NAME

ne_ssl_set_verify - register an SSL certificate verification callback

SYNOPSIS

#include <ne_session.h>

typedef int ne_ssl_verify_fn(void ****userdata*, int** *failures***, const ne_ssl_certificate ****cert***);**

void ne_ssl_set_verify(ne_session ****session*, ne_ssl_verify_fn** *verify_fn***, void ****userdata***);**

DESCRIPTION

To enable manual SSL certificate verification, a callback can be registered using **ne_ssl_set_verify**. If such a callback is not registered, when a connection is established to an SSL server which does not present a certificate signed by a trusted CA (see ne_ssl_trust_cert), or if the certificate presented is invalid in some way, the connection will fail.

When the callback is invoked, the *failures* parameter gives a bitmask indicating in what way the automatic certificate verification failed. The value is equal to the bit-wise OR of one or more of the following constants (and is guaranteed to be non-zero):

NE_SSL_NOTYETVALID

The certificate is not yet valid.

NE_SSL_EXPIRED

The certificate has expired.

NE_SSL_IDMISMATCH

The hostname used for the session does not match the hostname to which the certificate was issued.

NE_SSL_UNTRUSTED

The Certificate Authority which signed the certificate is not trusted.

Note that if either of the **NE_SSL_IDMISMATCH** or **NE_SSL_UNTRUSTED** failures is given, the connection may have been intercepted by a third party, and must not be presumed to be "secure".

The *cert* parameter passed to the callback represents the certificate which was presented by the server. If the server presented a chain of certificates, the chain can be accessed using ne_ssl_cert_signedby. The *cert* object given is not valid after the callback returns.

RETURN VALUE

The verification callback must return zero to indicate that the certificate should be trusted; and non-zero otherwise (in which case, the connection will fail).

EXAMPLES

The following code implements an example verification callback, using the **dump_cert** function from ne_ssl_cert_subject to display certification information. Notice that the hostname of the server used for the session is passed as the *userdata* parameter to the callback.

```
static int
my_verify(void *userdata, int failures, const ne_ssl_certificate *cert)
{
  const char *hostname = userdata;

  dump_cert(cert);

  puts("Certificate verification failed - the connection may have been "
       "intercepted by a third party!");

  if (failures & NE_SSL_IDMISMATCH) {
    const char *id = ne_ssl_cert_identity(cert);
    if (id)
      printf("Server certificate was issued to ’%s’ not ’%s’.\n",
             id, hostname);
    else
      printf("The certificate was not issued for ’%s’\n", hostname);
  }

  if (failures & NE_SSL_UNTRUSTED)
    puts("The certificate is not signed by a trusted Certificate Authority.");

  /* ... check for validity failures ... */

  if (prompt_user())
    return 1; /* fail verification */
  else
    return 0; /* trust the certificate anyway */
}

int
```

```
main(...)
{
  ne_session *sess = ne_session_create("https", "some.host.name", 443);
  ne_ssl_set_verify(sess, my_verify, "some.host.name");
  ...
}
```

SEE ALSO

ne_ssl_trust_cert, ne_ssl_readable_dname, ne_ssl_cert_subject

AUTHOR

Joe Orton

Author.

COPYRIGHT