PLATAFORMA DE DESENVOLVIMENTO PINHÃO PARANÁ

INSTALAÇÃO DE CERTIFICADO DIGITAL EM APLICAÇÃO WEB - JBOSS.

Agosto – 2006

Sumário de Informações do Documento

Tipo do Documento: Definição

Título do Documento: INSTALAÇÃO DE CERTIFICADO DIGITAL EM APLICAÇÃO WEB-JBOSS.

Estado do Documento: EB (Elaboração)

Responsáveis: Emerson Sachio Saito

Palavras-Chaves: Certificado, Digital, Certificação, Aplicação, WEB, ICP-BRASIL, Tabelião.

Resumo: Documento para orientação de instalação de Certificado Digital em Aplicações WEB.

Número de páginas: 18

Software utilizados: OpenOffice Writer

Versão	Data	Mudanças
1.0	01/08/2006	Criação
1.1	28/08/2007	Incluído item para vários certificados no mesmo servidor de aplicação – revisado por Elisabeth Hoffmann.

SUMÁRIO

1 CONTEXTO GERAL

1.1 Introdução

A CELEPAR através da GIC (Gerência de Inovação Corporativa) definiu algumas normas para utilização de Certificados Digitais. A principal delas é que será adotado o padrão ICP-BRASIL para quase todos os níveis de certificação, excetuando-se apenas aquelas aplicações de uso estritamente interno e estes casos, devem ser resolvidos por consulta à área demandante. Sendo assim, para utilizar as informações contidas neste manual, é necessária a aquisição de um Certificado Digital de uma Autoridade de Registro vinculada a uma Autoridade Certificadora CREDENCIADA junto à ICP-BRASIL.

Outros aspectos e normas relacionados à Certificação Digital são abordados no curso de Nivelamento Teórico que é oferecido pela GIC através do projeto TABELIÃO do qual este documento faz parte. Os conhecimentos adquiridos no citado curso são importantes, e considerados pré-requisitos para a utilização deste manual.

1.2 Tipos e Níveis de Certificado.

Iremos abordar no texto abaixo, apesar deste conhecimento constar no curso de Nivelamento Teórico, os tipos e nível que serão aceitos. É apenas uma orientação para a seqüencia normal das explicações.

Dentro das normas da ICP-BRASIL existem 2 (dois) tipos de certificado: Assinatura e Sigilo e cada um deles possui 4 níveis que são os seguintes:

De Assinatura:

A1: Chave criptográfica de 1024 bits gerada por software válida por 1 ano

A2: Chave criptográfica de 1024 bits gerada por hardware válida por 2 anos

A3: Chave criptográfica de 1024 bits gerada por hardware válida por 3 anos

A4: Chave criptográfica de 2048 bits gerada por hardware válida por 3 anos

De Sigilo:

S1: Chave criptográfica de 1024 bits gerada por software válida por 1 ano

S2: Chave criptográfica de 1024 bits gerada por hardware válida por 2 anos

S3: Chave criptográfica de 1024 bits gerada por hardware válida por 3 anos

S4: Chave criptográfica de 2048 bits gerada por hardware válida por 3 anos

A avaliação e determinação de qual tipo e nível que será utilizado cabe ao analista responsável pela aplicação, para isso, existe o curso de Nivelamento Teórico que capacita o analista a ter o conhecimento necessário para desempenhar está tarefa.

2 OBJETIVO

Este documento orienta a configuração para uso de Certificados Digitais, conforme os requisitos esclarecidos anteriormente, para aplicações em servidores JBOSS (nas versões homologadas pela CELEPAR). O uso em outros tipos de servidores de aplicação WEB ou certificados fora do padrão ICP-BRASIL não serão abordados neste documento.

3 CERTIFICAÇÃO DE APLICAÇÃO EM SERVIDORES JBOSS.

3.1 Requisitos iniciais.

Conforme definido no item 1.2 é preciso que o analista responsável determine que tipo e nível de certificado será utilizado pela aplicação e providencie a sua aquisição antes do inicio das tarefas que serão apresentadas.

a) Preparação

Criar uma pasta aonde serão temporariamente armazenados os arquivos a serem gerados e os certificados adquiridos:

mkdir .../certificados

cd .../certificados

b) Armazenar as cadeias para o Certificado adquirido (xxx.cer) no diretório criado. São necessárias as cadeias da ICP-BRASIL (RAIZ) e todas as outras conforme o certificado que foi adquirido, por norma da ICP-BRASIL a Autoridade Certificadora deve manter disponível na

Internet estas cadeias de certificados, mas também é comum que na entrega do certificado as cadeias vejam junto.

c) Armazenar o certificado adquirido.

3.2 Configuração do SSL para o protocolo HTTP.

No JBOSS o sub módulo que implementa o contêiner Web e que, portanto, implementa o protocolo HTTP é o TOMCAT. Então, para configurar o HTTP/SSL no JBOSS, é preciso configurar o TOMCAT dentro do JBOSS.

Além disso, será necessário utilizar ferramentas da JDK, para a manipulação de chaves e certificados que serão utilizados no TOMCAT.

3.2.1 O que é a tag Connector.

Para cada protocolo de acesso suportado pelo Jboss/Tomcat, é necessário criar uma configuração usando a tag *Connector*. Por default, existem 2 protocolos configurados e habilitados: o HTTP e o AJP. O HTTP, é utilizado para o acesso direto dos Navegadores(browsers) e o AJP é utilizado quando se utiliza um outro servidor HTTP junto com o Jboss/Tomcat, tal como o Apache HTTPD. Nesse caso, o Navegador se comunica com o Apache HTTPD via protocolo HTTP e o Apache se comunica com o Jboss/Tomcat através do protocolo AJP.

3.2.2 KeystoreFile

Para a configuração do JBOSS para uso do protocolo HTTPS é recomendado efetuar a criação de um arquivo keystore (chaveiro) padrão java (JKS). Este arquivo funciona como um banco de dados de certificados digitais, no qual armazenamos as chaves públicas e privadas. Essas chaves são usadas para vários propósitos, incluindo autenticação e integridade de dados. No caso dos certificados adquiridos de Autoridades de Registro CREDENCIADAS na ICP-BRASIL o formato do arquivo armazena uma chave pública e outra privada no formato PKCS12 que também podem ser utilizadas como um KeystoreFile, mas não estão no formato padrão.

3.2.3 Configuração do Tomcat

Para configuração do TOMCAT (/usr/lib/jboss4/server/default/deploy/jbossweb-tomcat55.sar) é preciso editar o arquivo server.xml que fica no diretório apontado.

Neste arquivo deverá ser encontrado o seguinte trecho, que é a nossa tag Connetor (que provavelmente estará como comentário):

```
<!-- SSL/TLS Connector configuration using the admin devl guide keystore
    <Connector port="8443" address="${jboss.bind.address}"
        maxThreads="100" strategy="ms" maxHttpHeaderSize="8192"
        emptySessionPath="true"
        scheme="https" secure="true" clientAuth="false"
        keystoreFile="${jboss.server.home.dir}/conf/chap8.keystore"
        keystorePass="rmi+ssl" sslProtocol = "TLS" />
    -->
```

O primeiro passo, então, é retirar o comentário. É possível manter os valores *default*, mas será necessário alterar pelo menos 2 parâmetros da tag Connector: ***keystoreFile*** e ***keystorePass***. Também é importante trocar o número da porta de 8443 para 443 que é porta padrão. Só mantenha o valor 8443 ou outro qualquer se já existir um servidor, ou instância, HTTP/SSL ativo no seu computador, ou ainda, se a inicialização do servidor for feita com um usuário comum que poderá causar problemas de autorização para uso da porta 443.

3.2.3.1 Cadeia de Certificados.

Como estamos usando certificados padrão ICP-BRASIL teremos primeiro que importar as cadeias de certificado para o Keystore do JAVA($JAVA_HOME/jre/lib/security/cacerts) para garantir a validade dos certificados. Nos casos onde o servidor já está configurado e já existe um certificado da mesma Autoridade Certificadora não será necessário os passos seguintes, porém, caso o servidor esteja configurado, mas o certificado é de outra Autoridade Certificadora, será necessário incluir as cadeias desta, mas não será necessária a inclusão da cadeia RAIZ.

Outra possibilidade é importar as cadeias válidas para dentro do chaveiro java (JKS),

esta modalidade é possível com o uso da ferramenta KeytoolIUI-Plus (veja o documento: **GIC_KeytoolIUI_manual**) e é a forma mais recomendada para garantir a independência entre as configurações de cada aplicação que rodará no servidor.

Primeiro passo é abrir uma tela de terminal (shell linux).

Verifique o diretório de instalação do Java (JAVA_HOME), de preferência utilize uma variável de sessão. O padrão da SUN é /usr/lib/jvm/_versão_, mas pode mudar conforme a instalação. Daqui em diante assumiremos a nomenclatura $JAVA_HOME para indicar o diretório de instalação do Java.

Para verificar quais cadeias já existem, utilize o comando abaixo:

$JAVA_HOME/bin/keytool -v -list -keystore $JAVA_HOME/jre/lib/security/cacerts

Para executar as tarefas seguintes é preciso permissão de ROOT (su).

Caso seja a primeira vez que está sendo feita esta operação é preciso mudar a senha do Keystore padrão do JAVA com o seguinte comando: $JAVA_HOME/bin/keytool -storepasswd -new <nova_senha> -storepass changeit -keystore $JAVA_HOME/jre/lib/security/cacerts

A senha "changeit" é a senha ***default,*** é possível que ela já tenha sido mudada caso não seja a primeira vez que a operação esteja sendo executada no servidor.

Em todas as operações listadas neste item a senha do Keystore será exigida.

É recomendável também que sejam excluídas quaisquer outras cadeias que não estejam sendo utilizadas ou que não estejam no padrão ICP-BRASIL:

Para fazer isto basta verificar quais os "Alias name" que se encaixam neste quesito, (utilize o comando: $JAVA_HOME/bin/keytool -v -list -keystore $JAVA_HOME/jre/lib/security/cacerts), e em seguida exclua-os com o seguinte comando:

$JAVA_HOME/bin/keytool -v -delete -alias <alias_a_excluir> -keystore $JAVA_HOME/jre/lib/security/cacerts

Para facilitar esta tarefa foram criados alguns "*shell scripts (.sh)*" que são disponibilizados pela GIC no site do PINHÃO.

Depois de eliminadas todas as entradas indesejadas é preciso importar todas as cadeias necessárias:

Para verificar a integridade das cadeias que foram armazenadas no item 3.1 utiliza-se o comando abaixo para cada arquivo:

$JAVA_HOME/bin/keytool -printcert -file ACxxx.cer

-Para inclusão da cadeia da ACRAIZ:

$JAVA_HOME/bin/keytool -import -alias ACRAIZ -keystore $JAVA_HOME/jre/lib/security/cacerts -trustcacerts -file ACraiz.cer

-Para ACs de Nivel1 e demais sub-níveis existentes ,repetir o comando de acordo com o número de níveis (arquivos):

$JAVA_HOME/bin/keytool -import -alias AC<xxx> -keystore $JAVA_HOME/jre/lib/security/cacerts -trustcacerts -file AC<xxx>.cer

Para cada comando uma mensagem parecida com a que está abaixo aparecerá:

Enter keystore password: (informar uma senha para o keystore)

Owner: CN=Autoridade Certificadora Raiz Brasileira, ST=DF, L=Brasilia,
OU=Instituto Nacional de Tecnologia da Informacao - ITI, O=ICP-Brasil, C=BR

Issuer: CN=Autoridade Certificadora Raiz Brasileira, ST=DF, L=Brasilia,
OU=Instituto Nacional de Tecnologia da Informacao - ITI, O=ICP-Brasil, C=BR

Serial number: 8b4cc1708955cc5e

Valid from: Tue Mar 29 14:22:41 BRT 2005 until: Fri Mar 27 14:22:41 BRT 2015

Certificate fingerprints:

MD5: 85:22:8A:15:EC:82:83:91:79:CA:25:C5:30:6C:03:92

SHA1: 9B:EB:59:2B:A1:D8:4E:FE:2C:5D:0E:80:6D:A9:44:64:B5:C8:AB:D8

Trust this certificate? [no]: yes

Certificate was added to keystore

Pode-se verificar o conteúdo do keystore com o comando:

$JAVA_HOME/bin/keytool -v -list -keystore $JAVA_HOME/jre/lib/security/cacerts

3.2.3.2 Configuração para uso do certificado no formato PKCS12.

Na maioria dos casos os certificados adquiridos de uma Autoridade de Registro CREDENCIADA na ICP-BRASIL vem em arquivos em formato PKCS12 (.pfx ou .p12), quando gerados em arquivo, que pode ser utilizado como um Keystore do JBOOS/TOMCAT, nestes casos a configuração do TOMCAT deverá ser feita da seguinte maneira:

Editar arquivo SERVER.XML que é apresentado abaixo:

```
<!--SSL/TLS Connector configuration using the admin devl guide keystore -->
<Connector port="8443" address="${jboss.bind.address}"
    maxThreads="100" strategy="ms" maxHttpHeaderSize="8192"
    emptySessionPath="true"
    scheme="https" secure="true" clientAuth="false"
    keystoreFile="/usr/lib/jboss4/server/default/conf/<nome_arquivo.pfx>"
    keystoreType="PKCS12"
    keystorePass="senha do certificado" sslProtocol = "TLS" />
```

Os itens destacados em verde é que deverão ser alterados, e no caso do uso do certificado em PKCS12 a tag KeystoreType deve ser adicionada, lembrar de colocar o arquivo no diretório apontado por keystoreFile.

Feito isto é só reiniciar o servidor e testar a aplicação.

Para utilizar o arquivos em formato PKCS12, e não o padrão JKS, é imprescindível a inclusão da cadeia de certificados (item anterior) no keystore do sistema.

3.2.3.3 Configuração para uso do certificado em Keystore JAVA.

Uma outra forma de utilização de certificados é utilizar um Keystore (chaveiro) no padrão JAVA, normalmente na extensão .jks

Com esta abordagem é possível utilizar vários certificados em um mesmo Keystore e conforme a necessidade, a inclusão de novos certificados para novas aplicações no mesmo servidor.

Esta é a uma forma de poder certificar várias aplicações do mesmo servidor, sem a necessidade de espalhar vários arquivos de certificado para cada aplicação. Além de permitir a proteção da senha de cada certificado.

Outra vantagem também é que não será necessário incluir as cadeias no Keystore padrão do Java. (item 3.2.3.1)

Caso haja a necessidade de validação de clientes (como por exemplo: WebServices), esta também é a forma mais elegante de implementar.

Para este tipo de solução é recomendado o uso de uma outra ferramenta de gerenciamento de chaveiros(keystore) chamado KeyToolIUI (utilizar o documento **GIC_KeytoolIUI_Manual**).

Com o uso desta ferramenta os passos para configuração serão os seguintes:

Criar um Keystore vazio no formato JKS (definir nome e senha, como padrão pode ser adota a seguinte nomenclatura para o nome do arquivo: Jboss_<servidor>.jks)

Depois importar as chaves do certificado no formato PKCS12 (.pfx, .p12) para o keystore (chaveiro), e também importar os certificados das ACs (Autoridades Certificadoras).

Este trabalho também pode ser feito através de linha de comando utilizando apenas a ferramenta keytool que é padrão do Java, mas não detalharemos esta tarefa pois recomendamos o uso da interface gráfica que facilita muito o trabalho.

Em seguida configurar o arquivo SERVER.XML conforme exibido abaixo:

```
<!--SSL/TLS Connector configuration using the admin devl guide keystore -->
   <Connector port="8443" address="${jboss.bind.address}"

      maxThreads="100" strategy="ms" maxHttpHeaderSize="8192"

      emptySessionPath="true"
```

```
scheme="https" secure="true" clientAuth="false"
keystoreFile="/usr/lib/jboss4/server/default/conf/chaveiro.jks>"
keystorePass="senha do certificado" sslProtocol = "TLS" />
```

Neste caso não é mais necessária a tag KeystoreType.

Não esquecer de incluir o keystore (chaveiro) criado, no diretório definido por keystoreFile.

Feito isto é só reiniciar o servidor e testar as aplicações.

3.2.3.4 Configuração para uso do certificado armazenado em Token/SmartCard.

Quando os certificados adquiridos estiverem armazenados em dispositivos criptográficos como Token ou SmartCard a configuração mudará um pouco em relação aos armazenados em arquivo.

Certificados em Token ou SmartCard garantem um nível muito maior de segurança, pois garantem a sua unicidade.

Primeiramente será necessário fazer a instalação e configuração do Token ou Leitora de SmartCard no equipamento, conforme as instruções do documento **GIC_ManualInstalacaoLeitoraseToken,** que também está disponível no site do PINHÃO.

Com o dispositivo instalado e configurado o segundo passo é a criação de um arquivo de configuração para acesso ao dispositivo que será utilizado pelo TOMCAT/JBOSS.

Configuração mínima que o arquivo deverá conter:

- Alias para o provider
- Nome da biblioteca PKCS#11
- Ex: name = smartcard

  library = /usr/lib/opensc/opensc-pkcs11.so

Também é possível habilitar outras configurações conforme a referência da documentação da SUN.

Salvar o arquivo com um nome sugestivo e em um diretório acessível como por exemplo: /etc/pkcs11.cfg

Em seguida será preciso editar o arquivo JAVA.SECURITY ($JAVA_HOME/jre/lib/security/java.security), para registrar o provider da SUN para acesso ao dispositivo conforme listado abaixo:

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.crypto.provider.SunJCE
security.provider.5=sun.security.jgss.SunProvider
security.provider.6=com.sun.security.sasl.Provider
security.provider.7=sun.security.pkcs11.SunPKCS11 /etc/pkcs11.cfg
```

O último passo é a configuração do arquivo SERVER.XML (usr/lib/jboss4/server/default/deploy/jbossweb-tomcat55.sar) conforme listado abaixo:

```
<Connector port="8443" address="localhost"
           maxThreads="100" strategy="ms" maxHttpHeaderSize="8192"
          emptySessionPath="true"
           scheme="https" secure="true" clientAuth="false"
          keystoreType="PKCS11"
           ciphers="TLS_RSA_WITH_AES_128_CBC_SHA"
          keystorePass="senha"    sslProtocol = "TLS" />
```

4 UTILIZANDO VÁRIOS CERTIFICADOS NO MESMO SERVIDOR DE APLICAÇÃO.

Podem haver casos em que mais de um sistema ou aplicação irá rodar no mesmo servidor JBOSS.

Como o certificado digital é emitido para identificar cada sistema, não é possível utilizar o mesmo certificado para dois ou mais sistemas diferentes, que geralmente é identificado pela sua URL.

A solução mais simples, porém mais onerosa para o equipamento, seria a de iniciar mais de um servidor (instância) do JBOSS utilizando portas diferentes. Em muitos casos esta prática não é recomendável por exigir mais do equipamento em termos de memória e processamento, além de dificultar a administração do ambiente.

Assim, nos casos em que a necessidade é utilizar somente uma instância de JBOSS o correto é configurar tanto o JBOSS como os sistemas, para que o mesmo servidor responda por duas ou mais URLs diferentes, e cada um possa utilizar o seu próprio certificado digital.

4.1 Configuração do Servidor.

Para o servidor de aplicação será necessário configurar o conceito de "*Host Virtual*", isto fará com que o servidor responda por "*Alias*" representando URLs ou Ips conforme a necessidade, esta configuração é feita, também, no arquivo SERVER.XML que foi referenciado nos itens anteriores.

Dentro da *TAG* ***Engine*** já existe uma *tag* para *Host* com *name=localhost*, que é o padrão da instalação. O que deve ser feito é a inclusão de novas TAGs para cada contexto que será virtualizado.

Abaixo um exemplo de configuração:

```
<!-- host virtual -->
        <Host name="AplicacaoUm"
          autoDeploy="false" deployOnStartup="false" deployXML="false"
        configClass="org.jboss.web.tomcat.security.config.JBossContextConfig">
          <Alias>aplicacaoum.pr.gov.br</Alias>
```

```
                <Valve className="org.jboss.web.tomcat.tc5.jca.CachedConnectionValve"
             cachedConnectionManagerObjectName="jboss.jca:service=CachedConnectionMana
ger"
            transactionManagerObjectName="jboss:service=TransactionManager"/>
                <Valve className="org.apache.catalina.valves.FastCommonAccessLogValve"
                prefix="aplicacaoum" suffix=".log" pattern="common"
            directory="${jboss.server.home.dir}/log"
                resolveHosts="true"/>
                <DefaultContext cookies="true" crossContext="true" override="true"/>
        </Host>
<Host name="AplicacaoDois"
            autoDeploy="false" deployOnStartup="false" deployXML="false"
          configClass="org.jboss.web.tomcat.security.config.JBossContextConfig">
            <Alias>aplicacaodois.pr.gov.br</Alias>
                <Valve className="org.jboss.web.tomcat.tc5.jca.CachedConnectionValve"
             cachedConnectionManagerObjectName="jboss.jca:service=CachedConnectionMana
ger"
            transactionManagerObjectName="jboss:service=TransactionManager"/>
                <Valve className="org.apache.catalina.valves.FastCommonAccessLogValve"
                prefix="aplicacaodois" suffix=".log" pattern="common"
            directory="${jboss.server.home.dir}/log"
                resolveHosts="true"/>
                <DefaultContext cookies="true" crossContext="true" override="true"/>
        </Host>
```

Desta forma além do padrão LOCALHOST, o servidor também responderá por requisições para aplicacaoum.pr.gov.br e aplicacaodois.pr.gov.br

Não entraremos em detalhes sobre configurações de DNS que deverão ser resolvidas pela área responsável. Para necessidades de testes a edição do arquivo de *hosts* do equipamento já deverá ser suficiente.

Depois desta configuração será necessário configurar uma tag CONNECTOR de SSL/TLS para cada Host Virtual que foi criado. Com isto, habilitaremos os certificados digitais para cada um deles.

Abaixo um exemplo de configuração:

```
<Connector port="8443" address="aplicacaoum.pr.gov.br"
        maxThreads="100" strategy="ms" maxHttpHeaderSize="8192"
        emptySessionPath="true"
        scheme="https" secure="true" clientAuth="false"
        keyAlias="aplicacaoum.pr.gov.br"
        keystoreFile="${jboss.server.home.dir}/conf/chaveiro.jks"
        keystorePass="senha"
          sslProtocol = "TLS" />
<Connector port="8443" address="aplicacaodois.pr.gov.br"
        maxThreads="100" strategy="ms" maxHttpHeaderSize="8192"
        emptySessionPath="true"
        scheme="https" secure="true" clientAuth="false"
        keyAlias="aplicacaodois.pr.gov.br"
        keystoreFile="${jboss.server.home.dir}/conf/chaveiro.jks"
        keystorePass="senha"
          sslProtocol = "TLS" />
```

Note que mais uma *tag* foi inserida, que é a *keyAlias*, ela identificará qual é o certificado armazenado no Keystore (chaveiro) que deverá ser utilizado pela tag.

De acordo com esta configuração teríamos as aplicações Um e Dois respondendo cada qual pela sua URL e utilizando cada um dos certificados.

É necessário reiniciar o servidor de aplicação.

4.2 Configuração do Sistema.

Com a parte do servidor já resolvida, será necessário fazer algumas configurações na aplicação que foi desenvolvida.

A principal configuração é a criação de um arquivo XML na pasta WEB-INF de cada aplicação para as quais foram criados os Hosts Virtuais. O arquivo dever ter o nome de **jboss-web.xml** e deverá conter as seguintes informações:

```
<jboss-web>
    <context-root>/</context-root>
    <virtual-host>aplicacaoum.pr.gov.br</virtual-host>
</jboss-web>
```

Esta configuração já será suficiente para que o servidor entenda qual aplicação responderá por cada Host Virtual.

Outra configuração que pode ser implementada é a que evitará o acesso à aplicação pela porta (80 ou 8080) não segura (sem uso do certificado).

No arquivo WEB.XML localizado também na pasta WEB-INF deve-se inserir a tag **security-constraint**.

```
<!-- evita acesso via http -->
 <security-constraint>
    <web-resource-collection>
    <web-resource-name>nome da aplicaçao</web-resource-name>
     <url-pattern>/*</url-pattern> <!-- pode definir quais urls-->
     <http-method>GET</http-method> <!-- para o método GET-->
     <http-method>POST</http-method> <!-- para o método POST-->
```

```
    </web-resource-collection>
    <user-data-constraint>
      <transport-guarantee>CONFIDENTIAL</transport-guarantee> <!-- garante HTTPS-->
    </user-data-constraint>
</security-constraint>
```

Com estas configurações teríamos duas aplicações totalmente independentes com relação aos certificados e também utilizando somente o protocolo seguro.