NORDUGRID

NORDUGRID-TECH-17
15/11/08

MANUAL OF SECURITY FRAMEWORK

W.Qiang, A.Konstantinov

Abstract

This document is about user manual of security framework, as well as development manual and administration manual.

Table of Contents

1. User Manual

This section describes how to configure and use SecHandler and PDP elements included in the ARC1 and provides few examples of the ARC Policy documents. The target readers are those users who will use the ARC1 middleware. Currently this section is very short on details. It is going to be continuously extended. Especially taking user feedback into account.

1.1. Authorization SecHandler and PDPs

There is a specific Authorization SecHandler (arc.authz) which is implemented for calling the Policy Decision Points (PDP) and serves as their container.

Usually the Authorization SecHandler and included PDPs are used on the service side of communication channel. Although it is also possible to use them on the client side. All possibilities are achieved by modifying the configuration file (hereafter mentioned as service.xml) and possibly providing the authorization policy in a separate file.

Here the “echo” test service is used to explaining the usage, but the explanation applies to other services as well.

The procedure for configuring Authorization SecHandler in service.xml is following:

1. Add the Authorization SecHandler as child element <SecHandler/> of <Service/> element.

   The “name” and “event” attribute of <SecHandler/> element are both important. The “name” attribute is used for distinguishing betwenn loaded SecHandler objects. The “event” attribute defines for which message authorization would be enforced. Usually and reasonably it is done for “incoming” messages. But some services and other Message Chain components may define other internal types of messages. For possible values please refer to documentation of particular Service or MCC. In our particular case “echo” service only supports “incoming” messages for this purpose.

2. Add the PDP configuration as child element <PDP/> under <SecHandler/>. Currently there are four usable PDPs distributed as part of the ARC1 middleware:

   - simplelist.pdp – compares Subject of user's X509 certificate to those stored in a file.
   - arc.pdp – compares authorization related information parsed from message at various processing steps to Policy document specified in configuration of this PDP.
   - pdpservice.invoker - composes the ARC Request, puts request into SOAP message, and invokes the remote PDP service to get the response SOAP which includes authorization decision. The PDP service functionality is similar to arc.pdp.
   - delegation.pdp – compares authorization related information parsed from message at various processing steps and Policy document embedded in proxy certificate used by remote side.

Default behavior of Authorization SecHandler is to execute all configured PDPs sequentially till eithe r one of them fails or all produced positive results. This behavior may be modified by attribute “action” of <PDP/> element.

The description of PDP configuration and ARC Policy example are available in Section

1.2. Delegation SecHandler, Delegation PDP and Proxy Certificate Generation

Delegation SecHandler and Delegation PDP in their current state provide an infrastructure for limiting capabilities of delegated credentials. Their collect and process policies attached to X509 Proxy Certificates respectively. Hence to have delegation restriction working both must be enabled in configuration of service. Configuration of Delegation SecHandler is described in section .

The possible location for Delegation PDP is inside Authorization SecHandler (arc.authz). Depending

on how fine grained policy of delegated credentials is supposed to be corresponding Authrorization SecHandler may be attached to different MCCs or directly to Service component. However, the precondition for using Delegation PDP is that there must be Delegation SecHandler instantiated earlier in chain.

On the client side, command line utility “approxy” utility can be used to generate Proxy Certificate with Delegation Policy embedded.

Normally approxy appears in $ARC_LOCATION/bin. The usage of approxy is like:

approxy -P proxy.pem -C cert.pem -K key.pem -c constraints

By using argument "-c", some constraints can be specified for proxy certificate. Currently, the life time can be limited by using "-c validityStart=..." and "-c validityEnd=...", "-c validityStart=..." and "-c validityPeriod=...". Like for example

-c validityStart=2008-05-29T10:20:30Z

-c validityEnd=2008-06-29T10:20:30Z

The Delegation Policy can be specified by using "-c proxyPolicyFile=..." or "-c proxyPolicy=...". Like

-c proxyPolicyFile=delegation_policy.xml

The Delegation Policy is the same as the ARC Policy explained in section . Simple example below renders delegated credentials usable only for contacting service attached to HTTP communication channel under path /arex (line 6) and allows HTTP operation POST (line 9) on it.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <Policy xmlns="http://www.nordugrid.org/schemas/policy-arc"
3.          PolicyId="sm-example:policy1" CombiningAlg="Deny-Overrides">
4.     <Rule RuleId="rule1" Effect="Permit">
5.        <Resources>
6.           <Resource Type="string" AttributeId="http://www.nordugrid.org/schemas/
    policy-arc/types/http/path">/arex</Resource>
7.        </Resources>
8.        <Actions>
9.                                                    <Action      Type="string"
    AttributeId="http://www.nordugrid.org/schemas/policy-
    arc/types/http/method">POST</Action>
10.       </Actions>
11.    </Rule>
12. </Policy>
```

Another example of delegation policy is presented below. This policy restricts usage of delegated credentials to SOAP operation CreateActivity (line 5) of Basic Execution Service (BES) [4] namespace (line 9). Such policy could be embedded into credentials delegated to high level Brokering service performing Grid job submission to low level BES on behalf of user.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <Policy    xmlns="http://www.nordugrid.org/schemas/policy-arc"    PolicyId="sm-
    example:policy1" CombiningAlg="Deny-Overrides">
3.     <Rule RuleId="rule1" Effect="Permit">
4.        <Actions>
5.                                                    <Action      Type="string"
    AttributeId="http://www.nordugrid.org/schemas/policy-
    arc/types/soap/operation">CreateActivity</Action>
6.        </Actions>
7.        <Conditions>
8.          <Condition>
```

```
9.                                                    <Attribute     Type="string"
    AttributeId="http://www.nordugrid.org/schemas/policy-
    arc/types/soap/namespace">http://schemas.ggf.org/bes/2006/08/bes-
    factory</Attribute>
10.         </Condition>
11.       </Conditions>
12.    </Rule>
13. </Policy>
```

1.3. VOMS Proxy Certificate

The commonly used voms proxy certificate can be used in ARC1 for authentication as normal proxy certificate, and attribute acquiring in order to make authorization decision based on attributes.

1.How to create voms proxy certificate

Currently the proxy creation utility in ARC1—arcproxy can not be used for creating voms proxy certificate because the GSI-based communication requirement from voms server side has not been supported in ARC1 (ARC1 uses standard SSL/TLS communication). So the way to create voms proxy certificate is still to use the “voms-proxy-init” utility.

2.How to use voms proxy certificate in ARC1

The attribute certificate (AC, which is created by voms server and then embedded in voms proxy certificate as one of the certificate extension by “voms-proxy-init” client) will be parsed by TLS handling plugin (called MCCTLS) of ARC1, and saved in the message context in the format “grantor=knowarc://testvoms.knowarc.eu:50000/knowarc:role=guest”.

Service administrator needs to configure the trusted DN chain for verifying the AC under tls MCC's configurationwith the following format. You should specify one node (<tls:VOMSCertTrustDNChain/>) for each vo which you trust. The reason why using trusted DN chain is for service to restrict which vo server is trusted.

In the following table, the first <tls:VOMSCertTrustDNChain> in the following table defines two items: the first one is the DN of voms server, the second line is the DN of the corresponding CA. The second <tls:VOMSCertTrustDNChain> defines regular expression for matching, the DN of voms server and DN of the corresponding CA should both match it. The third one defines the external file which includes the <tls:VOMSCertTrustDNChain>.

```
<tls:VOMSCertTrustDNChain>

 <tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/OU=fys.uio.no/CN=ABCDE</tls:VOMSCertTrustDN>

 <tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority</tls:VOMSCertTrustDN>

</tls:VOMSCertTrustDNChain>

<tls:VOMSCertTrustDNChain>

 <tls:VOMSCertTrustRegex>^/O=Grid/O=NorduGrid</tls:VOMSCertTrustRegex>

</tls:VOMSCertTrustDNChain>

<tls:VOMSCertTrustDNChainsLocation>./pathto/external/file</tls:VOMSCertTrustDNChainsLocation>
```

Service administrator can then use the attribute into access control policy:

```
1.        <Subjects>
2.           <Subject>
3.              <Attribute AttributeId="http://www.nordugrid.org/schemas/policy-arc/
    types/tls/vomsattribute"
    Type="string">grantor=knowarc://testvoms.knowarc.eu:50000/knowarc:role=guest</A
    ttribute>
4.                 <Attribute  AttributeId="http://www.nordugrid.org/schemas/policy-
    arc/types/tls/ca" Type="string">/C=NO/ST=Oslo/O=UiO/CN=CA</Attribute>
5.                  <Attribute AttributeId="http://www.nordugrid.org/schemas/policy-
    arc/types/tls/identity" Type="string">/C=NO/ST=Oslo/O=UiO/CN=test</Attribute>
6.           </Subject>
7.        </Subjects>
```

1.4. UsernameToken SecHandler

The UsernameToken SecHandler is meant for processing - generating and extracting - WS-Security [5] UsernameToken from SOAP header. Hence it must be attached to SOAP MCC of service or/and client communication channel.

On the service side, the functionality of extracting UsernameToken may be configured as described in section .

On the client side, the UsernameToken SecHandler may be configured either by using client specific methods (for example see test_clientinterface.cpp src/tests/echo directory of source tree) or through generic client configuration file as shown in example below (also can see share/doc/arc/echo_client.xml.example under the installation directory). This example will generate token with username “user” and password “pass” inside any SOAP message sent by client tools of the ARC1.

```
<ArcConfig>
 <Plugins overlay="add"><Name>arcpdc</Name></Plugins>
 <Chain>
  <Component name="soap.client">
   <SecHandler name="usernametoken.handler" event="outgoing" overlay="add">
    <Process>generate</Process>
    <Username>user</Username>
    <Password>pass</Password>
    <PasswordEncoding>digest</PasswordEncoding>
   </SecHandler>
  </Component>
 </Chain>
</ArcConfig>
```

1.5. X509Token SecHandler

The X509Token SecHandler is meant for processing - generating and extracting - WS-Security [5] X509Token from SOAP header. Hence it must be attached to SOAP MCC of service or/and client communication channel.

On the service side, the functionality of extracting X509Token may be configured as described in section 7.12.

On the client side, the X509Token SecHandler may be configured either by using client specific methods (for example see test_clientinterface.cpp src/tests/echo directory of source tree) or through generic client configuration file as shown in example below. This example will generate x509 token (by using the specified certificate and key file) inside any SOAP message sent by client tools of the ARC1.

```
<ArcConfig>
 <Plugins overlay="add"><Name>arcpdc</Name></Plugins>
 <Chain>
  <Component name="soap.client">
   <SecHandler name="x509token.handler" event="outgoing" overlay="add">
    <Process>generate</Process>
    <KeyPath>/path/to/key.pem</KeyPath>
    <CertificatePath>/path/to/cert.pem</CertificatePath>
   </SecHandler>
  </Component>
 </Chain></ArcConfig>
```

References

1. RFC3820- Internet X.509 Public Key Infrastracture (PKI) Proxy, http://rfc.net/rfc3820.html

2. D1.2-2 The ARC container (first prototype), http://www.knowarc.eu/documents/Knowarc_D1.2-2_07.pdf

3.Web Service Security Username Token Profile 1.1. http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf

4. OGSA® Basic Execution Service Version 1.0, http://www.ogf.org/documents/GFD.108.pdf

5. OASIS Web Services Security (WSS) TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss