X.509-Zertifikate mit OpenSSL

Mario Lorenz

mailto:ml@vdazone.org

18. November 2002

1 Grundlagen

- Wir nehmen an, dass mathematische Algorithmen zur sicheren Verschlüsselung und zur Signatur verfügbar seien
  - Siehe Diskussion um Rijndael/AES im neuesten Cryptogram
  - geeignete Schlüssellänge
  - Symetrisch und Asymetrisch, d.h. Public/Private Key.
- Wir nehmen an, dass diese Algorithmen fehlerfrei, und auf sicheren Maschinen implementiert seien
  - BugTRAQ (OpenSSL)
  - Konqueror

- Wir nehmen an, dass die in den Protokollen vorgesehenen Sicherheitsmechanismen vollständig implementiert sind
  - lynx, (e)links. . .
- Wir nehmen an, dass dem Hersteller der Software vertraut wird
  - Klassiker: „Reflections on Trusting Trust“
  - Verfügbarkeit von Security-Updates/Patches
  - Verfügbarkeit von Sourcen
  - Verfügbarkeit von Patches
- Wir nehmen an, dass die Menschen, die mit HW+SW umgehen, dies richtig tun, nicht gegen Social Engineering anfällig sind.
  - Das ist besonders bei CAs wichtig
  - Es gab mal zwei Codesigning-Zertifikate von Verisign für „Microsoft“

2 Ziele

2.1 Theorie

- Sichere Kommunikation
- Sicher gegen Abhören und Manipulation durch Dritte
- Authentifizierung der Endpunkte (unterhalte ich mich wirklich mit Amazon.com ?)

Herstellung von Vertrauen durch Bezug auf bestehendes Vertrauen

Kurz: Ich vertraue Dir, deswegen vertraue ich auch dem, dem Du vertraust.

2.2 Praxis

Ich will, für wenig Geld, einen “Sicheren Server“ mit einer https-URL, damit, ohne weitere Warnungen durch den Browser, und mit geschlossenem Schloss, damit tausende Kunden in meinem E-Commerce - Shoppingcenter Millionenumsätze[a] machen.

[a] Per Kreditkarte. Und damit alle ganz sicher sind, dass die Bestellung bearbeitet wird, bekommt der Kunde natürlich eine Bestätigung per EMail, mit der Kreditkartennummer zur Kontrolle.

3 Realisierung

3.1 Zertifikate

- “Zertifikat“ == Elektronische Signatur unter Identitätsbeschreibung sowie öffentlichem Schlüssel, damit mit diesem Zertifikat weitere Vorgänge (oder Zertifikate) unterschrieben werden können.
- Problem: Bedeutung der Signatur
  - Identitätsbeweis ? Authorisation ? Mitgliedsbeweis ?
  - “Policy“ der CA. Diese sollte vor dem Vertrauen einer CA gekannt werden
  - Nur wenn weitere Randbedingungen eingehalten werden, gilt die Signatur als Signatur im Sinne des SigG

- Problem: Zertifikate können ungültig werden
  - Durch Kompromittierung (geheimer Schlüssel nicht mehr geheim)
  - Austritt, Kündigung...
  - Zeitliche Gültigkeit
  - CRL

- Web of Trust
  - PGP
  - GnuPG
- Zentrale Hierarchie
  - Alle vertrauen der(den) Root(s) (Wurzelinstanzen, CAs), welche ggf. weitere Unter-CAs zertifiziert
  - Root(CA)-Zertifikate im Browser enthalten. Dem Browser-Hersteller muss man sowieso vertrauen. . .
  - Problempunkt: Einzelne CAs, mit ggf. schwächerer Policy
  - Eigene CA. Dies ist aber, wenn mans “richtig“ machen will, mit Aufwand verbunden
  - Standardformat für Zertifikate: X.509 (ITU-T/CCITT-Standard, RFC 3280)

3.2 Aufbau von X.509-Zertifikaten

- ASN.1 DER Encoded
  - Binärformat Type,Length, Value
  - relativ schwierig zu parsen und zu erzeugen, weil Längen der Elemente vorher unbekannt. Es gibt keine festen Strukturen.
  - Dafür einfach zu erweitern.

Tabelle 1: Aufbau eines X.509-Zertifikates

version	Version (default: v1)
serialNumber	Seriennummer
signature	Algorithmus, Parameter
issuer	Name des Ausstellers(X.501 DN)
validity	Gültigkeit
notBefore	von
notAfter	bis
subject	Name des Zertifikatsinhabers
subjectPublicKeyInfo	Public Key des Zertifkatsinhabers
issuerUniqueID	Eindeutige ID des Ausstellers (v2 oder v3)
subjectUniqueID	Eindeutige ID des Inhabers (v2 oder v3)
extensions	Erweiterungen
signatureAlgorithm	Algorithmus, Parameter
signatureValue	Eigentliche Signatur

3.3 Certificate Extensions

- key identifiers
  - Authority key identifier
  - Subject key identifiers
- key usage
  - digitalSignature
  - nonRepudiation
  - keyEncipherment
  - dataEncipherment
  - keyAgreement
  - keyCertSign
  - cRLSign
  - encipherOnly
  - decipherOnly

- extended key usage
  - id-kp-serverAuth
  - id-kp-clientAuth
  - id-kp-codeSigning
  - id-kp-emailProtection
  - id-kp-timeStamping
  - id-kp-OCSPSigning
- basic constraints
  - ca
  - PathLenConstraint
- subject alternative name
- name constraints
- certificate policies, policy mappings, policy constraints
- crlDistributionPoints
- Netscape-eigene Extensions.

3.4 Zertifizierungsvorgang

- Registrierung bei der CA (ggf. über RA)
- Erzeugung der Keys (beim User!)
- Erzeugung eines CSR - Certificate Signing Request, übermitteln an CA
- CA erzeugt daraus das Zertifikat (CRT) und schickt es an den User zurück
- User hat nun Key und Zertifikat, und kann sich ausweisen...
- (evtl.) CA revoked Zertifikat vorzeitig durch Aufnahme in CRL

3.5 Certificate Revocation Lists

Tabelle 2: Aufbau einer X.509-CRL

version	Version
signature	Algorithmus, Parameter
issuer	Name des Ausstellers(X.501 DN)
thisUpdate	Zeitstempel dieses Updates
nextUpdate	Zeitpunkt des nächsten Updates
revokedCertificates	Liste der Zertifikate, mit jeweils
userCertificate	CertificateSerialNumber
revocationDate	Zeitpunkt der Sperre
crlEntryExtensions	Optionale Extensions des Eintrages
crlExtensions	Extensions der CRL
signatureAlgorithm	Algorithmus, Parameter
signatureValue	Eigentliche Signatur

- crlExtensions
  - AuthorityKeyIdentifier
  - IssuerAlternativeName
  - CRL Number
  - Delta CRL Indicator
- crlEntryExtensions
  - ReasonCode
    - unspecified
    - keyCompromise
    - cACompromise
    - affiliationChanged
    - superseded
    - cessationOfOperation
    - certificateHold
    - privilegeWithdrawn
    - aACompromise
  - HoldInstructionCode
  - InvalidityDate

3.6 Zertifikatsvalidierung

- Bis Root-Zertifikat erreicht ist, wiederhole..
  - Zertifikat prüfen (Sinnige Werte)
  - Signatur stimmt
  - Zertifikat zeitlich gültig (validity)
  - Zertifikat ist nicht revoked
    - ggf. CRL aktualisieren
    - CRL auf Gültigkeit (gültige Signatur) prüfen
  - Constraints werden eingehalten (Name gültig, Ziel gültig, Zertifikat ist CA. . .)
  - Elternzertifikat holen, Elternzertifikat prüfen.

3.7 Dateiformate

- DER Standard-Form der Zertifikate, 8Bit-Binärdatei
- PEM DER, aber Base64-Codiert
- NET Netscape-Server-Format (obsolet)
- PKCS7 generisches Format für verschlüsselte Nachrichten, wieder als DER oder PEM
- PKCS8 Format für geheime private Keys, ggf. verschlüsselt/passwortgeschützt
- PKCS12 Format für Bündel aus Schlüssel, Zertifikaten und Zertifikatsketten, zum Import/Export in Mozilla, MSIE ....

4 Praxis

Hier kommen jetzt ein paar praktische Beispiele. . .