# Autoridad de Certificación OpenSSL

Esta guía muestra cómo actuar como su propia autoridad de certificación ("certificate authority" en inglés, abreviado por "CA") usando las herramientas de línea de órdenes de `OpenSSL`. Esto es útil en varias situaciones, tales como la emisión de certificados de servidor para asegurar un sitio web de intranet o para la emisión de certificados a los clientes para que puedan autenticarse en un servidor.

## Introducción

`OpenSSL` es una biblioteca criptográfica libre y de código abierto que proporciona varias herramientas de línea de órdenes para el manejo de certificados digitales. Algunas de estas herramientas se pueden utilizar para actuar como una autoridad de certificación.

Una autoridad de certificación (CA) es una entidad que firma certificados digitales. Muchos sitios web tienen que dejar sentado a sus clientes que la conexión es segura, por lo que pagan a una CA de confianza a nivel internacional (por ejemplo: `VeriSign`, `DigiCert`) para firmar un certificado para su dominio.

En algunos casos puede tener más sentido actuar como su propia CA, en lugar de pagar una CA como `DigiCert`. Los casos más comunes incluyen asegurar un sitio web de intranet o emisión de certificados a los clientes para que puedan autenticarse en un servidor (por ejemplo; Apache, OpenVPN).

## Crear la pareja de `root`

Actuar como una autoridad de certificación (CA) significa tratar con pares de cifrado de claves privadas y certificados públicos. El primer par criptográfico que crearemos es el par de `root`. Éste consiste en la clave de `root` (`ca.key.pem`) y el certificado de `root` (`ca.cert.pem`). Este par forma la identidad de su CA.

Por lo general, la entidad `root` no firma directamente los certificados de servidor o cliente. La entidad de certificación `root` sólo se usa para la creación de una o más entidades CA intermedias, que son de confianza para la entidad emisora `root` como para firmar certificados en su nombre. Esta es la mejor práctica. Permite que la clave `root` permanezca fuera de línea y no utilizada tanto como sea posible, ya que cualquier compromiso de la clave de `root` es desastroso.

```
-----------------------------------------------------------------
(¡!) Nota

Es la mejor práctica para crear el par de root en un entorno
seguro. Idealmente, esto debería ser en un equipo totalmente
encriptado, con huecos de aire, que está permanentemente aislada
de Internet. Retire la tarjeta inalámbrica y llene el puerto de
Ethernet con pegamento.
-----------------------------------------------------------------
```

### Preparar el directorio

Elija un directorio (`/root/ca`) para almacenar todas las claves y certificados.

```
# mkdir -p /root/ca
```

Cree la estructura de directorios. Los ficheros `index.txt` y `serial` actuan como una base de datos de archivos planos para realizar un seguimiento de los certificados firmados.

```
# cd /root/ca
# mkdir certs crl newcerts private
# chmod 700 private
# touch index.txt
# echo 1000 > serial
```

### Preparar el archivo de configuración

Se debe crear un archivo de configuración para OpenSSL para su uso. Copiar el archivo de configuración de CA `root` del apéndice en `/root/ca/openssl.cnf`.

La sección `[ca]` es obligatoria. Aquí le decimos a `OpenSSL` que utilice las opciones de la sección `[ CA_default ]`.

```
[ ca ]
# `man ca`
default_ca = CA_default
```

La sección `[ CA_default ]` contiene una serie de valores predeterminados. Asegúrese de que declara el directorio que eligió anteriormente `/root/ca`.

```
[ CA_default ]
# Directory and file locations.
dir               = /root/ca
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
serial            = $dir/serial
RANDFILE          = $dir/private/.rand

# The root key and root certificate.
private_key       = $dir/private/ca.key.pem
certificate       = $dir/certs/ca.cert.pem

# For certificate revocation lists.
crlnumber         = $dir/crlnumber
crl               = $dir/crl/ca.crl.pem
crl_extensions    = crl_ext
default_crl_days  = 30

# SHA-1 is deprecated, so use SHA-2 instead.
default_md        = sha256

name_opt          = ca_default
cert_opt          = ca_default
default_days      = 375
```

```
preserve          = no
policy            = policy_strict
```

Vamos a aplicar `policy_strict` para todas las firmas de la `root` CA, en tanto que la `root` CA será usada sólo para crear entidades emisoras intermedias.

```
[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

Vamos a aplicar `policy_loose` para todas las firmas intermedias CA, en tanto que la CA intermedia firma certificados de servidor y cliente que pueden venir de terceros varios.

```
[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

Las opciones de la sección `[ req ]` son aplicadas al crear certificados o solicitudes de firmas de certificados.

```
[ req ]
# Options for the `req` tool (`man req`).
default_bits        = 2048
distinguished_name  = req_distinguished_name
string_mask         = utf8only

# SHA-1 is deprecated, so use SHA-2 instead.
default_md          = sha256

# Extension to add when the -x509 option is used.
x509_extensions     = v3_ca
```

La sección `[ req_distinguished_name ]` declara a los datos requeridos normalmente en una solicitud de firma de certificado. Opcionalmente, puede especificar algunos valores por defecto.

```
[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName                     = Country Name (2 letter code)
```

```
stateOrProvinceName             = State or Province Name
localityName                    = Locality Name
0.organizationName              = Organization Name
organizationalUnitName          = Organizational Unit Name
commonName                      = Common Name
emailAddress                    = Email Address

# Optionally, specify some defaults.
countryName_default             = GB
stateOrProvinceName_default     = England
localityName_default            =
0.organizationName_default      = Alice Ltd
#organizationalUnitName_default =
#emailAddress_default           =
```

Las siguientes secciones son extensiones que se pueden aplicar al firmar certificados. Por ejemplo, pasando el argumento de línea de órdenes `-extensions v3_ca` aplicará las opciones establecidas en `[ v3_ca ]`.

Vamos a aplicar la extensión `v3_ca` cuando creamos el certificado `root`.

```
[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

Vamos a aplicar la extensión `v3_ca_intermediate` cuando creamos el certificado intermedio. `pathlen:0` asegura que no puede haber autoridades de certificación más allá de la entidad emisora CA intermedia.

```
[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

Vamos a aplicar la extensión `usr_cert` al firmar los certificados de cliente, tales como los que se utilizan para la autenticación de usuarios remotos.

```
[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection
```

Vamos a aplicar la extensión `server_cert` cuando sean firmados certificados del servidor, tales como los utilizados para los servidores web.

```
[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

La extensión `crl_ext` se aplica automáticamente al crear listas de revocación de certificados.

```
[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always
```

Vamos a aplicar la extensión `ocsp` al firmar el certificado Online Certificate Status Protocol (OCSP).

```
[ ocsp ]
# Extension for OCSP signing certificates (`man ocsp`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

### Crear la clave `root`

Cree la clave de `root` (`ca.key.pem`) y manténgala en absoluta seguridad. Cualquier persona en posesión de la clave raíz puede emitir certificados de confianza. Cifre la clave de raíz con el cifrado AES de 256 bits y una contraseña segura.

```
-----------------------------------------------------------------
(¡!) Nota

Utilice 4096 bits para todas las claves de `root` y la autoridad de
certificación intermedia. Usted todavía será capaz de firmar
certificados de servidor y cliente de una longitud más corta.
-----------------------------------------------------------------

# cd /root/ca
# openssl genrsa -aes256 -out private/ca.key.pem 4096

Enter pass phrase for ca.key.pem: secretpassword
Verifying - Enter pass phrase for ca.key.pem: secretpassword

# chmod 400 private/ca.key.pem
```

### Crear el certificado de `root`

Utilice la clave de `root` (`ca.key.pem`) para crear un certificado de `root` (`ca.cert.pem`). Dote al certificado de `root` de una fecha de caducidad larga, tal como veinte años. Una vez que el certificado de `root` expira, todos los certificados firmados por la CA dejan de ser válidos.

```
-----------------------------------------------------------------
(¡!) Advertencia

Siempre que utilice la herramienta req, debe especificar un archivo
de configuración para usarlo con la opción -config, de lo contrario
OpenSSL remitirá por defecto a /etc/pki/tls/openssl.cnf.
-----------------------------------------------------------------

# cd /root/ca
# openssl req -config openssl.cnf \
      -key private/ca.key.pem \
      -new -x509 -days 7300 -sha256 -extensions v3_ca \
      -out certs/ca.cert.pem

Enter pass phrase for ca.key.pem: secretpassword
You are about to be asked to enter information that will be incorporated
into your certificate request.
-----
Country Name (2 letter code) [XX]:GB
State or Province Name []:England
Locality Name []:
Organization Name []:Alice Ltd
Organizational Unit Name []:Alice Ltd Certificate Authority
Common Name []:Alice Ltd Root CA
Email Address []:

# chmod 444 certs/ca.cert.pem
```

### Verificar el certificado de `root`

```
# openssl x509 -noout -text -in certs/ca.cert.pem
```

La salida muestra:

- el `Signature Algorithm` utilizado
- las fechas de certificado `Validity`
- la longitud en bits de la `Public-Key`
- la `Issuer`, que es la entidad que firmó el certificado
- el `Subject`, que se refiere al propio certificado

La `Issuer` y el `Subject` son idénticos, ya que el certificado es autofirmado. Tenga en cuenta que todos los certificados de `root` son auto-firmado.

```
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=GB, ST=England,
            O=Alice Ltd, OU=Alice Ltd Certificate Authority,
```

```
            CN=Alice Ltd Root CA
    Validity
        Not Before: Apr 11 12:22:58 2015 GMT
        Not After : Apr  6 12:22:58 2035 GMT
    Subject: C=GB, ST=England,
             O=Alice Ltd, OU=Alice Ltd Certificate Authority,
             CN=Alice Ltd Root CA
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
```

La salida también muestra las extensiones `X509v3`. Aplicamos la extensión `v3_ca`, por lo que las opciones de `[ v3_ca ]` deberían quedar reflejadas en la salida.

```
X509v3 extensions:
    X509v3 Subject Key Identifier:
        38:58:29:2F:6B:57:79:4F:39:FD:32:35:60:74:92:60:6E:E8:2A:31
    X509v3 Authority Key Identifier:
        keyid:38:58:29:2F:6B:57:79:4F:39:FD:32:35:60:74:92:60:6E:E8:2A:31

    X509v3 Basic Constraints: critical
        CA:TRUE
    X509v3 Key Usage: critical
        Digital Signature, Certificate Sign, CRL Sign
```