NAME

SSL_get_client_random, SSL_get_server_random, SSL_SESSION_get_master_key, SSL_SESSION_set1_master_key - get internal TLS/SSL random values and get/set master key

SYNOPSIS

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

size_t SSL_get_client_random(const SSL *ssl, unsigned char *out, size_t outlen);
size_t SSL_get_server_random(const SSL *ssl, unsigned char *out, size_t outlen);
size_t SSL_SESSION_get_master_key(const SSL_SESSION *session,
                                  unsigned char *out, size_t outlen);
int SSL_SESSION_set1_master_key(SSL_SESSION *sess, const unsigned char *in,
                                size_t len);
```

DESCRIPTION

SSL_get_client_random() extracts the random value sent from the client to the server during the initial SSL/TLS handshake. It copies as many bytes as it can of this value into the buffer provided in **out**, which must have at least **outlen** bytes available. It returns the total number of bytes that were actually copied. If **outlen** is zero, **SSL_get_client_random()** copies nothing, and returns the total size of the client_random value.

SSL_get_server_random() behaves the same, but extracts the random value sent from the server to the client during the initial SSL/TLS handshake.

SSL_SESSION_get_master_key() behaves the same, but extracts the master secret used to guarantee the security of the SSL/TLS session. This one can be dangerous if misused; see NOTES below.

SSL_SESSION_set1_master_key() sets the master key value associated with the SSL_SESSION **sess**. For example, this could be used to set up a session based PSK (see **SSL_CTX_set_psk_use_session_callback**(3)). The master key of length **len** should be provided at **in**. The supplied master key is copied by the function, so the caller is responsible for freeing and cleaning any memory associated with **in**. The caller must ensure that the length of the key is suitable for the ciphersuite associated with the SSL_SESSION.

NOTES

You probably shouldn’t use these functions.

These functions expose internal values from the TLS handshake, for use in low-level protocols. You probably should not use them, unless you are implementing something that needs access to the internal protocol details.

Despite the names of **SSL_get_client_random()** and **SSL_get_server_random()**, they ARE NOT random number generators. Instead, they return the mostly-random values that were already generated and used in the TLS protocol. Using them in place of **RAND_bytes()** would be grossly foolish.

The security of your TLS session depends on keeping the master key secret: do not expose it, or any information about it, to anybody. If you need to calculate another secret value that depends on the master secret, you should probably use **SSL_export_keying_material()** instead, and forget that you ever saw these functions.

In current versions of the TLS protocols, the length of client_random (and also server_random) is always SSL3_RANDOM_SIZE bytes. Support for other outlen arguments to the SSL_get_*_**random()** functions is provided in case of the unlikely event that a future version or variant of TLS uses some other length there.

Finally, though the "client_random" and "server_random" values are called "random", many TLS implementations will generate four bytes of those values based on their view of the current time.

RETURN VALUES

SSL_SESSION_set1_master_key() returns 1 on success or 0 on failure.

For the other functions, if **outlen** is greater than 0 then these functions return the number of bytes actually copied, which will be less than or equal to **outlen**. If **outlen** is 0 then these functions return the maximum number of bytes they would copy -- that is, the length of the underlying field.

SEE ALSO

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

COPYRIGHT