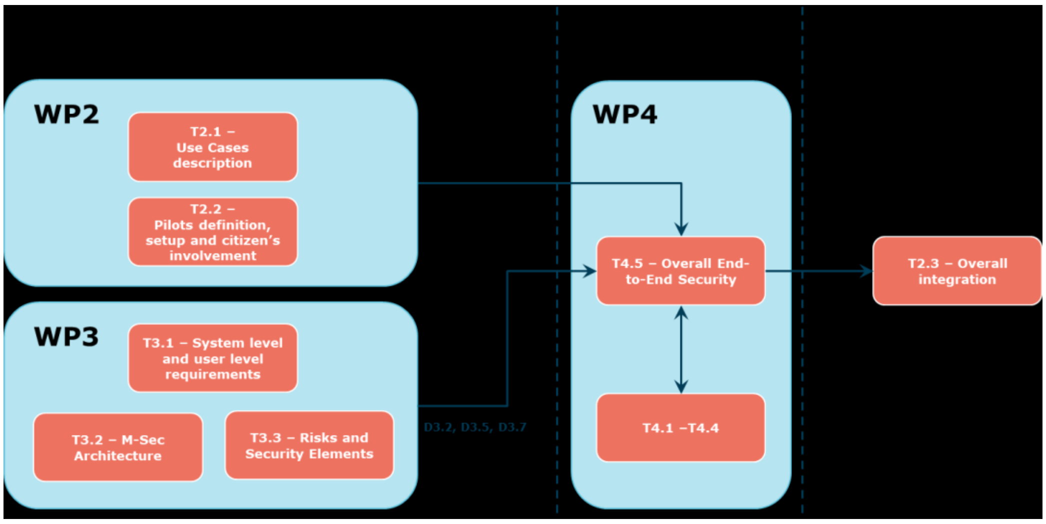

WP2
T2.1 – Use Cases description
T2.2 – Pilots definition, setup and citizen's involvement
WP3
T3.1 – System level and user level requirements
T3.2 – M-Sec Architecture
T3.3 – Risks and Security Elements
D3.2, D3.5, D3.7
WP4
T4.5 – Overall End-to-End Security
T4.1 –T4.4
T2.3 – Overall integration

Figure 1. T4.5 and D4.10 relation to other WPs and Tasks

The work done in Task 4.5 is directly related to WP3. T4.5 receives as input system and user requirements from T3.1 and Risks- and Threats-related information from T3.2. Moreover, it follows the common Architectural framework that has been identified in T3.2 for the coordination of all the technical activities. Similarly, the Task receives input from WP2 related to the coverage of the needs of the UCs and the pilots.

The deliverable D4.10 is strongly related to Tasks 4.1 (IoT Security), 4.2 (Cloud and data-level security), 4.3 (P2P level security and blockchains), and 4.4 (Application-level security) as also shown in the Annex. It serves all other functional groups with a security backend enabling homogenous accounting and other security features that are required for overall security management.

Finally, the results of this report are directly provided as input to T2.3 which is focusing on the overall integration activities. Together with the other final deliverables of WP4, D4.10 provides all the information and functionalities required for an integrated security solution.

1.3 Methodology followed

The M-Sec WP4 is built upon an IoT reference model, having in our case four distinct domains: IoT, Data and Cloud, P2P with Blockchains, and Applications. Each one of these domains is securing itself by integrating state-of-the-art technologies. Such technologies are being worked out in other WP4 tasks. Thus, having end-to-end security requires having common functions to ensure a continuum within many aspects such as authorization, authentication, anonymization, attestations, etc.

The main approach of task 4.5 is to act as a backend for the other domains, meaning mutualizing security functions and providing security awareness between each layer in order to enable a cyber-resilient framework. The technological risk and appropriate countermeasures are inherited from each specific layer of the project. End-to-end security fills the gap regarding common security funcitons to cover "non-technological risks" such as human errors, misconfigurations and organizational differences between potentially heterogenous environments.

In more details, completing the methodologies inherited from each layer bound to the risk assessement and arhcitecture definition, methodology for end-to-end security is to follow NIST Framework which is presented in section 2. This approach allows us to have a framework facilitating the evolution of the countermeasures implemented with regard to new threats to consider an evolving level of security, especially after the project.

2. End-to-end security Functional Group

2.1 Description

The end-to-end security functional group aims to provide a global security backend for mixed OT/IT large infrastructures. It needs to emerge from usually inconsistent security management between these two paradigms:

- In IT (Information Technology) access control is usually manageable, as well as inventories and so on. The freshness of security controls can be performed with equipment renewal.
- In OT (Operational Technology) access control is not manageable, and devices may last for 20 years or more so the freshness of security (encryption algorithm and strength) may not be achieved easily.

The security manager philosophy is to provide interaction between existing layers in the spirit of the NIST Cybersecurity Framework[1] as described in Figure 2. This framework described five interdependent actions which altogether ensure a secure and resilient system:

- ***Identify threats***: with threat and risk analysis (as presented in deliverable 3.5), the goal is to identify the exposure of the system and consequences of the most meaningful and critical attack
- ***Protect:*** using cybersecurity and privacy technologies to reduce the likelihood and criticality of the risk
- ***Detect:*** continuously observe the system thru metrics adapted to the threat and risks in order to
- ***Respond:*** execute a response plan to stop and limit the impact of an ongoing undesired event
- ***Recover:*** restore the full capability of the system

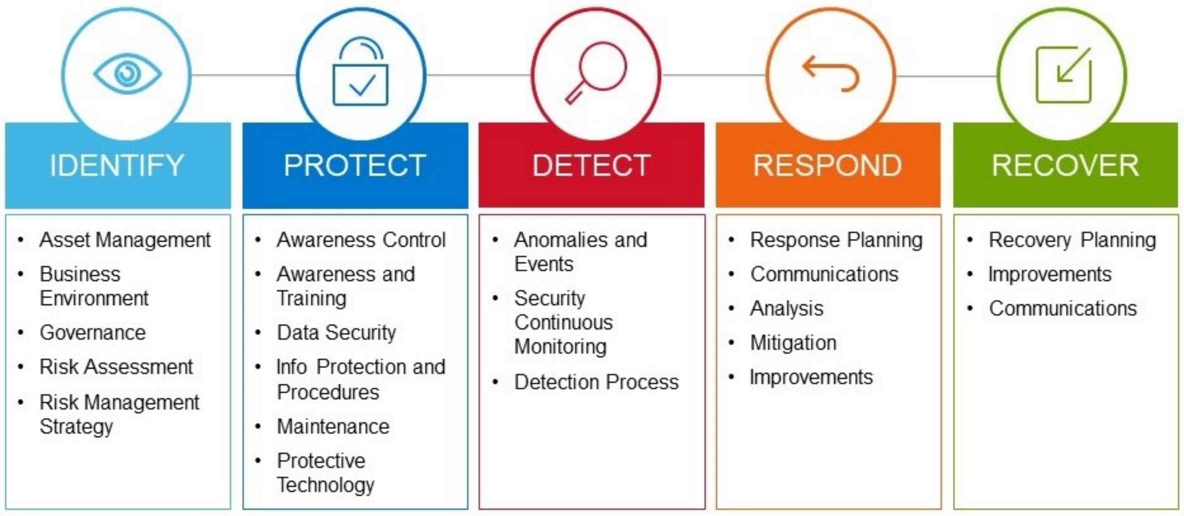


Figure 2. NIST Cybersecurity framework

[1] https://www.nist.gov/cyberframework

M-Sec assets provide in majority means of protection thru different cybersecurity technologies at different levels.

The functional group is composed of three mains function:

- A **Public Key Infrastructure** which ensures end-to-end security by providing asymmetric encryption capabilities
- A **Directory service** that manages Accounting, Authorization, and Authentication for either device, infrastructure, or cloud elements as developed in other tasks
- An **Identity Federation** module, which enables the inclusion of the citizen in the security management

2.2 Components

Security manager

The security manager is composed of three logical layers as illustrated in Figure 3:

- The *security layer* contains the security components like FreeIPA for accounting, Keycloak for identity management, and the M-Sec Manager API.
- The *routing layer* contains a reverse-proxy component (e.g., Nginx) that distributes the load from incoming requests to the right local servers.
- The *applications layer*, which is an optional layer, contains the set of application servers.

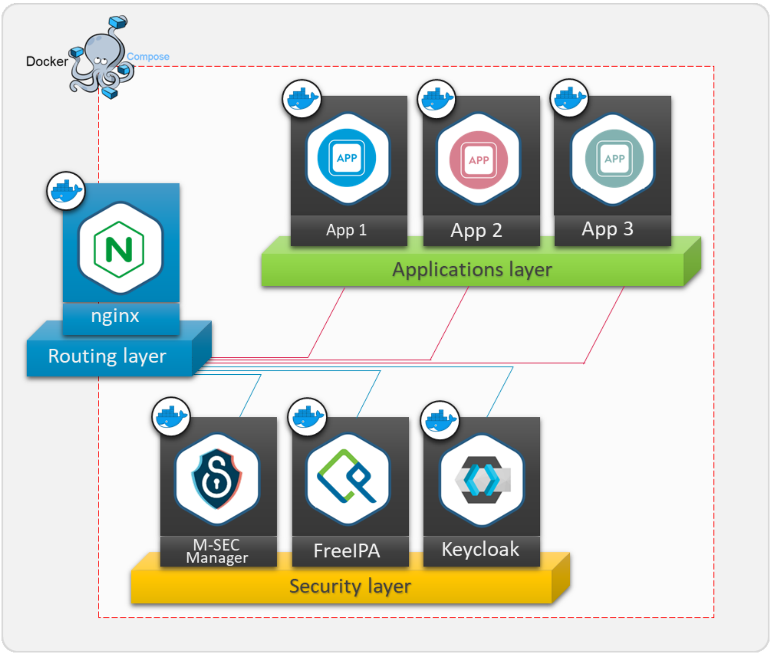
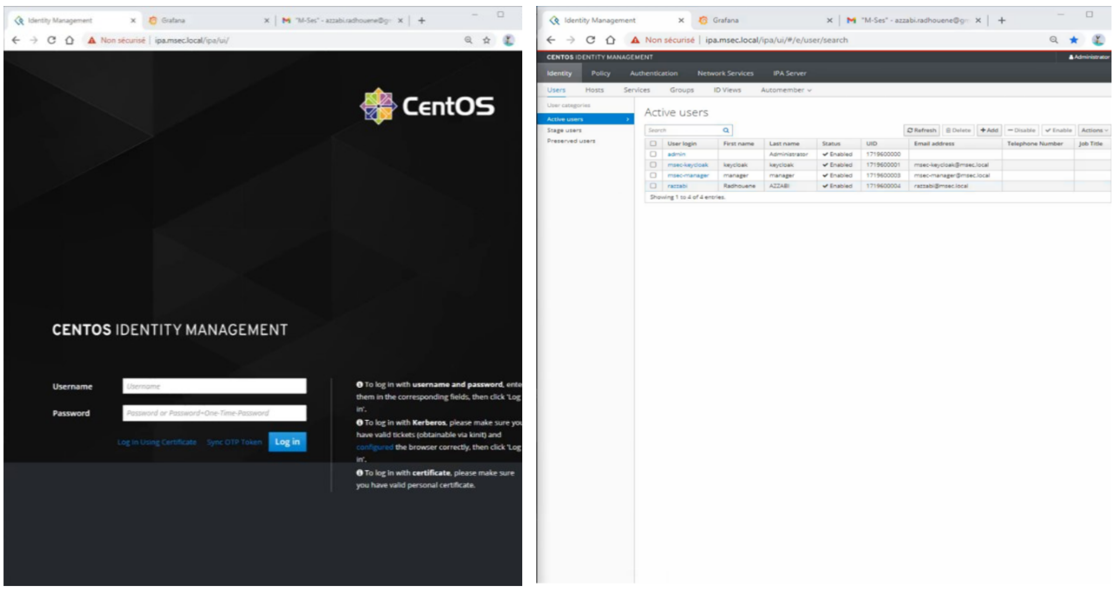


Figure 3. Security Manager inner-layers

FreeIPA

FreeIPA is an integrated **Identity** and **Authentication** solution for **Linux/UNIX** networked environments. FreeIPA server provides **centralized authentication**, **authorization,** and account information by storing data about **user, groups, hosts,** and other objects necessary to manage the security aspects of a network of computers.

FreeIPA offers services such as ***Directory Server, Kerberos, NTP, DNS, Dogtag (Certificate System).*** It can be managed via a Web UI and CLI administration tools. Its management interface is provided in Figure 4



CENTOS IDENTITY MANAGEMENT

To log in with **username and password**, ente them in the corresponding fields, then click 'Log in'.

To log in with **Kerberos**, please make sure you have valid tickets (obtainable via kinit) and configured the browser correctly, then click 'Log in'.

To log in with **certificate**, please make sure you have valid personal certificate.

Active users

	User login	First name	Last name	Status	UID	Email address	Telephone Number	Job Title
☐	admin		Administrator	✓ Enabled	1719600000			
☐	msec-keycloak	keycloak	keycloak	✓ Enabled	1719600001	msec-keycloak@msec.local		
☐	msec-manager	manager	manager	✓ Enabled	1719600003	msec-manager@msec.local		
☐	razzabi	Radhouene	AZZABI	✓ Enabled	1719600004	razzabi@msec.local		

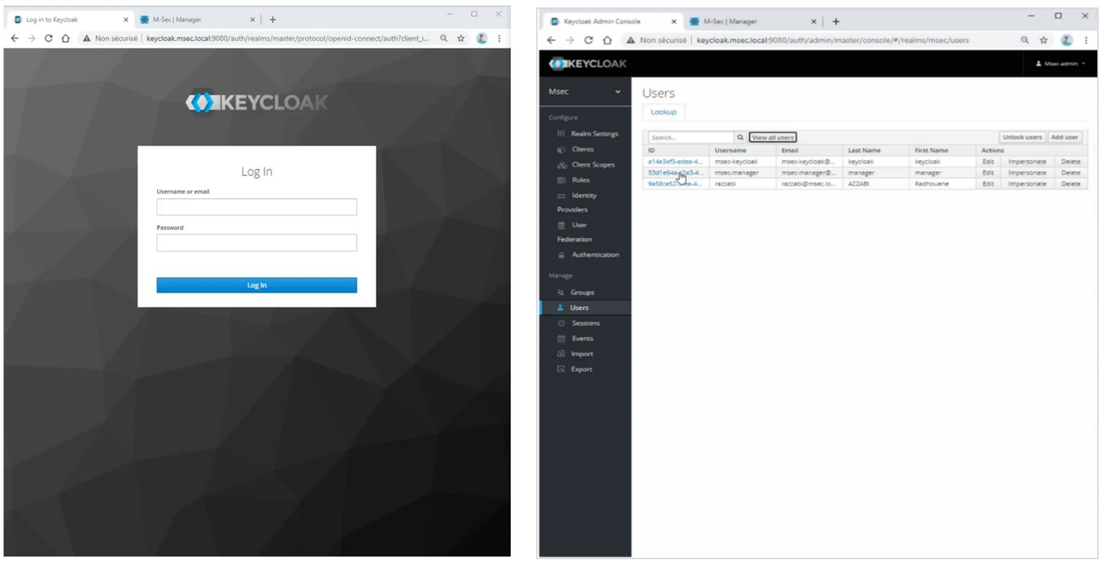
Showing 1 to 4 of 4 entries.

Figure 4. FreeIPA web-based user interface

KeyCloak

Keycloak is an open-source **Identity and Access Management** solution targeted towards modern applications and services. Users can authenticate with Keycloak rather than individual applications.

Keycloak offers features such as Single-Sign-On (SSO), Identity Brokering (**openID** and **Oauth2**) and **Social Login**, **User Federation**, Client Adapters, an Admin Console, and an Account Management Console as provided in Figure 5



ID	Username	Email	Last Name	First Name	Actions		
a14a3ef5-edee-4...	msec-keycloak	msec-keycloak@...	keycloak	keycloak	Edit	Impersonate	Delete
53d1e64a-[illegible]2a3-4...	msec-manager	msec-manager@...	manager	manager	Edit	Impersonate	Delete
9a58ce52-[illegible]-4...	razzabi	razzabi@msec.lo...	AZZABI	Radhouene	Edit	Impersonate	Delete

Figure 5. Keycloak web-based frontend

M-Sec Manager

The M-Sec manager relies on previously cited components such as KeyCloack and FreeIPA to provide a security backend for IoT infrastructures with enrolment and lifecycle management. The M-Sec Manager component is composed by a server and client-side as in Figure 6:

- The client-side (front-end in the picture follow) act as an HIM interface developed with ReactJS and secured via a Keycloak wrapper. The M-Sec Manager Front-End is only used for demo purposes.
- The server-side is designed as a REST-Full API, secured via Keycloak, and allowing to interact with Cert Management Service (available via DogDag from FreeIPA), with Host management service (via FreeIPA), and finally to configure **Secure Devices** and execute the TPMs remote attestation mechanism.

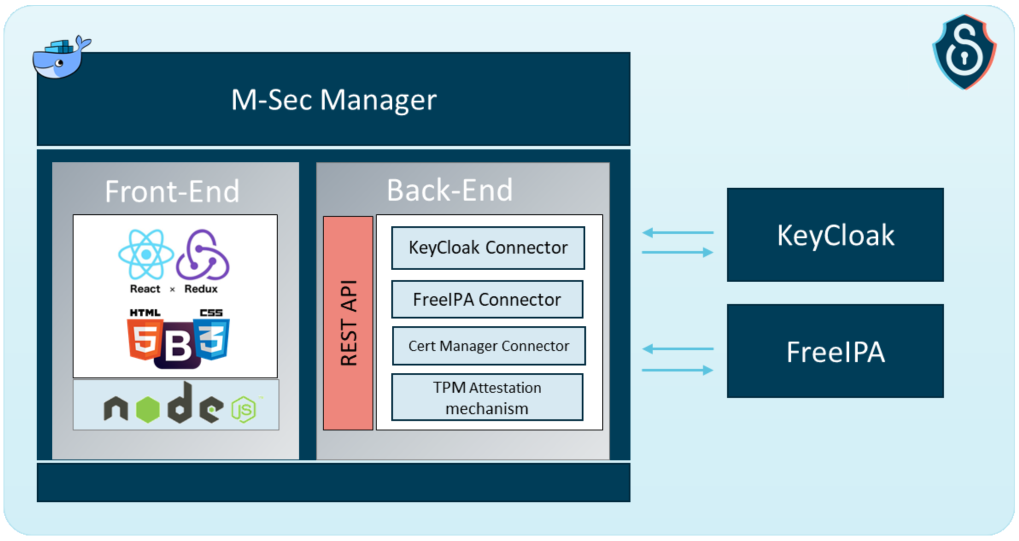


Figure 6. M-Sec Manager inner architecture

2.3 APIs

The philosophy regarding how end-to-end security interacts and behaves towards other functional groups and clients is to follow existing standards as long as they exist. One reason is that these standards benefit from the design, audits, and reference implementation that facilitates integration and therefore ensures regular security updates.

Figure 7 shows the three main modules of the security manager aligned with the M-Sec layers and dominant standards and API given each layer.

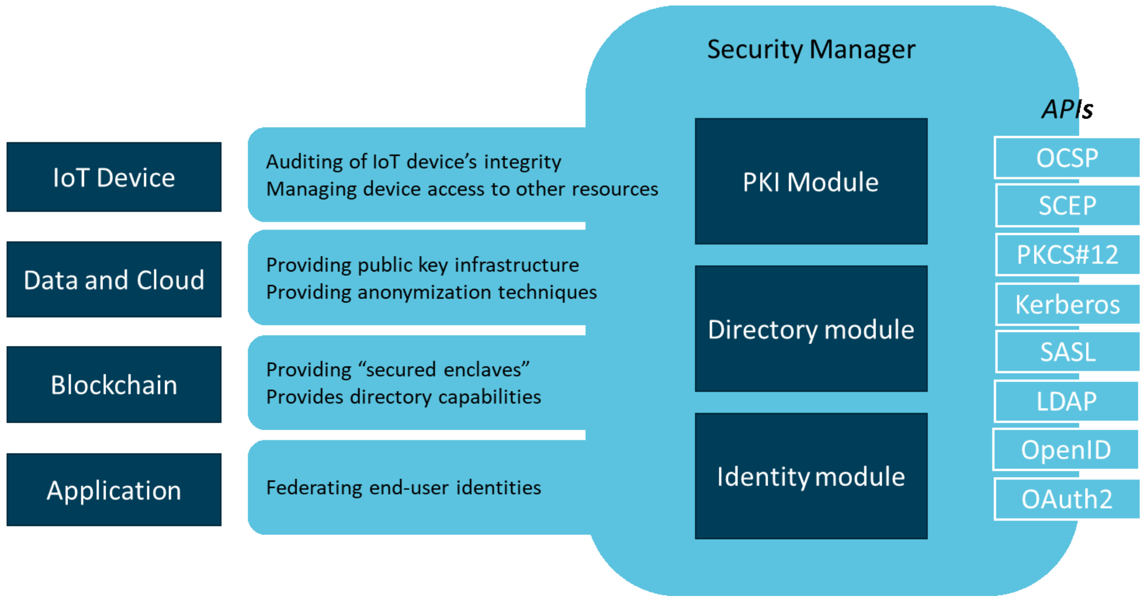

Security Manager
APIs
IoT Device
Data and Cloud
Blockchain
Application
Auditing of IoT device's integrity
Managing device access to other resources
Providing public key infrastructure
Providing anonymization techniques
Providing "secured enclaves"
Provides directory capabilities
Federating end-user identities
PKI Module
Directory module
Identity module
OCSP
SCEP
PKCS#12
Kerberos
SASL
LDAP
OpenID
OAuth2

Figure 7. Architecture of the security manager with relationships to M-Sec layers

API for certificate management

This functional group used the following standardized APIs:

Online Certificate Status Protocol: OCSP is used mostly to verify at runtime the validity of a certificate, in particular, it enables to verify if the client certificate has been put on a revocation list by an authority. This online verification is often referred to as "OCSP Stapling".

On the famous apache2 web server, it can be enabled using the following line within the configuration file.

```
SSLUseStapling on
```

On the Nginx web server, the configuration directive is

```
ssl_stapling on;
ssl_stapling_verify on;
```

API for accounting

Accounting is proposed using the LDAP with SASL authentication. Accounting and user authentication are managed via FreeIPA Framework. FreeIPA is an open-source security solution for Linux which is built on top of multiple open source projects, including the LDAP (389 Directory Server), MIT Kerberos, and SSSD.

FreeIPA has clients for CentOS 7, Fedora, and Ubuntu. These clients make it fairly straightforward to add machines into your IPA domain.

There are two ways to enrol a machine into the IPA domain:

a) using FreeIPA client package (usage with a valid domain admin user)
b) using M-Sec manager API (usage with a non-domain authenticated user)

Enrolling host machine using FreeIPA API

To enrol a new machine into the IPA domain, we need firstly to prepare the machine, install some requirements as a freeIPA client package, and then we will start enrolment. Once enrolled, the domain administrator will be able to manage which users and groups may log into this machine (using bash or ssh) and which users can use sudo. Figure 8 shows the enrolment workflow.

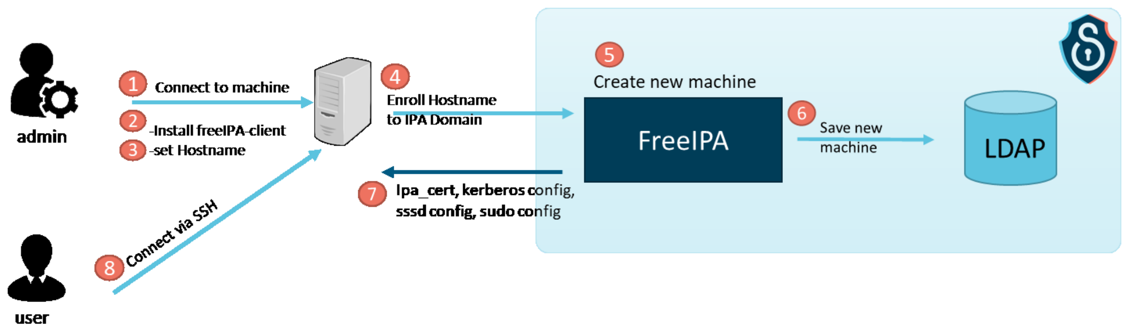


Figure 8. Devices enrollment in FreeIPA workflow

To begin, the hostname of the new machine needs to match with the fully qualified domain name (FQDN), for example, new_machine.msec.local with msec.local is our domain.

```
# hostname new_machine.msec.local
```

Change the hostname on the /etc/hosts

```
# vi /etc/hosts
new_machine.msec.local
```

Then save and close the file.

Once the hostname is set correctly, update the package repositories.

```
# apt-get update
```

Install the FreeIPA Client

```
# apt-get install freeipa-client
# ipa-client-install --domain=msec.local --hostname=new_machine.msec.local –
server=ipa.msec.local –p admin --mkhomedir –force-join
```

The –mkhomedir flag tells FreeIPA to create home directories for IPA users when they login to the machine for the first time

Finally, enter the password for your IPA admin user. This was set during the FreeIPA server configuration. After entering the password, the FreeIPA client will configure the system. The output will be ***Client configuration complete***. This indicates a successful installation.

Enrolling host machine using M-Sec Manager API

Only authorized users are allowed to enrol hosts into the IPA domain. User/entity must have a valid Kerberos principal to access and update the FreeIPA and LDAP store.

To allow a non-domain user to enrol machines into the IPA domain, we added a two-factor authentication mechanism that allows users to enroll hosts into the IPA domain only if there is a valid **access_token** (generated after an OpenID authentication), and a valid OTP generated by the FreeIPA Framework for a given hostname.

There are three components involved in this flow as shown in Figure 9:

- The Keycloak server used to authenticate users using OpenID
- M-Sec Manager API used to verify user's access_token and request an OTP
- FreeIPA server used to enroll and manage hosts

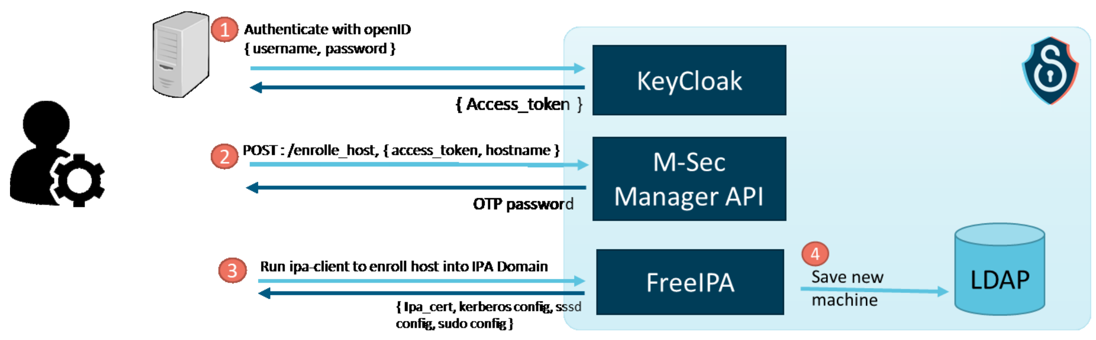

1 Authenticate with openID
{ username, password }
{ Access_token }
KeyCloak
2 POST : /enrolle_host, { access_token, hostname }
OTP password
M-Sec Manager API
3 Run ipa-client to enroll host into IPA Domain
{ Ipa_cert, kerberos config, sssd config, sudo config }
FreeIPA
4 Save new machine
LDAP

Figure 9. Devices enrollment within the three modules workflow

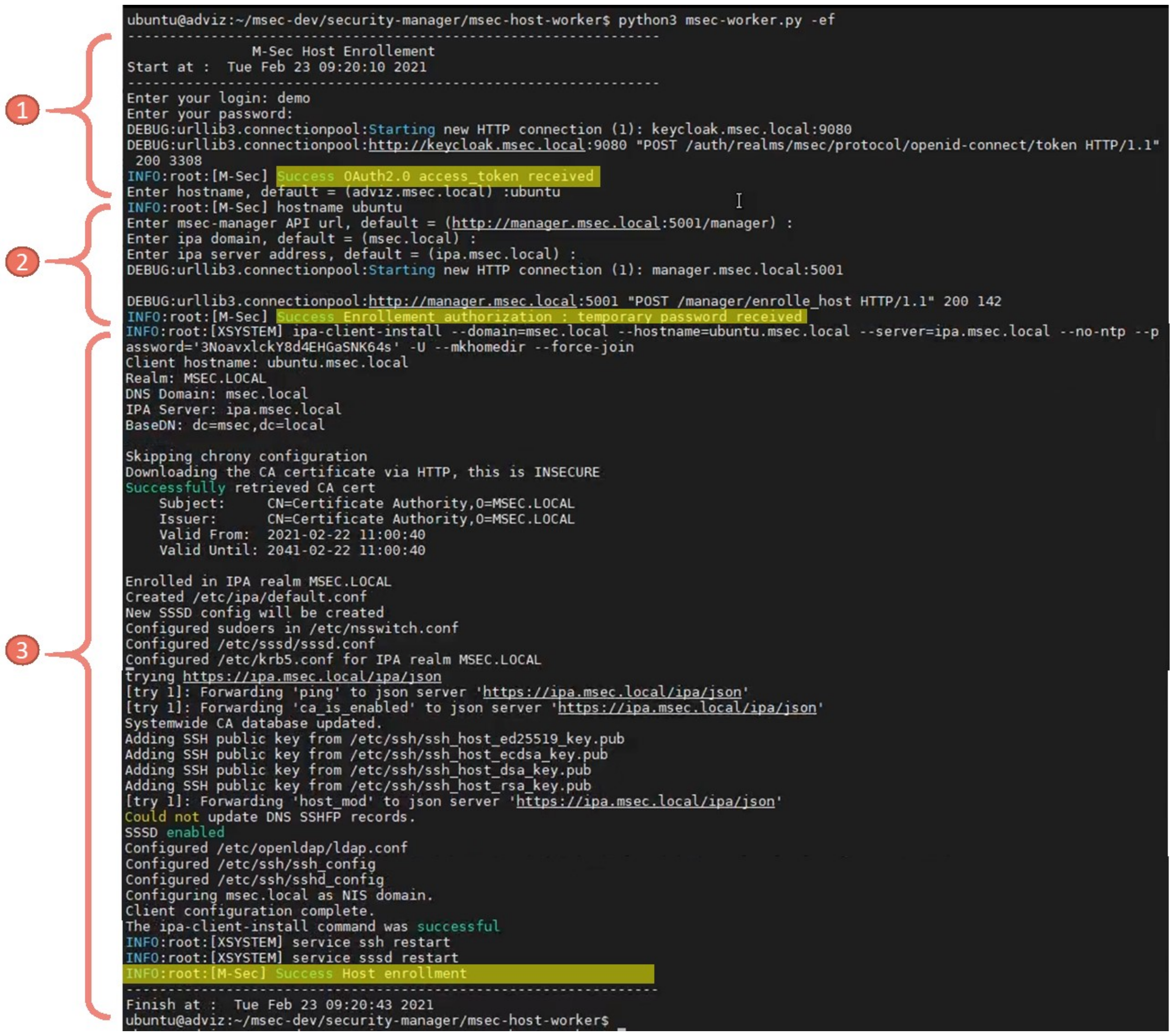
To execute this flow, the user needs to install the **M-Sec Worker** package, which contains a python script automatizing the host enrolment process.

The **M-Sec Worker** script is used with the -ef option for force-enrolment.

Once executed, the user will be prompt to set (1) authentication credentials (username and password), the hostname, and define some URIs (Keycloak, M-Sec Manager API, IPA server).

The **M-Sec Worker** script requests an **access_token** from Keycloak using the username and the password as defined in step 1. Then, if authentication success, an **access_token** will be generated and used to exchange with **M-Sec Manager API**.

In step (2) **M-Sec Worker** script will request an OTP from the **M-Sec Manager API** and use it in step (3) to run the host enrolment process. These steps are illustrated in Figure 10.



```
ubuntu@adviz:~/msec-dev/security-manager/msec-host-worker$ python3 msec-worker.py -ef
----------------------------------------------------------------
               M-Sec Host Enrollement
Start at :  Tue Feb 23 09:20:10 2021
----------------------------------------------------------------
Enter your login: demo
Enter your password:
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): keycloak.msec.local:9080
DEBUG:urllib3.connectionpool:http://keycloak.msec.local:9080 "POST /auth/realms/msec/protocol/openid-connect/token HTTP/1.1"
 200 3308
INFO:root:[M-Sec] Success OAuth2.0 access_token received
Enter hostname, default = (adviz.msec.local) :ubuntu
INFO:root:[M-Sec] hostname ubuntu
Enter msec-manager API url, default = (http://manager.msec.local:5001/manager) :
Enter ipa domain, default = (msec.local) :
Enter ipa server address, default = (ipa.msec.local) :
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): manager.msec.local:5001

DEBUG:urllib3.connectionpool:http://manager.msec.local:5001 "POST /manager/enrolle_host HTTP/1.1" 200 142
INFO:root:[M-Sec] Success Enrollement authorization : temporary password received
INFO:root:[XSYSTEM] ipa-client-install --domain=msec.local --hostname=ubuntu.msec.local --server=ipa.msec.local --no-ntp --p
assword='3NoavxlckY8d4EHGaSNK64s' -U --mkhomedir --force-join
Client hostname: ubuntu.msec.local
Realm: MSEC.LOCAL
DNS Domain: msec.local
IPA Server: ipa.msec.local
BaseDN: dc=msec,dc=local

Skipping chrony configuration
Downloading the CA certificate via HTTP, this is INSECURE
Successfully retrieved CA cert
    Subject:     CN=Certificate Authority,O=MSEC.LOCAL
    Issuer:      CN=Certificate Authority,O=MSEC.LOCAL
    Valid From:  2021-02-22 11:00:40
    Valid Until: 2041-02-22 11:00:40

Enrolled in IPA realm MSEC.LOCAL
Created /etc/ipa/default.conf
New SSSD config will be created
Configured sudoers in /etc/nsswitch.conf
Configured /etc/sssd/sssd.conf
Configured /etc/krb5.conf for IPA realm MSEC.LOCAL
trying https://ipa.msec.local/ipa/json
[try 1]: Forwarding 'ping' to json server 'https://ipa.msec.local/ipa/json'
[try 1]: Forwarding 'ca_is_enabled' to json server 'https://ipa.msec.local/ipa/json'
Systemwide CA database updated.
Adding SSH public key from /etc/ssh/ssh_host_ed25519_key.pub
Adding SSH public key from /etc/ssh/ssh_host_ecdsa_key.pub
Adding SSH public key from /etc/ssh/ssh_host_dsa_key.pub
Adding SSH public key from /etc/ssh/ssh_host_rsa_key.pub
[try 1]: Forwarding 'host_mod' to json server 'https://ipa.msec.local/ipa/json'
Could not update DNS SSHFP records.
SSSD enabled
Configured /etc/openldap/ldap.conf
Configured /etc/ssh/ssh_config
Configured /etc/ssh/sshd_config
Configuring msec.local as NIS domain.
Client configuration complete.
The ipa-client-install command was successful
INFO:root:[XSYSTEM] service ssh restart
INFO:root:[XSYSTEM] service sssd restart
INFO:root:[M-Sec] Success Host enrollment
----------------------------------------------------------------
Finish at :  Tue Feb 23 09:20:43 2021
ubuntu@adviz:~/msec-dev/security-manager/msec-host-worker$ _
```

Figure 10. Enrolment script output

API for identity federation

The security manager supports OAuth2 and OpenID, thanks to the Keycloak component.

Keycloak is a Java-based open-source Identity and Access Management solution. It supports both **OAuth 2.0** and **OpenID**. It also offers features like **identity brokering**, **user federation**, and SSO.

OAuth 2.0

OAuth 2.0 is an authorization framework that lets an authenticated user grant access to third parties via tokens. A token is usually limited to some scopes with a limited lifetime. Therefore, it's a safe alternative to the user's credentials.

OAuth 2.0 comes with four main components:

- ***Resource Owner*** – the end-user or a system that owns a protected resource or data
- ***Resource Server*** – the service exposes a protected resource usually through an HTTP-based API
- ***Client*** – calls the protected resource on behalf of the resource owner
- ***Authorization Server*** – issues an OAuth 2.0 token and delivers it to the client after authenticating the resource owner

OpenID Connect

OpenID Connect (OIDC) is built on top of OAuth 2.0 to add an identity management layer to the protocol. Hence, it allows clients to verify the end user's identity and access basic profile information via a standard OAuth 2.0 flow. OIDC has introduced a few standard scopes to OAuth 2.0, like OpenID, profile, and email.

Identity Brokering

An Identity Broker is an intermediary service that connects multiple service providers with different identity providers. As an intermediary service, the identity broker is responsible for creating a trust relationship with an external identity provider to use its identities to access internal services exposed by service providers.

From a user perspective, an identity broker provides a centralized way to manage identities across different security domains or realms. An existing account can be linked with one or more identities from different identity providers or even created based on the identity information obtained from them.

An identity provider is usually based on a specific protocol that is used to authenticate and communicate authentication and authorization information to their users. It can be a social provider such as Facebook, Google, or Twitter.

Identity federation

Keycloak can federate existing external user databases. Keycloak supports LDAP and Active Directory, as user storage providers. When a user logs in, Keycloak will look into its internal user store to find the user. If it can’t find it there, it will iterate over every user storage provider configured until it finds a match. Figure 11 shows the synchronization model between user identities in Keycloak and FreeIPA.

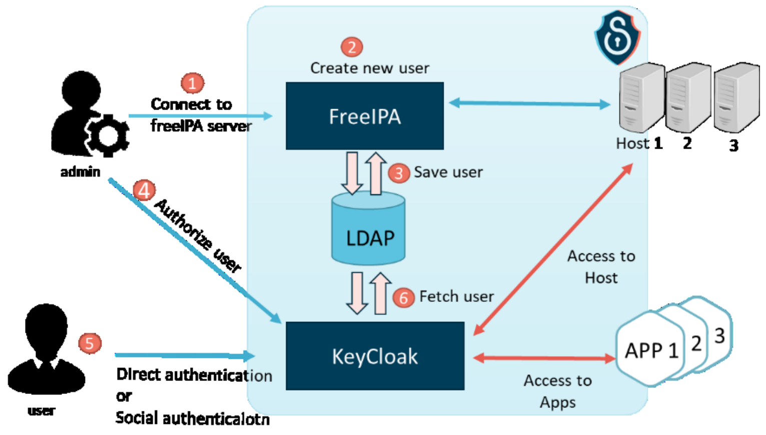


Figure 11. Synchronization of user accounts between FreeIPA and keycloak using LDAP as a backend

Keycloak API

Keycloak exposes a variety of REST endpoints for OAuth 2.0 flows.

OpenID Configuration Endpoint

The configuration endpoint is like the root directory. It returns all other available endpoints, supported scopes and claims, and signing algorithms.

```
GET: {{server}}/auth/realms/{{realm}}/.well-known/openid-configuration.
```

```
Resp :
```

```
{
    "issuer": "http://localhost:8083/auth/realms/msec",
    "authorization_endpoint": "http://localhost:8083/auth/realms/msec/protocol/openid-connect/auth",
    "token_endpoint": "http://localhost:8083/auth/realms/msec/protocol/openid-connect/token",
    "token_introspection_endpoint": "http://localhost:8083/auth/realms/msec/protocol/openid-
connect/token/introspect",
    "userinfo_endpoint": "http://localhost:8083/auth/realms/msec/protocol/openid-connect/userinfo",
    "end_session_endpoint": "http://localhost:8083/auth/realms/msec/protocol/openid-connect/logout",
    "jwks_uri": "http://localhost:8083/auth/realms/msec/protocol/openid-connect/certs",
    "check_session_iframe": "http://localhost:8083/auth/realms/msec/protocol/openid-connect/login-status-
iframe.html",
    "grant_types_supported": [...],
    ...
    "registration_endpoint": "http://localhost:8083/auth/realms/msec/clients-registrations/openid-connect",
    ...
    "introspection_endpoint": "http://localhost:8083/auth/realms/msec/protocol/openid-connect/token/introspect"
}
```

Token Endpoint

The token endpoint allows retrieving an access token, refresh token, or id token. OAuth 2.0 supports different grant types, like authorization_code, refresh_token, or password.

```
POST: {{server}}/auth/realms/{{realm}}/protocol/openid-connect/token
```

```
Body : {
  "username ": "demo",
  "password" : "demo"
  "client_id" : "msec",
  "client_secret" : "msec-secret",
  "grant_type" : "password"
}
```

Resp :

```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsINVSHGhepnDu13SwRBL-v-y-04_6e6IJbMzreZwPI-epwdVPQe
    "expires_in": 300,
    "refresh_expires_in": 1800,
    "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJRRnB5YlloMGVEektId
    "token_type": "bearer",
    "not-before-policy": 0,
    "session_state": "bb1c586a-e880-4b96-ac16-30e42c0f46dc"
}
```

Token Introspect Endpoint

The token introspect endpoint is used when a resource server needs to verify that an access token is active or wants more metadata about it.

```
POST : {{server}}/auth/realms/{{realm}}/protocol/openid-connect/token/introspect
```

```
Body : {
  "token ": "access_token herer",
  "client_id" : "msec",
  "client_secret" : "msec-secret",
}
```

Resp :

```
{
    "exp": 1601824811,
    "iat": 1601824511,
    "jti": "d5a4831d-7236-4686-a17b-784cd8b5805d",
    "iss": "http://localhost:8083/auth/realms/msec",
    "sub": "a5461470-33eb-4b2d-82d4-b0484e96ad7f",
    "typ": "Bearer",
    "azp": "jwtClient",
    "session_state": "96030af2-1e48-4243-ba0b-dd4980c6e8fd",
    "preferred_username": "demo@msec.local",
    "email_verified": false,
    "acr": "1",
    "scope": "profile email read",
    "DOB": "1987-11-24",
    "organization": "msec",
    "client_id": "msec",
    "username": "demo@msec.local",
    "active": true
}
```

User Information Endpoint

The user information endpoint allows retrieving user profile data such as first name, Lastname, email, organization, group

User information Endpoint requires an access_token

```
GET : {{server}}/auth/realms/{{realm}}/protocol/openid-connect/userinfo
```

Resp :

```
{
    "sub": "a5461470-33eb-4b2d-82d4-b0484e96ad7f",
    "preferred_username": "demo@msec.local",
    "DOB": "1987-11-24",
    "organization": "msec"
}
```

API for lifecycle management

Lifecycle management is provided by a component called “M-Sec Manager” exposing the underlying API.

```
GET : {{server}}/manager/get_nonce?h={{hostname}}
```

```
Resp :
{
    "nonce": "f6c132949b3e98ebf20b32517d1f9ed848bb3be1"
}
```

```
POST : {{server}}/manager/enroll_host
```

```
Headers :
  {
    "Authorization": "bearer {{access_token}}",
  }
```

```
Body :
   {
    "nonce": "f6c132949b3e98ebf20b32517d1f9ed848bb3be1",
    "hostname": "new_machine.msec.local"
   }
```

```
Resp :
200 : {
        "status": "success",
        "nonce": "f6c132949b3e98ebf20b32517d1f9ed848bb3be1",
        "password": otp_from_ipa(),
        "timestamp": str(time.time()
      }
```

```
500 : {
    "status": "failure",
    "reason": ["rejected" ,"nonce not valid", "server_error"],
     "timestamp": str(time.time()
}
```

```
POST : {{server}}/enroll_rpi
```

```
Headers :
  {
    "Authorization": "bearer {{access_token}}",
  }
```

```
Body :
  {
    "nonce": "f6c132949b3e98ebf20b32517d1f9ed848bb3be1",
    "hostname": "new_machine.msec.local"
    "file": base64. b64encode (quote.tgz)
  }
```

```
200 : {
        "status": "success",
        "nonce": "f6c132949b3e98ebf20b32517d1f9ed848bb3be1",
        "password": otp_from_ipa(),
        "timestamp": str(time.time()
      }
```

```
500 : {
        "status": "failure",
        "reason": ["rejected" ,"nonce not valid", "server_error"],
        "timestamp": str(time.time()
      }
```

```
POST : {{server}}/attest_rpi
```

```
Headers :
  {
    "Authorization": "bearer {{access_token}}",
  }
```

```
Body :
  {
    "nonce": "f6c132949b3e98ebf20b32517d1f9ed848bb3be1",
    "hostname": "new_machine.msec.local"
    "file": base64.b64encode(quote.tgz)
  }
```

```
200 : {
```

```
500 : {
```

"status": "success", }	"status": "failure", "reason": ["rejected" ,"nonce not valid", "attestation_failure"], "timestamp": str(time.time() }
POST : {{server}}/generate_cert	

Headers :
{
"Authorization": "bearer {{access_token}}",
}
	Body :
	{
	"nonce": "f6c132949b3e98ebf20b32517d1f9ed848bb3be1",
	"hostname": "new_machine.msec.local"
	"file": base64. b64encode (certificate_request.csr)
	}

200 : {
"status": "success",
"nonce": "f6c132949b3e98ebf20b32517d1f9ed848bb3be1",
"cert": base64.b64encode (certificate_request.csr),
"timestamp": str(time.time()
}
	500 : {
	"status": "failure",
	"reason": ["rejected" ,"nonce not valid", "server_error"],
	"timestamp": str(time.time())
	}

2.4 Interaction with other FG

The figure in the Annex presents the overall positioning of the end-to-end security FG inside the M-Sec Architecture. The following subsections present in more detail the interactions between this FG and the rest of the M-Sec system.

Interaction with IoT Marketplace FG

End-to-end security functional group provides accounting for the IoT Marketplace functional group. It enables clients of the marketplace to be authenticated either as the owner of devices, owner of data, or simply a consumer.

One interest in using the security manager instead of an internal database is to:

- prevent the inclusion of falsified data in the marketplace by verifying and attesting the authenticity of the source cryptographically.
- enable to trace the usage of the marketplace, in particular, to provide automatic breach notification and other forms of remediation.

We present in the diagram below a mutual authentication between an IoT marketplace app and a secure device. The secured device uses HTTPS flow to verify the IoT. The IoT marketplace app uses TLS Client Authentication to verify the identity of the client (secure device).

In the diagram in Figure 12, we present a use-case allowing a secure device to send monitoring data to an IoT marketplace app (here replaced by an InfluxDB databse for development and testing purposes) every x seconds. The proxy server (Nginx in the figure) verifies and checks the cert validity of every request using the M-Sec OCSP mechanism. If the cert is valid, data will be stored on the influxDB. Otherwise, data will be rejected, and users' owners are notified.

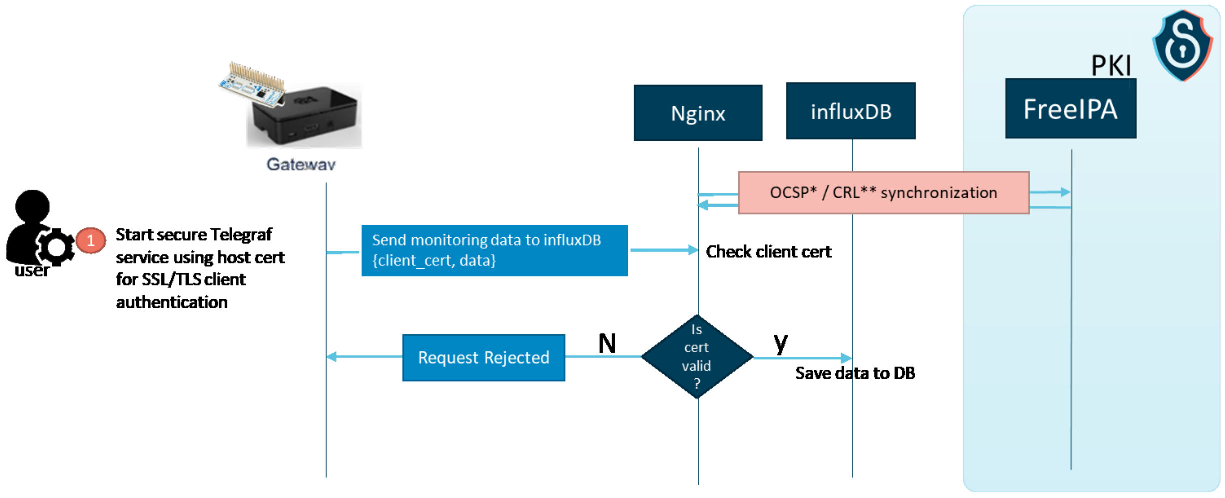

Gateway
Nginx
influxDB
PKI
FreeIPA
OCSP* / CRL** synchronization
user
1
Start secure Telegraf service using host cert for SSL/TLS client authentication
Send monitoring data to influxDB {client_cert, data}
Check client cert
Request Rejected
N
Is cert valid ?
y
Save data to DB

Figure 12. Example of certificate-based authentication and authorization between an IoT device and a backend

Interaction with the Devices/ Devices Security FG

Provisioning

End-to-end security functional group provides a PKI for devices with enables to provision this device with asymmetric cryptography. IT also provides an AAA capability that can enable the manufacturer or maintainer of the device to access this device with controlled credentials instead of generic ones. The attestation process is illustrated in Figure 13.

<u>TPM Remote Attestation Protocol:</u>

- Initiation
  - The client contacts the attestation server requests a nonce that is used to prevent reply attacks, and the list of PCRs to be signed.
  - The server sends the nonce
- Quote signing
  - Extracts the public part of the TPM Endorsement Key and the x509 certificate signed by the TPM manufacturer
  - Generates a signing-only Attestation Key (AK) inside the TPM and exports the public key (ak.pub)
  - Uses the TPM to sign a "quote" of the requested PCRs plus the nonce with the Attestation Key
  - Bundle up all into a tar file: ek.crt, ek.pub, ak.pub, quote.sig, quote.pcr, and the nonce
  - The Client then sends this quote file to the Server.
- Quote validation

  When the Server receives the quote file from the client, it runs:

  - Validates the SSL certificate chain on the client TPM EK cert to ensure that it came from a real TPM
  - Validates that the quote is signed by the AK with the correct nonce (if the nonce is not checked, then this could be a replay attack by the Client)
- PCRs Verification
  - The server optionally validates that the PCRs match the expected values

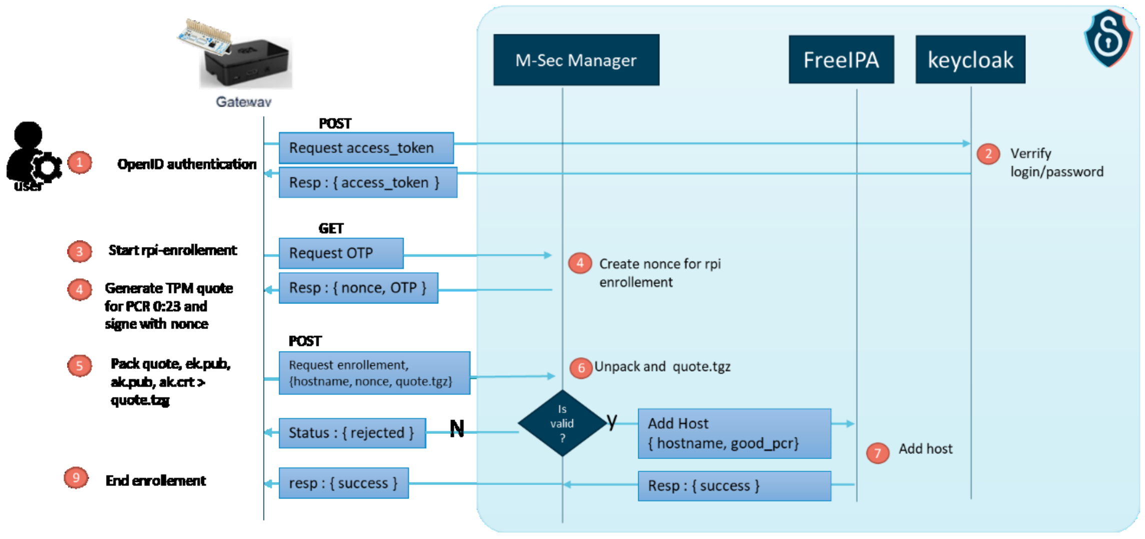
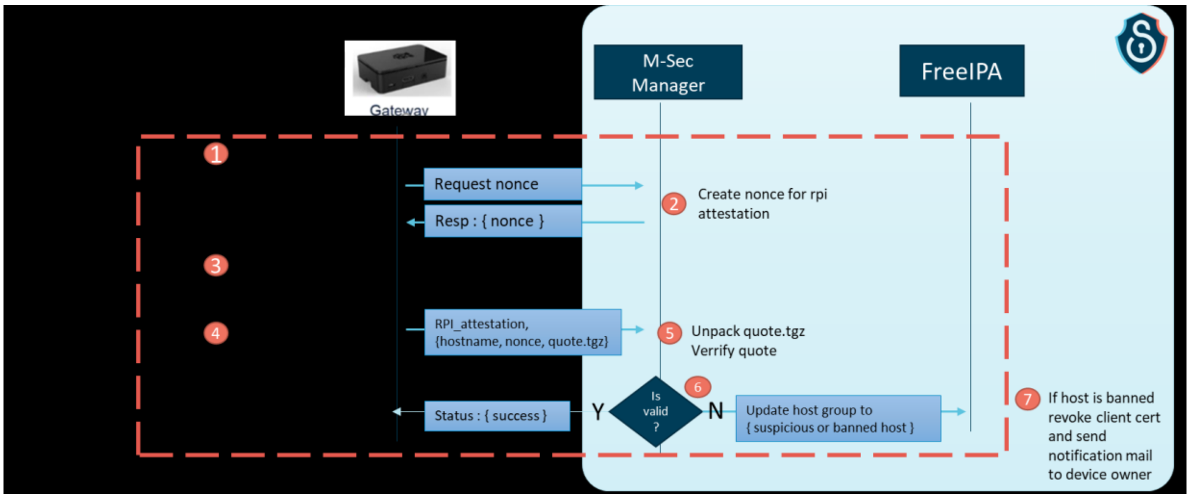


Figure 13. Host enrollment using OTP and TPM attestation

Integrity management

The functional group also provides integrity management for the secure device FG using the TCP attestation model as described in Figure 14.



Gateway
M-Sec Manager
FreeIPA
1
Request nonce
Resp : { nonce }
2 Create nonce for rpi attestation
3
4
RPI_attestation, {hostname, nonce, quote.tgz}
5 Unpack quote.tgz Verrify quote
6
Is valid ?
Y
N
Status : { success }
Update host group to { suspicious or banned host }
7 If host is banned revoke client cert and send notification mail to device owner

Figure 14. Attestation of integrity of the device based on the TPM

Interaction with the Secure City Data Access FG

The security manager proves authentication and authorization for secured city-data access components. Typically, it authenticates flows given their origin and destination. This enables allowing or denying access to some resources. For example, when a group of the device shall have restricted access, such as for city maintenance only, the secured data city access can rely on authentication provided by the directory service of the security manager.

An integration of the security manager with the sensiNact component in the Secured City Data Access FG was described in the previous version of this deliverable and summarized with Figure 15. The northbound bridges uses the identity module while the southbound bridges can use encryption based on the security manager PKI, if the source protocol allows it.

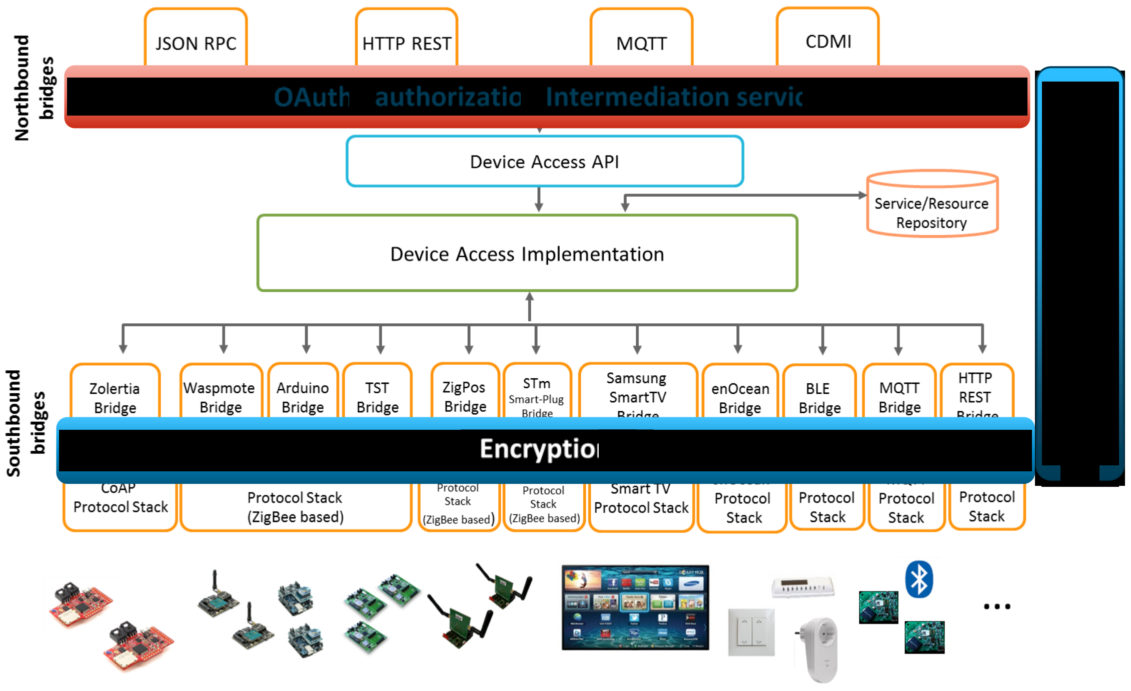


Figure 15. Security features provided by the Security Manager and integrated within sensiNact

Interaction with the Applications FG

Several kinds of interaction may happen with the Application functional group:

Citizen accounting

The first approach is to authenticate citizens using the identity federation module. By doing so, the citizen can register with their existing identities (OpenID and/or OAuth2) and use Single-Sign-On over the whole M-Sec framework. This feature may be useful in particular for the different frontends used by citizens such as smartphone apps or web-based portals.

The use of an authenticated approach is mandatory to preserve the user rights such as privacy preferences (consents are given, etc.).

Cyber-resiliency management

A second approach involving the Application FG regards cyber-resilience, in particular in response and recovery towards cyber-events. In this scope, the security manager can receive events from different sources:

- From the Devices Security FG: events regarding possible integrity failure (with the **Secured component for devices** asset), or events regarding suspicious activity (asset: **Intrusion Detection System**)
- From the secured & trusted storage FG: events regarding possible abnormal activity and distrust of a user using the **T&R model engine tool**.

- From any other external application having security monitoring capabilities (typically using (IDMEF / RFC 4765)

From these events, action can be taken gradually to adapt the level of protection and defense of the while M-Sec architecture. These scenarios can be designed and reinforced using the Security Analysis Tool, Development Method for Secure Services and Modal System Transition Analyzer (for Security) - MTSA.

3. Conclusion

In this document, we have presented a security management tool to implement “End-to-End Security” Function Group for M-Sec. This security management tool provides all-in-one security functions for large-scale IoT infrastructures such as a Public Key Infrastructure for certificates, a directory module for accounting, and an identity federation module for user management. This tool has been built around various standards to facilitate its usability, especially for people who do not have strong cybersecurity expertise. The benefit of this tool is to provide interoperable security management across multi-organizational environments.

Some examples of integration with other functional groups are provided in this document that enables to manage, in an automated way, incidents related to cybersecurity. One typical case is the automatic breach notification required by the privacy regulations such as PIPA and GDPR. These examples show how incidents from devices are detected and how mitigation can be automated leveraging the security and functional features from each other layers.

Annex

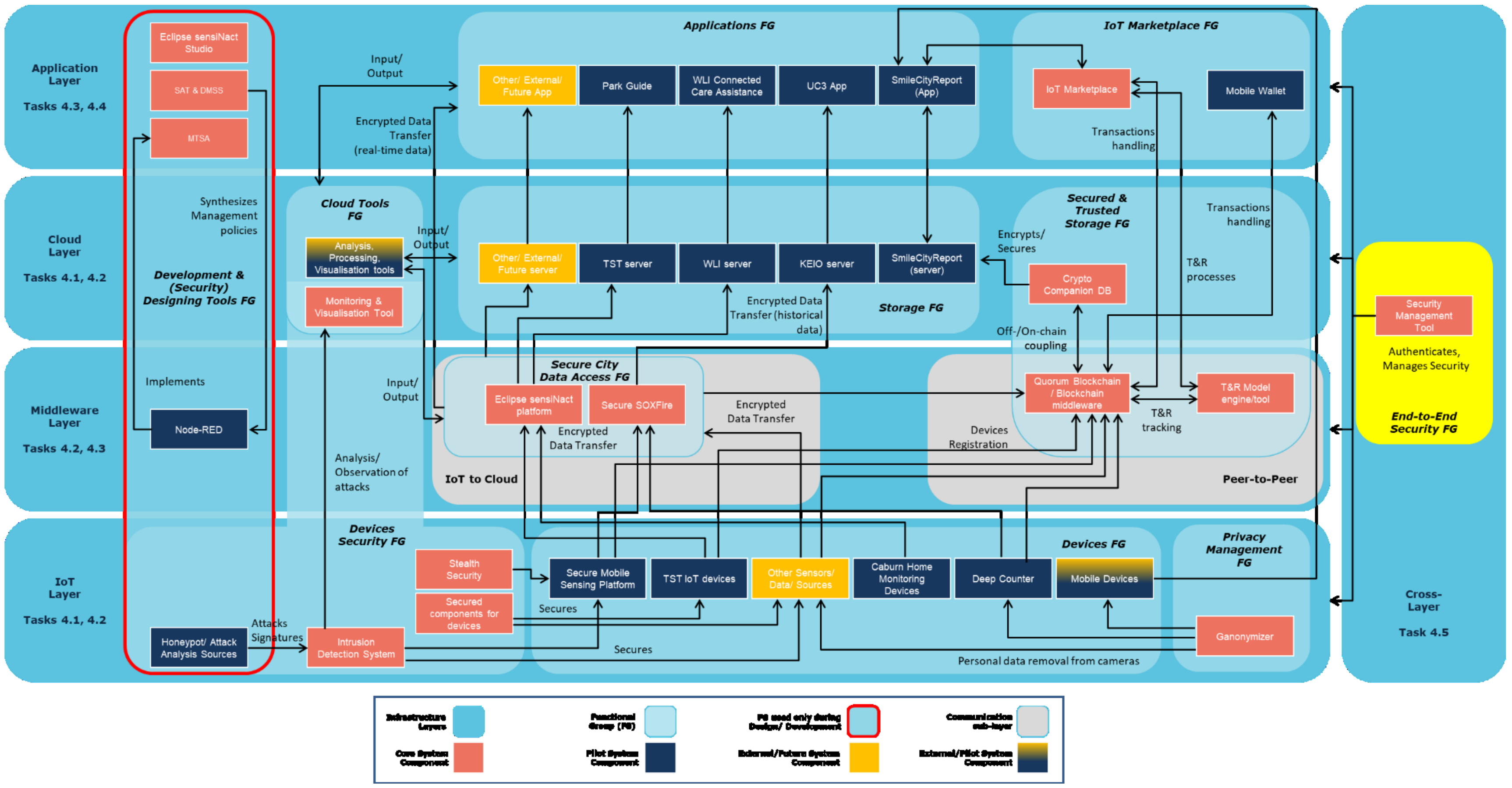


Figure 16. The M-Sec Architecture (T4.5 FG in yellow)