NAME

SSL_CTX_set_tlsext_ticket_key_evp_cb, SSL_CTX_set_tlsext_ticket_key_cb - set a callback for session ticket processing

SYNOPSIS

```
 #include <openssl/tls1.h>

 int SSL_CTX_set_tlsext_ticket_key_evp_cb(SSL_CTX sslctx,
     int (*cb)(SSL *s, unsigned char key_name[16],
               unsigned char iv[EVP_MAX_IV_LENGTH],
               EVP_CIPHER_CTX *ctx, EVP_MAC_CTX *hctx, int enc));
```

The following function has been deprecated since OpenSSL 3.0, and can be hidden entirely by defining **OPENSSL_API_COMPAT** with a suitable version value, see **openssl_user_macros**(7):

```
 int SSL_CTX_set_tlsext_ticket_key_cb(SSL_CTX sslctx,
     int (*cb)(SSL *s, unsigned char key_name[16],
               unsigned char iv[EVP_MAX_IV_LENGTH],
               EVP_CIPHER_CTX *ctx, HMAC_CTX *hctx, int enc));
```

DESCRIPTION

SSL_CTX_set_tlsext_ticket_key_evp_cb() sets a callback function *cb* for handling session tickets for the ssl context *sslctx*. Session tickets, defined in RFC5077 provide an enhanced session resumption capability where the server implementation is not required to maintain per session state. It only applies to TLS and there is no SSLv3 implementation.

The callback function *cb* will be called for every client instigated TLS session when session ticket extension is presented in the TLS hello message. It is the responsibility of this function to create or retrieve the cryptographic parameters and to maintain their state.

The OpenSSL library uses your callback function to help implement a common TLS ticket construction state according to RFC5077 Section 4 such that per session state is unnecessary and a small set of cryptographic variables needs to be maintained by the callback function implementation.

In order to reuse a session, a TLS client must send the a session ticket extension to the server. The client can only send exactly one session ticket. The server, through the callback function, either agrees to reuse the session ticket information or it starts a full TLS handshake to create a new session ticket.

Before the callback function is started *ctx* and *hctx* have been initialised with **EVP_CIPHER_CTX_reset**(3) and **EVP_MAC_CTX_new**(3) respectively.

For new sessions tickets, when the client doesn't present a session ticket, or an attempted retrieval of the ticket failed, or a renew option was indicated, the callback function will be called with *enc* equal to 1. The OpenSSL library expects that the function will set an arbitrary *name*, initialize *iv*, and set the cipher context *ctx* and the hash context *hctx*.

The *name* is 16 characters long and is used as a key identifier.

The *iv* length is the length of the IV of the corresponding cipher. The maximum IV length is **EVP_MAX_IV_LENGTH** bytes defined in <*openssl/evp.h*>.

The initialization vector *iv* should be a random value. The cipher context *ctx* should use the initialisation vector *iv*. The cipher context can be set using **EVP_EncryptInit_ex**(3). The hmac context and digest can be set using **EVP_MAC_CTX_set_params**(3) with the **OSSL_MAC_PARAM_KEY** and **OSSL_MAC_PARAM_DIGEST** parameters respectively.

When the client presents a session ticket, the callback function with be called with *enc* set to 0 indicating that the *cb* function should retrieve a set of parameters. In this case *name* and *iv* have already been parsed out of the session ticket. The OpenSSL library expects that the *name* will be used to retrieve a cryptographic parameters and that the cryptographic context *ctx* will be set with the retrieved parameters and the initialization vector *iv*. using a function like **EVP_DecryptInit_ex**(3). The key material and digest for *hctx* need to be set using **EVP_MAC_CTX_set_params**(3) with the **OSSL_MAC_PARAM_KEY** and **OSSL_MAC_PARAM_DIGEST** parameters respectively.

If the *name* is still valid but a renewal of the ticket is required the callback function should return 2. The library will call the callback again with an argument of enc equal to 1 to set the new ticket.

The return value of the *cb* function is used by OpenSSL to determine what further processing will occur. The following return values have meaning:

2 This indicates that the *ctx* and *hctx* have been set and the session can continue on those parameters. Additionally it indicates that the session ticket is in a renewal period and should be replaced. The OpenSSL library will call *cb* again with an enc argument of 1 to set the new ticket (see RFC5077 3.3 paragraph 2).

1 This indicates that the *ctx* and *hctx* have been set and the session can continue on those parameters.

0 This indicates that it was not possible to set/retrieve a session ticket and the SSL/TLS session will continue by negotiating a set of cryptographic parameters or using the alternate SSL/TLS resumption mechanism, session ids.

If called with enc equal to 0 the library will call the *cb* again to get a new set of parameters.

less than 0
This indicates an error.

The **SSL_CTX_set_tlsext_ticket_key_cb()** function is identical to **SSL_CTX_set_tlsext_ticket_key_evp_cb()** except that it takes a deprecated HMAC_CTX pointer instead of an EVP_MAC_CTX one. Before this callback function is started *hctx* will have been initialised with **EVP_MAC_CTX_new**(3) and the digest set with **EVP_MAC_CTX_set_params**(3). The *hctx* key material can be set using **HMAC_Init_ex**(3).

NOTES

Session resumption shortcuts the TLS so that the client certificate negotiation don't occur. It makes up for this by storing client certificate an all other negotiated state information encrypted within the ticket. In a resumed session the applications will have all this state information available exactly as if a full negotiation had occurred.

If an attacker can obtain the key used to encrypt a session ticket, they can obtain the master secret for any ticket using that key and decrypt any traffic using that session: even if the cipher suite supports forward secrecy. As a result applications may wish to use multiple keys and avoid using long term keys stored in files.

Applications can use longer keys to maintain a consistent level of security. For example if a cipher suite uses 256 bit ciphers but only a 128 bit ticket key the overall security is only 128 bits because breaking the ticket key will enable an attacker to obtain the session keys.

RETURN VALUES

Returns 1 to indicate the callback function was set and 0 otherwise.

EXAMPLES

Reference Implementation:

```
SSL_CTX_set_tlsext_ticket_key_evp_cb(SSL, ssl_tlsext_ticket_key_cb);
...

static int ssl_tlsext_ticket_key_cb(SSL *s, unsigned char key_name[16],
                                    unsigned char *iv, EVP_CIPHER_CTX *ctx,
                                    EVP_MAC_CTX *hctx, int enc)
{
    OSSL_PARAM params[3];
```

```
your_type_t *key; /* something that you need to implement */

if (enc) { /* create new session */
    if (RAND_bytes(iv, EVP_MAX_IV_LENGTH) <= 0)
        return -1; /* insufficient random */

    key = currentkey(); /* something that you need to implement */
    if (key == NULL) {
        /* current key doesn't exist or isn't valid */
        key = createkey(); /*
                            * Something that you need to implement.
                            * createkey needs to initialise a name,
                            * an aes_key, a hmac_key and optionally
                            * an expire time.
                            */
        if (key == NULL) /* key couldn't be created */
            return 0;
    }
    memcpy(key_name, key->name, 16);

    if (EVP_EncryptInit_ex(&ctx, EVP_aes_256_cbc(), NULL, key->aes_key,
                           iv) == 0)
       return -1; /* error in cipher initialisation */

    params[0] = OSSL_PARAM_construct_octet_string(OSSL_MAC_PARAM_KEY,
                                                  key->hmac_key, 32);
    params[1] = OSSL_PARAM_construct_utf8_string(OSSL_MAC_PARAM_DIGEST,
                                                 "sha256", 0);
    params[2] = OSSL_PARAM_construct_end();
    if (EVP_MAC_CTX_set_params(hctx, params) == 0)
       return -1; /* error in mac initialisation */

    return 1;

} else { /* retrieve session */
    time_t t = time(NULL);
    key = findkey(key_name); /* something that you need to implement */

    if (key == NULL || key->expire < t)
        return 0;
        params[0] = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_KEY,
                                                      key->hmac_key, 32);
        params[1] = OSSL_PARAM_construct_utf8_string(OSSL_MAC_PARAM_DIGEST,
                                                     "sha256", 0);
        params[2] = OSSL_PARAM_construct_end();
        if (EVP_MAC_CTX_set_params(hctx, params) == 0)
            return -1; /* error in mac initialisation */

        if (EVP_DecryptInit_ex(&ctx, EVP_aes_256_cbc(), NULL, key->aes_key,
                               iv) == 0)
            return -1; /* error in cipher initialisation */

        if (key->expire < t - RENEW_TIME) { /* RENEW_TIME: implement */
            /*
             * return 2 - This session will get a new ticket even though the
             * current one is still valid.
             */
            return 2;
        }
        return 1;
    }
}
```

SEE ALSO

ssl(7), **SSL_set_session**(3), **SSL_session_reused**(3), **SSL_CTX_add_session**(3), **SSL_CTX_sess_number**(3), **SSL_CTX_sess_set_get_cb**(3), **SSL_CTX_set_session_id_context**(3),

HISTORY

The **SSL_CTX_set_tlsext_ticket_key_cb()** function was deprecated in OpenSSL 3.0.

The **SSL_CTX_set_tlsext_ticket_key_evp_cb()** function was introduced in OpenSSL 3.0.

COPYRIGHT