NAME

openssl-s_server, s_server - SSL/TLS server program

SYNOPSIS

openssl s_server [**-help**] [**-port port**] [**-accept val**] [**-naccept count**] [**-unix val**] [**-unlink**] [**-4**] [**-6**] [**-context id**] [**-verify depth**] [**-Verify depth**] [**-crl_check**] [**-crl_check_all**] [**-cert filename**] [**-certform DER|PEM**] [**-key keyfile**] [**-keyform DER|PEM**] [**-pass arg**] [**-dcert filename**] [**-dcertform DER|PEM**] [**-dkey keyfile**] [**-dkeyform DER|PEM**] [**-dpass arg**] [**-dhparam filename**] [**-nbio**] [**-nbio_test**] [**-crlf**] [**-debug**] [**-msg**] [**-state**] [**-CApath directory**] [**-CAfile filename**] [**-no-CAfile**] [**-no-CApath**] [**-attime timestamp**] [**-check_ss_sig**] [**-explicit_policy**] [**-extended_crl**] [**-ignore_critical**] [**-inhibit_any**] [**-inhibit_map**] [**-no_check_time**] [**-partial_chain**] [**-policy arg**] [**-policy_check**] [**-policy_print**] [**-purpose purpose**] [**-suiteB_128**] [**-suiteB_128_only**] [**-suiteB_192**] [**-trusted_first**] [**-no_alt_chains**] [**-use_deltas**] [**-auth_level num**] [**-verify_depth num**] [**-verify_return_error**] [**-verify_email email**] [**-verify_hostname hostname**] [**-verify_ip ip**] [**-verify_name name**] [**-x509_strict**] [**-nocert**] [**-client_sigalgs sigalglist**] [**-named_curve curve**] [**-cipher cipherlist**] [**-serverpref**] [**-quiet**] [**-ssl3**] [**-tls1**] [**-tls1_1**] [**-tls1_2**] [**-dtls**] [**-dtls1**] [**-dtls1_2**] [**-listen**] [**-async**] [**-split_send_frag**] [**-max_pipelines**] [**-read_buf**] [**-no_ssl3**] [**-no_tls1**] [**-no_tls1_1**] [**-no_tls1_2**] [**-no_dhe**] [**-bugs**] [**-comp**] [**-no_comp**] [**-brief**] [**-www**] [**-WWW**] [**-HTTP**] [**-engine id**] [**-tlsextdebug**] [**-no_ticket**] [**-id_prefix arg**] [**-rand file(s)**] [**-serverinfo file**] [**-no_resumption_on_reneg**] [**-status**] [**-status_verbose**] [**-status_timeout nsec**] [**-status_url url**] [**-alpn protocols**] [**-nextprotoneg protocols**]

DESCRIPTION

The **s_server** command implements a generic SSL/TLS server which listens for connections on a given port using SSL/TLS.

OPTIONS

In addition to the options below the **s_server** utility also supports the common and server only options documented in the in the ‘‘Supported Command Line Commands’’ section of the *SSL_CONF_cmd(3)* manual page.

-help
Print out a usage message.

-port port
The TCP port to listen on for connections. If not specified 4433 is used.

-accept val
The optional TCP host and port to listen on for connections. If not specified, *:4433 is used.

-naccept count
The server will exit after receiving **number** connections, default unlimited.

-unix val
Unix domain socket to accept on.

-unlink
For -unix, unlink existing socket first.

-4 Use IPv4 only.

-6 Use IPv6 only.

-context id
Sets the SSL context id. It can be given any string value. If this option is not present a default value will be used.

-cert certname
The certificate to use, most servers cipher suites require the use of a certificate and some require a certificate with a certain public key type: for example the DSS cipher suites require a certificate containing a DSS (DSA) key. If not specified then the filename ‘‘server.pem’’ will be used.

-certform format
The certificate format to use: DER or PEM. PEM is the default.

-key keyfile
The private key to use. If not specified then the certificate file will be used.

-keyform format
The private format to use: DER or PEM. PEM is the default.

-pass arg
The private key password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl(1)* .

-dcert filename, **-dkey keyname**
Specify an additional certificate and private key, these behave in the same manner as the **-cert** and **-key** options except there is no default if they are not specified (no additional certificate and key is used). As noted above some cipher suites require a certificate containing a key of a certain type. Some cipher suites need a certificate carrying an RSA key and some a DSS (DSA) key. By using RSA and DSS certificates and keys a server can support clients which only support RSA or DSS cipher suites by using an appropriate certificate.

-dcertform format, **-dkeyform format**, **-dpass arg**
Additional certificate and private key format and passphrase respectively.

-nocert
If this option is set then no certificate is used. This restricts the cipher suites available to the anonymous ones (currently just anonymous DH).

-dhparam filename
The DH parameter file to use. The ephemeral DH cipher suites generate keys using a set of DH parameters. If not specified then an attempt is made to load the parameters from the server certificate file. If this fails then a static set of parameters hard coded into the **s_server** program will be used.

-no_dhe
If this option is set then no DH parameters will be loaded effectively disabling the ephemeral DH cipher suites.

-crl_check, **-crl_check_all**
Check the peer certificate has not been revoked by its CA. The CRL(s) are appended to the certificate file. With the **-crl_check_all** option all CRLs of all CAs in the chain are checked.

-CApath directory
The directory to use for client certificate verification. This directory must be in ''hash format'', see **verify** for more information. These are also used when building the server certificate chain.

-CAfile file
A file containing trusted certificates to use during client authentication and to use when attempting to build the server certificate chain. The list is also used in the list of acceptable client CAs passed to the client when a certificate is requested.

-no-CAfile
Do not load the trusted CA certificates from the default file location

-no-CApath
Do not load the trusted CA certificates from the default directory location

-verify depth, **-Verify depth**
The verify depth to use. This specifies the maximum length of the client certificate chain and makes the server request a certificate from the client. With the **-verify** option a certificate is requested but the client does not have to send one, with the **-Verify** option the client must supply a certificate or an error occurs.

If the ciphersuite cannot request a client certificate (for example an anonymous ciphersuite or PSK) this option has no effect.

-attime, **-check_ss_sig**, **-crl_check**, **-crl_check_all**, **-explicit_policy**, **-extended_crl**, **-ignore_critical**, **-inhibit_any**, **-inhibit_map**, **-no_alt_chains**, **-no_check_time**, **-partial_chain**, **-policy**, **-policy_check**, **-policy_print**, **-purpose**, **-suiteB_128**, **-suiteB_128_only**, **-suiteB_192**, **-trusted_first**, **-use_deltas**, **-auth_level**, **-verify_depth**, **-verify_email**, **-verify_hostname**, **-verify_ip**, **-verify_name**, **-x509_strict**

Set different peer certificate verification options. See the *verify(1)* manual page for details.

-verify_return_error

Verification errors normally just print a message but allow the connection to continue, for debugging purposes. If this option is used, then verification errors close the connection.

-state

Prints the SSL session states.

-debug

Print extensive debugging information including a hex dump of all traffic.

-msg

Show all protocol messages with hex dump.

-trace

Show verbose trace output of protocol messages. OpenSSL needs to be compiled with **enable-ssl-trace** for this option to work.

-msgfile

File to send output of **-msg** or **-trace** to, default standard output.

-nbio_test

Tests non blocking I/O

-nbio

Turns on non blocking I/O

-crlf

This option translated a line feed from the terminal into CR+LF.

-quiet

Inhibit printing of session and certificate information.

-psk_hint hint

Use the PSK identity hint **hint** when using a PSK cipher suite.

-psk key

Use the PSK key **key** when using a PSK cipher suite. The key is given as a hexadecimal number without leading 0x, for example -psk 1a2b3c4d. This option must be provided in order to use a PSK cipher.

-ssl2, **-ssl3**, **-tls1**, **-tls1_1**, **-tls1_2**, **-no_ssl2**, **-no_ssl3**, **-no_tls1**, **-no_tls1_1**, **-no_tls1_2**

These options require or disable the use of the specified SSL or TLS protocols. By default **s_server** will negotiate the highest mutually supported protocol version. When a specific TLS version is required, only that version will be accepted from the client.

-dtls, **-dtls1**, **-dtls1_2**

These options make **s_server** use DTLS protocols instead of TLS. With **-dtls**, **s_server** will negotiate any supported DTLS protocol version, whilst **-dtls1** and **-dtls1_2** will only support DTLSv1.0 and DTLSv1.2 respectively.

-listen

This option can only be used in conjunction with one of the DTLS options above. With this option **s_server** will listen on a UDP port for incoming connections. Any ClientHellos that arrive will be checked to see if they have a cookie in them or not. Any without a cookie will be responded to with a HelloVerifyRequest. If a ClientHello with a cookie is received then **s_server** will connect to that peer and complete the handshake.

-async
Switch on asynchronous mode. Cryptographic operations will be performed asynchronously. This will only have an effect if an asynchronous capable engine is also used via the **-engine** option. For test purposes the dummy async engine (dasync) can be used (if available).

-split_send_frag int
The size used to split data for encrypt pipelines. If more data is written in one go than this value then it will be split into multiple pipelines, up to the maximum number of pipelines defined by max_pipelines. This only has an effect if a suitable ciphersuite has been negotiated, an engine that supports pipelining has been loaded, and max_pipelines is greater than 1. See *SSL_CTX_set_split_send_fragment(3)* for further information.

-max_pipelines int
The maximum number of encrypt/decrypt pipelines to be used. This will only have an effect if an engine has been loaded that supports pipelining (e.g. the dasync engine) and a suitable ciphersuite has been negotiated. The default value is 1. See *SSL_CTX_set_max_pipelines(3)* for further information.

-read_buf int
The default read buffer size to be used for connections. This will only have an effect if the buffer size is larger than the size that would otherwise be used and pipelining is in use (see *SSL_CTX_set_default_read_buffer_len(3)* for further information).

-bugs
There are several known bug in SSL and TLS implementations. Adding this option enables various workarounds.

-comp
Enable negotiation of TLS compression. This option was introduced in OpenSSL 1.1.0. TLS compression is not recommended and is off by default as of OpenSSL 1.1.0.

-no_comp
Disable negotiation of TLS compression. TLS compression is not recommended and is off by default as of OpenSSL 1.1.0.

-brief
Provide a brief summary of connection parameters instead of the normal verbose output.

-client_sigalgs sigalglist
Signature algorithms to support for client certificate authentication (colon-separated list)

-named_curve curve
Specifies the elliptic curve to use. NOTE: this is single curve, not a list. For a list of all possible curves, use:

```
 $ openssl ecparam -list_curves
```

-cipher cipherlist
This allows the cipher list used by the server to be modified. When the client sends a list of supported ciphers the first client cipher also included in the server list is used. Because the client specifies the preference order, the order of the server cipherlist irrelevant. See the **ciphers** command for more information.

-serverpref
Use the server's cipher preferences, rather than the client's preferences.

-tlsextdebug
Print a hex dump of any TLS extensions received from the server.

-no_ticket
Disable RFC4507bis session ticket support.

-www
Sends a status message back to the client when it connects. This includes information about the ciphers used and various session parameters. The output is in HTML format so this option will normally be used with a web browser.

-WWW
Emulates a simple web server. Pages will be resolved relative to the current directory, for example if the URL https://myhost/page.html is requested the file ./page.html will be loaded.

-HTTP
Emulates a simple web server. Pages will be resolved relative to the current directory, for example if the URL https://myhost/page.html is requested the file ./page.html will be loaded. The files loaded are assumed to contain a complete and correct HTTP response (lines that are part of the HTTP response line and headers must end with CRLF).

-rev
Simple test server which just reverses the text received from the client and sends it back to the server. Also sets **-brief**.

-engine id
Specifying an engine (by its unique **id** string) will cause **s_server** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

-id_prefix arg
Generate SSL/TLS session IDs prefixed by **arg**. This is mostly useful for testing any SSL/TLS code (eg. proxies) that wish to deal with multiple servers, when each of which might be generating a unique range of session IDs (eg. with a certain prefix).

-rand file(s)
A file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd(3)*). Multiple files can be specified separated by an OS-dependent character. The separator is **;** for MS-Windows, **,** for OpenVMS, and **:** for all others.

-serverinfo file
A file containing one or more blocks of PEM data. Each PEM block must encode a TLS ServerHello extension (2 bytes type, 2 bytes length, followed by "length" bytes of extension data). If the client sends an empty TLS ClientHello extension matching the type, the corresponding ServerHello extension will be returned.

-no_resumption_on_reneg
Set the **SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION** option.

-status
Enables certificate status request support (aka OCSP stapling).

-status_verbose
Enables certificate status request support (aka OCSP stapling) and gives a verbose printout of the OCSP response.

-status_timeout nsec
Sets the timeout for OCSP response to **nsec** seconds.

-status_url url
Sets a fallback responder URL to use if no responder URL is present in the server certificate. Without this option an error is returned if the server certificate does not contain a responder address.

-alpn protocols, **-nextprotoneg protocols**
these flags enable the Enable the Application-Layer Protocol Negotiation or Next Protocol Negotiation extension, respectively. ALPN is the IETF standard and replaces NPN. The**pr otocols** list is a comma-separated list of supported protocol names. The list should contain most wanted protocols first. Protocol names are printable ASCII strings, for example "http/1.1" or "spdy/3".

CONNECTED COMMANDS

If a connection request is established with an SSL client and neither the **-www** nor the **-WWW** option has been used then normally any data received from the client is displayed and any key presses will be sent to the client.

Certain single letter commands are also recognized which perform special operations: these are listed below.

q end the current SSL connection but still accept new connections.

Q end the current SSL connection and exit.

r renegotiate the SSL session.

R renegotiate the SSL session and request a client certificate.

P send some plain text down the underlying TCP connection: this should cause the client to disconnect due to a protocol violation.

S print out some session cache status information.

NOTES

s_server can be used to debug SSL clients. To accept connections from a web browser the command:

```
 openssl s_server -accept 443 -www
```

can be used for example.

Although specifying an empty list of CAs when requesting a client certificate is strictly speaking a protocol violation, some SSL clients interpret this to mean any CA is acceptable. This is useful for debugging purposes.

The session parameters can printed out using the **sess_id** program.

BUGS

Because this program has a lot of options and also because some of the techniques used are rather old, the C source of **s_server** is rather hard to read and not a model of how things should be done. A typical SSL server program would be much simpler.

The output of common ciphers is wrong: it just gives the list of ciphers that OpenSSL recognizes and the client supports.

There should be a way for the **s_server** program to print out details of any unknown cipher suites a client says it supports.

SEE ALSO

SSL_CONF_cmd(3) , *sess_id(1)* , *s_client(1)* , *ciphers(1)*

HISTORY

The -no_alt_chains options was first added to OpenSSL 1.1.0.

COPYRIGHT