ARCWAY AG

Hinweise zu Sicherheitsmaßnahmen

Deutsch

16.12.2022

ARCWAY AG

Potsdamer Platz 10
10785 Berlin
GERMANY
+49 30 800 97 83 0 (Tel.)
+49 30 800 97 83 111 (Fax)
info@arcway.com
Vorstand: Karsten Wulfes
Vorsitzender des Aufsichtsrats: Uwe Barchmann
Dokument: AC.007.DE-1602*
16.12.2022
Dieses Dokument ist erstellt mit ARCWAY Cockpit.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

1 EINLEITUNG

Vorwort

Dieses Dokument bietet eine Anleitung für die Einrichtung einer sicheren Umgebung für Apache Tomcat 9. Diese Anleitung berücksichtigt Version 9.0 und beinhaltet keine Änderungen gegen von Apache bereitgestellte Updates.

> *Durch einige der beschriebenen Maßnahmen kann die Performance negativ beeinflusst werden.*

Zielgruppe

Dieses Dokument richtet sich an System-/Anwendungsadministratoren und Sicherheitsexperten die einen ARCWAY Cockpit Server auf einer Apache Tomcat 9 Umgebung zu installieren.

2 TOMCAT SSL KONFIGURATION

Die Datenübertragung zwischen Client und Server kann über SSL verschlüsselte HTTP Verbindungen erfolgen (HTTPS).

Die SSL Unterstützung ist nach der Standardinstallation eines ARCWAY Cockpit Servers nicht aktiviert. Bevor SSL-verschlüsselte HTTP Verbindungen genutzt werden können, müssen sowohl am Cockpit Server als auch an den Cockpit Clients Konfigurationsschritte durchgeführt werden, auf die im folgendem anhand der Standardinstallation eines ARCWAY Cockpit Servers eingegangen wird.

2.1 Server-seitige Konfiguration

Erstellung eines Zertifikats / asymmetrischen Schlüsselpaars

Voraussetzung für die Aktivierung der SSL-Unterstützung ist ein für die SSL-Verschlüsselung brauchbares asymmetrisches Schlüsselpaar.

Für das weitere Verständnis ist der Begriff Zertifikat und die Behandlung solcher Zertifikate durch Client Java VM's und den Tomcat Server wichtig. Daher folgt nun eine kurze Erläuterung dieser Zusammenhänge: Ein Zertifikat bezeichnet ein Datenpaket, das von einer Zertifizierungsstelle ausgestellt ist und dazu dient, den Eigentümer sowie weitere Eigenschaften eines veröffentlichten Schlüssels nachprüfbar zu machen. Jede Java VM Installation umfasst eine Datenbank mit Informationen über vertrauenswürdige Zertifizierungsstellen. Zertifikate, die von Zertifizierungsstellen ausgestellt wurden, über die keine Information in dieser Datenbank verzeichnet ist, können nicht auf Echtheit geprüft werden. Sofern der Tomcat Server über ein Zertifikat für den von ihm verwendeten öffentlichen SSL-Schlüssel verfügt, übermittelt er dieses Zertifikat im Rahmen des Aufbaus von SSL-Verbindungen zu seinen Clients.

Damit eine SSL-Verbindung erfolgreich aufgebaut werden kann, muss die Client Java VM den öffentlichen SSL-Schlüssel des Servers als vertrauenswürdig einstufen. Vertrauenswürdige Schlüssel müssen einer der folgenden Bedingungen genügen:

- Der Client Java VM liegt ein Zertifikat vor, dessen Echtheit Sie zu prüfen in der Lage ist und welches den öffentlichen Schlüssel des Servers einem Inhaber zuordnet.
- Der öffentliche Schlüssel des Servers wurde der Client VM zuvor als vertrauenswürdig bekannt gemacht

Aufgrund dieser Zusammenhänge wird bereits bei kleineren Installationen empfohlen, ein Zertifikat für den öffentlichen Schlüssel des Servers ausstellen zu lassen, welches von allen zum Betrieb von Clients in Betracht kommenden Java VMs überprüfbar ist. Dies ist einerseits aus Sicherheitsgründen empfehlenswert. Andererseits kann der Konfigurationsaufwand zur Einrichtung neuer Clients auf ein Minimum reduziert werden, wenn das passende Zertifikat auf dem Server hinterlegt wird. Insbesondere muss dann kein Aufwand betrieben werden, um den öffentlichen Schlüssel des Servers zu allen Client VMs zu *transportieren* und dort als vertrauenswürdig bekannt zu machen.

Die Erstellung eines für den vorliegenden Anwendungsfall brauchbaren Schlüsselpaars/Zertifikats beginnt praktisch immer damit, dass man mit Hilfe des *keytool* Kommandos aus dem Java SDK ein Schlüsselpaar erzeugt, welches nach Ausführung des Kommandos in einer Archivdatei gespeichert ist, die üblicherweise als **Keystore** bezeichnet wird.

Beispiel:

```
keytool -genkey -alias tomcat -keyalg RSA -keysize 2048 -keystore
tomcat.keystore
```

Das Ergebnis des obigen Befehls ist eine Datei mit dem Namen **tomcat.keystore** welches ein Schlüsselpaar der gewünschten Art enthält.

Falls auf die Nutzung eines Zertifikats verzichtet wird, enthält der in diesem ersten Schritt erstellte **tomcat.keystore** bereits jetzt alle für die Nutzung durch den Server relevanten Inhalte. Wenn ein Zertifikat genutzt werden soll, sind folgende Schritte zusätzlich erforderlich:

1. *Erstellung eines Signing Requests*. Um ein Zertifikat zu beantragen, müssen die dafür relevanten Daten an die Zertifizierungsstelle übermittelt werden. Die Übermittlung dieser Daten geschieht üblicherweise durch Versand

einer **Signing Request**-Datei, die mit Hilfe des *keytool* Kommandos erzeugt werden kann:

```
keytool -certreq -keystore tomcat.keystore -alias tomcat -file tomcat.csr
```

2. *Übersenden der Signing Request-Datei an die Zertifizierungsstelle*. Nach erfolgreicher Prüfung des Zertifizierungsantrags bekommt man von der Zertifizierungsstelle das fertige Zertifikat in Form einer **.crt** Datei.

3. *Zertifikat importieren / auf dem Server hinterlegen*. Um das Zertifikat *auf dem Server zu hinterlegen*, muss man das Zertifikat in den zuvor erstellten **tomcat.keystore** importieren. Danach wird es dem Server zusammen mit dem darin enthaltenen Schlüsselpaar bekannt gemacht. Der Import des Zertifikats in den Keystore kann durch ein Kommando der folgenden Form durchgeführt werden:

```
keytool -import -trustcacerts -file tomcat.crt -alias tomcat -keystore
tomcat.keystore
```

HTTPS- / SSL-Unterstützung aktivieren

Die Aktivierung der SSL-Unterstützung in Tomcat geschieht, indem die in der **server.xml** bereits als Kommentar vorhandene *Connector*-Definition aktiviert und die zugehörigen Parameterwerte angepasst werden. Die folgende *Connector*-Definition führt dazu, dass Tomcat auf dem Port *8443* eintreffende HTTPS Verbindungsanfragen bearbeitet und zur SSL-Verschlüsselung das erste in der Keystoredatei **c:\tomcat.keystore** enthaltene Schlüsselpaar/Zertifikat verwendet. Als Entschlüsselungspasswort für die Keystoredatei wird das Passwort *changeit* verwendet.

```
<-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
        port="8443"
        maxHttpHeaderSize="8192"
        maxThreads="150"
        minSpareThreads="25"
        maxSpareThreads="75"
        enableLookups="false"
        disableUploadTimeout="true"
        acceptCount="100"
        scheme="https"
        secure="true"
```

```
        clientAuth="false"
        sslProtocol="TLS"
        keystoreFile="c:\tomcat.keystore"
        keystorePass="changeit"
    />
```

Damit ist die SSL Unterstützung bereits aktiviert. Weitere Hinweise zur Konfiguration der SSL Unterstützung finden Sie auch in der Online Dokumentation von Apache Tomcat *SSL Configuration HOW-TO*:

http://tomcat.apache.org/tomcat-9.0-doc/ssl-howto.html

> *Die Hinweise der offiziell anerkannten Zertifizierungsstellen sind teilweise etwas verwirrend und möglicherweise gibt es auch tatsächlich andere Möglichkeiten zur Erzeugung des Schlüsselpaars. Wenn eine andere Vorgehensweise zur Erstellung des Zertifikats vorgeschlagen wird, sollte man sich vergewissern, ob das auf die beschriebene Weise erstellte Zertifikat tatsächlich für den vorliegenden Anwendungsfall geeignet ist.*

Zugriff auf HTTP unterbinden

Das Einschalten der HTTPS/SSL-Unterstützung bewirkt, dass zusätzlich zur standardmäßig eingeschalteten Unterstützung für HTTP anschließend auch HTTPS als Kommunikationsprotokoll verwendet werden kann. Es stellt für den Cockpit Server kein Problem dar, wenn beide Verbindungsarten gleichzeitig zugelassen und genutzt werden. Falls jedoch nur HTTPS-Verbindungen erwünscht sind, empfiehlt es sich die Unterstützung für HTTP explizit auszuschalten.

Um die HTTP-Unterstützung vollständig auszuschalten, kann man die Definition des HTTPS-„Connector“ in der „server.xml“ löschen oder auskommentieren. Von dieser Möglichkeit darf in der Standardinstallation von ARCWAY Cockpit aber kein Gebrauch gemacht werden, da zusätzlich zur eigentlichen Cockpit-Server Web-Applikation auch die Cockpit-Lizenz-Server Web-Applikation auf dem Tomcat Servlet Container betrieben wird. Würde man die HTTP Unterstützung vollständig abschalten, wäre keine Kommunikation zwischen Cockpit-Server und Cockpit-Lizenz-Server mehr möglich und die auf dem Cockpit-Lizenz-Server eventuell vorhandenen „Concurrent Lizenzen“ wären nicht länger nutzbar. Deshalb wird

empfohlen, die HTTP-Unterstützung nicht gänzlich abzuschalten, sondern den Zugriff über HTTP nur einzuschränken.

In der Vergangenheit hat es sich in diesem Zusammenhang als zweckmäßig erwiesen, den Zugriff per HTTP so zu beschränken, dass HTTP-Verbindungen nur am Loopback-Interface zugelassen werden. Da das Loopback-Interface nur vom lokalen Rechner aus erreichbar ist, sind Zugriffe von anderen Rechnern anschließend unmöglich. Um diese Beschränkung zu erreichen, muss der „Connector“-Definition des HTTP-Connectors der Parameter address="127.0.0.1" hinzugefügt werden. Die „Connector“-Definition des HTTP-Connectors könnte dann beispielsweise so aussehen:

```
<Connector
       port="8080"
       address="127.0.0.1"
       redirectPort="8443"
       minSpareThreads="25"
       connectionTimeout="20000"
       maxSpareThreads="75"
       maxThreads="150">
    </Connector>
```

2.2 Client-seitige Konfiguration

Client-seitig ergeben sich durch die Nutzung von HTTPS- / SSL-Verschlüsselung anstelle von HTTP folgende Unterschiede:

URLs

Die Server URLs beginnen nun mit „https“, und typischerweise erfolgt der Zugriff auf den Server auch über einen anderen Port (8443 statt 8080). Hier ein Beispiel für eine HTTPS-URL:

https://192.168.35.12:8443/CockpitServer/

Öffentlichen Server-Schlüssel der Client VM bekannt machen

Bevor eine Verbindung über HTTPS aufgebaut werden kann, muss die Client Java VM den öffentlichen SSL-Schlüssel des Servers als vertrauenswürdig einstufen. Dies erfolgt anhand der eingangs erwähnter Kriterien.

Daher ist die Durchführung der in diesem Abschnitt beschriebenen Konfigurationsschritte nur erforderlich, wenn auf dem Server kein Zertifikat hinterlegt wurde.

Der öffentliche Schlüssel des Servers wird dem Client durch Angabe einer „Trust Store“-Datei, welche diesen Schlüssel enthält, und eines Passwortes zum Entschlüsseln des Inhaltes dieser Datei bekannt gemacht. Der Speicherort der „Trust Store“-Datei und das zugehörige Passwort können im Dialog für **Benutzervorgaben** unter dem Punkt **ARCWAY Cockpit** eingestellt werden. Dieser Dialog kann über den Menüpunkt **Fenster » Benutzervorgaben** geöffnet werden:

Eine solche „Trust Store“-Datei hat das gleiche Format, wie die bei der Einrichtung des Servers angefertigte „tomcat.keystore” Datei. Tatsächlich könnte man die „tomcat.keystore” Datei und das zugehörige Passwort sogar direkt als „Trust Store“-Datei und „Trust Store“-Passwort verwenden.

Dies wird jedoch nicht empfohlen, da diese Datei neben dem öffentlichen Schlüssel auch den dazu passenden privaten Schlüssel des Servers enthält, der aus Sicherheitsgründen geheim gehalten werden sollte.

Um eine „Trust Store“-Datei zu erstellen, welche den öffentlichen Schlüssel des Servers nicht jedoch den privaten Schlüssel des Servers enthält, können Sie wie folgt vorgehen:

4. Extrahieren Sie den in der Datei „tomcat.keystore” enthaltenen öffentlichen Schlüssel mit Hilfe des folgenden Befehls:

```
keytool -export -alias tomcat -file tomcat.cert -keystore tomcat.keystore
```

5. Erzeugen Sie einen neuen Keystore, in den der öffentliche Schlüssel des Servers anschließend „hineingespeichert“ werden kann. Leider kann man mit den „keytool“-Kommandos keine leeren Keystores erzeugen. Daher muss zunächst ein neuer, gefüllter Keystore erzeugt werden. Aus diesem Keystore muss anschließend das bei der Erzeugung eingespeicherte Schlüsselpaar gelöscht werden. Hier die passenden Kommandos:

```
keytool -genkey -alias ungenutzt -keystore client.truststore
keytool -delete -alias ungenutzt -keystore client.truststore
```

6. Dann kann der öffentliche Schlüssel des Servers in den so vorbereiteten Keystore importiert werden:

```
keytool -import -keystore client.truststore -file tomcat.cert
```

Das unten dargestellte Bildschirmfoto zeigt ein Protokoll, das bei der beispielhaften Ausführung der oben genannten Schritte aufgezeichnet wurde:

Im Rahmen der protokollierten Ausführung der Schritte wurde eine „Trust Store"-Datei mit dem Namen „client.truststore" erzeugt, welche mit dem Passwort „michelangelo" geschützt ist.

Proxy Konfiguration

Falls der Zugriff nicht direkt, sondern über einen Proxy erfolgen soll, müssen zusätzlich zu einem bereits eingerichteten HTTP-Proxy auch die SSL-Proxy-Einstellungen konfiguriert werden.

Die Proxy-Einstellungen sind im Dialog **Benutzervorgaben** unter dem Punkt **Allgemein » Network Connections** vorzunehmen. Dieser Dialog kann über den Menüpunkt **Fenster » Benutzervorgaben** geöffnet werden.

3 ENTFERNEN FREMDER RESSOURCEN

Beschreibung

Die Installation von Apache Tomcat 9 kann Beispiele für Anwendungen, Dokumentationen und andere Verzeichnisse enthalten, die nicht für die Produktion verwendet werden sollten.

Standardmäßig sind die Unterverzeichnisse **docs, examples, ROOT, manager und host-manager** in **TOMCAT_HOME/webapps** vorhanden.

Handlungsempfehlung

Sie sollten alle Standardanwendungen, die Sie nicht nutzen aus dem Verzeichnis löschen.

Wenn Sie den **manager und host-manager** nutzen, brauchen Sie diese Anwendungen nicht entfernen, dieses Dokument wird auf eventuelle Sicherheitsaspekte bei diesen Anwendungen nicht weiter eingehen.

Referenz

Default_web_applications/General

4 SERVER INFORMATIONSLÜCKEN

Viele Informationen machen es einem Angreifer leichter zu bestimmen welche Schwachstellen es auf einem Server gibt.

Daher ist es zu empfehlen einige Informationen einzuschränken.

Beachten Sie, wenn sie regelmäßig ihren Tomcat Server patchen, müssen folgenden Schritte aus diesem Kapitel immer wieder durchgeführt werden.

4.1 Einschränkung von Informationen über den Tomcat

Das Attribut *server.info* enthält Informationen über die Version und den Build des Tomcat Servers.

Dieser Wert wird übermittelt, wenn sich Clients mit dem Tomcat Server verbinden.

> *Durch Anpassungen der Datei **ServerInfo.properties** kann die Komplexität für Angreifer erhöht werden, zu bestimmen welche Schwachstellen den Server betreffen.*

Wenn Sie das Attribut *server.info* ändern möchten gehen Sie wie folgt vor.

1. Extrahieren der **catalina.jar**

Das Attribut *server.info* ist Teil der **catalina.jar**.

Um dieses Attribut anzupassen ist es notwendig die **catalina.jar** zu dekompilieren und anschließend wieder zu kompilieren.

```
$cd $CATALINA_HOME/lib
$jar xf catalina.jar org/apache/catalina/util/ServerInfo.properties
```

2. Zum erstellten util Ordner navigieren

/org/apache/cataline/util

3. **ServerInfo.properties** im Editor bearbeiten

Der Standardwert für das Attribut *server.info* ist **Apache Tomcat /ServerVersion**

Ändern sie diesen Wert wie folgt und speichern sie anschließend ihre Änderungen:

```
server.info=Arcway Cockpit Server
```

4. Anpassung des Attributs *server.build*

Der Standardwert für das *server.build* Attribut ist Datum und Zeit des builds.

> *Beispiel: Jul 8 2008 11:40:35*

```
server.build=Jan 1 2000 01:01:01
```

5. **catalina.jar** mit den modifizierten Attributen der *server.info* updaten

```
$cd $CATALINA_HOME/lib
$jar uf catalina.jar org/apache/catalina/util/ServerInfo.properties
```

4.2 Server Header modifizieren

Das Server-Attribut steuert den Wert des HTTP-Headers des Servers.

Der Standardwert dieses Headers ist für Tomcat 4.1.x bis 8.0.x *Apache-Coyote/1.1*.

Ab 8.5.x ist dieser Header nicht mehr voreingestellt. Dieser Header kann sowohl Clients als auch Angreifern Informationen zur Art des Servers zu Verfügung stellen.

In der Datei **$CATALINA_HOME/conf/server.xml** im Abschnitt [[STYLE#EMPHASIS|<Connector />]] folgende Änderung durchführen:

```
<Connector port="8443" server="I am a Cockpit Server"
redirectionPort="8080" />
```

4.3 Tomcat Shutdown Port

Standardmäßig lauscht der Tomcat Server auf dem TCP Port **8005** und nimmt Shutdown request entgegen.

Eine Verbindung mit diesem Port und der Übertragung des Befehls **SHUTDOWN** (in der Standardkonfiguration), führt dazu das der Tomcat Server heruntergefahren wird.

Es wird empfohlen den Standardwert **SHUTDOWN** durch eine beliebige Zeichenfolge auszutauschen oder besser noch den Port für die SHUTDOWN Sequenz zu deaktivieren.

In der Datei **$CATALINA_HOME/conf/server.xml** im Abschnitt *<Connector />* folgende Änderung durchführen:

```
<Connector port="8443" server="I am a Cockpit Server"
redirectionPort="8080" />
```

5 LINUX

5.1 Tomcat Benutzer

Aus Sicherheitsgründen sollte der Tomcat Server nicht unter dem Benutzer **root** ausgeführt werden.

Dazu sollte ein neuer Systembenutzer und eine neue Gruppe mit dem Home-Verzeichnis des Tomcats **TOMCAT_HOME** erstellt werden, die den Tomcat-Dienst ausführen wird.

```
sudo useradd -r -m -U -d /TOMCAT_HOME -s /bin/false tomcat
```

> TOMCAT_HOME *im Codeblock muss durch das Verzeichnis des Tomcats ersetzt werden*

5.2 Verzeichnis- und Dateirechte

Wie wir im vorigen Abschnitt erwähnt haben, wird Tomcat unter dem Tomcat-Benutzer laufen. Dieser Benutzer muss Zugriff auf das Verzeichnis **TOMCAT_HOME** haben.

> ***TOMCAT_HOME*** *im Codeblock muss durch das Verzeichnis des Tomcats ersetzt werden.*

Der folgende Befehl ändert den Verzeichnisbesitz auf den Benutzer und die Gruppe tomcat:

```
sudo chown -RH tomcat: /TOMCAT_HOME
```

Die Skripte im bin-Verzeichnis müssen das executable-Flag haben:

```
sudo sh -c 'chmod +x /TOMCAT_HOME/bin/*.sh'
```

5.3 Tomcat SYSTEMD Deamon

Um Tomcat als Dienst auszuführen, müssen Sie eine neue Datei für **systemd** erstellen.

1. Öffnen Sie Ihren Texteditor und erstellen Sie eine Datei namens **tomcat.service** im Verzeichnis **/etc/systemd/system/**:

```
sudo nano /etc/systemd/system/tomcat.service
```

2. Fügen Sie die folgende Konfiguration ein:

```
[Unit]
Description=Tomcat 9 - Arcway Cockpit Server
After=network.target
[Service]
Type=forking
User=tomcat
Group=tomcat
Environment="JAVA_HOME=/usr/lib/jvm/default-java"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom -
Djava.awt.headless=true"
Environment="CATALINA_BASE=/TOMCAT_HOME"
Environment="CATALINA_HOME=/TOMCAT_HOME"
Environment="CATALINA_PID=/TOMCAT_HOME/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"
ExecStart=/TOMCAT_HOME/bin/startup.sh
ExecStop=/TOMCAT_HOME/bin/shutdown.sh
[Install]
WantedBy=multi-user.target
```

> ***TOMCAT_HOME*** *im Codeblock muss durch das Verzeichnis des Tomcats ersetzt werden.*
>
> *Ändern Sie den Wert von* ***JAVA_HOME****, wenn der Pfad zu Ihrer Java-Installation anders ist.*

3. Speichern und schließen Sie die Datei und teilen Sie dem System mit, dass eine neue Systemd Datei vorhanden ist:

```
sudo systemctl daemon-reload
```

4. Starten Sie den Tomcat-Dienst durch Ausführen von:

```
sudo systemctl start tomcat
```

5. Überprüfen Sie den Status des Dienstes mit dem folgenden Befehl:

```
sudo systemctl status tomcat
```

Prüfen sie die Ausgabe auf Fehler und das der Tomcat erreichbar ist.

6. Richten sie den Autostart des Server ein:

```
sudo systemctl enable tomcat
```