Taskwarrior – Troubleshooting Taskserver

Your guide to success

Dirk Deimeke

Taskwarrior Academy

May 2016

Content

- **Introduction**
- **Cases**
  - ’task sync’ showed task list
  - Taskwarrior without GnuTLS
  - nodename nor servname provided
  - Could not connect
  - Unable to use port
  - Handshake failed
  - Certificate still valid?
  - GnuTLS outdated?
  - Ancestor not found
- **Debugging**
  - Hostname and IP address
  - Diagnostics
  - Debug Mode
  - Tools
    - Server

Troubleshooting Sync

Here is a list of problems you may encounter, with the most common ones listed first.

The single most common problem has been that the Taskserver Setup Instructions were not properly followed. Please review the steps you took.

It is always a good idea to make sure that you are using the latest release of Taskwarrior and Taskserver, not just because bugs are fixed that may help you, but also because the solutions below are geared toward the current releases.

If you upgrade from an older release of Taskserver, you will need to follow the upgrade instructions.

’task sync’ showed task list

You tried `task sync` but Taskwarrior showed you a task list instead

You have a version of Taskwarrior older than `2.3.0`, which means there was no `sync` command, and you are seeing a list filtered by the search term ’sync’. Upgrading is the only solution.

Taskwarrior without GnuTLS

You tried `task sync` and saw 'Taskwarrior was built without GnuTLS support. Sync is not available.'

You are using version `2.3.0` or later, but the Taskwarrior binary was compiled without GnuTLS support.

If you installed Taskwarrior using your OS's package manager, you may be suffering from an out of date package. Prod your OS's package maintainer for an update.

Recent releases make GnuTLS support opt-out instead of opt-in, so upgrading to the latest version may help. Otherwise, you will need to build Taskwarrior from the latest source tarball, following the instructions in the `INSTALL` file. If you are a developer, do that. If you are not, then installing a development environment is probably not something you want to do, in which case contact your OS's package maintainer.

Continued on next page.

Taskwarrior without GnuTLS – continued

Verify that your Taskwarrior was built with GnuTLS support by running `task diagnostics`:

```
$ task diagnostics | grep libgnutls
libgnutls: 3.3.18
```

nodename nor servname provided

nodename nor servname provided, or not known

Despite the terrible wording, this means the Taskwarrior setting `taskd.server=<host>:<port>` refers to a host name that cannot be seen by Taskwarrior.

- Is it spelled correctly?
- Is the domain correct?
- Is there a valid DNS resolution for the name?
- Is there a firewall between Taskwarrior and Taskserver that is not letting through `<port>` traffic?

Could not connect

Could not connect to <host> <port>

Taskserver may not be running on `<host>`.

Check with `ps -leaf | grep taskd`.

Sometimes the error message is misleading, please check "Handshake failed" as well.

Unable to use port

Unable to use port 53589?

By default, port 53589 is used, but whichever you chose must be open on the server.

If you are unable to open firewall ports, you can use an SSH Tunnel to route port 53589 traffic over port 22:

```
$ ssh -L localport:dsthost:dstport user@example.com
```

Handshake failed

There are many reasons that the TLS handshake can fail.

Please do the following on your client:

```
$ openssl s_client -CAfile .task/ca.cert.pem -host HOST -port PORT
```

The result should be something like `Verify return code: 0 (ok)`.

Handshake failed (continued)

Certificate fails validation, Handshake failed

Could not connect to <host> <port>

When you generated certificates, you modified a `vars` file, in particular the `CN=<name>` setting. That name must match the output of `$ hostname -f` on the server for the certificate to validate.

```
$ certtool -i < server.cert.pem | grep Subject:

$ openssl x509 -noout -in server.cert.pem -subject
```

Handshake failed (continued)

Additionally, that name must also be used in the `taskd.server=<host>:<port>` setting for Taskwarrior. Otherwise you'll need `taskd.trust=ignore hostname`.

If you are using a self-signed certificate, did you specify it using the `taskd.ca` setting?

Setting `taskd.ciphers` can force the use of different ciphers. Use `gnutls-cli --list` to see a list of installed ciphers, and confirm that there is overlap between client and server. There needs to be a cipher that is available to both, otherwise they cannot communicate.

Certificate still valid?

Is your certificate still valid?

Certificates have expiration dates, and if you followed our instructions, you created a certificate that is valid for one year. Check your certificate with this command:

```
$ certtool -i --infile ~/.task/<YOUR NAME>.cert.pem
```

If your certificate has expired, you need a new one. You may also need to regenerate expired server certificates.

Note that creating certificates that never expire is a bad idea. Certificates may be compromised. A certificate that is considered secure today, may not be considered secure in a year. Is the key length you chose something that will remain suitable in the future? Will the algorithms you chose remain secure? For these reasons, choose an expiration date that lets you reevaluate your choices in the relatively near future.

GnuTLS outdated?

Is your GnuTLS library up to date?

As a security product, it is imperative that you keep your GnuTLS up to date.

As with many security products, GnuTLS is maintained by a responsible and quick-responding team that takes security very seriously. Benefit from their diligence by keeping your GnuTLS up to date.

We have received reports of issues with older GnuTLS releases. Specifically, version 2.12.20 has problems validating certificates under certain conditions. Newer releases have addressed memory leaks that were able to take down Taskserver.

Please keep in mind that you have to recompile Taskserver completely to benefit from the new version.

Continued on next page.

GnuTLS outdated? – continued

Upgrading GnuTLS does nothing to upgrade taskd – it has to be rebuilt from scratch, which means:

```
$ cd taskd.git
$ rm CMakeCache.txt
$ cmake -DCMAKE_BUILD_TYPE=release .
$ make
$ sudo make install
```

This can then be verified using `taskd diagnostics`.

Ancestor not found

ERROR: Could not find common ancestor for ...
Client sync key not found.

You skipped the important step of running:

```
$ task sync init
```

This performs an initial upload of your tasks, and sets up a local sync key, which identifies the last sync transaction.

Taskwarrior before Version 2.5.1

Please note that older Taskwarrior versions only sync the **pending** tasks and not all tasks.

Hostname and IP address

No kidding!

There is a difference between *hostname* and *IP address*, honestly. We are referencing the hostname throughout the documentation because you need a valid hostname for the certificate and working name resolution for the client to connect to the server.

So please use the hostname where you are advised to do it!

Debugging – Diagnostics

You may wish to try and debug the problem yourself. You will probably not. But if you do, here is how.

Both Taskwarrior and Taskserver have a `diagnostics` command, the purpose of which is to show you relevant troubleshooting details. Additionally it will indicate problems directly, for example, it will tell you if your cert/key files are not readable. The output from `diagnostics` is intended to be included in bug reports, and doing so saves you a lot of time, because it's the first thing we'll ask for.

```
$ task diagnostics
...
$ taskd diagnostics
...
```

Read the output of the `diagnostics` commands carefully, there may be several types of problems mentioned, which need to be addressed before going further.

Debug Mode

The next step would be to run the server in debug mode. First shutdown your Taskserver, then launch it interactively:

```
$ taskdctl stop
...
$ taskd server
...
```

You can hit `Ctrl-C` to stop this server. For highly verbose output, try this:

```
$ taskd server --debug --debug.tls=2
...
```

Similarly, Taskwarrior has a verbose debug mode, and debug TLS mode:

```
$ task rc.debug=1 rc.debug.tls=2 sync
...
```

Server: IP Address

Show the IP addresses (and network interfaces) on the system.

```
$ ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.100  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 52:54:a2:01:25:9d  txqueuelen 1000  (Ethernet)
        RX packets 42091  bytes 46300226 (44.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17689  bytes 2606921 (2.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 0  (Local Loopback)
        RX packets 120  bytes 43540 (42.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 120  bytes 43540 (42.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

$ ip addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:a2:01:25:9d brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Server: Ports

Shows which process listen on the port, shows the user and – more important – the IP address / interface the server runs on.

```
$ lsof -i TCP:53589 -s TCP:LISTEN # as taskserver user or with sudo
COMMAND  PID       USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
taskd   1167 taskserver    4u  IPv4  16682      0t0  TCP taskd.example.net:53589 (LISTEN)

$ netstat -tl | grep 53589
tcp        0      0 taskd.example.net:53589 0.0.0.0:*               LISTEN
```

Server: Certificate

Shows the hostname from Client: General Reachability after CN=

```
$ openssl x509 -noout -in server.cert.pem -subject
subject= /CN=taskd.example.net/O=My own IT

$ certtool -i --infile=server.cert.pem | grep Subject:
        Subject: CN=taskd.example.net,O=My own IT
```

Server: taskd diagnostics

Shows server diagnostic data.

```
$ taskd diagnostics --data /path/to/taskserver/data/

taskd 1.2.0
    Platform: Linux
    Hostname: taskd.example.net

Compiler
...

Build Features
...

Configuration
   TASKDDATA: /path/to/taskserver/data
        root: /path/to/taskserver/data/, readable, writable
      config: /path/to/taskserver/data//config, readable, writable546 bytes
          CA: /path/to/taskserver/data/ca.cert.pem, readable, 2017 bytes
 Certificate: /path/to/taskserver/data/server.cert.pem, readable, 2004 bytes
         Key: /path/to/taskserver/data/server.key.pem, readable, 10996 bytes
         CRL: /path/to/taskserver/data/server.crl.pem, readable, 1080 bytes
         Log: /path/to/taskserver/data/taskd.log (found)
    PID File: /path/to/taskserver/data/taskd.pid (missing)
      Server: taskd.example.net:53589
 Max Request: 1048576 bytes
     Ciphers:
       Trust: strict
```

Client: General Reachability

Show if the hostname is reachable (if ping is not blocked).

```
$ ping taskd.example.net
PING taskd.example.net (192.168.1.100) 56(84) bytes of data.
64 bytes from taskd.example.net (192.168.1.100): icmp_seq=1 ttl=64 time=0.118 ms
64 bytes from taskd.example.net (192.168.1.100): icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from taskd.example.net (192.168.1.100): icmp_seq=3 ttl=64 time=0.064 ms
64 bytes from taskd.example.net (192.168.1.100): icmp_seq=4 ttl=64 time=0.065 ms
^C
--- taskd.example.net ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.064/0.078/0.118/0.023 ms

$ nmap -sP taskd.example.net

Starting Nmap 6.40 ( http://nmap.org ) at 2016-05-15 11:24 CEST
Nmap scan report for taskd.example.net (192.168.1.100)
Host is up (0.00012s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.01 seconds
```

Client: Port Reachability

Shows if the port is reachable by client (or locally).

```
$ telnet taskd.example.net 53589
Trying 192.168.1.100...
Connected to taskd.example.net.
Escape character is ’^]’.
^CConnection closed by foreign host.

$ sudo nmap -p 53589 taskd.example.net

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-15 11:28 CEST
Nmap scan report for taskd.example.net (192.168.1.100)
Host is up (0.036s latency).
PORT      STATE SERVICE
53589/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds

$ openssl s_client -host taskd.example.net -port 53589
CONNECTED(00000003)
...
```

Client: Hostname resolution

Shows the ip address of hostname. Should match one of the IP addresses found in Server: IP Address and the server should be bound to that address Server: Ports.

```
$ host taskd.example.net
taskd.example.net has address 192.168.1.100

$ nslookup taskd.example.net # checks only nameservers
Server:         192.168.178.1
Address:        192.168.178.1#53

Non-authoritative answer:
Name:   taskd.example.net
Address: 192.168.1.100
```

Client: task diagnostics

Shows client diagnostic data.

```
task diagnostics

task 2.6.0
  Platform: Linux
...
Compiler
...
Build Features
...
Configuration
      File: /home/user/.taskrc (found), 2428 bytes, mode 100664
      Data: /home/user/.task (found), dir, mode 40775
   Locking: Enabled
        GC: Enabled
   $EDITOR: vim
    Server: taskd.example.net:53589
        CA: ~/.task/ca.cert.pem, readable, 3741 bytes
     Trust: strict
Certificate: ~/.task/Dirk_Deimeke.cert.pem, readable, 3724 bytes
       Key: ~/.task/Dirk_Deimeke.key.pem, readable, 25364 bytes
   Ciphers: NORMAL
     Creds: My own IT/Dirk Deimeke/************************************

Hooks
...

Tests
...
```

Getting Help

As a last resort, ask for help. But please make sure you have carefully reviewed your setup, and gone through the checks above before asking. No one wants to lead you through the steps above to discover that you didn’t.

We’ll ask you to provide the `diagnostics` output for both Taskwarrior and Taskserver, then we’re going to go through the steps above, because this is our checklist also.

Getting Help

There are several ways of getting help and there is **no realtime support**, even though answers might come pretty fast:

- Email us at support@taskwarrior.org, then wait patiently for a volunteer to respond.
- Join us on IRC in the #taskwarrior channel on Freenode.net, and get a quicker response from the community, where, as you have anticipated, we will walk you through the checklist above.
- Even though Twitter is no means of support, you can get in touch with @taskwarrior.
- We have a User Mailinglist which you can join anytime to discuss about Taskwarrior and techniques. The Developer Mailinglist is focussing on a more technical oriented audience.
- Submit your details to our Q & A site, then wait patiently for the community to respond.