NAME
gnutls-serv - GnuTLS server

SYNOPSIS
gnutls-serv [**-flags**] [**-flag** [*value*]] [**--option-name**[[=|]*value*]]

All arguments must be options.

DESCRIPTION
Server program that listens to incoming TLS connections.

OPTIONS
-d *num*, **--debug**=*num*

Enable debugging. This option takes an integer number as its argument. The value of *num* is constrained to being:

in the range 0 through 9999

Specifies the debug level.

--sni-hostname=*str*

Server's hostname for server name extension.

Server name of type host_name that the server will recognise as its own. If the server receives client hello with different name, it will send a warning-level unrecognized_name alert.

--sni-hostname-fatal

Send fatal alert on sni-hostname mismatch.

--alpn=*str*

Specify ALPN protocol to be enabled by the server. This option may appear an unlimited number of times.

Specify the (textual) ALPN protocol for the server to use.

--alpn-fatal

Send fatal alert on non-matching ALPN name.

--noticket

    Don't accept session tickets.

--earlydata

    Accept early data.

--maxearlydata=*num*

    The maximum early data size to accept. This option takes an integer number as its argument. The value of *num* is constrained to being:

        in the range 1 through 2147483648

--nocookie

    Don't require cookie on DTLS sessions.

-g, **--generate**

    Generate Diffie-Hellman parameters.

-q, **--quiet**

    Suppress some messages.

--nodb

    Do not use a resumption database.

--http

    Act as an HTTP server.

--echo

    Act as an Echo server.

--crlf

    Do not replace CRLF by LF in Echo server mode.

-u, **--udp**

    Use DTLS (datagram TLS) over UDP.

--mtu=*num*

    Set MTU for datagram TLS. This option takes an integer number as its argument. The value of
    num is constrained to being:
        in the range 0 through 17000

--srtp-profiles=*str*

    Offer SRTP profiles.

-a, **--disable-client-cert**

    Do not request a client certificate. This option must not appear in combination with any of the
    following options: require-client-cert.

-r, **--require-client-cert**

    Require a client certificate.

    This option before 3.6.0 used to imply --verify-client-cert. Since 3.6.0 it will no longer verify the
    certificate by default.

--verify-client-cert

    If a client certificate is sent then verify it.

    Do not require, but if a client certificate is sent then verify it and close the connection if invalid.

--compress-cert=*str*

    Compress certificate. This option may appear an unlimited number of times.

    This option sets a supported compression method for certificate compression.

-b, **--heartbeat**

    Activate heartbeat support.

    Regularly ping client via heartbeat extension messages

--x509fmtder
> Use DER format for certificates to read from.

--priority=*str*
> Priorities string.

> TLS algorithms and protocols to enable. You can use predefined sets of ciphersuites such as PERFORMANCE, NORMAL, SECURE128, SECURE256. The default is NORMAL.

> Check the GnuTLS manual on section "Priority strings" for more information on allowed keywords

--dhparams=*file*
> DH params file to use.

--x509cafile=*str*
> Certificate file or PKCS #11 URL to use.

--x509crlfile=*file*
> CRL file to use.

--pgpkeyfile=*file*
> PGP Key file to use.

> **NOTE: THIS OPTION IS DEPRECATED**

--x509keyfile=*str*
> X.509 key file or PKCS #11 URL to use. This option may appear an unlimited number of times.

> Specify the private key file or URI to use; it must correspond to the certificate specified in --x509certfile. Multiple keys and certificates can be specified with this option and in that case each occurrence of keyfile must be followed by the corresponding x509certfile or vice-versa.

--x509certfile=*str*
> X.509 Certificate file or PKCS #11 URL to use. This option may appear an unlimited number of

times.

Specify the certificate file or URI to use; it must correspond to the key specified in --x509keyfile.
Multiple keys and certificates can be specified with this option and in that case each occurrence of
keyfile must be followed by the corresponding x509certfile or vice-versa.

--x509dsakeyfile
This is an alias for the *--x509keyfile* option.

NOTE: THIS OPTION IS DEPRECATED

--x509dsacertfile
This is an alias for the *--x509certfile* option.

NOTE: THIS OPTION IS DEPRECATED

--x509ecckeyfile
This is an alias for the *--x509keyfile* option.

NOTE: THIS OPTION IS DEPRECATED

--x509ecccertfile
This is an alias for the *--x509certfile* option.

NOTE: THIS OPTION IS DEPRECATED

--rawpkkeyfile=*str*
Private key file (PKCS #8 or PKCS #12) or PKCS #11 URL to use. This option may appear an
unlimited number of times.

Specify the private key file or URI to use; it must correspond to the raw public-key specified in
--rawpkfile. Multiple key pairs can be specified with this option and in that case each occurrence
of keyfile must be followed by the corresponding rawpkfile or vice-versa.

In order to instruct the application to negotiate raw public keys one must enable the respective
certificate types via the priority strings (i.e. CTYPE-CLI-* and CTYPE-SRV-* flags).

Check the GnuTLS manual on section "Priority strings" for more information on how to set
certificate types.

--rawpkfile=*str*

> Raw public-key file to use. This option may appear an unlimited number of times. This option must appear in combination with the following options: rawpkkeyfile.

> Specify the raw public-key file to use; it must correspond to the private key specified in --rawpkkeyfile. Multiple key pairs can be specified with this option and in that case each occurrence of keyfile must be followed by the corresponding rawpkfile or vice-versa.

> In order to instruct the application to negotiate raw public keys one must enable the respective certificate types via the priority strings (i.e. CTYPE-CLI-* and CTYPE-SRV-* flags).

> Check the GnuTLS manual on section "Priority strings" for more information on how to set certificate types.

--srppasswd=*file*

> SRP password file to use.

--srppasswdconf=*file*

> SRP password configuration file to use.

--pskpasswd=*file*

> PSK password file to use.

--pskhint=*str*

> PSK identity hint to use.

--ocsp-response=*str*

> The OCSP response to send to client. This option may appear an unlimited number of times.

> If the client requested an OCSP response, return data from this file to the client.

--ignore-ocsp-response-errors

> Ignore any errors when setting the OCSP response.

> That option instructs gnutls to not attempt to match the provided OCSP responses with the certificates.

-p *num*, **--port**=*num*
> The port to connect to. This option takes an integer number as its argument.

-l, **--list**
> Print a list of the supported algorithms and modes.

> Print a list of the supported algorithms and modes. If a priority string is given then only the
> enabled ciphersuites are shown.

--provider=*file*
> Specify the PKCS #11 provider library.

> This will override the default options in /usr/local/etc/gnutls/pkcs11.conf

--keymatexport=*str*
> Label used for exporting keying material.

--keymatexportsize=*num*
> Size of the exported keying material. This option takes an integer number as its argument.

--recordsize=*num*
> The maximum record size to advertise. This option takes an integer number as its argument. The
> value of *num* is constrained to being:
> > in the range 0 through 16384

--httpdata=*file*
> The data used as HTTP response.

-v *arg*, **--version**=*arg*
> Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will
> print copyright information and 'n' will print the full copyright notice.

-h, **--help**
> Display usage information and exit.

-!, **--more-help**

    Pass the extended usage information through a pager.

EXAMPLES

    Running your own TLS server based on GnuTLS can be useful when debugging clients and/or
    GnuTLS itself. This section describes how to use **gnutls-serv** as a simple HTTPS server.

    The most basic server can be started as:

        gnutls-serv --http --priority "NORMAL:+ANON-ECDH:+ANON-DH"

    It will only support anonymous ciphersuites, which many TLS clients refuse to use.

    The next step is to add support for X.509. First we generate a CA:

        $ certtool --generate-privkey > x509-ca-key.pem
        $ echo 'cn = GnuTLS test CA' > ca.tmpl
        $ echo 'ca' >> ca.tmpl
        $ echo 'cert_signing_key' >> ca.tmpl
        $ certtool --generate-self-signed --load-privkey x509-ca-key.pem   --template ca.tmpl --outfile x509-ca.pem

    Then generate a server certificate. Remember to change the dns_name value to the name of your server
    host, or skip that command to avoid the field.

        $ certtool --generate-privkey > x509-server-key.pem
        $ echo 'organization = GnuTLS test server' > server.tmpl
        $ echo 'cn = test.gnutls.org' >> server.tmpl
        $ echo 'tls_www_server' >> server.tmpl
        $ echo 'encryption_key' >> server.tmpl
        $ echo 'signing_key' >> server.tmpl
        $ echo 'dns_name = test.gnutls.org' >> server.tmpl
        $ certtool --generate-certificate --load-privkey x509-server-key.pem   --load-ca-certificate x509-ca.pem --load-ca-p

    For use in the client, you may want to generate a client certificate as well.

        $ certtool --generate-privkey > x509-client-key.pem
        $ echo 'cn = GnuTLS test client' > client.tmpl
        $ echo 'tls_www_client' >> client.tmpl

    $ echo 'encryption_key' >> client.tmpl
    $ echo 'signing_key' >> client.tmpl
    $ certtool --generate-certificate --load-privkey x509-client-key.pem   --load-ca-certificate x509-ca.pem --load-ca-p

To be able to import the client key/certificate into some applications, you will need to convert them
into a PKCS#12 structure. This also encrypts the security sensitive key with a password.

    $ certtool --to-p12 --load-ca-certificate x509-ca.pem   --load-privkey x509-client-key.pem --load-certificate x509-c

For icing, we'll create a proxy certificate for the client too.

    $ certtool --generate-privkey > x509-proxy-key.pem
    $ echo 'cn = GnuTLS test client proxy' > proxy.tmpl
    $ certtool --generate-proxy --load-privkey x509-proxy-key.pem   --load-ca-certificate x509-client.pem --load-ca-p

Then start the server again:

    $ gnutls-serv --http   --x509cafile x509-ca.pem   --x509keyfile x509-server-key.pem   --x509cer

Try connecting to the server using your web browser. Note that the server listens to port 5556 by
default.

While you are at it, to allow connections using ECDSA, you can also create a ECDSA key and
certificate for the server. These credentials will be used in the final example below.

    $ certtool --generate-privkey --ecdsa > x509-server-key-ecc.pem
    $ certtool --generate-certificate --load-privkey x509-server-key-ecc.pem   --load-ca-certificate x509-ca.pem --load

The next step is to add support for SRP authentication. This requires an SRP password file created with
srptool. To start the server with SRP support:

    gnutls-serv --http --priority NORMAL:+SRP-RSA:+SRP   --srppasswdconf srp-tpasswd.conf   --srppa

Let's also start a server with support for PSK. This would require a password file created with **psktool**.

    gnutls-serv --http --priority NORMAL:+ECDHE-PSK:+PSK   --pskpasswd psk-passwd.txt

If you want a server with support for raw public-keys we can also add these credentials. Note however
that there is no identity information linked to these keys as is the case with regular x509 certificates.

Authentication must be done via different means. Also we need to explicitly enable raw public-key certificates via the priority strings.

    gnutls-serv --http --priority NORMAL:+CTYPE-CLI-RAWPK:+CTYPE-SRV-RAWPK   --rawpkfile srv.ra

Finally, we start the server with all the earlier parameters and you get this command:

    gnutls-serv --http --priority NORMAL:+PSK:+SRP:+CTYPE-CLI-RAWPK:+CTYPE-SRV-RAWPK   --x5

EXIT STATUS

One of the following exit values will be returned:

0 (EXIT_SUCCESS)
    Successful program execution.

1 (EXIT_FAILURE)
    The operation failed or the command syntax was not valid.

SEE ALSO

gnutls-cli-debug(1), gnutls-cli(1)

AUTHORS
COPYRIGHT

Copyright (C) 2020-2021 Free Software Foundation, and others all rights reserved. This program is released under the terms of the GNU General Public License, version 3 or later

BUGS

Please send bug reports to: bugs@gnutls.org