Gestión de Certificados y Mecanismos de Seguridad con OpenSSL

OpenSSL (I)

- Herramienta para implementar mecanismos y protocolos de seguridad
- Basada en el proyecto SSLeay, iniciado en 1995 por Eric A. Young y Tim J.Houston
- Es distribuida, usando una licencia de código libre
- Dos formas de usar
  - Línea de comandos
  - Librería Criptográfica, c/c++
- Enlaces
  - Proyecto OpenSSL (http://www.openssl.org/)
  - Distribución SO Windows (http://www.openssl.org/related/binaries.html)

OpenSSL(II)

- Creación y gestión de claves privadas/públicas
- Creación y gestión de certificados digitales X.509,
  - Solicitudes de certificados digitales (CSR) y listas de revocación de certificados (CRL).
- Cálculo de "resúmenes" de mensajes mediante funciones hash. Firma digital
- Cifrado y descifrado simétrico con un amplio conjunto de algoritmos
- Desarrollo de aplicaciones SSL/TLS cliente-servidor
- Manejo de mensajes S/MIME cifrados y firmados

Línea comandos OpenSSL

Símbolo del sistema - openssl

```
des-ofb          des3             desx             idea             idea-cbc
idea-cfb         idea-ecb         idea-ofb         rc2              rc2-40-cbc
rc2-64-cbc       rc2-cbc          rc2-cfb          rc2-ecb          rc2-ofb
rc4              rc4-40

OpenSSL> help
openssl:Error: 'help' is an invalid command.

Standard commands
asn1parse        ca               ciphers          crl              crl2pkcs7
dgst             dh               dhparam          dsa              dsaparam
ec               ecparam          enc              engine           errstr
gendh            gendsa           genrsa           nseq             ocsp
passwd           pkcs12           pkcs7            pkcs8            prime
rand             req              rsa              rsautl           s_client
s_server         s_time           sess_id          smime            speed
spkac            verify           version          x509

Message Digest commands (see the `dgst' command for more details)
md2              md4              md5              rmd160           sha
sha1

Cipher commands (see the `enc' command for more details)
aes-128-cbc      aes-128-ecb      aes-192-cbc      aes-192-ecb      aes-256-cbc
aes-256-ecb      base64           bf               bf-cbc           bf-cfb
bf-ecb           bf-ofb           cast             cast-cbc         cast5-cbc
cast5-cfb        cast5-ecb        cast5-ofb        des              des-cbc
des-cfb          des-ecb          des-ede          des-ede-cbc      des-ede-cfb
des-ede-ofb      des-ede3         des-ede3-cbc     des-ede3-cfb     des-ede3-ofb
des-ofb          des3             desx             idea             idea-cbc
idea-cfb         idea-ecb         idea-ofb         rc2              rc2-40-cbc
rc2-64-cbc       rc2-cbc          rc2-cfb          rc2-ecb          rc2-ofb
rc4              rc4-40

OpenSSL>
```

Formatos Certificados y Claves

- Certificados X509
  - PEM (*.pem)
  - DER (*.crt, *cer, *.der)
- Certificados X509 + Clave Privada
  - PKCS12 (*.p12, *.pfx)
- Claves Privadas (PKCS#8)
  - PEM (*.pem)
  - DER (*.der)

Generación de Claves RSA

openssl>

(Generación de un par clave pública-privada)
genrsa -out CAClavePrivada.pem 4096
rsa -noout -text -in CAClavePrivada.pem

(Con esto ciframos la clave privada, opciones des, des3, aes128, aes192, aes256)
genrsa -des3 -out CAClavePrivadaCifrada.pem 4096
genrsa -aes256 -out CAClavePrivadaCifrada2.pem 4096

(Obtenemos la clave pública de la CA a partir de la Clave privada)
rsa -in CAClavePrivada.pem -pubout -out CAClavePublica.pem

Creación de certificados X.509
Certificado de CA*

(Generación de un Certificado de CA o Certificado Autofirmado)

req -new -x509 -days 3650 -key CAClavePrivada.pem -out CACertificado.pem

(Convertimos el certificado de la CA en formato crt y lo podemos ya instalar automáticamente en windows en las entidades raíz fiable fiables)

x509 –inform PEM –in CACertificado.pem –outform DER –out CACertificado.crt

(Creamos un certificado en formato p12, con la clave privada incorporada protegida con un clave cifrada con 3des ("ppppp"). Los certificados con extensión .p12 se instalan automáticamente en la carpeta certificados personales incluyendo la clave privada)

pkcs12 -export -in CACertificado.pem -inkey CAClavePrivada.pem -out cacert.p12

(Vemos el certificado)

x509 -in CACertificado.pem -noout -text

x509 -in CACertificado.pem -noout -text -out CACertificado.pem.txt

*CA: Certification Authority

Creación de certificados X.509
Certificado de CA*

OpenSSL> x509 -in CACertificado.pem -noout -text

```
Certificate:
   Data:
      Version: 3 (0x2)
      Serial Number:      dc:bb:3d:a4:33:9a:34:61
      Signature Algorithm: sha1WithRSAEncryption
      Issuer: C=ES, ST=MADRID, L=BOADILLA, O=UPM, OU=FIM, CN=LMENGUAL/emailAddress=lmengual@fi.upm.es
      Validity
         Not Before: Sep 28 13:57:00 2009 GMT
         Not After : Sep 26 13:57:00 2019 GMT
      Subject: C=ES, ST=MADRID, L=BOADILLA, O=UPM, OU=FIM, CN=LMENGUAL/emailAddress=lmengual@fi.upm.es
      Subject Public Key Info:
         Public Key Algorithm: rsaEncryption
         RSA Public Key: (4096 bit)      Modulus (4096 bit):
               00:c1:bb:e7:c5:55:9d:76:39:12:b7:46:f2:61:6b:
               42:a7:ff:b9:1a:3c:4c:4d:30:e8:1b:bc:0b:66:49:
               ab:92:ff:ed:d1:00:be:b1:9e:63:fb:68:63:b2:2b:
               88:bc:47:59:df:bf:81:46:a4:26:a6:49:d3:86:37:
               6b:53:6b:e5:4c:b9:ba:69:9f:28:db:a6:30:ca:b8:

Signature Algorithm: sha1WithRSAEncryption
      45:2e:ea:27:8d:7d:d7:bb:8e:c4:eb:8e:d9:91:e3:f5:e6:64:
      6d:1e:e7:d0:6a:18:12:a2:eb:56:4a:41:7f:68:34:5b:85:29:
      e0:1e:5d:a1:2a:5c:58:30:55:ad:6e:98:c0:03:29:17:47:35:c
      f2:e9:6e:6e:03:2f:02:5e:cc:4d:bc:81:e5:68:4f:6c:aa:c6:
```

*CA: Certification Authority

Creación de certificados X.509
Certificado de usuario firmado por CA (I)

(Generación de un par clave privada-publica por un usuario)

```
genrsa -out usrClavePrivada.pem 1024
rsa -noout -text -in usrClavePrivada.pem
```

(Obtención de la clave pública)

```
rsa -in usrClavePrivada.pem -pubout -out usrClavePublica.pem
```

(Generación de una solicitud de certificado CSR, Certificate Signing Request)

```
req –new -key usrClavePrivada.pem -out usrPeticion.csr
req -in usrPeticion.csr -noout -text
```

(Firma por la CA de la solicitud de usuario)

```
x509 -req -days 365 -in usrPeticion.csr -CA CACertificado.pem -CAkey
CAClavePrivada.pem -set_serial 01 -out usrCertificado.pem
```

Creación de certificados X.509
Certificado de usuario firmado por CA (II)

(Con esta llamada convertimos de PEM a DER el certificado)

x509 –inform PEM –in usrCertificado.pem –outform DER –out usrCertificado.crt

(Exportamos el certificado a formato *.p12)

pkcs12 -export -in usrCertificado.pem -inkey usrClavePrivada.pem -out cert_user.p12

(Ver certificado)

x509 -in usrCertificado.pem -noout -text

x509 -in usrCertificado.pem -noout -text -out usrCertificado.pem.txt

Creación de certificados X.509
Certificado de usuario firmado por CA (III)

```
OpenSSL> x509 -in usrCertificado.pem -noout -text
Certificate:
   Data:
      Version: 1 (0x0)
      Serial Number: 1 (0x1)
      Signature Algorithm: sha1WithRSAEncryption
      Issuer: C=ES, ST=MADRID, L=BOADILLA, O=UPM, OU=FIM, CN=LMENGUAL/emailAddress=lmengual@fi.upm.es
      Validity
         Not Before: Sep 28 15:53:00 2009 GMT
         Not After : Sep 28 15:53:00 2010 GMT
      Subject: C=ES, ST=MA, L=BOA, O=UPM, OU=FIM, CN=Alumno1/emailAddress=Alumno1@fi.upm.es
      Subject Public Key Info:
         Public Key Algorithm: rsaEncryption
         RSA Public Key: (1024 bit)
            Modulus (1024 bit):
               00:d4:76:e0:d4:cb:8e:2d:eb:1e:c6:ea:cd:f6:a6:
               f7:6c:47:13:2f:a2:64:48:76:8c:42:b0:f8:5b:b2:
dc:34:63:2e:0c:4d:08:57:fe:04:d6:34:67:e8:12:
               d0:8d:7f:ec:81:37:08:60:37
            Exponent: 65537 (0x10001)
   Signature Algorithm: sha1WithRSAEncryption
      8d:bb:3b:61:7c:90:9a:e5:ef:6e:fe:ab:8e:a0:0e:ae:22:67:
      c9:64:86:d0:55:cd:e5:c9:ea:26:d3:2e:08:68:41:50:be:80:
```

Firma de Documentos (I)

(HASH del fichero documento.txt e imprime en pantalla)

dgst -md5 documento.txt

(firma con la usrClavePrivada.pem del fichero documento.doc e imprime en fichero documento.signature)

dgst -sign usrClavePrivada.pem -out documento.signature documento.txt

(verificación de firma con la clave pública usrClavepublica.pem)

dgst -verify usrClavePublica.pem -signature documento.signature documento.txt

(este comando imprime la clave pública)

rsa -in usrClavePrivada.pem -pubout -out usrClavePublica.pem

Cifrado con Clave pública

(usrCertificado.pem contiene la clave pública RSA que se utiliza para cifrar)

rsautl -in documento.txt -out documento_cifrado_pub.rsa -inkey usrCertificado.pem -certin -encrypt

(usrClavePrivada.pem contiene la clave privada RSA que se utiliza para Descifrar)

rsautl -in documento_cifrado_pub.rsa -out documento_originalcifrado_pub.txt -inkey usrClavePrivada.pem -decrypt

Cifrado Simétrico (I)

CIFRADO SIMÉTRICO

des -in documento.txt -out documento_cifrado_des -pass pass:clave

DECRIFRADO SIMÉTRICO

des –d -in documento_cifrado_des -out documento_original_des.txt -pass pass:clave

(Para utilizar otro algoritmo basta cambiar “des” por “des3”, “rc2”, “rc4”, “idea-ebc”, “aes-256-ecb”…..

Para descifrar el mismo comando, pero añadiendo la opción –d)

Cifrado Simétrico (II)

```
Cipher commands (see the `enc' command for more details)
aes-128-cbc     aes-128-ecb     aes-192-cbc     aes-192-ecb     aes-256-cbc
aes-256-ecb     base64          bf              bf-cbc          bf-cfb
bf-ecb          bf-ofb          cast            cast-cbc        cast5-cbc
cast5-cfb       cast5-ecb       cast5-ofb       des             des-cbc
des-cfb         des-ecb         des-ede         des-ede-cbc     des-ede-cfb
des-ede-ofb     des-ede3        des-ede3-cbc    des-ede3-cfb    des-ede3-ofb
des-ofb         des3            desx            idea            idea-cbc
idea-cfb        idea-ecb        idea-ofb        rc2             rc2-40-cbc
rc2-64-cbc      rc2-cbc         rc2-cfb         rc2-ecb         rc2-ofb
rc4             rc4-40
```