

Protect your CollabNet TeamForge site

Set up SELinux

If SELinux is active on the machine where your CollabNet TeamForge site is running, modify it to allow the services that TeamForge requires.

This type of configuration is required for a complete TeamForge installation.

1. Enable Apache (running on port 80) to proxy traffic to JBoss (running on port 8080).
 - `setsebool -P httpd_can_network_connect 1`

2. When you are installing TeamForge, follow these steps to enable services to come up when needed.
 - Change the context for the TeamForge log.


```
chcon -R -h -t httpd_sys_content_t
/opt/collabnet/teamforge/log/httpd
```
 - Change the context for PostgreSQL.


```
chcon -R -h -t postgresql_db_t /var/lib/pgsql
```
 - Create the PostgreSQL log directory and set the appropriate permissions.


```
mkdir /opt/collabnet/teamforge/log/pgsql
chown -R postgres:postgres
/opt/collabnet/teamforge/log/pgsql
chmod 777 /opt/collabnet/teamforge/log/pgsql/
```
 - Change the context for the PostgreSQL log.


```
chcon -R -h -t postgresql_db_t
/opt/collabnet/teamforge/log/pgsql/
```

3. After your site is installed and contains some live data, follow these steps to allow source control services to work.
 - Change the context for your Subversion source code service.


```
chcon -R -h -t httpd_sys_content_t /svnroot
```
 - Change the context of the Subversion repository that handles the branding (look and feel) of your site.


```
chcon -R -h -t httpd_sys_content_t /sf-svnroot
chcon -R -h -t httpd_sys_content_t
/opt/collabnet/teamforge/var/overrides
```

- Change the context for your local CVS repository, if you have one.
`chcon -R -h -t httpd_sys_content_t /cvsroot`

Set up Apache for SSL encryption

To force all CollabNet TeamForge traffic to use SSL encryption (HTTPS), state that preference in your configuration file.

1. Back up your existing `/etc/httpd/conf/httpd.conf` file.
2. Update the `/opt/collabnet/teamforge-installer/<release-number>/conf/site-options.conf` file to support SSL.
 - a. Set the value of the `SSL` variable to `on`.
 - b. Set the value of the `SSL_CERT_FILE` variable to the location of the file that contains your site's SSL certificates.
 - `SSL_CERT_FILE=www.example.com.crt`
 - c. Set the value of the `SSL_KEY_FILE` variable to the location of the file that contains your site's RSA private keys.
 - `SSL_KEY_FILE=www.example.com.key`

Important: Select a location for your cert file and your key file that is permanent across restarts. Don't use a temp directory that can be wiped out.

3. Rename the `/etc/httpd/conf.d/ssl.conf` file to `/etc/httpd/conf.d/ssl.conf.old`, if it exists.
4. Recreate the runtime environment.
 - a. `./install.sh -V -r -d <SITE_DIR>`
5. Rename the `/etc/httpd/conf/httpd.conf.cn_new` file to `httpd.conf`.
6. Restart the Apache service.

When you point your browser at CollabNet TeamForge , it should now automatically redirect to HTTPS traffic.

Generate Apache SSL certificates

To use https for web traffic, you will need to obtain a valid Apache SSL certificate.

When generating an Apache (mod_ssl) SSL certificate, you have two options:

- Purchase a SSL certificate from a certificate authority (CA). Searching the Web for "certificate authority" will present several choices.
- Generate a self-signed certificate. This option costs nothing and provides the same level of encryption as a certificate purchased from a certificate authority (CA). However, this option can be a mild annoyance to some users, because Internet Explorer (IE) issues a harmless warning each time a user visits a site that uses a self-signed certificate.

Regardless of which option you select, the process is almost identical.

1. Know the fully qualified domain name (FQDN) of the website for which you want to request a certificate. If you want to access your site through `https://www.example.com`, then the FQDN of your website is `www.example.com`.

Note: This is also known as your common name.

2. Generate the key with the SSL `genrsa` command.
 - `openssl genrsa -out www.example.com.key 1024`

This command generates a 1024 bit RSA private key and stores it in the file [www.example.com.key](#).

Tip: Back up your `www.example.com.key` file, because without this file your SSL certificate will not be valid.

3. Generate the CSR with SSL `req` command.
 - `openssl req -new -key www.example.com.key -out www.example.com.csr`

This command will prompt you for the X.509 attributes of your certificate. Give the fully qualified domain name, such as `www.example.com`, when prompted for **Common Name**.

Note: Do not enter your personal name here. It is requesting a certificate for a webserver, so the Common Name has to match the FQDN of your website.

4. Generate a self-signed certificate.
 - `openssl x509 -req -days 370 -in www.example.com.csr -signkey www.example.com.key -out www.example.com.crt`

This command will generate a self-signed certificate in [www.example.com.crt](#).

You will now have an RSA private key in `www.example.com.key`, a Certificate Signing Request in `www.example.com.csr`, and an SSL certificate in `www.example.com.crt`. The self-signed SSL certificate that you generated will be valid for 370 days.

Prevent HTTPS cracking

To reduce the risk of HTTPS ciphers being cracked, allow only the strongest ciphers available.

Deploying an Apache SSL certificate and forcing https ensures that all data is encrypted. It does not, however, ensure that the encryption methods (also known as ciphers) that are used are strong. With the ever-increasing power of today's computers, many older or weaker ciphers can be cracked in a matter of days or even hours by a determined person with malicious intentions.

1. In the `/etc/httpd/conf.d/ssl.conf` file, find the headings `SSLProtocol` and `SSLCipherSuite`.
Note: If they do not exist, add them below the `SSLEngine` line.
2. In each section, add the following two lines:

```
SSLProtocol all -SSLv2 SSLCipherSuite
RSA: !EXP: !NULL: +HIGH: +MEDIUM: -LOW
```
3. Save the file and restart Apache.
 - `apachectl restart`

Protect integrations with SSL

If you have registered Secure Socket Layer (SSL) certificates, your site's users can use SSL when they set up an SCM integration server.

If you use certificates that are generated in-house, self-signed, or signed by a non-established Certificate Authority, they must be registered with each client system that will connect to the CollabNet TeamForge server. Registration consists of importing custom certificates into the Java runtime's global keystore on each server.

Important: This will affect any other Java applications on the server that use the same Java runtime.

1. Collect server certificates from all servers. On RHEL, CentOS and other RedHat-based distributions, these are contained in `/etc/httpd/conf/ssl.crt/server.crt`.

Tip: Be sure to use exactly this path, as there are other files with similar names, plus server certificates are not really secret, but some other files are. So, files must be copied (e.g., via `scp`) to the same directory, and renamed if necessary to avoid clashes. We recommend that you use the short server name of the corresponding server for this.

2. Locate the Java keystore. This is `PATH_TO_JAVA/jre/lib/security/cacerts`. For example, this may be `/usr/local/j2sdk1.4.2_10/jre/lib/security/cacerts`.

3. Locate the Java **keytool** utility. This is `PATH_TO_JAVA/bin/keytool` For example, `/usr/local/j2sdk1.4.2_10/bin/keytool`.
4. Import each server certificate into the keystore.
 - `PATH_TO_JAVA/bin/keytool -import -keystore PATH_TO_JAVA/jre/lib/security/cacerts -file <server>.cert -alias <server>`

Note: Any value is accepted for server in `-alias <server>`.

5. At the password prompt, use `changeit`. Confirm that you trust the certificate by typing `yes`.
6. Verify that all your certificates are added.
 - `PATH_TO_JAVA/bin/keytool -list -keystore PATH_TO_JAVA/jre/lib/security/cacerts |less`

Note: The list will contain many more certificates. These are top-level CA certificates, provided with Java.

7. Update `/etc/sourceforge.properties` to enable secure communication.
 - Set `sfmain.integration.listener_ssl` to true.
 - Set `sfmain.integration.listener_port` to 443.
8. If you are running more than one separate server, repeat these steps for each server.
9. Restart TeamForge

Now you can check the Use SSL checkbox when creating an SCM integration.