European Maritime Safety Agency

Access Management Requirements

v.0.5

Author: EMSA Dept. 3
Date: 19 June 2020

Table of Contents:

EMSA Access Management Requirements

1 Introduction

The purpose of EMSA's Access Management solution is to provide horizontal support services to all EMSA Maritime Applications (MarApps) in the following areas: Authentication of accounts (including Single Sign-On/Out capabilities), Authorization to access resources, Password management capabilities. Excluded from this list is the actual managing of a user's account. This is the responsibility of EMSA's Identity Management solution.

In addition to the information provided in "Appendix TS4 - System_and_Application_Technical_Landscape_V36_published", the purpose of this particular document is to provide a list of requirements to be met by the substitution of EMSA's Access Management solution by upgrade of the existing Oracle Access Manager 10g to the latest version available.
The list of requirements is classified in two categories:

- Mandatory: currently implemented in the actual system and therefore, in the scope of this upgrade project
- Optional: to be seen as future implementations.

Offers for the current tender must consider the Mandatory items.
Optional items are presented for information and knowledge that EMSA plans also to implement them in a near future; however, if any of the optional items is foreseen impossible due to a constraint rising from the proposed architecture or implementation, that fact must be stated in the bid.

Within their bids, bidders shall describe as detailed as possible how mandatory requirements will be addressed, upgraded or implemented.

2 Provide Authentication of accounts for use in EMSA's MarApps and supporting systems

REQ_1 **MANDATORY**

The proposed system should provide the capability of authenticating existing EMSA MarApp accounts. The purpose of this requirement is to provide a means for effectively protecting access to EMSA resources by validating the correct identity of a user. As an end-result, all existing accounts have to be capable of authenticating against the new system.

2.1 Authentication of existing MarApp accounts

Authentication is currently implemented at EMSA by means of the following mechanism:

- A user attempts to access some URL that belongs to EMSA's infrastructure. For the sole sake of clarity, let's assume that the URL is the base entry point to EMSA's Liferay Portal.
- The user will be shown a "Login Screen" that is typically composed of four sections:
    - A placeholder for the user's accountId (marked *Username* in the screen);

  - A placeholder for the user's Password (marked *Password* in the screen;
  - A *link* to Login to EMSA's infrastructure (marked by the text *Login* and followed by the symbol of a key);
  - A *link* to a functionality that allows recovering the user's password (marked appropriately by the text *LostPassword*)
- After filling in the fields corresponding to the user's credentials, i.e. the accountId and the password, the user submits the form via the Login link (via HTTP Post). The Login URL is registered in the OAM access policy configuration section. Please note that more than one such link may currently exist configured in the system.
  - The login processing URL is in first instance sent to a cluster of Apache Web Servers that are running specific Oracle Modules implementing what Oracle calls a WebGate. This module talks directly to the Oracle Access Manager component to validate all decisions taken.
- EMSA's current Access Management System (OAM) will proceed to validate the credentials against those existing in EMSA's OpenLDAP system. At this time one of the following situations will arise:
  - The accountId does not exist and as such an "Invalid Credentials" message will be returned. This situation is registered in the audit logs;
  - The accountId does exist and is treated as described in the following requirements;

REQ_2	**MANDATORY**

The validation of a user's identity is done by asserting his credentials, i.e. an account_id and a password. As the current implementation of access control at EMSA is based upon an account_id and a password, for compatibility purposes this should be the preferred way to authenticate the account.

2.2 Validation of credentials

As previously mentioned, the user's credentials are submitted to a URL being handled by the OAM/WebGates. It will attempt to match the provided accountId with one existing in the OpenLDAP directory. If such an account exists, for example, it will encrypt the password submitted and compare it to the one existing in the LDAP entry. A successful match of both fields means that the user is correctly authenticated.

REQ_3	**OPTIONAL**

A user's identity can be asserted by a technical means other than that of simple account_id/password. For example, by use of 2-way SSL with a client certificate issued by EMSA.

2.3 Other technical means of authentication

EMSA currently only uses the AccountId/password being submitted to a known control URL for authentication so this requirement is considered as a nice to have. An important point to take into consideration for this requirement is that the SSL terminates in the F5 and as such is not propagated to the WebGates that communicate with OAM.

REQ_4 **OPTIONAL**

Multiple technical means of authentications should be made available. Currently some of EMSA's MarApps are deployed using technology other than simple HTTP requests. For these, alternatives to simple HTTP Basic Authentication should be provided. An example of such an alternative is a JSON service to authenticate an account.

2.4 Multiple technical means

The technological stack in current use at EMSA consists of an Oracle bespoke module loaded into Apache Web Server, i.e. the WebGate that interacts with the OAM (Oracle Access Manager). The current requirement relates to the fact that EMSA has applications that attempt to authenticate accounts by use of "pseudo JSON" service implemented by EMSA. The service is designated as "pseudo JSON" because most of the results returned by the service are "pure" JSON messages but in some cases the result is an HTML message instead. The current problem is that the WebGate only "speaks" HTML and as such does not allow a "pure" JSON message to be generated. Any new technology to be adopted should natively allow the use of other alternatives to pure HTML.

REQ_5 **MANDATORY**

Multiple technical means of authentications should be made available in parallel with the main technology. Further alternatives to simple HTTP Basic Authentication should be provided. Examples of such alternatives are the use of SAML, OAuth and openID. These technologies can be aimed at "future" systems, but they must keep compatibility with the existing systems (in the sense that an authenticated account has both access to a new system using OAuth and also to a legacy system being passed an HTTP header variable – SM_USER, for example).

2.5 Multiple Parallel technical means

EMSA's Access Management solution is critical to EMSA's operational success. It has been working smoothly for almost 10 years now and any change made should not introduce issues to EMSA operations. The message behind this requirement is that, while new technologies are accepted and embraced, legacy systems should keep working seamlessly until such time as the MarApps can be upgraded to use all of the benefits of the new technologies.

REQ_6 **OPTIONAL**

The proposed solution should provide multiple authentication access points, due to different behaviours and technologies used. As a complement to the previous points, REQ_3 and REQ_4, the solution should be able to cope with different "entry points" for validating account authentication. This is a direct consequence of the various technologies involved but also an indirect consequence of EMSA publicly exposing multiple URLs for serving MarApps to its set of users (for example: portal.emsa.europa.eu, eulritdc.emsa.europa.eu, etc.).

2.6 Federation

The simplest terminology relating to this requirement is Federation. The term was not used directly out-forth because: first, each vendor interprets federation in its own way, and second, all of EMSA's sites are under the same base domain and as such not a true federation (as Oracle states

it). The current implementation used at EMSA is based on Oracle products. The requirement is fulfilled even though Oracle does not call the solution a Federation (as they have a different product for their take on federation). Implementation wise, EMSA runs an autonomous instance of an Apache Web Server for each published URL. Each instance has its own base configuration file while sharing other common configuration files thus allowing a flexible solution that can be stopped independently of all others while at the same time allowing a single point of configuration for any given MarApp independently of how it is accessed. An additional benefit is the fact that if the same application is access via different Apache instances (i.e. base URLs), then it can display different behaviours. An example of such is the SEG interface that has multiple implementations each accessible via a different starting point while the rest of the MarApps continue to work the same via ll of those different URLs (i.e,."normal" SEG versus HPSEG – High Performance SEG).
A nice to have that can be included in this requirement is the use of a true federation service (i.e. one that spans "real" multiple domains and not only sub-domains as up to now).

REQ_7 **MANDATORY**

The validation of a user's credentials should be done against data existing in EMSA's LDAP infrastructure. To be valid, the account_id must exist in EMSA's supporting LDAP infrastructure and the password must match with that existing in the same LDAP structure for the given account.

2.7 EMSA's LDAP is OpenLDAP

Currently EMSA relies on OpenLDAP to support a myriad of systems and applications. How an account is validated has already been explained so now we will explain why OpenLDAP is important.
As stated previously, EMSA's Access Management solution has been stable for a very long time and as such it is important that a new solution to be adopted does not break current compatibility with the OpenLDAP structure. Great care has to be taken to validate that any new solution can live peacefully with existing legacy systems.

REQ_8 **OPTIONAL**

The proposed solution should be able to authenticate system accounts and not only human accounts (i.e. user accounts). It should be noted that EMSA's LDAP structure contains slight variations as to what is considered a human account and what is considered a system account. The main difference is related to the schemas associated/allowed for each type.

2.8 Human versus System

"Human" accounts must contain various structural objectClass's in LDAP, namely: *emsaPerson*, *inetOrgPerson*, *person, organizationalPerson* and *top*. System accounts mostly only have the *externalSystem* objectClass (note that there are exceptions with more objectClass's). The current implementation of EMSA's OAM also adds *oblixAuxPerson4LPM*, *oblixorgperson*, and *oblixPersonPwdPolicy* for authentication / auditing purposes.
It is important to note that these differences are very important in terms of provisioning, but they are not that relevant in terms of authentication as the main fields used are the *uid* and *userPassword*.

Normalization of the LDAP attributes needed to harmonize the two account types can be done in a later stage of the project.

REQ_9	**MANDATORY**

EMSA's LDAP infrastructure contains an EMSA specific schema that may contribute to the authentication of the account. For human accounts, the current implementation of EMSA's LDAP schema contains a field named "Status" that must contain the value "TRUE" for the user to be authenticated. This requirement can be ignored if a similar validation is done at the Authorization level, as such inhibiting access to EMSA resources when the value is "FALSE" (see reference REQ_25). The decision for which option to implement can be delayed to a later date thus allowing some flexibility in the solution. For system accounts, the "Status" field does not exist so there is a possible conflict over REQ_8.

2.9 EMSA specific schema

The objectClass *emsaPerson* defines some EMSA specific attributes associated to "human" accounts in LDAP. It is important to access if the new Access Management solution is (or can be) compatible with LDAP schema extension as this may imply modification of part of EMSA's infrastructure.

REQ_10	**MANDATORY**

The solution should provide a means of logging authentication attempts for auditing purposes. All failed attempts to authenticate must be registered. Successful authentications may be logged or not but ideally should.

2.10 Auditing Authentication attempts

EMSA's current access management solution provides authentication auditing via two distinct forms: by using custom fields contained in the LDAP schemas (i.e. oblastsuccessfullogin, oblastfailedlogin, obpasswordcreationdate, etc) and by registering information in specific database tables (i.e. table OBLIX_AUDIT_EVENTS in the OAM_Audit schema). The attributes in LDAP are primarily used for the actual access control and not so much for auditing purposes but they are legible and understandable enough to be used for auditing. The DB tables have the advantage of maintaining historic values while the LDAP contains only the last entry.

REQ_11	**MANDATORY**

The solution should provide Single Sign-On capabilities (SSO). As mentioned earlier in this document, as EMSA has exposed various URLs for accessing MarApps, the user should not be forced to re-authenticate himself if he ends up accessing more than one MarApp via more than one of the available access points.

2.11 Single Sign On

This is such an important feature that it has "gained a name for itself". Inside of EMSA, most people refer to IdM as the user management system (attribute management) and also refer to

SSO as the access management component. SSO was the "one big benefit" that has tied IdM together.
The current implementation uses the out-of-the-box capacity of OAM to provide a single sign-on solution. Technically, the use of an HTTP session cookie ObSsoCookie that is managed by OAM via WebGate controls the status of the session allowing further access (if the user is authenticated) or imposing a login screen (if he is still not authenticated). The SSO session is configurable and currently set to 90 minutes of inactivity for session expiration. Any "movement" via the WebGate / OAM will revalidate the session for a further 90 minutes or up to the maximum limit of 24 hours after which a new login is required.
Part of the configuration for SSO is the definition of the sub-domains that are allowed to be accessed with the session token (previously referred cookie). It should be noted that EMSA does not have in place Oracle's Federation Server allowing the integration of domains outside of EMSA (for FRONTEX or the European Council, for example).

REQ_12 **OPTIONAL**

The establishment of an authenticated session should be useable via multiple technologies. A session that is established via one particular technology, for example via HTTP Basic Authentication, should be transferable to other technologies accessing the same conceptual resources, for example a web service via JSON or OAuth session. Final implementation of this requirement shall be subject to a deeper and detailed analysis in order to understand if potential security issues may raise from it.

2.12 Session via multiple technologies

As previously mentioned, EMSA's SSO session is monitored via an HTTP session cookie (ObSsoCookie). Any technology that cannot have access to the HTTP variables to obtain and send to the server the referred cookie, will have issues in maintaining a session. This is already true for COTS solutions such as Adobe Acrobat Reader and other technologies that may or may not reflect back the cookies given to them. The use of the APIGateWay (AxWay product) has somewhat minimized the issues related to this but the integration is still not complete or perfect.

REQ_13 **MANDATORY**

MarApps and/or systems being protected should be allowed to be agnostic of the authentication scheme used. This requirement states that any MarApp or system effectively protected by the solution should not have to interact in any way with the actual technology that was used for authenticating the account. This could be summed up by explaining that the current Access Management system implemented at EMSA always passes an HTTP Header variable to the "protected" systems, the content of which is the account_id of the authenticated user.

2.13 Authentication Agnostic

EMSA's current access management solution relies heavily on the use of HTTP, namely the passing of an HTTP header variable named SM_USER to the back-end server systems. Any server located within EMSA's infrastructure, knowing that it is protected via OAM, will assume that the user has

been correctly authenticated if it finds the SM_USER variable in the HTTP headers it receives. Current industry standards no longer rely on such frail mechanisms as passing HTTP headers to applications to guarantee safeguard of authentication.
The main issue of using a newer technology is maintaining compatibility with older legacy systems that cannot adapt in "useful time".

REQ_14	**MANDATORY**

Authentication knowledge should be passed to the protected MarApps via the use of an HTTP Header variable named SM_USER that contains the account_id of the authenticated account. This requirement is a complement of REQ_13 and is necessary for maintaining compatibility with the existing access management solution. If this requirement is not fulfillable then the impact on EMSA's infrastructure will be huge.

2.14 Backward Compatibility

As mentioned, the current access management solution makes exclusive use of an HTTP header named SM_USER to communicate to back-end servers that a user is correctly authenticated. As was also mentioned, EMSA still runs legacy systems that need to have this mechanism in place to be able to continue working correctly. Any new solution being adopted will have to provide a means of passing the HTTP variable with the account Id (as a minimum because some applications need more than this – but always as HTTP headers) OR the legacy applications will have to be changed in the scope of the new project to cope with the new technology to be used.

REQ_15	**MANDATORY**

The solution should provide Single Logout capabilities (SLO). Similar to logging in via single sign-on, when logging out of one MarApp, the logout should be effective for all MarApps. This is especially important as the various MarApps and/or horizontal systems/platforms that are protected by the SSO solution are not aware of each other and as such tend to maintain their individual sessions until otherwise notified.

2.15 Single Logout

The use of a non-industry standard form of propagating authentication information (such as an HTTP header) brings various challenges to EMSA. One of the most difficult to overcome was the successful invalidation of all individual sessions that a user may have established during his current SSO session. An example of this will be: OAM will establish the global session token for validating accesses while Liferay Portal will generate its own session token (typically an HTTP cookie) as may other applications running "under" the Portal (i.e. applications running as Portlets within Liferay Portal). Likewise, if the user side-steps the Portal and accesses a MarApp that is self-contained (such as LRITDC), it will create its own session token completely ignoring any other that may exist.
The problem with logging out is that it can be done from any context (for example logged in via Portal but logged out via LRITDC) and there is the need to guarantee that all "visited" applications are informed that the user is logging out so that they may proceed to do their "house-keeping" and remove any session tokens for the user logging out. This becomes especially tricky if one

considers that some of the MarApps run under separate sub-domains thus fouling a centralized mechanism for cleaning session cookies (due to cross site scripting issues).
EMSA currently maintains a list (stored in an HTTP cookie) of all "visited" applications for the session such that when any of the logout links are followed, a JavaScript + XDM script will invoke the respective logout URLs for each application attempting to guarantee that all MarApp sessions are invalidated. This is done via Apache redirect rules (within each instance running for each sub-domain) as well as via Lohout.html and Logoff.html pages also served by the Apache Web Servers.

REQ_16	**MANDATORY**

The solution should guarantee all sessions are invalidated when logging out. A problem with EMSA's current Access Management solution is that there are (rare) occasions upon which the action of logging out is not complete through all accessed MarApps and the newly logged in account is interpreted as still being the previously logged in account.

2.16 Invalidate all sessions

This is a direct follow-up of what was previously stated. All MarApp sessions must be invalidated once any of the possible logout links is invoked. Under very specific circumstances not under EMSA's control, sometimes the user's browser will not completely execute the necessary code for invalidating a MarApp session. The consequence is that once the user logs in with a different account, he may see the "old" account still active in some applications instead of the newly logged in account.
The good news is that this is mostly an issue for EMSA users that have multiple distinct accounts and that for business reasons do a lot of login-logout in a tight sequence. Most end users only have a single account and as such do not experience the issue as frequently. However, this does not in any way diminish the severity of the problem of showing an erroneous account to someone just logged in.

REQ_17	**MANDATORY**

The solution cannot allow account enumeration. Invalid account_ids or invalid passwords should be treated in the exact same fashion such that it is not possible to enumerate accounts by means of information resulting from a failed login attempt.

2.17 Account Enumeration

As the internet becomes older, it becomes a more hazardous and dangerous place. Every day an army of malicious hackers will attempt to break in to EMSA's systems using any means possible. One such means is by attempting to use a valid account for spoofing the systems. EMSA has taken great care to try and not divulge the list of valid accounts used within the MarApps. An example of such care is the "Lost Password" mechanism that allows any value to be used as an accountId always treating it if it was a correct value indicating the internal ticket number as well as indicating that a message has been sent to the account holder. Obviously if the account does not exist, nothing further than auditing will happen, but the attacker does not know if he has succeeded in discovering a valid accountId or not.

The downside of this is that when a "real user" misspells his accountId, he will never receive the email for setting a new password and will inevitably report this as a bug in the system. EMSA prefers to deal with these "bug-reports" rather than risk the exposure of accounts.

REQ_18	**MANDATORY**

The solution must support the concept of invalidating a session after a fixed pre-determined period of time, irrespective of activity during that period. The proposed solution should be such that an authenticated session cannot "live forever". After a specific (configurable) amount of time, the session should be invalidated irrespectively of the users executing actions or not. An example of such a limit is 24 hours.

2.18 Invalidating Sessions

In section 2.11~~2.11~~ Single Sign On, mention was made to a maximum time limit for a session (24 hours). The value established was a trade-off between tighter security (invalidating a session based on a short period of time) and business needs for maintain a valid session for as long as possible (due to MSS support, for example). The current OAM product allows the definition of the value to be used for the maximum period a session can remain valid (even when being constantly used – producing traffic via the WebGates). The value is established via the OAM configuration web application.

REQ_19	**MANDATORY**

The solution must support the concept of invalidating a session due to inactivity. The proposed solution should be such that an authenticated session expires after a pre-determined (configurable) amount of time if there are no actions performed during that period. By no actions one can intend no operations (such as HTTP traffic for example) that make use of the Access Management services. An example of such a limit is 90 minutes.

2.19 Inactivity Session Invalidation

For security reasons, an "abandoned" session must time-out and invalidate itself. EMSA's current solution via OAM has such a mechanism via a configurable value (set via web interface of OAM) that defines the number of minutes that a session that is NOT producing traffic via the WebGates is allowed to live. The current value set is 90 minutes which, again, is a trade-off between tight security and business needs. It should be noted that all systems that allow the definition of such a value are configured for exactly the same amount of time to allow a synchronous invalidation in all systems.

It should be noted that any traffic that does not go through the WebGates via pre-configured URLs (direct JSON requests, for example) will not revalidate a session and may thus be cut-off after the session invalidates.

3 Provide Authorization for accessing Resources

REQ_20	**MANDATORY**

Authorization for accessing resources should be based upon an RBAC model. The predominant model used at EMSA for authorization of access to resources is the RBAC model (Role Based ACcess model). Through the RBAC model, a coarser grain of authorizations can be used (for example based solely on membership of an LDAP group).

3.20 RBAC Model

At EMSA, each MarApp has the flexibility to define its own level of authorizations according to its own needs. Some MarApps only want authorization to go as far as having access or not whilst others go all the way down to requesting individual Permissions.
EMSA has adopted the RBAC model as the "norm" in terms of authorizing access rights.
It should be noted that MarApps needing individual permissions obtain these for a given user based upon the UserInfo webservice available via the IdM provisioning component.

REQ_21	**MANDATORY**

The solution proposed should allow a single optimized configuration for sets of logically grouped resources. EMSA's current Access Management system implements the concept of an "Access Policy" that is nothing more than a grouping of resources that are to be managed by the same set of access rules or rights, i.e. they share a common set of needs in what regards access management.

3.21 Access Policy Domain

EMSA's current access management solution implements to concept of a Policy Domain that defines a set of Resources (URLs), Authorization Rules, Policies and the groupings of how these work together, this for any particular "logical" grouping (an example of which is each individual MarApp). The above concepts are standard for Oracle Access Manager 10g (10.1.4).
The idea behind this concept is that for a particular logical entity (for example a MarApp), all access validations are treated in the exact same fashion and as such should only be declared once and re-used for the rest of the application.

REQ_22	**OPTIONAL**

Fine grain authorizations should be supported. Coarse grain access rules practically match a Role or given set of Roles. In some circumstances, a finer level of authorization control is needed, namely that based upon DATA. An example of such a data-based authorization could be filtering access by use of an HTTP GET parameter.

3.22 Fine Grain Authorization

This is a "nice to have" as EMSA's current access management solution does not support this concept. Some EMSA MarApps internally use such a concept but it is implemented by OES (Oracle Entitlement Server). It should be noted that Oracle is phasing out support for OES as it no longer continues developing the product.
The current trend for substituting OES is to invoke the UserInfo web service and interpret some of the "roles" as "permissions".

REQ_23	**MANDATORY**

The authorization solution should be auditable, i.e. logging of authorization validation. Similar to the authentication process, but with potentially less interest, the authorization validation process should be subject to logging and thus be auditable.

3.23 Auditable Authorization

This is another "nice to have" requirement as currently EMSA does not have auditing capabilities on authorizations. EMSA can enable detailed logging for debugging purposes but there is no correlation with any actual account, so no auditing is possible.

REQ_24	**OPTIONAL**

On-request logging of authorization validation. Due to the large quantity of logging information foreseen for authorization requests, the logging should have a switch on/off mechanism and as such only be activated upon request.

3.24 On-request logging of Authorizations

As stated in the previous requirement explanation, EMSA has the capability of enabling or disabling logging at the Apache Web Server level thus permitting debugging of abnormal situations. A current inconvenience is that this affects all traffic passing through the Apache Web Server instance that has logging enabled.

REQ_25	**MANDATORY**

EMSA's LDAP infrastructure contains an EMSA specific schema that may contribute to the authorization of the account. For human accounts, the current implementation of EMSA's LDAP schema contains a field named "Status" that must contain the value "TRUE" for the user to be authorized access to any resource.

3.25 EMSA specific LDAP schema

Under the authentication section of the requirements, it was already mentioned that EMSA has added a specific schema to the OpenLDAP (objectClass *emsaPerson*). One attribute of the *emsaPerson* class is STATUS which should have a TRUE/FALSE value. When TRUE, the user is globally enabled and can thus access any EMSA resource (to which he has the corresponding privileges, of course). If the field contains FALSE, then the Access Management system will not allow the user to login stating that the account is disabled.
As the logic associated to the STATUS field may be used in more contexts than that just mentioned, it is important that the new Access Management solution accept extensions to the base LDAP used in the product.

REQ_26	**OPTIONAL**

The authorization solution should be applicable to system accounts. EMSA supports the concept of system accounts which are those typically used for system-to-system communication (as opposed to human accounts that are used for direct User Interaction, typically via a GUI). EMSA's current Access Management solution does not contemplate directly the concept of securing system-to-system interfaces.

3.26 System Account Authorization

EMSA's current access management implementation does not contemplate securing system-to-system accesses. Possible reasons for this are: the communication is not done via HTTP and therefore not treatable by the WebGate infrastructure; the system accounts in LDAP do not contain the necessary objectClasses that are needed for handling authorization; the communication end-points fall outside of the sub-domains protected by the current access management solution; and the EMSA IdM team is not made aware of the communications occurring between systems and as such cannot put in place a mechanism to secure those communications.
The new access management solution should contemplate these pitfalls and (ideally) provide a means for integrating all access security in one solution.

REQ_27	**MANDATORY**

The authorization solution should be applicable to parts of the URL being used to access the systems. The base component of the URL being used (see REQ_6) should be irrelevant and the path component should be the main driver for authorization. A practical example is: https://rulecheck.emsa.europa.eu/identity and https://portal.emsa,.europa.eu/identity are two possible entry points for the exact same resource (in this particular case – access to Identity Management) and as such should have the exact same authorization policy applied, irrelevant of the base being used (i.e. rulecheck or portal)

3.27 Authorization of partial URLs

EMSA currently uses the OAM tools capability to base a filter not on an entire URL itself, but on parts contained within the URL. An example is the exclusion of the domain from the access filters using only the relative portion of the URL to limit access to resources. This effectively allows the "same" resource to be treated differently based on the Apache WebGate instance that handles the request (please remember that each instance is associated to an EMSA sub-domain). An example of this usage is for SSN training. The relative URI /ssn-gi can be authorized for training accounts and redirected to the SSN training server if the base URL is https://portal-training.emsa.europa.eu whilst the exact same URI can authorize "normal" pre-production accounts and redirect to the SSN pre-production server if the base URL is https://portal-pp.emsa.europa.eu. A similar situation is set-up for IMDatE via access to a "normal" SEG or to the HPSEG (High Performance SEG) in the production environment.

Field

Field

REQ_28	**MANDATORY**

The authorization solution should be applicable to parts of extra-information within the URL being used to access the systems. EMSA supports the use of HTTP GET parameters whose value is directly linked to authorization schemes. An example of such a rule could be contentId=99*99 versus contentId=88*88 (with * denoting any combination of numbers therein) in which each case could have different access rights. A final URL would be something similar to https://rulecheck.emsa.europa.eu/srcweb?contentId=991234599.

3.28 Authorization of partial URLs based on extra-information

A compliment to the previous requirement is the ability to base access authorization on variables contained within URLs (via GET parameters). Prior to version 12 of RuleCheck, access authorization was done via OAM by evaluating the content a user was trying to access and applying restrictions based on the LDAP groups he belonged to. As from version 12 of RuleCheck on, the application has incorporated access authorizations based on "roles" and therefore this requirement can now be classified as a "nice to have".

REQ_29	**MANDATORY**

The authorization solution should allow the possibility of choosing the strategy: Black list versus White list. The default behaviour should be the White List approach in which access is denied to every URL by default. For access to be authorized, it must be specifically listed or be caught in a rule (see REQ_28) allowing access. The Black list approach allows access to all resources expect those that are explicitly denied, again via list or rule.

3.29 Authorization Strategy

EMSA has historically attempted the two approaches but currently favours the White List approach. By default, any "new" URL is automatically denied unless it is included in the WebGate Reverse Proxy rules and as a Resource in any of the OAM Policy Domains.

REQ_30	**MANDATORY**

The authorization solution should support the concept of Private and Public resources. Any resource protected by the proposed solution could theoretically be classified as either private (needing explicit authentication and authorization to be accessed), or public (allowing anyone to access).

3.30 Private vs Public Resources

Most of EMSA's MarApps are completely "private" in the sense that only authenticated and authorized access is allowed. Nonetheless some MarApps have sections that are readily available to the general public, and thus not needing authenticated access. Part of EMSA's horizontal platforms also provide public functionality (for example, the Liferay Portal). In this sense it is necessary to distinguish between content (Resources) that can be freely accessed versus content that is subject to access rules.

It should be known that the tool used by EMSA (OAM 10g) considers all content as "private" in the sense that it will do validation of access rights. Whenever EMSA wants to provide resources that are truly "public", they are not registered in OAM but typically only in the Reverse Proxies that are the WebGates.

Coming back to "public" resources in OAM, the trick to circumvent the tool is to create Access Policies that use "anonymous authentication" (meaning no authentication is required) as well as configuring the access rule as "allow anyone". Why is this not a truly "public" access? Because under certain circumstances the tool will identify the accountId accessing these resources as OblixAnonymous and it will set this value in the SM_USER http header. Any MarApp that is under OAM protection will internally have to make the check and ignore the accountId OblixAnonymous if it is ever received by the application.

REQ_31 **MANDATORY**

The authorization solution should support the concept of an anonymous user when Public resources are accessed. A public resource, by definition, is automatically granted access to by everyone. Therefore, account information is normally not relevant for public resources. However, there are cases in which the underlying systems do not technically support the inexistence of account information and in those cases an "anonymous" account must be provided.

3.31 Anonymous public access

As explained in the previous requirement clarifications, under specific circumstances OAM will set the SM_USER variable to OblixAnonymous whenever a "public" resource is accessed. If the user is correctly authenticated with a valid accountId, then OblixAnonymous should never be sent. However, if the user is accessing "public" content protected by OAM and did not authenticate with any accountId, OblixAnonymous will be sent to the protected applications.
It should be noted that at the present moment there are probably no EMSA MarApps that require an anonymous accountId be sent for "public" content, but this has to be validated case by case.

REQ_32 **OPTIONAL**

The authorization solution should be auditable, even when a Public resource is accessed. REQ_23 (logging of authorization actions) and REQ_24 (On-request logging) should both be applicable to Public resources.

3.32 Auditing public resources

EMSA's current implementation logs all accesses to configured resources as it internally considers all resources to be "private" and "protected". However, as explained in previous requirements, under specific circumstances the accountId will be OblixAnonymous instead of an actual, "real", accountId. All accesses done via OblixAnonymous are registered in the exact same fashion as for a normal account.

4 Provide Password Management capabilities

REQ_33 **MANDATORY**

The proposed solution must provide Password Management capabilities. The solution will have to handle such aspects as password policies, password expiration dates, lost password management, etc.

4.33 Password Management

This is a generic requirement that is fully detailed by the following requirements, so a detailed description will not be made at this time.

REQ_34 **MANDATORY**

The solution must permit the establishment and enforcement of a Password Policy. Accounts created via EMSA's Identity Management system set a new password upon account creation. The

solution must have the capability to set a password policy that can be compliant with EMSA's current setting.

4.34 Password Policy

EMSA has established a set of rules for acceptable passwords. These rules are based upon what is considered industry "best practices". Any account that is to be protected via EMSA's access management should comply with the pre-defined password policy.
Even though EMSA's password policy has been stable for quite some time now, the new access management tool should allow on the fly configuration of password compliancy rules.
It should be noted that currently the password policy has to be set in two different systems: access management (for normal management functions) and identity management (for the initial password creation).

REQ_35	**MANDATORY**

The proposed system must be able to manage password expiration by dates. Part of the security hardening of EMSA's access management platform is the mandate that users change their passwords on a frequent basis. The frequency of this change should be configurable via the solution.

4.35 Password expiration by date

As mentioned earlier, one of the changes made to the OpenLDAP used at EMSA is the inclusion of specific objectClass schemas. OAM adds a series of such schemas to be able to use extended attributes for various purposes. One such purpose is the control of password expiration.
The attribute being used by OAM for validating the password expiration is named *obpasswordcreationdate* and contains a timestamp with the moment of last change of password (done via OAM functionality – manual changes directly within LDAP are not considered). An example value is 2019-06-01T00:00:24Z.
The actual value of the duration of the validity of the password is configurable within the OAM configuration web pages.

REQ_36	**MANDATORY**

An account should be informed that a password is about to expire. As a consequence of REQ_35, the users should be warned that their account password is about to expire at a certain period. Logging in should not be affected by this warning.

4.36 Password about to expire

In the previous requirement, two facts were mentioned: there is an attribute in the OpenLDAP that contains the date of last change of a password (via OAM functionality) and, there is a parameter set within the OAM configuration web pages to indicate the duration of validity of the password.
Within the OAM web pages there is another parameter in which one can specify how far in advance a user will be warned that his password is about to expire. EMSA configurations use a value of 5

days so effectively a user will we warned that his password is about to expire during the 5 days prior to actual expiration (expiration being the creation date + validity duration).

REQ_37 **OPTIONAL**

Password expiration should be available via multiple technologies. One of the shortcomings of EMSA's current access management system is the fact that password expiration is always presented to the user in HTML format. MarApps that use the JSON format for authenticating accounts are sometimes confronted with invalid JSON content (i.e. the previously referred HTML page).

4.37 Password expiration via multiple technologies

EMSA's current access management solution, based on OAM, only presents an HTML page with customizable content indicating that a password is about to expire. The issue with this is that if authentication is done via JSON, the return of the JSON call may be HTML instead of a correctly formed JSON message indicating password about to expire.
As EMSA does not have this functionality currently, this is considered important if the new solution allows technology other than simple HTML pages (current solution).

REQ_38 **MANDATORY**

Expired passwords must mandatorily be changed. Upon password expiry (i.e. the current date is later than the password expiration date), the current password must be reset to a new value for the user to access any application or system protected by the proposed solution.

4.38 Expired Passwords

By force of the previous requirements related to password expiration, as soon as a password has expired (current date later than password creation date + validity duration), the account should not be allowed to complete the login process. Under such circumstances, an HTML page stating the fact is returned to the user requesting that the user introduce his current (expired) password, as well as define a new password (twice for typo validation). After the correct definition of a new password, the user must repeat the login process again to gain normal access to EMSA systems.
A current pitfall associated to this behaviour is when applications attempt to authenticate an account via JSON messages and end up receiving an HTML response (the aforementioned page) instead of a correctly formed JSON message indicating the issue (hence the need for the "multiple technology" requirement).

REQ_39 **MANDATORY**

Expiring and/or expired passwords should not allow account enumeration. The proposed solution should not allow the possibility of account enumeration due to passwords reaching the pre-expiry period nor after they have actually expired.

4.39 Enumeration of accounts via expired passwords

As explained in previous requirements, an attempt to authenticate an account with an expired or about-to-expire password shows a different behaviour from an account with a normal timeframe

of password validity. A malicious attacker can use this difference of behaviour to enumerate existing accounts.
EMSA currently does not have the capacity to impede this enumeration.

REQ_40	**MANDATORY**

The proposed solution should have a password history capability to avoid repeated passwords (within each account). In order to guarantee password variability, a history mechanism should exist allowing at most the last N different passwords (N being a configurable value) for each account.

4.40 Password History

The current OAM solution used at EMSA has the capability to validate that a newly defined password is part of a list of *n* recently used passwords and thus inhibit the use of such password. The actual value of *n* is definable via OAM web pages and is currently set to 5 at EMSA.
The mechanism used by OAM for storing the password is via use of an LDAP attribute (*obpasswordhistory*) that contains an encrypted list of recently used passwords.

REQ_41	**MANDATORY**

The proposed solution should have a failed password counter inhibiting access after N consecutive failed attempts (N being configurable), aka "locked-out". Having a failed password counter is a security measure to foil attacks to "guess" a password associated to a known account.

4.41 Failed password counter

The current solution used at EMSA permits the definition of a counter for failed password attempts (value currently set to 5 via OAM web pages). This means that a user may fail his password 5 times before his account is temporarily locked. OAM stores the information related to this mechanism in the OpenLDAP in various attributes:

- *oblogintrycount* when existing, contains the actual number of failed attempts. For obvious reasons, at EMSA the maximum number possible will be 5
- *oblastfailedlogin* contains an epoch counter timestamp (example 1563446199) indicating the moment of the last failed attempt to login
- *oblastloginattemptdate* contains a human readable timestamp (example 2019-07-18T11:36:44Z) of the last login attempt, whether successful or failed
- *oblockoutime* contains an epoch counter timestamp (example 1563457004) used for impeding access to EMSA's systems (lockout period). The value configured at EMSA for the lockout period is 2 hours after the 5th failed attempt

REQ_42	**MANDATORY**

The proposed solution should have an automatic mechanism for allowing a user to attempt logging in after having waited a specific period of time after being locked-out (the time period being configurable). It is quite normal that a user forgets his password, normally after long periods of non-use of MarApps, resulting in the user having to make multiple attempts to "guess" what his password was (see REQ_41). It is also "normal" that a mal-intended hacker tries to guess the password of an account_id found by him. To deter the latter and permit the former to continue

guessing his own password, this automatic mechanism is a useful option. If there are other possibilities, such as REQ_43 and REQ_44 described below, then this option becomes somewhat redundant.

4.42 Lockout period timer

Continuing the reasoning from the previous requirement, after passage of the lockout period (the 2 hours mentioned), an account will be available again for attempting to login with a correct password. However, there will only be 1 attempt allowed before another 2-hour lockout period is set.

Once/whenever the correct password is introduced, the counter is reset, and the lockout period is cleared. It can also be manually cleared either by clearing the values directly in LDAP or via custom code developed by EMSA to done via management console.

REQ_43	**MANDATORY**

The proposed solution should have a mechanism allowing a privileged admin user to make password changes on behalf of other accounts (i.e. "Reset a password"). As previously stated, (see REQ_41), it is typical that a user forgets his own password. It is often much easier for a user that has forgotten his password to request it be "reset" by an admin than for him to remember exactly what the password was.

4.43 Privileged account allowed to reset a password

As the account locking mechanism is all implemented via LDAP attributes, EMSA has developed bespoke code to clear the relevant attributes used for locking an account. The same code allows setting a new password manually or generating a new password to be provided to the user. Typically, this code was only available to the EMSA MSS (Maritime Support Services) but is now deprecated and no longer used. It should also be mentioned that the "reset" password was only valid for one entry due to the use of another LDAP attribute named *obpasswordchangeflag* that could contain the values *true* or *false*. The inexistence of the attribute is equivalent to the value *false*. A value of *true* means that the user can login but mandatorily has to change his password being redirected to an HTML page for that purpose.

REQ_44	**MANDATORY**

Passwords generated by "third parties" other than the actual account owner, should always result in being a one-time password. As a direct consequence to REQ_43, in which an admin resets a user's password, the generated password should be valid only long enough for the account owner to set a new, "permanent" password.

4.44 One-time passwords

As mentioned in the previous requirement, within EMSA's implementation of access management, there is an LDAP attribute named *obpasswordchangeflag* that can be used to force a password to be changed upon next access. This effectively allows the creation of one-time passwords.

REQ_45	MANDATORY

Passwords generated by "third parties" other than the actual account owner, should not count in the password history. As a direct consequence to REQ_43, in which an admin resets a user's password, the generated password should be valid only long enough for the account owner to set a new, "permanent" password (subject to other requirements involving time constraints on passwords).

4.45 One-time passwords not part of password history

As EMSA allows an admin user to reset a password for any given account, the new value set for the password should not be part of the password history (or be in any way influenced by that mechanism) as it would allow the admin to attempt to enumerate previous passwords defined by the actual account owner.

REQ_46	MANDATORY

The proposed solution should have a mechanism that allows a user to set a new password for his account at any time as long as he is authenticated (i.e. "Change your own password"). This is a standard feature of any access management system, the ability for a user to change his own password. This change is subject to the password history limitations (REQ_40).

4.46 Change your own password

The current version of OAM in use at EMSA has a web page (customizable) that allows a user to change his own password. The page uses OAM internal functionality to execute the action of updating a password. Some of the consequences are: password added to history list; failed password counter is reset; lockout timer is reset.

REQ_47	MANDATORY

The proposed solution should have a mechanism that allows a user to set a new password for his account in a secure fashion even not knowing the previous value of the password (i.e. "Lost Password"). As an alternative to REQ_43 (the admin resetting a user's password), the actual user should be able to set a new password for himself in a secure way.

4.47 Lost Password

The default implementation of OAM used at EMSA supported the concept of "lost password" through the use of web page asking that some personal questions (first school frequented, mother's maiden name, etc) be answered by the user. The questions asked, and the values for these questions was set by the user on his very first login attempt, and the answers were kept encrypted in an LDAP attribute. Time proved this solution to not be very useful because it was discovered that the users not only forgot their password, but they also forgot the answers they had provided to the personal questions and as such were not able to recover their password. As a consequence, this scheme was abandoned.

A bespoke solution was implemented in which a request would be registered, an email with a link sent out and the user would finally set a new value for their password (all of this is explained in great detail in the IdM Guide document). This solution has worked efficiently for a long time and

has resisted various security audits, etc. As such, during the project phase, EMSA will evaluate if the bespoke solution is to be kept or if the out-of-the-box solution will be applied.

REQ_48	**MANDATORY**

The proposed solution should have an auditing mechanism for password changes. Every change performed on a user's password should be logged thus allowing auditing of the said change.

4.48 Password change audit

EMSA's current solution audits password changes via a database table that gives support to the bespoke "lost password" mechanism, registering who changed what, when they did the change and if they were successful or not. No actual passwords are ever stored.
It is also possible, but to a lesser degree, infer password changes via the OBLIX_AUDIT_EVENTS database table.

5 Non-Functional Requirements

REQ_49	**MANDATORY**

The proposed solution should be customizable to suit EMSA's look-and-feel. EMSA's existing infrastructure, namely the Liferay Portal, has a look-and-feel that is well established with the end users. This means that the proposed solution should integrate cleanly and re-use all existing look-and-feel code and components.

5.49 Look-and-feel

EMSA's Access Management system is a COTS system and as such complying very much to the way it was designed and developed by the software creator (Oracle in EMSA's current case). It is very important that any solution adopted by EMSA be as integrated as possible with the rest of EMSA's software infrastructure. An important part of the integration is the look-and-feel of he various components that are part of the solution adopted.
The current solution is customizable either by creating completely new, custom pages which are used instead of the internal default pages, or, by changing the XML based pages of parts of the actual product. Examples of each case are: Login screen is completely custom while the Change Password is done via XML and HTML manipulation of the original template. The latter is done as such because there is no other way to do the customizations.

REQ_50	**MANDATORY**

The proposed solution should allow custom extensions. The proposed solution should allow EMSA to extend functionality, namely to inject JS script for traffic analysis via AppDynamics.

5.50 Custom extensions

Under specific circumstances, EMSA needs to incorporate into the access management pages custom code (typically JavaScript code) to do some very specific task. One such example is the use of AppDynamics to track a transaction all the way from the user interface to the database.

REQ_51	MANDATORY

The proposed solution should allow integration with existing access management products in EMSA's infrastructure. EMSA has deployed an APIGateWay, developed by Axway, for securing web services. The referred product integrates directly with the existing Access Management solution using the configurations made herein for securing the web services. The proposed solution should be able to leverage the existing symbiosis of the two systems.

5.51 Integration with other products

It has already been mentioned in various requirements thus presented that the current Access Management solution used at EMSA only allows the use of HTML over HTTP. This is true in both what is "controllable" as well as what is returned to the "client" (browser). Currently some of EMSA's exposed web services are being monitored by third-party applications (such as Axway APIGateWay) which integrate cleanly into OAM using the definition of Access Policies and Policy Domains defined directly in OAM.
The proposed solution should either implement all of the functionality currently provided by the third-party systems or allow seamless integration of such products so that EMSA can maintain its current level of service in terms of security coverage.

REQ_52	MANDATORY

The proposed solution should allow integration with existing identity management products in EMSA's infrastructure. EMSA has deployed an Identity Management system based on Oracle products. The referred product integrates directly with the existing Access Management solution using the configurations made herein.

5.52 Integration with Identity Management

EMSA's current Identity and Access Management solution is composed of two important components: Identity Management and Access Management.
Identity Management has been upgraded relatively recently and is not expected to change sometime soon.
Access Management is needing an urgent upgrade to be up-to-date with the current security standards.
Identity Management can be treated as a completely separate issue as long as the integration between the two is implemented (basically in terms of Single Sign-On and logging out).

REQ_53	MANDATORY

Provide protection of independent domain specific access points. EMSA is exposed to the user community via several different domains (sub-domains to be more exact). These sub-domains have a direct correspondence with the MarApps associated to them. Specific examples are eulritdc, lct, csndc and portal. Under the portal sub-domain various MarApps lay hidden (i.e. Thetis, STCW, IMDatE, etc.). As is currently possible, these sub-domains should continue to be run-time independent of one-another in the first level of protection provided by the proposed solution (please note that the first level is a reverse proxy used for "switching" traffic to the correct back-end servers).

5.53 Domain Specific Access Points

EMSA uses a concept that it has labelled as SAP (Specific Access Point). Basically, this means that there is a separate instance of an Apache Web Server (running the Oracle WebGate module) for every sub-domain EMSA intends to protect. Each instance provides reverse proxy capabilities as well as allows independent enabling/disabling of modules, etc.
EMSA would like that the new solution provides logical (and if possible physical) separation of control over sub-domains.

REQ_54	**OPTIONAL**

The proposed solution should offer Federation protection allowing multiple instances to interact in a unified fashion. EMSA has various environments that need access management protection, namely: TEST, PRE-PRODUCTION, TRAINING and PRODUCTION. The current implementation does not allow cross environment accesses limiting the scope of usability of some MarApps. If possible, a unique account accessing multiple environments would be a benefit for EMSA.

5.54 Federation

Federation is a concept that depends on who is defining the term. For some Access Management solution providers, federation means that multiple domains and/or sub-domains can be protected and managed under the same SSO solution. For other providers it means that the coverage is domain independent as long as the underlying infrastructure is unique (for example, only a single point of LDAP – could be clustered though).
EMSA currently does not have Federation services (as defined by Oracle) due to the fact that it does license and deploy Oracles Federation Server infrastructure.

REQ_55	**MANDATORY**

The proposed solution should offer run-time compatibility with the existing solution. The purpose of this requirement is to have both systems, the new and the old, running in parallel without interfering with each other whilst sharing a common infrastructure (for example, the LDAP servers). This would allow a transparent migration of protections from one system to the other without interfering in MarApp versioning and deployment.

5.55 Runtime compatibility

EMSA's Access Management solution is a very important and crucial component in what regards the use of EMSA services, both via a web user interface or directly by system services. Due to the fact that it has been kept correctly running for quite a long period of time without any major (or even minor) incidents, most of the "community" using the services are completely agnostic to its existence.
For EMSA to be able to keep the same level of transparency in what regards access management, any new solution should allow a gradual integration so as not to disrupt any existing services. EMSA SLA's are very tight due to real-world usage so any disruption may cause serious issues.

REQ_56	**MANDATORY**

The proposed solution should comply to the "common, standard," set of non-functional requirements such as High Availability, Resilience, Fault Tolerance and Performance. EMSA provides services that have a high impact on both human as well as environmental well-being. Two examples are SAR – Search and Rescue and Marine Pollution. Both of these services depend heavily on the availability and correct execution of EMSA systems. As such, the current Access Management non-functional characteristics shall be kept or improved

5.56 Non-functional Requirements

This requirement is obvious and self-explanatory, so no further comments are deemed needed. However, due to the criticality of the Access Management solution it is important to refer mandatory requirements such as (but not limited to), availability, resilience, reliability, performance, security, scalability and maintainability are crucial and shall be considered in the design and architecture. Decreasing any of the non-functional characteristics of the Access Management system cannot be accepted

6 Current OAM Access Policies and Specific Access Points

REQ_57 **MANDATORY**

The proposed solution should comply, support and migrate the current OAM Access Policies (AP) and Specific Access Points (SAP) identified in the list below[1]*:*

[1] List refers to current PRODUCTION configurations

Identification	AP	SAP
API WebServices	X	
CHD/Marcis2	X	
CMC	X	
CSN/Orchestra	X	
EMCIP	X	
EOS	X	
IMDatE	X	
IMS Mobile	X	
IRD	X	
Jasper	X	
LCT		X
Liferay	X	
Portal		X
Portal2		X
LMS	X	
LRIT Ship Db		X
LRIT-DC	X	X
Marcis2 Mobile	X	
RuleCheck	X	
RuleCheck Mobile	X	
SEG	X	X
Regular	X	
HPSEG (High Performance SEG)		X
HPSEGIC (In-the-Cloud SEG)		X
SSN	X	
STCW	X	
Thetis	X	
Thetis Med	X	
Identity domain	X	
OAM (login, logout, reverse proxy)	X	
OIM	X	
Other miscellaneous	X	

6.57 Migration of the current Access Policies and Service Access Points

This requirement establishes the list of OAM Access Policies and Specific Access Points that are in the scope of the services to be provided.

As defined in REQ_55, migration of these elements might be done in a gradual approach to be defined during project lifecycle.

In addition, a Project Acceptance criterion shall be defined based on the list defined above.