Cookbook- SSL Zertifikat Installation

Dokumententitel	Cookbook- SSL Zertifikat Installation			
Projekt	Servolution Ldap Logon Client			
Thema	Ldap über Secure Socket Layer			
Versionsdatum	30.08.2002			
Druckdatum				
Dateiname	CookbookLdapSSLLogonClient.doc			
Revision	1.0			
Schlüsselworte	SSL, Zertifikate, X509, LDAP, Public Key, Private Key			
Version	Seiten	Datum	Bemerkung	Bearbeitet von:
0.1		31.05.2002	Erstellung	Johann Brandstetter
1.0		30.08.2002	Update	Stefan Pfingstner

1 Einleitung

Der Comtarsia Logon Client unterstützt ab Release 3.0 auch LDAP. Um die Vertraulichkeit der übertragenen Daten (Benutzerpasswörter, Benuterberechtigungsdaten, usw.) zwischen Logon Client und LDAP Server zu gewährleisten, ist es möglich SSL Verschlüsselung einzusetzen.
SSL (Secure Socket Layer) wurde ursprünglich von Netscape entwickelt. Inzwischen unterstützen viele namhafte Softwarehersteller diese Protokoll für Datenverschlüsselung und für elektronische Unterschriften.

SSL beruht auf asymmetrische Verschlüsselung (Private Key / Public Key) und der Verwendung von X.509 Zertifikaten am Server bzw am Client.
Dabei sind folgende Kombinationen möglich:

a) Am Server wird ein so genanntes Self Signed Certificate verwendet, der Client verwendet kein Zertifikat.
b) Der Server verfügt über ein CA (Certificate Authority) Signed Certificate, dann ist am Client zumindest das CA Zertifikat erforderlich um die Echtheit des Server Zertifikates zu Überprüfen. (Server Authentication)
c) Der Server verfügt über ein CA signed Certificate, der Client verwendet ein Self Signed Certificate und benötigt zusätzlich das CA Zertifikat (zu Prüfung des Server Zertifikates).
d) Sowohl Client als auch Server verfügen über CA Signed Certificates, dann muss dem Client ebenfalls das CA Zertifikat zur Verfügung gestellt werden, damit es dem Server möglich ist, die Echtheit des Client Zertifikates zu überprüfen. (man spricht von Client Authentication).

2 Herstellerstandards für X.509 Zertifikate

Folgende Hersteller verwenden eigene Normen und Formate zum Erzeugen und Speichern von Zertifikaten und PKI- (Public Key Infrastructure) Keys.

RSA (Rhivest, Shamir, Adelman): unterstützt PKCS#n Standards. Entwickelten das nach ihnen benannte asymmetrische RSA Verschlüsselungs Verfahren.

Netscape: Unterstützt PKCS#11 - Cryptographic Token Interface Standard, PKCS#7 zum Speichern von Zertifikaten und für Certifikate Revokation List, PKCS#12 zum Austausch von Zertifikaten und PKI Keys, keyX.db und certX.db als permenter Speicher für Zertifikate und PKI Keys im Dateisystem (Key- bzw Certifikate Store). Zur asymmetrischen Verschlüsselung wird das RSA Verfahren unterstützt.
Verfügbare Tools: certutil, signtool, …

OpenSSL: Unterstützt folgende Formate: PKCS#7, PKCS#12, X509.
Als asymmetrische Verfahren werden sowohl RSA als auch Diffie-Hellman (DH) verwendet. Zum Signieren wird DSA (Digital Signature Algorithm) unterstützt. Als encoding type für Zertifikate stehen bei OpenSSL das DER Format, das PEM Format (base64 encoded version of DER) und das NET Format zur Verfügung.
Verfügbare Tools: openssl x509, openssl pkcs7, openssl crl2pkcs7, openssl pkcs12, openssl genrsa, …

Sun Java Secure Socket Extensions (JSSE): unterstützt PKCS#7 (PEM encoded) für den Import von signierten Zertifikaten in der Java Key Store. Verfügbare Tools: keytools, java signer-ein Programm zum signieren von Java Archiven (.jar),…

Microsoft Cryptographic Service Provider: Unterstützt kein PKCS#11! Verwedet ein eigenes Verfahren zum Zugriff auf Key- und Certifikate Store. Das Erstellen von Client Zertifikaten erfolgt über Microsoft Certificate Services. Ein Certificate Request muss über eine bestimmte Web Page am Internet Information Server (IIS) der Zertifizierungsstelle gemacht werden. Diese Page stößt auch die Generierung des Private/Public Key Paares im Key Store am. Der signierte CSR kann als PKCS#7 eingespielt werden. Microsoft unterstützt auch das PKCS#12 Format zum Import/Export von Client und Server Zertifikaten samt Keys in bzw aus dem Microsoft Key Store.
Als PKI Verschlüsselungsverfahren verden sowohl RSA als auch Diffie-Hellman unterstützt. Verwendeter encoding type des MS-CSP ist das DER Format für PKCS#7.
Microsoft verwaltet einen Certificate Store mit dem Namen „MY“ pro Benutzer im Benutzerprofil. Zusätzlich gibt es systemweite Certificate Stores für den jeweiligen Arbeitsrechner (und für Services). Zertifikate und Keys werden sowohl als Dateien im Dateisystem als auch in der Registry abgelegt.
Verfügbare Tools: certutil, certificate snap in für Management Console (mmc), certificate management im IE, MS Certificate Services (bei Windows 2000 Server).

(Die obige Liste stellt keine Anspruch auf Vollständigkeit.)

3 SSL und Comtarsia Logon Client

Um möglichst hohe Konformität und Kompartibilität mit dem Zielbetriebsystem des Comtarsia Logon Clients zu erreichen (Windows), um etwaige Synergieeffekte (Wiederverwendung zur Verfügung gestellter Client Zertifikate von anderen Applikationen) und der Möglichkeit der Verwendung von Smart-Cards ausnützen zu können wurde entschieden, für den Comtarsia Logon Client den Microsoft Cryptographic Service Provider zu verwenden.
Um jedoch eine zu starke Herstellerabhängikeit zu verhinden ist geplant eine automatische Funktion zum Import, Export und Austausch von gebräuchlichen Zertifikat- bzw. Key - Formaten im Logon Client zu implementieren.
Angestrebt wird hier die Verwendung der Formate PKCS#7 bzw PKCS#12 die wie oben erwähnt sowohl von RSA, Netscape, OpenSSL, Sun JSSE als auch Microsoft unterstützt werden.

Es ist angedacht ein weiteres Zusatzprodukt (mit graphischer Oberfläche) zu entwickeln, welches es ermöglicht direkt auf Key und Zertifikat Stores andere Hersteller zuzugreifen (etwa Netscape's certX.db und keyX.db) um Zertifikate bzw Keys mit dem Microsoft Certificate Store austauschen zu können, und damit z.B. die Vorbereitung von automatischer Softwareverteilung zu erleichtern.

4 Technische Realisierung

In obiger Dokumentation wird nur von der Verwendung asymmetrischer Keys gesprochen.
Aus Gründen der Einfachheit der Darstellung wurde verschwiegen, dass asymmetrische Verschlüsselung nur dem Austausch von symmetrischen Schlüsseln dient (man spricht von s.g. „Session Keys“), mit denen die übertragenen Daten nun tatsächlich verschlüsselt werden.
Grund fuer die Verwendung der symmetrischen Schlüssel ist der geringere Ver- bzw. Entschlüsselungsaufwand (erforderliche Rechenleistung).
Wie oben erähnt wird im Logon Client der Microsoft SSL Stack verwendet.
Das Architekturmodell der Microsoft implementiert diese Funktionalität mit der s.g. CryptAPI, die ähnlich PKCS#11 aus einer abstrakten Definition von Schnittstellen und Funktionsaufrufen besteht. Die Funktionsaufrufe der CryptAPI werden an einen „Cryptographic Service Provider (CSP)“ weitergeleitet, der die Ver- und Entschlüsselung der Daten (bzw. alle SSL relevanten Funktionen) übernimmt. Er wird als eigenes Modul bereitgestellt.
Standardmässig hat Windows 2000 den „Microsoft Base Cryptographic Provider” mitinstalliert. Dieser unterstützt jedoch nur symmetrische Schlüssellängen von 40 bzw. 56 Bit (DES), da die amerikanischen Exportbestimmungen einen Verkauf von US Produkten imUS-Ausland die stärkere Verschlüsselung unterstützen bis dato untersagten.
Diese Bestimmung ist nun nicht mehr gültig, es ist daher zu emplehlen, mittels Windows Update den „Microsoft Enhaunced Cryptographic Provider“ bzw. den „Microsoft Strong Cryptographic Provider“, die beide 128 Bit asymmetrische Schlüssellänge ermöglichen, einzuspielen [1].
Der Logon Client unterstützt alle drei Provider und wählt im Fall des Vorhandenseins mehrer CSPs jenen aus, der die größte Datensicherheit gewährleistet.

Bevor nun SSL Verschlüsselung verwendet werden kann sind folgende Voraussetzungen zu erfüllen:
Am jeweiligen LDAP Server muss SSL aktiviert sein und ein Server Zertifikat vorhanden sein, entweder ein Self Signed Certificate oder aber auch ein CA Signed Certificate (siehe Einleitung Punkt a) bzw. b)).
Zusätzlich kann am Client auch entweder ein Self Signed- oder ein CA Signed Certificate eingespielt werden (Einleitung Punkte c) und d)).
Die Zertifikate und die zugehörigen Private Keys (nur für Client bzw. Server Zertifikat, nicht aber beim CA Zertifikat) müssen in den s.g. Certificate Store am Client bzw. Server eingespielt werden. Dies kann am Client mit dem mitgelieferten Programm import_key.exe bewerkstelligt werden, dessen Verwendung weiter unten erläutert wird. Am Server erfolgt dass Einspielen gemäß Herstellerangaben (exemplarisches HowTo für OpenLDAP befinet sich bei der mitgelieferten Dokumentation).
Eine Beschreibung wie eine Certificate Authority überhaupt erst hergestellt werden kann folgt nun.

[1] Die hier erwaehnten Provider sind RSA Full Provider die der Ldap Logon Client verwendet.

5 Erstellen einer Testumgebung

Als Software zur Erstellung der Test Certificate Authority wurde Openssl gewählt, weil diese geschützt durch die GNU Public Licence im Internet frei verfügbar ist, als Standardsoftware angesehen werden kann, dadurch jede Menge Doku im Internet vorhanden ist und Openssl sowohl unter Unix (Linux) wie auch unter Windows (durch Verwednung von cygwin, siehe www.redhat.com/cygwin) lauffähig ist.

Nach der Installation von Openssl z.B.: unter /usr existiert ein Unterverzeichnis /usr/ssl das die Konfigurationsdatei openssl.cnf beinhält.

Eine gute Dokumentation für Openssl Version 0.9.2b findet man unter http://www.dfn-pca.de/certify/ssl/handbuch/ossl092/ossl092.html.

Erzeugen ein Root Certificate Authority:

```
Openssl req -out ca.pem -new -x509
```
-erzeugt CA file "ca.pem" und CA key "privkey.pem"
```
openssl crl2pkcs7 -nocrl -certfile ca.pem -out ca.p7b -inform PEM -outform
DER
```

Erzeugen eines Server Zertifikates/Key Paares:

```
openssl genrsa -out server.key 1024
openssl req -key server.key -new -out server.req
openssl x509 -req -in server.req -CA CA.pem -CAkey privkey.pem -CAserial
file.srl -out server.pem
```
-Inhalt der Datei "file.srl" ist eine Zweistellige Nummer z.B.: "00"

Erzeugen eines Client Zertifikates/Key Paares:

```
openssl genrsa -out client.key 1024
openssl req -key client.key -new -out client.req
openssl x509 -req -in client.req -CA CA.pem -CAkey privkey.pem -CAserial
file.srl -out client.pem
```
-Inhalt der Datei "file.srl" ist eine Zweistellige Nummer z.B.:"00"

Konvertieren eines Zertifikates in PKCS#12 Format

```
openssl pkcs12 -export -in client.pem -inkey client.key -keyex -CAfile
ca.pem -name "client" -out client.pfx
```

Ueberprüfen eines Zertifikates

openssl.exe x509 -text -noout -sha1 -fingerprint -in clien.pem

Import eines Zertifikates

Das erzeugte Client Zertifikat, der zugehörige Private Key und das CA Zertifikat (falls vorhanden) kann mit import_key folgendermassen in den Key Store des Clients importiert werden.

```
USAGE: import_key -s<format_option> [-v] [<options>]
    -s<format_option>                   Switch between PKCS7 and PKCS12 format
    -v                                  Use verbose mode

  PKCS7 format options (-sPKCS7):
    -f<pkcs#7_file>                     PKCS#7 certificate file
    -k<keyfile>                         PEM format private key (not encrypted)
    -C                                  Certificate only.
    -A                                  Add certificate to the CA store

  PKCS12 format options (-sPKCS12):
    -f<pkcs#12_file>                    PKCS#12 certificate and key file.
    -p<pkcs#12_password>                PKCS#12 password.

  examples:
    to import a pkcs#12 certificate and key into the user store:
      import_key -sPKCS12 -v -fclient.pfx -psecret

    to import a pkcs#7 certificate and a PEM encoded key into the user store:
      import_key -sPKCS7 -v -fclient.p7b -kclient.key
    to import a pkcs#7 certificate without a key into the user store
      import_key -sPKCS7 -v -C -fserver.p7b
    to import a pkcs#7 certificate without a key into the system store (CA)
      import_key -sPKCS7 -v -A -fca.pem
```

Unterstützte Formate:

Bei Import eines Client Zertifikates werden die Formate PKCS#12 fuer Zertifikat und Key bzw. PKCS#7 fuer Zertifikat und PEM nur für Key (ohne Passwort Verschlüsselung) unterstützt.

z.B.: `import_key –sPKCS12 –fMyClientCert.pfx –pSECRET`
`import_key –sPKCS7 –fMyClientCert.p7b -kMyPrivateKey.pem`

Der Import eines Certificate Authority Zertifikates muß mittel PKCS#7 erfolgen.

z.B.: `import_key –sPKCS7 –fMyCACert.p7b –A`

Unterstützte Sicherheits Modi im Logon Client:

Der Logon Client unterscheidet folgende Sicherheitseinstellungen:

0: Keine SSL Verschlüsselung
1: Self Signed Server Zertifikat wird akzeptiert, kein Client Zertifikat vorhanden.
2: CA Signed Server Zertifikat erforderlich, kein Client Zertifikat vorhanden.
3: CA Signed Server Zertifikat erforderlich, Self Signed- oder CA Signed Client Zertifikat vorhanden.

Beim Auffinden der Zertifikate im Certifikate Store verwendet der Logon Client folgenden Algorithmus:

Das Client Zertifikat wird im „My-“ User Certificate Store des jeweiligen Benutzers gesucht. Zuerst wird versucht ein Zertifikat zu finden dessen ‚Subject Name' mit dem Benutzernamen des aktuell angemeldeten Benutzers zu finden.
Mißlingt der Versuch, wird das erste Zertifikat genommen, dass sich im User Certificate Store befindet.

Das CA Zertifikat (falls verwendet) muß sich entweder im „Root-“ User Certificate Store (nur für den aktuellen Benutzer zugänglich) oder im „Root-“ System Certificate Store (für alle Benutzer dieser Maschine zugänglich) befinden. CA Zertifikate die mit import_key.exe mit aktivierter Option –A importiert werden, werden immer im „Root-“ System Certificate Store abgelegt.