Security - Shared responsibility model for Delft-FEWS system installations

The Delft-FEWS server software historically was most commonly installed on-premise at the customer site on servers that were not directly connected to the internet. Nowadays, there are also more and more Delft-FEWS applications that are being deployed in the cloud. This means security standards and guidelines for the installation of live systems have become more critical than ever before. Delft-FEWS runs on top of a stack of components like 3rd party components: databases, Tomcat and an embedded JRE.

> *It is the primary responsibility of the customer to apply the latest security fixes to the OS, database, Tomcat and all other components.*

For updates for the embedded JRE it is recommended to contact Deltares. The role of Deltares is to supply guidelines and facilitate security best practices where possible. Deltares maintains a separate section on the WIKI especially for system and database administrators. To view these pages, personal credentials can be supplied. These pages contain highly detailed information for installing and upgrading Delft-FEWS, amongst others about security aspects. For the near future it is foreseen that more and more managed services from cloud providers (e.g. Tomcat, database) can be applied. All Delft-FEWS developers are security aware and evaluate the existing and potential vulnerabilities on a regular basis. Together with our colleagues from our ICT department they meet regularly to discuss (potential) improvements for each Delft-FEWS release.

Tomcat

Tomcat is required for the deployment of the Admin Interface, Database HTTPS Proxy, Fews Webservices and the Deltares Open Archive. Tomcat is installed and maintained by the customer organization. Deltares indicates which version of tomcat is compatible with / required for which version of Delft-FEWS. All security related aspects available in Tomcat can be applied and are under the responsibility of the customer organization.

> For Admin Interface clients / proxies that are exposed to the internet it is crucial that the highest stable release version of Tomcat with security fixes is used. This prevents exposure from common vulnerabilities and exposures (CVEs).
>
> For releases up to 2022.02, any tomcat9 version should be able to work for our Admin Interface / HttpProxy / PI Service / ArchiveServer web containers. This requires that the correct Java version matching the indicated JRE version for the Delft-FEWS release version is used and this Java version must be compatible with the Tomcat distribution.
>
> See http://tomcat.apache.org/security-9.html
>
> - Run Tomcat server as an unprivileged user and NOT root / Administrator.
> - Tomcat user has read-only permission to the contents of the `conf/`, `bin/`, and `lib/` directories in `${CATALINA_HOME.}`.
> - Limit the Tomcat user's access and permissions to only the needed directories and files work / temp / webapps / logs.
> - Uninstall all non-essential web applications in the `webapps/` directory, including the applications that come with Tomcat.

JRE/Java

In several components of Delft-FEWS a (stripped down) version of Java/JRE (Java Runtime Environment) is embedded. This JRE folder is a recognizable and standard part of the Delft-FEWS binary package for Operator Clients and Forecasting Shell Servers. This means that Deltares delivers an *optimized* (and minimal) Java Runtime Environment based on Amazon Corretto's series. This so-called base-build can be updated and Deltares will release new base-builds if required. Since the JRE folder is recognizable within the Delft-FEWS binaries, organizations may decide to replace this JRE folder in favour of another (compatible) version of the JRE. It is certainly possible to use a different provider (e.g. Oracle Sun or the openJDK). Replacing the JRE can be done by creating a soft link to the JRE directory or by replacing the JRE folder.

Local databases (Operator Client, Stand Alone)

In recent versions of Delft-FEWS there is no need for a local database (datastore) for an Operator Client (OC) in a client-server environment. Although it is still possible to have a 'fully synchronized' (local) database in an OC or to create a 'replicate' of the central database to continue working as a standalone (SA). There are two data formats available: Derby or Firebird. These are just *local files* (just like any other file on the file system) and they do not require any software installed for managing it. The Delft-FEWS Operator Client or Stand Alone application just reads from and writes to this database format. This mechanism cannot be used as a 'hub' to enter other server side components.

Central Database access

Delft-FEWS can be equipped with one of three common brands of central databases: Oracle, PostgreSQL or MS SQLServer. Access to the central database is required for several Delft-FEWS servers side components. These components are normally located behind the organization's firewall (same network) or in the secure domain of a data centre or cloud provider. Operator client access to this database is also required, but when set up from 'outside' the organization's network, a https (proxy) server (including IP whitelisting) should be in between. Deltares can provide this.

Forecasting Shells

1. The Delft-FEWS binaries folder should be made read-only.
2. Forecasting Shell Servers (FSS) should have limited permissions (rights). Only write access within their own directory.
3. Only provide access to the data feed shared folders for FSSs.
4. The account for *installing* should be different than the account *running* processes
5. When applying external simulation software, ensure the executables and other libraries have only permission to be run locally.

Operator clients

1. The Delft-FEWS binaries folder should be made read-only.
2. When using the optional JCEF browser, white-listing is used to grant access to webpages.

Multi-layered security approach

- The inner layer is the central database (and optionally Deltares Open Archive).
- The middle layer are Delft-FEWS components that communicate directly with the database using encryption.
- The third layer (optional) is a reverse proxy to the database that can be accessed externally.
- The outer layer is the bastion host (optional).