Extended public key cryptography in OpenSSL

HW7-CNS Sapienza

Manuel Ivagnes 1698903

19/12/19

1 Introduction

Given the fact that, this homework is an extension of the homework 6, I will not explain again all the considerations already made in the previous paper.

In this case, we want to build a virtual machines network, and use one VM as a Certificate Authority. Other VMs can send certificate signing requests and certificate revocations to the CA.

Run different VMs requires much computational power, so I decided to use **Docker**, which uses the resource isolation features of the Linux kernel (such as cgroups and kernel namespaces) and a union-capable file system (such as OverlayFS) to allow containers to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines. Containers are isolated from one another and bundle their own software, libraries and configuration files. Moreover, they can communicate with each other through well-defined channels [2].

To exchange messages between the containers, through the docker default bridge network, I have used **NetCat**, which is a computer networking utility for reading from and writing to network connections using TCP or UDP. [5]

2 Generate/Revoke Certificates

2.1 Setup

First of all, we need to choose an image and generate containers, which can be considered as lightweight virtual machines. For this task, I have used the standard Ubuntu image, provided by Docker.

To create the containers run the following command on each machine:
docker run -it ubuntu

The *-it* option is used to return the VM shell.
Now, it is possible to see the composition of the network with:
docker inspect bridge

The basic version of Ubuntu given by Docker does not provide the libraries needed to accomplish the task, so we need to install:

- OpenSSL
- NetCat
- nano (or equivalent, to modify files)

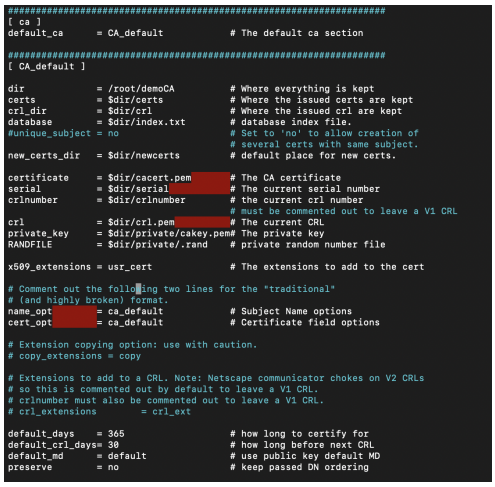
2.2 CA initialization

In this case, I want to simulate a real Certificate Authority, which can be managed by using the 2 file:

- **CA.pl** ⇒ It allows to change the configuration parameters and automatically generate all the components needed by the CA. For this task, we can change only the default path, to the absolute path "/root/demoCA". This will make easier the CA operations.

```
use strict;
use warnings;

my $openssl = "openssl";
if(defined $ENV{'OPENSSL'}) {
    $openssl = $ENV{'OPENSSL'};
} else {
    $ENV{'OPENSSL'} = $openssl;
}

my $verbose = 1;

my $OPENSSL_CONFIG = $ENV{"OPENSSL_CONFIG"} || "";
my $DAYS = "-days 365";
my $CADAYS = "-days 1095";      # 3 years
my $REQ = "$openssl req $OPENSSL_CONFIG";
my $CA = "$openssl ca $OPENSSL_CONFIG";
my $VERIFY = "$openssl verify";
my $X509 = "$openssl x509";
my $PKCS12 = "$openssl pkcs12";

# default openssl.cnf file has setup as per the following
my $CATOP = "/root/demoCA";
my $CAKEY = "cakey.pem";
my $CAREQ = "careq.pem";
my $CACERT = "cacert.pem";
my $CACRL = "crl.pem";
my $DIRMODE = 0777;

my $NEWKEY = "newkey.pem";
my $NEWREQ = "newreq.pem";
my $NEWCERT = "newcert.pem";
my $NEWP12 = "newcert.p12";
my $RET = 0;
my $WHAT = shift @ARGV || "";
my @OPENSSL_CMDS = ("req", "ca", "pkcs12", "x509", "verify");
my %EXTRA = extra_args(\@ARGV, "-extra-");
my $FILE;
```

- **Openssl.cnf** ⇒ It is the configuration file of OpenSSL. Also here we have to change the default dir path, which must match with the previous one.



```
####################################################################
[ ca ]
default_ca      = CA_default            # The default ca section

####################################################################
[ CA_default ]

dir             = /root/demoCA          # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
#unique_subject = no                    # Set to 'no' to allow creation of
                                        # several certs with same subject.
new_certs_dir   = $dir/newcerts         # default place for new certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial           # The current serial number
crlnumber       = $dir/crlnumber        # the current crl number
                                        # must be commented out to leave a V1 CRL
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand    # private random number file

x509_extensions = usr_cert              # The extensions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt        = ca_default            # Subject Name options
cert_opt        = ca_default            # Certificate field options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions        = crl_ext

default_days    = 365                   # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md      = default               # use public key default MD
preserve        = no                    # keep passed DN ordering
```

Now, the following command can create the CA:
/usr/lib/ssl/misc/CA.pl -newca

It generates the following:

- **cacert.pem** ⇒ The certificate of the CA
- **careq.pem** ⇒ The request used to generate the cacert.pem
- **certs** ⇒ A folder with the trusted certificates
- **crl** ⇒ Certificates Revocation List
- **crlnumber** ⇒ Certificates revocation list number
- **index.txt** ⇒ Database
- **index.txt.attr** ⇒ Database attributes
- **index.txt.old**
- **newcerts** ⇒ New certificates validates from the CA
- **private** ⇒ Private key of the CA
- **serial** ⇒ Serial number of the CA

2.3 Interaction

Now, the Certificate Authority is ready, let's use netcat to simulate the interaction with an user:

1. User ⇒ Generate the private key and the CSR (as seen in the previous homework). Here, the CSR parameters must match the CA configuration parameters, or it will not accept the request.

```
# For the CA policy
[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

2. CA ⇒ Start to listen by Netcat, using the command:

   nc -l -p port_number > request**.cst**

3. User ⇒ Sends the certificate to the CA:

   nc -w time ip_CA port_number < request**.csr**

   In this simple example, working on VMs, a short time is more than enough to complete the transmission.

4. CA ⇒ Sign the CSR, by using:

   openssl ca -in request**.csr**

   Here, I have used a different command from the one used for the previous homework. Indeed, in this case we have all the structure of the CA, and this command automatically updates everything.

   The new certificate will be saved in the newcerts folder.

5. CA ⇒ Send back the new certificate to the user, with it's own certificate (to perform the verification, as seen in hw6)

2.4 Certificate revocation

A Certificate Revocation List (CRL) is a list of digital certificates that have been revoked by the issuing Certificate Authority (CA) before their scheduled expiration date and should no longer be trusted. For instance, this can happen if the private key has been compromised.

To revoke a certificate on the CA machine:
openssl ca -revoke newcerts/certificate_name**.pem**

then, to update the CRL:
openssl ca -gencrl -out crl/crl_name**.pem**

Now, if the CA tries to check the validity of the certificate with the command of the hw6, this will still be approved. To check if the certificate has been revoked, it needs to add:
openssl verify -CAfile cacert.pem -CRLfile crl/crlfile**.pem -crl_check** certificate**.pem**

References

[1] course slides

[2] https://en.wikipedia.org/wiki/Docker_(software)

[3] https://en.wikipedia.org/wiki/Netcat

[4] https://www.phildev.net/ssl/opensslconf.html

[5] https://kapeli.com/cheat_sheets/Netcat.docset/Contents/Resources/Documents/index