이미 발급된 인증서를 이용해 SSL 구성하기

사전 준비

- OpenSSL
- Certificate (예: *.pem, *.crt, ...)
- Private key file & 암호
- java (/opt/java/bin/keytool 필요)

Java default certs에 추가하는 방법:

```
/opt/jdk1.8.0_144/bin/keytool -import -alias almdemo.curvc.com -keystore /opt/jdk1.8.0_144/jre/lib/security
/cacerts -file almdemo.curvc.com.cert
```

Step 1) 준비물 확인

private key와 암호를 확인한다.

```
# > openssl rsa -in privateKey.key
```

Step 2) PKCS12 keystore 생성

인증서와 private key를 이용해 PKC12 type keystore 생성

```
# > openssl pkcs12 -export -in Wildcard.curvc.com.crt -inkey privateKey.key -out keystore.p12 -name atlassian
```

위의 예에서 jira는 alias 이름이다.

```
openssl pkcs12 -export -in mycert.crt -inkey mykey.key
                       -out mycert.p12 -name tomcat -CAfile myCA.crt
                       -caname root -chain
```

Step 3) Java keystore 형식으로 변환

```
# > keytool -importkeystore -srckeystore keystore.p12 -srcstoretype pkcs12 -destkeystore atlassian.jks
```

확인 방법:

```
# > keytool -list -keystore atlassian.jks
 Enter keystore password:

 Keystore type: JKS
 Keystore provider: SUN

 Your keystore contains 2 entries

 1, Feb 4, 2018, PrivateKeyEntry,
 Certificate fingerprint (SHA1): 1D:A9:E4:44:0F:EC:EE:9B:B1:17:9B:B2:59:9E:CC:89:4E:3B:50:6A
 jira, Feb 4, 2018, PrivateKeyEntry,
 Certificate fingerprint (SHA1): 1D:A9:E4:44:0F:EC:EE:9B:B1:17:9B:B2:59:9E:CC:89:4E:3B:50:6A
```

Step 4) Atlassian Application SSL 설정

Update Tomcat with the KeyStore

1. Create a backup of `<JIRA_INSTALL>/conf/server.xml` before editing it.
2. Edit the HTTPS connector so that it has the parameters that point to the KeyStore:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
               maxHttpHeaderSize="8192" SSLEnabled="true"
               maxThreads="150" minSpareThreads="25"
               enableLookups="false" disableUploadTimeout="true"
               acceptCount="100" scheme="https" secure="true"
               sslEnabledProtocol="TLSv1.2"
               clientAuth="false" sslProtocol="TLSv1.2" useBodyEncodingForURI="true"
               keyAlias="jira" keystoreFile="<JIRA_HOME>/jira.jks" keystorePass="changeit" keystoreType="JKS"
/>
```

Ensure to put the appropriate path in place of `<JIRA_HOME>` and change the port as needed.

> If the organization doesn't support the latest TLS version, you can fallback to version 1.0. Change:
>
> `sslEnabledProtocol="TLSv1.2"`
>
> To:
>
> `sslEnabledProtocol="TLS"`

3. Edit the HTTP connector so that it redirects to the HTTPS connector:

```
<Connector acceptCount="100"connectionTimeout="20000"disableUploadTimeout="true"enableLookups="false"maxHt
tpHeaderSize="8192"maxThreads="150"minSpareThreads="25"port="8080"protocol="HTTP/1.1"redirectPort="<PORT_F
ROM_STEP_1>"useBodyEncodingForURI="true"/>
```

Ensure the `<PORT_FROM_STEP_1>` is change to the appropriate value. In this example it would be 8443.

4. Save the changes to `server.xml`.

JIRA

- https://wiki2.easycloudsolutions.com/display/JIRA/Running+JIRA+applications+over+SSL+or+HTTPS

1. If redirection to HTTPS will be used (this is recommended), edit the `<JIRA_INSTALL>/WEB-INF/web.xml` file and add the following section at the end of the file, before the closing `</web-app>`. In this example, all URLs except attachments are redirected from HTTP to HTTPS.

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>all-except-attachments</web-resource-name>
        <url-pattern>*.jsp</url-pattern>
        <url-pattern>*.jspa</url-pattern>
        <url-pattern>/browse/*</url-pattern>
        <url-pattern>/issues/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

2. Restart JIRA after you have saved your changes.

You can also redirect users from HTTP URLs to HTTPS URLs by choosing the 'HTTP & HTTPS' profile in the JIRA configuration tool. This will redirect all HTTP URLs to HTTPS URLs.

If you want to only redirect certain pages to HTTPS, you need to do this manually. To do this, select the 'HTTPS only' profile in the JIRA configuration tool and save the configuration, and then create an htaccess file on your web server that will manually redirect the HTTP URLs to the corresponding HTTPS URLs.

Confluence

1. If redirection to HTTPS will be used (this is recommended), edit the `<JIRA_INSTALL>/WEB-INF/web.xml` file and add the following section at the end of the file, before the closing `</web-app>`. In this example, all URLs except attachments are redirected from HTTP to HTTPS.

```
<web-app>

...

<security-constraint>
  <web-resource-collection>
    <web-resource-name>Restricted URLs</web-resource-name>
    <url-pattern>/</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>

</web-app>
```

2. Restart JIRA after you have saved your changes.

Tlps

Java Keytool Commands for Creating and Importing

These commands allow you to generate a new Java Keytool keystore file, create a CSR, and import certificates. Any root or intermediate certificates will need to be imported before importing the primary certificate for your domain.

- Generate a Java keystore and key pair
  keytool -genkey -alias mydomain -keyalg RSA -keystore keystore.jks -keysize 2048
- Generate a certificate signing request (CSR) for an existing Java keystore
  keytool -certreq -alias mydomain -keystore keystore.jks -file mydomain.csr
- Import a root or intermediate CA certificate to an existing Java keystore
  keytool -import -trustcacerts -alias root -file Thawte.crt -keystore keystore.jks
- Import a signed primary certificate to an existing Java keystore
  keytool -import -trustcacerts -alias mydomain -file mydomain.crt -keystore keystore.jks
- Generate a keystore and self-signed certificate (see How to Create a Self Signed Certificate using Java Keytoolfor more info)

  keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass password -validity 360 -keysize 2048

Java Keytool Commands for Checking

If you need to check the information within a certificate, or Java keystore, use these commands.

- Check a stand-alone certificate
  keytool -printcert -v -file mydomain.crt
- Check which certificates are in a Java keystore
  keytool -list -v -keystore keystore.jks
- Check a particular keystore entry using an alias

keytool -list -v -keystore keystore.jks -alias mydomain

Export a cert from one keystore to another as a trusted CA cert

1. Export the cert from the first keystore (it can be a cert/key pair or a trusted CA cert)
   keytool -exportcert -rfc -keystore keystore1.jks -storepass changeit -alias aliastoexport -file aliastoexport.pem

2. Import the cert into the second keystore
   keytool -importcert -file aliastoexport.pem -keystore keystore2.jks -storepass changeit -alias aliastoexport -trustcacerts -noprompt

Other Java Keytool Commands

- Delete a certificate from a Java Keytool keystore
  keytool -delete -alias mydomain -keystore keystore.jks
- Change a Java keystore password
  keytool -storepasswd -new new_storepass -keystore keystore.jks
- Export a certificate from a keystore
  keytool -export -alias mydomain -file mydomain.crt -keystore keystore.jks
- List Trusted CA Certs
  keytool -list -v -keystore $JAVA_HOME/jre/lib/security/cacerts
- Import New CA into Trusted Certs
  keytool -import -trustcacerts -file /path/to/ca/ca.pem -alias CA_ALIAS -keystore $JAVA_HOME/jre/lib/security/cacerts