IDP 3 - Step 1 - Tomcat TLS Non-Root

Back to the beginning of this guide

This Guide assumes

- A fresh, minimal (e.g. netinst.iso) install of **Debian 10** ("Buster") with no "tasks" except `openssh-server`
  - Ubuntu 18.04 LTS ("xenial") Server works the same as Debian 10 for the purpose of this guide
- Accessed via **SSH** or the **console** (no X11 required nor recommended),
- Correct server time configuration using NTP (e.g. using `systemd-timesyncd` or `ntpd`)
- Packet filters or firewall rules in place, e.g.:
  - With *outgoing* (ports TCP/80 and TCP/443) network access:
    - Port 80 for Debian APT updates, i.e., for downloading signed software packages
    - Port 80 and 443 for downloading signed eduID.at Metadata
    - Port 443 is also needed for downloads of the Shibboleth IDP software (you can copy that to the server yourself, of course)
    - The IDP will also need to connect to your LDAP Directory Servers for authentication and attribute lookup,
      - either on the standard port TCP/389 for LDAP(+STARTTLS),
      - or on port TCP/636 for LDAPS (which which no formal specification exists),
      - or maybe on the "global catalog" port of your Microsoft Active Directory (only if you need to access that).
    - For NTP you also need outgoing connectivity to the configured NTP servers (e.g. ACOnet's)
  - And *incoming* HTTPS access (port TCP/443 *only*, **no port 80 necessary** nor recommended),
    - also incoming port 22 for access *only from a management network*, if the server is managed via SSH,
- All commands in this guide are to be issued by user `root` (uid=0), and will make of `setuidgid` as needed to change to other accounts.
- The shell to use is `/bin/bash` (you can get fancy with fish/zsh/etc. after finishing the install/configuration)
- Use of systemd for service management using the amended service unit as described in this documentation

Table of contents for step 1

Install pre-requisites

Install required and recommended packages, replacing `vim` with your `$EDITOR` of choice (e.g. `emacs-nox` or `nano`, both of which also have syntax highlighting, which helps when editing XML files) and stop the automatically started tomcat again (until we've done more configuration):

```
apt install --no-install-recommends default-jdk-headless tomcat9 libservlet3.1-java \
  vim less openssl unzip curl expat multitail gnupg net-tools

systemctl stop tomcat9
```

Post-install fix-ups

Redirect requests to Tomcat's web root ("`/`") to a URL of your choice, e.g. your institution's home page, **replacing "www.example.edu" below**. The Shibboleth IDP application by default will run at `/idp`, allowing you to easily add and update *other* content outside of `/idp`, e.g. logos or CSS stylesheets without having them to integrate them with the "idp" context/application. The document root for that is in `/var/lib/tomcat9/webapps/ROOT/` and nothing in the Shibboleth IDP software (or during use of SAML) by default links to `/` of the server, so you can use that for locally hosted content without interfering with the IDP application. For example you will want to add a robots.txt file to prevent unnecessary scanning by well-behaving search bots.

```
rm /var/lib/tomcat9/webapps/ROOT/index.html
echo '<% response.sendRedirect("http://www.example.edu"); %>' > /var/lib/tomcat9/webapps/ROOT/index.jsp
echo -e "User-agent: *\nDisallow: /" > /var/lib/tomcat9/webapps/ROOT/robots.txt
```

Set `JAVA_HOME` for the current (and future) shell(s):

```
eval $(echo "export JAVA_HOME=/usr" | tee /etc/profile.d/java.sh)
```

Enable TLS/SSL

Create keypair and certificate chain

⚠ Do *not* use an existing wildcard certificate (if one is available that would also cover your IDP webserver) – just do the work as described below and create another certificate for your IDP. Under ACOnet's TCS agreement you can get *unlimited* globally valid commercial certificates. Alternatively (though not recommended for your eduID.at IDP) there's also https://letsencrypt.org/. So there should be no excuse to promiscuously share an existing TLS key pair across unrelated servers and services.

On the IDP server create an RSA private key and CSR for the web server's TLS certificate, e.g.:

```
openssl req -new -newkey rsa:2048 -nodes -out webserver.csr -keyout webserver.key -new
```

Request a TLS certificate based on the CSR generated, e.g. from the ACOnet TCS. In the resulting email with the certificate there's also the CA chain file to use (e.g. called `DigiCertCA.crt`). Copy the certificate and CA file to the server you're installing the IDP on (where the private key should already be, having been generated there).

Convert the TLS/SSL keypair into PKCS12

Create and note down a random password for your PKCS#12 keystore that will hold the above created TLS/SSL keypair plus the CA chain file:

```
openssl rand -hex 16
```

Convert the TLS certificate you recieved from your CA (i.e., from DigiCert, if using ACOnet TCS), the locally generated private key and the certificate chain file into one password-protected PKCS#12 keystore file. When being asked for an "export password" set the previously generated (and noted down) password. Below you'll also add that password to the Tomcat server configuration:

```
openssl pkcs12 -export -in webserver.crt -inkey webserver.key -certfile DigiCertCA.crt -name "webserver" -out
webserver.p12
```

Move the newly created keystore to its final location (we're chosing Tomcat's config directory) and set strict file system permissions on it:

```
mv webserver.p12 /etc/tomcat9/
chmod 640 /etc/tomcat9/webserver.p12
chgrp tomcat /etc/tomcat9/webserver.p12
```

Configure Tomcat Connector

Remove or comment out all other Connectors in `/etc/tomcat9/server.xml`, then add the two Connectors as per below, replacing `keystorePass` with the PKCS#12 keystore password generated earlier:

```
<!-- Localhost-only connector for IDP command line tools -->
<Connector address="127.0.0.1" port="80" />

<!-- https://tomcat.apache.org/tomcat-9.0-doc/ssl-howto.html -->
<Connector
  port="443"
  protocol="org.apache.coyote.http11.Http11NioProtocol"
  maxThreads="150"
  maxPostSize="100000"
  SSLEnabled="true"
  scheme="https"
  secure="true"
  clientAuth="false"
  sslProtocol="TLS"
  keystoreType="pkcs12"
  keystoreFile="/etc/tomcat9/webserver.p12"
  keystorePass="see above" />
```

Start Tomcat, check for listening ports, and access `https://webserver-fqdn/foo` which should result in an `HTTP Status 404` error (since /foo won't exist) **but** allows you to confirm a hopefully valid TLS/SSL webserver configuration:

```
systemctl restart tomcat9
netstat -lntp | fgrep java  # should show 443, and 80 only on the loopback interface
```

Verify TLS/SSL

Next validate the TLS/SSL configuration on the system itself with `openssl` (and quit again with ctrl-c or by typing `QUIT` into the prompt):

```
openssl s_client -CApath /etc/ssl/certs/ -connect webserver-fqdn:443
```

Look for "Certificate chain" in the output from that command, e.g.

```
openssl s_client -CApath /etc/ssl/certs/ -connect webserver-fqdn:443 2>&1 </dev/null | grep -A8 "^Certificate
chain"
```

and verify that it looks something like the "Certificate chain" presented below. The Subject of cert 0 will obviously differ, and depending on your choice of CA or certificate product ("SSL Plus", "Unified Communications", "EV" etc.) the CA may also be different. A correct chain (and therfore PKCS#12 keystore) for TLS usage should contain all the certificates up until but *excluding* the root CA certificate. I.e, in the example below the certificate with `CN=DigiCert Assured ID Root CA` is *not* included in the chain sent from the server:

```
Certificate chain
 0 s:/C=AT/ST=Vienna/L=Wien/O=ACOnet/CN=webserver-fqdn
   i:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
 1 s:/C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Assured ID Root CA
```

In case of errors check the output of "journalctl -u tomcat9 -ef".

Tune log file creation

By default Tomcat logs additional (and duplicate) events to `/var/log/tomcat9/catalina.$date.log` and `/var/log/tomcat9/localhost.$date.log`, which we don't care for. So let's create a backup copy of Tomcat's `logging.properties` and replace its content with the minumum needed to get an access log comparable to Apache `httpd` in `/var/log/tomcat9/access.log`. Tomcat's stdout/stderr will go to the systemd journal.

```
systemctl stop tomcat9
cp -a /etc/tomcat9/logging.properties /etc/tomcat9/logging.properties.orig

echo -n 'handlers = org.apache.juli.FileHandler, java.util.logging.ConsoleHandler
org.apache.juli.FileHandler.level = SEVERE
org.apache.juli.FileHandler.rotatable = false
org.apache.juli.FileHandler.directory = /dev
org.apache.juli.FileHandler.prefix = null
org.apache.juli.FileHandler.suffix =
java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].level = INFO
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].handlers = org.apache.juli.FileHandler
' > /etc/tomcat9/logging.properties
```

Then comment out or delete the whole `Valve` element at the end of your `/etc/tomcat9/server.xml`, and replace it with the following one:

```
<Valve className="org.apache.catalina.valves.AccessLogValve" maxLogMessageBufferSize="320"
       prefix="access" suffix=".log" renameOnRotate="true" pattern="combined" />
```

Then delete all Tomcat logs now and after a bit only `access.log` should have been generated:

```
rm -f /var/log/tomcat9/*
systemctl restart tomcat9
ls -l /var/log/tomcat9/
multitail /var/log/tomcat9/* -l 'journalctl -u tomcat9.service -f'  # exit with 'q'
systemctl stop tomcat9
```

Systemd service

Debian 10's Tomcat comes with an almost-usable systemd service that needs to be amended in order to (1) avoid the systemd-house-of-horror that's still all too common with Tomcat packaging, and (b) allow the IDP application to write logs and metadata to the filesystem. Since we're creating an override for the system-supplied systemd service unit anyway we'll also set the maximum memory usage there (to 3GB in the example "-Xmx3g" below) – adjust as needed (3-4GB should be sufficient), also leaving a bit of RAM for the OS. 😉 Not that you should be running anything else on an IDP server.

```
install -o root -g root -m 0755 -d /etc/systemd/system/tomcat9.service.d

cat <<'EOF' > /etc/systemd/system/tomcat9.service.d/override.conf
[Service]
Environment="JAVA_OPTS=-Djava.awt.headless=true -Xmx3g"
Environment="JSSE_OPTS=-Djdk.tls.ephemeralDHKeySize=2048"
ExecStart=
ExecStart=/usr/bin/java \
  $JAVA_OPTS $JSSE_OPTS \
  -classpath ${CATALINA_HOME}/bin/bootstrap.jar:${CATALINA_HOME}/bin/tomcat-juli.jar \
  -Dcatalina.base=${CATALINA_BASE} \
  -Dcatalina.home=${CATALINA_HOME} \
  -Djava.util.logging.config.file=${CATALINA_BASE}/conf/logging.properties \
  -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager \
  -Djava.io.tmpdir=${CATALINA_TMPDIR} \
  org.apache.catalina.startup.Bootstrap
ReadWritePaths=/opt/shibboleth-idp/logs/
ReadWritePaths=/opt/shibboleth-idp/metadata/
EOF
```

Activate the override with `systemctl daemon-reload`, maybe also verify with `systemd-delta | fgrep tomcat`

Next step: Installing the IDP software