



i-locate - Indoor/outdoor LOcAtion and Asset management Through open gEodata (GA 621040)

## COVER PAGE



# DELIVERABLE

**Project Acronym: i-locate**

**Grant Agreement number: 621040**

**Project Title: Indoor/outdoor LOcAtion and Asset management Through open gEodata**

## D.3.5 - Security and privacy layer (Accompanying report)

**Revision: 1.0**

**Authors:**

**Scott CADZOW (C3L)**

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

<b>File:</b> D3.5 - Security and privacy layer.docx	<b>D.3.5</b>
<b>Page:</b> 1 of 26	<b>Security and privacy layer</b>

## REVISION HISTORY AND STATEMENT OF ORIGINALITY

### Revision History

Revision	Date	Author	Organisatio	Description
V0.1	June 2015	Scott Cadzow	C3L	First draft
V0.2	21/06/2015	Giuseppe Conti	Trilogis	Complete review
V0.3	28/06/2015	Scott Cadzow	C3L	Inclusion of contribution from system integrator
V1.0	30/06/2015	Giuseppe Conti	Trilogis	Complete review

### Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

## 1 List of references

Number	Full reference
1	OpenGIS Geospatial eXtensible Access Control Markup Language (GeoXACML) Draft Implementation Specification (OGC Document 07-026)

## 2 Table of Acronyms

Acronym	Description
ABAC	Attribute Based Access Control
ADE	Application Domain Extension
AES	Advanced Encryption Standard
API	Application Programming Interface
CBAC	Consent Based Access Control
CIA	Confidentiality, Integrity, Availability
CityGML	CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models
DBMS	DataBase Management System
ENISA	European Network and Information Security Agency
GeoXACML	Geospatial eXensible Access Control Markup Language
GML	Geography Markup Language
IdM	Identity Management
IdP	Identity Provider
IndoorGML	OGC standard for an open data model and XML schema for indoor spatial information
IP	Internet Protocol
IPSec	IP Security
MD5	Message Digest 5 (algorithm for generating a cryptographic digest of text)
NIST	National Institute of Standards and Technology
OGC	Open Geospatial Consortium
OASIS	Advancing Open Standards for the Information Society ( <a href="http://www.oasis-open.org">www.oasis-open.org</a> )
OpenAM	Open Access Manager
OpenIAM	OpenIAM is a comprehensive Identity and Access Management solution
PIA	Privacy Impact Assessment

PKC	Public Key Certificate
PKI	Public Key Infrastructure
PMI	Privilege Management Infrastructure
RBAC	Role Based Access Control
RP	Relying Party
RSA	Rivest-Shamir-Adleman
SAML	Security Assertion Markup Language
SHA	Secure Hash Algorithm
SpoA	Service point of Attachment
SSO	Single Sign On
TBAC	Trust Based Access Control
TLS	Transport Layer Security
TVRA	Threat Vulnerability Risk Analysis
UML	Unified Modeling Language
URN	Uniform Resource Name
XACML	eXtensible Access Control Markup Language
WSO2	Product name from www.WSO2.com

### 3 Executive Abstract

The present document describes the functions of the security layer and privacy protection layer (as an integral element of the security layer), an abstraction of the i-locate architecture, implementing the Confidentiality, Integrity, Availability (CIA) functions identified in the Privacy Impact Assessment/Threat Vulnerability Risk Analysis (PIA/TVRA) deliverable of the project.

The functionality is presented in two forms:

- Simple user name password with non-cryptographic isolation of functions
- Federated identity management with cryptographic isolation of functions

The rationale for two presentations of the security layer is that cryptographic operations are complex and merely strengthen a system, of themselves they do not mitigate weaknesses but make such weaknesses more difficult to exploit. The primary security measure is to eliminate system weaknesses themselves, or failing that to isolate the weak systems in such a way that breaking that weakness does not lead to a cascade of security failures across the system.

The core model of protection in i-locate is attribute based access control where those attributes represent variously: Role; Organisation; Geographic location. As policy is developed further attributes may be added to the list. The primary aspects of i-locate are the geo-location of objects, and the routing to them. In any geographic space or volume some areas are out of bounds thus routing has to take account of such areas. Some areas may be made visible to some roles and not to others, for example in a museum staff may be allowed to make shortcuts through storage areas, whereas visitors cannot. Whilst conventional geo-fencing addresses many of the simpler access control requirements, the generic model does not fit to i-locate but is to be extended for asset control in which assets need to be located across a geo-fenced environment, their movements across geo-fences noted (an anti-theft measure), but not all assets should be visible in the virtualised environment to all users. The principle of data minimisation is attached to asset control in such a way that only if an asset is allowed to be visible to a class of user, or a specific user, it will be made visible in the i-locate mapping applications.

For the pilot in its first release only the most basic security functionality is offered using the Single Sign On (SSO) model centralised through the OpenAM core element. The intent behind this very conservative approach is to give assurance that the core functionality of i-locate is operational and once that has been assured to add the restrictive permissions required to give full protection and restriction of assets in i-locate for the post- and late- pilot phases.

## 4 Table of contents

<b>1 LIST OF REFERENCES.....</b>	<b>3</b>
<b>2 TABLE OF ACRONYMS .....</b>	<b>4</b>
<b>3 EXECUTIVE ABSTRACT .....</b>	<b>6</b>
<b>4 TABLE OF CONTENTS .....</b>	<b>7</b>
<b>5 TABLE OF FIGURES .....</b>	<b>8</b>
<b>6 OVERVIEW OF SERVICES PROVISIONED IN I-LOCATE PILOT TO JULY 2015 .....</b>	<b>9</b>
<b>7 SERVICES TO BE ADDED TO PILOT SYSTEM.....</b>	<b>11</b>
<b>8 APPENDIX A: IDENTIFICATION AND AUTHENTICATION SERVICE.....</b>	<b>12</b>
<b>9 APPENDIX B: ACCESS CONTROL SERVICE .....</b>	<b>15</b>
9.1 Overview.....	15
9.2 Policy and rule creation .....	15
9.3 Role definition.....	15
<b>10 APPENDIX C – PASSWORD POLICY.....</b>	<b>17</b>
10.1 Password strength checkers.....	19
10.2 Local password policy using OpenAM.....	19
<b>11 APPENDIX D – OVERVIEW OF XACML .....</b>	<b>20</b>
<b>11.1 Writing XACML Policies .....</b>	<b>20</b>
11.1.1 The Policy Vocabulary .....	20
11.1.2 Policy Basics: Identifier, Description, and Rule Combining Algorithm .....	20
11.1.3 Defining the Policy Target.....	21
11.1.4 Defining Policy Rules.....	24
11.1.5 XACML Functions.....	25
11.1.6 Required vs. Optional Attributes in a Policy.....	26
11.1.7 Declaring mandatory attributes.....	26
11.1.8 Declaring optional attributes .....	26



## **5 Table of Figures**

Figure 1: Screenshot of mobile app user login and registration screen	10
Figure 2: The three primary roles in the common IdM thematic model	12
Figure 3: Role creation for i-locate using the OpenAM web-console	16
Figure 4: OpenAM password policy toolkit	19



## 6 Overview of services provisioned in i-locate pilot to July 2015

The following functionalities are currently supported in the i-locate system:

- **Mobile app user self-registration and login.** A mobile app user has to be registered [D3.6] which is performed by sending a registration request to the OpenAM REST APIs, which triggers the user self-registration functionality. When registering, the user is asked for a valid email. A confirmation email is sent by OpenAM to the specified email address, including an activation link (i.e. the authentication identity has to be confirmed). Once the activation link is clicked by the user a registration form is presented enabling the selection of authentication credentials (i.e. the password that complies to the i-locate password policy as outlined in the Annex). Once the form is completed and submitted, the account is activated in OpenAM. By default users are assigned to the default user group (role = ilocateDefaultUser) but not assigned to any higher privilege group. Groups can be used to handle authorization for access control purposes.

*NOTE: OpenAM and thus i-locate allows users to regain access if a password is lost/forgotten, in which case an email is sent to the given email address containing an authentication link for resetting the password.*

- **Mobile app user login.** Users log in using username and password. Login requests are sent to the OpenAM endpoint, which in case of a positive match returns an SSO token to be used by the client for further interactions with all other i-locate components.
- **Asset management user registration and login.** Users of the Box3 asset management service are required to login directly in the Box3 interface. The whole Box3 application authenticates with OpenAM using login/password. This decoupling allows to keep the fine-grained access control already part of the Trilogis Box3 solution without adding excessive burden to the i-locate OpenAM instance. Further, it allows legacy users of Box3 to keep logging in using their login and password, while location updates of relevant assets are handled through the i-locate system.
- **Component authentication.** i-locate toolkit components authenticate with OpenAM by means of a login and password. Registration is manually handled by system administrators at system deployment time. The SSO token received by the component is used in all subsequent transactions.
- **Access to resources.** Once an authorized component wants to access a given resource (in the i-locate case, typically the position of an entity, be it an asset or a person) it issues the request to a verifier (in the current implementation the proxy and MQTT broker are registered as verifiers, inline with the i-locate architecture [D1.4, D3.1]), together with its SSO token. The verifier first queries OpenAM to check the validity of the token received. Upon a successful response, it then forwards the request to OpenAM. (This step includes the translation of the resource description from the one in the request, which will be typically related to a given i-locate entity ID, to a resource descriptor that matches an authorization policy resource defined in OpenAM's policies store. This step is typically accomplished by the proxy, potentially interacting with the configuration module for looking up resource descriptors.) The OpenAM engine checks whether the token provided is valid

File: D3.5 - Security and privacy layer.docx	D.3.5
Page: 9 of 26	Security and privacy layer

for accessing (with the operation type included in the request) the requested resources. If the response is positive, the verifier can then provide access to the intended resource.

- **MQTT publish and subscribe operations.** The i-locate toolkit uses MQTT for handling the dispatching of location data among components [D3.1, D3.2]. MQTT is a lightweight publish/subscribe system specifically designed for IoT scenarios. In order to comply with security requirements, a plugin was developed to support authorization to read and write operations. More specifically, when a component (e.g., the geofencing component) wants to publish on the MQTT broker, it sends a request to the broker, describing the operation it wants to perform (in this case: write) along with its username and password. The broker, which is authenticated as verifier, processes this request and exploits OpenAM to perform authentication (obtaining an SSO token associated to the component) and authorization. The OpenAM authorization engine checks whether the requesting entity has the right to perform the requested operation. In case of a successful response, the data is published on the broker under the requested topic. The response is cached so that the authentication with OpenAM is done only once. The process is similar for the subscription. In this case when a subscription request is issued by a component, together with its SSO token, the broker processes the request and issues a request to OpenAM to check whether the required entity is allowed, according to current access control policies, to access the required stream of data. In case of a positive response, the broker forwards all data published on the given channel to the requesting entity. In case of a negative response, the requesting entity receives no data published on that channel.

A sample integration script can be retrieved from the online repository: <https://gitlab.com/ilocate/ilocate-middlewre-identity-management>

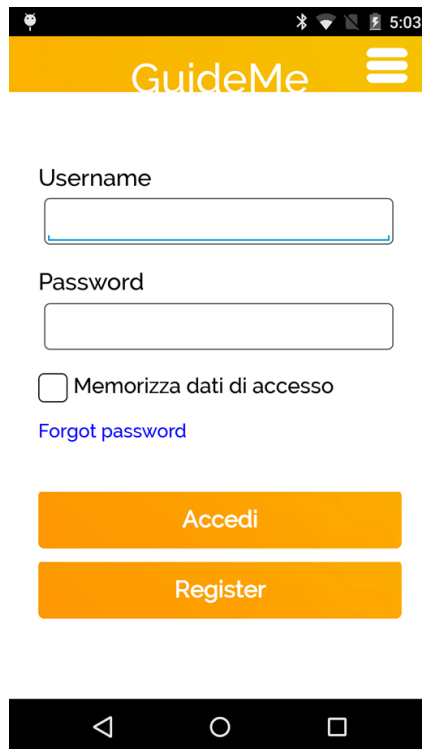


Figure 1: Screenshot of mobile app user login and registration screen

<b>File:</b> D3.5 - Security and privacy layer.docx	<b>D.3.5</b>
<b>Page:</b> 10 of 26	<b>Security and privacy layer</b>

## **7 Services to be added to pilot system**

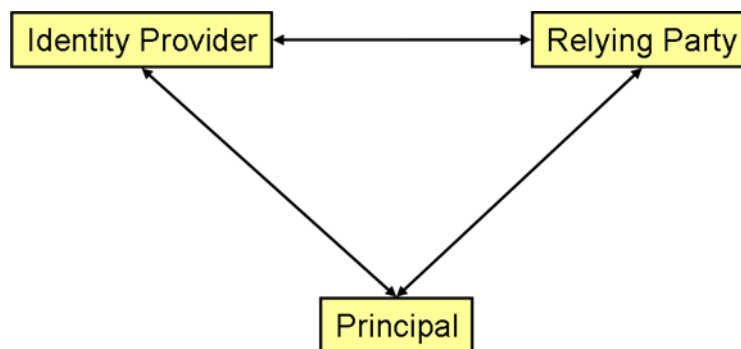
The core services all exist in the initial pilot system. As the system is more fully deployed more and more advanced policies will be added to the OpenAM core thus changing the visibility of assets in the system by role, location and consent.

<b>File:</b> D3.5 - Security and privacy layer.docx	<b>D.3.5</b>
<b>Page:</b> 11 of 26	<b>Security and privacy layer</b>

## 8 Appendix A: Identification and authentication service

The three key Identity Management (IdM) roles to be considered in i-locate that are identified from an analysis of the on-going work and output of a number of bodies active in the field are:

- Principal:
  - Often synonymous with the end-user or an electronic agent of the end-user and represented for i-locate by the client.
- Identity Provider (IdP):
  - The entity or organization generally required to authenticate the Principal and to provide an assertion of this authentication to the Relying Party, the role in i-locate taken by the OpenAM Identity Manager.
- Relying Party (RP):
  - An organization or entity providing a service to the Principal. The Principal may authenticate to the RP but, the RP is also willing to rely on an assertion provided by the IdP as in the i-locate middleware model. The RP in i-locate are all the key i-locate services.



*Figure 2: The three primary roles in the common IdM thematic model*

While all IdM approaches analysed define these roles (sometimes the terminology used is different) they differ in the communication protocols used between these roles. In the i-locate model IdM acts as both a centralised entity for Single Sign On (SSO) but also as the authority for the eXtensible Access Control Markup Language (XACML) like Attribute Based Access Control (ABAC) model. The term Service point of Attachment (SpoA) is used to indicate and to generalise the point at which the service is connected to the user and may represent an Application Programming Interface (API) or equivalent interface.

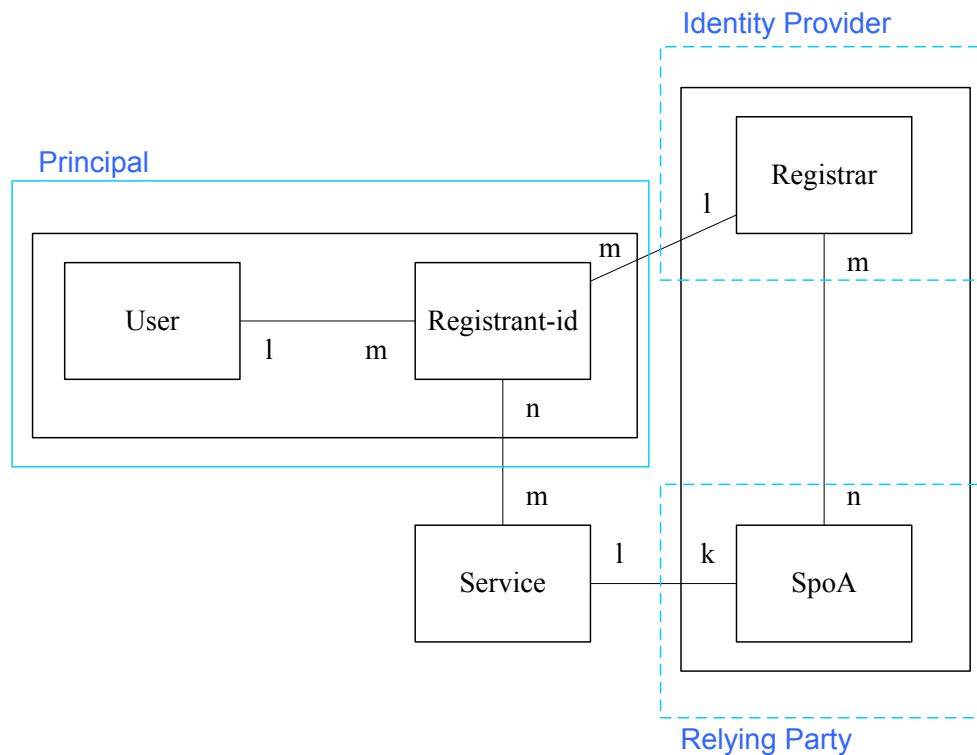


Figure 3: The basic authorization model

- NOTE 1: A single user may be associated with many registrant-ids.
- NOTE 2: A registrant-id shall be associated with only one user.
- NOTE 3: A registrant-id shall be associated with only one registrar.
- NOTE 4: A registrar may be associated with many registrant-ids.
- NOTE 5: A service may be associated with many SpoAs.
- NOTE 6: In any registration instance a service shall be associated with only one SpoA.
- NOTE 7: An SpoA shall be associated with only one Service.
- NOTE 8: A registrant-id may be associated with many Services.

Authentication is used as a reinforcement of identity prior to authorization. Authentication procedures are variously symmetric and asymmetric keyed schemes with both challenge-response protocols and digest based calculation or signature methods.

An identifier has a specific meaning within a specific context and identity problems occur most frequently when either contexts are not unique or the scope of an identifier is extended beyond its original context (for example, by using a telephone number for something other than for making telephone calls). As elements of identity, identifiers are often used to support a claim of identity but such out-of-context uses of identifiers can generally be easily guessed in an identity fraud attempt. Where an identity can be restricted to a single purpose (with or without supporting authentication data) it may be more difficult to use it as the basis of identity fraud.



**i-locate - Indoor/outdoor LOCation and Asset management Through open gEodata (GA 621040)**

*NOTE: In many cases it may not be practical to limit an identity to a single purpose as it would require retrospective modification of common practice. However, the principle should still be considered.*

<b>File:</b> D3.5 - Security and privacy layer.docx	<b>D.3.5</b>
<b>Page:</b> 14 of 26	<b>Security and privacy layer</b>

## 9 Appendix B: Access control service

### 9.1 Overview

Access control is the model most often used to implement the Authorisation element of the CIA paradigm. Any object (e.g. file, executable code, API function) has a number of visibility options that include read, write, execute. Access control very simply restricts who can perform each of these functions. In the i-locate environment objects may be made invisible (no access), may be visible only (read access), or may be executable. The means to enable access control in i-locate is taken from the XACML model of Policy Enforcement in which every system request is captured and only permitted if the requestor meets the policy applying to the entity, else the request is denied. The specific rules are given in section 4 above, whilst the remainder of this section identifies the general model for creating rules (and further expanded in “Appendix D – overview of XACML”).

### 9.2 Policy and rule creation

The i-locate policy dictionary has been derived from the overall Unified Modeling Language (UML) model and source code of the i-locate implementation. In like manner to the Geospatial eXensible Access Control Markup Language (GeoXACML) document [1] the primary dictionary of types in i-locate have been the Geography Markup Language (GML), IndoorGML, CityGML and CityGML-ADE (Application Domain Extension) types.

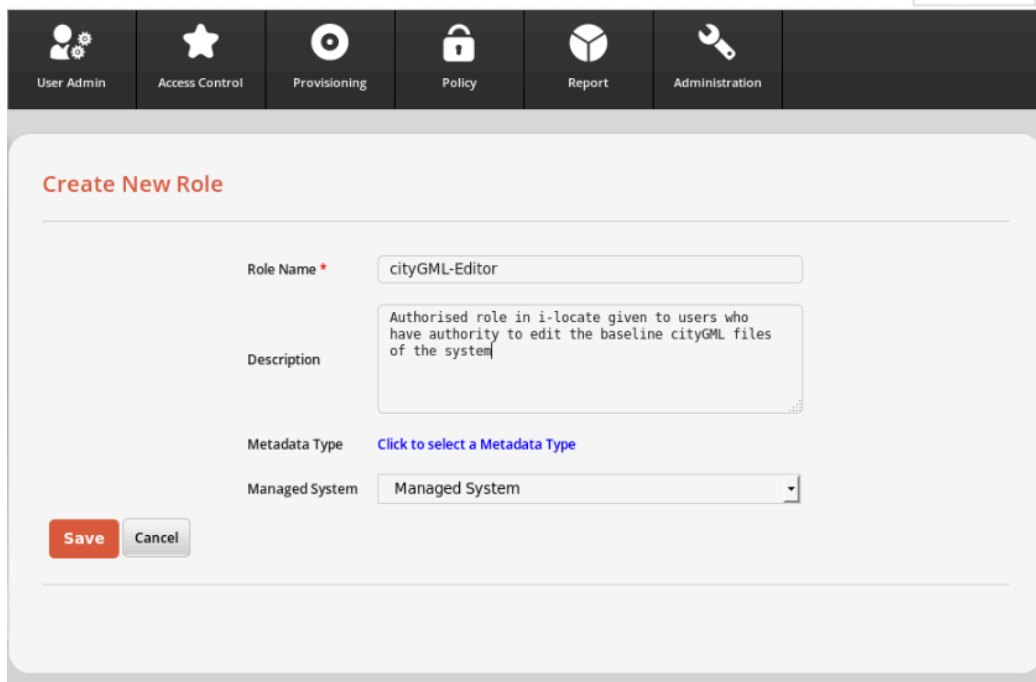
Note that GeoXACML extends XACML by only one new data type that is named “urn:ogc:def:dataType:geoxacml:1.0:geometry”.

The result of this is that the data type of every geometric attribute value, bag of geometric values or pointers to geometric data (i.e. AttributeSelector or AttributeDesignator) in a GeoXACML policy SHALL always be “urn:ogc:def:dataType:geoxacml:1.0:geometry”.

### 9.3 Role definition

Within ABAC the most common model is RBAC (Role Based Access Control) and for this we identify a number of specialist roles in i-locate. The first of these are a combination of restricted roles of CityGML and IndoorGML editor, the rationale for which is that the underlying CityGML and IndoorGML resources are key to the definition of the entire i-locate functionality. The screen shot in Figure 3 shows the use of the OpenAM Web-console to define roles for the i-locate system.

<b>File:</b> D3.5 - Security and privacy layer.docx	<b>D.3.5</b>
<b>Page:</b> 15 of 26	<b>Security and privacy layer</b>



**Create New Role**

Role Name \*

Description

Metadata Type [Click to select a Metadata Type](#)

Managed System

Figure 3: Role creation for i-locate using the OpenAM web-console

The following table identifies the roles defined for i-locate.

Role name	Purpose
AssetManagementConfiguration	Restricted to users who have authority to configure the asset management entity's backend
AssetManagementUser	Restricted to users who have authority to maintain the execution model of the asset management entity
internalGMLeditor	Authorised role in i-locate given to users who have authority to edit the baseline IndoorGML files of the system
CityGMLeditor	Authorised role in i-locate given to users who have authority to edit the baseline CityGML files of the system
iLocateDBManager	Update and modification access to all system databases
iLocateDefaultUser	Read access to all active system assets
iLocateSystemAdministrator	Read/Write/Execute to all active system assets



## 10 Appendix C – Password Policy

**NOTE:** *This text has been given in a number of forms by the author over a number of years and in a number of documents and public presentations and is quite generalised but maintained on every re-use to take account of changes in global security knowledge.*

Is there a safe password? This seems to be the most frequent question put to those who claim to deal with security and it is useful to give some rational explanation of what constitutes a good password and password policy. Let's look at the password problem and from there we can identify good practice to follow in defining passwords.

Very simply password security, measured by the time an attacker will need to guess it, is proportional to the length of the password and the size of the alphabet used to create it. An alphabet of only digits (0,1,2,3,4,5,6,7,8,9) to create an 8-digit PIN would only give  $10^8$  possible combinations, using only lower case letters an 8-character password would give  $26^8$  possible combinations, and obviously using a mixed combination of upper and lower case letters and characters would give a dictionary of 62 characters and thus  $62^8$  combinations, then adding in either more allowed characters or a longer minimum length extends the size even further. The guideline for cryptographic strength based on National Institute of Standards and Technology (NIST) and European Network and Information Security Agency (ENISA) is 128 bits which is still very much longer even than (say) a 12 character password made from an alphabet of 100 or so characters and symbols (or less than 20 bits equivalence in cryptographic security). So we should never forget that passwords are at the weak end of the security spectrum and with that in mind we need to look at ways to make them stronger, and thus less likely to be "guessed" by some malicious foe.

The length of a secret key in cryptography is partly influenced by the state of the art against brute force attack, i.e. how long it will take on average to find the key if every key in the key space is tried, and to an increasing extent by fashion and media panic. The existing 128-bit recommendation from amongst others ENISA and NIST is based on the assumption that data protected today will still be "safe" from brute force attack in 30 years<sup>1</sup>. Translation to asymmetric cryptography gives equivalent strength estimates (for Rivest-Shamir-Adleman - RSA based asymmetric cryptography 128-bit strength requires keys to be 4096 bits or longer)<sup>2</sup>. The time to remain secure is a key element and whilst conventionally cryptography works on the basis that the key should never be found<sup>3</sup> in the world of passwords the recommendation is to change them more frequently than the name space (key space) can be checked.

Choice of password is often poor and given that it is estimated that there are 220,000 dictionary base words for passwords it would not take an attacker long to work through all of them, and not much longer if all of these base words were "strengthened" using substitution of (say) "a" with "@"

<sup>1</sup> The arrival of viable quantum computers in the foreseeable future will cut the cryptographic strength of symmetric ciphers by 2, thus algorithms offering 128-bit strength today will be reduced to 64-bit strength under a quantum computing era.

<sup>2</sup> There are other issues at play not least that asymmetric cryptography relies on problems that are considered "hard", e.g. prime factor factorisation at the root of RSA. If such problems are solved then any security based on them is bypassed.

<sup>3</sup> If you could test 1 billion keys a minute then you'd need  $2^{98}$  minutes to explore the entire key space, and this works at roughly  $2^{78}$  years or 302231454903657293676544 years in conventional notation which given that the earth is only some 4.54 billion years old, and the universe some 13.8 billion years old, this level of cryptographic strength should outlive most of us.

<b>File:</b> D3.5 - Security and privacy layer.docx	<b>D.3.5</b>
<b>Page:</b> 17 of 26	<b>Security and privacy layer</b>

or “s” with “5”. Attackers will develop and exchange password dictionaries containing all of these common combinations, alongside their hashes using the common hashing algorithms (Message Digest 5 - MD5, Secure Hash Algorithm - SHA, etc.). In practice password dictionaries, password attack networks, the use of botnets to capture transferred hashes, make immunity from password attacks difficult over a long period and passwords should be routinely changed to minimise exposure. Even using protocols that send the hash of the password such that the password is not easily visible in the clear does not guarantee safety. What the well prepared attacker will do is look up the hash in his dictionary of password hashes and if a match is found he will have the password. This does not require any breaking of the hash function, or direct “guessing” of the password.

The guideline in security thinking of “one key – one purpose” should be followed in selecting passwords and recommended wherever passwords are needed. This does inevitably raise the horror of users that if they need to access (say) 50 sites they’d have to remember 50 very cryptic passwords, in which case users should be pointed towards use of a password “safe” tool that creates strong passwords and ensures a different password in use in every site. The checklist below will improve password security (in the sense that passwords will be more difficult to guess and less likely to appear on password dictionaries) if all the checks are answered “yes”.

The alternative, which has been pursued in the i-locate project, is to federate identity and credentials. This scheme, often referred to as Single Sign On, uses a single trusted entity to distribute credentials to other trusted entities on the network. Thus a user logs on (signs in) to the system once at the identity manager and that identity manager then manages the credentials of all other systems in the network that would require the user to sign in. The common models for this are Kerberos and X.509 based PMI (Privilege Management Infrastructure). The model of choice in i-locate is made through the choice of a packaged identity manager product.

Action	Yes	No
Password for all user accounts shall be a minimum of 8 characters long, and contain each of the following:		
One or more lower case letter		
One or more upper case letter		
One or more numbers		
One or more symbols		
Passwords that take a “dictionary” word (e.g. authenticate) and replace letters with numbers that refer to the position of the letter in the word, and which replace letters with their closest symbolic representation should be filtered out (e.g. Auth5ntic@te) as they are susceptible to dictionary attacks.		

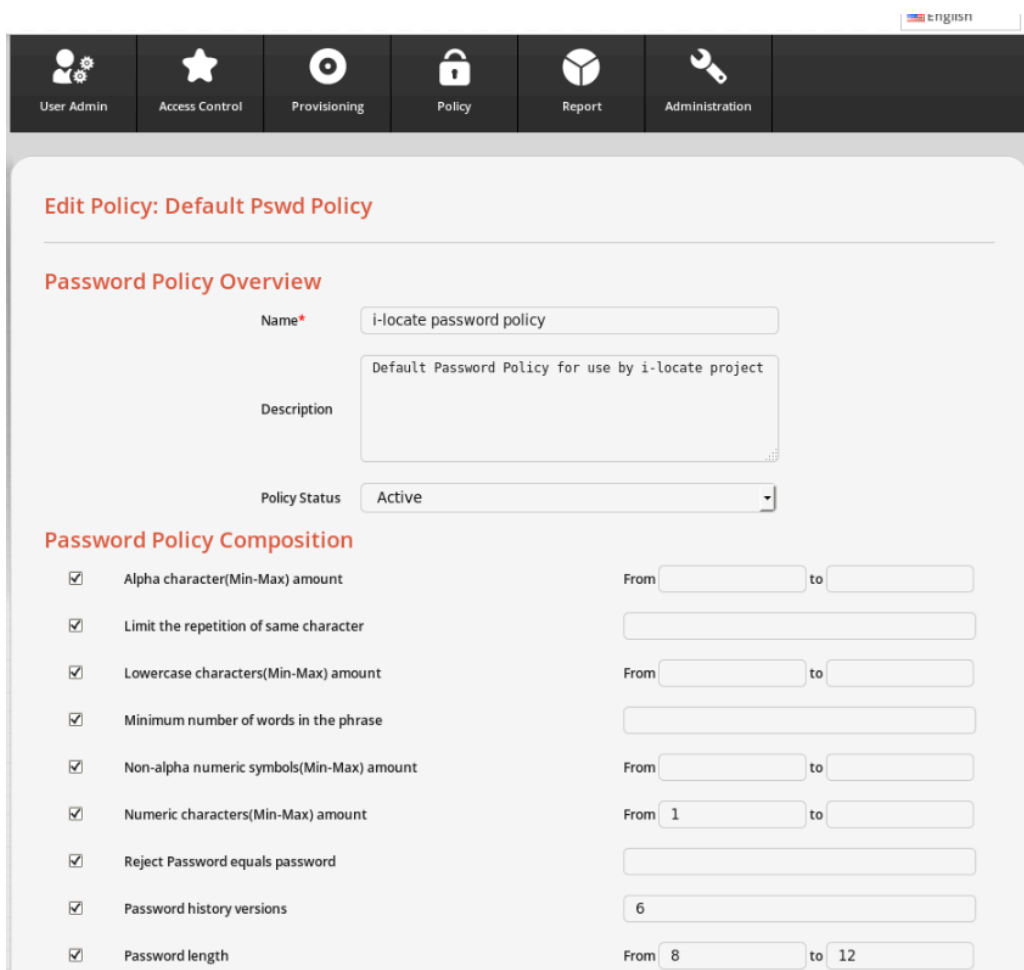
## 10.1 Password strength checkers

There are a number of ways of indicating to a user that their selected password is strong or weak. A password policy such as that given in the table should be visible at the point when a user selects a password and should be indicative only. Ultimately the strength, familiarity and other factors associated to a password, should be left to the user as the system itself should not hold the password in its user entered form but should be stored in a secure manner (there are a number of tools to enable this that are embedded in modern software and which have been identified in a number of guidance documents including those from ENISA and NIST).

The risk of password exploit should be made clear to users. In the i-locate environment the level of risk of exposure of private information is designed to be low as very little such information is gathered.

## 10.2 Local password policy using OpenAM

Using the installed tools for i-locate the password policy can be fixed with multiple variables. The screenshot in Figure 4 shows the building up of the i-locate default password policy using the OpenAM administration tools.



The screenshot displays the OpenAM administration interface for editing a password policy. The top navigation bar includes icons for User Admin, Access Control, Provisioning, Policy, Report, and Administration. The main content area is titled 'Edit Policy: Default Pswd Policy' and is divided into two sections: 'Password Policy Overview' and 'Password Policy Composition'.

**Password Policy Overview:**

- Name\***: i-locate password policy
- Description**: Default Password Policy for use by i-locate project
- Policy Status**: Active

**Password Policy Composition:**

- Alpha character(Min-Max) amount: From [ ] to [ ]
- Limit the repetition of same character: [ ]
- Lowercase characters(Min-Max) amount: From [ ] to [ ]
- Minimum number of words in the phrase: [ ]
- Non-alpha numeric symbols(Min-Max) amount: From [ ] to [ ]
- Numeric characters(Min-Max) amount: From 1 to [ ]
- Reject Password equals password: [ ]
- Password history versions: 6
- Password length: From 8 to 12

Figure 4: OpenAM password policy toolkit

## 11 Appendix D – overview of XACML

**NOTE:** *The bulk of this material is masked when using the tools to create rules.*

The eXtensible Access Control Markup Language (XACML) is a policy centric target language for the implementation of privacy and security controls in i-locate. The controls in i-locate are all part of a privilege management infrastructure using models of Role Based Access Control (RBAC), Trust Based Access Control (TBAC) and Consent Based Access Control (CBAC). RBAC in i-locate identifies specific roles and assigns users to them. Each role has restricted read, write and update access to data elements. The TBAC restrictions are designed to ensure only those with a provable relationship to an entity can comment on that entity. The CBAC controls ensure that where PII is involved that consent has been given and is traceable to a specific event (generally the establishment of a trust contract).

There is increasing work being done in the R&D community on Attribute Based Access Control (ABAC) wherein access rights are granted to users through the use of policies which combine attributes together. The policies can use any type of attributes (user attributes, resource attribute, etc...). Attributes can be compared to static values or to one another thus enabling relation-based access control. The application of ABAC in i-locate is under study but will use the same base language of XACML as for RBAC, TBAC and CBAC.

There are a number of approaches to defining policy but essentially there are policies that contain one rule, such an approach to policy writing results in many individual policies with the advantage that each policy is atomic and uncomplicated, or a policy with many rules. The multi-rule approach will often result in more complicated policies but is generally considered appropriate where a single policy states all the rules for a particular digital object.

Irrespective of the style of writing policies there are algorithms that define how different policies are combined and where precedence of one rule over another occurs.

### 11.1 Writing XACML Policies

#### 11.1.1 The Policy Vocabulary

The use of a specific vocabulary is useful in ensuring that key terms or words are externally defined and themselves extensible. Ideally the vocabulary will define a set of Uniform Resource Names (URNs) that can be used to identify specific operations, object attributes, and the wider environment within an XACML policy. These URNs are used as attribute designators in XACML policies, specifically within a SubjectAttributeDesignator, ResourceAttributeDesignator, ActionAttributeDesignator, or EnvironmentAttributeDesignator.

#### 11.1.2 Policy Basics: Identifier, Description, and Rule Combining Algorithm

Every policy has an identifier, a rule combining algorithm, and a description. In the root element of an XACML policy there is an attribute to provide the policy with a unique identifier. Also, the <Description> element provides a place to put a textual description of the purpose of the policy.

```
<Policy PolicyId="..." RuleCombiningAlgId="..." ...>  
  <Description> free text </Description>  
  <Target>
```

<b>File:</b> D3.5 - Security and privacy layer.docx	<b>D.3.5</b>
<b>Page:</b> 20 of 26	<b>Security and privacy layer</b>

```

...
</Target>
<Rule>
...
</Rule>
</Policy>

```

The main body of a policy consists of a **Policy Target** and one or more **Rules**.

### 11.1.3 Defining the Policy Target

A Policy Target is the part of a policy that specifies matching criteria for figuring out whether a particular policy is applicable to an incoming service request. A Target contains three basic "matching" components: **Subjects**, **Actions**, and **Resources**. All of these components must be matched to the context of an incoming request for the policy to be applicable.

```

<Target>
  <Subjects>
    ...
  </Subjects>

  <Resources>
    ...
  </Resources>

  <Actions>
    ...
  </Actions>
</Target>

```

A <Target> element is defined at the Policy level (as a child of the root <Policy> element). A Policy Target applies to any contained Rules that are expressed in that policy. However, a Rule may have its own Target, in which case the Rule-level Target overrides - for that Rule only - the Policy level Target. Typically, a Target defined at the Rule level is used to replace and so tighten a broader match specification found at the overall Policy level. (This is described below.)

The <Resources> element of a Policy Target is used to wrap one or more descriptions of the kinds of resources (objects, datastreams, disseminations, etc.) that the policy should apply to. At runtime, the Policy Enforcement Module will compare attributes of a requested resource against the criteria in the <Resources> specification within the policy Target to determine if the policy is applicable to the incoming request, or as below to any resource in the system.

```

<Resources>
  <AnyResource/>
</Resources>

```

Within a single <Resource> specification, there may be one or more attributes that together determine whether a policy match should occur. Each <ResourceMatch> element is used to specify the name/value of an attribute of a resource. Multiple <ResourceMatch> elements are used to specify multiple attributes of a resource, and are **logically AND-ed together**. This means that for a policy to be applicable to an incoming service request, **all** <ResourceMatch>

<b>File:</b> D3.5 - Security and privacy layer.docx	<b>D.3.5</b>
<b>Page:</b> 21 of 26	<b>Security and privacy layer</b>

specifications must match the attributes of the requested resource. The AttributeID in the <ResourceAttributeDesignator> element is used to identify a particular resource attribute by a URN, as defined in the policy vocabulary.

```
<Resources>
  <Resource>
    <ResourceMatch MatchId=" string-equal">
      <ResourceAttributeDesignator
        AttributeId=":datastream:id"
        DataType=" string"/>
      <AttributeValue
        DataType=" string">THESIS
      </AttributeValue>
    </ResourceMatch>
  </Resource>
</Resources>
```

To create an **OR condition** for resource matching, multiple <Resource> elements must be specified. If there are multiple <Resource> elements within the <Resources> wrapper component, the <Resource> elements are **logically OR-ed together**. This means that a match on only one of the Resource specifications is necessary for the policy to apply to a service request.

The <Actions> element of a Policy Target is used to wrap one or more service operations that this policy should apply to. At runtime, the Policy Enforcement Module will compare the identity of an incoming request against the criteria specific in the <Actions> of a Target in a policy.

```
<Actions>
  <AnyAction/>
</Actions>
```

The <Subjects> element of a Policy Target is used to wrap one or more descriptions of users or agents that this policy should apply to. At runtime, the Policy Enforcement Module will compare attributes of the user/agent making a service request against the criteria specific in the <Subjects> specification of the policy Target to determine if the policy is applicable to the incoming request. For example, to define a policy that is applicable to any kind of user or agent, the following is specified:

```
<Subjects>
  <AnySubject/>
</Subjects>
```

Within a single <Subject> specification, there may be one or more XACML attributes that together determine whether a policy match should occur. Each <SubjectMatch> element is used to specify an name/value of an attribute of a user/agent. Multiple <SubjectMatch> elements are used to specify multiple attributes of a subject, and are **logically AND-ed together**. This means that for a policy to be applicable to an incoming service request, **all** <SubjectMatch> specifications must match the attributes of the requesting user/agent. In the example below, there is only one attribute to match on (i.e., "i-locateRole"). The AttributeID in the <SubjectAttributeDesignator> element is used to identify a particular subject attribute by its local or global identifier. The snippet says that a policy match will occur if the incoming request context indicates that the user/agent has a role attribute with the value of "administrator."



```
<Subjects>
  <Subject>
    <SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">administrator
<
      /AttributeValue>
    <SubjectAttributeDesignator AttributeId="i-locateRole" MustBePresent="false"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
  </Subject>
</Subjects>
```

To create an **OR condition** for subject matching, multiple `<Subject>` elements must be specified. If there are multiple **<Subject>** elements within the `<Subjects>` wrapper component, the `<Subject>` elements are **logically OR-ed together**. This means that a match on only one of the Subject specifications is necessary for the policy to apply to a service request. For example, the snippet below says that a subject match will occur if the requesting user has the role of either “administrator” or “superuser.”

```
<Subjects>
<Subject>
<SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">administrator</AttributeValue
>
<SubjectAttributeDesignator AttributeId="i-locateRole" MustBePresent="false"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>

<Subject>
<SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">superuser</AttributeValue>
<SubjectAttributeDesignator AttributeId="i-locateRole" MustBePresent="false"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
</SubjectMatch>
</Subject>
</Subjects>
```

The **Environments** component of a Target is intended to specify aspects of the runtime environment that would make the policy match the incoming request. Such attributes include the current date, current time, the IP address of the client, and the protocol being used for the request. The **Environments** element is discussed in the OASIS XACML specification, but it is **not yet implemented by the Sun XACML engine**.

Although the **Environments** element is not currently supported by the Sun XACML engine for use within a Target, this does not mean that you cannot encode matching criteria for environmental attributes within a policy. Within a policy Rule, you can specify a Condition that contains matching criteria for environmental attributes.

### 11.1.4 Defining Policy Rules

Each policy has at least one, and possibly more, rules. There must be at least one Rule in a policy that matches the incoming request for a policy to be deemed applicable to that request. The way the Sun XACML engine determines whether a rule is applicable to an incoming request is by evaluating the Target and optional Condition (if it exists). These are ANDed together, and the rule's effect achieved if the ANDed value is TRUE. (If there is no Condition, this result is simply the value of the Target.) The rule's Target is so used, and if it has no Target, the policy's Target is used instead.

```
<Policy>
  <Target>
  ...
</Target>

  <Rule RuleId="1" Effect="Deny">
    <Target>
    ...
    </Target>

    <Condition>
    ...
    </Condition>
  </Rule>
</Policy>
```

A policy contains one or more Rules. Each rule has a **RuleId** and an **Effect**. An Effect is the intended consequence of a satisfied rule, which can be either "Deny" or "Permit." This means that if the rule is deemed applicable to an incoming service request, and the rule's conditions evaluate to TRUE, then the specified effect should be enforced.

Each Rule in a policy can have its own Rule Target. While a Policy Target generally describes the kinds of requests to which an entire policy applies, a Rule Target describes the kinds of request to which a particular rule applies. If a Rule Target is not present, the Policy Target is used to determine whether the Rule is applicable to an incoming request. When a policy target exists, it is applicable to every rule in the policy which does not have its own Target. In practice, a rule target is often more constrained than the associated policy target, fine tuning to specific Subject/Resource/Action match criteria that are in the context of a particular rule.

A Condition is a predicate that must be satisfied for a rule to be assigned its effect.

While Targets are appealing, frame-like expressions, they have a constrained logic which isn't always expressive enough to narrow down whether a policy is applicable to a service request. Hence, the need for Condition elements. If either the policy Target or the rule Target is not able to adequately express a constraint, a Condition can be added to a Rule. **A Condition can appear only within a Rule.** It cannot appear within a Target, nor directly under Policy or PolicySet. If a Condition is intended to be applicable to the entire Policy, the Condition must be repeated in every Rule in that Policy. Unlike the relationship of rule targets to policy targets, conditions do in fact begin with the associate (rule or policy) target, and proceed to further constrain that target.



### 11.1.5 XACML Functions

The XACML specification defines numerous functions that can be used in defining attribute match criteria in Targets and in defining predicates for Conditions and they include the following.

- **Equality predicates**
  - **String Equality** - urn:oasis:names:tc:xacml:1.0:function:string-equal
  - **Boolean Equality** - urn:oasis:names:tc:xacml:1.0:function:boolean-equal
  - **Date/Time Equality** - urn:oasis:names:tc:xacml:1.0:function:dateTime-equal
  - Others
- **Logical functions**
  - **OR** - urn:oasis:names:tc:xacml:1.0:function:or
  - **AND** - urn:oasis:names:tc:xacml:1.0:function:and
  - **NOT** - urn:oasis:names:tc:xacml:1.0:function:not
  - others
- **Comparison functions**
  - **Greater Than** - urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
  - **Less Than** - urn:oasis:names:tc:xacml:1.0:function:type-bag
  - **Greater Than or Equal** - urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
  - **Less Than or Equal** - urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
  - **Date/Time Greater Than** - urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than
- **Bag and Set functions**
  - **Bag of Strings** - urn:oasis:names:tc:xacml:1.0:function:string-bag
  - **Member of Set** - urn:oasis:names:tc:xacml:1.0:function:type-at-least-one-member-of
  - others

Below is an example Condition that uses several of these functions. This Condition evaluates to TRUE if the client IP address (from the environment of the incoming request) is NOT a member of a set of privileged IP addresses. The Condition element itself contains an outer-most function which is a **negation function**. Within the condition, we see the application of the **set membership function**, which specifies that the environment attribute "clientIpAddress" should be evaluated. Finally, the inner most **bag function** wraps a set of possible values for the clientIpAddress attribute. Again, if the clientIpAddress on the incoming request is not one of those in the bag of addresses, then the rule's Deny effect should take place.

```
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
```

File: D3.5 - Security and privacy layer.docx	D.3.5
Page: 25 of 26	Security and privacy layer

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
  <EnvironmentAttributeDesignator
AttributeId="urn:fedora:names:fedora:2.1:environment:httpRequest:clientIpAddress"
  >
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string"/>127.0.0.1</AttributeValue>
  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">128.84.103.11</AttributeValue>
  </Apply>
</Apply>
</Condition>
```

In summary, each policy must have at least one Rule. For a Rule to have an effect, (1) the Rule must match the incoming request by virtue of a Target match (either via a policy Target, or a constraining rule Target), and (2) if a Condition is specified, the condition predicate evaluates to TRUE. An applicable rule will result in a Permit or Deny for an incoming request, based on what is specified in the Rule Effect.

### 11.1.6 Required vs. Optional Attributes in a Policy

There are times when an attribute that is referred to by a policy target will not be available on an incoming service request. By default, when the policy matching activity occurs - and an attribute specified in a policy is not found in the incoming request context - an Indeterminate result is returned and an authorization exception is thrown.

### 11.1.7 Declaring mandatory attributes

Whilst the default is to assume that an attribute if declared in a target it is always required and thus mandated. A “belt and braces” approach is to achieve this by setting `MustBePresent="true"` on a `SubjectAttributeDesignator`, `ResourceAttributeDesignator`, `ActionAttributeDesignator`, or `EnvironmentAttributeDesignator` element.

### 11.1.8 Declaring optional attributes

Policy authors can avoid unwanted Indeterminate results by indicating in the attribute designators of a Target or Condition that a particular attribute can be considered optional in terms of whether it must exist in the incoming request context. This is done by setting `MustBePresent="false"` on a `SubjectAttributeDesignator`, `ResourceAttributeDesignator`, `ActionAttributeDesignator`, or `EnvironmentAttributeDesignator` element.