Integration Guide

Apache Tomcat 6 with SSL
Linux 2.6

utimaco®

Imprint

copyright 2014	Utimaco IS GmbH Germanusstrasse 4 D-52080 Aachen Germany
phone	+49 (0)241 / 1696-200
fax	+49 (0)241 / 1696-199
web	http://hsm.utimaco.com
email	support-cs@utimaco.com
document version	1.2.0
date	June 2014
author	System Engineering HSM
document no.	SGCS_IG_TOMCATSSL

Contents

1 Introduction

The SafeGuard CryptoServeris the hardware security module (HSM) of Utimaco Safeware AG. Developed as a specialized physical-only processing unit it performs sensitive cryptographic functions and to ensure safe management of cryptographic keys and data. In a SafeGuard CryptoServersafety system, safety-relevant action is taken and security-related information is stored. It can be used as a universal, independent security component for heterogeneous computing systems.

2 Overview

The current use of inadequate transport layer security enables easy interception of communications of a web application or web server to a web browser from untrusted third parties. If an unencrypted transport layer - such as HTTP - is used for the transport of business critical information, it can be easily compromised or intercepted. In modern web applications SSL / TLS connections are used to secure the HTTP communications. In this case, the communication is encrypted with the help of symmetric cryptographic keys, which had previously been negotiated by an asymmetric cryptographic key exchange process. Foundations of these methods are always the public key of the participating parties or a proof of identity in the form of certificates. In addition to the storage of public keys cryptographic certificates include normally more information to help identify the owner or a trusted machine. Public internet web servers usually use certificates that are issued by a trusted certificate authority, to prove their trusted identity.

The Apache Tomcat is not a web server such as Apache HTTP or Microsoft IIS, but he is also used to for this activity in a limited extent. Sometimes there exist also mixed installation forms of Apache Web server and Apache Tomcat. Both software components have the ability to transport information using encryption. While Apache HTTP and Microsoft IIS have their strengths in delivering static information, the main focus of Apache Tomcat is to deliver dynamic and business-critical information. This document describes the essentially steps to establish an SSL connection with the Apache Tomcat on the basis of a SafeGuard CryptoServeras a certificate store. Usually the Apache Tomcat is installed with a file-based certificate in context SSL. Even if this certificate is protected by a password, potential attackers can for example simply copy the file to a portable medium or over a network and this can lead to a well taken seriously identity theft. A hardware security module (HSM) as the SafeGuard CryptoServer ensures that a logical and physical access is possible on the certificate only

to trusted individuals or applications. Copying of the private key and sensitive information and the actual certificate is not possible using the SafeGuard CryptoServer.

3 Requirements

Ensure for the further integration that you have experience with the Apache Tomcat and the administrative handling of the SafeGuard CryptoServer. The description is limited to the essential steps of the SSL integration and not on a full installation of Apache Tomcat. Consult the appropriate documentation for Apache Tomcat if you should require more assistance in this regard. Since the integration is based on the PKCS#11 interface of the SafeGuard CryptoServer, please use the CryptoServer PKCS#11 interface documentation for further questions to PKCS#11.

Software- and Hardware Requirements	
HSM Model	SafeGuard CryptoServer CS-Series/S-Series/S(e)-Series PCI(e) SafeGuard CryptoServer CS-Series/S-Series/S(e)-Series LAN SafeGuard CryptoServer Simulator
HSM Firmware	SafeGuard SecurityServer 3.00
Software	Apache Tomcat 6.0.33 (32 Bit) Java 1.6 (32 Bit) SafeGuard SecurityServer 2.10.2

3.1 Supported Systems

The integration of Apache Tomcat and SafeGuard CryptoServerwas successful tested with the following operating systems and SafeGuard CryptoServerfirmware:

Operating System	**SafeGuard SecurityServer Version**	**PCI Support**	**Ethernet Support**
Linux 2.6 (32 Bit)	3.00	Yes	Yes

4 Installation

Before proceeding with the integration, please ensure that you have installed a Java Runtime Environment (JRE), Apache Tomcat and the SafeGuard CryptoServersoftware on your Linux system already. The following assumptions for the naming of installation directories or file names are provided for

further description:

PKCS#11

- Configuration file for SafeGuard CryptoServerPKCS#11

  Path: `/etc/utimaco`

  Name: `cs2_pkcs11.ini`

- Environment variable for SafeGuard CryptoServerPKCS#11

  Name: `CS2_PKCS11_INI` Value: `/etc/utimaco/cs2_pkcs11.ini`

- SafeGuard CryptoServerPKCS#11 library

  Path: `/usr/lib/`

  Name: `libcs2_pkcs11.so`

PKCS#11 R2

- Configuration file for SafeGuard CryptoServerPKCS#11

  Path: `/etc/utimaco`

  Name: `cs_pkcs11_R2.cfg`

- Environment variable for SafeGuard CryptoServerPKCS#11

  Name: `CS_PKCS11_R2_CFG` Value: `/etc/utimaco/cs_pkcs11_R2.cfg`

- SafeGuard CryptoServerPKCS#11 library

  Path: `/usr/lib/`

  Name: `libcs_pkcs11_R2.so`

4.1 PKCS#11

A detailed description of the installation and additional capabilities of the PKCS#11 software component of the SafeGuard CryptoServeron a Linux operating system, refer to the documentation *SafeGuard CryptoServerPKCS#11 interface*. In so far not already done, copy the 32 bit version of the PKCS#11 library `libcs2_pkcs11.so` (`cs_pkcs11_R2.cfg`) from the SafeGuard SecurityServerproduct CD in the directory `/usr/lib/`. In addition, create a directory `/etc/utimaco` for the other configuration files. Copy the configuration file `cs2_pkcs11.ini` (`cs_pkcs11_R2.cfg`) from the SafeGuard SecurityServerproduct CD also there. This file can be used later addressing the SafeGuard CryptoServerhard-

ware. Create an environment variable `CS2_PKCS11_INI` (`CS_PKCS11_R2_CFG`) that contains a reference to the installation location (e.g. `/etc/utimaco/ cs2_pkcs11.ini` respectively `/etc/utimaco/cs_pkcs11_R2.cfg`) of the configuration file `cs2_pkcs11.ini` (`cs_pkcs11_R2.cfg`). For example, you can create the environment variable in a BASH environment with the next command:

PKCS#11

```
export CS2_PKCS11_INI=/etc/utimaco/cs2_pkcs11.ini
```

PKCS#11 R2

```
export CS_PKCS11_R2_CFG=/etc/utimaco/cs_pkcs11_R2.cfg
```

In order to have this environment variable available each time the computer starts, it is recommended to include the previous command in a startup script. For example the file `catalina.sh` is ideal to include the above command. It ensures that every time you start Apache Tomcat, the PKCS#11 environment variable is available. Change your in PKCS#11 configuration file the parameter device according to your SafeGuard CryptoServerdevice. You may limit the number of available PKCS#11 slots, which the SafeGuard CryptoServershall provide.

```
[Global]
Timeout = 5000
Logging = 0
Logpath = /tmp

[CryptoServer]
Device = 192.168.0.5
Timeout = 600000
AppTimeout = 1800
SlotCount = 1
```

In case of a protection by SSL/TLS for only one web application or website a restriction to one available PKCS#11 slot is very useful. Further slots needs to be created when a multi domain installation of the Apache Tomcat is made. Use the command line tool p11tool, what you should have previously copied from the SafeGuard SecurityServerproduct CD in the directory `/usr/bin/`, to create a new PKCS#11 slot in the SafeGuard CryptoServer.

PKCS#11

```
p11tool slot=0 Label=TomcatSSLDemo InitToken=654321
p11tool slot=0 LoginSO=654321 InitPin=123456
```

PKCS#11 R2

```
p11tool2 slot=0 Login=ADMIN,:cs2:cyb:USB Label=TomcatSSLDemo \\
         InitToken=654321
p11tool2 slot=0 LoginSO=654321 InitPin=123456
```

After the creation of PKCS#11 slot, the SunPKCS11 JCE provider needs to be registered and configured within the Java security environment. For this purpose a configuration file is created like `pkcs11.cfg` which provides the logical connection from the PKCS#11 provider over the PKCS#11 library for the actual SafeGuard CryptoServerhardware.

```
name = CryptoServer
library = /usr/lib/libcs2_pkcs11.so
#library = /usr/lib/libcs_pkcs11_R2.so (PKCS#11 R2)

slotListIndex = 0

attributes(*,*,CKK_EC) = {
  CKA_DERIVE = true
}

attributes(generate,*,CKK_GENERIC_SECRET) = {
  CKA_SENSITIVE = false
  CKA_EXTRACTABLE = true
}
```

In general, the ECA firmware module that provides elliptic curve functionality is installed in the SafeGuard CryptoServerby default. This lets you use the key exchange method based on elliptic curve with SSL/TLS. Table 2 shows the corresponding configuration for an elliptic curve support.

The SSL/TLS protocol automatically determines the most stringent cryptographic algorithms for the two party's web browser and web server. In some cases it may be desirable not to choose an elliptic curve algorithm. The following configuration shows how one you can disable SunPKCS11 on Elliptic

Curve JCE provider level. By disabling the Elliptic Curve algorithm the RSA key exchange is used as the next exchange process.

```
name = CryptoServer
library = /usr/lib/libcs2_pkcs11.so
#library = /usr/lib/libcs_pkcs11_R2.so (PKCS#11 R2)

slotListIndex = 0

disabledMechanisms = {
  CKM_EC_KEY_PAIR_GEN
}
```

The SunPKCS11 JCE provider name that is registered in the Java Security environment is composed of the prefix "SunPKCS11" and the parameter name of the above configuration file. For the previous examples the provider name is *SunPKCS11-CryptoServer*. In a multi domain installation the provider name gets a special meaning. Due to the unique provider name various web domains or their certificates can be distinguished in the Apache Tomcat.

In order to use the SunPKCS11 provider with Apache Tomcat the provider must be registered in beforehand in the Java runtime environment For this purpose an another provider entry in the java.security file is added as you can see in the example below.

```
...
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=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.8=sun.security.smartcardio.SunPCSC
security.provider.9=sun.security.pkcs11.SunPKCS11 /etc/utimaco/[...]
                                                  pkcs11.cfg
...
```

It is important when you register the new provider to specify the configuration file `pkcs11.cfg`. It contains the unique provider name for this entry and the logical connection to the SafeGuard CryptoServerPKCS#11 slot. A multiple provider registration of SunPKCS11 is possible. In this case it is important to ensure that the number which follows the prefix "security.provider." is unique in the list and each row contains a different PKCS#11 configuration file.

4.2 Certificate

Each SSL/TLS connection requires a certificate that can be created or acquired generally by a trusted certificate authority. Self signed certificates can also be used if it is sufficient if it concerns only the encryption of the connection and not to the trusted identity. The Java *keytool* has all the skills with to create a self-signed certificate or a certificate request for a certificate authority.

4.2.1 Self-signed Certificate

The *keytool* command line utility can be used to generate a self-signed certificate. The necessary cryptographic key pair for the required certificate as well as the certificate itself is created and stored directly by the SafeGuard CryptoServer.

```
keytool -genkey -keyalg RSA -keysize 2048 -keystore NONE \\
        -storetype PKCS11 -storepass 123456 \\
        -providername SunPKCS11-CryptoServer -alias tomcatdemo \\
        -dname "CN=www.utimaco.de, OU=System Engineering HSM, \\
                O=Utimaco Safeware,L=Aachen, S=NRW,C=DE"
```

Adjust parameters such as key size (keysize) and alias (alias) to suit your needs. Important when specifying the parameter value of the attribute dname CN either specify the ip address of the web server or the fully qualified domain name of the site (e.g. www.utimaco.de).

4.2.2 Certificate Authority Certificate

A self-signed certificate has the disadvantage that the trustworthiness mainly is assessed by the users of a web application. Modern web browsers are able show a warning if he does not trust a certificate. When examining a certificate a web browser is dependent on a chain of trusted certificate signing authorities. In case of a self-signed certificate, the browser does not have any information about the signing authority and therefore warns the user. Technically at this time an encryption with

this kind of certificate can be done and this is surely for testing fully sufficient. Apache Tomcat installations that need to be accessible on the internet and protect business-critical transactions require a certificate from a trusted certificate authority. Only this will ensure that the certificate and whose identity can be verified by the web browser. The following command creates the necessary cryptographic key pair for a certificate signing request. This certificate signing request is then sent to a certification authority to issue an official certificate.

```
keytool -v -genkeypair -keyalg RSA -keysize 2048 -alias tomcatdemo \\
        -keystore NONE -storetype PKCS11 -storepass 123456 \\
        -providername SunPKCS11-CryptoServer
```

The key pair is generated and stored in the SafeGuard CryptoServer. The private key from now on no longer leaves the SafeGuard CryptoServer. Only the public key is used for the certificate request.

```
keytool -v -certreq -alias tomcatdemo -file tomcatdemo.csr \\
        -keystore NONE -storetype PKCS11 -storepass 123456 \\
        -providername SunPKCS11-CryptoServer
```

By the previous command, the command line utility keytool stores the certificate request in a PKCS#10 format in the file `tomcatdemo.csr`. Upon receipt of the signed certificate of certification authority, it must still be imported into the SafeGuard CryptoServerto the already existing private key *tomcatdemo*.

4.3 Apache Tomcat

By default, the Apache Tomcat comes configured with a connector for an HTTP port 8080. This port enables reception of HTTP requests for one or several web applications using Apache Tomcat. Essentially the configuration file server.xml configures the Apache. This file is located in the subdirectory `conf/` of your Apache Tomcat installation. Even as a possible configuration included but commented out by default and therefore inactive, is a connector for the HTTPS protocol. The following excerpt shows the configuration of the server.xml HTTPS connector. Essential here are the parameters for the keystore like the PIN of the PKCS#11 slot with keystorePass, a *PKCS11* keystore type and the indication of the provider which addresses indirectly the previously configured PKCS#11 slot.

```
<Connector port="443" protocol="HTTP/1.1" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS"
```

```
keystore="NONE" keystorePass="123456" keystoreType="PKCS11"
keystoreProvider="SunPKCS11-CryptoServer" />
```

Any changes performed to the `server.xml` require a restart of Apache Tomcat. If the configuration was successful, then by default you can access Apache Tomcat locally with a web browser with *https://localhost*. On the first call the web browser views in case of a self-signed certificate, a warning that he is not familiar with the self-signed certificate. In general, the browser provides a way to verify the certificate by visual inspection and then take it to the list of trusted certificates.

4.4 Two-way mutual authentication with SSL

In two-way SSL authentication, the SSL client application asserts the identity of the SSL server application, and then the SSL server application asserts the identity of the SSL-client application. Two-way SSL authentication is also referred to as client authentication because the application acting as an SSL client presents its certificate to the SSL server after the SSL server authenticates itself to the SSL client. Key-pairs and certificates can be generated by using Openssl or Java Keytool. To achieve two way SSL authentication, one need to generate a self signed certificate for the tomcat application and the client (Browser for example). Client certificate should be in PKCS#12 format, since firefox accepts PKCS#12 file for certificate.

```
openssl pkcs12 -export -out clientkeystore.p12 -in clientcert.pem -inkey clientprivatekey.pe
```

Consider, clientcert.pem as an Opnessl self-signed certificate. 'clientAuth' needs to be set to true to enable two-way ssl authentication and 'keyAlias' needs to be set to server cert alias name in the keystore file. Here is relevant modification snapshot of server.xml file:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
               maxThreads="150" scheme="https" secure="true"
                           keyAlias="tomcat"
                           keystoreFile="C:\keys\tomcat.keystore" keystorePass="123456"
                           truststoreFile="C:\keys\tomcat.keystore" truststorePass="123456"
                           clientAuth="true" sslProtocol="TLS"/>
```

5 Further Information

This document forms a part of information and support which is provided by the Utimaco Safeware. Additional documentation can be found on the SafeGuard SecurityServerproduct CD in the documentation directory. All SafeGuard CryptoServerproduct documentation is also available from the Utimaco website: http://hsm.utimaco.com

Contact

Utimaco IS GmbH
Germanusstraße 4
D - 52080 Aachen
Germany

phone +49 241 1696 - 200
fax +49 241 1696 - 199

web http://hsm.utimaco.com
email support-cs@utimaco.com