SSL für Java-basierte Dienste: Java Keystore

Setups mit Java-basierten Diensten sehen häufig ähnlich aus:

- Java-Dienst z.B. basierend auf Tomcat (oder Jetty), lauschender Port auf > 1024 (z.B. 8443)
- Apache Reverse Proxy, der stellvertretend auf Port 80 und 443 lauscht
    - das Zertifikat im Apache ist ein gültiges Zertifikat (z.B. aus der DFN PKI)
- die Verbindung zwischen Reverse Proxy und Tomcat ist auch SSL gesichert (weil die Applikationen meistens auch SSL erzwingen)
    - das Zertifikat im Tomcat kann selbstsigniert sein, sollte aber *nicht abgelaufen** sein

Keystore

Die SSL-Zertifikate unter Java werden in einem besonderen Keystore verschlüsselt gespeichert. Darin liegen sowohl die Private Keys als auch die Zertifikate.

- wenn man nichts anderes angibt, wird ~/.keystore verwendet
- der Keystore hat ein äußeres Passwort (Keystore password) und ein inneres (Truststore password)
    - default für tomcat: „changeit“
- Zertifikate werden mit einem Alias gespeichert
    - default für tomcat: tomcat

1. Zertifikat

Siehe
https://www.sslshopper.com/article-how-to-create-a-self-signed-certificate-using-java-keytool.html

```
keytool -genkey -keyalg RSA -alias tomcat -storepass <neues Passwort> -
validity 730 -keysize 2048
```

Zertifikat erneuern

```
keytool -selfcert -alias tomcat -validity 1825
```

Zertifikat anzeigen

```
keytool -list -v
```

Keytool Doku: http://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html

Jetty

Jetty speichert mittlerweile die Keystore Passwörter obfuscated in der config, siehe http://wiki.eclipse.org/Jetty/Howto/Secure_Passwords.

Python-Skript zum De-obfuscaten:

```
# Jetty Deobfuscation Tool

import sys

def d_jetty(ct):

    pt = ""
    b = bytearray(len(ct)/4)
    i=0

    for x in b:

        t = ct[i:i+4]
        i0 = int(t,36)
        i1 = i0 / 256
        i2 = i0 % 256
        x = (i1+i2-254)/2
        pt+=chr(x)
        i+=4

    return pt



if (len(sys.argv) == 2):
    raw_ct = sys.argv[1]
else:
    print "Jetty Deobfuscation Tool v1.0"
    print "./jdt <string>"
    exit(0)

print d_jetty(raw_ct)
```

Quelle: http://stackoverflow.com/questions/8883951/passwords-in-ssl-with-jetty-tutorial