bitwarden | Help Center Article

SELF-HOSTING

Certificate Options

View in the help center:
https://bitwarden.com/help/certificates/

Certificate Options

This article defines the certificate options available to self-hosted instances of Bitwarden. You will select your certificate option during installation. **Setting up or changing your certificate configuration will always require** you to run the `./bitwarden.sh rebuild` or `.\bitwarden.ps1 -rebuild` command before starting Bitwarden to apply the changes to your config.yml file.

> **Note**
>
> The information in this article may not apply to Bitwarden Unified self-hosted deployments.

Generate a certificate with Let's Encrypt

Let's Encrypt is a certificate authority (CA) that issues trusted SSL certificates free of charge for any domain. The Bitwarden installation script offers the option to generate a trusted SSL certificate for your domain using Let's Encrypt and Certbot.

Certificate renewal checks occur each time Bitwarden is restarted. Using Let's Encrypt will require you to enter an email address for certificate expiration reminders.

> **Warning**
>
> Let's Encrypt is a third-party certificate authority that requires inbound ports 80 and 443 have access from the internet in order to validate your domain and issue a certificate. If you do not have or want to set up inbound internet access, you may use one of the other certificate options in this document.

Manually update a Let's Encrypt certificate

If you change the domain name of your Bitwarden server, you will need to manually update your generated certificate. Run the following commands to create a backup, update your certificate, and rebuild Bitwarden:

Bash

```Bash
./bitwarden.sh stop

mv ./bwdata/letsencrypt ./bwdata/letsencrypt_backup

mkdir ./bwdata/letsencrypt

chown -R bitwarden:bitwarden ./bwdata/letsencrypt

chmod -R 740 ./bwdata/letsencrypt

docker pull certbot/certbot

docker run -i --rm --name certbot -p 443:443 -p 80:80 -v <Full Path from / >/bwdata/letsencrypt:/et
c/letsencrypt/ certbot/certbot certonly --email <user@email.com> --logs-dir /etc/letsencrypt/logs
```

Select 1, then follow the instructions:

```Bash
openssl dhparam -out ./bwdata/letsencrypt/live/<your.domain.com>/dhparam.pem 2048
./bitwarden.sh rebuild
./bitwarden.sh start
```

PowerShell

> **Tip**
> You will need to install a build of OpenSSL for Windows.

```Bash
.\bitwarden.ps1 -stop
mv .\bwdata\letsencrypt .\bwdata\letsencrypt_backup
mkdir .\bwdata\letsencrypt
docker pull certbot/certbot
docker run -i --rm --name certbot -p 443:443 -p 80:80 -v <Full Path from \ >\bwdata\letsencrypt\:/e
tc/letsencrypt/ certbot/certbot certonly --email <user@email.com> --logs-dir /etc/letsencrypt/logs
Select 1, then follow instructions
<path/to/openssl.exe> dhparam -out .\bwdata\letsencrypt\live\<your.domain.com>\dhparam.pem 2048
.\bitwarden.ps1 -rebuild
.\bitwarden.ps1 -start
```

Use an existing SSL certificate

You may alternatively opt to use an existing SSL certificate, which will require you to have the following files:

- A server certificate (`certificate.crt`)
- A private key (`private.key`)
- A CA certificate (`ca.crt`)

You may need to bundle your primary certificate with intermediate CA certificates to prevent SSL trust errors. All certificates should be included in the server certificate file when using a CA certificate. The first certificate in the file should be your server certificate, followed by any intermediate CA certificate(s), followed by the root CA.

Under the default configuration, place your files in `./bwdata/ssl/your.domain`. You may specify a different location for your certificate files by editing the following values in `./bwdata/config.yml`:

```Bash
ssl_certificate_path: <path>
ssl_key_path: <path>
ssl_ca_path: <path>
```

Note

The values defined in `config.yml` represent locations inside the NGINX container. Directories on the host are mapped to directories within the NGINX container. Under the default configuration, mappings line up as follows:

The following values in `config.yml`:

```
Bash

ssl_certificate_path: /etc/ssl/your.domain/certificate.crt
ssl_key_path: /etc/ssl/your.domain/private.key
ssl_ca_path: /etc/ssl/your.domain/ca.crt
```

Map to the following files on the host:

```
Bash

./bwdata/ssl/your.domain/certificate.crt
./bwdata/ssl/your.domain/private.key
./bwdata/ssl/your.domain/ca.crt
```

You should only ever need to work with files in `./bwdata/ssl/`. Working with files directly in the NGINX container is not recommended.

Using Diffie-Hellman key exchange

Optionally, if using Diffie-Hellman key exchange to generate ephemeral parameters:

- Include a `dhparam.pem` file in the same directory.
- Set the `ssl_diffie_hellman_path:` value in `config.yml`.

Note

You can create your own `dhparam.pem` file using OpenSSL with `openssl dhparam -out ./dhparam.pem 2048`.

Using a self-signed Certificate

You may alternatively opt to use a self-signed certificate, however this is only recommended for testing.

Self-signed certificates will not be trusted by Bitwarden client applications by default. You will be required to manually install this certificate to the trusted store of each device you plan to use Bitwarden with.

Generate a self-signed certificate:

```Bash
mkdir ./bwdata/ssl/bitwarden.example.com
openssl req -x509 -newkey rsa:4096 -sha256 -nodes -days 365 \
  -keyout ./bwdata/ssl/bitwarden.example.com/private.key \
  -out ./bwdata/ssl/bitwarden.example.com/certificate.crt \
  -reqexts SAN -extensions SAN \
  -config <(cat /usr/lib/ssl/openssl.cnf <(printf '[SAN]\nsubjectAltName=DNS:bitwarden.example.com
\nbasicConstraints=CA:true')) \
  -subj "/C=US/ST=New York/L=New York/O=Company Name/OU=Bitwarden/CN=bitwarden.example.com"
```

Your self-signed certificate (`.crt`) and private key (`private.key`) can be placed in the `./bwdata/ssl/self/your.domain` directory and configured in the `./bwdata/config.yml`:

```Bash
 ssl_certificate_path: /etc/ssl/bitwarden.example.com/certificate.crt
 ssl_key_path: /etc/ssl/bitwarden.example.com/private.key
```

Trust a self-signed certificate

Windows

To trust a self-signed certificate on Windows, run `certmgr.msc` and import your certificate into the Trusted Root Certification Authorities.

Linux

To trust a self-signed certificate on Linux, add your certificate to the following directories:

```Bash
/usr/local/share/ca-certificates/
/usr/share/ca-certificates/
```

And run the following commands:

```Bash
sudo dpkg-reconfigure ca-certificates
sudo update-ca-certificates
```

For our Linux desktop app, accessing the web vault using Chromium-based browsers, and the Directory Connector desktop app, you also need to complete this Linux cert management procedure.

For the Bitwarden CLI and Directory Connector CLI, your self-signed certificate must be stored in a local file and referenced by a `NODE_EXTRA_CA_CERTS=` environment variable, for example:

```bash
export NODE_EXTRA_CA_CERTS=~/.config/Bitwarden/certificate.crt
```

Android

To trust a self-signed certificate on an Android device, refer to Google's Add & remove certificates documentation.

Note

If you are **not self-hosting** and encounter the following certificate error on your android device:

```bash
Exception message: java.security.cert.CertPathValidatorException: Trust anchor for certificati
on path not found.
```

You will need to upload Bitwarden's certificates to your device. Refer to this community thread for help finding the certificates.

Use no certificate

Warning

If you opt to use no certificate, you **must front your installation with a proxy that serves Bitwarden over SSL**. This is because Bitwarden requires HTTPS; trying to use Bitwarden without the HTTPS protocol will trigger errors.