Table of Contents

Straw-man design

This is a straw-man design for how to support SRM over SSL, part of the SRMSEC activity. It comes partly from the meeting between EMI Security and Data during the EGI meeting in Amsterdam. It also has some arbitrary decisions by Paul Millar.

The document is written to get discussion started. It's not necessarily right.

Initial observations

First, one of the major differences between SSL and GSI is that GSI allows for the server to ask for a delegated credential. The delegation step happens immediately after establishing the security context and before the SOAP message describing an SRM command has been sent. Therefore the server has no knowledge if the client will make a request that will require a delegated credential and must always request a delegated credential. This places unnecessary load on the server.

Second, SRM implementations may have a different set of commands that require delegation; for example, one SRM instance may require a delegated credential to conduct an `srmBringOnline` while another SRM implementation may no need this. Therefore, in general, only the server can decide whether delegation is needed for any particular command.

Third, SSL does not support delegation so some kind of delegation service will be required.

Available delegation services

There are several available delegation protocols available. Details of them are available in this page.

SRM over SSL Protocol description

This section gives examples of client/server interaction. It is not a formal definition of the protocol, but attempts to illustrate the main ideas by describing what happens in four different scenarios.

The first three scenarios, `CMD_WITHOUT_DELEGATION`, `CMD_NEEDS_DELEGATION` and `CMD_ALREADY_DELEGATED`, are the minimum to support SRM over SSL. The final scenario, `DELEGATED_CERT_EXPIRES`, describes an optional behaviour that may improve throughput without making the client more complicated.

CMD_WITHOUT_DELEGATION

In this scenario, a client makes an SRM request of the server that does not require a delegated credential. The behaviour is similar to current behaviour with GSI-based SRM.

1. If not already established, the client makes a connection to the SRM server with the usual TCP and SSL handshakes.
2. the client sends the request to the server (SOAP/HTTP binding).
3. the server sends the reply (SOAP/HTTP binding). The reply is the same as for GSI-based SRM: it contains the result of the command, or the token to allow the client to poll for the result.

CMD_NEEDS_DELEGATION

In this scenario, a client makes an SRM request of the server and the server requires a delegated credential to complete the command.

1. If not already established, the client makes a connection to the SRM server with the usual TCP and SSL handshakes.
2. the client sends the request to the server (SOAP/HTTP binding).
3. the server sends the SOAP-encoded reply message. The reply message indicates that delegation is required and the location of a suitable delegation service.
4. the client creates a delegated certificate for the server's use, using the delegation service.
5. the client attempts the command again, starting from Step 1. above

Notes:

- the delegation service need have no knowledge of the SRM service or the requested operation, although such information may be passed.
- by delegating the credential, the scenario changes to `CMD_ALREADY_DELEGATED` (see below)
- the phrase "the reply message indicates that delegation is required..." is deliberately vague; see below for details.

`CMD_ALREADY_DELEGATED`

In this scenario, a client makes an SRM request of the server that requires a delegated certificate. The server has obtained, by previous delegation activity, a delegated certificate from the user. Therefore, the request can proceed without requiring delegation.

1. If not already established, the client makes a connection to the SRM server with the usual TCP and SSL handshakes.
2. the client sends the request to the server (SOAP/HTTP binding).
3. the server sends the reply (SOAP/HTTP binding). The reply is the same as for GSI-based SRM: it contains the result of the command, or the token to allow the client to poll for the result.

Notes:

- the procedure is the same as `CMD_WITHOUT_DELEGATION`
- It is optional whether or not the server binds a delegated credential to an operation. Both are supported.

`DELEGATED_CERT_EXPIRES`

In this scenario, the client has made a request from the server that is both long-lived and requires a delegated certificate. `srmCopy` is an example of a command that might be both long-living and require a delegated certificate.

In this scenario, we assume server is using the polling method; i.e., it returns a token that the client uses to obtain updated information about the request periodically. These polling requests have the form `srmStatusOf*Request`; for example, `srmStatusOfCopyRequest`.

In order to accept the command, the server must have a delegated certificate. This follows `CMD_NEEDS_DELEGATION` and `CMD_ALREADY_DELEGATED`. However, in this scenario, the delegated certificate has expired before the operation completed and while the delegated certificate is still needed; for example, the `srmCopy` command is in a queue and, while queued, the delegated certificate expires.

A valid option in this scenario is for the server to fail the command. The client would then be required to resubmit the request. This is suboptimal for several reasons:

- the command takes longer to complete,
- resources may have been consumed to process the command so-far; these resources will be wasted,

- the problem may simply repeat: the client and server may not know in advance how long the delegated certificate will be needed.

An alternative approach is to allow the server to request a fresh delegated certificate and for the comment to continue.

1. If not already established, the client makes a connection to the SRM server with the usual TCP and SSL handshakes.
2. the client sends the appropriate srmStatusOf*Request request to the server (SOAP/HTTP binding), for example `srmStatusOfCopyRequest`
3. the server sends the reply (SOAP/HTTP binding). The reply message indicates that delegation is required and the location of a suitable delegation service.
4. the client creates a delegated certificate for the server's use, using the delegation service.
5. the client attempts the command again, starting from Step 1. above

Notes:

- the behaviour is similar to `CMD_NEEDS_DELEGATION`.
- this behaviour is optional in the server. The client need only respond appropriately when it receives the need-to-delegate reply.

Triggering delegation

The above scenarios use the vague description that a reply message indicates delegation is required. This (deliberately) fails to specifying how this is achieved. This section describes how the server indicates to the client that delegation is required.

It is desirable that SRM be supported over SSL without altering the WSDL. To do this, the server must indicate that delegation is required and where a suitable delegation service is located. The field that is common to all SRM replies is the `ReturnStatus`. The `ReturnStatus` has two components: an enumerated type TStatusCode and an unstructured string "explanation".

The proposal is that, when using the SSL transport and the SRM server wishes the client to delegate a credential, the server returns a `ReturnStatus` with the `TStatusCode` field taking the value `SRM_AUTHORIZATION_FAILURE`. The explanation string contains two or more lines of ASCII (7-bit clean, plane-0 of UTF-8) text; each line is terminated by a new-line character ('\n'). The first line is exactly

```
# Needs delegation.
```

The remainder of the explanation string contains information about one (or more) delegation services that the client should use. This information is a set of tuples. Each tuple has two pieces of information: a type keyword (describing the kind of delegation service) and a URI (the service's end-point).

The data is marshalled using YAML, as a list with each list-item is an associated array. The order in which delegation services are marshalled into the list is unimportant. The keywords for the associate array are `type` (for the delegation service type) and `uri` (for the service end-point).

The value for `type` must not contain a comma-space sequence and is recommended that it doesn't contains any spaces.

The URI value is always percent-encoded, as described in RFC 3986. The URI may contain a query-part. This allows the SRM-server to pass request-specific information to the delegation service (see below).

A complete example of a valid explanation string is:

```
# Needs delegation.
- {type: foo, uri: https://delegation1.example.org:8443/services/delegation}
- {type: foo, uri: https://delegation2.example.org:8443/delegation}
```

Note:

- the complete explanation string is a valid YAML document.
- the server should send a needs-delegation reply only with SSL-based transports, never when using the GSI transport.

Client behaviour

When a client receives an SRM reply with a `TStatusCode` value of `SRM_AUTHORIZATION_FAILURE` and explanation that starts `# Needs delegation.`, the client should parse the explanation as a YAML document to obtain the list of delegation services.

Having obtained a list of delegation services and that service's type the client should attempt to create a delegated credential that the server may use. The strategy the client employs to achieve delegation is not specified; however, here are some recommendations:

- the list of delegation services may include service types the client doesn't support; unsupported services should be filtered out.
- the client's choice of which delegation service to choose may be influenced by several factors, included past behaviour of the services.
- the client should try all supported delegation services before considering delegation to have failed.

If the delegation is successful, the client should retry the command. If the command then fails with the same ReturnStatus the client should fail the command, indicating that the server did not accept the delegated credential.

If the delegation is unsuccessful, the client should fail the command, indicating that it was unable to create a delegated credential.

It is anticipated that all SRM implementations have one common delegation service. The `type` field of the list of delegation services is included to allow for possible migration to alternative delegation services in the future.

Binding delegated credential to the request

An SRM server may choose to bind a delegated credential to a specific request; that is, a server may choose to require a delegated credential for each delegation-requiring operation that a user requests rather than reusing an existing (and still valid) certificate from a previous command. The alternative is to reuse an existing certificate, if it is still valid.

Note that the decision whether to bind delegated credentials to a request is given to each SRM implementation. The scheme described here allows either approach; for example, to bind a delegated credential to a request, the delegation URI(s) may include the request ID.

```
# Needs delegation.
- {type: foo, uri: https://delegation1.example.org:8443/services/delegation?req=5434}
- {type: foo, uri: https://delegation2.example.org:8443/delegation?req=5434}
```

-- PaulMillar - 04-Oct-2010

This topic: EMI > EmiJra1T3-SRMSEC-Strawman
Topic revision: r3 - 2010-11-17 - PaulMillar