Université Cheikh Anta DIOP de Dakar Laboratoire d’Algèbre de Cryptologie

Faculté des Sciences et Techniques de Géométrie Algébrique et Applications

Département Mathématiques et Informatique LACGAA

Licence Transmission de Données et Sécurité de l’Information

*Exposée :*

***PKI AVEC EJBCA ET LE PASSAGE DE HTTP VERS HTTPS :***

***Présentateurs n :***

*M. Amad Saïd NDIAYE*

*M. Moustapha PENE*

*Mlle Soda NDIAYE*

**Plan :**

I Introduction

II Présentation de EJBCA

1.EJBCA

2. Installation de EJBCA

3. Architecture fonctionnelle

4. Architecture applicative

5. Les normes et standards

6. Indépendance

III Concepts

1.Gestion des clés

2.Gestion des CA

3.Signature

VI Passage de http vers https

I. Introduction :

La montée en puissance des réseaux sociaux couplée à la popularité des nouvelles technologies mobiles et la dématérialisation des échanges, entraînent une vive polémique à propos de la confidentialité et de la sécurité de l’information transigée au sein des entreprises. Les infrastructures à clé publique ou PKI, qui ont suscité beaucoup d’intérêt à la fin des années 90, redeviennent un sujet informatique d’actualité. Ainsi, dans le but de mieux résoudre ce polémique, nous nous basons sur EJBCA qui est sous Licence LGPL.

Cette solution libre fonctionne sur un serveur d’application JBOSS. Elle est mature, et surtout elle a fait ses preuves. En plus des briques de base d’une PKI, elle propose l’ensemble des fonctionnalités recherchées pour la mise en place d’une infrastructure complète, telles que les services de révocation et d’horodatage. Nous associons EJBCA à Open LDAP pour les services d’annuaire (stockage des certificats et des clés publiques) et MySQL pour les bases de données. Et bien sûr, nous nous basons sur les standards tels que PKIX, PKCS ou X.509.

II. Présentation :

1. EJBCA

* Définition de EJBCA :

Dans un premier temps on définit PKI qui est une infrastructure à clés publiques (ICP) ou infrastructure de gestion de clés (IGC) ou encore Public Key Infrastructure (PKI), est un ensemble de composants physiques (des ordinateurs, des équipements cryptographiques logiciels ou matériel type HSM ou encore des cartes à puces), de procédures humaines (vérifications, validation) et de logiciels (système et application) destiné à gérer les clés publiques des utilisateurs d’un système. EJBCA est une PKI (Public Key infrastructure) ou IGC (Infrastructure de gestion de clés) sous licence Open Source (LGPL développée en Java/J2EE.

* EJBCA bien plus qu’une PKI :

EJBCA fournit à la fois les fonctions classiques que l’on retrouve dans la plupart des PKI du marché. Mais elle fournit en également un serveur OCSP, un serveur d’horodatage et un serveur de signature. Chaque composant peut être déployé de manière autonome ou intégrée dans EJBCA.

* EJBCA un générateur de vos besoins

EJBCA a été construit comme un générateur, permettant de répondre rapidement et sans développement complémentaire aux contraintes liées aux certificats ou au SI de l’entreprise. Le générateur permet de créer des profils de certificat, des formats de requête, des cinématiques de gestion du cycle de vie des certifications, de personnaliser des cartes à puce ou de dongle USB.

1. Installation

EJBCA est une autorité de certification entièrement fonctionnelle construite en Java. Basé sur la technologie JEE6, il constitue une autorité de certification(AC) robuste, performante et basée sur les composants. À la fois flexible et indépendant de la plate-forme, EJBCA peut être utilisé seul ou intégré dans n’importe quelle application JEE6.EJBCA est complètement écrit en Java et devrait donc fonctionner sur n’importe quelle plate-forme où un

serveur JEE s’exécute. Le développement et les tests sont effectués sur les plateformes Linux et Windows.

Conditions préalables de l’installation ejbca:

Remarque : EJBCA utilise des mots de passe crypto et keystore forts de plus de 7 caractères. Pour que cela fonctionne, vous devez installer les ’Unlimited Strength Jurisdiction Policy Files’ pour JDK. Les fichiers de règles peuvent être trouvés au même endroit que le téléchargement JDK chez Oracle. Le texte "Utilisation de la cryptographie exportable" s’affiche sur la première page de l’interface graphique d’administration si vous ne parvenez pas à installer ce package. Vous trouverez plus d’informations à ce sujet dans la documentation Oracle sur le JCE. Nécessaire pour construire et courir sont :

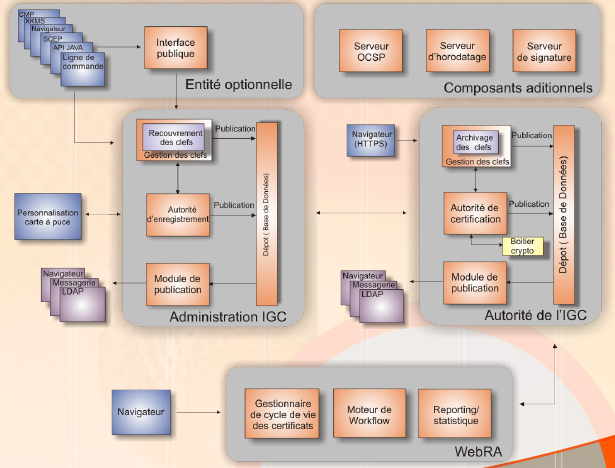
JDK 7 ou 8, Open JDK ou Oracle JDK. Si disponible, Open JDK est recommandé. Uniquement si vous utilisez OracleJDK - JPE (Java Cryptography Extension - extension de cryptographie Java) pour votre JDK.

JBoss Application Server 7.1.x ou version ultérieur JBoss EAP 6 ou version ultérieure (à télécharger, notez que la dernière version absolue ne peut pas toujours être testée) ou WildFly 9 ou plus tard.

Apache Ant 1.8 ou version ultérieure à construire (téléchargement). Notez que le support javascript peut être nécessaire dans ant pour certains composants.

Windows /Unix : Lorsque nous décrivons les commandes de ligne de commande ci-dessous, nous utilisons la notation unix, par exemple ’ejbca.sh’ pour les fichiers de commandes exécutables. Les mêmes fichiers de commandes sont disponibles pour Windows en tant que fichiers cmd, par exemple ’ejbca.cmd’. Si vous ne savez pas quelle version d’EJBCA vous utilisez, tapez ’ant ejbca version’ dans le répertoire EJBCA\_HOME. EJBCA propose cinq niveaux fonctionnels. Chaque ou l’ensemble des niveaux fonctionnel peut être déployé de manière unitaire ou consolidée sur une même machine ou dans une autre application.

3.Architecture fonctionnelle :



EJBCA propose quatre niveaux fonctionnels :

Chaque ou l’ensemble des niveaux fonctionnel peut être déployé de manière unitaire ou consolidée sur une même machine ou dans une autre application.

* Entité publique ou autorité d’enregistrement locale (AEL)

Elle est composée d’éléments permettant de réaliser des requêtes de certificats, de révocation ou pour récupérer des éléments publics de la PKI, tels les certificats d’AC, CRL, certificat utilisateur, etc.

Les méthodes d’appels fournies par EJBCA sont les suivantes :

* Au format Web, par le biais de page à intégrer dans un intranet ou de manière autonome
* Au format ligne de commande, permet en complément de réaliser des opérations d’administrations
* Sous forme d’API Java à intégrer dans des composants métiers
* Sous forme de client ou d’accès par un des protocoles supportés par EJBCA (voir caractéristique EJBCA)
* Par le biais du protocole SCEP pour les routeurs.
* Autorité de la PKI ou Autorité de certification :

Elles sont responsables de la délivrance et de la révocation des certificats

* Une même instance peut gérer plusieurs AC racine et AC subordonnées
* Les clefs peuvent être conservées sur des boîtiers cryptographiques (voir caractéristiques EJBCA), sur carte à puce ou dans la base de données
* Entité administrative ou Autorité d’enregistrement

Elle englobe les fonctions de gestion de la PKI, telles les fonctions d’authentification, de gestion de vie des certificats, de journalisation, de publication et de paramétrages de la PKI.

En complément, elle permet de définir le format des entités publiques et le contenu des certificats. Les fonctions incluses dans l’AE :

* Gestion des profils de certificat et d’entité (permet de définir le contenu d’un formulaire de demande de certificats)
* gestionnaire de clefs (séquestre et recouvrement)
* Définition du service de publication
* Gestion des administrateurs et des droits associés
* Gestion des profils pour les cartes à puce et Dongle USB
* Demande, rejet et Validation des requêtes de certificat étude révocation
* Processus de gestion du renouvellement
* Gestionnaire de journaux (paramétrage, consultation, export, etc.).
* WebRA

Dernier né des entités d’EJBCA, il a été conçu pour renforcer la partie fonctionnelle d’EJBCA, il comprend les éléments suivants :

* Modules de gestion de vie des certificats (prè-condition, demande de certificats, révocation, renouvellement, etc.)
* Moteur de Worflow, permet de définir les cinématiques du cycle de vie des certificats
* Outils de reporting, statistiques, facturation
* Indépendance du gestionnaire d’authentification et d’habilitation

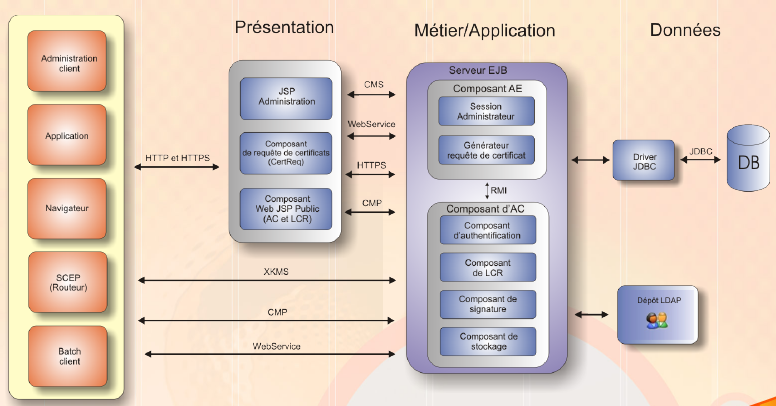
Communication en WebService sécurisé avec EJBCA.

* Composants additionnels

Ces composants sont proposés en compléments de la PKI EJBCA, l’ensemble de leurs fonctionnalités n’est pas décrit dans cette présentation

* Serveur de signature (SignServer), permet de réaliser des opérations de signatures sur des documents
* Serveur d’horodatage (SignServer), garantir électroniquement la date et l’heure d’une opération
* Serveur OCSP (EJBCA), permet de fournir une réponse immédiate et à jour aux questions sur le statut d’un certificat

4.Architecture applicative



EJBCA est découpée en trois couches :

* Couche présentation

Elle fournit les interfaces aux clients et administrateurs pour les appels vers l’autorité de certification ou vers l’autorité d’enregistrement.

* Une couche applicative

Elle englobe les fonctions métiers de la PKI, elle comprend les éléments suivants :

* Les fonctions d’authentification, de gestion de vie des certificats, de journalisation, de publication et de paramétrages de la PKI
* En complément, elle fournit un gestionnaire de clefs (séquestre et recouvrement)
* Une couche de données :

Cette couche permet de stocker l’ensemble des données de la PKI dans une base de données

Il possible de connecter un ou plusieurs annuaires LDAP pour lui adresser les données publiques de la PKI (utilisateurs, certificats, CRL, etc.).

5.Les normes et standards

EJBCA est développé au plus près des standards, il prend en charges les préconisations de :

• l’IETF (Internet Engineering Task Force) pour la partie X509v3 (certificats numériques) et CRLv2, ainsi que les normes PKCS, SCEP. En complément, EJBCA intègre CMP (Certificate Management Protocol) pour l’interopérabilité avec les autres solutions de PKI.

• Le W3C (World Wide Web Consortium) pour les technologies orientées architecture

(SOA), tels XMLDsign (signature numérique), XKMS pour la validation et l’enregistrement des certificats, WebService pour les appels entre les entités.

6. Indépendance

EJBCA est indépendant des systèmes d’exploitation, des bases de données ou des serveurs applicatifs.

• Il se déploie sur la plupart des systèmes Microsoft ou Unix, tels que Linux, FreeBSD ou Solaris.

• EJBCA fonctionne sur un serveur d’application, il est compatible avec les grands produits du marché, tels JBOSS, JONAS, GlashFish, WebLogic, WebSphere.

• EJBCA est compatible avec les bases de données supportées par les serveurs d’application, les bases suivantes sont supportées :

• MySQL

• Oracle

• PostgreSQL

• DB2

• Sysbase

• Informix

• MS-SQL2000

III. Concepts

1.Gestion des clés:

* Création de clé :

• certSignKey - la clé à utiliser lors de la signature des certificats, peut être RSA ou ECDSA.

• crlSignKey - la clé à utiliser lors de la signature de CLS, peut être RSA ou ECDSA.

• keyEncryptKey - la clé à utiliser pour le cryptage et le décryptage des clés, il doit s'agir d'une clé RSA.

• testKey - la clé à utiliser par les contrôles d'état HSM, peut être RSA ou ECDSA.

• hardTokenEncrypt - la clé à utiliser pour le cryptage et le décryptage hardtoken. PUK sera décrypté par cette clé.

• defaultKey - la clé à utiliser lorsqu'aucune autre clé n'est définie dans un but. Si c'est la seule définition, cette clé sera utilisée à toutes fins.

openssl genrsa -out test.key

* Générez une nouvelle paire de clés:

$ ./openssl genrsa -out certs / cl\_new\_key.pem 2048

* Générer des clés sur un emplacement

$ ./ejbcaClientToolBox.sh PKCS11HSMKeyTool générer /usr/lunasa/lib/libCryptoki2\_64.so 2048 rsa2048\_1 1

0 [main] INFO org.ejbca.util.keystore.KeyTools - Utilisation du fournisseur SUN PKCS11: sun.security.pkcs11.SunPKCS11

PKCS11 Token [SunPKCS11-Luna] Mot de passe:

Certificat créé avec l'entrée rsa2048\_1.

$ ./ejbcaClientToolBox.sh PKCS11HSMKeyTool générer /usr/lunasa/lib/libCryptoki2\_64.so secp160r1 secp160r1\_1 1

0 [main] INFO org.ejbca.util.keystore.KeyTools - Utilisation du fournisseur SUN PKCS11: sun.security.pkcs11.SunPKCS11

PKCS11 Token [SunPKCS11-Luna] Mot de passe:

Certificat créé avec l'entrée secp160r1\_1.

* Lister et tester toutes les clés qui pourraient être utilisées par EJBCA

$ ./ejbcaClientToolBox.sh PKCS11HSMKeyTool test /usr/lunasa/lib/libCryptoki2\_64.so 1

Test du magasin de clés avec ID 1.

0 [main] INFO org.ejbca.util.keystore.KeyTools - Utilisation du fournisseur SUN PKCS11:

sun.security.pkcs11.SunPKCS11

PKCS11 Token [SunPKCS11-libCryptoki2\_64.so-slot2] Mot de passe:

Test de clé: rsa2048\_1

SunJCE version 1.7SunPKCS11-libCryptoki2\_64.so-slot2 version 1.7; longueur du module: 2048; octet longueur 245. La chaîne octet décodée est égale à l'original!

Test de signature de la clé rsa2048\_1: longueur de signature 256; premier octet 28; vérifier vrai

Statistiques clés

Signalisations par seconde: 369; Décryptions par seconde: 135

Test de clé: secp160r1\_1

Test de signature de la clé secp160r1\_1: longueur de la signature 48; premier octet 30; vérifier vrai

Statistiques clés

Signalisations par seconde: 68 Pas de chiffrement disponible pour cette clé.

2.Gestion des CA :

* Gréation de CA

Génération d'une clé

openssl gen[type] -out [fichier] [taille]

Création du CSR

openssl req -new -config [config]

Création du certificat CA autosigné

openssl ca -infile [fichier] -selfsign ..

* Création d'une demande de certificate
* Création du fichier req.cnf
* Création de la clé du certificat
  + *openssl genrsa -out [fichier] taille*
* Création du fichier CSR
  + *openssl req -config req.cnf -new -out [fichier] -key [clé]*
* Validation du certificat
* La clé privée du certificat n'est pas nécessaire
* La clé du certificat racine est requise
* Création du certificat

*openssl ca -policy [police] -config [config] -in [csr.pem] -out [cert.pem]*

* Envoie du fichier CSR à l'autorité
* Exporter CA

Pour exporter une autorité de certification nommée "TestCA" vers le fichier PKCS # 12 "/path/TestCA.p12" avec le mot de passe "foo123", entrez ce qui suit à partir du répertoire $ EJBCA\_HOME:

$ bin / ejbca.sh ca exportca TestCA ./TestCA.p12

Utilisation du fournisseur JBoss JNDI ...

Entrez le mot de passe du fichier de clés: foo123

$

* Importer un CA

Pour importer les clés de sauvegarde pour "TestCA" plus tard, entrez ce qui suit à partir du répertoire $ EJBCA\_HOME:

$ bin / ejbca.sh ca importca TestCA /chemin/TestCA.p12 SignatureKeyAlias ​​EncryptionKeyAlias

Utilisation du fournisseur JBoss JNDI ...

Entrez le mot de passe du fichier de clés: foo123

$

* Supprimer un CA

Avertissement:

Avant de supprimer le fichier de clés, assurez-vous de l'avoir exporté si vous souhaitez pouvoir le restaurer ultérieurement. Voir la section 'Exporter et importer des AC' .

Pour supprimer les clés catoken de "TestCA", entrez les informations suivantes dans le répertoire $ EJBCA\_HOME:

$ bin / ejbca.sh ca removekeystore TestCA

Utilisation du fournisseur JBoss JNDI ...

$

* Restaurer un CA

Pour restaurer les clés catoken à nouveau pour "TestCA" avec le fichier de clés exporté sous "TestCA-exported.p12", entrez les informations suivantes dans le répertoire $ EJBCA\_HOME:

$ bin / ejbca.sh ca restorekeystore TestCA /chemin/TestCA-exported.p12 -s SignatureKeyAlias ​​-e EncryptionKeyAlias

Utilisation du fournisseur JBoss JNDI ...

Entrez le mot de passe du fichier de clés: foo123

$

* Inscription initiale du certificat client en utilisant un code d'inscription unique.

Avant l'inscription initiale, vous ajoutez une nouvelle entité finale dans EJBCA, dans cet exemple avec le nom d'utilisateur cmptest et le sujet DN 'CN = cmptest', et le code d'inscription 'CMP-pwd'.

$ ./openssl cmp -cmd ir \

-server ejbca-server: 8080 \

-chemin ejbca / publicweb / cmp \

-srvcert certs / 3GPPCA.pem \

-utilisateur cmptest \

-pass CMP-pwd \

-newkey certs / cl\_key.pem \

-certout certs / cl\_cert.pem \

-subject "/ CN = cmptest"

* Renouveler avec un nouveau certificat:

$ ./openssl cmp -cmd kur \

-server ejbca-server: 8080 \

-chemin ejbca / publicweb / cmp \

-srvcert certs / 3GPPCA.pem \

-key certs / cl\_key.pem \

-cert certs / cl\_cert.pem \

-newkey certs / cl\_new\_key.pem \

-certout certs / cl\_new\_cert.pem

1. Signature numérique :

La possibilité de vérifier l'intégrité d'une version ou d'une archive EAR déployée peut être requise pour certaines configurations. Actuellement, les deux cibles ant "ziprelease" et "ca.ear" (invoquées depuis "deploy" et la cible par défaut) prennent en charge la signature jar avec la commande "jarsigner" incluse avec Java. Notez que vous pouvez toujours supprimer des fichiers d'une archive signée sans que personne ne s'en aperçoive puisque les fichiers sont signés individuellement. Pour créer un ziprelease signé de EJBCA:

ant ziprelease -Dejbca.zipversion = x\_y\_z -Dsignjar.keystore = p12 / releasesigner.jks -Dsignjar.keystorealias = libérateur -Dsignjar.keystorepass = foo123

Le certificat utilisé pour la signature doit avoir une utilisation de clé "Signature numérique" et une utilisation de clé étendue "Signature de code". L'archive signée peut être vérifiée à l'aide de la commande "jarsigner" et du certificat CA. Cet exemple affichera tout fichier ou fichier non signé avec une mauvaise signature:

$ jarsigner -verify -keystore p12 / truststore.jks -verbose ../ejbca\_x\_y\_z.zip | grep -v "^ smk" | grep -v "^ \* 0"

246809 Mar 21 Oct 13:28:48 CEST 2008 META-INF / MANIFEST.MF

246930 Mar 21 Oct 13:28:48 CEST 2008 META-INF / RELEASES.SF

1859 Mar 21 Oct 13:28:48 CEST 2008 META-INF / RELEASES.RSA

s = la signature a été vérifiée

m = l'entrée est répertoriée dans le manifeste

k = au moins un certificat a été trouvé dans keystore

i = au moins un certificat a été trouvé dans la portée de l'identité

pot vérifié.

OpenSSL peut être utilisé pour signer et vérifier une archive entière, mais requiert la clé publique du certificat de signature:

$ openssl dgst -sha1 -sign p12 / pem / releasesigner-Key.pem -out

../ejbca\_x\_y\_z.zip.SHA1withRSA ../ejbca\_x\_y\_z.zip

$ openssl x509 -inform pem -dans p12 / pem / releasesigner.pem -pubkey -noout> p12 / pem / releasesigner-Pub.pem

$ openssl dgst -sha1 -verify p12 / pem / libérateur-Pub.pem -signature ../ejbca\_x\_y\_z.zip.SHA1withRSA ../ejbca\_x\_y\_z.zip

Vérifié OK

VI Passage de http vers https :

C’est à démontrer dans la pratique.