EMI common authentication library
Java version adoption

Krzysztof Benedyczak

Basic info

- Version 1.0.0 released in EMI as OS packages
- Version 1.0.1 == 1.0.0 code-wise, but was updated to live in Maven Central repo.
  - and is available from there
- If using via Maven use the 1.0.1.
- If using deb/rpm remember that in classpath those guys are needed:
  - BouncyCastle 1.46 (exact!, not in SL5 and Deb6 repo yet)
  - Apache Commons IO >= 1.4

Documentation

- Library has an **extensive** documentation:
  - Introduction (HTML & PDF)
  - JavaDocs (HTML)
- This presentation covers only small parts of the full documentation.
- Available from (as of now):

  http://unicore.eu/documentation/manuals/unicore6/files/canl-<VERSION>

Integrating Java components

- CanI never logs by itself
  - logging/GUI/console output must be provided by application
- CanI does not enforce or directly support any concrete configuration syntax
  - classes to load an application specific-configuration and instantiate canl objects are needed.

General integration pattern (suggested;-)

- Fundamental work to integrate caNl should be done in a common middleware specific library. Examples:
  - gLite Trustmanager (tomcat/axis)
  - UNICORE securityLibrary (Jetty)
  - Such library should offer a thin layer over canl with middleware specific configuration, logging, user interaction etc..
- However **all components** which anyhow manipulate directly:
  - Certificate, public and private keys (load, save, print, validate, ...)
  - Proxies
  - Distinguished Names (parse, print, compare)

  should be also reviewed and updated.

Hints

- Scan your code for all occurrences of:
  - X509Certificate, Certificate, X500Principal, PublicKey and PrivateKey
  - and BouncyCastle counterparts if BC API was heavily used.
  - proxy handling
- Replace all DN operations with `X500NameUtils` methods, like:
  - user presentation: `getReadableForm(String)`
  - comparing: `equal(String, String)`
  - yes, caNl fixes several issues which are present when using JDK or BC directly.
- Similarly for certificates handling use `CertificateUtils`

Examples

- Create a standalone certificate chain validator, using the OpenSSL-like truststore.
- Validate a certificate chain and check result.

```
X509Certificate[] toBeChecked = null;
X509CertChainValidator vff = new OpensslCertChainValidator(
        "/etc/grid-security/certificates",
        NamespaceCheckingMode.EUGRIDPMA_AND_GLOBUS, 60000);

ValidationResult result = vff.validate(toBeChecked);
if (result.isValid()) {
        ...
} else {
        List<ValidationError> errors = result.getErrors();
        ...
}
```

Examples

```
X509Certificate someCertificate = CertificateUtils.loadCertificate(
                                                inputStream, Encoding.PEM);
X500Principal dn1 = someCertificate.getSubjectX500Principal();
String dn2 = "CN=Bob,O=Example,C=EX";
boolean equal = X500NameUtils.equal(dn1, dn2);
```

- Load certificate chain and compare its subject name with another DN in a proper way.
- *Note: It is known that caNl DN comparison, when at least one of DNs is in text format, is not perfect, i.e. you can get false positives (never false negatives). This can not be fixed, but is done in a way which shouldn't cause security issues.*

Next example...

- Shows a complete configurations of UNICORE-integrated caNl component.
- First part shows a full configuration of a truststore of directory type (probably the easiest to be used for admins)
- Te second shows an other configuration using OpenSSL-style truststore.

UNICORE conf example

```
truststore.type=directory

truststore.allowProxy=DENY
truststore.updateInterval=1234
truststore.directoryLocations=/trust/dir/*.pem
                                                http://caserver/ca.pem
truststore.directoryEncoding=PEM
truststore.directoryConnectionTimeout=100
truststore.directoryDiskCachePath=/tmp

truststore.crlLocations=/trust/dir/*.crl http://caserver/crl.pem
truststore.crlUpdateInterval=400
truststore.crlMode=REQUIRE
truststore.crlConnectionTimeout=200
truststore.crlDiskCachePath=/tmp
```

```
truststore.type=openssl
truststore.opensslPath=/truststores/openssl
truststore.opensslNsMode=EUGRIDPMA_GLOBUS_REQUIRE
truststore.allowProxy=ALLOW
truststore.updateInterval=1234
truststore.crlMode=IF_VALID
```

Jetty 6 integration

```
public class CustomSslSocketConnector extends SslSocketConnector {
  private final X509CertChainValidator validator;
  private final X509Credential credential;

...

  @Override
  protected SSLServerSocketFactory createFactory() throws Exception {
    KeyManager[] keyManagers = new KeyManager[]
                                    {credential.getKeyManager()};
    X509TrustManager trustManager =
                     SocketFactoryCreator.getSSLTrustManager(validator);
    TrustManager[] trustManagers = new X509TrustManager[]
                     {trustManager};

    SecureRandom secureRandom = SecureRandom.getInstance(secRandomAlg);
    SSLContext context = SSLContext.getInstance(protocol);

    context.init(keyManagers, trustManagers, secureRandom);
    return context.getServerSocketFactory();
}
```

Jetty (cont.)

- Jetty 8 should be similar (other class needs to be extended but still SSLContext need to be created and returned).
- Credential can be easily loaded using caNl API (in appropriate format: PKCS12, pem pair, ...)