NAME

SSL_CTX_set_security_level, SSL_set_security_level, SSL_CTX_get_security_level, SSL_get_security_level, SSL_CTX_set_security_callback, SSL_set_security_callback, SSL_CTX_get_security_callback, SSL_get_security_callback, SSL_CTX_set0_security_ex_data, SSL_set0_security_ex_data, SSL_CTX_get0_security_ex_data, SSL_get0_security_ex_data - SSL/TLS security framework

SYNOPSIS

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

void SSL_CTX_set_security_level(SSL_CTX *ctx, int level);
void SSL_set_security_level(SSL *s, int level);

int SSL_CTX_get_security_level(const SSL_CTX *ctx);
int SSL_get_security_level(const SSL *s);

void SSL_CTX_set_security_callback(SSL_CTX *ctx,
                                   int (*cb)(SSL *s, SSL_CTX *ctx, int op,
                                             int bits, int nid,
                                             void *other, void *ex));

void SSL_set_security_callback(SSL *s, int (*cb)(SSL *s, SSL_CTX *ctx, int op,
                                                 int bits, int nid,
                                                 void *other, void *ex));

int (*SSL_CTX_get_security_callback(const SSL_CTX *ctx))(SSL *s, SSL_CTX *ctx, int op,
                                                         int bits, int nid, void *other,
                                                         void *ex);
int (*SSL_get_security_callback(const SSL *s))(SSL *s, SSL_CTX *ctx, int op,
                                               int bits, int nid, void *other,
                                               void *ex);

void SSL_CTX_set0_security_ex_data(SSL_CTX *ctx, void *ex);
void SSL_set0_security_ex_data(SSL *s, void *ex);

void *SSL_CTX_get0_security_ex_data(const SSL_CTX *ctx);
void *SSL_get0_security_ex_data(const SSL *s);
```

DESCRIPTION

The functions **SSL_CTX_set_security_level()** and **SSL_set_security_level()** set the security level to

level. If not set the library default security level is used.

The functions **SSL_CTX_get_security_level()** and **SSL_get_security_level()** retrieve the current security level.

SSL_CTX_set_security_callback(), **SSL_set_security_callback()**, **SSL_CTX_get_security_callback()** and **SSL_get_security_callback()** get or set the security callback associated with **ctx** or **s**. If not set a default security callback is used. The meaning of the parameters and the behaviour of the default callbacks is described below.

SSL_CTX_set0_security_ex_data(), **SSL_set0_security_ex_data()**, **SSL_CTX_get0_security_ex_data()** and **SSL_get0_security_ex_data()** set the extra data pointer passed to the **ex** parameter of the callback. This value is passed to the callback verbatim and can be set to any convenient application specific value.

DEFAULT CALLBACK BEHAVIOUR

If an application doesn't set its own security callback the default callback is used. It is intended to provide sane defaults. The meaning of each level is described below.

Level 0

Everything is permitted. This retains compatibility with previous versions of OpenSSL.

Level 1

The security level corresponds to a minimum of 80 bits of security. Any parameters offering below 80 bits of security are excluded. As a result RSA, DSA and DH keys shorter than 1024 bits and ECC keys shorter than 160 bits are prohibited. All export cipher suites are prohibited since they all offer less than 80 bits of security. SSL version 2 is prohibited. Any cipher suite using MD5 for the MAC is also prohibited. Note that signatures using SHA1 and MD5 are also forbidden at this level as they have less than 80 security bits.

Level 2

Security level set to 112 bits of security. As a result RSA, DSA and DH keys shorter than 2048 bits and ECC keys shorter than 224 bits are prohibited. In addition to the level 1 exclusions any cipher suite using RC4 is also prohibited. SSL version 3 is also not allowed. Compression is disabled.

Level 3

Security level set to 128 bits of security. As a result RSA, DSA and DH keys shorter than 3072 bits and ECC keys shorter than 256 bits are prohibited. In addition to the level 2 exclusions cipher suites not offering forward secrecy are prohibited. TLS versions below 1.1 are not permitted.

Session tickets are disabled.

Level 4

Security level set to 192 bits of security. As a result RSA, DSA and DH keys shorter than 7680 bits and ECC keys shorter than 384 bits are prohibited. Cipher suites using SHA1 for the MAC are prohibited. TLS versions below 1.2 are not permitted.

Level 5

Security level set to 256 bits of security. As a result RSA, DSA and DH keys shorter than 15360 bits and ECC keys shorter than 512 bits are prohibited.

APPLICATION DEFINED SECURITY CALLBACKS

Documentation to be provided.

NOTES

The default security level can be configured when OpenSSL is compiled by setting **-DOPENSSL_TLS_SECURITY_LEVEL=level**. If not set then 1 is used.

The security framework disables or reject parameters inconsistent with the set security level. In the past this was difficult as applications had to set a number of distinct parameters (supported ciphers, supported curves supported signature algorithms) to achieve this end and some cases (DH parameter size for example) could not be checked at all.

By setting an appropriate security level much of this complexity can be avoided.

The bits of security limits affect all relevant parameters including cipher suite encryption algorithms, supported ECC curves, supported signature algorithms, DH parameter sizes, certificate key sizes and signature algorithms. This limit applies no matter what other custom settings an application has set: so if the cipher suite is set to **ALL** then only cipher suites consistent with the security level are permissible.

See SP800-57 for how the security limits are related to individual algorithms.

Some security levels require large key sizes for non-ECC public key algorithms which can severely degrade performance. For example 256 bits of security requires the use of RSA keys of at least 15360 bits in size.

Some restrictions can be gracefully handled: for example cipher suites offering insufficient security are not sent by the client and will not be selected by the server. Other restrictions such as the peer certificate key size or the DH parameter size will abort the handshake with a fatal alert.

Attempts to set certificates or parameters with insufficient security are also blocked. For example trying to set a certificate using a 512 bit RSA key or a certificate with a signature with SHA1 digest at level 1 using **SSL_CTX_use_certificate()**. Applications which do not check the return values for errors will misbehave: for example it might appear that a certificate is not set at all because it had been rejected.

RETURN VALUES

SSL_CTX_set_security_level() and **SSL_set_security_level()** do not return values.

SSL_CTX_get_security_level() and **SSL_get_security_level()** return a integer that represents the security level with **SSL_CTX** or **SSL**, respectively.

SSL_CTX_set_security_callback() and **SSL_set_security_callback()** do not return values.

SSL_CTX_get_security_callback() and **SSL_get_security_callback()** return the pointer to the security callback or NULL if the callback is not set.

SSL_CTX_get0_security_ex_data() and **SSL_get0_security_ex_data()** return the extra data pointer or NULL if the ex data is not set.

SEE ALSO

ssl(7)

HISTORY

These functions were added in OpenSSL 1.1.0.

COPYRIGHT