Generating X.509 Certificates

Today almost all VPN implementations allow the usage of X.509 certificate for the authentication of the peers. These are the same certificates as used for the implementation of the Secure Socket Layer (SSL) in the HTTP protocol.

This chapter will briefly cover the creation of these certificates.

Using OpenSSL

The easiest way to create X.509 certificates on Linux is the **openssl** command and the auxiliary tools. When the OpenSSL package has been installed usually an auxillary command **CA** and/or **CA.pl**, has been installed, too. We will use this command to create the certificates.

First check where the command has been installed. It is usually not in your path! On Red Hat Linux distributions it is installed in `/usr/share/ssl/misc/CA`.

Now create your certificate authority first.

```
$ mkdir certs
$ cd certs
$ /usr/share/ssl/misc/CA -newca
CA certificate filename (or enter to create) <enter>

Making CA certificate ...
Using configuration from /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
................++++++
.............++++++
writing new private key to './demoCA/private/./cakey.pem'
Enter PEM pass phrase: capassword
```

```
Verifying password - Enter PEM pass phrase: capassword
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [NRW]:
Locality Name (eg, city) [Steinfurt]:
Organization Name (eg, company) [Spenneberg.com]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:RootCA 2003
Email Address []:ralf@spenneberg.net
```

Please enter the appropiate values when asked for Country Name, etc. If you would like to have the correct values proposed (like above in my case) edit your `openssl.cnf` file. On Red Hat Linux systems you may usually find it at `/usr/share/ssl/openssl.cnf`.

The created certificate authority is only valid for one year. Often you want a longer lifetime for the certificate of your CA. Since the certificates you are signing later on usually have a shorter lifetime it is not practical to edit the `openssl.cnf` file. Rather change the lifetime manually:

```
$ cd demoCA/
$ openssl x509 -in cacert.pem -days 3650 -out cacert.pem
-signkey ./private/cakey.pem
Getting Private key
Enter PEM pass phrase: capassword
$ cd ..
```

The certificate authority is now ready to go. Let's create a certificate signing request:

```
$ /usr/share/ssl/misc/CA -newreq
Using configuration from /usr/share/ssl/openssl.cnf
```

```
Generating a 1024 bit RSA private key
..............................++++++
.................................++++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase: certpassword
Verifying password - Enter PEM pass phrase: certpassword
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [NRW]:
Locality Name (eg, city) [Steinfurt]:
Organization Name (eg, company) [Spenneberg.com]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:VPN-Gateway
Email Address []:ralf@spenneberg.net

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request (and private key) is in newreq.pem
```

The file `newreq.pem` contains the certificate signing request and the encrypted private key. This file can later be used as a private key for FreeS/WAN or Racoon. Once the request is created, we can sign it using the certificate authority.

```
$ /usr/share/ssl/misc/CA -sign
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase: capassword
Check that the request matches the signature
Signature ok
```

```
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'NRW'
localityName          :PRINTABLE:'Steinfurt'
organizationName      :PRINTABLE:'Spenneberg.com'
commonName            :PRINTABLE:'VPN-Gateway'
emailAddress          :IA5STRING:'ralf@spenneberg.net'
Certificate is to be certified until Apr 29 06:08:56 2004 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Depending on the version of the command **CA** the certificate might be print to stdout. This will be similar to the following certificate:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
CN=RootCA 2003/Email=ralf@spenneberg.net
        Validity
            Not Before: Apr 30 06:08:56 2003 GMT
            Not After : Apr 29 06:08:56 2004 GMT
        Subject: C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
CN=VPN-Gateway/Email=ralf@spenneberg.net
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:c5:3b:9c:36:3a:19:6c:a9:f2:ba:e9:d2:ed:84:
                    33:36:48:07:b2:a3:2d:59:92:b0:86:4c:81:2c:ea:
                    5c:ed:f3:ba:eb:17:4e:b3:3a:cc:b7:5b:5d:ca:b3:
                    04:ed:fb:59:3c:c5:25:3e:f3:ff:b0:22:10:fb:de:
```

```
                    72:0a:ee:42:4b:9a:d3:27:d3:b6:fb:e9:88:10:c8:
                    47:b7:26:4f:71:40:e4:75:c4:c0:ee:6b:87:b8:6f:
                    c9:5e:66:cf:bb:e7:ad:72:68:b8:6d:fd:8f:4c:1f:
                    3a:a2:0d:43:25:06:b9:92:e7:20:6c:86:15:a0:eb:
                    7f:f7:0b:9a:99:5d:14:88:9b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                CB:5C:19:9B:E6:8A:8A:FE:0E:C4:FD:5E:DF:F7:BF:3D:A8:
18:7C:08
            X509v3 Authority Key Identifier:
                keyid:01:BB:C6:33:BE:F5:9A:5E:B0:0C:5D:BD:41:E9:78:
6C:54:AD:66:8E
                DirName:/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
CN=RootCA 2003/Email=ralf@spenneberg.net
                serial:00

    Signature Algorithm: md5WithRSAEncryption
        6f:89:2b:95:af:f1:8d:4d:b7:df:e8:6d:f7:92:fb:48:8c:c4:
        1a:43:68:65:97:01:87:a6:84:b5:a1:38:bd:62:74:70:db:9e:
        78:19:d9:0c:af:18:ad:13:77:56:7d:3f:19:61:da:ba:74:30:
        8e:c5:50:0e:e3:eb:ff:95:cd:8d:d6:7e:c3:0e:ab:5b:34:94:
        bc:16:0f:ef:dc:de:40:bb:7d:ba:a2:b8:5d:f9:74:e7:28:58:
        75:a0:66:d2:8d:85:ba:38:82:08:10:33:ef:be:29:c9:31:9d:
        63:a9:f7:e0:99:ea:a7:ed:b6:b5:33:1b:1c:4a:a4:05:40:6e:
        40:7b
-----BEGIN CERTIFICATE-----
MIIDjDCCAvWgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBgjELMAkGA1UEBhMCREUx
DDAKBgNVBAgTA05SVzESMBAGA1UEBxMJU3RlaW5mdXJ0MRcwFQYDVQQKEw5TcGVu
bmViZXJnLmNvbTEUMBIGA1UEAxMLUm9vdENBIDIwMDMxIjAgBgkqhkiG9w0BCQEW
E3JhbGZAc3Blbm5lYmVyZy5uZXQwHhcNMDMwNDMwMDYwODU2WhcNMDQwNDI5MDYw
ODU2WjCBgjELMAkGA1UEBhMCREUxDDAKBgNVBAgTA05SVzESMBAGA1UEBxMJU3Rl
aW5mdXJ0MRcwFQYDVQQKEw5TcGVubmViZXJnLmNvbTEUMBIGA1UEAxMLVlBOLUdh
dGV3YXkxIjAgBgkqhkiG9w0BCQEWE3JhbGZAc3Blbm5lYmVyZy5uZXQwgZ8wDQYJ
KoZIhvcNAQEBBQADgY0AMIGJAoGBAMU7nDY6GWyp8rrp0u2EMzZIB7KjLVmSsIZM
```

```
gSzqXO3zuusXTrM6zLdbXcqzBO37WTzFJT7z/7AiEPvecgruQkua0yfTtvvpiBDI
R7cmT3FA5HXEwO5rh7hvyV5mz7vnrXJouG39j0wfOqINQyUGuZLnIGyGFaDrf/cL
mpldFIibAgMBAAGjggEOMIIBCjAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1P
cGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUy1wZm+aKiv4O
xP1e3/e/PagYfAgwga8GA1UdIwSBpzCBpIAUAbvGM771ml6wDF29Qel4bFStZo6h
gYikgYUwgYIxCzAJBgNVBAYTAkRFMQwwCgYDVQQIEwNOUlcxEjAQBgNVBAcTCVN0
ZWluZnVydDEXMBUGA1UEChMOU3Blbm5lYmVyZy5jb20xFDASBgNVBAMTC1Jvb3RD
QSAyMDAzMSIwIAYJKoZIhvcNAQkBFhNyYWxmQHNwZW5uZWJlcmcubmV0ggEAMA0G
CSqGSIb3DQEBBAUAA4GBAG+JK5Wv8Y1Nt9/obfeS+0iMxBpDaGWXAYemhLWhOL1i
dHDbnngZ2QyvGK0Td1Z9Pxlh2rp0MI7FUA7j6/+VzY3WfsMOq1s0lLwWD+/c3kC7
fbqiuF35dOcoWHWgZtKNhbo4gggQM+++KckxnWOp9+CZ6qfttrUzGxxKpAVAbkB7
-----END CERTIFICATE-----
Signed certificate is in newcert.pem
```

It is now advisable to rename the files `newreq.pem` and `newcert.pem` to something more meaningful.

```
$ mv newcert.pem vpngateway_cert.pem
$ mv newreq.pem vpngateway_key.pem
```

Now have fun creating certificates for every peer in the VPN.

In case a private key gets stolen or compromised, you have to revoke it because based on its lifetime it is still valid. The revoked keys are stored in the certificate revocation list (CRL). First, create an (empty) list:

```
$ openssl ca -gencrl -out crl.pem
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase: capassword
```

To revoke a certificate you need to have the certificate file. This is also stored in `demoCA/newcerts/`. The name of the certificate can be read in `demoCA/index.txt`. Then use the following command.

```
$ openssl ca -revoke compromised_cert.pem
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase: capassword
Revoking Certificate 01.
Data Base Updated
```

Once the certificate has been revoked, the certificate revocation list has to be recreated using the above command.

<<< Previous Home
Linux Kernel 2.5/2.6 using OpenBSD's isakmpd