Howto

Implementing certificate based authentication for remote users with Firewall-1/SecuRemote and openssl as CA

Version 11/30/2001-01

by Martin Freiss (martin@atsec.com) and Enno Rey (erey@security-academy.de)

This document describes how openssl can be used to issue certificates for SecuRemote users to be validated by Checkpoint's FW-1.
We'll explain the necessary steps in a short and (hopefully) concise manner. If you think something is missing or if you have any further questions, feel free to mail us directly. The document itself is under the Open Content License.
This document's home is at http://www.atsec.com/docs/fw1-openssl.howto.pdf. Please check there for the latest version.

0.) Basic principle

Certificate-based authentication means that remote users get authenticated using X.509 certificates. This is more secure than Password-based authentication, since in addition to the certificate, the user needs a password on the client side to access (decrypt) the certificate in the first place.
I.e., you need to have something (the certificate) and to know something (the password to access the Certificate).

Do not confuse the password for the certificate with the usual password-based authentication – the VPN-server never gets to see the passwords.

We have 4 major components here:

1- the CA (Certification Authority)
   The CA generates all the keys and certificates that are needed, and signs them. This is implemented by using openssl.
2- The FW1 (VPN-1) computer
   The firewall is the endpoint of your VPNs. It needs to check that the certificates presented by users doing logon are valid. VPN-1 does this by looking at the certificates, checking that they are signed (i.e. validated) by the CA described above (Optionally, VPN-1 can accept certificates signed by any CA, but this is obviously insecure). It also checks that the certificates have not been revoked (i.e., invalidated by an administrator or by the user himself) by checking with a CRL-Server (CRLs are Certificate Revocation Lists).
3- The CRL-Server
   This is simply a webserver where the CRL described above resides.
4- The Client (SecuRemote or SecureClient)
   .... without which all this would of course be pointless. The user specific certificate generated by the CA must be installed on the client.

Note that while the Firewall checks the user certificates, SecuRemote appears not to check the certificate for the Firewall, i.e. users authenticate to the firewall, but the firewall itself does not authenticate to the users. In a perfect world, it should happen both ways.

1.) Set up openssl.

If there are questions on this... RTFM or have a look at www.openssl.org. For those who are able to read & understand the german language (or to use altavista/babelfish...) the *openssl handbook* published by DFN would be a good choice [www.pca.dfn.de/dfnpca/certify/ssl/handbuch].

2.) Set up webserver

FW-1 has to be able to check the *certificate revocation list* (CRL) whenever a certificate is presented by a SecuRemote client. This can/should/must be done via HTTP (you can check LDAP as a protocol in FW-1, but we did not test this). That's why you have to set up a webserver. This webserver must be accessible by FW-1, modify routing/DNS/rule base appropriately.

3.) Configure openssl.cnf

In addition to all standard adjustments to *openssl.cnf* [see step 1. ...] **you have** to add support for the v3-extension *CRL distribution point* (see RFC 2459 for further details on this). This is the only way (at least we know of) to include the server for checking the CRL [see step 2.] in the certificate. **This is absolutely necessary**. Without this, the FW would never know where to ask for the validity of a cert and authentication will fail.
The *CRL distribution point* has to be configured in the [v3_ca] section of *openssl.cnf*. Example:

```
[ v3_ca ]

...
crlDistributionPoints   = URI:http://thewebserver.storingthecrl.com/CA.crl
```

4.) Create certificates

The process of getting a certificate can here be split up into four steps:

Generate keys to be used
Request certificate that includes keys generated in the last step
Get certificate signed by CA
Install certificate on system

These steps have to be followed in any case, regardless of the system using a certificate later on.

We assume there is some directory you installed openssl in (that's where we are working at the moment...) and we call that directory $SSLDIR. The subdirectories we use are our choice (even if based on some kind of convention), so you can use any you want.

The CA's own certificate has to be created first in order to 'publish' the CA to all systems using certificates issued by this CA.

So...

4.1) The CA's own certificate

As we said... everything starts with the generation of the keys (the private and the public one). This is accomplished via

```
openssl -genrsa -des3 -out ./etc/CAkey.pem 2048
```

This will generate RSA keys with a modulus length of 2048 bit. Note: openssl defaults to a modulus length of 512 bits which is much too short for modern use [as you can get of Arjen Lenstras table (www.cryptosavvy.com), thanks to David Ochel and Oliver Weissmann for this hint]. The keys will be encrypted themselves with 3DES (so don't be confused to see 3DES in this context...) and they will be stored in the file *CAkey.pem* which is located in $SSLDIR/etc. This encryption needs a passphrase, so

enter one (and remember it!).

To dispose of initial random numbers for key generation set environment variable $RANDFILE first and use additional rand files if paranoid (see manual for all this).

We then have to request the CA's signature for the certificate-in-creation that includes (at least) the public key and some identification data.

```
openssl req -new -x509 -days 1800 -key ./etc/CAkey.pem -out
./etc/certs/CA_cert:
```

We are requesting an X.509 cert that is valid approx. 5 years (it's the CA's own cert with a high lifetime). As input data we take the previously generated key file, output will be written to the file *CA_cert* (note directory).

You have to enter some identification (in X.500-style) here. **Pay special attention to the organisation object... keep the same name here in all certificates.** (The CA will refuse to sign certificates for organisations differing from its own).

The third step of the general procedure (signing by CA) is omitted here for obvious reasons. The CAs signature is used to prove the authenticity of a certificate. This proof is kind of replaced here by a trusted transfer of this certificate to the FW, see below [as for the client... well that's another story... see PCKS12-export in step 7].

As openssl does it's internal sorting mechanism based on the certs hash value this must be calculated. Furthermore the certs are named in a special manner (derived from a serial number)

```
cd $SSHDIR/etc/certs
mv ./CA_cert ./00.pem
ln -s 00.pem `openssl x509 -hash -noout -in 00.pem`.0
```

5.) Prepare FW-1 and import the CA's certificate

Define a server object for the CA (*Manage - Servers - New - CA*). Choose **OPSEC PKI**.
Then import ('Get') certificate generated in step 4.1 (cut & paste from a terminal window, or whatever).

6.) Request certificate for the firewall.

As usual... we follow the steps outlined above.
The first step (generating keys) is done automatically during the request, so we start with the request:

Open the firewall's workstation object, go to *Certificates - Certificate Properties - Generate*.
After having given the cert a nickname (to distinguish it from others possibly used by the firewall), enter necessary information (take care of the organisation's name, see above). After finishing take the generated request, cut & paste it to a textfile (*FW1REQ.txt*) and move this file to openssl to get it signed. As preliminary measures we had to touch a file index.txt (in $SSLDIR) and to create a file 'serial' with a numeric value 0 in it (but maybe we just missed something before and maybe you don't have to do this).

The signature is done by:

```
openssl ca -keyfile $DIR/etc/CAkey.pem -extensions v3_ca -in /tmp/FW1REQ.txt -
out firewall.pem -outdir $DIR/etc/certs
```

This should create a cert with the number from the *serial* file.
Create a link on this (see end of step 4.1):

```
ln -s serialnumber.pem ...
```

Copy this file to the GUI client (from step 5.). The cert should now get the status 'signed'.

7.) Request certificate for the SecuRemote client.

All this entirely done by openssl, not on the client's side.
First, generate keys:

```
openssl genrsa -des3 -out ./etc/username.pem -rand file_for_randomization 1024
```

create request

```
openssl req -new -key ./etc/username.pem -out ./etc/usernamereq.pem
```

sign it

```
openssl ca -keyfile ./etc/CAkey.pem -extensions v3_ca -in
./etc/usernamereq.pem -out username.pem -outdir ./etc/certs
```

link it (here *lastserial* is the 'number' of the freshly created file, based on the *serial*)

```
ln -s lastserial.pem `openssl x509 -hash -noout -in lastserial.pem`.0
```

and prepare it for the client, i.e. export it in a format the client understands

```
openssl pkcs12 -in ./etc/lastserial.pem -inkey ./etc/username.pem -export -out
username.p12 -certfile ./etc/certs/00.pem
```

You will be asked for a password. Note it... you will need it on the client during import.

Attention: in the request the organisation's name must be the same as above and the user's common name must match the user's name of the object definition on the FW.

8.) Import to SecuRemote-Client

Take the file *username*.p12, move it to the client's machine. Start SecuRemote.
Go to *Certificates - Import*. Select the file from step 7 and enter the password given during export.
Now, save the profile as an Entrust Profile (.EPF, even if Entrust is not used here in any way).
Here a new password is needed. This is the password a user enters when using his certificate.
Do not share this password. It must be user-specific (the whole step should be done by the user). There is no need of an involvement of an admin. Advise your users to use good passwords.

Create a new site, download topology, use authentication by certificate, point to .epf-file.

9.) Disable authentication by preshared secret in IKE properties of the user's group on the FW.

10.) Revocation

A certificate is revoked by

```
openssl ca -revoke ./etc/certs/certnumber.pem
```

Even if there are no revoked certs at the time you **must create** the CRL to be requested by the fw:

```
openssl ca -gencrl -out crlfile.pem
```

and convert this to *der*-format (only this is readable by FW-1)

```
openssl ca -in crlfile.pem -outform der -out CA.crl
```

Move this file to the webserver, to the dir where it is published. This **must match** the *CRL distribution point* configured in step 3.

Again, note that a valid CRL must be present.

11.) Things To Consider

Certificate Lifetimes

Certificates expire after some time. This is a good thing, because otherwise unused certificates would clutter up databases everywhere. However, expiring certificates can be an administrative nightmare.

When the CA's own certificate expires, all the signatures suddenly get invalid. This is an immediate problem, because nobody will be able to logon any more. We suggest using a high lifetime (five years or more), and think about what you (or your successor on the job, poor sod) should be doing at this time.

When the user's certificates expire, you will have to generate new ones for them. Using a VPN, we are normally working in a very controlled environment. If you do not have easy access to users (lots of travelling people), it might make sense to also use a high lifetime here (perhaps 3 years). Note that you can ”expire” certificates any time by revoking them. But do make a plan what needs to be done after the first wave of certificates start to expire. If you have a large number of users, a process should be in place to handle users' requests for certificates, and also requests for revocation of certificates, for example when a notebook with an installed certificate gets stolen.

Scripts

Some very simplistic scripts are at the end of this document. You should also at least once a day update your CRL list (cron-job).

Time

Certificates are valid during a certain lifetime. If the clocks on your CA and firewall are out of sync, freshly generated certificates might be invalid on the firewall until the firewall's clock catches up with the CA. Same for revocations – revoked certificates might still be valid on the firewall!
NTP is your friend. Clocks should be synchronized anyway, otherwise you will never have a consistent audit trail if there is a break-in.

IKE renegotiation

To enforce regular IKE renegotiation [=> regular check of revoked certificates], set lifetime for IKE [phase 1 lifetime] on 1440 minutes (1 day). This should be done in

Policy - Properties - Encryption

Attention: lifetime for phase 2 ('IPsec lifetime') should be less than lifetime for phase 1. Do not mix up values: lifetime for phase 2 is measured in seconds! If implementing heterogenous IKE [e.g. with Cisco boxes] consider not using/activating PFS on that side. FW-1 does not seem to use PFS at all (can

anybody confirm this?).

Attention: SAs in place when the CRL is changed should be cleared manually. Otherwise you won't see any effect! Even reconnecting from a client [with revoked cert] works. To check immediate enforcement of the new CRL/the revoked certificates consider restarting isakmpd and/or restarting fwd to clear SAs.

12.) Thanks

Big thanks to the following people for their help:

Theo Belke, Theo.Belke@ubg-medienzentrum.de
Ian Guthrie, iguthrie@ezenet.com
David Ochel, david@atsec.com
Dr. Oliver Weissmann, oliver@atsec.com

The scripts

Below you find two simple scripts to facilitate the whole process. To minimize effort when creating users consider changing default values for DN in openssl.cnf as a further measure.
(These scripts are completely braindead and are shown here just as a quick & dirty way of demonstrating the steps you need to go through. Somebody, please do an expect-script out of this or a nice TCL interface and share it with the world).

Create user's certificates

```
#
# Usage: genuser.sh username


BASEDIR=/our/home/dir/for/openssl/apps
RANDFILE=/some/file/wit/random/contents/access.log


echo "Please enter the user's password:"

openssl genrsa -des3 -out $BASEDIR/etc/$1.pem -rand $RANDFILE 1024

openssl req -new -key $BASEDIR/etc/$1.pem -out ./etc/$1req.pem

echo "Please enter the CA's password:"

openssl ca -keyfile $BASEDIR/etc/CAkey.pem -in $BASEDIR/etc/$1req.pem -out
$1cert.pem -extensions v3_ca -outdir $BASEDIR/etc/certs

SERIAL=`cat $BASEDIR/serial`
SERIAL=`expr $SERIAL - 1`

HASH=`openssl x509 -hash -noout -in $BASEDIR/etc/certs/$SERIAL.pem`
# delete if existing
rm -f $BASEDIR/${HASH}.0

ln -s $BASEDIR/etc/certs/$SERIAL.pem $BASEDIR/$HASH.0
```

```
echo "Please enter the user's password:"

openssl pkcs12 -in $BASEDIR/etc/certs/$SERIAL.pem -inkey $BASEDIR/etc/$1.pem -
export -out /tmp/$1.p12 -certfile $BASEDIR/etc/certs/00.pem
```

Generate CRL

```
# generate CRL in public directory and convert to DER
#
WWWDIR= /the/dir/the/CRL/is/published/in

openssl ca -gencrl -out $WWWDIR/temp.crl
openssl crl -in $WWWDIR/temp.crl -outform der -out $WWWDIR/CA.crl
```