NAME

SSL_CTX_set_tlsext_use_srtp, SSL_set_tlsext_use_srtp, SSL_get_srtp_profiles, SSL_get_selected_srtp_profile - Configure and query SRTP support

SYNOPSIS

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

int SSL_CTX_set_tlsext_use_srtp(SSL_CTX *ctx, const char *profiles);
int SSL_set_tlsext_use_srtp(SSL *ssl, const char *profiles);

STACK_OF(SRTP_PROTECTION_PROFILE) *SSL_get_srtp_profiles(SSL *ssl);
SRTP_PROTECTION_PROFILE *SSL_get_selected_srtp_profile(SSL *s);
```

DESCRIPTION

SRTP is the Secure Real-Time Transport Protocol. OpenSSL implements support for the "use_srtp" DTLS extension defined in RFC5764. This provides a mechanism for establishing SRTP keying material, algorithms and parameters using DTLS. This capability may be used as part of an implementation that conforms to RFC5763. OpenSSL does not implement SRTP itself or RFC5763. Note that OpenSSL does not support the use of SRTP Master Key Identifiers (MKIs). Also note that this extension is only supported in DTLS. Any SRTP configuration will be ignored if a TLS connection is attempted.

An OpenSSL client wishing to send the "use_srtp" extension should call **SSL_CTX_set_tlsext_use_srtp()** to set its use for all SSL objects subsequently created from an SSL_CTX. Alternatively a client may call **SSL_set_tlsext_use_srtp()** to set its use for an individual SSL object. The **profiles** parameters should point to a NUL-terminated, colon delimited list of SRTP protection profile names.

The currently supported protection profile names are:

SRTP_AES128_CM_SHA1_80
: This corresponds to SRTP_AES128_CM_HMAC_SHA1_80 defined in RFC5764.

SRTP_AES128_CM_SHA1_32
: This corresponds to SRTP_AES128_CM_HMAC_SHA1_32 defined in RFC5764.

SRTP_AEAD_AES_128_GCM
: This corresponds to the profile of the same name defined in RFC7714.

SRTP_AEAD_AES_256_GCM

This corresponds to the profile of the same name defined in RFC7714.

Supplying an unrecognised protection profile name will result in an error.

An OpenSSL server wishing to support the "use_srtp" extension should also call **SSL_CTX_set_tlsext_use_srtp()** or **SSL_set_tlsext_use_srtp()** to indicate the protection profiles that it is willing to negotiate.

The currently configured list of protection profiles for either a client or a server can be obtained by calling **SSL_get_srtp_profiles()**. This returns a stack of SRTP_PROTECTION_PROFILE objects. The memory pointed to in the return value of this function should not be freed by the caller.

After a handshake has been completed the negotiated SRTP protection profile (if any) can be obtained (on the client or the server) by calling **SSL_get_selected_srtp_profile()**. This function will return NULL if no SRTP protection profile was negotiated. The memory returned from this function should not be freed by the caller.

If an SRTP protection profile has been successfully negotiated then the SRTP keying material (on both the client and server) should be obtained via a call to **SSL_export_keying_material**(3). This call should provide a label value of "EXTRACTOR-dtls_srtp" and a NULL context value (use_context is 0). The total length of keying material obtained should be equal to two times the sum of the master key length and the salt length as defined for the protection profile in use. This provides the client write master key, the server write master key, the client write master salt and the server write master salt in that order.

RETURN VALUES

SSL_CTX_set_tlsext_use_srtp() and **SSL_set_tlsext_use_srtp()** return 0 on success or 1 on error.

SSL_get_srtp_profiles() returns a stack of SRTP_PROTECTION_PROFILE objects on success or NULL on error or if no protection profiles have been configured.

SSL_get_selected_srtp_profile() returns a pointer to an SRTP_PROTECTION_PROFILE object if one has been negotiated or NULL otherwise.

SEE ALSO

ssl(7), **SSL_export_keying_material**(3)

COPYRIGHT