13.6. OpenSSL

Written by Tom Rhodes.

OpenSSL is an open source implementation of the SSL and TLS protocols. It provides an encryption transport layer on top of the normal communications layer, allowing it to be intertwined with many network applications and services.

The version of OpenSSL included in FreeBSD supports the Secure Sockets Layer v2/v3 (SSLv2/SSLv3) and Transport Layer Security v1 (TLSv1) network security protocols and can be used as a general cryptographic library.

OpenSSL is often used to encrypt authentication of mail clients and to secure web based transactions such as credit card payments. Some ports, such as *www/apache24* and *databases/postgresql91-server*, include a compile option for building with OpenSSL.

FreeBSD provides two versions of OpenSSL: one in the base system and one in the Ports Collection. Users can choose which version to use by default for other ports using the following knobs:

- WITH_OPENSSL_PORT: when set, the port will use OpenSSL from the `security/openssl` port, even if the version in the base system is up to date or newer.
- WITH_OPENSSL_BASE: when set, the port will compile against OpenSSL provided by the base system.

Another common use of OpenSSL is to provide certificates for use with software applications. Certificates can be used to verify the credentials of a company or individual. If a certificate has not been signed by an external *Certificate Authority* (CA), such as http://www.verisign.com, the application that uses the certificate will produce a warning. There is a cost associated with obtaining a signed certificate and using a signed certificate is not mandatory as certificates can be self-signed. However, using an external authority will prevent warnings and can put users at ease.

This section demonstrates how to create and use certificates on a FreeBSD system. Refer to Section 28.5.2, “Configuring an LDAP Server” for an example of how to create a CA for signing one's own certificates.

For more information about SSL, read the free OpenSSL Cookbook.

13.6.1. Generating Certificates

To generate a certificate that will be signed by an external CA, issue the following command and input the information requested at the prompts. This input information will be written to the certificate. At the `Common Name` prompt, input the fully qualified name for the system that will use the certificate. If this name does not match the server, the application verifying the certificate will issue a warning to the user, rendering the verification provided by the certificate as useless.

```
# openssl req -new -nodes -out req.pem -keyout cert.key -sha25
Generating a 2048 bit RSA private key
..................+++
.....................................................+
writing new private key to 'cert.key'
-----
You are about to be asked to enter information that will be ir
into your certificate request.
What you are about to enter is what is called a Distinguished
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) [AU]:US
State or Province Name (full name) [Some-State]:PA
Locality Name (eg, city) []:Pittsburgh
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My
Organizational Unit Name (eg, section) []:Systems Administrato
Common Name (eg, YOUR name) []:localhost.example.org
Email Address []:trhodes@FreeBSD.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:Another Name
```

Other options, such as the expire time and alternate encryption algorithms, are available when creating a certificate. A complete list of options is described in openssl(1).

This command will create two files in the current directory. The certificate request, `req.pem`, can be sent to a CA who will validate the entered credentials, sign the request, and return the signed certificate. The second file, `cert.key`, is the private key for the certificate and should be stored in a secure location. If this falls in the hands of others, it can be used to impersonate the user or the server.

Alternately, if a signature from a CA is not required, a self-signed certificate can be created. First, generate the RSA key:

```
# openssl genrsa -rand -genkey -out cert.key 2048
0 semi-random bytes loaded
Generating RSA private key, 2048 bit long modulus
.....................................+++
.....................................................
e is 65537 (0x10001)
```

Use this key to create a self-signed certificate. Follow the usual prompts for creating a certificate:

```
# openssl req -new -x509 -days 365 -key cert.key -out cert.crt
You are about to be asked to enter information that will be ir
into your certificate request.
What you are about to enter is what is called a Distinguished
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) [AU]:US
State or Province Name (full name) [Some-State]:PA
```

```
Locality Name (eg, city) []:Pittsburgh
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My
Organizational Unit Name (eg, section) []:Systems Administrato
Common Name (e.g. server FQDN or YOUR name) []:localhost.examp
Email Address []:trhodes@FreeBSD.org
```

This will create two new files in the current directory: a private key file `cert.key`, and the certificate itself, `cert.crt`. These should be placed in a directory, preferably under `/etc/ssl/`, which is readable only by `root`. Permissions of `0700` are appropriate for these files and can be set using `chmod`.

13.6.2. Using Certificates

One use for a certificate is to encrypt connections to the Sendmail mail server in order to prevent the use of clear text authentication.

> **Note:**
> Some mail clients will display an error if the user has not installed a local copy of the certificate. Refer to the documentation included with the software for more information on certificate installation.

In FreeBSD 10.0-RELEASE and above, it is possible to create a self-signed certificate for Sendmail automatically. To enable this, add the following lines to `/etc/rc.conf`:

```
sendmail_enable="YES"
sendmail_cert_create="YES"
sendmail_cert_cn="localhost.example.org"
```

This will automatically create a self-signed certificate, `/etc/mail/certs/host.cert`, a signing key, `/etc/mail/certs/host.key`, and a CA certificate, `/etc/mail/certs/cacert.pem`. The certificate will use the `Common Name` specified in `sendmail_cert_cn`. After saving the edits, restart Sendmail:

```
# service sendmail restart
```

If all went well, there will be no error messages in `/var/log/maillog`. For a simple test, connect to the mail server's listening port using `telnet`:

```
# telnet example.com 25
Trying 192.0.34.166...
Connected to example.com.
```

```
Escape character is '^]'.
220 example.com ESMTP Sendmail 8.14.7/8.14.7; Fri, 18 Apr 2014
ehlo example.com
250-example.com Hello example.com [192.0.34.166], pleased to m
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-STARTTLS
250-DELIVERBY
250 HELP
quit
221 2.0.0 example.com closing connection
Connection closed by foreign host.
```

If the `STARTTLS` line appears in the output, everything is working correctly.

Prev 13.5. Kerberos | Up Home | Next 13.7. VPN over IPsec

Questions that are not answered by the documentation may be sent to <freebsd-questions@FreeBSD.org>.
Send questions about this document to <freebsd-doc@FreeBSD.org>.