SSL MODE SEND FALLBACK SCSV

[TLS\_FALLBACK\_SCSV](http://tools.ietf.org/html/draft-bmoeller-tls-downgrade-scsv-02) is a TLS Signaling Cipher Suite Value (SCSV) that can be used to guard against protocol downgrade attacks. The extension can be useful for clients like web browsers, which fall back to a lesser protocol version if attempts to use a higher protocol version fail.

In the attack, the adversary would force a negotiation failure of a higher protocol like TLS 1.2 or 1.1 in hopes the client will retry with TLS 1.0 or SSLv3. Then, the adversary will use defects in the down level protocols to carry out other attacks. Table 1 below lists some of the losses that could occur when traversing down protocol versions.

|  |  |  |
| --- | --- | --- |
| **Protocols, Downgrades and Features** | | |
| **Desired Protocol** | **Downgraded Protocol** | **Loss on Downgrade** |
| TLS 1.2 | TLS 1.1 | [AEAD cipher suites](https://wiki.openssl.org/index.php/EVP_Authenticated_Encryption_and_Decryption) (CCM, GCM, etc) |
| TLS 1.1 | TLS 1.0 | Perfect Forward Secrecy (PFS) |
| TLS 1.0 | SSL v3 | Too many to list[1] |

[1] See [Differences Between SSLv2, SSLv3, and TLS](http://yaksman.org/~lweith/ssl.pdf) and [This POODLE Bites: Exploiting The SSL 3.0 Fallback](http://www.openssl.org/~bodo/ssl-poodle.pdf).

The SSL\_MODE\_SEND\_FALLBACK\_SCSV extension can be used to remediate the [POODLE bug](http://www.openssl.org/~bodo/ssl-poodle.pdf) by ensuring clients don't fall back to SSLv3 ***if*** the client performs fallbacks. However, the extension does not fix the underlying padding oracle. Rather, it just avoids the defective protocol version.

SCSV means TLS\_FALLBACK\_SCSV shows up as a cipher suite in the ClientHello. The spurious cipher is added to the cipher suites if the **SSL\_MODE\_SEND\_FALLBACK\_SCSV** option is present.

***Note Well***: if the client does not perform fallbacks, then the TLS\_FALLBACK\_SCSV extension is not needed.

**Contents**

 [[hide](https://wiki.openssl.org/index.php/SSL_MODE_SEND_FALLBACK_SCSV)]

* [1Reason for the Extension](https://wiki.openssl.org/index.php/SSL_MODE_SEND_FALLBACK_SCSV#Reason_for_the_Extension)
* [2Avoiding the Extension](https://wiki.openssl.org/index.php/SSL_MODE_SEND_FALLBACK_SCSV#Avoiding_the_Extension)
* [3Using the Extension](https://wiki.openssl.org/index.php/SSL_MODE_SEND_FALLBACK_SCSV#Using_the_Extension)
* [4Related Discussions](https://wiki.openssl.org/index.php/SSL_MODE_SEND_FALLBACK_SCSV#Related_Discussions)

Reason for the Extension[[edit](https://wiki.openssl.org/index.php?title=SSL_MODE_SEND_FALLBACK_SCSV&action=edit&section=1)]

Browsers and other similar software attempt to support every server created on {God|Allah|Brahman|...}'s green earth. If the browser attempts to connect to 1990s era server and the connection fails, then the browser will fallback to a lesser protocol. It falls back to a lesser protocol by initiating a new connection with the down level protocol.

Its curious why the browsers chose to compromise the security of the dominant/standard use cases for a trivial minority. The W3C clearly states two design principals: [Secure By Design](http://www.w3.org/TR/html-design-principles/#secure-by-design) and [Priority of Constituencies](http://www.w3.org/TR/html-design-principles/#priority-of-constituencies). When a browser falls back to insecure protocols when the user asks for a secure scheme, they violate both principals.

The browsers could have provided modern TLS support with 15 or 20 reasonable ciphers (or let the users choose their vanity ciphers); and then require a plug-in to achieve the existing insecure behavior. Browsers seem more than happy to require plug-ins for the most basic functionality, like opening a homepage (try to get [Chrome](http://productforums.google.com/forum/#!topic/chrome/JPLugnTBiMQ), [Firefox](http://support.mozilla.org/en-US/questions/962910) or [Opera](http://addons.opera.com/en/extensions/details/homepage-in-new-tab) to open your homepage on a new tab).

Even more befuddling, the SSL\_MODE\_SEND\_FALLBACK\_SCSV is not needed now that SSLv3 is insecure due to the padding oracle (as if Loern Weith was not clear that [SSLv3 was insecure in 2006](http://yaksman.org/~lweith/ssl.pdf)). In fact, the down level software that is part of the problem likely won't be updated to understand the TLS extension anyway.

Avoiding the Extension[[edit](https://wiki.openssl.org/index.php?title=SSL_MODE_SEND_FALLBACK_SCSV&action=edit&section=2)]

As stated earlier, SSL\_MODE\_SEND\_FALLBACK\_SCSV is ***not*** needed if the client does not perform fallbacks. The easiest way to avoid use of the SSL\_MODE\_SEND\_FALLBACK\_SCSV is to always specify the protocols you are willing to accept. The detail is you ***always*** send the highest protocol version with the ClientHello.

For example, suppose you want to accept TLS 1.0 through TLS 1.2. In this case, you send a ClientHello with that information, and you don't include SSL\_MODE\_SEND\_FALLBACK\_SCSV. Below is a portion of the code to do so (its likely similar to the code you have been using for years):

const SSL\_METHOD\* method = SSLv23\_method();

if(method == NULL)) handleFailure();

SSL\_CTX\* ctx = SSL\_CTX\_new(method);

if(ctx == NULL) handleFailure();

const long flags = SSL\_OP\_NO\_SSLv2 | SSL\_OP\_NO\_SSLv3 | SSL\_OP\_NO\_COMPRESSION;

SSL\_CTX\_set\_options(ctx, flags);

In the code above, the SSLv23\_method enables all protocol version - from SSLv2 through TLS 1.2. The method is then used to create a context object. After the context object is created, weak/wounded/broken protocols and options are removed by setting SSL\_OP\_NO\_SSLv2, SSL\_OP\_NO\_SSLv3 and SSL\_OP\_NO\_COMPRESSION.

The ClientHello will set a minimum protocol version of TLS 1.0 and a maximum protocol version of TLS 1.2 in the ClientHello. The TLS protocol ensures TLS 1.2 is used if available; and TLS 1.1 is used is TLS 1.2 is *not* available; and TLS 1.0 is used if TLS 1.2 and 1.1 are *not* available. During key exchange, the ClientHello is MAC'd and used (in part) to derive the premaster\_secret, so tampering with protocol versions will be detected.

When TLS 1.3 is added to OpenSSL, it will include TLS 1.3. You won't have to change your code.

Using the Extension[[edit](https://wiki.openssl.org/index.php?title=SSL_MODE_SEND_FALLBACK_SCSV&action=edit&section=3)]

If you need to use the extension, then the following is an example of how to use it:

const SSL\_METHOD\* method = TLSv1\_2\_method();

if(method == NULL)) handleFailure();

SSL\_CTX\* ctx = SSL\_CTX\_new(method);

if(ctx == NULL) handleFailure();

#ifdef SSL\_MODE\_SEND\_FALLBACK\_SCSV

SSL\_CTX\_set\_mode(ctx, SSL\_MODE\_SEND\_FALLBACK\_SCSV)

#endif

Some folks on the OpenSSL mailing list recommend using TLS\_FALLBACK\_SCSV and SSL\_MODE\_SEND\_FALLBACK\_SCSV whenever available. See, for example, [Use of TLS\_FALLBACK\_SCSV](http://www.mail-archive.com/openssl-users@openssl.org/msg75278.html) on the OpenSSL mailing list.

Related Discussions[[edit](https://wiki.openssl.org/index.php?title=SSL_MODE_SEND_FALLBACK_SCSV&action=edit&section=4)]

Below are some discussions that occurred on the OpenSSL and IETF mailing lists and around the web.

* [Context options and SSL\_MODE\_SEND\_FALLBACK\_SCSV](http://www.mail-archive.com/openssl-users@openssl.org/)
* [Use of TLS\_FALLBACK\_SCSV](http://www.mail-archive.com/openssl-users@openssl.org/)
* [Please document the new SSL\_MODE\_SEND\_FALLBACK\_SCSV](http://www.mail-archive.com/openssl-users@openssl.org/)
* [Working Group Last Call for draft-ietf​-tls-downg​rade-scsv-​00](http://www.ietf.org/mail-archive/web/tls/current/msg13713.html)
* [The TLS\_FALLBA​CK\_SCSV time bomb](http://www.ietf.org/mail-archive/web/tls/current/msg13945.html)
* [SSL is dead, long live TLS](http://blogs.akamai.com/2014/10/ssl-is-dead-long-live-tls.html)