Certificate revocation lists

A certificate revocation list (CRL) provides a list of certificates that have been revoked. A client application, such as a web browser, can use a CRL to check a server's authenticity. A server application, such as Apache or OpenVPN, can use a CRL to deny access to clients that are no longer trusted.

Publish the CRL at a publicly accessible location (eg, `http://example.com/intermediate.crl.pem`). Third-parties can fetch the CRL from this location to check whether any certificates they rely on have been revoked.

> **Note**
>
> Some applications vendors have deprecated CRLs and are instead using the *Online Certificate Status Protocol (OCSP)*.

Prepare the configuration file

When a certificate authority signs a certificate, it will normally encode the CRL location into the certificate. Add `crlDistributionPoints` to the appropriate sections. In our case, add it to the `[ server_cert ]` section.

```
[ server_cert ]
# ... snipped ...
crlDistributionPoints = URI:http://example.com/intermediate.crl.
```

Create the CRL

```
# cd /root/ca
# openssl ca -config intermediate/openssl.cnf \
      -gencrl -out intermediate/crl/intermediate.crl.pem
```

> **Note**
>
> The `CRL OPTIONS` section of the `ca` man page contains more information on how to create CRLs.

You can check the contents of the CRL with the `crl` tool.

```
# openssl crl -in intermediate/crl/intermediate.crl.pem -noout -
```

No certificates have been revoked yet, so the output will state `No Revoked Certificates`.

You should re-create the CRL at regular intervals. By default, the CRL expires after 30 days. This is controlled by the `default_crl_days` option in the `[ CA_default ]` section.

Revoke a certificate

Let's walk through an example. Alice is running the Apache web server and has a private folder of heart-meltingly cute kitten pictures. Alice wants to grant her friend, Bob, access to this collection.

Bob creates a private key and certificate signing request (CSR).

```
$ cd /home/bob
$ openssl genrsa -out bob@example.com.key.pem 2048
$ openssl req -new -key bob@example.com.key.pem \
```

```
      -out bob@example.com.csr.pem

You are about to be asked to enter information that will be inco
into your certificate request.
-----
Country Name [XX]:US
State or Province Name []:California
Locality Name []:San Francisco
Organization Name []:Bob Ltd
Organizational Unit Name []:
Common Name []:bob@example.com
Email Address []:
```

Bob sends his CSR to Alice, who then signs it.

```
# cd /root/ca
# openssl ca -config intermediate/openssl.cnf \
      -extensions usr_cert -notext -md sha256 \
      -in intermediate/csr/bob@example.com.csr.pem \
      -out intermediate/certs/bob@example.com.cert.pem

Sign the certificate? [y/n]: y
1 out of 1 certificate requests certified, commit? [y/n]: y
```

Alice verifies that the certificate is valid:

```
# openssl verify -CAfile intermediate/certs/ca-chain.cert.pem \
      intermediate/certs/bob@example.com.cert.pem

bob@example.com.cert.pem: OK
```

The `index.txt` file should contain a new entry.

```
V 160420124740Z 1001 unknown ... /CN=bob@example.com
```

Alice sends Bob the signed certificate. Bob installs the certificate

in his web browser and is now able to access Alice's kitten pictures. Hurray!

Sadly, it turns out that Bob is misbehaving. Bob has posted Alice's kitten pictures to Hacker News, claiming that they're his own and gaining huge popularity. Alice finds out and needs to revoke his access immediately.

```
# cd /root/ca
# openssl ca -config intermediate/openssl.cnf \
      -revoke intermediate/certs/bob@example.com.cert.pem

Enter pass phrase for intermediate.key.pem: secretpassword
Revoking Certificate 1001.
Data Base Updated
```

The line in `index.txt` that corresponds to Bob's certificate now begins with the character `R`. This means the certificate has been revoked.

```
R 160420124740Z 150411125310Z 1001 unknown ... /CN=bob@example.c
```

After revoking Bob's certificate, Alice must re-create the CRL.

Server-side use of the CRL

For client certificates, it's typically a server-side application (eg, Apache) that is doing the verification. This application needs to have local access to the CRL.

In Alice's case, she can add the `SSLCARevocationPath` directive to her Apache configuration and copy the CRL to her web server. The next time that Bob connects to the web server, Apache will check his client certificate against the CRL and deny access.

Similarly, OpenVPN has a `crl-verify` directive so that it can block clients that have had their certificates revoked.

Client-side use of the CRL

For server certificates, it's typically a client-side application (eg, a web browser) that performs the verification. This application must have remote access to the CRL.

If a certificate was signed with an extension that includes `crlDistributionPoints`, a client-side application can read this information and fetch the CRL from the specified location.

The CRL distribution points are visible in the certificate **X509v3** details.

```
# openssl x509 -in cute-kitten-pictures.example.com.cert.pem -no

    X509v3 CRL Distribution Points:

        Full Name:
          URI:http://example.com/intermediate.crl.pem
```

Comments

← Previous

Next →

Version 1.0.4 — Last updated on 2015-12-09.