WPP	Praktikum IT-Sicherheit	HBN/NMN
WS 2013	Aufgabe 4 – Zertifikate	10.12./17.12./7.1.

1. Zertifikate erstellen

Erstellen Sie mit Hilfe des Werkzeuges **`openssl`** eine eigene Root-CA, die aufgrund von Zertifikatsanträgen weitere Zertifikate ausstellen kann.

1.1 Bauen Sie sich für Ihre eigene Root-CA eine geeignete Verzeichnisstruktur auf.
1.2 Erstellen Sie sich für Ihre Root-CA eine Konfigurationsdatei (z.B. auf Basis der Beispieldatei openssl.cnf).
1.3 Erstellen Sie Ihr Root-CA-Schlüsselpaar und ein selbst-signiertes Root-CA-Zertifikat.
1.4 Erstellen Sie ein Schlüsselpaar und Serverzertifikat, das von Ihrer Root-CA signiert ist.
1.5 Exportieren Sie das Serverzertifikat und den zugehörigen privaten Schlüssel in eine Datei im PKCS#12-Format.

Tipps:

- **`openssl`** ist in Linux-Systemen meist vorinstalliert, den Code für Windows finden Sie unter http://www.openssl.org/related/binaries.html
- Eine gute Anleitung finden Sie im Buch „Network Security with OpenSSL“ (hier kostenlos erhältlich): http://it-ebooks.info/book/263/

2. Zertifikate testen

2.1 Modifizieren Sie den Webserver-Vorgabecode (→ Pub) so, dass Client-Anfragen nur noch per SSL-Socket auf Port 443 bedient werden (https). Verwenden Sie für den Webserver das in Aufgabe 1 erstellte Serverzertifikat.

2.2 Konfigurieren Sie ihren Browser so, dass er alle Zertifikate, die von Ihrer eigenen Root-CA aus Aufgabe 1 signiert wurden, ohne Rückfrage akzeptiert.

2.3 Starten Sie Ihren modifizierten Webserver und zeigen Sie, dass ihr Browser eine fehlerfreie SSL-Verbindung (https) mit ihm aufbauen kann. Die mitgelieferten Test-Webinhalte müssen korrekt angezeigt werden.

Tipps

- Das unter 1.5 gespeicherte Serverzertifikat müssen Sie inkl. zugehörigem privatem Schlüssel in einen KeyStore ihrer Java-Umgebung laden. Zur Erzeugung eines geeigneten KeyStore können Sie das Werkzeug **`keytool`** verwenden, welches Bestandteil des JDK's ist. Hinweis: Eine PKCS#12-Datei ist bereits ein eigener KeyStore (allerdings nicht im JAVA-Standard-Format), kann daher aber vom **`keytool`** als KeyStore importiert werden, so dass anschließend eine gültige JAVA-KeyStore-Datei zur Verfügung steht!
- Sie müssen in Ihrem Java-Code folgende Schritte ausführen, bevor Sie ein entsprechendes SSLServerSocket erzeugen können (die default-Werte reichen hier nicht aus):
  - **`KeyStore`**-Objekt erzeugen und Informationen aus KeyStore-Datei (s.o.) laden
  - **`KeyManagerFactory`**-Objekt erzeugen und initialisieren
  - **`SSLContext`**-Objekt erzeugen und initialisieren (**`TrustManager`**- und **`SecureRandom`**-Argument können beide null sein, auf das **`KeyManager`**-Array der **`KeyFactory`** kommt bei es der Initialisierung an!)
  - Für den SSLContext ein **`SSLServerSocketFactory`**-Objekt erzeugen, was dann in der Lage ist, konkrete SSLServerSockets zu liefern.
- Verwenden Sie u.a. die JAVA-Packages **`java.security`** und **`javax.net.ssl`** !

WPP	Praktikum IT-Sicherheit	HBN/NMN
WS 2013	Aufgabe 4 – Zertifikate	10.12./17.12./7.1.

Anforderungen an das Praktikums-Protokoll

Zur Abgabe fertigen Sie bitte ein Protokoll an, in dem für Aufgabe 1 mindestens die Aufrufe in der Kommandozeile und Verweise auf die verwendeten Konfigurationsdateien sowie die erzeugten Zertifikats- und Schlüsseldateien enthalten sind. Aufgabe 2 dokumentieren Sie bitte mit Screenshots und kommentiertem Webserver-Code.

Viel Spaß!