NAME

stunnel - TLS offloading and load-balancing proxy

SYNOPSIS

Unix:

stunnel [FILE] | -fd N | -help | -version | -sockets | -options

WIN32:

stunnel [[-install | -uninstall | -start | -stop |
-reload | -reopen | -exit] [-quiet] [FILE]] |
-help | -version | -sockets | -options

DESCRIPTION

The **stunnel** program is designed to work as *TLS* encryption wrapper between remote clients and local (*inetd*-startable) or remote servers. The concept is that having non-TLS aware daemons running on your system you can easily set them up to communicate with clients over secure *TLS* channels.

stunnel can be used to add *TLS* functionality to commonly used *Inetd* daemons like POP-2, POP-3, and IMAP servers, to standalone daemons like NNTP, SMTP and HTTP, and in tunneling PPP over network sockets without changes to the source code.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

OPTIONS

FILE

Use specified configuration file

-fd N (Unix only)

Read the config file from specified file descriptor

-help

Print **stunnel** help menu

-version

Print **stunnel** version and compile time defaults

-sockets

Print default socket options

-options

Print supported TLS options

-install (Windows NT and later only)
Install NT Service

-uninstall (Windows NT and later only)
Uninstall NT Service

-start (Windows NT and later only)
Start NT Service

-stop (Windows NT and later only)
Stop NT Service

-reload (Windows NT and later only)
Reload the configuration file of the running NT Service

-reopen (Windows NT and later only)
Reopen the log file of the running NT Service

-exit (Win32 only)
Exit an already started stunnel

-quiet (Win32 only)
Don’t display any message boxes

CONFIGURATION FILE

Each line of the configuration file can be either:

- An empty line (ignored).
- A comment starting with ’;’ (ignored).
- An ’option_name = option_value’ pair.
- ’[service_name]’ indicating a start of a service definition.

An address parameter of an option may be either:

- A port number.

- A colon-separated pair of IP address (either IPv4, IPv6, or domain name) and port number.

- A Unix socket path (Unix only).

GLOBAL OPTIONS

chroot = DIRECTORY (Unix only)

directory to chroot **stunnel** process

chroot keeps **stunnel** in a chrooted jail. *CApath*, *CRLpath*, *pid* and *exec* are located inside the jail and the patches have to be relative to the directory specified with **chroot**.

Several functions of the operating system also need their files to be located within the chroot jail, e.g.:

- Delayed resolver typically needs /etc/nsswitch.conf and /etc/resolv.conf.

- Local time in log files needs /etc/timezone.

- Some other functions may need devices, e.g. /dev/zero or /dev/null.

compression = deflate | zlib

select data compression algorithm

default: no compression

Deflate is the standard compression method as described in RFC 1951.

debug = [FACILITY.]LEVEL

debugging level

Level is one of the syslog level names or numbers emerg (0), alert (1), crit (2), err (3), warning (4), notice (5), info (6), or debug (7). All logs for the specified level and all levels numerically less than it will be shown.

The *debug = debug* (or the equivalent <debug = 7>) level produces for the most verbose log output. This logging level is only meant to be understood by stunnel developers, and not by users. Please either use the debug level when requested to do so by an stunnel developer, or when you intend to get confused.

The default logging level is notice (5).

The syslog 'daemon' facility will be used unless a facility name is supplied. (Facilities are not supported on Win32.)

Case is ignored for both facilities and levels.

EGD = EGD_PATH (Unix only)
path to Entropy Gathering Daemon socket

Entropy Gathering Daemon socket to use to feed the **OpenSSL** random number generator.

engine = auto | ENGINE_ID
select hardware or software cryptographic engine

default: software-only cryptography

See Examples section for an engine configuration to use the certificate and the corresponding private key from a cryptographic device.

engineCtrl = COMMAND[:PARAMETER]
control hardware engine

engineDefault = TASK_LIST
set OpenSSL tasks delegated to the current engine

The parameter specifies a comma-separated list of task to be delegated to the current engine.

The following tasks may be available, if supported by the engine: ALL, RSA, DSA, ECDH, ECDSA, DH, RAND, CIPHERS, DIGESTS, PKEY, PKEY_CRYPTO, PKEY_ASN1.

fips = yes | no
enable or disable FIPS 140-2 mode.

This option allows you to disable entering FIPS mode if **stunnel** was compiled with FIPS 140-2 support.

default: no (since version 5.00)

foreground = yes | quiet | no (Unix only)
foreground mode

Stay in foreground (don’t fork).

With the *yes* parameter it also logs to stderr in addition to the destinations specified with *syslog* and *output*.

default: background in daemon mode

iconActive = ICON_FILE (GUI only)
GUI icon to be displayed when there are established connections

On Windows platform the parameter should be an .ico file containing a 16x16 pixel image.

iconError = ICON_FILE (GUI only)
GUI icon to be displayed when no valid configuration is loaded

On Windows platform the parameter should be an .ico file containing a 16x16 pixel image.

iconIdle = ICON_FILE (GUI only)
GUI icon to be displayed when there are no established connections

On Windows platform the parameter should be an .ico file containing a 16x16 pixel image.

log = append | overwrite
log file handling

This option allows you to choose whether the log file (specified with the *output* option) is appended or overwritten when opened or re-opened.

default: append

output = FILE
append log messages to a file

/dev/stdout device can be used to send log messages to the standard output (for example to log them with daemontools splogger).

pid = FILE (Unix only)
pid file location

If the argument is empty, then no pid file will be created.

pid path is relative to the *chroot* directory if specified.

RNDbytes = BYTES

bytes to read from random seed files

RNDfile = FILE

path to file with random seed data

The OpenSSL library will use data from this file first to seed the random number generator.

RNDoverwrite = yes | no

overwrite the random seed files with new random data

default: yes

service = SERVICE (Unix only)

stunnel service name

The specified service name is used for syslog and as the *inetd* mode service name for TCP Wrappers. While this option can technically be specified in the service sections, it is only useful in global options.

default: stunnel

syslog = yes | no (Unix only)

enable logging via syslog

default: yes

taskbar = yes | no (WIN32 only)

enable the taskbar icon

default: yes

SERVICE-LEVEL OPTIONS

Each configuration section begins with a service name in square brackets. The service name is used for libwrap (TCP Wrappers) access control and lets you distinguish **stunnel** services in your log files.

Note that if you wish to run **stunnel** in *inetd* mode (where it is provided a network socket by a server such as *inetd*, *xinetd*, or *tcpserver*) then you should read the section entitled *INETD MODE* below.

accept = [HOST:]PORT
accept connections on specified address

If no host specified, defaults to all IPv4 addresses for the local host.

To listen on all IPv6 addresses use:

```
accept = :::PORT
```

CAengine = ENGINE-SPECIFIC_CA_CERTIFICATE_IDENTIFIER
load a trusted CA certificate from an engine

The loaded CA certificates will be used with the *verifyChain* and *verifyPeer* options.

Multiple *CAengine* options are allowed in a single service section.

Currently supported engines: pkcs11, cng.

CApath = CA_DIRECTORY
load trusted CA certificates from a directory

The loaded CA certificates will be used with the *verifyChain* and *verifyPeer* options. Note that the certificates in this directory should be named XXXXXXXX.0 where XXXXXXXX is the hash value of the DER encoded subject of the cert.

The hash algorithm has been changed in **OpenSSL 1.0.0**. It is required to c_rehash the directory on upgrade from **OpenSSL 0.x.x** to **OpenSSL 1.x.x** or later.

CApath path is relative to the *chroot* directory if specified.

CAfile = CA_FILE
load trusted CA certificates from a file

The loaded CA certificates will be used with the *verifyChain* and *verifyPeer* options.

cert = CERT_FILE
certificate chain file name

The parameter specifies the file containing certificates used by **stunnel** to authenticate itself against the remote client or server. The file should contain the whole certificate chain starting from the actual server/client certificate, and ending with the self-signed root CA certificate. The file must be either in PEM or P12 format.

A certificate chain is required in server mode, and optional in client mode.

This parameter is also used as the certificate identifier when a hardware engine is enabled.

checkEmail = EMAIL
verify the email address of the end-entity (leaf) peer certificate subject

Certificates are accepted if no subject checks were specified, or the email address of the end-entity (leaf) peer certificate matches any of the email addresses specified with *checkEmail*.

Multiple *checkEmail* options are allowed in a single service section.

This option requires OpenSSL 1.0.2 or later.

checkHost = HOST
verify the host of the end-entity (leaf) peer certificate subject

Certificates are accepted if no subject checks were specified, or the host name of the end-entity (leaf) peer certificate matches any of the hosts specified with *checkHost*.

Multiple *checkHost* options are allowed in a single service section.

This option requires OpenSSL 1.0.2 or later.

checkIP = IP
verify the IP address of the end-entity (leaf) peer certificate subject

Certificates are accepted if no subject checks were specified, or the IP address of the end-entity (leaf) peer certificate matches any of the IP addresses specified with *checkIP*.

Multiple *checkIP* options are allowed in a single service section.

This option requires OpenSSL 1.0.2 or later.

ciphers = CIPHER_LIST
select permitted TLS ciphers (TLSv1.2 and below)

This option does not impact TLSv1.3 ciphersuites.

A colon-delimited list of the ciphers to allow in the TLS connection, for example DES-CBC3-SHA:IDEA-CBC-MD5.

ciphersuites = CIPHERSUITES_LIST
select permitted TLSv1.3 ciphersuites

A colon-delimited list of TLSv1.3 ciphersuites names in order of preference.

This option requires OpenSSL 1.1.1 or later.

default:
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_256_GCM_SHA384:TLS_AES_128_GCM_SHA256

client = yes | no
client mode (remote service uses TLS)

default: no (server mode)

config = COMMAND[:PARAMETER]
OpenSSL configuration command

The **OpenSSL** configuration command is executed with the specified parameter. This allows any configuration commands to be invoked from the stunnel configuration file. Supported commands are described on the ***SSL_CONF_cmd****(3ssl)* manual page.

Several *config* lines can be used to specify multiple configuration commands.

Use *curves* option instead of enabling *config = Curves:list_curves* to support elliptic curves.

This option requires OpenSSL 1.0.2 or later.

connect = [HOST:]PORT
connect to a remote address

If no host is specified, the host defaults to localhost.

Multiple *connect* options are allowed in a single service section. If host resolves to multiple addresses and/or if multiple *connect* options are specified, then the remote address is chosen using a round-robin algorithm.

CRLpath = DIRECTORY

Certificate Revocation Lists directory

This is the directory in which **stunnel** will look for CRLs when using the *verifyChain* and *verifyPeer* options. Note that the CRLs in this directory should be named XXXXXXXX.r0 where XXXXXXXX is the hash value of the CRL.

The hash algorithm has been changed in **OpenSSL 1.0.0**. It is required to c_rehash the directory on upgrade from **OpenSSL 0.x.x** to **OpenSSL 1.x.x**.

CRLpath path is relative to the *chroot* directory if specified.

CRLfile = CRL_FILE

Certificate Revocation Lists file

This file contains multiple CRLs, used with the *verifyChain* and *verifyPeer* options.

curves = list

ECDH curves separated with ':'

Only a single curve name is allowed for OpenSSL older than 1.1.1.

To get a list of supported curves use:

```
openssl ecparam -list_curves
```

default:

X25519:P-256:X448:P-521:P-384 (OpenSSL 1.1.1 or later)

prime256v1 (OpenSSL older than 1.1.1)

logId = TYPE

connection identifier type

This identifier allows you to distinguish log entries generated for each of the connections.

Currently supported types:

sequential

The numeric sequential identifier is only unique within a single instance of **stunnel**, but very compact. It is most useful for manual log analysis.

unique

This alphanumeric identifier is globally unique, but longer than the sequential number. It is most useful for automated log analysis.

thread

The operating system thread identifier is neither unique (even within a single instance of **stunnel**) nor short. It is most useful for debugging software or configuration issues.

process

The operating system process identifier (PID) may be useful in the inetd mode.

default: sequential

debug = LEVEL

debugging level

Level is a one of the syslog level names or numbers emerg (0), alert (1), crit (2), err (3), warning (4), notice (5), info (6), or debug (7). All logs for the specified level and all levels numerically less than it will be shown. The default is notice (5).

While the *debug = debug* or *debug = 7* level generates the most verbose output, it is only intended to be used by stunnel developers. Please only use this value if you are a developer, or you intend to send your logs to our technical support. Otherwise, the generated logs **will** be confusing.

delay = yes | no

delay DNS lookup for the *connect* option

This option is useful for dynamic DNS, or when DNS is not available during **stunnel** startup (road warrior VPN, dial-up configurations).

Delayed resolver mode is automatically engaged when stunnel fails to resolve on startup any of the *connect* targets for a service.

Delayed resolver inflicts *failover = prio*.

default: no

engineId = ENGINE_ID

select engine ID for the service

engineNum = ENGINE_NUMBER

select engine number for the service

The engines are numbered starting from 1.

exec = EXECUTABLE_PATH

execute a local inetd-type program

exec path is relative to the *chroot* directory if specified.

The following environmental variables are set on Unix platforms: REMOTE_HOST, REMOTE_PORT, SSL_CLIENT_DN, SSL_CLIENT_I_DN.

execArgs = $0 $1 $2 ...

arguments for *exec* including the program name ($0)

Quoting is currently not supported. Arguments are separated with an arbitrary amount of whitespace.

failover = rr | prio

Failover strategy for multiple "connect" targets.

rr round robin - fair load distribution

prio priority - use the order specified in config file

default: prio

ident = USERNAME

use IDENT (RFC 1413) username checking

include = DIRECTORY

include all configuration file parts located in DIRECTORY

The files are included in the ascending alphabetical order of their names. The recommended filename convention is for global options:

```
00-global.conf
```

for local service-level options:

```
01-service.conf

02-service.conf
```

key = KEY_FILE

private key for the certificate specified with *cert* option

A private key is needed to authenticate the certificate owner. Since this file should be kept secret it should only be readable by its owner. On Unix systems you can use the following command:

```
chmod 600 keyfile
```

This parameter is also used as the private key identifier when a hardware engine is enabled.

default: the value of the *cert* option

libwrap = yes | no

Enable or disable the use of /etc/hosts.allow and /etc/hosts.deny.

default: no (since version 5.00)

local = HOST

By default, the IP address of the outgoing interface is used as the source for remote connections. Use this option to bind a static local IP address instead.

OCSP = URL

select OCSP responder for the end-entity (leaf) peer certificate verification

OCSPaia = yes | no

validate certificates with their AIA OCSP responders

This option enables *stunnel* to validate certificates with the list of OCSP responder URLs retrieved from their AIA (Authority Information Access) extension.

OCSPflag = OCSP_FLAG

specify OCSP responder flag

Several *OCSPflag* can be used to specify multiple flags.

currently supported flags: NOCERTS, NOINTERN, NOSIGS, NOCHAIN, NOVERIFY, NOEXPLICIT, NOCASIGN, NODELEGATED, NOCHECKS, TRUSTOTHER, RESPID_KEY, NOTIME

OCSPnonce = yes | no

send and verify the OCSP nonce extension

This option protects the OCSP protocol against replay attacks. Due to its computational overhead, the nonce extension is usually only supported on internal (e.g. corporate) responders, and not on public OCSP responders.

OCSPrequire = yes | no

require a conclusive OCSP response

Disable this option to allow a connection even though no conclusive OCSP response was retrieved from stapling and a direct request to the OCSP responder.

default: yes

options = SSL_OPTIONS

OpenSSL library options

The parameter is the **OpenSSL** option name as described in the ***SSL_CTX_set_options****(3ssl)* manual, but without *SSL_OP_* prefix. *stunnel -options* lists the options found to be allowed in the current combination of *stunnel* and the *OpenSSL* library used to build it.

Several *option* lines can be used to specify multiple options. An option name can be prepended with a dash ("-") to disable the option.

For example, for compatibility with the erroneous Eudora TLS implementation, the following option can be used:

```
options = DONT_INSERT_EMPTY_FRAGMENTS
```

default:

```
options = NO_SSLv2
options = NO_SSLv3
```

Use *sslVersionMax* or *sslVersionMin* option instead of disabling specific TLS protocol versions when compiled with **OpenSSL 1.1.0** or later.

protocol = PROTO

application protocol to negotiate TLS

This option enables initial, protocol-specific negotiation of the TLS encryption. The *protocol* option should not be used with TLS encryption on a separate port.

Currently supported protocols:

cifs Proprietary (undocummented) extension of CIFS protocol implemented in Samba. Support for this extension was dropped in Samba 3.0.0.

capwin

http://www.capwin.org/ application support

capwinctrl

http://www.capwin.org/ application support

This protocol is only supported in client mode.

connect

Based on RFC 2817 - *Upgrading to TLS Within HTTP/1.1*, section 5.2 - *Requesting a Tunnel with CONNECT*

This protocol is only supported in client mode.

imap

Based on RFC 2595 - *Using TLS with IMAP, POP3 and ACAP*

ldap Based on RFC 2830 - *Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security*

nntp

Based on RFC 4642 - *Using Transport Layer Security (TLS) with Network News Transfer Protocol (NNTP)*

This protocol is only supported in client mode.

pgsql

Based on *http://www.postgresql.org/docs/8.3/static/protocol-flow.html#AEN73982*

pop3

Based on RFC 2449 - *POP3 Extension Mechanism*

proxy

Passing of the original client IP address with HAProxy PROXY protocol version 1 *https://www.haproxy.org/download/1.8/doc/proxy-protocol.txt*

smtp

Based on RFC 2487 - *SMTP Service Extension for Secure SMTP over TLS*

socks

SOCKS versions 4, 4a, and 5 are supported. The SOCKS protocol itself is encapsulated within TLS encryption layer to protect the final destination address.

http://www.openssh.com/txt/socks4.protocol

http://www.openssh.com/txt/socks4a.protocol

The BIND command of the SOCKS protocol is not supported. The USERID parameter is ignored.

See Examples section for sample configuration files for VPN based on SOCKS encryption.

protocolAuthentication = AUTHENTICATION

authentication type for the protocol negotiations

Currently, this option is only supported in the client-side ’connect’ and ’smtp’ protocols.

Supported authentication types for the ’connect’ protocol are ’basic’ or ’ntlm’. The default ’connect’ authentication type is ’basic’.

Supported authentication types for the ’smtp’ protocol are ’plain’ or ’login’. The default ’smtp’ authentication type is ’plain’.

protocolDomain = DOMAIN

domain for the protocol negotiations

Currently, this option is only supported in the client-side ’connect’ protocol.

protocolHeader = HEADER
header for the protocol negotiations

Currently, this option is only supported in the client-side ’connect’ protocol.

protocolHost = ADDRESS
host address for the protocol negotiations

For the ’connect’ protocol negotiations, *protocolHost* specifies HOST:PORT of the final TLS server to be connected to by the proxy. The proxy server directly connected by **stunnel** must be specified with the *connect* option.

For the ’smtp’ protocol negotiations, *protocolHost* controls the client SMTP HELO/EHLO value.

protocolPassword = PASSWORD
password for the protocol negotiations

Currently, this option is only supported in the client-side ’connect’ and ’smtp’ protocols.

protocolUsername = USERNAME
username for the protocol negotiations

Currently, this option is only supported in the client-side ’connect’ and ’smtp’ protocols.

PSKidentity = IDENTITY
PSK identity for the PSK client

PSKidentity can be used on **stunnel** clients to select the PSK identity used for authentication. This option is ignored in server sections.

default: the first identity specified in the *PSKsecrets* file.

PSKsecrets = FILE
file with PSK identities and corresponding keys

Each line of the file in the following format:

```
IDENTITY:KEY
```

Hexadecimal keys are automatically converted to binary form. Keys are required to be at least 16 bytes long, which implies at least 32 characters for hexadecimal keys. The file should neither be world-readable nor world-writable.

pty = yes | no (Unix only)
allocate a pseudoterminal for ’exec’ option

redirect = [HOST:]PORT
redirect TLS client connections on certificate-based authentication failures

This option only works in server mode. Some protocol negotiations are also incompatible with the *redirect* option.

renegotiation = yes | no
support TLS renegotiation

Applications of the TLS renegotiation include some authentication scenarios, or re-keying long lasting connections.

On the other hand this feature can facilitate a trivial CPU-exhaustion DoS attack:

http://vincent.bernat.im/en/blog/2011-ssl-dos-mitigation.html

Please note that disabling TLS renegotiation does not fully mitigate this issue.

default: yes (if supported by **OpenSSL**)

reset = yes | no
attempt to use the TCP RST flag to indicate an error

This option is not supported on some platforms.

default: yes

retry = yes | no | DELAY
reconnect a connect+exec section after it was disconnected

The DELAY value specifies the number of milliseconds before retrying. "retry = yes" has the same effect as "retry = 1000".

default: no

securityLevel = LEVEL

set the security level

The meaning of each level is described below:

level 0

Everything is permitted.

level 1

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

level 2

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

level 3

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

level 4

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

level 5

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

default: 2

The *securityLevel* option is only available when compiled with **OpenSSL 1.1.0** and later.

requireCert = yes | no

require a client certificate for *verifyChain* or *verifyPeer*

With *requireCert* set to *no*, the **stunnel** server accepts client connections that did not present a certificate.

Both *verifyChain = yes* and *verifyPeer = yes* imply *requireCert = yes*.

default: no

setgid = GROUP (Unix only)

Unix group id

As a global option: **setgid()** to the specified group in daemon mode and clear all other groups.

As a service-level option: set the group of the Unix socket specified with "accept".

setuid = USER (Unix only)

Unix user id

As a global option: **setuid()** to the specified user in daemon mode.

As a service-level option: set the owner of the Unix socket specified with "accept".

sessionCacheSize = NUM_ENTRIES

session cache size

sessionCacheSize specifies the maximum number of the internal session cache entries.

The value of 0 can be used for unlimited size. It is not recommended for production use due to the risk of a memory exhaustion DoS attack.

sessionCacheTimeout = TIMEOUT

session cache timeout

This is the number of seconds to keep cached TLS sessions.

sessionResume = yes | no

allow or disallow session resumption

default: yes

sessiond = HOST:PORT

address of sessiond TLS cache server

sni = SERVICE_NAME:SERVER_NAME_PATTERN (server mode)

Use the service as a secondary service (a name-based virtual server) for Server Name Indication TLS extension (RFC 3546).

SERVICE_NAME specifies the primary service that accepts client connections with the *accept* option. *SERVER_NAME_PATTERN* specifies the host name to be redirected. The pattern may start with the ’*’ character, e.g. ’*.example.com’. Multiple secondary services are normally specified for a single primary service. The *sni* option can also be specified more than once within a single secondary service.

This service, as well as the primary service, may not be configured in client mode.

The *connect* option of the secondary service is ignored when the *protocol* option is specified, as *protocol* connects to the remote host before TLS handshake.

Libwrap checks (Unix only) are performed twice: with the primary service name after TCP connection is accepted, and with the secondary service name during the TLS handshake.

The *sni* option is only available when compiled with **OpenSSL 1.0.0** and later.

sni = SERVER_NAME (client mode)

Use the parameter as the value of TLS Server Name Indication (RFC 3546) extension.

Empty SERVER_NAME disables sending the SNI extension.

The *sni* option is only available when compiled with **OpenSSL 1.0.0** and later.

socket = a|l|r:OPTION=VALUE[:VALUE]

Set an option on the accept/local/remote socket

The values for the linger option are l_onof:l_linger. The values for the time are tv_sec:tv_usec.

Examples:

```
socket = l:SO_LINGER=1:60
    set one minute timeout for closing local socket
socket = r:SO_OOBINLINE=yes
    place out-of-band data directly into the
    receive data stream for remote sockets
socket = a:SO_REUSEADDR=no
    disable address reuse (enabled by default)
socket = a:SO_BINDTODEVICE=lo
    only accept connections on loopback interface
```

sslVersion = SSL_VERSION

select the TLS protocol version

Supported versions: all, SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

Availability of specific protocols depends on the linked OpenSSL library. Older versions of OpenSSL do not support TLSv1.1, TLSv1.2 and TLSv1.3. Newer versions of OpenSSL do not support SSLv2.

Obsolete SSLv2 and SSLv3 are currently disabled by default.

Setting the option

```
sslVersion = SSL_VERSION
```

is equivalent to options

```
sslVersionMax = SSL_VERSION
sslVersionMin = SSL_VERSION
```

when compiled with **OpenSSL 1.1.0** and later.

sslVersionMax = SSL_VERSION

maximum supported protocol versions

Supported versions: all, SSLv3, TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

all enable protocol versions up to the highest version supported by the linked OpenSSL library.

Availability of specific protocols depends on the linked OpenSSL library.

The *sslVersionMax* option is only available when compiled with **OpenSSL 1.1.0** and later.

default: all

sslVersionMin = SSL_VERSION
minimum supported protocol versions

Supported versions: all, SSLv3, TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

all enable protocol versions down to the lowest version supported by the linked OpenSSL library.

Availability of specific protocols depends on the linked OpenSSL library.

The *sslVersionMin* option is only available when compiled with **OpenSSL 1.1.0** and later.

default: TLSv1

stack = BYTES (except for FORK model)
CPU stack size of created threads

Excessive thread stack size increases virtual memory usage. Insufficient thread stack size may cause application crashes.

default: 65536 bytes (sufficient for all platforms we tested)

ticketKeySecret = SECRET
hexadecimal symmetric key used for session ticket confidentiality protection

Session tickets defined in RFC 5077 provide an enhanced session resumption capability, where the server-side caching is not required to maintain per session state.

Combining *ticketKeySecret* and *ticketMacSecret* options allow to resume a negotiated session on other cluster nodes, or to resume a negotiated session after server restart.

The key is required to be either 16 or 32 bytes long, which implies exactly 32 or 64 hexadecimal digits. Colons may optionally be used between two-character hexadecimal bytes.

This option only works in server mode.

The *ticketKeySecret* option is only available when compiled with **OpenSSL 1.0.0** and later.

Disabling *NO_TICKET* option is required for the ticket support in OpenSSL older than 1.1.1, but note that this option is incompatible with the *redirect* option.

ticketMacSecret = SECRET

hexadecimal symmetric key used for session ticket integrity protection

The key is required to be either 16 or 32 bytes long, which implies exactly 32 or 64 hexadecimal digits. Colons may optionally be used between two-character hexadecimal bytes.

This option only works in server mode.

The *ticketMacSecret* option is only available when compiled with **OpenSSL 1.0.0** and later.

TIMEOUTbusy = SECONDS

time to wait for expected data

TIMEOUTclose = SECONDS

time to wait for close_notify (set to 0 for buggy MSIE)

TIMEOUTconnect = SECONDS

time to wait to connect a remote host

TIMEOUTidle = SECONDS

time to keep an idle connection

TIMEOUTocsp = SECONDS

time to wait to connect an OCSP responder

transparent = none | source | destination | both (Unix only)

enable transparent proxy support on selected platforms

Supported values:

none

Disable transparent proxy support. This is the default.

source

Re-write the address to appear as if a wrapped daemon is connecting from the TLS client machine instead of the machine running **stunnel**.

This option is currently available in:

Remote mode (*connect* option) on *Linux >=2.6.28*

This configuration requires **stunnel** to be executed as root and without the *setuid* option.

This configuration requires the following setup for iptables and routing (possibly in /etc/rc.local or equivalent file):

```
iptables -t mangle -N DIVERT
iptables -t mangle -A PREROUTING -p tcp -m socket -j DIVERT
iptables -t mangle -A DIVERT -j MARK --set-mark 1
iptables -t mangle -A DIVERT -j ACCEPT
ip rule add fwmark 1 lookup 100
ip route add local 0.0.0.0/0 dev lo table 100
echo 0 >/proc/sys/net/ipv4/conf/lo/rp_filter
```

stunnel must also to be executed as root and without the *setuid* option.

Remote mode (*connect* option) on *Linux 2.2.x*

This configuration requires the kernel to be compiled with the *transparent proxy* option. Connected service must be installed on a separate host. Routing towards the clients has to go through the **stunnel** box.

stunnel must also to be executed as root and without the *setuid* option.

Remote mode (*connect* option) on *FreeBSD >=8.0*

This configuration requires additional firewall and routing setup. **stunnel** must also to be executed as root and without the *setuid* option.

Local mode (*exec* option)

This configuration works by pre-loading the *libstunnel.so* shared library. _RLD_LIST environment variable is used on Tru64, and LD_PRELOAD variable on other platforms.

destination

The original destination is used instead of the *connect* option.

A service section for transparent destination may look like this:

```
[transparent]
client = yes
accept = <stunnel_port>
transparent = destination
```

This configuration requires iptables setup to work, possibly in /etc/rc.local or equivalent file.

For a connect target installed on the same host:

```
/sbin/iptables -t nat -I OUTPUT -p tcp --dport <redirected_port> \
    -m ! --uid-owner <stunnel_user_id> \
    -j DNAT --to-destination <local_ip>:<stunnel_port>
```

For a connect target installed on a remote host:

```
/sbin/iptables -I INPUT -i eth0 -p tcp --dport <stunnel_port> -j ACCEPT
/sbin/iptables -t nat -I PREROUTING -p tcp --dport <redirected_port> \
    -i eth0 -j DNAT --to-destination <local_ip>:<stunnel_port>
```

The transparent destination option is currently only supported on Linux.

both

Use both *source* and *destination* transparent proxy.

Two legacy options are also supported for backward compatibility:

yes This option has been renamed to *source*.

no This option has been renamed to *none*.

verify = LEVEL

verify the peer certificate

This option is obsolete and should be replaced with the *verifyChain* and *verifyPeer* options.

level 0

Request and ignore the peer certificate chain.

level 1

Verify the peer certificate chain if present.

level 2

Verify the peer certificate chain.

level 3

Verify the peer certificate chain and the end-entity (leaf) peer certificate against a locally installed certificate.

level 4

Ignore the peer certificate chain and only verify the end-entity (leaf) peer certificate against a locally installed certificate.

default

No verify.

verifyChain = yes | no

verify the peer certificate chain starting from the root CA

For server certificate verification it is essential to also require a specific certificate with *checkHost* or *checkIP*.

The self-signed root CA certificate needs to be stored either in the file specified with *CAfile*, or in the directory specified with *CApath*.

default: no

verifyPeer = yes | no

verify the end-entity (leaf) peer certificate

The end-entity (leaf) peer certificate needs to be stored either in the file specified with *CAfile*, or in the directory specified with *CApath*.

default: no

RETURN VALUE

stunnel returns zero on success, non-zero on error.

SIGNALS

The following signals can be used to control **stunnel** in Unix environment:

SIGHUP

Force a reload of the configuration file.

Some global options will not be reloaded:

- chroot
- foreground
- pid
- setgid
- setuid

The use of the ’setuid’ option will also prevent **stunnel** from binding to privileged (<1024) ports during configuration reloading.

When the ’chroot’ option is used, **stunnel** will look for all its files (including the configuration file, certificates, the log file and the pid file) within the chroot jail.

SIGUSR1
Close and reopen the **stunnel** log file. This function can be used for log rotation.

SIGUSR2
Log the list of active connections.

SIGTERM, SIGQUIT, SIGINT
Shut **stunnel** down.

The result of sending any other signals to the server is undefined.

EXAMPLES

In order to provide TLS encapsulation to your local *imapd* service, use:

```
[imapd]
accept = 993
exec = /usr/sbin/imapd
execArgs = imapd
```

or in remote mode:

```
[imapd]
accept = 993
connect = 143
```

In order to let your local e-mail client connect to a TLS-enabled *imapd* service on another server, configure the e-mail client to connect to localhost on port 119 and use:

```
[imap]
client = yes
accept = 143
connect = servername:993
```

If you want to provide tunneling to your *pppd* daemon on port 2020, use something like:

```
[vpn]
accept = 2020
exec = /usr/sbin/pppd
execArgs = pppd local
pty = yes
```

If you want to use **stunnel** in *inetd* mode to launch your imapd process, you'd use this *stunnel.conf*. Note there must be no *[service_name]* section.

```
exec = /usr/sbin/imapd
execArgs = imapd
```

To setup SOCKS VPN configure the following client service:

```
[socks_client]
client = yes
accept = 127.0.0.1:1080
connect = vpn_server:9080
verifyPeer = yes
CAfile = stunnel.pem
```

The corresponding configuration on the vpn_server host:

```
[socks_server]
protocol = socks
accept = 9080
```

```
cert = stunnel.pem
key = stunnel.key
```

Now test your configuration on the client machine with:

```
curl --socks4a localhost http://www.example.com/
```

An example server mode SNI configuration:

```
[virtual]
; primary service
accept = 443
cert =  default.pem
connect = default.internal.mydomain.com:8080

[sni1]
; secondary service 1
sni = virtual:server1.mydomain.com
cert = server1.pem
connect = server1.internal.mydomain.com:8081

[sni2]
; secondary service 2
sni = virtual:server2.mydomain.com
cert = server2.pem
connect = server2.internal.mydomain.com:8082
verifyPeer = yes
CAfile = server2-allowed-clients.pem
```

An example of advanced engine configuration allows for authentication with private keys stored in the Windows certificate store (Windows only). With the CAPI engine you don’t need to manually select the client key to use. The client key is automatically selected based on the list of CAs trusted by the server.

```
engine = capi

[service]
engineId = capi
client = yes
accept = 127.0.0.1:8080
```

```
connect = example.com:8443
```

An example of advanced engine configuration to use the certificate and the corresponding private key from a pkcs11 engine:

```
engine = pkcs11
engineCtrl = MODULE_PATH:opensc-pkcs11.so
engineCtrl = PIN:123456

[service]
engineId = pkcs11
client = yes
accept = 127.0.0.1:8080
connect = example.com:843
cert = pkcs11:token=MyToken;object=MyCert
key = pkcs11:token=MyToken;object=MyKey
```

An example of advanced engine configuration to use the certificate and the corresponding private key from a SoftHSM token:

```
engine = pkcs11
engineCtrl = MODULE_PATH:softhsm2.dll
engineCtrl = PIN:12345

[service]
engineId = pkcs11
client = yes
accept = 127.0.0.1:8080
connect = example.com:843
cert = pkcs11:token=MyToken;object=KeyCert
```

NOTES

RESTRICTIONS

stunnel cannot be used for the FTP daemon because of the nature of the FTP protocol which utilizes multiple ports for data transfers. There are available TLS-enabled versions of FTP and telnet daemons, however.

INETD MODE

The most common use of **stunnel** is to listen on a network port and establish communication with either a new port via the connect option, or a new program via the *exec* option. However there is a special case when you wish to have some other program accept incoming connections and launch **stunnel**, for example with *inetd*, *xinetd*, or *tcpserver*.

For example, if you have the following line in *inetd.conf*:

```
imaps stream tcp nowait root /usr/local/bin/stunnel stunnel /usr/local/etc/stunnel/imaps.conf
```

In these cases, the *inetd*-style program is responsible for binding a network socket (*imaps* above) and handing it to **stunnel** when a connection is received. Thus you do not want **stunnel** to have any *accept* option. All the *Service Level Options* should be placed in the global options section, and no *[service_name]* section will be present. See the *EXAMPLES* section for example configurations.

CERTIFICATES

Each TLS-enabled daemon needs to present a valid X.509 certificate to the peer. It also needs a private key to decrypt the incoming data. The easiest way to obtain a certificate and a key is to generate them with the free **OpenSSL** package. You can find more information on certificates generation on pages listed below.

The *.pem* file should contain the unencrypted private key and a signed certificate (not certificate request). So the file should look like this:

```
-----BEGIN RSA PRIVATE KEY-----
[encoded key]
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
[encoded certificate]
-----END CERTIFICATE-----
```

RANDOMNESS

stunnel needs to seed the PRNG (pseudo-random number generator) in order for TLS to use good randomness. The following sources are loaded in order until sufficient random data has been gathered:

- The file specified with the *RNDfile* flag.
- The file specified by the RANDFILE environment variable, if set.
- The file .rnd in your home directory, if RANDFILE not set.
- The file specified with '--with-random' at compile time.

- The contents of the screen if running on Windows.
- The egd socket specified with the *EGD* flag.
- The egd socket specified with '--with-egd-sock' at compile time.
- The /dev/urandom device.

Note that on Windows machines that do not have console user interaction (mouse movements, creating windows, etc.) the screen contents are not variable enough to be sufficient, and you should provide a random file for use with the *RNDfile* flag.

Note that the file specified with the *RNDfile* flag should contain random data -- that means it should contain different information each time **stunnel** is run. This is handled automatically unless the *RNDoverwrite* flag is used. If you wish to update this file manually, the *openssl rand* command in recent versions of **OpenSSL**, would be useful.

Important note: If /dev/urandom is available, **OpenSSL** often seeds the PRNG with it while checking the random state. On systems with /dev/urandom **OpenSSL** is likely to use it even though it is listed at the very bottom of the list above. This is the behaviour of **OpenSSL** and not **stunnel**.

DH PARAMETERS

stunnel 4.40 and later contains hardcoded 2048-bit DH parameters. Starting with **stunnel** 5.18, these hardcoded DH parameters are replaced every 24 hours with autogenerated temporary DH parameters. DH parameter generation may take several minutes.

Alternatively, it is possible to specify static DH parameters in the certificate file, which disables generating temporary DH parameters:

```
openssl dhparam 2048 >> stunnel.pem
```

FILES

/usr/local/etc/stunnel/stunnel.conf
stunnel configuration file

BUGS

The *execArgs* option and the Win32 command line do not support quoting.

SEE ALSO

tcpd(8)

access control facility for internet services

inetd(8)
internet ’super-server’

http://www.stunnel.org/
stunnel homepage

http://www.openssl.org/
OpenSSL project website

AUTHOR

Michał Trojnara
<*Michal.Trojnara@stunnel.org*>