Common Authentication Library

Krzysztof Benedyczak (UWAR)

Plan

- Summary of what has been done&decided.
- General/strategic issues regarding the library.
- Review of the affected components list.
- Java API
- C API
- C++ API
- Schedule on finalization of the API & consultations with affected PTs.
- What's next?

Common authN library

- Aim: provide a **commonly used** library supporting authentication and tightly coupled functions.
- Why?
  - As we have multiple libraries in our 4 stacks doing authentication.
    - reduced maintenance
    - uniform features for UMD
- Examples:
  - In gLite some components check if DN conforms to the CA namespace without case-sensitiveness. Some with.
  - In UNICORE Gateway provides more authN features then other servers.

What has been done/decided?

- Wiki page created.
  https://twiki.cern.ch/twiki/bin/view/EMI/EmiJra1T4SecurityCommonAuthNLib
- Three languages - Java, C, C++
- Requirements were collected and unified.
- Initial list of components:
  - using authentication libraries,
  - implementing authentication

  was established.
- Requirements collection step showed that there are quite different opinions on the library shape and even purpose.

Library models and purpose

- We support TLS and X.509 authentication - obvious. Other options:
  - add SAML Authentication support too or
  - arbitrary protocol support (Kerberos, LDAP, ...)
- Library models which were considered:
  - consolidated, tuned for the concrete authN mechanism,
  - pluggable with different options hidden under a common API (Joni, Andrea, Daniel)
    - e.g. GSS API

Current status

- We support TLS/X.509 only.
- **If** there is a need for a **common** SAML authentication library it will be provided as a separate entity.

Motivation

- Decision to use STS/SLCS/... eliminates need for wide adoption of SAML AuthN in all components.
- Currently SAML AuthN is not present (or nearly not present?) among EMI components.
- Authentication API is very dependent on the authentication mechanism and offered functionality. Therefore abstract API is very generic and hard to use.
  - Different integration scenarios in various containers.
  - Different credentials.
- Credentials and respective utilities are needed AFTER authN layer too.
- We don't have enough time to work for a long time on this.

Open topics

- General: is the current model widely accepted?
- Minor (those are for the **implementation**):
  - Shall we support RFC 5280?
    - How? By default or only optionally in some selected functions?
  - OCSP support?

List of existing authN code

- gLite: VOMS (C, Java)
  - Nothing more?
  - ?(KB) trustmanager, util-java, L&B, ...?
- ARC: ARC Container (C++ with Java&Python bindings)
  - Nothing more? E.g. update-crls, arcproxy?
- dCache: ????
  - No input so far!
- UNICORE - complete list on the wiki.

Review: affected components

- gLite: TFS, CREAM, WMS, VOMS Admin, Argus, SLCS
  - No other components? What's about APEL*, DGAS*, BDII*, STS, delegation-java, gridsite, proxyrenewal?
- ARC: ARC Container
  - Is everything using the container? E.g. libarcclient?
- dCache: No input so far!!
  - ???
- UNICORE: complete list on the wiki

Java API

- We want to reuse X.509 JDK code (JCA), and integrate with SSL engine (JSSE).
- It is impossible to reuse certificate validation code as implementation is hidden and it is not possible to go round proxy certificate validation (lack of CA extension)
  - Additionally the JCA API is quite awkward.
- We can quite easily integrate with JSSE.

Java API (2)

- interface X509CertChainValidator
  - Implementations are used to perform a manual certificate chain validation. Implementations shall reuse as many of CertificateChecker implementations as possible. Implementations must be thread safe.

```
ValidationResult validate(java.security.cert.CertPath cp)
ValidationResult validate(java.security.cert.X509Certificate[] cp)

java.security.cert.X509Certificate[] getTrustedIssuers()

void addValidationListener(ValidationListener l)
void removeValidationListener(ValidationListener l)
-------------------------EXAMPLE-------------------------------
OpensslCertChainValidator v = new OpensslCertChainValidator(
    "/my/certs", CRL.REQUIRE,
    OpensslCertChainValidator.NAMESPACE.IGNORE, 100);

ValidationResult result = v.validate(toBeChecked);
```

Java API: validation implementations

- Internally various X509CertChainValidator implementations should reuse as much of code as possible.
- For further extensibility (new or modified validators) we can define validator building blocks (called checkers)
  - May be stateful, implementations need not to be thread safe - one instance must not be used to check two certificate chains simultaneously.

```
interface CertificateChecker {

 List<ValidationError> check(X509Certificate[] cert, int
position,
                  Collection<String> unresolvedCritExts)
 void init()
}
```

Java API: JSSE integration

- class CanlX509TrustManager implements javax.net.ssl.X509TrustManager
  - Wraps X509CertChainValidator so it can be easily used in standard Java SSL API.
- interface Credential
  - java.security.KeyStore getAsKeyStore()
  - javax.net.ssl.KeyManager getAsKeymanager()
- class HostnameToCertificateChecker implements javax.net.ssl.HandshakeCompletedListener
  - Implementation which can be registered on a SSLSocket to verify if target hostname is matching a DN of its certificate.

Java API: JSSE integration

- class SocketFactoryCreator
  - Simple utility allowing programmers to quickly create SSLSocket factories.

```
class SocketFactoryCreator {
   static SSLServerSocketFactory createSSF(Credential c,
                    X509CertChainValidator v)
   static SSLSocketFactory createSF(Credential c,
                   X509CertChainValidator v)
}
------------------------EXAMPLE-----------------------------
KeystoreCertChainValidator v = new KeystoreCertChainValidator(
    "truststore.jks", CRL.REQUIRE, passwd, "JKS", crls, 100);

Credential c = new KeystoreCredential("/my/keystore.jks",
           ksPasswd, keyPasswd, "JKS");
SSLServerSocketFactory sslSsf =
                     SocketFactoryCreator.createSSF(c, v);
ServerSocket sslSS = sslSsf.createServerSocket();
```

Java API: Utilities

- Testing two DNs for equality.
  - What more?
- Proxy certificate utilities:
  - As a class being Credential or
  - X509Certificate?
  - Operations:
    - write to disk
    - load from disk
    - ? creation of a proxy request
  - Creation arguments?

C API

- Vinzenzo

C++ API

- Aleksandr

APIs documentation

- What exactly we shall provide?
  - Overview
  - Examples
  - Functions or methods & classes description
- What format?
- Where to put it?

Short term schedule

- Finalization of the API
  - Applying changes from AHM
    - DATE
  - Documentation
    - How?
    - DATE
  - Final internal review
    - Who which API?
    - DATE
- Consultations with affected PTs
  - End DATE

What's next?

- Common authentication library **Product Team**
  - What are the responsibilities?
    - (?) Actual implementation of the interface, support and evolution (?)
  - Who is involved?
  - Schedule:
    - RC:
    - 1.0 Final:

Thank you

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