1. Security

Two parts: Authentication and confidentiality.

Authentication:

- The service identity must be guaranteed
- The user identity must be guaranteed

Privacy:

- Secure communication between services
- Encryption of data

1.1. Authentication of services

With standard X509 certificates, and standard Jini / SSL methods. Two options:

- Service Whitelist: Every service must have the public keys or every service it communicates with. (Better security, but less managable)
- Trust whitelist: Every service certificate must be signed by a trusted third party. Getting the signature is allowing the service on the network. (Not as secure, but more managable)

1.2. Secure communication between services

Using X509 and standard Jini / SSL methods as for authentication. Here having a signed certificate is as secure as having the public keys.

1.3. User authentication

Users authenticate via JAAS modules. It must be a module that:

- Provides private credentials for signing of messages (either directly or is able to sign messages)
- The public credentials must be available to all services.

Examples:

- KeyStore Login. Provides X500 certificates. These can be in whitelist or signed by TTP.
- SmartCard Login. Provides operation to sign messages. Public key is in whitelist or signed by TTP
- Kerberos. Provides Ticket, which is used to sign messages. Service can check calidity of ticket with keystore.

We can then use "SignedServiceTask" for the requests.

1.4. Data encryption

Users authenticate via JAAS modules. It must be a module that:

- Provides a private key OR
- Provides a way to encrypt messages.
- The public key must be available on the client.

Examples:

render X

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- KeyStore provides X500 certificates. They are used to encrypt.
- SmartCard Login provides way to encrypt.
- Kerberos alone does NOT work! (But it does with Role Manager Service, see below)

Storing:

For every file there is a new symetric key generated. This key is encrypted with the public key. It is then stored in the MDS. If the file is available to multiple users the key is encrypted for all users and then stored.

Decryption:

retrieve the key from MDS, decrypt it with private credentials, use that to decrypt file.

1.5. Role-Manager Service

The presented system works for individuals, but not for roles (and not for Kerberos).

Problem: To support multiple roles we would need multiple secret keys

Solution: Role-Manager Service

Offers: A JAAS interface. Login is done via one of the authentication methods given above.

Offers: Methods to

- Get a list of Roles
- Sign for a Role
- Encrypt for a Role

The Role-Keys NEVER leave the RMS.

Each RMS may have a different admin -> scalability

RMS may synchronize with each other if both have the same admins

Nomadic RMS provides subset of keys.

1.6. Using Roles

Two Models:

- Classical Model: An admin creates directories. Problem: Scalability
- Silenus Model: Root is unwriteable for anyone but admin. Virtual directories (not listable) are auto-created for every user / role. When working in a directory, SILENUS defaults to the role for that directory.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/