NAME

SSL_CTX_set_options, SSL_set_options, SSL_CTX_clear_options, SSL_clear_options, SSL_CTX_get_options, SSL_get_options, SSL_get_secure_renegotiation_support - manipulate SSL options

SYNOPSIS

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

long SSL_CTX_set_options(SSL_CTX *ctx, long options);
long SSL_set_options(SSL *ssl, long options);

long SSL_CTX_clear_options(SSL_CTX *ctx, long options);
long SSL_clear_options(SSL *ssl, long options);

long SSL_CTX_get_options(SSL_CTX *ctx);
long SSL_get_options(SSL *ssl);

long SSL_get_secure_renegotiation_support(SSL *ssl);
```

DESCRIPTION

SSL_CTX_set_options() adds the options set via bitmask in **options** to **ctx**. Options already set before are not cleared!

SSL_set_options() adds the options set via bitmask in **options** to **ssl**. Options already set before are not cleared!

SSL_CTX_clear_options() clears the options set via bitmask in **options** to **ctx**.

SSL_clear_options() clears the options set via bitmask in **options** to **ssl**.

SSL_CTX_get_options() returns the options set for **ctx**.

SSL_get_options() returns the options set for **ssl**.

SSL_get_secure_renegotiation_support() indicates whether the peer supports secure renegotiation. Note, this is implemented via a macro.

NOTES

The behaviour of the SSL library can be changed by setting several options. The options are coded as bitmasks and can be combined by a bitwise **or** operation (|).

SSL_CTX_set_options() and *SSL_set_options()* affect the (external) protocol behaviour of the SSL library. The (internal) behaviour of the API can be changed by using the similar *SSL_CTX_set_mode(3)* and *SSL_set_mode()* functions.

During a handshake, the option settings of the SSL object are used. When a new SSL object is created from a context using *SSL_new()*, the current option setting is copied. Changes to **ctx** do not affect already created SSL objects. *SSL_clear()* does not affect the settings.

The following **bug workaround** options are available:

SSL_OP_SSLREF2_REUSE_CERT_TYPE_BUG

...

SSL_OP_MICROSOFT_BIG_SSLV3_BUFFER

...

SSL_OP_SAFARI_ECDHE_ECDSA_BUG

Don’t prefer ECDHE-ECDSA ciphers when the client appears to be Safari on OS X. OS X 10.8..10.8.3 has broken support for ECDHE-ECDSA ciphers.

SSL_OP_SSLEAY_080_CLIENT_DH_BUG

...

SSL_OP_TLS_D5_BUG
: ...

SSL_OP_DONT_INSERT_EMPTY_FRAGMENTS
: Disables a countermeasure against a SSL 3.0/TLS 1.0 protocol vulnerability affecting CBC ciphers, which cannot be handled by some broken SSL implementations. This option has no effect for connections using other ciphers.

SSL_OP_TLSEXT_PADDING
: Adds a padding extension to ensure the ClientHello size is never between 256 and 511 bytes in length. This is needed as a workaround for some implementations.

SSL_OP_ALL
: All of the above bug workarounds.

It is usually safe to use **SSL_OP_ALL** to enable the bug workaround options if compatibility with somewhat broken implementations is desired.

The following **modifying** options are available:

SSL_OP_TLS_ROLLBACK_BUG
: Disable version rollback attack detection.

  During the client key exchange, the client must send the same information about acceptable SSL/TLS protocol levels as during the first hello. Some clients violate this rule by adapting to the server's answer. (Example: the client sends a SSLv2 hello and accepts up to SSLv3.1=TLSv1, the server only understands up to SSLv3. In this case the client must still use the same SSLv3.1=TLSv1 announcement. Some clients step down to SSLv3 with respect to the server's answer and violate the version rollback protection.)

SSL_OP_SINGLE_DH_USE
: Always create a new key when using temporary/ephemeral DH parameters (see *SSL_CTX_set_tmp_dh_callback(3)*). This option must be used to prevent small subgroup attacks, when the DH parameters were not generated using ''strong'' primes (e.g. when using DSA-parameters, see *dhparam(1)*). If ''strong'' primes were used, it is not strictly necessary to generate a new DH key during each handshake but it is also recommended. **SSL_OP_SINGLE_DH_USE** should therefore be enabled whenever temporary/ephemeral DH parameters are used.

SSL_OP_EPHEMERAL_RSA
: This option is no longer implemented and is treated as no op.

SSL_OP_CIPHER_SERVER_PREFERENCE
: When choosing a cipher, use the server's preferences instead of the client preferences. When not set, the SSL server will always follow the clients preferences. When set, the SSL/TLS server will choose following its own preferences.

SSL_OP_PKCS1_CHECK_1
: ...

SSL_OP_PKCS1_CHECK_2
: ...

SSL_OP_NO_SSLv3, SSL_OP_NO_TLSv1, SSL_OP_NO_TLSv1_1, SSL_OP_NO_TLSv1_2, SSL_OP_NO_DTLSv1, SSL_OP_NO_DTLSv1_2
: These options turn off the SSLv3, TLSv1, TLSv1.1 or TLSv1.2 protocol versions with TLS or the DTLSv1, DTLSv1.2 versions with DTLS, respectively. As of OpenSSL 1.1.0, these options are deprecated, use *SSL_CTX_set_min_proto_version(3)* and *SSL_CTX_set_max_proto_version(3)* instead.

SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION
: When performing renegotiation as a server, always start a new session (i.e., session resumption requests are only accepted in the initial handshake). This option is not needed for clients.

SSL_OP_NO_TICKET
: Normally clients and servers will, where possible, transparently make use of RFC4507bis tickets for stateless session resumption.

  If this option is set this functionality is disabled and tickets will not be used by clients or servers.

SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION
: Allow legacy insecure renegotiation between OpenSSL and unpatched clients or servers. See the **SECURE RENEGOTIATION** section for more details.

SSL_OP_LEGACY_SERVER_CONNECT
: Allow legacy insecure renegotiation between OpenSSL and unpatched servers **only**: this option is currently set by default. See the **SECURE RENEGOTIATION** section for more details.

SSL_OP_NO_ENCRYPT_THEN_MAC
: Normally clients and servers will transparently attempt to negotiate the RFC7366 Encrypt-then-MAC option on TLS and DTLS connection.

  If this option is set, Encrypt-then-MAC is disabled. Clients will not propose, and servers will not accept the extension.

SSL_OP_NO_RENEGOTIATION
: Disable all renegotiation in TLSv1.2 and earlier. Do not send HelloRequest messages, and ignore renegotiation requests via ClientHello.

SECURE RENEGOTIATION

OpenSSL always attempts to use secure renegotiation as described in RFC5746. This counters the prefix attack described in CVE-2009-3555 and elsewhere.

This attack has far reaching consequences which application writers should be aware of. In the description below an implementation supporting secure renegotiation is referred to as *patched*. A server not supporting secure renegotiation is referred to as *unpatched*.

The following sections describe the operations permitted by OpenSSL's secure renegotiation implementation.

Patched client and server

Connections and renegotiation are always permitted by OpenSSL implementations.

Unpatched client and patched OpenSSL server

The initial connection succeeds but client renegotiation is denied by the server with a **no_renegotiation** warning alert if TLS v1.0 is used or a fatal **handshake_failure** alert in SSL v3.0.

If the patched OpenSSL server attempts to renegotiate a fatal **handshake_failure** alert is sent. This is because the server code may be unaware of the unpatched nature of the client.

If the option **SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION** is set then renegotiation **always** succeeds.

Patched OpenSSL client and unpatched server.

If the option **SSL_OP_LEGACY_SERVER_CONNECT** or **SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION** is set then initial connections and renegotiation between patched OpenSSL clients and unpatched servers succeeds. If neither option is set then initial connections to unpatched servers will fail.

The option **SSL_OP_LEGACY_SERVER_CONNECT** is currently set by default even though it has security implications: otherwise it would be impossible to connect to unpatched servers (i.e. all of them initially) and this is clearly not acceptable. Renegotiation is permitted because this does not add any additional security issues: during an attack clients do not see any renegotiations anyway.

As more servers become patched the option **SSL_OP_LEGACY_SERVER_CONNECT** will **not** be set by default in a future version of OpenSSL.

OpenSSL client applications wishing to ensure they can connect to unpatched servers should always **set**

SSL_OP_LEGACY_SERVER_CONNECT

OpenSSL client applications that want to ensure they can **not** connect to unpatched servers (and thus avoid any security issues) should always **clear** **SSL_OP_LEGACY_SERVER_CONNECT** using *SSL_CTX_clear_options()* or *SSL_clear_options()*.

The difference between the **SSL_OP_LEGACY_SERVER_CONNECT** and **SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION** options is that **SSL_OP_LEGACY_SERVER_CONNECT** enables initial connections and secure renegotiation between OpenSSL clients and unpatched servers **only**, while **SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION** allows initial connections and renegotiation between OpenSSL and unpatched clients or servers.

RETURN VALUES

SSL_CTX_set_options() and *SSL_set_options()* return the new options bitmask after adding **options**.

SSL_CTX_clear_options() and *SSL_clear_options()* return the new options bitmask after clearing **options**.

SSL_CTX_get_options() and *SSL_get_options()* return the current bitmask.

SSL_get_secure_renegotiation_support() returns 1 is the peer supports secure renegotiation and 0 if it does not.

SEE ALSO

ssl(3), *SSL_new(3)*, *SSL_clear(3)*, *SSL_CTX_set_tmp_dh_callback(3)*, *SSL_CTX_set_min_proto_version(3)*, *dhparam(1)*

HISTORY

The attempt to always try to use secure renegotiation was added in Openssl 0.9.8m.

SSL_OP_NO_RENEGOTIATION was added in OpenSSL 1.1.0h.

COPYRIGHT