EUROPEAN MIDDLEWARE INITIATIVE

Common Authentication Library

Krzysztof Benedyczak (UWAR)

Plan

- Short report: what happened since AHM, what is the current status.
- Discussion on remaining issues. Mostly C++.
- Feature-wise comparison of the APIs. Do we have to make them eliminate some of the differences?
- Next steps.

Since AHM

- We have decided on API documentation format.
  - SA2 stepped back,
  - Internally with some support from other bodies we decided on:
    - JavaDocs for Java
    - Doxyden for C & C++

Since AHM: Java API

- Base part of Java API was finalized.
  - Core API was only updated with all the remarks from the AHM (minor updates).
  - Certificate and PEM utilities were added.
    - Long discussion on DNs. Summary:
      - RFC2253 format is the only one supported(!!)
      - DN comparison is done either
        - using RFC 3820 algorithm when both ASN representations are given or
        - using the JDK X500Principial.equals() method when at least one argument is provided as a string.
- API documentation was written.

Since AHM: Java API

- Proxy part was developed.
  - Heavily based on Proxy API from util-java.
  - Changes:
    - cleaned up from classes being used internally,
    - updated to JDK 5,
    - split some big classes to have better separation of concerns.
    - added a new code for helping in generation of proxy CSRs.
- Full API is available here:

https://twiki.cern.ch/twiki/pub/EMI/EmiJra1T4CaNlAPIJava/caNl-javadoc-rc1.zip

What is missing?

- No real input from dCache :-(
  - I've decided not to wait any more.
- Small issue: what authors shall be put in documentation?
  - Currently generated docs don't have authors defined
  - Available options:
    - no author
    - real author who wrote the API (e.g. Joni & myself in case of Java)
    - "EMI common authN task force" or something like this.

C API

- Base part complete.
- Proxy part missing.
- Documentation missing.
- Other questions - later.

C++ API

- Aleksandr submitted a revised version.
- Open questions:
  - Error reporting. Possible options are exceptions and status values.
  - Memory handling. Possible choices:
    - Common approach is to let programmer do memory handling and clearly define in documentation usage of object instances.
    - Everything can be passed by value.
    - Java approach can be adopted and exposed classes would be just proxies with actual content hidden and protected from accidental destruction.

C++ API

- OpenSSL can be completely hidden by providing virtual Read and Write methods in IO class instead of attaching it to BIO.
  - In case of Credential class it would have o be extended with methods for manipulating/accessing attributes of credentials. That would probably make it possible to integrate proxy management here.
- The current proxy API in ARC https://svn.nordugrid.org/repos/nordugrid/arc1/trunk/src/hed/libs/credential/Credential.h
  - can provide some feedback on what functionality is needed.
  - Needs a clean-up.

C++ API

- Thread safe: Yes or No?
- No way to pass owned credentials in other way then providing a file path, is it OK?
  - What format? Other then PEM pair are foreseen (e.g. pkcs12 keystore)?

The three APIs: functional comparison

- Java API allows for many Validators. The rest of implementations assume one with configurable verification level.
  - TODO: what do those levels means? Do we need a more fine-grained settings (like set of flags)? Examples:
    - support proxies or not; require CRL; use CRL if present only; same with OCSP.
- Java API doesn't provide IO methods as C/C++ - it integrates with JSSE instead.
- Java API provides possibility to use different owned credentials (JKS, PKCS12, PEM files). C/C++ APIs rather not.
- Java API doesn't provide callback for getting a password.
- Java API provides set of utility methods missing in C/C++:
  - DN compare, printing credentials in human-readable form, load & save.

Schedule

- End of internal review of Java API
  - 4th of March
- Finalized version of C API for internal review.
  - ?
  - Review ends: ?
- Finalized version of C++ API for internal review.
  - ?
  - Review ends: ?
- Consultations with affected PTs
  - Starts ASA internal reviews are finished.

Thank you

EMI is partially funded by the European Commission under Grant Agreement INFSO-RI-261611