NAME

SSL_CTX_set_cert_verify_callback - set peer certificate verification procedure

SYNOPSIS

```
 #include <openssl/ssl.h>

 void SSL_CTX_set_cert_verify_callback(SSL_CTX *ctx,
                                       int (*callback)(X509_STORE_CTX *, void *),
                                       void *arg);
```

DESCRIPTION

SSL_CTX_set_cert_verify_callback() sets the verification callback function for *ctx*. SSL objects that are created from *ctx* inherit the setting valid at the time when **SSL_new**(3) is called.

NOTES

When a peer certificate has been received during a SSL/TLS handshake, a verification function is called regardless of the verification mode. If the application does not explicitly specify a verification callback function, the built-in verification function is used. If a verification callback *callback* is specified via **SSL_CTX_set_cert_verify_callback()**, the supplied callback function is called instead with the arguments callback(X509_STORE_CTX *x509_store_ctx, void *arg). The argument *arg* is specified by the application when setting *callback*. By setting *callback* to NULL, the default behaviour is restored.

callback should return 1 to indicate verification success and 0 to indicate verification failure. In server mode, a return value of 0 leads to handshake failure. In client mode, the behaviour is as follows. All values, including 0, are ignored if the verification mode is **SSL_VERIFY_NONE**. Otherwise, when the return value is less than or equal to 0, the handshake will fail.

In client mode *callback* may also call the **SSL_set_retry_verify**(3) function on the **SSL** object set in the *x509_store_ctx* ex data (see **SSL_get_ex_data_X509_STORE_CTX_idx**(3)) and return 1. This would be typically done in case the certificate verification was not yet able to succeed. This makes the handshake suspend and return control to the calling application with **SSL_ERROR_WANT_RETRY_VERIFY**. The app can for instance fetch further certificates or cert status information needed for the verification. Calling **SSL_connect**(3) again resumes the connection attempt by retrying the server certificate verification step. This process may even be repeated if need be.

In any case a viable verification result value must be reflected in the **error** member of *x509_store_ctx*, which can be done using **X509_STORE_CTX_set_error**(3). This is particularly important in case the *callback* allows the connection to continue (by returning 1). Note that the verification status in the store context is a possibly durable indication of the chain's validity! This gets recorded in the SSL session (and thus also in session tickets) and the validity of the originally presented chain is then visible on resumption, even though no chain is presented int that case. Moreover, the calling application will be informed about the detailed result of the verification procedure and may elect to base further decisions on it.

Within *x509_store_ctx*, *callback* has access to the *verify_callback* function set using **SSL_CTX_set_verify**(3).

RETURN VALUES

SSL_CTX_set_cert_verify_callback() does not return a value.

WARNINGS

Do not mix the verification callback described in this function with the **verify_callback** function called during the verification process. The latter is set using the **SSL_CTX_set_verify**(3) family of functions.

Providing a complete verification procedure including certificate purpose settings etc is a complex task. The built-in procedure is quite powerful and in most cases it should be sufficient to modify its behaviour using the **verify_callback** function.

BUGS

SSL_CTX_set_cert_verify_callback() does not provide diagnostic information.

SEE ALSO

ssl(7), **SSL_CTX_set_verify**(3), **X509_STORE_CTX_set_error**(3), **SSL_get_verify_result**(3), **SSL_set_retry_verify**(3), **SSL_CTX_load_verify_locations**(3)

COPYRIGHT