Inviare Posta Elettronica in modalità sicura

SSM: swaks, Net::SSLeay and OPENSSL build

raffaele DOT granito @ tiscali DOT it – 23/04/2020 02:20

Premessa / Why?

Devo inviare mail tramite un server di posta che utilizza TLS.
Ho scelto di utilizzare il client perl swaks che mi permette di effettuare l'invio da shell.
Questo (swaks) a sua volta utilizza le OPENSSL (libssl e libcrypto) come implementazione di TLS, tramite la Net::SSLevy che fa da ponte tra PERL e LIBC.

Swaks (Perl) → Net::SSLeay (Perl) → OPENSSL (LIBC)

Normalmente OPENSSL è già installato sulle macchine UNIX (Solaris, HP-UX etc.) e GNU/Linux, e Net::SSLeay lo si trova in formati binari già pacchettizzato dalla propria distribuzione. Per cui normalmente per inviar mail TLS occorre semplicemente scaricare il pacchetto swaks ed eseguirlo con i giusti parametri (mail server, mittente, destinatario, body, intestazione) per. Su quasi tutte le macchine ho risolto velocemente.

Purtroppo capitano le eccezioni. Mi ritrovo una macchina IA64 con RHLinux5 con :

- ➔ Versioni OpenSSL 0.9.8 che non supporta TLSv1.2, e non aggiornabile perché versioni successive non sono state pacchettizzate
- ➔ Con una versione di Net::SSLeay sulla macchina che effettua staticamente i link a OpenSSL 0.9.8

Aggiungo, che non sono amministratore.

La soluzione è quella di effettuare il BUILD di OPENSSL e Net::SSLeay da utente non privilegiato. Di seguito la procedura.

I Paragrafi con il prefisso [HACKTIME] sono passi mentali miei per arrivare a capire alcune cose. Per una lettura veloce possono tranquillamente essere saltati, perché non descrivono la procedura di BUILD. Li ho relegati in fondo, negli ALLEGATI.

BUILD di OPENSSL

Ho scaricato il pacchetto sorgente da openssl.org

Vado nella HOME_SRC_PACKAGE (`/tmp/ssm/package/`) copierò tutti i pacchetti sorgenti in questa directory. Apro il pacchetto openssl e mi ci posiziono all'interno

```
cd /tmp/ssm/package/
tar xfvz openssl-1.0.2o.tar.gz
cd openssl-1.0.2o
```

Eseguo l'operazione di configure che al termine mi genererà il Makefile, necessario per la compilazione ed installazione. Imposto col parametro prefix la HOME_INSTALL (`/tmp/ssm/openssl-1.0.2o.linux-x8664-LIBC25.build`)

```
./config -fPIC \
--prefix=/tmp/ssm/openssl-1.0.2o.linux-x8664-LIBC25.build
```

Compilo, effettuo il test di funzionamento di tutte le funzionalità implementate dalle OpenSSL lib.

```
Make
Make test
```

Infine installo (copia dei binary nella HOME_INSTALLA definitia precedentemente con il parametro prefix.

```
Make install
```

Effettuo il packing di OPENSSL (openssl-1.0.2o.linux-x8664-LIBC25.build.tar.gz) che sposto nella mia cassetta degli attrezzi (repository SVN), mi potrà essere utile in altri contesti.

```
cd /tmp/ssm/
tar -czf openssl-1.0.2o.linux-x8664-LIBC25.build.tar.gz openssl-1.0.2o.linux-x8664-
LIBC25.build
```

Posso anche eliminare la directory dei sorgenti ed il pacchetto sorgente (una copia di quest'ultimo l'ho già copiato nel mio repository svn)

```
rm -rf /tmp/ssm/package/openssl-1.0.2o
rm /tmp/ssm/package/openssl-1.0.2o.tar.gz
```

Di OPENSSL lascio solamente la directory del build (`/tmp/ssm/openssl-1.0.2o.linux-x8664-LIBC25.build`).

```
/
|
‘--- /tmp/ssm <DIR>
        \
        ‘--- openssl-1.0.2o.linux-x8664-LIBC25.build <DIR>
        ‘--- package <DIR>   *EMPTY*
```

Terminato.

BUILD di NET::SSLeay

Ho scaricato la versione 1.88 (l'ultima trovata) dal CPAN, che è il repository ufficiale delle estensioni PERL all'indirizzo https://metacpan.org/pod/distribution/Net-SSLeay/lib/Net/SSLeay.pod

Vado nella solita HOME_SRC_PACKAGE (/tmp/ssm/package/). Apro il pacchetto Net::SSLeay e mi ci posiziono all'interno

```
cd /tmp/ssm/package/
tar xfvz Net-SSLeay-1.88.tar.gz
cd Net-SSLeay-1.88
```

Eseguo l'operazione di configure che al termine mi genererà il Makefile, necessario per la compilazione ed installazione. Imposto la variabile di ambiente OPENSSL_PREFIX per specificare dove si trova l'OPENSSL che voglio utilizzare nella fase di link. Invece non sono riuscito a capire come impostare la HOME_INSTALL per le Net::SSLeay, lo farò successivamnete andando a modificare direttamente il Makefile, generato con l'esecuzione di Makefile.PL

```
export OPENSSL_PREFIX=/tmp/ssm/openssl-1.0.2o.linux-x8664-LIBC25.build/
perl Makefile.PL
```

Vado a modificare il Makefile, in maniera tale da impostare la mia HOME_INSTALL. Di default andrebbe a scrivere sulle directory di sistema (`usr/lib64/perl5/site_perl/5.8.8/x86_64-linux-thread-multi`), cosa che non voglio e volendo non posso fare essendo io un utente non privileggiato per scrivere in quella posizione

Quindi modifico nel file Makefile le 3 variabili : SITEARCHEXP, INSTALLARCHLIB e INSTALLARCHLIB in questo modo :

➔ PRIMA delle MODIFICA

```
SITEARCHEXP = /usr/lib64/perl5/site_perl/5.8.8/x86_64-linux-thread-multi
INSTALLARCHLIB = /usr/lib64/perl5/5.8.8/x86_64-linux-thread-multi
INSTALLSITEARCH = /usr/lib64/perl5/site_perl/5.8.8/x86_64-linux-thread-multi
```

➔ DOPO la MODIFICA

```
SITEARCHEXP = /tmp/ssm/Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build
INSTALLARCHLIB = /tmp/ssm/Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build
INSTALLSITEARCH = /tmp/ssm/Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build
```

Compilo, effettuo il test di funzionamento di tutte le funzionalità implementate dalle OpenSSL lib mppate da Net::SSLeay.

```
Make
Make test
```

Infine installo (copia dei binary nella HOME_INSTALL definitia col setting delle variabili viste al passo precedente. Come è possibile notare l'impostazione della HOME_INSTALL è stata un po sporca ma funziona, e al momento va bene così. Successivamente indagherò per farlo in maniera pulita

```
Make install
```

```
Warning: You do not have permissions to install into /usr/lib/perl5/site_perl/5.8.8 at
/usr/lib/perl5/5.8.8/ExtUtils/Install.pm line 114.
Files found in blib/arch: installing files in blib/lib into architecture dependent
library tree
Installing /usr/share/man/man3/Net::SSLeay::Handle.3p
Installing /usr/share/man/man3/Net::SSLeay.3pm
Writing /tmp/ssm/Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build/auto/Net/SSLeay/.packlist
Appending installation info to /tmp/ssm/Net-SSLeay-1.88.IA64-LIBC25-
GNULinux.build/perllocal.pod
```

Effettuo il packing di Net::SSLeay (`Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build.tar.gz`) che sposto e conservo anche questa nella cassetta degli attrezzi (repository SVN).

```
cd /tmp/ssm/
tar -czf Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build.tar.gz Net-SSLeay-1.88.IA64-
LIBC25-GNULinux.build
```

Posso anche eliminare la directory dei sorgenti ed il pacchetto sorgente (una copia di quest'ultimo l'ho già copiato nel mio repository svn)

```
rm -rf /tmp/ssm/package/Net-SSLeay-1.88
rm /tmp/ssm/package/Net-SSLeay-1.88.tar.gz
```

Di SSLeay lascio solamente la directory del build (`/tmp/ssm/Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build`).

```
/
|
`--- /tmp/ssm <DIR>
        \
        `--- openssl-1.0.2o.linux-x8664-LIBC25.build <DIR>
        `--- Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build <DIR>
        `--- package <DIR>   *EMPTY*
```

Terminato.

Installazione del client di posta SWAKS

Installate la librerie OPENSSL (implementazione di TLS) ed SSLeay per rimapparle da Perl, è necessario a questo punto installare il Client Perl minimale per inviar mail. La scelta obbligata è SWAKS

Tale script è reperibile al sito ufficiale dell'autore https://jetmore.org/john/code/swaks/

Ho reperito l'ultima versione disponibile, versionata 20170101.0 (swaks-20170101.0.tar.gz)

Ho copiato il file nella solita directory (/tmp/ssm/package). Chiaramente non va compilato essendo 100% interpretato PERL. Per installarlo è necessario semplicemente aprire il pacchetto ed estrapolare il file swaks.pl da copiare nella nostra solita HOME_INSTALL /tmp/ssm/.

```
cd /tmp/ssm/package
tar xfvz swaks-20170101.0.tar.gz
cp swaks-20170101.0/swaks /tmp/ssm/
```

Dopodichè cancello il pacchetto

```
rm -rf /tmp/ssm/package/swaks-20170101.0
rm /tmp/ssm/package/swaks-20170101.0.tar.gz
```

Lascio solamente lo script swaks

```
/
|
‘--- /tmp/ssm <DIR>
        \
        ‘--- openssl-1.0.2o.linux-x8664-LIBC25.build    <DIR>
        ‘--- Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build <DIR>
        ‘--- package                                     <DIR>   *EMPTY*
        ‘--- swaks
```

Terminato.

Creazione e Configurazione di SSM

Abbiamo tutto ciò che ci serve per inviar mail TLS, ci resta da mettere assieme i pezzi

Come prima cosa, ho creato il file ssm.sh nella home /tmp/ssm.
Questo script serve per agevolare la chiamata a swaks e ad impostare l'ENV corretto. Prevede 3 variabili di ambiente:
→**PERL5LIB** da impostare con la HOME_INSTALL di Net-SSLeay, in maniera tale che l'interprete perl e swaks utilizzi la nostra versione e non quella già preinstallata sulla macchina. Lascio la path relativa in manieta tale da poter pacchettizzare tutto SSM e spostare ovunque io voglia.
→**SMTPSeverIP** ed **SMTPServerPort** impostare (tra doppi apici) con IP (o FQDN) e TCP PORT del servizio di invio posta.

```
#!/bin/bash

# --=== sendmailScript v1.0 (auth:rgranito) ===---

HOMESSM=$(dirname "$0")

export PERL5LIB=$HOMESSM/Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build/

#--- parametri utente ----
SMTPServerIP="<MAILSERVER_IP|FQDN>"
SMTPServerPort="<MAILSERVER_PORT>"

#--- controllo parametri ---
if [ $# -lt 3 ]; then
   echo "formato: $0 <subject> <body--file> <credential--file> [<attachment>]"
   exit
fi

# home
HOMESSM=$(dirname "$0")

# subject
subject=$1

# body
if [ ! -f $2 ]; then
    echo "err. <body--file> $2 not found!"
    exit
fi
body=$2 # file

# credential
if [ ! -f $3 ]; then
    echo "err. <credential--file> $3 not found!"
    exit
fi
. $3

# attach -- optional
if [ $# -eq 4 ];then
attach=$4
fi

# --- Sendmail using swaks ---
dimensioneArrayemailToTab=10
```

```
for (( indice=0; indice<${dimensioneArrayemailToTab}; indice++ ));
do
    if [ "${emailToItem[$indice]}" != "" ]; then
       if [ $indice -eq 0 ]; then
          emailTo=${emailToItem[$indice]}
       else
          emailTo=$emailTo,${emailToItem[$indice]}
       fi
    fi

    if [ $indice -eq 10 ]; then
       echo "err. troppi destinatari (controllo antispam) -- max 10 destinatari"
       exit 1
    fi
done

echo "emailFrom $emailFrom"
# send secure mail with o within attach
if [ $# -eq 3 ];then
$HOMESSM/swaks.pl -tls \
--server $SMTPServerIP \
--port $SMTPServerPort \
--from $emailFrom \
--to $emailTo \
--auth \
--auth-user=$username \
--auth-password=$password \
--body $body \
--h-Subject $subject
elif [ $# -eq 4 ];then
$HOMESSM/swaks.pl -tls \
--server $SMTPServerIP \
--port $SMTPServerPort \
--from $emailFrom \
--to $emailTo \
--auth \
--auth-user=$username \
--auth-password=$password \
--body $body \
--h-Subject $subject \
--attach-type text/html \
--attach $attach
fi

#--- test esito
if [ $? -eq 0 ]; then
   echo send secure mail \(SUCCESS\)
else
   echo send secure mail \(FAIL retcode: $?\)
fi

exit
```

Lo ha il seguente formato:

```
muletto:/tmp/ssm>./ssm.sh
formato: ./ssm.sh <subject> <body--file> <credential--file> [<attachment>]
```

Il **<subject>** è l'oggetto della mail da inviare.
Il **<body—file>** è un file che dovrà contenere il corpo della mail da inviare
Per il **<credential—file>** va creato questo secondo file, che chiamo ssm.conf

E' possibile definire al massimo 10 destinatari. Una riga **emailToItem** con indice incrementato per ogni destinatario. **emailFrom** è il mittente della mail. **Username** e **password** son le credenziali con cui ci presentiamo al server di mail. Dovete riceverle dall'amministratore del servizio.

```
muletto:/tmp/ssm>cat ssm.conf
#.-------------------------------------------------------------.
#|                                                             |
#|  Sendmail / Parametri                                       |
#|                                                             |
#|  username        - user credenzali accesso server           |
#|  password        - password credenziali accesso server      |
#|  emailFrom       - casella di posta mittente                |
#|  emailToItem[0-9]- elenco destinatari (max 10)              |
#|                                                             |
#'-------------------------------------------------------------'
username="<USERID>"
password="<PASSWORD>"
emailFrom="<indirizzo-posta-mittente"
emailToItem[0]="<indirizzo-posta-destinatario>"
emailToItem[1]="<indirizzo-posta-destinatario>"
```

Opzionalmente, col parametro **<attachment>** è possibile specificare un file da allegare alla mail dsa inviare.

Lo script ssm.sh restituisce il returncode di swaks. Di seguito i codici di ritorno previsti per la versione di swaks utilizzata :

```
    EXIT CODES

    0   no errors occurred
    1   error parsing command line options
    2   error connecting to remote server
    3   unknown connection type
    4   while running with connection type of "pipe", fatal problem writing
        to or reading from the child process
    5   while running with connection type of "pipe", child process died
        unexpectedly. This can mean that the program specified with --pipe
        doesn't exist.
    6   Connection closed unexpectedly. If the close is detected in response
        to the 'QUIT' swaks sends following an unexpected response, the
        error code for that unexpected response is used instead. For
        instance, if a mail server returns a 550 response to a MAIL FROM:
        and then immediately closes the connection, swaks detects that the
        connection is closed, but uses the more specific exit code 23 to
        detail the nature of the failure. If instead the server return a 250
        code and then immediately closes the connection, swaks will use the
        exit code 6 because there is not a more specific exit code.
    10  error in prerequisites (needed module not available)
    21  error reading initial banner from server
    22  error in HELO transaction
    23  error in MAIL transaction
    24  no RCPTs accepted
```

```
    25  server returned error to DATA request
    26  server did not accept mail following data
    27  server returned error after normal-session quit request
    28  error in AUTH transaction
    29  error in TLS transaction
    30  PRDR requested/required but not advertised
    32  error in EHLO following TLS negotiation
    33  error in XCLIENT transaction
    34  error in EHLO following XCLIENT
    35  error in PROXY option processing
    36  error sending PROXY banner
```

Con la creazione di ssm.sh ed ssm.conf abbiamo nella nostra HOME (/tmp/ssm) la seguente situazione :

```
/
|
‘--- /tmp/ssm <DIR>
        \
        ‘--- openssl-1.0.2o.linux-x8664-LIBC25.build    <DIR>
        ‘--- Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build <DIR>
        ‘--- package                                     <DIR>   *EMPTY*
        ‘--- ssm.sh
        ‘--- ssm.conf
        ‘--- swaks
```

Siamo pronti per un primo test di invio mail.

Invio Mail – PRIMO TEST

Il mio server di posta sicura risponde al seguente indirizzo 192.168.0.140:25, ho inpostato quindi
Le credianziali sono TESTUSER/FAKEPASS

L'indirizzo mittente è raffaele.granito@home.local.
I destinatari sono paolo.granito@home.local e emilia.granito@home.local

Infine ho creato il file body.txt, in cui riportato il messaggio da inviare nel corpo della mail.

```
-sh-3.2$ echo “messaggio di prova” > body.txt
```

Ho modificato quindi lo script ssm.sh, impostando IP e Porta del server di posta da contattare. E' un dato che imposto una sola volta.

```
#!/bin/bash

# --=== sendmailScript v1.0 (auth:rgranito) ===---

HOMESSM=$(dirname "$0")

export PERL5LIB=$HOMESSM/Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build/

#--- parametri utente ----
SMTPServerIP="192.168.0.140"
SMTPServerPort="25"

#--- controllo parametri ---
```

Ed ho modificato il file di configurazione ssm.conf con credenziali di connessione, mittente e destinatari del messaggio. SSM è pensato per delle applicazioni che inviano messaggi di alert agli addetti IT quando avvengono particolare eventi di interesse, per cui mittente e destinatari son quasi fisse come le credenziali.

```
muletto:/tmp/ssm>cat ssm.conf
#.-----------------------------------------------------------.
#|                                                           |
#|  Sendmail / Parametri                                     |
#|                                                           |
#|  username        - user credenzali accesso server         |
#|  password        - password credenziali accesso server    |
#|  emailFrom       - casella di posta mittente              |
#|  emailToItem[0-9]- elenco destinatari (max 10)            |
#|                                                           |
#'-----------------------------------------------------------'
username="TESTUSER"
password="FAKEPASS"
emailFrom="raffaele.granito@home.local"
emailToItem[0]="paolo.granito@home.local"
emailToItem[1]="emilia.granito@home.local"
```

Eseguo il comando di invio mail

```
-sh-3.2$ ./ssm.sh test body.txt ssm.conf
emailFrom angelo.raffaele@local.local
=== Trying 192.168.0.140:25...
=== Connected to 192.168.0.140.
<-  220 smtp.home.local - TELSMTP02RM030
 -> EHLO muletto.home.local
<-  250-smtp.home.local Hello [192.168.0.76]
<-  250-SIZE 36700160
<-  250-PIPELINING
<-  250-DSN
<-  250-ENHANCEDSTATUSCODES
<-  250-STARTTLS
<-  250-AUTH NTLM
<-  250-8BITMIME
<-  250-BINARYMIME
<-  250 CHUNKING
 -> STARTTLS
<-  220 2.0.0 SMTP server ready
=== TLS started with cipher TLSv1.2:ECDHE-RSA-AES256-SHA384:256
=== TLS no local certificate set
=== TLS peer DN="/CN=Smtp.home.local/emailAddress=adminMail@home.local"
 ~> EHLO muletto.home.local
<~  250-smtp.home.local Hello [192.168.0.76]
<~  250-SIZE 36700160
<~  250-PIPELINING
<~  250-DSN
<~  250-ENHANCEDSTATUSCODES
<~  250-AUTH NTLM LOGIN
<~  250-8BITMIME
<~  250-BINARYMIME
<~  250 CHUNKING
 ~> AUTH LOGIN
<~  334 VXNlqewqwbWU6
 ~> MzcqweqeqDc=
<~  334 UGqwqwqwqQ6
 ~> Y2FsYWJqwqweqwenMQ==
<~  235 2.7.0 Authentication successful
 ~> MAIL FROM:<angeloraffaele.granito@home.local>
<~  250 2.1.0 Sender OK
 ~> RCPT TO:<raffaele.granito@home.local>
<~  250 2.1.5 Recipient OK
 ~> DATA
<~  354 Start mail input; end with <CRLF>.<CRLF>
 ~> Date: Wed, 22 Apr 2020 21:46:10 +0200
 ~> To: raffaele.granito@home.local
~> From: paolo.granito@home.local
~> From: emilia.granito@home.local
~> Subject: test
 ~> Message-Id: <20200422214610.002308@muletto.home.local>
 ~> X-Mailer: swaks v20170101.0 jetmore.org/john/code/swaks/
 ~>
 ~> messaggio di prova
 ~>
 ~>
 ~> .
```

```
<~  250 2.6.0 <20200422214610.002308@muletto.home.local> [InternalId=20860656158486,
Hostname=TELSMTP02RM030.home.local] Queued mail for delivery
 ~> QUIT
<~  221 2.0.0 Service closing transmission channel
=== Connection closed with remote host.
send secure mail (SUCCESS)

-sh-3.2$
```

Impacchettamento e distribuzione di SSM

Per completare il pacchetto ho creato il README che contiene l'essenza di questo documento, ed il LICENSE che contiene le licenze con cui è rilasciabile swaks, NET:SSLeay ed OPENSSL, ossia GNU GPL v2 e licenza MIT

E' bene infine rinominare la home da /tmp/ssm a /tmp/ssm+SSLeay.IA64-LIBC25.GNULinux.build

```
/
|
`--- /tmp/ssm+SSLeay.IA64-LIBC25.GNULinux.build <DIR>
        \
        `--- openssl-1.0.2o.linux-x8664-LIBC25.build    <DIR>
        `--- Net-SSLeay-1.88.IA64-LIBC25-GNULinux.build <DIR>
        `--- package                                    <DIR>   *EMPTY*
        `--- ssm.sh
        `--- ssm.conf
        `--- swaks
        `--- README
        `--- LICENSE
```

Comprimo tutto, ottentendo il package che posso distribuire sulle altre macchine RH5-IA64

```
-sh-3.2$ cd /tmp
-sh-3.2$ tar -czf ssm+SSLeay.IA64-LIBC25.GNULinux.build.tar.gz
ssm+SSLeay.IA64-LIBC25.GNULinux.build
```

Prima di ridistribuire eseguire un ultimo TEST. L'invio mail installando SSM su un'altra macchina RH5-IA64. Basterà copiare il pacchetto ed aprirlo dove si vuole. Dopodichè verificare la configurazione (ssh.sh ed ssm.conf) ed eseguirlo.

La procedura vista fino a quetso momento può essere applicata ad altri ambienti operativi, dovrebbe funzionare senza alcuna modifica, o con piccole variazioni.

Se avete dubbi, domande o idee sull'argomento inviatemi una mail. Grazie.

APPENDICE

[HACKTIME] Analisi su NET::SSLeay già presente sulla macchina.

Mi sono addentrato in Perl, in un mondo che non conoscevo e che inizia ad affascinarmi, ma al momento mi fermo qui. Ho spaziato senza una meta bel precisa perché non sapevo nulla di perl e delle sue estensioni. Il CPAM. La PERL5LIB. Nulla 😋

Diario di una notte finita con una mail finalmente inviata da una macchina il cui sistema operativo non supportava TLS.

Vediamo la configurazione di PERL

```
bash-3.2$ perl -V
Summary of my perl5 (revision 5 version 8 subversion 8) configuration:
  Platform:
    osname=linux, osvers=2.6.18-398.el5, archname=x86_64-linux-thread-multi
    uname='linux x86-029.build.eng.bos.redhat.com 2.6.18-398.el5 #1 smp tue aug 12
06:26:17 edt 2014 x86_64 x86_64 x86_64 gnulinux '
    config_args='-des -Doptimize=-O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -
fexceptions -fstack-protector --param=ssp-buffer-size=4 -m64 -mtune=generic -
Dversion=5.8.8 -Dmyhostname=localhost -Dperladmin=root@localhost -Dcc=gcc -Dcf_by=Red
Hat, Inc. -Dinstallprefix=/usr -Dprefix=/usr -Dlibpth=/usr/local/lib64 /lib64
/usr/lib64 -Dprivlib=/usr/lib/perl5/5.8.8 -Dsitelib=/usr/lib/perl5/site_perl/5.8.8 -
Dvendorlib=/usr/lib/perl5/vendor_perl/5.8.8 -Darchlib=/usr/lib64/perl5/5.8.8/x86_64-
linux-thread-multi -Dsitearch=/usr/lib64/perl5/site_perl/5.8.8/x86_64-linux-thread-
multi -Dvendorarch=/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi -
Darchname=x86_64-linux-thread-multi -Dvendorprefix=/usr -Dsiteprefix=/usr -Duseshrplib
-Dusethreads -Duseithreads -Duselargefiles -Dd_dosuid -Dd_semctl_semun -Di_db -Ui_ndbm
-Di_gdbm -Di_shadow -Di_syslog -Dman3ext=3pm -Duseperlio -Dinstallusrbinperl=n -
Ubincompat5005 -Uversiononly -Dpager=/usr/bin/less -isr -Dd_gethostent_r_proto -
Ud_endhostent_r_proto -Ud_sethostent_r_proto -Ud_endprotoent_r_proto -
Ud_setprotoent_r_proto -Ud_endservent_r_proto -Ud_setservent_r_proto -
Dinc_version_list=5.8.7 5.8.6 5.8.5 -Dscriptdir=/usr/bin'
    hint=recommended, useposix=true, d_sigaction=define
    usethreads=define use5005threads=undef useithreads=define usemultiplicity=define
    useperlio=define d_sfio=undef uselargefiles=define usesocks=undef
    use64bitint=define use64bitall=define uselongdouble=undef
    usemymalloc=n, bincompat5005=undef
  Compiler:
    cc='gcc', ccflags ='-D_REENTRANT -D_GNU_SOURCE -fno-strict-aliasing -pipe -
Wdeclaration-after-statement -I/usr/local/include -D_LARGEFILE_SOURCE -
D_FILE_OFFSET_BITS=64 -I/usr/include/gdbm',
    optimize='-O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-
protector --param=ssp-buffer-size=4 -m64 -mtune=generic',
    cppflags='-D_REENTRANT -D_GNU_SOURCE -fno-strict-aliasing -pipe -Wdeclaration-
after-statement -I/usr/local/include -I/usr/include/gdbm'
    ccversion='', gccversion='4.1.2 20080704 (Red Hat 4.1.2-55)', gccosandvers=''
    intsize=4, longsize=8, ptrsize=8, doublesize=8, byteorder=12345678
    d_longlong=define, longlongsize=8, d_longdbl=define, longdblsize=16
    ivtype='long', ivsize=8, nvtype='double', nvsize=8, Off_t='off_t', lseeksize=8
    alignbytes=8, prototype=define
  Linker and Libraries:
    ld='gcc', ldflags =''
    libpth=/usr/local/lib64 /lib64 /usr/lib64
    libs=-lresolv -lnsl -lgdbm -ldb -ldl -lm -lcrypt -lutil -lpthread -lc
    perllibs=-lresolv -lnsl -ldl -lm -lcrypt -lutil -lpthread -lc
```

```
    libc=, so=so, useshrplib=true, libperl=libperl.so
    gnulibc_version='2.5'
  Dynamic Linking:
    dlsrc=dl_dlopen.xs, dlext=so, d_dlsymun=undef, ccdlflags='-Wl,-E -Wl,-
rpath,/usr/lib64/perl5/5.8.8/x86_64-linux-thread-multi/CORE'
    cccdlflags='-fPIC', lddlflags='-shared -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2
-fexceptions -fstack-protector --param=ssp-buffer-size=4 -m64 -mtune=generic'


Characteristics of this binary (from libperl):
  Compile-time options: MULTIPLICITY PERL_IMPLICIT_CONTEXT
                        PERL_MALLOC_WRAP USE_64_BIT_ALL USE_64_BIT_INT
                        USE_ITHREADS USE_LARGE_FILES USE_PERLIO
                        USE_REENTRANT_API
  Built under linux
  Compiled at Oct 20 2014 09:37:42
  @INC:
    /usr/lib64/perl5/site_perl/5.8.8/x86_64-linux-thread-multi
    /usr/lib/perl5/site_perl/5.8.8
    /usr/lib/perl5/site_perl
    /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi
    /usr/lib/perl5/vendor_perl/5.8.8
    /usr/lib/perl5/vendor_perl
    /usr/lib64/perl5/5.8.8/x86_64-linux-thread-multi
    /usr/lib/perl5/5.8.8
    .
```

La variabile di ambiente @INC in PERL è l'equivalente della **LD_LIBRARY_PATH** per il sistema operativo. Andando ad analizzare il contenuto di queste directory è possibile identificare file e posizione su filesystem dell'attuale SSLeay (son le parti in verde, sintetizzate di seguito).
Ho letto che Net::SSLeay è un estensione, e che il nome identifica una sorta di gerarchia su filesystem Net/SSLeay e che una di quelle directory incluse @INC doveva essere la homedir delle extension. Ho trovato delle Net/SSLeay in alcune sottodirectory in una delle PATH sopra elencate

```
-sh-3.2$ ls /usr/lib64/perl5/site_perl/5.8.8/x86_64-linux-thread-multi
auto
-sh-3.2$ ls /usr/lib64/perl5/site_perl/5.8.8/x86_64-linux-thread-multi/auto/

-sh-3.2$ ls /usr/lib/perl5/site_perl/5.8.8

-sh-3.2$ ls /usr/lib/perl5/site_perl
5.8.8
-sh-3.2$ ls /usr/lib/perl5/site_perl/5.8.8/

-sh-3.2$ ls /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi
auto  Compress  HTML  IO  Net  String

-sh-3.2$ ls /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/auto/
Compress  HTML  IO  Net  String
-sh-3.2$ ls /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/auto/Net/
SSLeay
-sh-3.2$ ls /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-
multi/auto/Net/SSLeay/
autosplit.ix              get_http4.al                  head_https3.al
new_x_ctx.al              post_httpx.al                 set_cert_and_key.al
tcp_read_all.al           debug_read.al                 get_http.al
head_https4.al            open_proxy_tcp_connection.al  put_http3.al
set_proxy.al              tcp_read_CRLF.al              do_https2.al
get_https3.al             head_https.al                 open_tcp_connection.al
put_http4.al              set_server_cert_and_key.al    tcp_read_until.al
```

```
do_https3.al                get_https4.al                head_httpx3.al
post_http3.al               put_http.al                  sslcat.al
tcp_write_all.al            do_https4.al                 get_https.al
head_httpx4.al              post_http4.al                put_https3.al
SSLeay.so                   tcp_write_CRLF.al            do_https.al
get_httpx3.al               head_httpx.al                post_http.al
put_https4.al               ssl_read_all.al              tcpxcat.al
do_httpx2.al                get_httpx4.al                http_cat.al
post_https3.al              put_https.al                 ssl_read_CRLF.al
want_nothing.al             do_httpx3.al                 get_httpx.al
https_cat.al                post_https4.al               put_httpx3.al
ssl_read_until.al           want_read.al                 do_httpx4.al
head_http3.al               httpx_cat.al                 post_https.al
put_httpx4.al               ssl_write_all.al             want_write.al
dump_peer_certificate.al    head_http4.al                make_form.al
post_httpx3.al              put_httpx.al                 ssl_write_CRLF.al
want_X509_lookup.al         get_http3.al                 head_http.al
make_headers.al             post_httpx4.al               randomize.al
tcpcat.al

-sh-3.2$ ls /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Net/
SSLeay  SSLeay.pm

-sh-3.2$ ls /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Net/SSLeay
Handle.pm


-sh-3.2$ ls /usr/lib/perl5/vendor_perl/
5.8.8
-sh-3.2$ ls /usr/lib/perl5/vendor_perl/5.8.8/
auto  Bundle  File  Foomatic  HTML  HTTP  LWP  lwpcook.pod  LWP.pm  lwptut.pod  Net
URI  URI.pm  WWW
-sh-3.2$ ls /usr/lib/perl5/vendor_perl/5.8.8/auto/
Net
-sh-3.2$ ls /usr/lib/perl5/vendor_perl/5.8.8/auto/Net/
IPv4Addr
-sh-3.2$ ls /usr/lib/perl5/vendor_perl/5.8.8/Net/
HTTP  HTTP.pm  HTTPS.pm  IPv4Addr.pm


-sh-3.2$ ls /usr/lib64/perl5/5.8.8/x86_64-linux-thread-multi
asm             bits              Cwd.pm          Encode.pm
Filter          IPC               Opcode.pm       Safe.pm
sys             Time              asm-generic     B.pm
Data            encoding.pm       GDBM_File.pm    lib.pm
O.pm            SDBM_File.pm      Sys             time.ph
asm-i386        ByteLoader.pm     DB_File.pm      endian.ph
gnu             limits.ph         ops.pm          signal.ph
syscall.ph      Unicode           asm-x86_64      Config_heavy.pl
Devel           Errno.pm          _h2ph_pre.ph    linux
PerlIO          Socket.pm         syslimits.ph    wait.ph
attrs.pm        Config.pm         Digest          Fcntl.pm
I18N            machine           POSIX.pm        stdarg.ph
syslog.ph       xlocale.ph        auto            Config.pod
DynaLoader.pm   features.ph       IO              MIME
POSIX.pod       stddef.ph         threads         XS
B               CORE              Encode          File
IO.pm           NDBM_File.pm      re.pm           Storable.pm
threads.pm      XSLoader.pm

-sh-3.2$ ls /usr/lib/perl5/5.8.8
abbrev.pl          bigint.pm        complete.pl      dotsh.pl
```

```
FileCache.pm        I18N              Net               Search
Term                utf8_heavy.pl     AnyDBM_File.pm    bignum.pm
constant.pm         Dumpvalue.pm      FileHandle.pm     if.pm
newgetopt.pl        SelectSaver.pm    termcap.pl        utf8.pm
assert.pl           bigrat.pl         CPAN              dumpvar.pl
filetest.pm         importenv.pl      NEXT.pm           SelfLoader.pm
Test                validate.pl       Attribute         bigrat.pm
CPAN.pm             Encode            Filter            integer.pm
open2.pl            Shell.pm          Test.pm           vars.pm
attributes.pm       blib.pm           ctime.pl          English.pm
FindBin.pm          IO                open3.pl          shellwords.pl
Text                vmsish.pm         auto              bytes_heavy.pl
DBM_Filter          Env.pm            finddepth.pl      IPC
open.pm             sigtrap.pm        Thread            warnings
AutoLoader.pm       bytes.pm          DBM_Filter.pm     exceptions.pl
find.pl             less.pm           overload.pm       sort.pm
Thread.pm           warnings.pm       AutoSplit.pm      cacheout.pl
dbm_filter_util.pl  Exporter          flush.pl          List
perl5db.pl          stat.pl           Tie               autouse.pm
Carp                DB.pm             Exporter.pm       getcwd.pl
Locale              PerlIO            strict.pm         Time
B                   Carp.pm           Devel             ExtUtils
Getopt              locale.pm         PerlIO.pm         subs.pm
timelocal.pl        base.pm           CGI               diagnostics.pm
fastcwd.pl          getopt.pl         look.pl           pod
Switch.pm           Unicode           Benchmark.pm      CGI.pm
Digest              Fatal.pm          getopts.pl        Math
Pod                 Symbol.pm         unicore           bigfloat.pl
charnames.pm        Digest.pm         fields.pm         Hash
Memoize             pwd.pl            syslog.pl         UNIVERSAL.pm
bigint.pl           Class             DirHandle.pm      File
hostname.pl         Memoize.pm        Scalar            tainted.pl
User
```

In sistesi questi i File e le Posizioni su filesystem dell'attuale Net::SSLeay

```
-sh-3.2$ ls /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/

/Net
  \
  `--- SSLeay <DIR>
  |       \
  |       `--- Handle.pm
  |
  `--- SSLeay.pm


/auto <DIR>
  \
  `--- Net <DIR>
  |     \
  |     `--- SSLeay <DIR>
  |            \
  |      autosplit.ix             get_http4.al                head_https3.al
  |      new_x_ctx.al             post_httpx.al               set_cert_and_key.al
  |      tcp_read_all.al          debug_read.al               get_http.al
  |      head_https4.al           open_proxy_tcp_connection.al put_http3.al
  |      set_proxy.al             tcp_read_CRLF.al            do_https2.al
```

```
   |      get_https3.al            head_https.al               open_tcp_connection.al
   |      put_http4.al             set_server_cert_and_key.al  tcp_read_until.al
   |      do_https3.al             get_https4.al               head_httpx3.al
   |      post_http3.al            put_http.al                 sslcat.al
   |      tcp_write_all.al         do_https4.al                get_https.al
   |      head_httpx4.al           post_http4.al               put_https3.al
   |      SSLeay.so                tcp_write_CRLF.al           do_https.al
   |      get_httpx3.al            head_httpx.al               post_http.al
   |      put_https4.al            ssl_read_all.al             tcpxcat.al
   |      do_httpx2.al             get_httpx4.al               http_cat.al
   |      post_https3.al           put_https.al                ssl_read_CRLF.al
   |      want_nothing.al          do_httpx3.al                get_httpx.al
   |      https_cat.al             post_https4.al              put_httpx3.al
   |      ssl_read_until.al        want_read.al                do_httpx4.al
   |      head_http3.al            httpx_cat.al                post_https.al
   |      put_httpx4.al            ssl_write_all.al            want_write.al
   |      dump_peer_certificate.al head_http4.al               make_form.al
   |      post_httpx3.al           put_httpx.al                ssl_write_CRLF.al
   |      want_X509_lookup.al      get_http3.al                head_http.al
   |      make_headers.al          post_httpx4.al              randomize.al
   |      tcpcat.al
```

Andiamo ad analizzare le varie tipologie di file contenute in queste posizione. Il primo sembra il file di presentazione dell'estensione, con versione impostata a 0.61

/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Net/SSLeay/Handle.pm

```
$ vi /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Net/SSLeay/Handle.pm

# $Id: Handle.pm,v 1.9 2003/08/03 20:52:40 sampo Exp $

package Net::SSLeay::Handle;

require 5.005_03;
use strict;

use Socket;
use Net::SSLeay;

require Exporter;

use vars qw(@ISA @EXPORT_OK $VERSION);
@ISA = qw(Exporter);
@EXPORT_OK = qw(shutdown);
$VERSION = '0.61';

#=== Class Variables ============================================================
#
# %Filenum_Object holds the attributes (see bottom of TIEHANDLE) of tied
# handles keyed by fileno.  This was the only way I could figure out how
# to "attach" attributes to a returned glob reference.
#
#================================================================================

my $Initialized;       #-- only _initialize() once
my %Filenum_Object;    #-- hash of hashes, keyed by fileno()
my $Debug = 0;         #-- pretty hokey
my %Glob_Ref;          #-- used to make unique \*S names for versions < 5.6
```

```
#== Tie Handle Methods ============================================================
#
# see perldoc perltie for details.
#
#===================================================================================

sub TIEHANDLE {
    my ($class, $socket, $port) = @_;
    $Debug > 10 and print "TIEHANDLE(@{[join ', ', @_]})\n";

    ref $socket eq "GLOB" or $socket = $class->make_socket($socket, $port);

    $class->_initialize();

    my $ctx = Net::SSLeay::CTX_new() or die_now("Failed to create SSL_CTX $!");
    my $ssl = Net::SSLeay::new($ctx) or die_now("Failed to create SSL $!");

    my $fileno = fileno($socket);

  Net::SSLeay::set_fd($ssl, $fileno);   # Must use fileno

    my $resp = Net::SSLeay::connect($ssl);

    $Debug and print "Cipher '" . Net::SSLeay::get_cipher($ssl) . "'\n";

    $Filenum_Object{$fileno} = {
        ssl    => $ssl,
        ctx    => $ctx,
        socket => $socket,
        fileno => $fileno,
    };

    return bless $socket, $class;
}

sub PRINT {
    my $socket = shift;

    my $ssl  = _get_ssl($socket);
    my $resp = 0;
    for my $msg (@_) {
        defined $msg or last;
        $resp = Net::SSLeay::write($ssl, $msg) or last;
    }
    return $resp;
}

sub READLINE {
    my $socket = shift;
    my $ssl  = _get_ssl($socket);
    my $line = Net::SSLeay::ssl_read_until($ssl);
    return $line ? $line : undef;
}

sub READ {
    my ($socket, $buf, $len, $offset) = \ (@_);
    my $ssl = _get_ssl($$socket);
    defined($$offset) or
      return length($$buf = Net::SSLeay::ssl_read_all($ssl, $$len));

    defined(my $read = Net::SSLeay::ssl_read_all($ssl, $$len))
```

```
      or return undef;

    my $buf_len = length($$buf);
    $$offset > $buf_len and $$buf .= chr(0) x ($$offset - $buf_len);
    substr($$buf, $$offset) = $read;
    return length($read);
}

sub WRITE {
    my $socket = shift;
    my ($buf, $len, $offset) = @_;
    $offset = 0 unless defined $offset;

    # Return number of characters written.
    my $ssl  = $socket->_get_ssl();
    return $len if Net::SSLeay::write($ssl, substr($buf, $offset, $len));
    return undef;
}

sub CLOSE {
    my $socket = shift;
    my $fileno = fileno($socket);
    $Debug > 10 and print "close($fileno)\n";
    my $self = $socket->_get_self();
    delete $Filenum_Object{$fileno};
  Net::SSLeay::free ($self->{ssl});
  Net::SSLeay::CTX_free ($self->{ctx});
    close $socket;
}

sub FILENO  { fileno($_[0]) }


#== Exportable Functions  ============================================

# TIEHANDLE, PRINT, READLINE, CLOSE FILENO, READ, WRITE

#--- shutdown(\*SOCKET, $mode) ------------------------------------------
# Calls to the main shutdown() don't work with tied sockets created with this
# module.  This shutdown should be able to distinquish between tied and untied
# sockets and do the right thing.
#------------------------------------------------------------------------

sub shutdown {
    my ($socket, @params) = @_;

    my $obj = _get_self($socket);
    $obj and $socket = $obj->{socket};
    return shutdown($socket, @params);
}

#========================================================================

sub debug {
    my ($class, $debug) = @_;
    my $old_debug = $Debug;
    @_ >1 and $Debug = $debug || 0;
    return $old_debug;
}

#=== Internal Methods ================================================
```

```
sub make_socket {
    my ($class, $host, $port) = @_;
    $Debug > 10 and print "_make_socket(@{[join ', ', @_]})\n";
    $host ||= 'localhost';
    $port ||= 443;

    my $phost = $Net::SSLeay::proxyhost;
    my $pport = $Net::SSLeay::proxyhost ? $Net::SSLeay::proxyport : $port;

    my $dest_ip     = gethostbyname( $phost || $host);
    my $host_params = sockaddr_in($pport, $dest_ip);
    my $socket = $^V ? $class->_glob_ref("$host:$port") : undef;

    socket($socket, &PF_INET(), &SOCK_STREAM(), 0) or die "socket: $!";
    connect($socket, $host_params)                 or die "connect: $!";

    my $old_select = select($socket); $| = 1; select($old_select);
    $phost and do {
        my $auth = $Net::SSLeay::proxyauth;
        my $CRLF = $Net::SSLeay::CRLF;
        print $socket "CONNECT $host:$port HTTP/1.0$auth$CRLF$CRLF";
        my $line = <$socket>;
    };
    return $socket;
}

#--- _glob_ref($strings) -----------------------------------------------------
#
# Create a unique namespace name and return a glob ref to it.  Would be great
# to use the fileno but need this before we get back the fileno.
# NEED TO LOCK THIS ROUTINE IF USING THREADS. (but it is only used for
# versions < 5.6 :)
#-------------------------------------------------------------------------------

sub _glob_ref {
    my $class = shift;
    my $preamb = join("", @_) || "_glob_ref";
    my $num = ++$Glob_Ref{$preamb};
    my $name = "$preamb:$num";
    no strict 'refs';
    my $glob_ref = \*$name;
    use strict 'refs';

    $Debug and do {
        print "GLOB_REF $preamb\n";
        while (my ($k, $v) = each %Glob_Ref) {print "$k = $v\n"}
        print "\n";
    };

    return $glob_ref;
}

sub _initialize {
    $Initialized++ and return;
  Net::SSLeay::load_error_strings();
  Net::SSLeay::SSLeay_add_ssl_algorithms();
  Net::SSLeay::randomize();
}

sub __dummy {
    my $host = $Net::SSLeay::proxyhost;
    my $port = $Net::SSLeay::proxyport;
```

```
    my $auth = $Net::SSLeay::proxyauth;
}

#--- _get_self($socket) ------------------------------------------------------
# Returns a hash containing attributes for $socket (= \*SOMETHING) based
# on fileno($socket).  Will return undef if $socket was not created here.
#-------------------------------------------------------------------------------

sub _get_self {
    return $Filenum_Object{fileno(shift)};
}

#--- _get_ssl($socket) -------------------------------------------------------
# Returns a the "ssl" attribute for $socket (= \*SOMETHING) based
# on fileno($socket).  Will cause a warning and return undef if $socket was not
# created here.
#-------------------------------------------------------------------------------

sub _get_ssl {
    my $socket = shift;
    return $Filenum_Object{fileno($socket)}->{ssl};
}

1;
__END__
```

Quello che segue dovrebbe essere il codice (perl) della SSLeay, con tanto di ChangeLog in test e rimappaggio delle funzioni implementate nelle OpenSSL. Anche qui trovo una variabile VERSION impostata ad 1.30. Mi sembra questa la versione di SSLeay.

/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Net/SSLeay.pm

```
$ vi /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Net/SSLeay.pm

# Net::SSLeay.pm - Perl module for using Eric Young's implementation of SSL-multi
#
# Copyright (c) 1996-2003 Sampo Kellomaki <sampo@iki.fi>, All Rights Reserved.
# Copyright (C) 2005 Florian Ragwitz <rafl@debian.org>, All Rights Reserved.
# Copyright (C) 2005 Mike McCauley <mikem@open.com.au>, All Rights Reserved.
#
# $Id: /local/net-ssleay/trunk/SSLeay.pm 23905 2005-12-21T03:27:18.584986Z rafl  $
#
# Version 1.04, 31.3.1999
# 30.7.1999, Tracking OpenSSL-0.9.3a changes, --Sampo
# 31.7.1999, version 1.05 --Sampo
# 7.4.2001,  fixed input error upon 0, OpenSSL-0.9.6a, version 1.06 --Sampo
# 18.4.2001, added TLSv1 support by Stephen C. Koehler
#            <koehler@securecomputing.com>, version 1.07, --Sampo
# 25.4.2001, 64 bit fixes by Marko Asplund <aspa@kronodoc.fi> --Sampo
# 17.4.2001, more error codes from aspa --Sampo
# 25.9.2001, added heaps and piles of newer OpenSSL auxiliary functions --Sampo
# 6.11.2001, got rid of $p_errs madness --Sampo
# 9.11.2001, added EGD (entropy gathering daemon) reference info --Sampo
# 7.12.2001, Added proxy support by Bruno De Wolf <bruno.dewolf@@pandora._be>
# 6.1.2002,  cosmetic fix to socket options from Kwindla Hultman Kramer
<kwindla@@allafrica_.com>
# 25.3.2002, added post_https_cert and friends per patch from
#            mock@@obscurity.ogr, --Sampo
# 3.4.2002,  added `use bytes' from Marcus Taylor <marcus@@semantico_.com>
#            This avoids unicode/utf8 (as may appear in some XML docs)
```

```
#            from fooling the length comuptations. Dropped support for
#            perl5.005_03 because I do not have opportunity to test it. --Sampo
# 5.4.2002,  improved Unicode gotcha eliminator to support old perls --Sampo
# 8.4.2002,  added a small line end fix from Petr Dousa (pdousa@@kerio_.com)
# 17.5.2002, Added BIO_s_mem, BIO_new, BIO_free, BIO_write, BIO_read
#            BIO_eof, BIO_pending, BIO_wpending, RSA_generate_key, RSA_free
#            --mikem@open._com.au
# 10.8.2002, Added SSL_peek patch to ssl_read_until from
#            Peter Behroozi <peter@@fhpwireless_.com> --Sampo
# 21.8.2002, Added SESSION_get_master_key, SSL_get_client_random,
SSL_get_server_random
#            --mikem@open.com_.au
# 2.9.2002,  Added SSL_CTX_get_cert_store, X509_STORE_add_cert, X509_STORE_add_crl
#            X509_STORE_set_flags, X509_load_cert_file, X509_load_crl_file
#            X509_load_cert_crl_file, PEM_read_bio_X509_CRL,
#            constants for X509_V_FLAG_* in order to support certificate revocation
lists.
#            --mikem@open.com_.au
# 6.9.2002,  fixed X509_STORE_set_flags to X509_STORE_CTX_set_flags, --Sampo
# 19.9.2002, applied patch from Tim Engler <tim@burntcouch_.com>
# 18.2.2003, applied patch from Toni Andjelkovic <toni@soth._at>
# 13.6.2003, partially applied leak patch by Marian Jancar <mjancar@suse._cz>
# 25.6.2003, write_partial() return value patch from
#            Kim Minh Kaplan <kmkaplan@selfoffice._com>
# 17.8.2003, added http support :-) --Sampo
# 17.8.2003, started 1.25 dev --Sampo
# 30.11.2005, Applied a patch by Peter Behroozi that adds get1_session() for session
caching --Florian
# 30.11.2005, Applied a patch by ex8k-hbn@asahi-net.or.jp that limits the chunk size
for tcp_read_all --Florian
# 30.11.2005, Applied a patch by ivan-cpan-rt@420.am that avoids adding a Host header
if an own is specified in do_httpx3
# 13.12.2005, Added comments re thread safety and resetting of default_passwd_callback
after use
#             --mikem@open.com.au
#
# The distribution and use of this module are subject to the conditions
# listed in LICENSE file at the root of OpenSSL-0.9.7b
# distribution (i.e. free, but mandatory attribution and NO WARRANTY).

package Net::SSLeay;

use strict;
use Carp;
use vars qw($VERSION @ISA @EXPORT @EXPORT_OK $AUTOLOAD $CRLF);
use Socket;
use Errno;

require Exporter;
require DynaLoader;
use AutoLoader;

# 0=no warns, 1=only errors, 2=ciphers, 3=progress, 4=dump data
$Net::SSLeay::trace = 0;  # Do not change here, use
                          # $Net::SSLeay::trace = [1-4]  in caller

# 2 = insist on v2 SSL protocol
# 3 = insist on v3 SSL
# 10 = insist on TLSv1
# 0 or undef = guess (v23)
#
$Net::SSLeay::ssl_version = 0;  # don't change here, use
```

```
                                # Net::SSLeay::version=[2,3,0] in caller

#define to enable the "cat /proc/$$/stat" stuff
$Net::SSLeay::linux_debug = 0;

# Number of seconds to sleep after sending message and before half
# closing connection. Useful with antiquated broken servers.
$Net::SSLeay::slowly = 0;

# RANDOM NUMBER INITIALIZATION
#
# Edit to your taste. Using /dev/random would be more secure, but may
# block if randomness is not available, thus the default is
# /dev/urandom. $how_random determines how many bits of randomness to take
# from the device. You should take enough (read SSLeay/doc/rand), but
# beware that randomness is limited resource so you should not waste
# it either or you may end up with randomness depletion (situation where
# /dev/random would block and /dev/urandom starts to return predictable
# numbers).
#
# N.B. /dev/urandom does not exit on all systems, such as Solaris 2.6. In that
#      case you should get a third party package that emulates /dev/urandom
#      (e.g. via named pipe) or supply a random number file. Some such
#      packages are documented in Caveat section of the POD documentation.

$Net::SSLeay::random_device = '/dev/urandom';
$Net::SSLeay::how_random = 512;

$VERSION = '1.30';
@ISA = qw(Exporter DynaLoader);
@EXPORT_OK = qw(
        AT_MD5_WITH_RSA_ENCRYPTION
        CB_ACCEPT_EXIT
        CB_ACCEPT_LOOP
        CB_CONNECT_EXIT
        CB_CONNECT_LOOP
        CK_DES_192_EDE3_CBC_WITH_MD5
        CK_DES_192_EDE3_CBC_WITH_SHA
        CK_DES_64_CBC_WITH_MD5
        CK_DES_64_CBC_WITH_SHA
        CK_DES_64_CFB64_WITH_MD5_1
        CK_IDEA_128_CBC_WITH_MD5
        CK_NULL
        CK_NULL_WITH_MD5
        CK_RC2_128_CBC_EXPORT40_WITH_MD5
        CK_RC2_128_CBC_WITH_MD5
        CK_RC4_128_EXPORT40_WITH_MD5
        CK_RC4_128_WITH_MD5
        CLIENT_VERSION
        ERROR_NONE
        ERROR_SSL
        ERROR_SYSCALL
        ERROR_WANT_CONNECT
        ERROR_WANT_READ
        ERROR_WANT_WRITE
        ERROR_WANT_X509_LOOKUP
        ERROR_ZERO_RETURN
        CT_X509_CERTIFICATE
        FILETYPE_ASN1
        FILETYPE_PEM
        F_CLIENT_CERTIFICATE
        F_CLIENT_HELLO
```

```
        F_CLIENT_MASTER_KEY
        F_D2I_SSL_SESSION
        F_GET_CLIENT_FINISHED
        F_GET_CLIENT_HELLO
        F_GET_CLIENT_MASTER_KEY
        F_GET_SERVER_FINISHED
        F_GET_SERVER_HELLO
        F_GET_SERVER_VERIFY
        F_I2D_SSL_SESSION
        F_READ_N
        F_REQUEST_CERTIFICATE
        F_SERVER_HELLO
        F_SSL_ACCEPT
        F_SSL_CERT_NEW
        F_SSL_CONNECT
        F_SSL_ENC_DES_CBC_INIT
        F_SSL_ENC_DES_CFB_INIT
        F_SSL_ENC_DES_EDE3_CBC_INIT
        F_SSL_ENC_IDEA_CBC_INIT
        F_SSL_ENC_NULL_INIT
        F_SSL_ENC_RC2_CBC_INIT
        F_SSL_ENC_RC4_INIT
        F_SSL_GET_NEW_SESSION
        F_SSL_MAKE_CIPHER_LIST
        F_SSL_NEW
        F_SSL_READ
        F_SSL_RSA_PRIVATE_DECRYPT
        F_SSL_RSA_PUBLIC_ENCRYPT
        F_SSL_SESSION_NEW
        F_SSL_SESSION_PRINT_FP
        F_SSL_SET_CERTIFICATE
        F_SSL_SET_FD
        F_SSL_SET_RFD
        F_SSL_SET_WFD
        F_SSL_STARTUP
        F_SSL_USE_CERTIFICATE
        F_SSL_USE_CERTIFICATE_ASN1
        F_SSL_USE_CERTIFICATE_FILE
        F_SSL_USE_PRIVATEKEY
        F_SSL_USE_PRIVATEKEY_ASN1
        F_SSL_USE_PRIVATEKEY_FILE
        F_SSL_USE_RSAPRIVATEKEY
        F_SSL_USE_RSAPRIVATEKEY_ASN1
        F_SSL_USE_RSAPRIVATEKEY_FILE
        F_WRITE_PENDING
        MAX_MASTER_KEY_LENGTH_IN_BITS
        MAX_RECORD_LENGTH_2_BYTE_HEADER
        MAX_RECORD_LENGTH_3_BYTE_HEADER
        MAX_SSL_SESSION_ID_LENGTH_IN_BYTES
        MIN_RSA_MODULUS_LENGTH_IN_BYTES
        MT_CLIENT_CERTIFICATE
        MT_CLIENT_FINISHED
        MT_CLIENT_HELLO
        MT_CLIENT_MASTER_KEY
        MT_ERROR
        MT_REQUEST_CERTIFICATE
        MT_SERVER_FINISHED
        MT_SERVER_HELLO
        MT_SERVER_VERIFY
        NOTHING
        OPENSSL_VERSION_NUMBER
        PE_BAD_CERTIFICATE
```

```
        PE_NO_CERTIFICATE
        PE_NO_CIPHER
        PE_UNSUPPORTED_CERTIFICATE_TYPE
        READING
        RWERR_BAD_MAC_DECODE
        RWERR_BAD_WRITE_RETRY
        RWERR_INTERNAL_ERROR
        R_BAD_AUTHENTICATION_TYPE
        R_BAD_CHECKSUM
        R_BAD_MAC_DECODE
        R_BAD_RESPONSE_ARGUMENT
        R_BAD_SSL_FILETYPE
        R_BAD_SSL_SESSION_ID_LENGTH
        R_BAD_STATE
        R_BAD_WRITE_RETRY
        R_CHALLENGE_IS_DIFFERENT
        R_CIPHER_CODE_TOO_LONG
        R_CIPHER_TABLE_SRC_ERROR
        R_CONECTION_ID_IS_DIFFERENT
        R_INVALID_CHALLENGE_LENGTH
        R_NO_CERTIFICATE_SET
        R_NO_CERTIFICATE_SPECIFIED
        R_NO_CIPHER_LIST
        R_NO_CIPHER_MATCH
        R_NO_CIPHER_WE_TRUST
        R_NO_PRIVATEKEY
        R_NO_PUBLICKEY
        R_NO_READ_METHOD_SET
        R_NO_WRITE_METHOD_SET
        R_NULL_SSL_CTX
        R_PEER_DID_NOT_RETURN_A_CERTIFICATE
        R_PEER_ERROR
        R_PEER_ERROR_CERTIFICATE
        R_PEER_ERROR_NO_CIPHER
        R_PEER_ERROR_UNSUPPORTED_CERTIFICATE_TYPE
        R_PERR_ERROR_NO_CERTIFICATE
        R_PUBLIC_KEY_ENCRYPT_ERROR
        R_PUBLIC_KEY_IS_NOT_RSA
        R_PUBLIC_KEY_NO_RSA
        R_READ_WRONG_PACKET_TYPE
        R_REVERSE_KEY_ARG_LENGTH_IS_WRONG
        R_REVERSE_MASTER_KEY_LENGTH_IS_WRONG
        R_REVERSE_SSL_SESSION_ID_LENGTH_IS_WRONG
        R_SHORT_READ
        R_SSL_SESSION_ID_IS_DIFFERENT
        R_UNABLE_TO_EXTRACT_PUBLIC_KEY
        R_UNDEFINED_INIT_STATE
        R_UNKNOWN_REMOTE_ERROR_TYPE
        R_UNKNOWN_STATE
        R_UNSUPORTED_CIPHER
        R_WRONG_PUBLIC_KEY_TYPE
        R_X509_LIB
        SERVER_VERSION
        SESSION
        SESSION_ASN1_VERSION
        ST_ACCEPT
        ST_BEFORE
        ST_CLIENT_START_ENCRYPTION
        ST_CONNECT
        ST_GET_CLIENT_FINISHED_A
        ST_GET_CLIENT_FINISHED_B
        ST_GET_CLIENT_HELLO_A
```

```
        ST_GET_CLIENT_HELLO_B
        ST_GET_CLIENT_MASTER_KEY_A
        ST_GET_CLIENT_MASTER_KEY_B
        ST_GET_SERVER_FINISHED_A
        ST_GET_SERVER_FINISHED_B
        ST_GET_SERVER_HELLO_A
        ST_GET_SERVER_HELLO_B
        ST_GET_SERVER_VERIFY_A
        ST_GET_SERVER_VERIFY_B
        ST_INIT
        ST_OK
        ST_READ_BODY
        ST_READ_HEADER
        ST_SEND_CLIENT_CERTIFICATE_A
        ST_SEND_CLIENT_CERTIFICATE_B
        ST_SEND_CLIENT_CERTIFICATE_C
        ST_SEND_CLIENT_CERTIFICATE_D
        ST_SEND_CLIENT_FINISHED_A
        ST_SEND_CLIENT_FINISHED_B
        ST_SEND_CLIENT_HELLO_A
        ST_SEND_CLIENT_HELLO_B
        ST_SEND_CLIENT_MASTER_KEY_A
        ST_SEND_CLIENT_MASTER_KEY_B
        ST_SEND_REQUEST_CERTIFICATE_A
        ST_SEND_REQUEST_CERTIFICATE_B
        ST_SEND_REQUEST_CERTIFICATE_C
        ST_SEND_REQUEST_CERTIFICATE_D
        ST_SEND_SERVER_FINISHED_A
        ST_SEND_SERVER_FINISHED_B
        ST_SEND_SERVER_HELLO_A
        ST_SEND_SERVER_HELLO_B
        ST_SEND_SERVER_VERIFY_A
        ST_SEND_SERVER_VERIFY_B
        ST_SERVER_START_ENCRYPTION
        ST_X509_GET_CLIENT_CERTIFICATE
        ST_X509_GET_SERVER_CERTIFICATE
        TXT_DES_192_EDE3_CBC_WITH_MD5
        TXT_DES_192_EDE3_CBC_WITH_SHA
        TXT_DES_64_CBC_WITH_MD5
        TXT_DES_64_CBC_WITH_SHA
        TXT_DES_64_CFB64_WITH_MD5_1
        TXT_IDEA_128_CBC_WITH_MD5
        TXT_NULL
        TXT_NULL_WITH_MD5
        TXT_RC2_128_CBC_EXPORT40_WITH_MD5
        TXT_RC2_128_CBC_WITH_MD5
        TXT_RC4_128_EXPORT40_WITH_MD5
        TXT_RC4_128_WITH_MD5
        VERIFY_CLIENT_ONCE
        VERIFY_FAIL_IF_NO_PEER_CERT
        VERIFY_NONE
        VERIFY_PEER
        WRITING
        X509_LOOKUP
        X509_V_FLAG_CB_ISSUER_CHECK
        X509_V_FLAG_USE_CHECK_TIME
        X509_V_FLAG_CRL_CHECK
        X509_V_FLAG_CRL_CHECK_ALL
        X509_V_FLAG_IGNORE_CRITICAL
        CTX_new
        CTX_v2_new
        CTX_v3_new
```

```
        CTX_v23_new
        CTX_free
        new
        free
        accept
        clear
        connect
        set_fd
        set_rfd
        set_wfd
        get_fd
        read
        write
        peek
        use_RSAPrivateKey
        use_RSAPrivateKey_ASN1
        use_RSAPrivateKey_file
        CTX_use_RSAPrivateKey_file
        use_PrivateKey
        use_PrivateKey_ASN1
        use_PrivateKey_file
        use_certificate
        use_certificate_ASN1
        use_certificate_file
        CTX_use_certificate_file
        load_error_strings
        ERR_load_SSL_strings
        ERR_load_RAND_strings
        state_string
        rstate_string
        state_string_long
        rstate_string_long
        get_time
        set_time
        get_timeout
        set_timeout
        copy_session_id
        set_read_ahead
        get_read_ahead
        pending
        get_cipher_list
        set_cipher_list
        get_cipher
        get_shared_ciphers
        get_peer_certificate
        set_verify
        flush_sessions
        set_bio
        get_rbio
        get_wbio
        SESSION_new
        SESSION_print
        SESSION_free
        i2d_SSL_SESSION
        set_session
        add_session
        remove_session
        d2i_SSL_SESSION
        BIO_f_ssl
        BIO_new
        BIO_new_file
        BIO_s_mem
```

```
        BIO_free
        BIO_read
        BIO_write
        BIO_eof
        BIO_pending
        BIO_wpending
        ERR_get_error
        ERR_error_string
        err
        clear_error
        X509_get_issuer_name
        X509_get_subject_name
        X509_NAME_oneline
        X509_NAME_get_text_by_NID
        CTX_get_cert_store
        X509_STORE_add_cert
        X509_STORE_add_crl
        X509_STORE_CTX_set_flags
        X509_load_cert_file
        X509_load_crl_file
        X509_load_cert_crl_file
        PEM_read_bio_X509_CRL
        die_if_ssl_error
        die_now
        print_errs
        set_cert_and_key
        set_server_cert_and_key
        make_form
        make_headers
        do_https
        get_https
        post_https
        get_https4
        post_https4
        sslcat
        ssl_read_CRLF
        ssl_read_all
        ssl_read_until
        ssl_write_CRLF
        ssl_write_all
        get_http
        post_http
        get_httpx
        post_httpx
        get_http4
        post_http4
        get_httpx4
        post_httpx4
        tcpcat
        tcpxcat
        tcp_read_CRLF
        tcp_read_all
        tcp_read_until
        tcp_write_CRLF
        tcp_write_all
        dump_peer_certificate
        RSA_generate_key
        RSA_free
        X509_free
        SESSION_get_master_key
        get_client_random
        get_server_random
```

```
);

sub AUTOLOAD {
    # This AUTOLOAD is used to 'autoload' constants from the constant()
    # XS function.  If a constant is not found then control is passed
    # to the AUTOLOAD in AutoLoader.

    my $constname;
    ($constname = $AUTOLOAD) =~ s/.*::///;
    my $val = constant($constname);
    if ($! != 0) {
        if ($! =~ /((Invalid)|(not valid))/i || $!{EINVAL}) {
            $AutoLoader::AUTOLOAD = $AUTOLOAD;
            goto &AutoLoader::AUTOLOAD;
        }
        else {
          croak "Your vendor has not defined SSLeay macro $constname";
        }
    }
    eval "sub $AUTOLOAD { $val }";
    goto &$AUTOLOAD;
}

bootstrap Net::SSLeay $VERSION;

# Preloaded methods go here.

$CRLF = "\x0d\x0a";  # because \r\n is not fully portable

### Print SSLeay error stack

sub print_errs {
    my ($msg) = @_;
    my ($count, $err, $errs, $e) = (0,0,'');
    while ($err = ERR_get_error()) {
        $count ++;
        $e = "$msg $$: $count - " . ERR_error_string($err) . "\n";
        $errs .= $e;
        warn $e if $Net::SSLeay::trace;
    }
    return $errs;
}

# Death is conditional to SSLeay errors existing, i.e. this function checks
# for errors and only dies in affirmative.
# usage: Net::SSLeay::write($ssl, "foo") or die_if_ssl_error("SSL write ($!)");

sub die_if_ssl_error {
    my ($msg) = @_;
    die "$$: $msg\n" if print_errs($msg);
}

# Unconditional death. Used to print SSLeay errors before dying.
# usage: Net::SSLeay:connect($ssl) or die_now("Failed SSL connect ($!)");

sub die_now {
    my ($msg) = @_;
    print_errs($msg);
    die "$$: $msg\n";
}

# Perl 5.6.* unicode support causes that length() no longer reliably
```

```
# reflects the byte length of a string. This eval is to fix that.
# Thanks to Sean Burke for the snippet.

BEGIN{
eval 'use bytes; sub blength ($) { length $_[0] }';
$@ and eval '    sub blength ($) { length $_[0] }' ;
}

# Autoload methods go after =cut, and are processed by the autosplit program.

1;
__END__
```

Segue un fuile che contiene l'indice dei file con estensione *.al, che potrebbero essere casi di test. Forse.

/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Auto/Net/SSLeay/autosplit.ix

```
$ vi /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-
multi/Auto/Net/SSLeay/autosplit.ix

# Index created by AutoSplit for blib/lib/Net/SSLeay.pm
#    (file acts as timestamp)
package Net::SSLeay;
sub want_nothing  ;
sub want_read  ;
sub want_write  ;
sub want_X509_lookup  ;
sub open_tcp_connection  ;
sub open_proxy_tcp_connection  ;
sub debug_read  ;
sub ssl_read_all  ;
sub tcp_read_all  ;
sub ssl_write_all  ;
sub tcp_write_all  ;
sub ssl_read_until  ($;$$);
sub tcp_read_until  ;
sub ssl_read_CRLF  ($;$);
sub tcp_read_CRLF  ;
sub ssl_write_CRLF  ($$);
sub tcp_write_CRLF  ;
sub dump_peer_certificate  ($);
sub randomize  (;$$);
sub new_x_ctx  ;
sub sslcat  ;
sub tcpcat  ;
sub tcpxcat  ;
sub https_cat  ;
sub http_cat  ;
sub httpx_cat  ;
sub set_cert_and_key  ($$$);
sub set_server_cert_and_key  ($$$);
sub set_proxy  ($$;**);
sub make_form  ;
sub make_headers  ;
sub do_httpx3  ;
sub do_https3  ;
sub do_httpx2  ;
sub do_https2  ;
sub do_httpx4  ;
```

```
sub do_https4  ;
sub get_https   ($$$;***);
sub post_https  ($$$;***);
sub put_https   ($$$;***);
sub head_https  ($$$;***);
sub get_https3   ($$$;***);
sub post_https3  ($$$;***);
sub put_https3   ($$$;***);
sub head_https3  ($$$;***);
sub get_https4   ($$$;***);
sub post_https4  ($$$;***);
sub put_https4   ($$$;***);
sub head_https4  ($$$;***);
sub get_http   ($$$;***);
sub post_http  ($$$;***);
sub put_http   ($$$;***);
sub head_http  ($$$;***);
sub get_http3   ($$$;***);
sub post_http3  ($$$;***);
sub put_http3   ($$$;***);
sub head_http3  ($$$;***);
sub get_http4   ($$$;***);
sub post_http4  ($$$;***);
sub put_http4   ($$$;***);
sub head_http4  ($$$;***);
sub get_httpx   ($$$;***);
sub post_httpx  ($$$;***);
sub put_httpx   ($$$;***);
sub head_httpx  ($$$;***);
sub get_httpx3   ($$$;***);
sub post_httpx3  ($$$;***);
sub put_httpx3   ($$$;***);
sub head_httpx3  ($$$;***);
sub get_httpx4   ($$$;***);
sub post_httpx4  ($$$;***);
sub put_httpx4   ($$$;***);
sub head_httpx4  ($$$;***);
sub do_https  ;
1;
-sh-3.2$
```

Seguono alcuni di questi file con estensione *.al

/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Auto/Net/SSLeay/post_http.al

```
# NOTE: Derived from blib/lib/Net/SSLeay.pm
# Changes made here will be lost when autosplit is run again.ead-multi/auto/Net/
# See AutoSplit.pm.al /perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/auto/Net
package Net::SSLeay;

#line 2298 "blib/lib/Net/SSLeay.pm (autosplit into
blib/lib/auto/Net/SSLeay/post_http.al)"
sub post_http ($$$;***) { do_httpx2(POST => 0, @_) }
# end of Net::SSLeay::post_http
1;
```

/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/Auto/Net/SSLeay/sslcat.al

```
# NOTE: Derived from blib/lib/Net/SSLeay.pm.
```

```
# Changes made here will be lost when autosplit is run again.
# See AutoSplit.pm.
package Net::SSLeay;

#line 1897 "blib/lib/Net/SSLeay.pm (autosplit into
blib/lib/auto/Net/SSLeay/sslcat.al)"
###
### Basic request - response primitive (don't use for https)
###

sub sslcat { # address, port, message, $crt, $key --> reply / (reply,errs,cert)
    my ($dest_serv, $port, $out_message, $crt_path, $key_path) = @_;
    my ($ctx, $ssl, $got, $errs, $written);

    ($got, $errs) = open_proxy_tcp_connection($dest_serv, $port);
    return (wantarray ? (undef, $errs) : undef) unless $got;

    ### Do SSL negotiation stuff

    warn "Creating SSL $ssl_version context...\n" if $trace>2;
    load_error_strings();         # Some bloat, but I'm after ease of use
    SSLeay_add_ssl_algorithms();  # and debuggability.
    randomize();

    $ctx = new_x_ctx();
    goto cleanup2 if $errs = print_errs('CTX_new') or !$ctx;

    CTX_set_options($ctx, &OP_ALL);
    goto cleanup2 if $errs = print_errs('CTX_set_options');

    warn "Cert `$crt_path' given without key" if $crt_path && !$key_path;
    set_cert_and_key($ctx, $crt_path, $key_path) if $crt_path;

    warn "Creating SSL connection (context was '$ctx')...\n" if $trace>2;
    $ssl = new($ctx);
    goto cleanup if $errs = print_errs('SSL_new') or !$ssl;

    warn "Setting fd (ctx $ctx, con $ssl)...\n" if $trace>2;
    set_fd($ssl, fileno(SSLCAT_S));
    goto cleanup if $errs = print_errs('set_fd');

    warn "Entering SSL negotiation phase...\n" if $trace>2;

    if ($trace>2) {
        my $i = 0;
        my $p = '';
        my $cipher_list = 'Cipher list: ';
        $p=Net::SSLeay::get_cipher_list($ssl,$i);
        $cipher_list .= $p if $p;
        do {
            $i++;
            $cipher_list .= ', ' . $p if $p;
            $p=Net::SSLeay::get_cipher_list($ssl,$i);
        } while $p;
        $cipher_list .= '\n';
        warn $cipher_list;
    }

    $got = Net::SSLeay::connect($ssl);
    warn "SSLeay connect returned $got\n" if $trace>2;
    goto cleanup if $errs = print_errs('SSL_connect');
```

```
    my $server_cert = get_peer_certificate($ssl);
    print_errs('get_peer_certificate');
    if ($trace>1) {
        warn "Cipher `" . get_cipher($ssl) . "'\n";
        print_errs('get_ciper');
        warn dump_peer_certificate($ssl);
    }

    ### Connected. Exchange some data (doing repeated tries if necessary).

    warn "sslcat $$: sending " . blength($out_message) . " bytes...\n"
        if $trace==3;
    warn "sslcat $$: sending `$out_message' (" . blength($out_message)
        . " bytes)...\n" if $trace>3;
    ($written, $errs) = ssl_write_all($ssl, $out_message);
    goto cleanup unless $written;

    sleep $slowly if $slowly;  # Closing too soon can abort broken servers
    CORE::shutdown SSLCAT_S, 1;  # Half close --> No more output, send EOF to server

    warn "waiting for reply...\n" if $trace>2;
    ($got, $errs) = ssl_read_all($ssl);
    warn "Got " . blength($got) . " bytes.\n" if $trace==3;
    warn "Got `$got' (" . blength($got) . " bytes)\n" if $trace>3;

cleanup:
    free ($ssl);
    $errs .= print_errs('SSL_free');
cleanup2:
    CTX_free ($ctx);
    $errs .= print_errs('CTX_free');
    close SSLCAT_S;
    return wantarray ? ($got, $errs, $server_cert) : $got;
}

# end of Net::SSLeay::sslcat
1;
```

La SSLeay presenta le seguenti dipendenze. Cerca la libssl.so.6 e libcrypto.so.6 sotto /lib64/

```
-sh-3.2$ cd /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-
multi/auto/Net/SSLeay/

-sh-3.2$ ldd SSLeay.so
        linux-vdso.so.1 =>  (0x00007fffaa7be000)
        libssl.so.6 => /lib64/libssl.so.6 (0x00002b21ccd9b000)
        libcrypto.so.6 => /lib64/libcrypto.so.6 (0x00002b21ccfea000)
        libc.so.6 => /lib64/libc.so.6 (0x00002b21cd33c000)
        libgssapi_krb5.so.2 => /usr/lib64/libgssapi_krb5.so.2 (0x00002b21cd695000)
        libkrb5.so.3 => /usr/lib64/libkrb5.so.3 (0x00002b21cd8c4000)
        libcom_err.so.2 => /lib64/libcom_err.so.2 (0x00002b21cdb59000)
        libk5crypto.so.3 => /usr/lib64/libk5crypto.so.3 (0x00002b21cdd5b000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00002b21cdf81000)
        libz.so.1 => /lib64/libz.so.1 (0x00002b21ce185000)
        /lib64/ld-linux-x86-64.so.2 (0x0000003d98a00000)
        libkrb5support.so.0 => /usr/lib64/libkrb5support.so.0 (0x00002b21ce399000)
        libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00002b21ce5a2000)
        libresolv.so.2 => /lib64/libresolv.so.2 (0x00002b21ce7a4000)
        libselinux.so.1 => /lib64/libselinux.so.1 (0x00002b21ce9ba000)
        libsepol.so.1 => /lib64/libsepol.so.1 (0x00002b21cebd2000)
-sh-3.2$
```

Versione di NET::SSLeay compilata in /tmp/ssm/package/Net-SSLeay-1.88

Dopo aver compilato la mia Net-SSLeay. Ho capito che con il comando “make” viene creata una directory chiamata blib, nella quale è contenuto tutto il build che con il successivo “make install” viene spostato nella home_install.
Ho confrontato la nuova Net-SSLeay con quella preinstallata per capire le differenze e come conferma di aver individuato correttamente il perimetro dell’estensione. Le 2 directoru coincidevano quasi alla perfezione, tranne che per un file *.al in più nella versione appena compilata, e sicuramente più aggiornata, e nella posizione della Net::SSLeai.so che non si trova nel ./blib

BUILD ./blib/lib/auto/Net/SSLeay/	INSTALLED /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/auto/Net/SSLeay/
autosplit.ix	autosplit.ix
debug_read.al	debug_read.al
do_https2.al	do_https2.al
do_https3.al	do_https3.al
do_https4.al	do_https4.al
do_https.al	do_https.al
do_httpx2.al	do_httpx2.al
do_httpx3.al	do_httpx3.al
do_httpx4.al	do_httpx4.al
dump_peer_certificate.al	dump_peer_certificate.al
get_http3.al	get_http3.al
get_http4.al	get_http4.al
get_http.al	get_http.al
get_https3.al	get_https3.al
get_https4.al	get_https4.al
get_https.al	get_https.al
get_httpx3.al	get_httpx3.al
get_httpx4.al	get_httpx4.al
get_httpx.al	get_httpx.al
head_http3.al	head_http3.al
head_http4.al	head_http4.al
head_http.al	head_http.al
head_https3.al	head_https3.al
head_https4.al	head_https4.al
head_https.al	head_https.al
head_httpx3.al	head_httpx3.al
head_httpx4.al	head_httpx4.al
head_httpx.al	head_httpx.al
http_cat.al	http_cat.al
https_cat.al	https_cat.al
httpx_cat.al	httpx_cat.al
initialize.al	
make_form.al	make_form.al
make_headers.al	make_headers.al
new_x_ctx.al	new_x_ctx.al
open_proxy_tcp_connection.al	open_proxy_tcp_connection.al
open_tcp_connection.al	open_tcp_connection.al
post_http3.al	post_http3.al
post_http4.al	post_http4.al
post_http.al	post_http.al
post_https3.al	post_https3.al
post_https4.al	post_https4.al
post_https.al	post_https.al
post_httpx3.al	post_httpx3.al

post_httpx4.al	post_httpx4.al
post_httpx.al	post_httpx.al
put_http3.al	put_http3.al
put_http4.al	put_http4.al
put_http.al	put_http.al
put_https3.al	put_https3.al
put_https4.al	put_https4.al
put_https.al	put_https.al
put_httpx3.al	put_httpx3.al
put_httpx4.al	put_httpx4.al
put_httpx.al	put_httpx.al
randomize.al	randomize.al
set_cert_and_key.al	set_cert_and_key.al
set_proxy.al	set_proxy.al
set_server_cert_and_key.al	set_server_cert_and_key.al
sslcat.al	sslcat.al
	SSLeay.so
ssl_read_all.al	ssl_read_all.al
ssl_read_CRLF.al	ssl_read_CRLF.al
ssl_read_until.al	ssl_read_until.al
ssl_write_all.al	ssl_write_all.al
ssl_write_CRLF.al	ssl_write_CRLF.al
tcpcat.al	tcpcat.al
tcp_read_all.al	tcp_read_all.al
tcp_read_CRLF.al	tcp_read_CRLF.al
tcp_read_until.al	tcp_read_until.al
tcp_write_all.al	tcp_write_all.al
tcp_write_CRLF.al	tcp_write_CRLF.al
tcpxcat.al	tcpxcat.al
want_nothing.al	want_nothing.al
want_read.al	want_read.al
want_write.al	want_write.al
want_X509_lookup.al	want_X509_lookup.al

Unico file .al a trovarsi in più nel build (./blib/lib/auto/Net/SSLeay/initialize.al)

```
-sh-3.2$ cat ./blib/lib/auto/Net/SSLeay/initialize.al

# NOTE: Derived from blib/lib/Net/SSLeay.pm.
# Changes made here will be lost when autosplit is run again.
# See AutoSplit.pm.
package Net::SSLeay;

#line 1032 "blib/lib/Net/SSLeay.pm (autosplit into
blib/lib/auto/Net/SSLeay/initialize.al)"
###
### Standard initialisation. Initialise the ssl library in the usual way
###  at most once. Override this if you need differnet initialisation
###  SSLeay_add_ssl_algorithms is also protected against multiple runs in SSLeay.xs
###  and is also mutex protected in threading perls
###

my $library_initialised;
sub initialize
{
    if (!$library_initialised)
    {
        load_error_strings();         # Some bloat, but I'm after ease of use
        SSLeay_add_ssl_algorithms();  # and debuggability.
        randomize();
```

```
        $library_initialised++;
    }
}

# end of Net::SSLeay::initialize
1;
```

La SSLeay.so presenta le seguenti differenze sulle dipendenze. Il nuovo build presenta molte meno dipendenze. Potrebbe essere che alcune funzionalità non sono state attivate o che (come nel caso delle lib di openssl: libssl e libcrypto) che siano state linkate staticamente, cioè portate all'interno. I test eseguiti positivamente confermano questo. Le lib OPENSSL possono essere eliminate dal pacchetto SSM+NetSSLeay, sono ritondanti e non utilizzate. Provate!

INSTALLED	**-sh-3.2$** cd /usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi/auto/Net/SSLeay/ **-sh-3.2$** ldd SSLeay.so linux-vdso.so.1 => (0x00007fffaa7be000) libssl.so.6 => /lib64/libssl.so.6 (0x00002b21ccd9b000) libcrypto.so.6 => /lib64/libcrypto.so.6 (0x00002b21ccfea000) libc.so.6 => /lib64/libc.so.6 (0x00002b21cd33c000) libgssapi_krb5.so.2 => /usr/lib64/libgssapi_krb5.so.2 (0x00002b21cd695000) libkrb5.so.3 => /usr/lib64/libkrb5.so.3 (0x00002b21cd8c4000) libcom_err.so.2 => /lib64/libcom_err.so.2 (0x00002b21cdb59000) libk5crypto.so.3 => /usr/lib64/libk5crypto.so.3 (0x00002b21cdd5b000) libdl.so.2 => /lib64/libdl.so.2 (0x00002b21cdf81000) libz.so.1 => /lib64/libz.so.1 (0x00002b21ce185000) /lib64/ld-linux-x86-64.so.2 (0x0000003d98a00000) libkrb5support.so.0 => /usr/lib64/libkrb5support.so.0 (0x00002b21ce399000) libkeyutils.so.1 => /lib64/libkeyutils.so.1 (0x00002b21ce5a2000) libresolv.so.2 => /lib64/libresolv.so.2 (0x00002b21ce7a4000) libselinux.so.1 => /lib64/libselinux.so.1 (0x00002b21ce9ba000) libsepol.so.1 => /lib64/libsepol.so.1 (0x00002b21cebd2000)
BUILD	-sh-3.2$ ldd ./blib/arch/auto/Net/SSLeay/SSLeay.so linux-vdso.so.1 => (0x00007fff3ddfd000) libz.so.1 => /lib64/libz.so.1 (0x00002b5b0ab78000) libc.so.6 => /lib64/libc.so.6 (0x00002b5b0ad8d000) /lib64/ld-linux-x86-64.so.2 (0x0000003d98a00000)

Le versioni (installed e build) a confronto

/usr/lib64/perl5/vendor_perl/5.8.8/x86_64-linux-thread-multi	/home/C7502307/ssm/Net-SSLeay-1.88/blib/lib/
$HOME_INSTALL/**Net/SSLeay/Handle.pm**	$HOME_BUILD/**Net/SSLeay/Handle.pm**
$HOME_INSTALL/**Net/SSLeay.pm** $VERSION='1.30'	$HOME_BUILD/**Net/SSLeay.pm** $VERSION='1.88'
$HOME_INSTALL/**auto/Net/SSLey/autosplit.ix**	$HOME_BUILD/**auto/Net/SSLeay/autosplit.ix**
$HOME_INSTALL/**auto/Net/SSLey/*.al**	$HOME_BUILD/**auto/Net/SSLeay/*.al**
$HOME_INSTALL/**auto/Net/SSLey/SSLeay.so**	$HOME_BUILD/**auto/Net/SSLeay/SSLeay.so**

Seguono per completezza i file principali della versione compilata.

./blib/lib/Net/SSLeay/Handle.pm

```
-sh-3.2$ find . -name Handle.pm
./blib/lib/Net/SSLeay/Handle.pm
./lib/Net/SSLeay/Handle.pm

-sh-3.2$ diff ./blib/lib/Net/SSLeay/Handle.pm ./lib/Net/SSLeay/Handle.pm

-sh-3.2$ cat ./blib/lib/Net/SSLeay/Handle.pm
package Net::SSLeay::Handle;

use 5.8.1;

use strict;

use Socket;
use Net::SSLeay;

require Exporter;

=encoding utf-8

=head1 NAME

Net::SSLeay::Handle - Perl module that lets SSL (HTTPS) sockets be
handled as standard file handles.

=head1 SYNOPSIS

  use Net::SSLeay::Handle qw/shutdown/;
  my ($host, $port) = ("localhost", 443);

  tie(*SSL, "Net::SSLeay::Handle", $host, $port);

  print SSL "GET / HTTP/1.0\r\n";
  shutdown(\*SSL, 1);
  print while (<SSL>);
  close SSL;

=head1 DESCRIPTION

Net::SSLeay::Handle allows you to request and receive HTTPS web pages
using "old-fashion" file handles as in:

    print SSL "GET / HTTP/1.0\r\n";

and

    print while (<SSL>);

If you export the shutdown routine, then the only extra code that
you need to add to your program is the tie function as in:

    my $socket;
    if ($scheme eq "https") {
        tie(*S2, "Net::SSLeay::Handle", $host, $port);
        $socket = \*S2;
    else {
        $socket = Net::SSLeay::Handle->make_socket($host, $port);
```

```
    }
    print $socket $request_headers;
    ...

=cut

use vars qw(@ISA @EXPORT_OK $VERSION);
@ISA = qw(Exporter);
@EXPORT_OK = qw(shutdown);
$VERSION = '1.88';

my $Initialized;       #-- only _initialize() once
my $Debug = 0;         #-- pretty hokey

#== Tie Handle Methods ============================================================
#
# see perldoc perltie for details.
#
#==================================================================================

sub TIEHANDLE {
    my ($class, $socket, $port) = @_;
    $Debug > 10 and print "TIEHANDLE(@{[join ', ', @_]})\n";

    ref $socket eq "GLOB" or $socket = $class->make_socket($socket, $port);

    $class->_initialize();

    my $ctx = Net::SSLeay::CTX_new() or die_now("Failed to create SSL_CTX $!");
    my $ssl = Net::SSLeay::new($ctx) or die_now("Failed to create SSL $!");

    my $fileno = fileno($socket);

  Net::SSLeay::set_fd($ssl, $fileno);   # Must use fileno

    my $resp = Net::SSLeay::connect($ssl);

    $Debug and print "Cipher '" . Net::SSLeay::get_cipher($ssl) . "'\n";

        my $self = bless {
        ssl    => $ssl,
        ctx    => $ctx,
        socket => $socket,
        fileno => $fileno,
    }, $class;

    return $self;
}

sub PRINT {
    my $self = shift;

    my $ssl  = _get_ssl($self);
    my $resp = 0;
    for my $msg (@_) {
        defined $msg or last;
        $resp = Net::SSLeay::write($ssl, $msg) or last;
    }
    return $resp;
}

sub READLINE {
```

```
    my $self = shift;
    my $ssl  = _get_ssl($self);
        if (wantarray) {
                my @lines;
                while (my $line = Net::SSLeay::ssl_read_until($ssl)) {
                        push @lines, $line;
                }
                return @lines;
        } else {
                my $line = Net::SSLeay::ssl_read_until($ssl);
                return $line ? $line : undef;
        }
}

sub READ {
    my ($self, $buf, $len, $offset) = \ (@_);
    my $ssl = _get_ssl($$self);
    defined($$offset) or
      return length($$buf = Net::SSLeay::ssl_read_all($ssl, $$len));

    defined(my $read = Net::SSLeay::ssl_read_all($ssl, $$len))
      or return undef;

    my $buf_len = length($$buf);
    $$offset > $buf_len and $$buf .= chr(0) x ($$offset - $buf_len);
    substr($$buf, $$offset) = $read;
    return length($read);
}

sub WRITE {
    my $self = shift;
    my ($buf, $len, $offset) = @_;
    $offset = 0 unless defined $offset;

    # Return number of characters written.
    my $ssl  = $self->_get_ssl();
    return $len if Net::SSLeay::write($ssl, substr($buf, $offset, $len));
    return undef;
}

sub CLOSE {
    my $self = shift;
    my $fileno = $self->{fileno};
    $Debug > 10 and print "close($fileno)\n";
    Net::SSLeay::free ($self->{ssl});
    Net::SSLeay::CTX_free ($self->{ctx});
    close $self->{socket};
}

sub FILENO  { $_[0]->{fileno} }


=head1 FUNCTIONS

=over

=item shutdown

  shutdown(\*SOCKET, $mode)

Calls to the main shutdown() don't work with tied sockets created with this
module.  This shutdown should be able to distinquish between tied and untied
```

```
sockets and do the right thing.

=cut

sub shutdown {
    my ($obj, @params) = @_;

        my $socket = UNIVERSAL::isa($obj, 'Net::SSLeay::Handle') ?
                $obj->{socket} : $obj;
    return shutdown($socket, @params);
}

=item debug

  my $debug = Net::SSLeay::Handle->debug()
  Net::SSLeay::Handle->debug(1)

Get/set debugging mode. Always returns the debug value before the function call.
if an additional argument is given the debug option will be set to this value.

=cut

sub debug {
    my ($class, $debug) = @_;
    my $old_debug = $Debug;
    @_ >1 and $Debug = $debug || 0;
    return $old_debug;
}

#=== Internal Methods ============================================================

=item make_socket

  my $sock = Net::SSLeay::Handle->make_socket($host, $port);

Creates a socket that is connected to $post using $port. It uses
$Net::SSLeay::proxyhost and proxyport if set and authentificates itself against
this proxy depending on $Net::SSLeay::proxyauth. It also turns autoflush on for
the created socket.

=cut

sub make_socket {
    my ($class, $host, $port) = @_;
    $Debug > 10 and print "_make_socket(@{[join ', ', @_]})\n";
    $host ||= 'localhost';
    $port ||= 443;

    my $phost = $Net::SSLeay::proxyhost;
    my $pport = $Net::SSLeay::proxyhost ? $Net::SSLeay::proxyport : $port;

    my $dest_ip     = gethostbyname($phost || $host);
    my $host_params = sockaddr_in($pport, $dest_ip);

    socket(my $socket, &PF_INET(), &SOCK_STREAM(), 0) or die "socket: $!";
    connect($socket, $host_params)                    or die "connect: $!";

    my $old_select = select($socket); $| = 1; select($old_select);
    $phost and do {
        my $auth = $Net::SSLeay::proxyauth;
        my $CRLF = $Net::SSLeay::CRLF;
        print $socket "CONNECT $host:$port HTTP/1.0$auth$CRLF$CRLF";
```

```
        my $line = <$socket>;
    };
    return $socket;
}

=back

=cut

sub _initialize {
    $Initialized++ and return;
  Net::SSLeay::load_error_strings();
  Net::SSLeay::SSLeay_add_ssl_algorithms();
  Net::SSLeay::randomize();
}

sub __dummy {
    my $host = $Net::SSLeay::proxyhost;
    my $port = $Net::SSLeay::proxyport;
    my $auth = $Net::SSLeay::proxyauth;
}

#--- _get_self($socket) ---------------------------------------------------------
# Returns a hash containing attributes for $socket (= \*SOMETHING) based
# on fileno($socket).  Will return undef if $socket was not created here.
#--------------------------------------------------------------------------------

sub _get_self { return $_[0]; }

#--- _get_ssl($socket) ----------------------------------------------------------
# Returns a the "ssl" attribute for $socket (= \*SOMETHING) based
# on fileno($socket).  Will cause a warning and return undef if $socket was not
# created here.
#--------------------------------------------------------------------------------

sub _get_ssl {
    return $_[0]->{ssl};
}

1;

__END__
```

./blib/lib/Net/SSLeay.pm

```
-sh-3.2$ find . -name SSLeay.pm
./blib/lib/Net/SSLeay.pm
./lib/Net/SSLeay.pm

-sh-3.2$ diff ./blib/lib/Net/SSLeay.pm ./lib/Net/SSLeay.pm

-sh-3.2$ cat ./blib/lib/Net/SSLeay.pm

# Net::SSLeay.pm - Perl module for using Eric Young's implementation of SSL
#
# Copyright (c) 1996-2003 Sampo Kellomäki <sampo@iki.fi>
# Copyright (c) 2005-2010 Florian Ragwitz <rafl@debian.org>
```

```
# Copyright (c) 2005-2018 Mike McCauley <mikem@airspayce.com>
# Copyright (c) 2018- Chris Novakovic <chris@chrisn.me.uk>
# Copyright (c) 2018- Tuure Vartiainen <vartiait@radiatorsoftware.com>
# Copyright (c) 2018- Heikki Vatiainen <hvn@radiatorsoftware.com>
#
# All rights reserved.
#
# This module is released under the terms of the Artistic License 2.0. For
# details, see the LICENSE file distributed with Net-SSLeay's source code.

package Net::SSLeay;

use 5.8.1;

use strict;
use Carp;
use vars qw($VERSION @ISA @EXPORT @EXPORT_OK $AUTOLOAD $CRLF);
use Socket;
use Errno;

require Exporter;
use AutoLoader;

# 0=no warns, 1=only errors, 2=ciphers, 3=progress, 4=dump data
$Net::SSLeay::trace = 0;  # Do not change here, use
                          # $Net::SSLeay::trace = [1-4]  in caller

# 2 = insist on v2 SSL protocol
# 3 = insist on v3 SSL
# 10 = insist on TLSv1
# 11 = insist on TLSv1.1
# 12 = insist on TLSv1.2
# 13 = insist on TLSv1.3
# 0 or undef = guess (v23)
#
$Net::SSLeay::ssl_version = 0;  # don't change here, use
                                # Net::SSLeay::version=[2,3,0] in caller

#define to enable the "cat /proc/$$/stat" stuff
$Net::SSLeay::linux_debug = 0;

# Number of seconds to sleep after sending message and before half
# closing connection. Useful with antiquated broken servers.
$Net::SSLeay::slowly = 0;

# RANDOM NUMBER INITIALIZATION
#
# Edit to your taste. Using /dev/random would be more secure, but may
# block if randomness is not available, thus the default is
# /dev/urandom. $how_random determines how many bits of randomness to take
# from the device. You should take enough (read SSLeay/doc/rand), but
# beware that randomness is limited resource so you should not waste
# it either or you may end up with randomness depletion (situation where
# /dev/random would block and /dev/urandom starts to return predictable
# numbers).
#
# N.B. /dev/urandom does not exist on all systems, such as Solaris 2.6. In that
#      case you should get a third party package that emulates /dev/urandom
#      (e.g. via named pipe) or supply a random number file. Some such
#      packages are documented in Caveat section of the POD documentation.

$Net::SSLeay::random_device = '/dev/urandom';
```

```
$Net::SSLeay::how_random = 512;

$VERSION = '1.88'; # Also update $Net::SSLeay::Handle::VERSION
@ISA = qw(Exporter);

#BEWARE:
# 3-columns part of @EXPORT_OK related to constants is the output of command:
# perl helper_script/regen_openssl_constants.pl -gen-pod
# if you add/remove any constant you need to update it manually

@EXPORT_OK = qw(
 ASN1_STRFLGS_ESC_CTRL           NID_netscape
R_UNKNOWN_REMOTE_ERROR_TYPE
 ASN1_STRFLGS_ESC_MSB            NID_netscape_base_url
R_UNKNOWN_STATE
 ASN1_STRFLGS_ESC_QUOTE          NID_netscape_ca_policy_url               R_X509_LIB
 ASN1_STRFLGS_RFC2253            NID_netscape_ca_revocation_url
SENT_SHUTDOWN
 CB_ACCEPT_EXIT                  NID_netscape_cert_extension
SESSION_ASN1_VERSION
 CB_ACCEPT_LOOP                  NID_netscape_cert_sequence
SESS_CACHE_BOTH
 CB_ALERT                        NID_netscape_cert_type
SESS_CACHE_CLIENT
 CB_CONNECT_EXIT                 NID_netscape_comment
SESS_CACHE_NO_AUTO_CLEAR
 CB_CONNECT_LOOP                 NID_netscape_data_type
SESS_CACHE_NO_INTERNAL
 CB_EXIT                         NID_netscape_renewal_url
SESS_CACHE_NO_INTERNAL_LOOKUP
 CB_HANDSHAKE_DONE               NID_netscape_revocation_url
SESS_CACHE_NO_INTERNAL_STORE
 CB_HANDSHAKE_START              NID_netscape_ssl_server_name
SESS_CACHE_OFF
 CB_LOOP                         NID_ns_sgc
SESS_CACHE_SERVER
 CB_READ                         NID_organizationName
SSL3_VERSION
 CB_READ_ALERT                   NID_organizationalUnitName
SSLEAY_BUILT_ON
 CB_WRITE                        NID_pbeWithMD2AndDES_CBC
SSLEAY_CFLAGS
 CB_WRITE_ALERT                  NID_pbeWithMD2AndRC2_CBC                 SSLEAY_DIR
 ERROR_NONE                      NID_pbeWithMD5AndCast5_CBC
SSLEAY_PLATFORM
 ERROR_SSL                       NID_pbeWithMD5AndDES_CBC
SSLEAY_VERSION
 ERROR_SYSCALL                   NID_pbeWithMD5AndRC2_CBC                 ST_ACCEPT
 ERROR_WANT_ACCEPT               NID_pbeWithSHA1AndDES_CBC                ST_BEFORE
 ERROR_WANT_CONNECT              NID_pbeWithSHA1AndRC2_CBC                ST_CONNECT
 ERROR_WANT_READ                 NID_pbe_WithSHA1And128BitRC2_CBC         ST_INIT
 ERROR_WANT_WRITE                NID_pbe_WithSHA1And128BitRC4             ST_OK
 ERROR_WANT_X509_LOOKUP          NID_pbe_WithSHA1And2_Key_TripleDES_CBC
ST_READ_BODY
 ERROR_ZERO_RETURN               NID_pbe_WithSHA1And3_Key_TripleDES_CBC
ST_READ_HEADER
 EVP_PKS_DSA                     NID_pbe_WithSHA1And40BitRC2_CBC
TLS1_1_VERSION
 EVP_PKS_EC                      NID_pbe_WithSHA1And40BitRC4
TLS1_2_VERSION
 EVP_PKS_RSA                     NID_pbes2
TLS1_3_VERSION
```

```
 EVP_PKT_ENC                     NID_pbmac1
TLS1_VERSION
 EVP_PKT_EXCH                    NID_pkcs
TLSEXT_STATUSTYPE_ocsp
 EVP_PKT_EXP                     NID_pkcs3
VERIFY_CLIENT_ONCE
 EVP_PKT_SIGN                    NID_pkcs7
VERIFY_FAIL_IF_NO_PEER_CERT
 EVP_PK_DH                       NID_pkcs7_data                          VERIFY_NONE
 EVP_PK_DSA                      NID_pkcs7_digest                        VERIFY_PEER
 EVP_PK_EC                       NID_pkcs7_encrypted
VERIFY_POST_HANDSHAKE
 EVP_PK_RSA                      NID_pkcs7_enveloped
V_OCSP_CERTSTATUS_GOOD
 FILETYPE_ASN1                   NID_pkcs7_signed
V_OCSP_CERTSTATUS_REVOKED
 FILETYPE_PEM                    NID_pkcs7_signedAndEnveloped
V_OCSP_CERTSTATUS_UNKNOWN
 F_CLIENT_CERTIFICATE            NID_pkcs8ShroudedKeyBag                 WRITING
 F_CLIENT_HELLO                  NID_pkcs9
X509_CHECK_FLAG_ALWAYS_CHECK_SUBJECT
 F_CLIENT_MASTER_KEY             NID_pkcs9_challengePassword
X509_CHECK_FLAG_MULTI_LABEL_WILDCARDS
 F_D2I_SSL_SESSION               NID_pkcs9_contentType
X509_CHECK_FLAG_NEVER_CHECK_SUBJECT
 F_GET_CLIENT_FINISHED           NID_pkcs9_countersignature
X509_CHECK_FLAG_NO_PARTIAL_WILDCARDS
 F_GET_CLIENT_HELLO              NID_pkcs9_emailAddress
X509_CHECK_FLAG_NO_WILDCARDS
 F_GET_CLIENT_MASTER_KEY         NID_pkcs9_extCertAttributes
X509_CHECK_FLAG_SINGLE_LABEL_SUBDOMAINS
 F_GET_SERVER_FINISHED           NID_pkcs9_messageDigest
X509_FILETYPE_ASN1
 F_GET_SERVER_HELLO              NID_pkcs9_signingTime
X509_FILETYPE_DEFAULT
 F_GET_SERVER_VERIFY             NID_pkcs9_unstructuredAddress
X509_FILETYPE_PEM
 F_I2D_SSL_SESSION               NID_pkcs9_unstructuredName              X509_LOOKUP
 F_READ_N                        NID_private_key_usage_period
X509_PURPOSE_ANY
 F_REQUEST_CERTIFICATE           NID_rc2_40_cbc
X509_PURPOSE_CRL_SIGN
 F_SERVER_HELLO                  NID_rc2_64_cbc
X509_PURPOSE_NS_SSL_SERVER
 F_SSL_CERT_NEW                  NID_rc2_cbc
X509_PURPOSE_OCSP_HELPER
 F_SSL_GET_NEW_SESSION           NID_rc2_cfb64
X509_PURPOSE_SMIME_ENCRYPT
 F_SSL_NEW                       NID_rc2_ecb
X509_PURPOSE_SMIME_SIGN
 F_SSL_READ                      NID_rc2_ofb64
X509_PURPOSE_SSL_CLIENT
 F_SSL_RSA_PRIVATE_DECRYPT       NID_rc4
X509_PURPOSE_SSL_SERVER
 F_SSL_RSA_PUBLIC_ENCRYPT        NID_rc4_40
X509_PURPOSE_TIMESTAMP_SIGN
 F_SSL_SESSION_NEW               NID_rc5_cbc
X509_TRUST_COMPAT
 F_SSL_SESSION_PRINT_FP          NID_rc5_cfb64
X509_TRUST_EMAIL
 F_SSL_SET_FD                    NID_rc5_ecb
X509_TRUST_OBJECT_SIGN
```

```
 F_SSL_SET_RFD                   NID_rc5_ofb64
X509_TRUST_OCSP_REQUEST
 F_SSL_SET_WFD                   NID_ripemd160
X509_TRUST_OCSP_SIGN
 F_SSL_USE_CERTIFICATE           NID_ripemd160WithRSA
X509_TRUST_SSL_CLIENT
 F_SSL_USE_CERTIFICATE_ASN1      NID_rle_compression
X509_TRUST_SSL_SERVER
 F_SSL_USE_CERTIFICATE_FILE      NID_rsa
X509_TRUST_TSA
 F_SSL_USE_PRIVATEKEY            NID_rsaEncryption
X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH
 F_SSL_USE_PRIVATEKEY_ASN1       NID_rsadsi
X509_V_ERR_AKID_SKID_MISMATCH
 F_SSL_USE_PRIVATEKEY_FILE       NID_safeContentsBag
X509_V_ERR_APPLICATION_VERIFICATION
 F_SSL_USE_RSAPRIVATEKEY         NID_sdsiCertificate
X509_V_ERR_CA_KEY_TOO_SMALL
 F_SSL_USE_RSAPRIVATEKEY_ASN1    NID_secretBag
X509_V_ERR_CA_MD_TOO_WEAK
 F_SSL_USE_RSAPRIVATEKEY_FILE    NID_serialNumber
X509_V_ERR_CERT_CHAIN_TOO_LONG
 F_WRITE_PENDING                 NID_server_auth
X509_V_ERR_CERT_HAS_EXPIRED
 GEN_DIRNAME                     NID_sha
X509_V_ERR_CERT_NOT_YET_VALID
 GEN_DNS                         NID_sha1
X509_V_ERR_CERT_REJECTED
 GEN_EDIPARTY                    NID_sha1WithRSA
X509_V_ERR_CERT_REVOKED
 GEN_EMAIL                       NID_sha1WithRSAEncryption
X509_V_ERR_CERT_SIGNATURE_FAILURE
 GEN_IPADD                       NID_shaWithRSAEncryption
X509_V_ERR_CERT_UNTRUSTED
 GEN_OTHERNAME                   NID_stateOrProvinceName
X509_V_ERR_CRL_HAS_EXPIRED
 GEN_RID                         NID_subject_alt_name
X509_V_ERR_CRL_NOT_YET_VALID
 GEN_URI                         NID_subject_key_identifier
X509_V_ERR_CRL_PATH_VALIDATION_ERROR
 GEN_X400                        NID_surname
X509_V_ERR_CRL_SIGNATURE_FAILURE
 LIBRESSL_VERSION_NUMBER         NID_sxnet
X509_V_ERR_DANE_NO_MATCH
 MBSTRING_ASC                    NID_time_stamp
X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT
 MBSTRING_BMP                    NID_title
X509_V_ERR_DIFFERENT_CRL_SCOPE
 MBSTRING_FLAG                   NID_undef
X509_V_ERR_EE_KEY_TOO_SMALL
 MBSTRING_UNIV                   NID_uniqueIdentifier
X509_V_ERR_EMAIL_MISMATCH
 MBSTRING_UTF8                   NID_x509Certificate
X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD
 MIN_RSA_MODULUS_LENGTH_IN_BYTES NID_x509Crl
X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD
 MODE_ACCEPT_MOVING_WRITE_BUFFER NID_zlib_compression
X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD
 MODE_AUTO_RETRY                 NOTHING
X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD
 MODE_ENABLE_PARTIAL_WRITE       OCSP_RESPONSE_STATUS_INTERNALERROR
X509_V_ERR_EXCLUDED_VIOLATION
```

```
 MODE_RELEASE_BUFFERS            OCSP_RESPONSE_STATUS_MALFORMEDREQUEST
X509_V_ERR_HOSTNAME_MISMATCH
 NID_OCSP_sign                   OCSP_RESPONSE_STATUS_SIGREQUIRED
X509_V_ERR_INVALID_CA
 NID_SMIMECapabilities           OCSP_RESPONSE_STATUS_SUCCESSFUL
X509_V_ERR_INVALID_CALL
 NID_X500                        OCSP_RESPONSE_STATUS_TRYLATER
X509_V_ERR_INVALID_EXTENSION
 NID_X509                        OCSP_RESPONSE_STATUS_UNAUTHORIZED
X509_V_ERR_INVALID_NON_CA
 NID_ad_OCSP                     OPENSSL_BUILT_ON
X509_V_ERR_INVALID_POLICY_EXTENSION
 NID_ad_ca_issuers               OPENSSL_CFLAGS
X509_V_ERR_INVALID_PURPOSE
 NID_algorithm                   OPENSSL_DIR
X509_V_ERR_IP_ADDRESS_MISMATCH
 NID_authority_key_identifier    OPENSSL_ENGINES_DIR
X509_V_ERR_KEYUSAGE_NO_CERTSIGN
 NID_basic_constraints           OPENSSL_PLATFORM
X509_V_ERR_KEYUSAGE_NO_CRL_SIGN
 NID_bf_cbc                      OPENSSL_VERSION
X509_V_ERR_KEYUSAGE_NO_DIGITAL_SIGNATURE
 NID_bf_cfb64                    OPENSSL_VERSION_NUMBER
X509_V_ERR_NO_EXPLICIT_POLICY
 NID_bf_ecb                      OP_ALL
X509_V_ERR_NO_VALID_SCTS
 NID_bf_ofb64                    OP_ALLOW_NO_DHE_KEX
X509_V_ERR_OCSP_CERT_UNKNOWN
 NID_cast5_cbc                   OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION
X509_V_ERR_OCSP_VERIFY_FAILED
 NID_cast5_cfb64                 OP_CIPHER_SERVER_PREFERENCE
X509_V_ERR_OCSP_VERIFY_NEEDED
 NID_cast5_ecb                   OP_CISCO_ANYCONNECT
X509_V_ERR_OUT_OF_MEM
 NID_cast5_ofb64                 OP_COOKIE_EXCHANGE
X509_V_ERR_PATH_LENGTH_EXCEEDED
 NID_certBag                     OP_CRYPTOPRO_TLSEXT_BUG
X509_V_ERR_PATH_LOOP
 NID_certificate_policies        OP_DONT_INSERT_EMPTY_FRAGMENTS
X509_V_ERR_PERMITTED_VIOLATION
 NID_client_auth                 OP_ENABLE_MIDDLEBOX_COMPAT
X509_V_ERR_PROXY_CERTIFICATES_NOT_ALLOWED
 NID_code_sign                   OP_EPHEMERAL_RSA
X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED
 NID_commonName                  OP_LEGACY_SERVER_CONNECT
X509_V_ERR_PROXY_SUBJECT_NAME_VIOLATION
 NID_countryName                 OP_MICROSOFT_BIG_SSLV3_BUFFER
X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN
 NID_crlBag                      OP_MICROSOFT_SESS_ID_BUG
X509_V_ERR_STORE_LOOKUP
 NID_crl_distribution_points     OP_MSIE_SSLV2_RSA_PADDING
X509_V_ERR_SUBJECT_ISSUER_MISMATCH
 NID_crl_number                  OP_NETSCAPE_CA_DN_BUG
X509_V_ERR_SUBTREE_MINMAX
 NID_crl_reason                  OP_NETSCAPE_CHALLENGE_BUG
X509_V_ERR_SUITE_B_CANNOT_SIGN_P_384_WITH_P_256
 NID_delta_crl                   OP_NETSCAPE_DEMO_CIPHER_CHANGE_BUG
X509_V_ERR_SUITE_B_INVALID_ALGORITHM
 NID_des_cbc                     OP_NETSCAPE_REUSE_CIPHER_CHANGE_BUG
X509_V_ERR_SUITE_B_INVALID_CURVE
 NID_des_cfb64                   OP_NON_EXPORT_FIRST
X509_V_ERR_SUITE_B_INVALID_SIGNATURE_ALGORITHM
```

```
 NID_des_ecb                     OP_NO_ANTI_REPLAY
X509_V_ERR_SUITE_B_INVALID_VERSION
 NID_des_ede                     OP_NO_CLIENT_RENEGOTIATION
X509_V_ERR_SUITE_B_LOS_NOT_ALLOWED
 NID_des_ede3                    OP_NO_COMPRESSION
X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY
 NID_des_ede3_cbc                OP_NO_ENCRYPT_THEN_MAC
X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE
 NID_des_ede3_cfb64              OP_NO_QUERY_MTU
X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE
 NID_des_ede3_ofb64              OP_NO_RENEGOTIATION
X509_V_ERR_UNABLE_TO_GET_CRL
 NID_des_ede_cbc                 OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION
X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER
 NID_des_ede_cfb64               OP_NO_SSL_MASK
X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT
 NID_des_ede_ofb64               OP_NO_SSLv2
X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY
 NID_des_ofb64                   OP_NO_SSLv3
X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE
 NID_description                 OP_NO_TICKET
X509_V_ERR_UNHANDLED_CRITICAL_CRL_EXTENSION
 NID_desx_cbc                    OP_NO_TLSv1
X509_V_ERR_UNHANDLED_CRITICAL_EXTENSION
 NID_dhKeyAgreement              OP_NO_TLSv1_1
X509_V_ERR_UNNESTED_RESOURCE
 NID_dnQualifier                 OP_NO_TLSv1_2
X509_V_ERR_UNSPECIFIED
 NID_dsa                         OP_NO_TLSv1_3
X509_V_ERR_UNSUPPORTED_CONSTRAINT_SYNTAX
 NID_dsaWithSHA                  OP_PKCS1_CHECK_1
X509_V_ERR_UNSUPPORTED_CONSTRAINT_TYPE
 NID_dsaWithSHA1                 OP_PKCS1_CHECK_2
X509_V_ERR_UNSUPPORTED_EXTENSION_FEATURE
 NID_dsaWithSHA1_2               OP_PRIORITIZE_CHACHA
X509_V_ERR_UNSUPPORTED_NAME_SYNTAX
 NID_dsa_2                       OP_SAFARI_ECDHE_ECDSA_BUG
X509_V_FLAG_ALLOW_PROXY_CERTS
 NID_email_protect               OP_SINGLE_DH_USE
X509_V_FLAG_CB_ISSUER_CHECK
 NID_ext_key_usage               OP_SINGLE_ECDH_USE
X509_V_FLAG_CHECK_SS_SIGNATURE
 NID_ext_req                     OP_SSLEAY_080_CLIENT_DH_BUG
X509_V_FLAG_CRL_CHECK
 NID_friendlyName                OP_SSLREF2_REUSE_CERT_TYPE_BUG
X509_V_FLAG_CRL_CHECK_ALL
 NID_givenName                   OP_TLSEXT_PADDING
X509_V_FLAG_EXPLICIT_POLICY
 NID_hmacWithSHA1                OP_TLS_BLOCK_PADDING_BUG
X509_V_FLAG_EXTENDED_CRL_SUPPORT
 NID_id_ad                       OP_TLS_D5_BUG
X509_V_FLAG_IGNORE_CRITICAL
 NID_id_ce                       OP_TLS_ROLLBACK_BUG
X509_V_FLAG_INHIBIT_ANY
 NID_id_kp                       READING
X509_V_FLAG_INHIBIT_MAP
 NID_id_pbkdf2                   RECEIVED_SHUTDOWN
X509_V_FLAG_NOTIFY_POLICY
 NID_id_pe                       RSA_3
X509_V_FLAG_NO_ALT_CHAINS
 NID_id_pkix                     RSA_F4
X509_V_FLAG_NO_CHECK_TIME
```

```
 NID_id_qt_cps                   R_BAD_AUTHENTICATION_TYPE
X509_V_FLAG_PARTIAL_CHAIN
 NID_id_qt_unotice               R_BAD_CHECKSUM
X509_V_FLAG_POLICY_CHECK
 NID_idea_cbc                    R_BAD_MAC_DECODE
X509_V_FLAG_POLICY_MASK
 NID_idea_cfb64                  R_BAD_RESPONSE_ARGUMENT
X509_V_FLAG_SUITEB_128_LOS
 NID_idea_ecb                    R_BAD_SSL_FILETYPE
X509_V_FLAG_SUITEB_128_LOS_ONLY
 NID_idea_ofb64                  R_BAD_SSL_SESSION_ID_LENGTH
X509_V_FLAG_SUITEB_192_LOS
 NID_info_access                 R_BAD_STATE
X509_V_FLAG_TRUSTED_FIRST
 NID_initials                    R_BAD_WRITE_RETRY
X509_V_FLAG_USE_CHECK_TIME
 NID_invalidity_date             R_CHALLENGE_IS_DIFFERENT
X509_V_FLAG_USE_DELTAS
 NID_issuer_alt_name             R_CIPHER_TABLE_SRC_ERROR
X509_V_FLAG_X509_STRICT
 NID_keyBag                      R_INVALID_CHALLENGE_LENGTH              X509_V_OK
 NID_key_usage                   R_NO_CERTIFICATE_SET
XN_FLAG_COMPAT
 NID_localKeyID                  R_NO_CERTIFICATE_SPECIFIED
XN_FLAG_DN_REV
 NID_localityName                R_NO_CIPHER_LIST
XN_FLAG_DUMP_UNKNOWN_FIELDS
 NID_md2                         R_NO_CIPHER_MATCH
XN_FLAG_FN_ALIGN
 NID_md2WithRSAEncryption        R_NO_PRIVATEKEY
XN_FLAG_FN_LN
 NID_md5                         R_NO_PUBLICKEY
XN_FLAG_FN_MASK
 NID_md5WithRSA                  R_NULL_SSL_CTX
XN_FLAG_FN_NONE
 NID_md5WithRSAEncryption        R_PEER_DID_NOT_RETURN_A_CERTIFICATE
XN_FLAG_FN_OID
 NID_md5_sha1                    R_PEER_ERROR
XN_FLAG_FN_SN
 NID_mdc2                        R_PEER_ERROR_CERTIFICATE
XN_FLAG_MULTILINE
 NID_mdc2WithRSA                 R_PEER_ERROR_NO_CIPHER
XN_FLAG_ONELINE
 NID_ms_code_com                 R_PEER_ERROR_UNSUPPORTED_CERTIFICATE_TYPE
XN_FLAG_RFC2253
 NID_ms_code_ind                 R_PUBLIC_KEY_ENCRYPT_ERROR
XN_FLAG_SEP_COMMA_PLUS
 NID_ms_ctl_sign                 R_PUBLIC_KEY_IS_NOT_RSA
XN_FLAG_SEP_CPLUS_SPC
 NID_ms_efs                      R_READ_WRONG_PACKET_TYPE
XN_FLAG_SEP_MASK
 NID_ms_ext_req                  R_SHORT_READ
XN_FLAG_SEP_MULTILINE
 NID_ms_sgc                      R_SSL_SESSION_ID_IS_DIFFERENT
XN_FLAG_SEP_SPLUS_SPC
 NID_name                        R_UNABLE_TO_EXTRACT_PUBLIC_KEY
XN_FLAG_SPC_EQ
    BIO_eof
    BIO_f_ssl
    BIO_free
    BIO_new
    BIO_new_file
```

```
    BIO_pending
    BIO_read
    BIO_s_mem
    BIO_wpending
    BIO_write
    CTX_free
    CTX_get_cert_store
    CTX_new
    CTX_use_RSAPrivateKey_file
    CTX_use_certificate_file
    CTX_v23_new
    CTX_v2_new
    CTX_v3_new
    ERR_error_string
    ERR_get_error
    ERR_load_RAND_strings
    ERR_load_SSL_strings
    PEM_read_bio_X509_CRL
    RSA_free
    RSA_generate_key
    SESSION
    SESSION_free
    SESSION_get_master_key
    SESSION_new
    SESSION_print
    X509_NAME_get_text_by_NID
    X509_NAME_oneline
    X509_STORE_CTX_set_flags
    X509_STORE_add_cert
    X509_STORE_add_crl
    X509_check_email
    X509_check_host
    X509_check_ip
    X509_check_ip_asc
    X509_free
    X509_get_issuer_name
    X509_get_subject_name
    X509_load_cert_crl_file
    X509_load_cert_file
    X509_load_crl_file
    accept
    add_session
    clear
    clear_error
    connect
    copy_session_id
    d2i_SSL_SESSION
    die_if_ssl_error
    die_now
    do_https
    dump_peer_certificate
    err
    flush_sessions
    free
    get_cipher
    get_cipher_list
    get_client_random
    get_fd
    get_http
    get_http4
    get_https
    get_https3
```

```
    get_https4
    get_httpx
    get_httpx4
    get_peer_certificate
    get_peer_cert_chain
    get_rbio
    get_read_ahead
    get_server_random
    get_shared_ciphers
    get_time
    get_timeout
    get_wbio
    i2d_SSL_SESSION
    load_error_strings
    make_form
    make_headers
    new
    peek
    pending
    post_http
    post_http4
    post_https
    post_https3
    post_https4
    post_httpx
    post_httpx4
    print_errs
    read
    remove_session
    rstate_string
    rstate_string_long
    set_bio
    set_cert_and_key
    set_cipher_list
    set_fd
    set_read_ahead
    set_rfd
    set_server_cert_and_key
    set_session
    set_time
    set_timeout
    set_verify
    set_wfd
    ssl_read_CRLF
    ssl_read_all
    ssl_read_until
    ssl_write_CRLF
    ssl_write_all
    sslcat
    state_string
    state_string_long
    tcp_read_CRLF
    tcp_read_all
    tcp_read_until
    tcp_write_CRLF
    tcp_write_all
    tcpcat
    tcpxcat
    use_PrivateKey
    use_PrivateKey_ASN1
    use_PrivateKey_file
    use_RSAPrivateKey
```

```
    use_RSAPrivateKey_ASN1
    use_RSAPrivateKey_file
    use_certificate
    use_certificate_ASN1
    use_certificate_file
    write
    d2i_OCSP_RESPONSE
    i2d_OCSP_RESPONSE
    OCSP_RESPONSE_free
    d2i_OCSP_REQUEST
    i2d_OCSP_REQUEST
    OCSP_REQUEST_free
    OCSP_cert2ids
    OCSP_ids2req
    OCSP_response_status
    OCSP_response_status_str
    OCSP_response_verify
    OCSP_response_results
    OCSP_RESPONSE_STATUS_INTERNALERROR
    OCSP_RESPONSE_STATUS_MALFORMEDREQUEST
    OCSP_RESPONSE_STATUS_SIGREQUIRED
    OCSP_RESPONSE_STATUS_SUCCESSFUL
    OCSP_RESPONSE_STATUS_TRYLATER
    OCSP_RESPONSE_STATUS_UNAUTHORIZED
    TLSEXT_STATUSTYPE_ocsp
    V_OCSP_CERTSTATUS_GOOD
    V_OCSP_CERTSTATUS_REVOKED
    V_OCSP_CERTSTATUS_UNKNOWN
);

sub AUTOLOAD {
    # This AUTOLOAD is used to 'autoload' constants from the constant()
    # XS function.  If a constant is not found then control is passed
    # to the AUTOLOAD in AutoLoader.

    my $constname;
    ($constname = $AUTOLOAD) =~ s/.*::///;
    my $val = constant($constname);
    if ($! != 0) {
        if ($! =~ /((Invalid)|(not valid))/i || $!{EINVAL}) {
            $AutoLoader::AUTOLOAD = $AUTOLOAD;
            goto &AutoLoader::AUTOLOAD;
        }
        else {
          croak "Your vendor has not defined SSLeay macro $constname";
        }
    }
    eval "sub $AUTOLOAD { $val }";
    goto &$AUTOLOAD;
}

eval {
        require XSLoader;
        XSLoader::load('Net::SSLeay', $VERSION);
        1;
} or do {
        require DynaLoader;
        push @ISA, 'DynaLoader';
        bootstrap Net::SSLeay $VERSION;
};

# Preloaded methods go here.
```

```
$CRLF = "\x0d\x0a";  # because \r\n is not fully portable

### Print SSLeay error stack

sub print_errs {
    my ($msg) = @_;
    my ($count, $err, $errs, $e) = (0,0,'');
    while ($err = ERR_get_error()) {
        $count ++;
        $e = "$msg $$: $count - " . ERR_error_string($err) . "\n";
        $errs .= $e;
        warn $e if $Net::SSLeay::trace;
    }
    return $errs;
}

# Death is conditional to SSLeay errors existing, i.e. this function checks
# for errors and only dies in affirmative.
# usage: Net::SSLeay::write($ssl, "foo") or die_if_ssl_error("SSL write ($!)");

sub die_if_ssl_error {
    my ($msg) = @_;
    die "$$: $msg\n" if print_errs($msg);
}

# Unconditional death. Used to print SSLeay errors before dying.
# usage: Net::SSLeay::connect($ssl) or die_now("Failed SSL connect ($!)");

sub die_now {
    my ($msg) = @_;
    print_errs($msg);
    die "$$: $msg\n";
}

# Perl 5.6.* unicode support causes that length() no longer reliably
# reflects the byte length of a string. This eval is to fix that.
# Thanks to Sean Burke for the snippet.

BEGIN{
eval 'use bytes; sub blength ($) { defined $_[0] ? length $_[0] : 0  }';
$@ and eval '    sub blength ($) { defined $_[0] ? length $_[0] : 0 }' ;
}

# Autoload methods go after __END__, and are processed by the autosplit program.


1;
__END__

### Some methods that are macros in C

sub want_nothing { want(shift) == 1 }
sub want_read { want(shift) == 2 }
sub want_write { want(shift) == 3 }
sub want_X509_lookup { want(shift) == 4 }

###
### Open TCP stream to given host and port, looking up the details
### from system databases or DNS.
###
```

```
sub open_tcp_connection {
    my ($dest_serv, $port) = @_;
    my ($errs);

    $port = getservbyname($port, 'tcp') unless $port =~ /^\d+$/;
    my $dest_serv_ip = gethostbyname($dest_serv);
    unless (defined($dest_serv_ip)) {
        $errs = "$0 $$: open_tcp_connection: destination host not found:"
            . " `$dest_serv' (port $port) ($!)\n";
        warn $errs if $trace;
        return wantarray ? (0, $errs) : 0;
    }
    my $sin = sockaddr_in($port, $dest_serv_ip);

    warn "Opening connection to $dest_serv:$port (" .
        inet_ntoa($dest_serv_ip) . ")" if $trace>2;

    my $proto = &Socket::IPPROTO_TCP; # getprotobyname('tcp') not available on android
    if (socket (SSLCAT_S, &PF_INET(), &SOCK_STREAM(), $proto)) {
        warn "next connect" if $trace>3;
        if (CORE::connect (SSLCAT_S, $sin)) {
            my $old_out = select (SSLCAT_S); $| = 1; select ($old_out);
            warn "connected to $dest_serv, $port" if $trace>3;
            return wantarray ? (1, undef) : 1; # Success
        }
    }
    $errs = "$0 $$: open_tcp_connection: failed `$dest_serv', $port ($!)\n";
    warn $errs if $trace;
    close SSLCAT_S;
    return wantarray ? (0, $errs) : 0; # Fail
}

### Open connection via standard web proxy, if one was defined
### using set_proxy().

sub open_proxy_tcp_connection {
    my ($dest_serv, $port) = @_;
    return open_tcp_connection($dest_serv, $port) if !$proxyhost;

    warn "Connect via proxy: $proxyhost:$proxyport" if $trace>2;
    my ($ret, $errs) = open_tcp_connection($proxyhost, $proxyport);
    return wantarray ? (0, $errs) : 0 if !$ret;  # Connection fail

    warn "Asking proxy to connect to $dest_serv:$port" if $trace>2;
    #print SSLCAT_S "CONNECT $dest_serv:$port HTTP/1.0$proxyauth$CRLF$CRLF";
    #my $line = <SSLCAT_S>;   # *** bug? Mixing stdio with syscall read?
    ($ret, $errs) =
        tcp_write_all("CONNECT $dest_serv:$port HTTP/1.0$proxyauth$CRLF$CRLF");
    return wantarray ? (0,$errs) : 0 if $errs;
    ($line, $errs) = tcp_read_until($CRLF . $CRLF, 1024);
    warn "Proxy response: $line" if $trace>2;
    return wantarray ? (0,$errs) : 0 if $errs;
    return wantarray ? (1,'') : 1;  # Success
}

###
### read and write helpers that block
###

sub debug_read {
    my ($replyr, $gotr) = @_;
    my $vm = $trace>2 && $linux_debug ?
```

```
        (split ' ', `cat /proc/$$/stat`)[22] : 'vm_unknown';
    warn "  got " . blength($$gotr) . ':'
        . blength($$replyr) . " bytes (VM=$vm).\n" if $trace == 3;
    warn "  got `$$gotr' (" . blength($$gotr) . ':'
        . blength($$replyr) . " bytes, VM=$vm)\n" if $trace>3;
}

sub ssl_read_all {
    my ($ssl,$how_much) = @_;
    $how_much = 2000000000 unless $how_much;
    my ($got, $rv, $errs);
    my $reply = '';

    while ($how_much > 0) {
        ($got, $rv) = Net::SSLeay::read($ssl,
                ($how_much > 32768) ? 32768 : $how_much
        );
        if (! defined $got) {
            my $err = Net::SSLeay::get_error($ssl, $rv);
            if ($err != Net::SSLeay::ERROR_WANT_READ() and
                $err != Net::SSLeay::ERROR_WANT_WRITE()) {
                $errs = print_errs('SSL_read');
                last;
            }
            next;
        }
        $how_much -= blength($got);
        debug_read(\$reply, \$got) if $trace>1;
        last if $got eq '';  # EOF
        $reply .= $got;
    }

    return wantarray ? ($reply, $errs) : $reply;
}

sub tcp_read_all {
    my ($how_much) = @_;
    $how_much = 2000000000 unless $how_much;
    my ($n, $got, $errs);
    my $reply = '';

    my $bsize = 0x10000;
    while ($how_much > 0) {
        $n = sysread(SSLCAT_S,$got, (($bsize < $how_much) ? $bsize : $how_much));
        warn "Read error: $! ($n,$how_much)" unless defined $n;
        last if !$n;  # EOF
        $how_much -= $n;
        debug_read(\$reply, \$got) if $trace>1;
        $reply .= $got;
    }
    return wantarray ? ($reply, $errs) : $reply;
}

sub ssl_write_all {
    my $ssl = $_[0];
    my ($data_ref, $errs);
    if (ref $_[1]) {
        $data_ref = $_[1];
    } else {
        $data_ref = \$_[1];
    }
    my ($wrote, $written, $to_write) = (0,0, blength($$data_ref));
```

```
    my $vm = $trace>2 && $linux_debug ?
        (split ' ', `cat /proc/$$/stat`)[22] : 'vm_unknown';
    warn "  write_all VM at entry=$vm\n" if $trace>2;
    while ($to_write) {
        #sleep 1; # *** DEBUG
        warn "partial `$$data_ref'\n" if $trace>3;
        $wrote = write_partial($ssl, $written, $to_write, $$data_ref);
        if (defined $wrote && ($wrote > 0)) {  # write_partial can return -1
            $written += $wrote;
            $to_write -= $wrote;
        } else {
          if (defined $wrote) {
            # check error conditions via SSL_get_error per man page
            if ( my $sslerr = get_error($ssl, $wrote) ) {
              my $errstr = ERR_error_string($sslerr);
              my $errname = '';
              SWITCH: {
                $sslerr == constant("ERROR_NONE") && do {
                  # according to map page SSL_get_error(3ssl):
                  #  The TLS/SSL I/O operation completed.
                  #  This result code is returned if and only if ret > 0
                  # so if we received it here complain...
                  warn "ERROR_NONE unexpected with invalid return value!"
                    if $trace;
                  $errname = "SSL_ERROR_NONE";
                };
                $sslerr == constant("ERROR_WANT_READ") && do {
                  # operation did not complete, call again later, so do not
                  # set errname and empty err_que since this is a known
                  # error that is expected but, we should continue to try
                  # writing the rest of our data with same io call and params.
                  warn "ERROR_WANT_READ (TLS/SSL Handshake, will continue)\n"
                    if $trace;
                  print_errs('SSL_write(want read)');
                  last SWITCH;
                };
                $sslerr == constant("ERROR_WANT_WRITE") && do {
                  # operation did not complete, call again later, so do not
                  # set errname and empty err_que since this is a known
                  # error that is expected but, we should continue to try
                  # writing the rest of our data with same io call and params.
                  warn "ERROR_WANT_WRITE (TLS/SSL Handshake, will continue)\n"
                    if $trace;
                  print_errs('SSL_write(want write)');
                  last SWITCH;
                };
                $sslerr == constant("ERROR_ZERO_RETURN") && do {
                  # valid protocol closure from other side, no longer able to
                  # write, since there is no longer a session...
                  warn "ERROR_ZERO_RETURN($wrote): TLS/SSLv3 Closure alert\n"
                    if $trace;
                  $errname = "SSL_ERROR_ZERO_RETURN";
                  last SWITCH;
                };
                $sslerr == constant("ERROR_SSL") && do {
                  # library/protocol error
                  warn "ERROR_SSL($wrote): Library/Protocol error occured\n"
                    if $trace;
                  $errname = "SSL_ERROR_SSL";
                  last SWITCH;
                };
                $sslerr == constant("ERROR_WANT_CONNECT") && do {
```

```
                # according to man page, should never happen on call to
                # SSL_write, so complain, but handle as known error type
                warn "ERROR_WANT_CONNECT: Unexpected error for SSL_write\n"
                  if $trace;
                $errname = "SSL_ERROR_WANT_CONNECT";
                last SWITCH;
              };
              $sslerr == constant("ERROR_WANT_ACCEPT") && do {
                # according to man page, should never happen on call to
                # SSL_write, so complain, but handle as known error type
                warn "ERROR_WANT_ACCEPT: Unexpected error for SSL_write\n"
                  if $trace;
                $errname = "SSL_ERROR_WANT_ACCEPT";
                last SWITCH;
              };
              $sslerr == constant("ERROR_WANT_X509_LOOKUP") && do {
                # operation did not complete: waiting on call back,
                # call again later, so do not set errname and empty err_que
                # since this is a known error that is expected but, we should
                # continue to try writing the rest of our data with same io
                # call parameter.
                warn "ERROR_WANT_X509_LOOKUP: (Cert Callback asked for in ".
                  "SSL_write will contine)\n" if $trace;
                print_errs('SSL_write(want x509');
                last SWITCH;
              };
              $sslerr == constant("ERROR_SYSCALL") && do {
                # some IO error occured. According to man page:
                # Check retval, ERR, fallback to errno
                if ($wrote==0) { # EOF
                  warn "ERROR_SYSCALL($wrote): EOF violates protocol.\n"
                    if $trace;
                  $errname = "SSL_ERROR_SYSCALL(EOF)";
                } else { # -1 underlying BIO error reported.
                  # check error que for details, don't set errname since we
                  # are directly appending to errs
                  my $chkerrs = print_errs('SSL_write (syscall)');
                  if ($chkerrs) {
                    warn "ERROR_SYSCALL($wrote): Have errors\n" if $trace;
                    $errs .= "ssl_write_all $$: 1 - ERROR_SYSCALL($wrote,".
                      "$sslerr,$errstr,$!)\n$chkerrs";
                  } else { # que was empty, use errno
                    warn "ERROR_SYSCALL($wrote): errno($!)\n" if $trace;
                    $errs .= "ssl_write_all $$: 1 - ERROR_SYSCALL($wrote,".
                      "$sslerr) : $!\n";
                  }
                }
                last SWITCH;
              };
              warn "Unhandled val $sslerr from SSL_get_error(SSL,$wrote)\n"
                if $trace;
              $errname = "SSL_ERROR_?($sslerr)";
            } # end of SWITCH block
            if ($errname) { # if we had an errname set add the error
              $errs .= "ssl_write_all $$: 1 - $errname($wrote,$sslerr,".
                "$errstr,$!)\n";
            }
          } # endif on have SSL_get_error val
        } # endif on $wrote defined
      } # endelse on $wrote > 0
      $vm = $trace>2 && $linux_debug ?
          (split ' ', `cat /proc/$$/stat`)[22] : 'vm_unknown';
```

```
        warn "  written so far $wrote:$written bytes (VM=$vm)\n" if $trace>2;
        # append remaining errors in que and report if errs exist
        $errs .= print_errs('SSL_write');
        return (wantarray ? (undef, $errs) : undef) if $errs;
    }
    return wantarray ? ($written, $errs) : $written;
}

sub tcp_write_all {
    my ($data_ref, $errs);
    if (ref $_[0]) {
        $data_ref = $_[0];
    } else {
        $data_ref = \$_[0];
    }
    my ($wrote, $written, $to_write) = (0,0, blength($$data_ref));
    my $vm = $trace>2 && $linux_debug ?
        (split ' ', `cat /proc/$$/stat`)[22] : 'vm_unknown';
    warn "  write_all VM at entry=$vm to_write=$to_write\n" if $trace>2;
    while ($to_write) {
        warn "partial `$$data_ref'\n" if $trace>3;
        $wrote = syswrite(SSLCAT_S, $$data_ref, $to_write, $written);
        if (defined $wrote && ($wrote > 0)) {  # write_partial can return -1
            $written += $wrote;
            $to_write -= $wrote;
        } elsif (!defined($wrote)) {
            warn "tcp_write_all: $!";
            return (wantarray ? (undef, "$!") : undef);
        }
        $vm = $trace>2 && $linux_debug ?
            (split ' ', `cat /proc/$$/stat`)[22] : 'vm_unknown';
        warn "  written so far $wrote:$written bytes (VM=$vm)\n" if $trace>2;
    }
    return wantarray ? ($written, '') : $written;
}

### from patch by Clinton Wong <clintdw@netcom.com>

# ssl_read_until($ssl [, $delimit [, $max_length]])
#  if $delimit missing, use $/ if it exists, otherwise use \n
#  read until delimiter reached, up to $max_length chars if defined

sub ssl_read_until ($;$$) {
    my ($ssl,$delim, $max_length) = @_;

    # guess the delim string if missing
    if ( ! defined $delim ) {
      if ( defined $/ && length $/  ) { $delim = $/ }
      else { $delim = "\n" }      # Note: \n,$/ value depends on the platform
    }
    my $len_delim = length $delim;

    my ($got);
    my $reply = '';

    # If we have OpenSSL 0.9.6a or later, we can use SSL_peek to
    # speed things up.
    # N.B. 0.9.6a has security problems, so the support for
    #      anything earlier than 0.9.6e will be dropped soon.
    if (&Net::SSLeay::OPENSSL_VERSION_NUMBER >= 0x0090601f) {
        $max_length = 2000000000 unless (defined $max_length);
        my ($pending, $peek_length, $found, $done);
```

```
        while (blength($reply) < $max_length and !$done) {
            #Block if necessary until we get some data
            $got = Net::SSLeay::peek($ssl,1);
            last if print_errs('SSL_peek');

            $pending = Net::SSLeay::pending($ssl) + blength($reply);
            $peek_length = ($pending > $max_length) ? $max_length : $pending;
            $peek_length -= blength($reply);
            $got = Net::SSLeay::peek($ssl, $peek_length);
            last if print_errs('SSL_peek');
            $peek_length = blength($got);

            #$found = index($got, $delim);  # Old and broken

            # the delimiter may be split across two gets, so we prepend
            # a little from the last get onto this one before we check
            # for a match
            my $match;
            if(blength($reply) >= blength($delim) - 1) {
                #if what we've read so far is greater or equal
                #in length of what we need to prepatch
                $match = substr $reply, blength($reply) - blength($delim) + 1;
            } else {
                $match = $reply;
            }

            $match .= $got;
            $found = index($match, $delim);

            if ($found > -1) {
                #$got = Net::SSLeay::ssl_read_all($ssl, $found+$len_delim);
                #read up to the end of the delimiter
                $got = Net::SSLeay::ssl_read_all($ssl,
                                    $found + $len_delim
                                    - ((blength($match)) - (blength($got))));
                $done = 1;
            } else {
                $got = Net::SSLeay::ssl_read_all($ssl, $peek_length);
                $done = 1 if ($peek_length == $max_length - blength($reply));
            }

            last if print_errs('SSL_read');
            debug_read(\$reply, \$got) if $trace>1;
            last if $got eq '';
            $reply .= $got;
        }
    } else {
        while (!defined $max_length || length $reply < $max_length) {
            $got = Net::SSLeay::ssl_read_all($ssl,1);  # one by one
            last if print_errs('SSL_read');
            debug_read(\$reply, \$got) if $trace>1;
            last if $got eq '';
            $reply .= $got;
            last if $len_delim
                && substr($reply, blength($reply)-$len_delim) eq $delim;
        }
    }
    return $reply;
}

sub tcp_read_until {
    my ($delim, $max_length) = @_;
```

```
    # guess the delim string if missing
    if ( ! defined $delim ) {
      if ( defined $/ && length $/  ) { $delim = $/ }
      else { $delim = "\n" }      # Note: \n,$/ value depends on the platform
    }
    my $len_delim = length $delim;

    my ($n,$got);
    my $reply = '';

    while (!defined $max_length || length $reply < $max_length) {
        $n = sysread(SSLCAT_S, $got, 1);  # one by one
        warn "tcp_read_until: $!" if !defined $n;
        debug_read(\$reply, \$got) if $trace>1;
        last if !$n;  # EOF
        $reply .= $got;
        last if $len_delim
            && substr($reply, blength($reply)-$len_delim) eq $delim;
    }
    return $reply;
}

# ssl_read_CRLF($ssl [, $max_length])
sub ssl_read_CRLF (;$$) { ssl_read_until($_[0], $CRLF, $_[1]) }
sub tcp_read_CRLF { tcp_read_until($CRLF, $_[0]) }

# ssl_write_CRLF($ssl, $message) writes $message and appends CRLF
sub ssl_write_CRLF ($$) {
  # the next line uses less memory but might use more network packets
  return ssl_write_all($_[0], $_[1]) + ssl_write_all($_[0], $CRLF);

  # the next few lines do the same thing at the expense of memory, with
  # the chance that it will use less packets, since CRLF is in the original
  # message and won't be sent separately.

  #my $data_ref;
  #if (ref $_[1]) { $data_ref = $_[1] }
  # else { $data_ref = \$_[1] }
  #my $message = $$data_ref . $CRLF;
  #return ssl_write_all($_[0], \$message);
}

sub tcp_write_CRLF {
  # the next line uses less memory but might use more network packets
  return tcp_write_all($_[0]) + tcp_write_all($CRLF);

  # the next few lines do the same thing at the expense of memory, with
  # the chance that it will use less packets, since CRLF is in the original
  # message and won't be sent separately.

  #my $data_ref;
  #if (ref $_[1]) { $data_ref = $_[1] }
  # else { $data_ref = \$_[1] }
  #my $message = $$data_ref . $CRLF;
  #return tcp_write_all($_[0], \$message);
}

### Quickly print out with whom we're talking

sub dump_peer_certificate ($) {
    my ($ssl) = @_;
```

```
    my $cert = get_peer_certificate($ssl);
    return if print_errs('get_peer_certificate');
    print "no cert defined\n" if !defined($cert);
    # Cipher=NONE with empty cert fix
    if (!defined($cert) || ($cert == 0)) {
        warn "cert = `$cert'\n" if $trace;
        return "Subject Name: undefined\nIssuer  Name: undefined\n";
    } else {
        my $x = 'Subject Name: '
            . X509_NAME_oneline(X509_get_subject_name($cert)) . "\n"
                . 'Issuer  Name: '
                    . X509_NAME_oneline(X509_get_issuer_name($cert))  . "\n";
        Net::SSLeay::X509_free($cert);
        return $x;
    }
}

### Arrange some randomness for eay PRNG

sub randomize (;$$$) {
    my ($rn_seed_file, $seed, $egd_path) = @_;
    my $rnsf = defined($rn_seed_file) && -r $rn_seed_file;

        $egd_path = '';
    $egd_path = $ENV{'EGD_PATH'} if $ENV{'EGD_PATH'};

    RAND_seed(rand() + $$);  # Stir it with time and pid

    unless ($rnsf || -r $Net::SSLeay::random_device || $seed || -S $egd_path) {
        my $poll_retval = Net::SSLeay::RAND_poll();
        warn "Random number generator not seeded!!!" if $trace && !$poll_retval;
    }

    RAND_load_file($rn_seed_file, -s _) if $rnsf;
    RAND_seed($seed) if $seed;
    RAND_seed($ENV{RND_SEED}) if $ENV{RND_SEED};
    RAND_load_file($Net::SSLeay::random_device, $Net::SSLeay::how_random/8)
        if -r $Net::SSLeay::random_device;
}

sub new_x_ctx {
    if ($ssl_version == 2)  {
        unless (exists &Net::SSLeay::CTX_v2_new) {
            warn "ssl_version has been set to 2, but this version of OpenSSL has been
compiled without SSLv2 support";
            return undef;
        }
        $ctx = CTX_v2_new();
    }
    elsif ($ssl_version == 3)  { $ctx = CTX_v3_new(); }
    elsif ($ssl_version == 10) { $ctx = CTX_tlsv1_new(); }
    elsif ($ssl_version == 11) {
        unless (exists &Net::SSLeay::CTX_tlsv1_1_new) {
            warn "ssl_version has been set to 11, but this version of OpenSSL has been
compiled without TLSv1.1 support";
            return undef;
        }
        $ctx = CTX_tlsv1_1_new;
    }
    elsif ($ssl_version == 12) {
        unless (exists &Net::SSLeay::CTX_tlsv1_2_new) {
            warn "ssl_version has been set to 12, but this version of OpenSSL has been
```

```
compiled without TLSv1.2 support";
            return undef;
        }
        $ctx = CTX_tlsv1_2_new;
    }
    elsif ($ssl_version == 13) {
        unless (eval { Net::SSLeay::TLS1_3_VERSION(); } ) {
            warn "ssl_version has been set to 13, but this version of OpenSSL has been
compiled without TLSv1.3 support";
            return undef;
        }
        $ctx = CTX_new();
        unless(Net::SSLeay::CTX_set_min_proto_version($ctx,
Net::SSLeay::TLS1_3_VERSION())) {
            warn "CTX_set_min_proto failed for TLSv1.3";
            return undef;
        }
        unless(Net::SSLeay::CTX_set_max_proto_version($ctx,
Net::SSLeay::TLS1_3_VERSION())) {
            warn "CTX_set_max_proto failed for TLSv1.3";
            return undef;
        }
    }
    else                        { $ctx = CTX_new(); }
    return $ctx;
}

###
### Standard initialisation. Initialise the ssl library in the usual way
###  at most once. Override this if you need differnet initialisation
###  SSLeay_add_ssl_algorithms is also protected against multiple runs in SSLeay.xs
###  and is also mutex protected in threading perls
###

my $library_initialised;
sub initialize
{
    if (!$library_initialised)
    {
        load_error_strings();         # Some bloat, but I'm after ease of use
        SSLeay_add_ssl_algorithms();  # and debuggability.
        randomize();
        $library_initialised++;
    }
}

###
### Basic request - response primitive (don't use for https)
###

sub sslcat { # address, port, message, $crt, $key --> reply / (reply,errs,cert)
    my ($dest_serv, $port, $out_message, $crt_path, $key_path) = @_;
    my ($ctx, $ssl, $got, $errs, $written);

    ($got, $errs) = open_proxy_tcp_connection($dest_serv, $port);
    return (wantarray ? (undef, $errs) : undef) unless $got;

    ### Do SSL negotiation stuff

    warn "Creating SSL $ssl_version context...\n" if $trace>2;
    initialize(); # Will init at most once
```

```
    $ctx = new_x_ctx();
    goto cleanup2 if $errs = print_errs('CTX_new') or !$ctx;

    CTX_set_options($ctx, &OP_ALL);
    goto cleanup2 if $errs = print_errs('CTX_set_options');

    warn "Cert `$crt_path' given without key" if $crt_path && !$key_path;
    set_cert_and_key($ctx, $crt_path, $key_path) if $crt_path;

    warn "Creating SSL connection (context was '$ctx')...\n" if $trace>2;
    $ssl = new($ctx);
    goto cleanup if $errs = print_errs('SSL_new') or !$ssl;

    warn "Setting fd (ctx $ctx, con $ssl)...\n" if $trace>2;
    set_fd($ssl, fileno(SSLCAT_S));
    goto cleanup if $errs = print_errs('set_fd');

    warn "Entering SSL negotiation phase...\n" if $trace>2;

    if ($trace>2) {
        my $i = 0;
        my $p = '';
        my $cipher_list = 'Cipher list: ';
        $p=Net::SSLeay::get_cipher_list($ssl,$i);
        $cipher_list .= $p if $p;
        do {
            $i++;
            $cipher_list .= ', ' . $p if $p;
            $p=Net::SSLeay::get_cipher_list($ssl,$i);
        } while $p;
        $cipher_list .= '\n';
        warn $cipher_list;
    }

    $got = Net::SSLeay::connect($ssl);
    warn "SSLeay connect returned $got\n" if $trace>2;
    goto cleanup if $errs = print_errs('SSL_connect');

    my $server_cert = get_peer_certificate($ssl);
    print_errs('get_peer_certificate');
    if ($trace>1) {
        warn "Cipher `" . get_cipher($ssl) . "'\n";
        print_errs('get_ciper');
        warn dump_peer_certificate($ssl);
    }

    ### Connected. Exchange some data (doing repeated tries if necessary).

    warn "sslcat $$: sending " . blength($out_message) . " bytes...\n"
        if $trace==3;
    warn "sslcat $$: sending `$out_message' (" . blength($out_message)
        . " bytes)...\n" if $trace>3;
    ($written, $errs) = ssl_write_all($ssl, $out_message);
    goto cleanup unless $written;

    sleep $slowly if $slowly;  # Closing too soon can abort broken servers
    CORE::shutdown SSLCAT_S, 1;  # Half close --> No more output, send EOF to server

    warn "waiting for reply...\n" if $trace>2;
    ($got, $errs) = ssl_read_all($ssl);
    warn "Got " . blength($got) . " bytes.\n" if $trace==3;
    warn "Got `$got' (" . blength($got) . " bytes)\n" if $trace>3;
```

```
cleanup:
    free ($ssl);
    $errs .= print_errs('SSL_free');
cleanup2:
    CTX_free ($ctx);
    $errs .= print_errs('CTX_free');
    close SSLCAT_S;
    return wantarray ? ($got, $errs, $server_cert) : $got;
}

sub tcpcat { # address, port, message, $crt, $key --> reply / (reply,errs,cert)
    my ($dest_serv, $port, $out_message) = @_;
    my ($got, $errs, $written);

    ($got, $errs) = open_proxy_tcp_connection($dest_serv, $port);
    return (wantarray ? (undef, $errs) : undef) unless $got;

    ### Connected. Exchange some data (doing repeated tries if necessary).

    warn "tcpcat $$: sending " . blength($out_message) . " bytes...\n"
        if $trace==3;
    warn "tcpcat $$: sending `$out_message' (" . blength($out_message)
        . " bytes)...\n" if $trace>3;
    ($written, $errs) = tcp_write_all($out_message);
    goto cleanup unless $written;

    sleep $slowly if $slowly;  # Closing too soon can abort broken servers
    CORE::shutdown SSLCAT_S, 1;  # Half close --> No more output, send EOF to server

    warn "waiting for reply...\n" if $trace>2;
    ($got, $errs) = tcp_read_all();
    warn "Got " . blength($got) . " bytes.\n" if $trace==3;
    warn "Got `$got' (" . blength($got) . " bytes)\n" if $trace>3;

cleanup:
    close SSLCAT_S;
    return wantarray ? ($got, $errs) : $got;
}

sub tcpxcat {
    my ($usessl, $site, $port, $req, $crt_path, $key_path) = @_;
    if ($usessl) {
        return sslcat($site, $port, $req, $crt_path, $key_path);
    } else {
        return tcpcat($site, $port, $req);
    }
}

###
### Basic request - response primitive, this is different from sslcat
###                 because this does not shutdown the connection.
###

sub https_cat { # address, port, message --> returns reply / (reply,errs,cert)
    my ($dest_serv, $port, $out_message, $crt_path, $key_path) = @_;
    my ($ctx, $ssl, $got, $errs, $written);

    ($got, $errs) = open_proxy_tcp_connection($dest_serv, $port);
    return (wantarray ? (undef, $errs) : undef) unless $got;

    ### Do SSL negotiation stuff
```

```
    warn "Creating SSL $ssl_version context...\n" if $trace>2;
    initialize();

    $ctx = new_x_ctx();
    goto cleanup2 if $errs = print_errs('CTX_new') or !$ctx;

    CTX_set_options($ctx, &OP_ALL);
    goto cleanup2 if $errs = print_errs('CTX_set_options');

    warn "Cert `$crt_path' given without key" if $crt_path && !$key_path;
    set_cert_and_key($ctx, $crt_path, $key_path) if $crt_path;

    warn "Creating SSL connection (context was '$ctx')...\n" if $trace>2;
    $ssl = new($ctx);
    goto cleanup if $errs = print_errs('SSL_new') or !$ssl;

    warn "Setting fd (ctx $ctx, con $ssl)...\n" if $trace>2;
    set_fd($ssl, fileno(SSLCAT_S));
    goto cleanup if $errs = print_errs('set_fd');

    warn "Entering SSL negotiation phase...\n" if $trace>2;

    if ($trace>2) {
        my $i = 0;
        my $p = '';
        my $cipher_list = 'Cipher list: ';
        $p=Net::SSLeay::get_cipher_list($ssl,$i);
        $cipher_list .= $p if $p;
        do {
            $i++;
            $cipher_list .= ', ' . $p if $p;
            $p=Net::SSLeay::get_cipher_list($ssl,$i);
        } while $p;
        $cipher_list .= '\n';
        warn $cipher_list;
    }

    $got = Net::SSLeay::connect($ssl);
    warn "SSLeay connect failed" if $trace>2 && $got==0;
    goto cleanup if $errs = print_errs('SSL_connect');

    my $server_cert = get_peer_certificate($ssl);
    print_errs('get_peer_certificate');
    if ($trace>1) {
        warn "Cipher `" . get_cipher($ssl) . "'\n";
        print_errs('get_ciper');
        warn dump_peer_certificate($ssl);
    }

    ### Connected. Exchange some data (doing repeated tries if necessary).

    warn "https_cat $$: sending " . blength($out_message) . " bytes...\n"
        if $trace==3;
    warn "https_cat $$: sending `$out_message' (" . blength($out_message)
        . " bytes)...\n" if $trace>3;
    ($written, $errs) = ssl_write_all($ssl, $out_message);
    goto cleanup unless $written;

    warn "waiting for reply...\n" if $trace>2;
    ($got, $errs) = ssl_read_all($ssl);
    warn "Got " . blength($got) . " bytes.\n" if $trace==3;
```

```
    warn "Got `$got' (" . blength($got) . " bytes)\n" if $trace>3;

cleanup:
    free ($ssl);
    $errs .= print_errs('SSL_free');
cleanup2:
    CTX_free ($ctx);
    $errs .= print_errs('CTX_free');
    close SSLCAT_S;
    return wantarray ? ($got, $errs, $server_cert) : $got;
}

sub http_cat { # address, port, message --> returns reply / (reply,errs,cert)
    my ($dest_serv, $port, $out_message) = @_;
    my ($got, $errs, $written);

    ($got, $errs) = open_proxy_tcp_connection($dest_serv, $port);
    return (wantarray ? (undef, $errs) : undef) unless $got;

    ### Connected. Exchange some data (doing repeated tries if necessary).

    warn "http_cat $$: sending " . blength($out_message) . " bytes...\n"
        if $trace==3;
    warn "http_cat $$: sending `$out_message' (" . blength($out_message)
        . " bytes)...\n" if $trace>3;
    ($written, $errs) = tcp_write_all($out_message);
    goto cleanup unless $written;

    warn "waiting for reply...\n" if $trace>2;
    ($got, $errs) = tcp_read_all();
    warn "Got " . blength($got) . " bytes.\n" if $trace==3;
    warn "Got `$got' (" . blength($got) . " bytes)\n" if $trace>3;

cleanup:
    close SSLCAT_S;
    return wantarray ? ($got, $errs) : $got;
}

sub httpx_cat {
    my ($usessl, $site, $port, $req, $crt_path, $key_path) = @_;
    warn "httpx_cat: usessl=$usessl ($site:$port)" if $trace;
    if ($usessl) {
        return https_cat($site, $port, $req, $crt_path, $key_path);
    } else {
        return http_cat($site, $port, $req);
    }
}

###
### Easy set up of private key and certificate
###

sub set_cert_and_key ($$$) {
    my ($ctx, $cert_path, $key_path) = @_;
    my $errs = '';
    # Following will ask password unless private key is not encrypted
    CTX_use_PrivateKey_file( $ctx, $key_path, &FILETYPE_PEM ) == 1
        or $errs .= print_errs("private key `$key_path' ($!)");
    CTX_use_certificate_file ($ctx, $cert_path, &FILETYPE_PEM) == 1
        or $errs .= print_errs("certificate `$cert_path' ($!)");
    return wantarray ? (undef, $errs) : ($errs eq '');
}
```

```
### Old deprecated API

sub set_server_cert_and_key ($$$) { &set_cert_and_key }

### Set up to use web proxy

sub set_proxy ($$;**) {
    ($proxyhost, $proxyport, $proxyuser, $proxypass) = @_;
    require MIME::Base64 if $proxyuser;
    $proxyauth = $proxyuser
         ? $CRLF . 'Proxy-authorization: Basic '
         . MIME::Base64::encode("$proxyuser:$proxypass", '')
         : '';
}

###
### Easy https manipulation routines
###

sub make_form {
    my (@fields) = @_;
    my $form;
    while (@fields) {
        my ($name, $data) = (shift(@fields), shift(@fields));
        $data =~ s/([^\w\-.\@\$ ])/sprintf("%%%2.2x",ord($1))/gse;
        $data =~ tr[ ][+];
        $form .= "$name=$data&";
    }
    chop $form;
    return $form;
}

sub make_headers {
    my (@headers) = @_;
    my $headers;
    while (@headers) {
        my $header = shift(@headers);
        my $value = shift(@headers);
        $header =~ s/:$//;
        $value =~ s/\x0d?\x0a$//; # because we add it soon, see below
        $headers .= "$header: $value$CRLF";
    }
    return $headers;
}

sub do_httpx3 {
    my ($method, $usessl, $site, $port, $path, $headers,
        $content, $mime_type, $crt_path, $key_path) = @_;
    my ($response, $page, $h,$v);

    my $len = blength($content);
    if ($len) {
        $mime_type = "application/x-www-form-urlencoded" unless $mime_type;
        $content = "Content-Type: $mime_type$CRLF"
             . "Content-Length: $len$CRLF$CRLF$content";
    } else {
        $content = "$CRLF$CRLF";
    }
    my $req = "$method $path HTTP/1.0$CRLF";
    unless (defined $headers && $headers =~ /^Host:/m) {
        $req .= "Host: $site";
```

```
        unless (($port == 80 && !$usessl) || ($port == 443 && $usessl)) {
            $req .= ":$port";
        }
        $req .= $CRLF;
        }
    $req .= (defined $headers ? $headers : '') . "Accept: */*$CRLF$content";

    warn "do_httpx3($method,$usessl,$site:$port)" if $trace;
    my ($http, $errs, $server_cert)
        = httpx_cat($usessl, $site, $port, $req, $crt_path, $key_path);
    return (undef, "HTTP/1.0 900 NET OR SSL ERROR$CRLF$CRLF$errs") if $errs;

    $http = '' if !defined $http;
    ($headers, $page) = split /\s?\n\s?\n/, $http, 2;
    warn "headers >$headers< page >>$page<< http >>>$http<<<" if $trace>1;
    ($response, $headers) = split /\s?\n/, $headers, 2;
    return ($page, $response, $headers, $server_cert);
}

sub do_https3 { splice(@_,1,0) = 1; do_httpx3; }  # Legacy undocumented

### do_https2() is a legacy version in the sense that it is unable
### to return all instances of duplicate headers.

sub do_httpx2 {
    my ($page, $response, $headers, $server_cert) = &do_httpx3;
    X509_free($server_cert) if defined $server_cert;
    return ($page, $response, defined $headers ?
            map( { ($h,$v)=/^(\S+)\:\s*(.*)$/; (uc($h),$v); }
                split(/\s?\n/, $headers)
                ) : ()
            );
}

sub do_https2 { splice(@_,1,0) = 1; do_httpx2; }  # Legacy undocumented

### Returns headers as a hash where multiple instances of same header
### are handled correctly.

sub do_httpx4 {
    my ($page, $response, $headers, $server_cert) = &do_httpx3;
    my %hr = ();
    for my $hh (split /\s?\n/, $headers) {
        my ($h,$v) = ($hh =~ /^(\S+)\:\s*(.*)$/);
        push @{$hr{uc($h)}}, $v;
    }
    return ($page, $response, \%hr, $server_cert);
}

sub do_https4 { splice(@_,1,0) = 1; do_httpx4; }  # Legacy undocumented

# https

sub get_https  { do_httpx2(GET  => 1, @_) }
sub post_https { do_httpx2(POST => 1, @_) }
sub put_https  { do_httpx2(PUT  => 1, @_) }
sub head_https { do_httpx2(HEAD => 1, @_) }

sub get_https3  { do_httpx3(GET  => 1, @_) }
sub post_https3 { do_httpx3(POST => 1, @_) }
sub put_https3  { do_httpx3(PUT  => 1, @_) }
sub head_https3 { do_httpx3(HEAD => 1, @_) }
```

```
sub get_https4  { do_httpx4(GET  => 1, @_) }
sub post_https4 { do_httpx4(POST => 1, @_) }
sub put_https4  { do_httpx4(PUT  => 1, @_) }
sub head_https4 { do_httpx4(HEAD => 1, @_) }

# http

sub get_http  { do_httpx2(GET  => 0, @_) }
sub post_http { do_httpx2(POST => 0, @_) }
sub put_http  { do_httpx2(PUT  => 0, @_) }
sub head_http { do_httpx2(HEAD => 0, @_) }

sub get_http3  { do_httpx3(GET  => 0, @_) }
sub post_http3 { do_httpx3(POST => 0, @_) }
sub put_http3  { do_httpx3(PUT  => 0, @_) }
sub head_http3 { do_httpx3(HEAD => 0, @_) }

sub get_http4  { do_httpx4(GET  => 0, @_) }
sub post_http4 { do_httpx4(POST => 0, @_) }
sub put_http4  { do_httpx4(PUT  => 0, @_) }
sub head_http4 { do_httpx4(HEAD => 0, @_) }

# Either https or http

sub get_httpx  { do_httpx2(GET  => @_) }
sub post_httpx { do_httpx2(POST => @_) }
sub put_httpx  { do_httpx2(PUT  => @_) }
sub head_httpx { do_httpx2(HEAD => @_) }

sub get_httpx3  { do_httpx3(GET  => @_) }
sub post_httpx3 { do_httpx3(POST => @_) }
sub put_httpx3  { do_httpx3(PUT  => @_) }
sub head_httpx3 { do_httpx3(HEAD => @_) }

sub get_httpx4  { do_httpx4(GET  => @_) }
sub post_httpx4 { do_httpx4(POST => @_) }
sub put_httpx4  { do_httpx4(PUT  => @_) }
sub head_httpx4 { do_httpx4(HEAD => @_) }

### Legacy, don't use
# ($page, $respone_or_err, %headers) = do_https(...);

sub do_https {
    my ($site, $port, $path, $method, $headers,
        $content, $mime_type, $crt_path, $key_path) = @_;

    do_https2($method, $site, $port, $path, $headers,
              $content, $mime_type, $crt_path, $key_path);
}

1;
__END__

-sh-3.2$
```

GOVEGAN
GOFREESOFTWARE