IDP 5 - 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 12** ("Bookworm") with no "tasks" except `openssh-server`
  - ~~Ubuntu 22.04 LTS ("Jammy Jellyfish") Server works the same as Debian 12 for the purpose of this guide~~
- Accessed via **SSH** or the **console** (no GUI/X11/wayland required and certainly *not 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 cryptographically signed eduID.at Metadata documents.
    - Port 443 is also needed for downloads of the Shibboleth IDP software or additional modules
    - With local authentication the IDP will likely 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 it's necessary you access that).
    - For access to NTP services you also need outgoing connectivity to the configured NTP servers (e.g. ACOnet's)
  - And **incoming HTTPS access on port TCP/443 *only***. Noone needs to access your IDP by manually entering its URL, so no need to even have the IDP listening on TCP/80 publically, and therefore also no need for a redirect from TCP/80 to TCP/443.
    - Also, if the server is managed via SSH you'll need to allow access to port TCP/22, though *only from a secured management network*.
- All commands in this guide to be issued by user `root` (uid=0) so `sudo -s` first as needed.
- The shell used is `/bin/bash` (you can get fancy with fish/zsh/etc. after finishing the install/configuration if you want)
- Use of systemd for service management, in order to use the amended service unit contained in this documentation.

Table of contents for step 1

Install pre-requisites

Install required (and used, throughout this documentation) packages, possibly replacing `vim` with your `$EDITOR` of choice (e.g. `emacs-nox` or `micro`, both of which also support syntax highlighting, which helps when editing XML files) and stop the automatically started tomcat until we've completed more configuration performed further below:

```
apt install --no-install-recommends default-jdk-headless tomcat10 \
  vim less openssl curl expat multitail gnupg net-tools

systemctl stop tomcat10
```

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" in the command 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/tomcat10/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 avoid unnecessary scanning by well-behaved search bots.

```
rm /var/lib/tomcat10/webapps/ROOT/index.html
echo '<% response.sendRedirect("https://www.example.edu/"); %>' > /var/lib/tomcat10/webapps/ROOT/index.jsp
echo -e "User-agent: *\nDisallow: /" > /var/lib/tomcat10/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 *at no cost* to you. (Alternatively, though not recommended for your eduID.at IDP, there's always letsencrypt.)
So there should be no excuse to promiscuously share an existing TLS key pair across unrelated servers and services.

First, create and note down a random passphrase to use for protecting/encrypting the private key at rest. (Use this passphrase in all the steps below when asked for a key passphrase or import/export password.)

```
openssl rand -hex 16
```

On the IDP server create an RSA private key of at least 2048 bits size and the CSR for the web server's TLS certificate, supplying the necessary data (at least the subject) on the command line *or* by entering any data interactively when being prompted for it (when *not* adding `-subj` to the command):

```
openssl req -new -newkey rsa:2048 -out webserver.csr -keyout webserver.key -subj "/CN=WEBSERVER-FQDN"
```

When asked to "Enter pass phrase for webserver.key" provide the passphrase from the previous step.

Renewing an existing TLS certificate?

In case you're replacing an expiring TLS certificate where the matching private key is still considered to be secure and of sufficient strength (in 2024 CE for RSA keys that means a key size of at least 2048 bits) you'll want to keep using the *existing* private key (and PKCS#12 keystore passphrase) and generate any CSRs from that key.
To do that first extract the private key from your keystore (instead of generating a new one):

```
openssl pkcs12 -in /etc/tomcat10/webserver.p12 -nocerts | tail +5 > webserver.key
```

When asked to "Enter Import Password" supply the existing `certificateKeystorePassword` for the `port="443" Connector` from your `/etc/tomcat10/server.xml` configuration file.
When asked to "Enter PEM pass phrase" simply enter/paste that same passphrase again.
And yet again, when asked to "Verifying - Enter PEM pass phrase".

Then generate a CSR from the extracted private key, either by supplying the necessary data (at least the subject) on the command line *or* by entering any data interactively when being prompted for it (when *not* adding `-subj` to the command):

```
openssl req -new -key webserver.key -out webserver.csr -subj "/CN=WEBSERVER-FQDN"
```

When asked to "Enter pass phrase for webserver.key" again provide the passphrase from the previous steps.

The content of webserver.csr is what you provide to your CA then, e.g. via `cat webserver.csr` and pasting the result into the CA's web interface.

Equipped with the CSR you can now request a TLS certificate based from your CA, e.g. using the ACOnet TCS supplier. Once the certificate has been issued copy it to the IDP server as webserver.crt.
You'll also need to copy any intermediary Certificate Authority (CA) certificates to the IDP server.

In case of ACOnet TCS, Sectigo and an RSA OV certificate the *only* intermediate CA certificate you'll need is the one with the subject "`C = NL, O = GEANT Vereniging, CN = GEANT OV RSA CA 4`", referenced as file `GEANT-OV-RSA-CA-4.crt` below.

Convert the TLS/SSL keypair into PKCS12

Copy the private key, new TLS certificate and any intermediate CA certificates into a PKCS#12 keystore file:

```
openssl pkcs12 -export -in webserver.crt -inkey webserver.key -certfile GEANT-OV-RSA-CA-4.crt -name webserver -out webserver.p12
```

When asked to "Enter pass phrase for webserver.key" provide the passphrase generated earlier.
Again when asked to "Enter Export Password".
And yet again, when asked to "Verifying - Enter Export Password".

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

```
[[ -f /etc/tomcat10/webserver.p12 ]] && cp -a /etc/tomcat10/webserver.p12 /etc/tomcat10/webserver.p12.`date -u
+%Y%m%d`
mv webserver.p12 /etc/tomcat10/
chown root:tomcat /etc/tomcat10/webserver.p12
chmod 640 /etc/tomcat10/webserver.p12
```

Configure Tomcat Connector

Remove or comment out all other Connectors in `/etc/tomcat10/server.xml`, then add the two Connectors as per below, replacing `certificateKey storePassword` with the password generated earlier:

If you still need to support clients that can *only* speak TLS 1.0 or TLS 1.1 you will have to amend the `sslEnabledProtocols` parameter below!

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

<!-- https://tomcat.apache.org/tomcat-10.1-doc/ssl-howto.html -->
<!-- https://tomcat.apache.org/tomcat-10.1-doc/config/http.html#SSL_Support -->
<Connector
  port="443"
  protocol="org.apache.coyote.http11.Http11NioProtocol"
  maxThreads="150"
  maxPostSize="100000"
  SSLEnabled="true"
  scheme="https"
  secure="true">
  <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
  <SSLHostConfig>
    <Certificate type="RSA"
      protocols="TLSv1.2,TLSv1.3"
      ciphers="ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-
RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-
RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-POLY1305"
      certificateKeystoreType="PKCS12"
      certificateKeystoreFile="/etc/tomcat10/webserver.p12"
      certificateKeystorePassword="see sections above" />
    </SSLHostConfig>
</Connector>
```

The ciphers list above comes straight from the moz://a SSL Configuration Generator using their "Intermediate" configuration for Tomcat. Feel free to use other ciphers but be aware of the multitude and variety of clients / web browsers you may need to support in practice.

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 tomcat10
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 </dev/null
```

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 the CA or certificate chain 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=USERTrust RSA Certification Authority` is *not* included in the chain sent from the server (but must be known by the web browser):

```
Certificate chain
 0 s:C = AT, postalCode = 1010, ST = Wien, L = Wien, street = Universitaetsstrasse 7, O = ACOnet, CN = idp.aco.
net
   i:C = NL, O = GEANT Vereniging, CN = GEANT OV RSA CA 4
 1 s:C = NL, O = GEANT Vereniging, CN = GEANT OV RSA CA 4
   i:C = US, ST = New Jersey, L = Jersey City, O = The USERTRUST Network, CN = USERTrust RSA Certification
Authority
```

In case of errors check the output of "journalctl -u tomcat10 -ef".

If everything works fine and the certificate chain looks as expected you can remove the private key and certificate again (as both can be extracted from the PKCS#12 keystore if needed), keeping the CSR in file webserver.csr around for next time you need to renew that certificate (as long as you still consider the matching private key secure):

```
rm webserver.{key,crt}
```

Tune log file creation

IDP logs

You might prefer to have the IDP application write its logs to a more standard location in the file system, specifically one *outside the application's own directory* and on a file system where data usage is expected to grow dynamically (e.g. on /var). To do that simply set the `idp.logfiles` property in any of the property files read by the IDP, e.g. within `conf/idp.properties`:

```
idp.logfiles=/var/log/shibboleth
```

We also have to create that directory. And in order for the example commands in this documentation to work with *either* log directory location we'll remove the (still empty) log dir created by the IDP installer and replace it with a symlink to one we just created ourselfs:

```
install -o tomcat -g root -m 0750 -d /var/log/shibboleth/
cd /opt/shibboleth-idp/ && rmdir logs && ln -s /var/log/shibboleth logs
```

Tomcat logs

By default Tomcat logs everything multiple times, including to `/var/log/tomcat10/catalina.out` and `/var/log/tomcat10/localhost.*`, which we don't care for. So create a backup copy of Tomcat's `logging.properties` and replace its content with the minumum needed to get Tomcat's stdout /stderr to the console (which ends up in the systemd journal in our configuration). To prevent catalina.out from being created we deacticate it further below (in our "Systemd service" override) by setting the `CATALINA_OUT=/dev/null` environment variable for the java process.

```
systemctl stop tomcat10
cp -a /etc/tomcat10/logging.properties /etc/tomcat10/logging.properties.`date -u +%Y%m%d`

echo -n 'handlers = java.util.logging.ConsoleHandler
java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter = org.apache.juli.SystemdFormatter
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].level = INFO
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].handlers = java.util.logging.ConsoleHandler
' > /etc/tomcat10/logging.properties
```

Then comment out or delete the whole `Valve` element at the end of your `/etc/tomcat10/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" />
```

After deleting all Tomcat logs (only) `access.log` should be generated in Tomcat's log directory going forward:

```
rm -f /var/log/tomcat10/*
systemctl restart tomcat10
ls -l /var/log/tomcat10/
multitail /var/log/tomcat10/* -l 'journalctl -u tomcat10.service -f'  # exit with 'q'
systemctl stop tomcat10
```

If you're certain there's no catalina.log file being generated anymore also disable the default logrotate config snippet for it:

```
sed -i 's/^/#/' /etc/logrotate.d/tomcat10
```

Systemd service

Debian'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/Java packaging
2. Avoid slow startup times due to use of a blocking /dev/random (cf. Myths about urandom, also linked to from the Shib wiki).
3. Allow the IDP application to write logs and metadata to the filesystem as needed (by adding more `ReadWritePaths`)
4. Try avoiding the creation of catalina.out (we already have its content in journald using this configuration)

And since we're creating an override for the OS-supplied systemd service unit anyway we'll also set the maximum memory usage there ("`-Xmx3g`" in the example below, i.e., 3GB).
Adjust this as needed, but 3-4GB should be sufficient even for large metadata aggregates (as are common with Interfederation). Also leave 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/tomcat10.service.d

cat <<'EOF' > /etc/systemd/system/tomcat10.service.d/override.conf
[Service]
Environment="CATALINA_OUT=/dev/null"
Environment="JAVA_OPTS=-Djava.security.egd=file:/dev/urandom -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=/var/log/shibboleth/
ReadWritePaths=/opt/shibboleth-idp/logs/
ReadWritePaths=/opt/shibboleth-idp/metadata/
EOF
```

(If you've set "idp.logfiles=/var/log/shibboleth" via the IDP's property files as described in section "IDP logs" above you can remove the line `ReadWritePaths=/opt/shibboleth-idp/logs/`. The above example config just makes sure IDP logs can get written using either log location.)

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

⚠ Next step: Installing the IDP software