ENVEIL

ENCRYPTED VEIL

ZeroReveal® Compute Fabric Configuration Guide for Common Criteria v3.1

Version 4.6.3

8171 Maple Lawn Drive, Suite 240, Fulton, MD 20759 USA | (443) 203-9413

EV

ENVEIL
ENCRYPTED VEIL

Contents:

E V

1. Preface

This guide explains how to configure ZeroReveal Compute Fabric components to be compliant with Common Criteria for Information Technology Security Evaluation version 3.1.

This guide is provided as a supplement to the ZeroReveal Compute Fabric manual, which is included with all installations of ZeroReveal Compute Fabric components. In order to put ZeroReveal Compute Fabric components into compliance with the Common Criteria, the instructions in this guide must be followed.

The Enveil ZeroReveal Compute Fabric v4.6.3 security features have been evaluated against the Common Criteria Evaluation and Validation Scheme (CCEVS). The evaluation demonstrates that the Enveil ZeroReveal Compute Fabric v4.6.3 conforms to the security requirements specified in Protection Profile for Application Software v1.4 when installed and operated in accordance with this document.

The evaluated configuration also includes several assumptions and requirements that must be met by the intended environment in order for the installed ZeroReveal Client v4.6.3 and/or ZeroReveal Server v4.6.3 to be in the evaluated configuration. These are as follows:

- The application software relies upon a trustworthy computing platform for its execution. This includes the underlying operating system and any discrete execution environment provided to the application software.
- The user of the application software is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy.
- The administrator of the application software is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

1.1 Targets of Evaluation and Scope

The Targets of Evaluation (TOE) are the Enveil ZeroReveal Compute Fabric Client software applications which communicates to one or more instances of the Enveil ZeroReveal Compute Fabric Server software application via REST API over mutually authenticated HTTPS over TLS. The TOE is a homomorphic encryption engine for database queries. In normal database operation, a query is submitted in plain text, and a plain text answer retrieved for the querier. Moreover, communication between the querier and the database engine itself may be transmitted through a tunnel such as IPsec, TLS, or SSH, the contents of the

query are always in plaintext. The ZeroReveal Compute Fabric Client takes an authenticated user's database query and encrypts it using Enveil's proprietary homomorphic encryption process. This encrypted query is passed via a mutually authenticated TLS trusted channel from ZeroReveal Client to ZeroReveal Server. The encrypted query is never decrypted during this process, which prevents ZeroReveal Server and its owners/administrators from being able to tell what the query was searching for and what items in the database (if any) matched the query. The output of this process is an encrypted response that is sent back to ZeroReveal Client. In this way, the database itself is not strictly aware of what the query was and no individual point in the chain between the user and the information know what was requested.

The ZeroReveal Client and ZeroReveal Server are evaluated as software applications only. For emphasis, the following are outside of the scope of this evaluation:

- The homomorphic encryption techniques used for the ZeroReveal Client and ZeroReveal Server operations.
- The interface used to modify the ZeroReveal Client and ZeroReveal Server configuration files.

For both ZeroReveal Client and ZeroReveal Server an administrator (not necessarily the same person) manages the TOE via the configuration files, there are no management interfaces other than that.

2. ZeroReveal Server Guide

This section explains how to configure ZeroReveal Server to be compliant with the Common Criteria.

2.1 System Requirements

The host must meet the following requirements in order for ZeroReveal Server to support Common Criteria compliance:

- 4 virtual CPUs (vCPUs) or more
- 16GB of RAM
- 100GB of free disk space
- Operating system: Rocky version 8.7 with SELinux
- x86_64 architecture: Intel Core i7-10710U
- The Amazon Corretto Java 8 Runtime Environment

ZeroReveal Server uses the network interface to communicate with connected ZeroReveal Clients and/or Data Sources.

2.2 Prerequisites

The following steps must be followed *before* installing ZeroReveal Server on the host machine. If these steps are not followed prior to installing ZeroReveal Server, Common Criteria compliance is not enabled, even if the steps are followed later:

1. Ensure that the OS is fully updated by running `yum update`, then reboot the system. Performing this step will protect against known and patched security vulnerabilities including CVEs CVE-2017-5753, CVE-2017-5715, and CVE-2017-5754 (Meltdown and Spectre).

2. Ensure the SELinux is enabled and in “enforcing mode”. As the root user, run the command:

```
bash$ sestatus
```

and ensure that the “SELinux status” field reads “enabled” and the “Current mode” field reads “enforcing”. If not, edit the file `/etc/sysconfig/selinux`, ensuring the line `SELINUX=enforcing` is present.

3. Ensure that full disk encryption has been enabled. See this guide for instructions on configuring disk encryption.

4. Ensure that all installed packages have been updated by running this command as the root user:

```
bash$ yum update
```

5. Ensure that ASLR has not been disabled by checking that the `randomize_va_space` option is set to `2` by using `cat`:

```
bash$ cat /proc/sys/kernel/randomize_va_space
2
```

If either `0` or `1` is returned instead of `2` set it to `2` by running:

```
bash$ echo 2 | sudo tee /proc/sys/kernel/randomize_va_space
```

and then reboot the system. Stack protection is already enabled for ZeroReveal Compute Fabric components and cannot be disabled.

6. Ensure that deprecated certificate signature hash algorithms are disabled in the Java Runtime’s `java.security` file. For OpenJDK Corretto 8 the file is located at `/usr/lib/jvm/java-1.8.0-amazon-corretto/jre/lib/security/java.security`. The property `jdk.certpath.disabledAlgorithms` must be set exactly as follows:

```
jdk.certpath.disabledAlgorithms=MD2, MD5, SHA1, \
   RSA keySize < 1024, DSA keySize < 1024, EC keySize < 224
```

7. Ensure the `firewalld` service is running and enabled to run at every boot:

```
bash$ systemctl start firewalld
bash$ systemctl enable firewalld
```

8. Open a port in the firewall for the ZeroReveal Client to reach the ZeroReveal Server’s TLS-protected

REST API:

```
bash$ firewall-cmd --zone=public --permanent --add-port 18443/tcp
bash$ firewall-cmd --reload
```

2.3 Installing ZeroReveal Server

Run the following commands to obtain repository access and trust the Enveil GPG key, filling in your token: Note: The server and search tokens are not the same.

```
bash$ curl -s https://<server_token>:@repository.enveil.com/server/script.rpm.sh | sudo bash
bash$ curl -s https://<search_token>:@repository.enveil.com/search/script.rpm.sh | sudo bash
bash$ sudo rpm --import https://www.enveil.com/s/EnveilPackageGpgKey.asc
```

You can then install ZeroReveal Server and ZeroReveal Search Plugin on the current host using `yum`. Run:

```
bash$ sudo yum install enveil-zeroreveal-server
bash$ sudo yum install enveil-zeroreveal-server-plugin-search
```

If you need to download the RPMs for installation on another machine that cannot access `repository.enveil.com`, first install the `yum-utils` package:

```
bash$ sudo yum install yum-utils
```

Then use the `yumdownloader` tool to download the RPMs:

```
bash$ sudo yumdownloader enveil-zeroreveal-server
bash$ sudo yumdownloader enveil-zeroreveal-server-plugin-search
```

This will save an RPMs containing ZeroReveal Server and ZeroReveal Search Plugin in your local directory. You can then transfer those RPMs to a different machine and install them with:

```
bash$ sudo rpm -i <rpm_filename>.rpm
```

Official Enveil RPMs are signed using Enveil's private signing key. When using `yum` to install Enveil ZeroReveal Compute Fabric packages, the GPG signatures on the RPM files will automatically be checked. If they are missing a signature or signed with the wrong GPG key then an error indicating that the GPG keys for the repository do not match the package will be displayed and the install will automatically abort. These checks are also run during the installation of every update.

The signature on the ZeroReveal Server installation package can be verified using this command:

```
bash$ rpm -K enveil-zeroreveal-server
```

2.4 Configuring ZeroReveal Server

Most ZeroReveal Server settings are contained in the server's configuration file which defaults to `server.conf`. The `server.conf` file is a HOCON configuration file in which a single option is specified on each line with its value placed after an equals sign. An example of this format can be found in the *Appendix*.

2.4.1 Obtaining a TLS certificate/key pair

Obtain a TLS certificate for ZeroReveal Server and copy them onto the ZeroReveal Server host machine. The TLS server certificates for ZeroReveal Server must have the Digital Signature Key Usage and the TLS server Extended Key Usage.

For the configuration to be Common Criteria compliant, the key stores must be in the `bcfks` format. See *Creating a bcfks from a p12* for instructions on converting key stores to the `bcfks` format. Furthermore, certificates must contain the hostname or the IP address of the ZeroReveal Server's host machine as a Subject Alternative Name. To generate a TLS keypair in a Common Criteria compliant fashion using ZeroReveal Server, consult *Using ZeroReveal to Generate Certificate Signing Requests*. The following are relevant settings in `server.conf`:

enveil.security.tls.keystore.path
: Path to the key store on ZeroReveal Server's local disk.

enveil.security.tls.keystore.type
: Type of the key store (possible options are `jks`, `pkcs12`, or `bcfks`).

enveil.security.tls.keystore.password
: The key store's password.

enveil.security.tls.truststore.path
: Path to the trust store on ZeroReveal Server's local disk.

enveil.security.tls.truststore.type
: Type of the trust store (possible options are `jks` or `bcfks`).

enveil.security.tls.truststore.password

The trust store's password.

If the certificate keys are generated using Elliptic Curve Cryptography, ensure that the curve used is either `secp256r1` or `secp384r1`. If RSA keys are used, they must be 2048, 3072 bits, or 4096 bits.

Ensure that all TLS key stores and TLS trust stores are stored in `etc/enveil/zeroreveal-server/certs/` and are readable only by the `enveil` user.

Make sure that `server.conf` is configured with the following constraints:

enveil.common.niap.enforce

(boolean) Enforces that the server is configured to meet the NIAP requirements.

Must be set to `true`.

enveil.security.tls.conscrypt.aes.enabled

(boolean) `true` enables the use of native AES ciphers from a bundled BoringSSL implementation. `false` will disable the native ciphers and use default Java implementation.

Must be set to `false`.

enveil.security.tls.keystore.check

(boolean) Validates the key store on startup.

Must be set to `true`.

enveil.security.tls.strict

(string) If true, requires TLSv1.2 and one of the following cipher suites for all connections: `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`, `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`. (Note that the elliptic curve used with these cipher suites for key establishment is only `secp384r1`.) If false, accepts any valid TLS protocol and cipher suite available in the local Java installation.

Must be set to `true`.

enveil.security.tls.client.certificate.check

(boolean) Whether to check the validity of a certificate presented by any TLS client (currently only ZeroReveal Client).

Must be set to `true`.

enveil.security.random.blockingDevice

(boolean) Whether to use a blocking device for random number generation. That is, wait for

enough entropy to be available before generating random numbers.

Must be set to `true`.

enveil.security.tls.niap.signature.algorithms
(boolean) Only used NIAP-approved signature algorithms

Must be set to `true`.

enveil.server.authFile
(path) Path to the file containing the list of clients authorized to connect to ZeroReveal Server. For the format of this file see *ZeroReveal Server Authorized Clients File*.

Must be set to "`/etc/enveil/zeroreveal-server/authorized_clients.json`".

enveil.server.data.dir
(path) The directory in which ZeroReveal Server should store its data.

Must be set to "`/var/lib/enveil/zeroreveal-server/storage`".

enveil.server.result.dir
(path) The directory in which ZeroReveal Server should store encrypted results.

Must be set to "`/var/lib/enveil/zeroreveal-server/storage/results`".

enveil.server.data.source.dir
(path) The directory from which ZeroReveal Server will read its Data Source XML files.

Must be set to "`/etc/enveil/zeroreveal-server/dataSources`".

Configure the `bootstrap.conf` for Spark with the following constraints:

- `execution_mode` must be `full`.
- `spark_mode` must be `spark-standalone`. (This will include the additional configuration file: `spark-standalone.conf`.)
- Do not change `lib_directory`, `server_properties`, `log_config`, or `error_log_file`.

Administrators may elect to configure certificate validity checking:

enveil.security.cert.revocation.check.mode
Whether to check for certificate revocation using any provided CRL endpoint. Defaults to `NONE`.

Must be set to "HARD_FAIL".

ZeroReveal Server automatically restricts all TLS connections to TLS version 1.2, denying all other TLS versions. No further configuration is required to configure the cryptographic engine beyond the steps above in this guide. *Note:* ZeroReveal Server only accepts connections with mutual TLS.

2.4.2 Configuring Connections to Data Sources

ZeroReveal Server is able to run queries over data provided by a variety of inputs. A Data Source XML file describes a data source and its format to ZeroReveal Server so that it can locate the data, interpret its format, and set options specific to that Data Source. ZeroReveal Server does not provide access to any databases or information repositories other than those it has explicitly been configured to connect to by the administrator to respond to ZeroReveal Queries.

Data Source XML files should be placed in the directory specified by the setting `enveil.server.data.source.dir`. When configuring Data Source XML files, ensure that they are only readable by the `enveil` user.

2.5 Determining the Installed Version of ZeroReveal Server

To examine which version of ZeroReveal Server is installed run the following command:

```
bash$ yum info enveil-zeroreveal-server
```

2.6 Updating ZeroReveal Server

ZeroReveal Server will never check for updates or update it own code, all updates must be initiated by the admin through the package manager, by running the following command:

```
bash$ yum update enveil-zeroreveal-server
```

This will display whether or not an update is available and if so, ask whether to apply the update. ZeroReveal Server may only be updated using the package manager.

The package manager will automatically reject any update that is either not signed or signed with the wrong key.

2.7 Uninstalling ZeroReveal Server

To uninstall ZeroReveal Server, run the following command:

```
bash$ yum remove enveil-zeroreveal-server
```

2.8 Troubleshooting ZeroReveal Server

ZeroReveal Server logs detailed information about standard operation and issues to `/var/log/enveil/zeroreveal-server/server.log` and `/var/log/enveil/zeroreveal-server/stacks.log`.

2.9 Compliant TLS Client Connection with MySQL Server

ZeroReveal Server can establish a Common Criteria compliant TLS client connection to a MySQL server with the following datasource configuration. Notice the presence of `permitMysqlScheme` in the JDBC URL. `sslMode=verify-full` is required to use SSL/TLS for encryption, certificate verification and hostname verification. Additional NIAP requirements are met by specifying `tlsSocketType=EnveilTlsSocketPlugin`. The keystore, key store password, and key store type that ZeroReveal Server uses to connect as a TLS client can be specified through the values in the JDBC URL as demonstrated below. **All three** of these values must be specified in the URL. In this document, related information can be found here: *Configuring ZeroReveal Server*. **Note:** To not confuse the reader, curly braces are used to demarcate variables in lieu of angle brackets due to XML's use of angle brackets.

```
<evds:dataSource xmlns:evds="https://schemas.enveil.com/v10/datasource.xsd">
  <dataSourceName>{dataSourceName}</dataSourceName>
  <dataInputFactoryClass>JdbcInputFactory</dataInputFactoryClass>
  <dataInputFactoryArguments>
    <argument>
      <key>jdbc.driver</key>
      <value>org.mariadb.jdbc.Driver</value>
    </argument>
    <argument>
      <key>jdbc.url</key>
      <value>jdbc:mysql://{hostname}:{port}/{database}?permitMysqlScheme&amp;sslMode=verify-
↪full&amp;keyStorePassword={password}&amp;keyStore={path}&amp;keyStoreType={type}&amp;
↪tlsSocketType=EnveilTlsSocketPlugin</value>
    </argument>
        ...
```

(continues on next page)

(continued from previous page)

```
  </dataInputFactoryArguments>

  <dataSchema>
      ...
  </dataSchema>
</evds:dataSource>
```

3. ZeroReveal Client Guide

This section explains how to configure ZeroReveal Client to be compliant with the Common Criteria.

3.1 System Requirements

The host must meet the following requirements in order for ZeroReveal Client to support Common Criteria compliance:

- 4 virtual CPUs (vCPUs) or more
- 8GB of RAM
- 100GB of free disk space
- Operating system: Rocky version 8.7 with SELinux
- `x86_64` architecture: Intel Core i7-10710U
- The Amazon Corretto Java 8 Runtime Environment, patch version >= 201

ZeroReveal Client uses the network interface to communicate with users and any ZeroReveal Servers configured with an HTTP connection. Additionally, ZeroReveal Client will use a network connection to the configured LDAP service and to validate certificates if they contain CRL endpoints.

3.2 Prerequisites

The following steps must be followed *before* installing ZeroReveal Client on the host machine. If these steps are not followed prior to installing ZeroReveal Client, Common Criteria compliance is not enabled, even if the steps are followed later:

1. Ensure that the OS is fully updated by running `yum update`, then reboot the system. Performing this step will protect against known and patched security vulnerabilities including CVEs CVE-2017-5753, CVE-2017-5715, and CVE-2017-5754 (Meltdown and Spectre).
2. Ensure the SELinux is enabled and in “enforcing” mode. As the root user, run the command:

```
bash$ sestatus
```

and ensure that the “SELinux status” field reads “enabled” and the “Current mode” field reads “enforcing”. If not, edit the file `/etc/sysconfig/selinux`, ensuring the line `SELINUX=enforcing` is present.

3. Ensure that full disk encryption has been enabled. See this guide for instructions on configuring disk encryption.

4. Ensure that all installed packages have been updated by running this command as the root user:

```
bash$ yum update
```

5. Ensure that ASLR has not been disabled by checking that the `randomize_va_space` option is set to `2` by using `cat`:

```
bash$ cat /proc/sys/kernel/randomize_va_space
2
```

If either `0` or `1` is returned instead of `2` set it to `2` by running:

```
bash$ echo 2 | sudo tee /proc/sys/kernel/randomize_va_space
```

and then reboot the system. Stack protection is already enabled for ZeroReveal Compute Fabric components and cannot be disabled.

6. Ensure that deprecated certificate signature hash algorithms are disabled in the Java Runtime’s `java.security` file. For OpenJDK Corretto 8 the file is located at `/usr/lib/jvm/java-1.8.0-amazon-corretto/jre/lib/security/java.security`. The property `jdk.certpath.disabledAlgorithms` must be set exactly as follows:

```
jdk.certpath.disabledAlgorithms=MD2, MD5, SHA1, \
    RSA keySize < 1024, DSA keySize < 1024, EC keySize < 224
```

7. Ensure the `firewalld` service is running and enabled to run at every boot:

```
bash$ systemctl start firewalld
bash$ systemctl enable firewalld
```

8. Open a port in the firewall for users of the ZeroReveal Client to reach the ZeroReveal Client’s

TLS-protected REST API:

```
bash$ firewall-cmd --zone=public --permanent --add-port 17443/tcp
bash$ firewall-cmd --reload
```

3.3 Installing ZeroReveal Client

Run the following commands to obtain repository access and trust the Enveil GPG key, filling in your token: Note: The client and search tokens are not the same.

```
bash$ curl -s https://<client_token>:@repository.enveil.com/client/script.rpm.sh | sudo bash
bash$ curl -s https://<search_token>:@repository.enveil.com/search/script.rpm.sh | sudo bash
bash$ sudo rpm --import https://www.enveil.com/s/EnveilPackageGpgKey.asc
```

You can then install ZeroReveal Client and ZeroReveal Search Plugin on the current host using `yum`. Run:

```
bash$ sudo yum install enveil-zeroreveal-client
bash$ sudo yum install enveil-zeroreveal-client-plugin-search
```

If you need to download the RPMs for installation on another machine that cannot access `repository.enveil.com`, first install the `yum-utils` package:

```
bash$ sudo yum install yum-utils
```

Then use the `yumdownloader` tool to download the RPMs:

```
bash$ sudo yumdownloader enveil-zeroreveal-client
bash$ sudo yumdownloader enveil-zeroreveal-client-plugin-search
```

This will save an RPM containing ZeroReveal Client and ZeroReveal Search Plugin in your local directory. You can then transfer those RPMs to a different machine and install them with:

```
bash$ sudo rpm -i <rpm_filename>.rpm
```

Official Enveil RPMs are signed using Enveil's private signing key. When using `yum` to install Enveil ZeroReveal Compute Fabric packages, the GPG signatures on the RPM files will automatically be checked. If they are missing a signature or signed with the wrong GPG key then an error indicating that the GPG keys for the repository do not match the package will be displayed and the install will automatically abort. These checks are also run during the installation of every update.

The signature on the ZeroReveal Client installation package can be verified using this command:

```
bash$ rpm -K enveil-zeroreveal-client
```

3.4 Configuring ZeroReveal Client

ZeroReveal Client requires three TLS Certificates:

- A TLS server certificate to authenticate incoming connections to web-facing services hosted by the ZeroReveal Client. These services can be accessed via a web browser or a REST API client.
- A TLS client certificate to authenticate outgoing connections to the the LDAP Service that must be signed using an an algorithm using `SHA384` or `SHA512` (and **not** `SHA256`)
- A TLS client certificate to authenticate outgoing connections to the ZeroReveal Server that must be signed using an algorithm using `SHA384` or `SHA512` (and **not** `SHA256`)

For the configuration to be Common Criteria compliant, the key stores must be in the `bcfks` format. See *Creating a bcfks from a p12* for instructions on converting key stores to the `bcfks` format. Furthermore, certificates must contain the hostname or the IP address of the ZeroReveal Client's host machine as a Subject Alternative Name. See *Mutual TLS Configuration* for other certificate requirements. To generate a TLS keypair in a Common Criteria compliant fashion using ZeroReveal Client, consult *Using ZeroReveal to Generate Certificate Signing Requests*. The following are relevant settings in `client.conf`. The first two certificates mentioned above are configured in the `client.conf` file in a single key store using the properties below. The trust store needs to contains certificates that are trusted by the ZeroReveal Client. The trust store is used to validate the certificates of the ZeroReveal Server and the LDAP server. The last certificate above is used here: *Connecting ZeroReveal Client and ZeroReveal Server*.

enveil.security.tls.keystore.path
: Path to the key store on ZeroReveal Client's local disk.

enveil.security.tls.keystore.type
: Type of the key store (possible options are `jks`, `pkcs12`, or `bcfks`).

enveil.security.tls.keystore.password
: The key store's password.

enveil.security.tls.truststore.path
: Path to the trust store on ZeroReveal Client's local disk.

enveil.security.tls.truststore.type
Type of the trust store (possible options are `jks` or `bcfks`).

enveil.security.tls.truststore.password
The trust store's password.

If the certificate keys are generated using Elliptic Curve Cryptography, ensure that the curve used is either `secp256r1` or `secp384r1`. If RSA keys are used, they must be 2048, 3072 bits, or 4096 bits.

Ensure that all TLS key stores and TLS trust stores are stored in `/etc/enveil/zeroreveal-client/certs/` and are readable only by the `enveil` user.

Make sure that the following Common Criteria specific options in `client.conf` are set with the following constraints:

enveil.common.niap.enforce
(boolean) Enforces that the server is configured to meet the NIAP requirements.

Must be set to `true`.

enveil.client.auth.mechanisms
Comma-separated list of authentication mechanisms to use.

Must be set to `[certificate]`.

enveil.client.auth.require.user.cert
(boolean) Whether to require users to present valid TLS client certificates.

Must be set to `true`

enveil.client.auth.certificate.user.source.mechanisms
(string) A comma-separated list of user stores for use with certificate authentication.

Must be set to `[ldap]`.

enveil.client.auth.certificate.ldap.ssl.enabled
(boolean) Whether to connect to the LDAP server under TLS for certificate enveil.client.auth.

Must be set to `true`.

enveil.client.auth.certificate.ldap.connect.with.sasl.external
(boolean) Whether to authenticate to the LDAP server using a TLS client certificate or not for

certificate auth ; see *Certificate*.

Must be set to `true`.

enveil.client.gateway.specification.dir

(path) Path to a directory containing specifications for ZeroReveal Servers to connect to. Each ZeroReveal Server is represented by a separate properties file. See *Adding a Gateway Specification File* for configuration instructions.

Must be set to `/etc/enveil/zeroreveal-client/gateways`.

enveil.security.tls.conscrypt.aes.enabled

(boolean) `true` enables the use of native AES ciphers from a bundled BoringSSL implementation. `false` will disable the native ciphers and use default Java implementation.

Must be set to `false`.

enveil.security.tls.keystore.check

(boolean) Validates the key store on startup.

Must be set to `true`.

enveil.security.tls.strict

(string) If true, requires TLSv1.2 and one of the following cipher suites for all connections: `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`, `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`. (Note that the elliptic curve used with these cipher suites for key establishment is only `secp384r1`.) If false, accepts any valid TLS protocol and cipher suite available in the local Java installation.

Must be set to `true`.

enveil.security.tls.client.certificate.check

(boolean) Whether to check the validity of a certificate presented by any TLS client (currently only ZeroReveal Client).

Must be set to `true`.

enveil.security.random.blockingDevice

(boolean) Whether to use a blocking device for random number generation. That is, wait for enough entropy to be available before generating random numbers.

Must be set to `true`.

enveil.security.tls.niap.signature.algorithms

(boolean) Only used NIAP-approved signature algorithms

Must be set to `true`.

Administrators may elect to configure certificate validity checking:

enveil.security.cert.revocation.check.mode

(string) Whether to check for certificate revocation using any provided CRL endpoint. Defaults to NONE.

Must be set to "HARD_FAIL".

ZeroReveal Client automatically restricts all TLS connections to TLS version 1.2, denying all other TLS versions. No further configuration is required to configure the cryptographic engine beyond the steps above in this guide. *Note:* The combination of the settings for **enveil.client.auth.mechanisms** and **enveil.client.auth.require.user.cert** ensures that ZeroReveal Client will only accept connections with mutual TLS.

ZeroReveal Client does not provide any access to sensitive information repositories, only the data sources backing the connected ZeroReveal Servers.

3.5 Determining the Installed Version of ZeroReveal Client

To examine which version of ZeroReveal Client is installed run the following command:

```
bash$ yum info enveil-zeroreveal-client
```

3.6 Updating ZeroReveal Client

ZeroReveal Client will never check for updates or update it own code, all updates must be initiated by the admin through the package manager, by running the following command:

```
bash$ yum update enveil-zeroreveal-client
```

This will display whether or not an update is available and if so, ask whether to apply the update. ZeroReveal Client may only be updated using the package manager.

The package manager will automatically reject any update that is either not signed or signed with the wrong key

3.7 Uninstalling ZeroReveal Client

To uninstall ZeroReveal Client, run the following command:

```
bash$ yum remove enveil-zeroreveal-client
```

3.8 Troubleshooting ZeroReveal Client

ZeroReveal Client logs detailed information about standard operation and issues to `/var/log/enveil/zeroreveal-client/client.log` and `/var/log/enveil/zeroreveal-client/stacks.log`.

4. Mutual TLS Configuration

The following sections explain how to configure ZeroReveal Server and ZeroReveal Client to use mutual TLS. Commonalities can be found in the *Certificate Requirements* section. The main difference is that ZeroReveal Client uses an LDAP server and ZeroReveal Server uses an authorized clients file.

4.1 Certificate Requirements

This section details the requirements that a TLS certificate must fulfill in order to be accepted by ZeroReveal Client and/or ZeroReveal Server.

The following requirements must be satisfied by all certificates, regardless of how they are used:

- All certificates must use the X.509v3 format.
- All certificate paths must terminate with a trusted CA certificate, per RFC5280.
- All certificates must be signed by and contain only RSA and/or Elliptic Curve (EC) keys.
  - If EC keys are used, they must be created using the NIST P-256 or P-384 curves.
  - NIST P-256 is often known to software as `secp256r1` but OpenSSL knows it as `prime256v1`.
  - NIST P-384 is often known to software (including OpenSSL) as `secp384r1`.
- The `notBefore` and `notAfter` dates included in the certificate must be before and after the current time, respectively.
- Any revocation checks specified by the certificate must pass. Online Certificate Status Protocol (OCSP) checking as specified in RFC 2560, Certificate Revocation List checking as specified in RFC 5759, and RFC 5280 Section 6.3 revocation checking will be attempted by ZeroReveal Client and ZeroReveal Server on certificates that have listed endpoints. The configuration setting `enveil.security.cert.revocation.check.mode` controls how ZeroReveal Client and ZeroReveal Server will respond to a non-successful check. ZeroReveal Client and ZeroReveal Server do not support OCSP stapling.
- Certificates used to sign OCSP responses must have the “OCSP Signing” Extended Key Usage (EKU).

The following sections describe requirements that are specific to certificates used as CA certificates, TLS server certificates, and TLS client certificates.

4.1.1 CA Certificates

ZeroReveal Client and ZeroReveal Server will treat a certificate as a CA certificate only if the X509v3 Basic Constraint "CA: TRUE" is present. Also, key stores containing CA certificates must use the `jks` format.

4.1.2 TLS Server Certificates

ZeroReveal Client and ZeroReveal Server will require all TLS server certificates to have a "Digital Signature" Key Usage (KU) and "TLS Web Server Authentication" EKU.

4.1.3 TLS Client Certificates

ZeroReveal Client and ZeroReveal Server will require all TLS client certificates to have a "Digital Signature" Key Usage (KU) and "TLS Web Client Authentication" EKU.

4.2 A Note On Certificates

What follows is a brief explanation of the role of certificates in ZeroReveal Compute Fabric components. This is not meant to be a comprehensive explanation of certificates or how they are used with respect to TLS v1.2. Follow the aforementioned link for a more detailed explanation of TLS v1.2 and some of the terms below.

4.2.1 Signature Algorithms as a TLS Client

In order to follow Common Criteria, a requirement exists that demands any `ClientHello` that originates from ZeroReveal Client or ZeroReveal Server presents a restricted set of `signature_algorithms` in the `supported_signature_algorithms` extension. Algorithms using `SHA256` are no longer allowed, which means that only algorithms using `SHA384` or `SHA512` may be used. These extensions are restricted by setting a system property (`jdk.tls.client.SignatureSchemes`). An example of an allowable value contained in this property is `ecdsa_secp384r1_sha384` or `rsa_pss_rsae_sha512`. An example of one that is **not** allowed is `rsa_pss_rsae_sha256`. **This is different from the requirement for a TLS server, which is explained in the next section.**

How does this requirement apply to ZeroReveal?

ZeroReveal Client establishes TLS connections (as a client) with ZeroReveal Server and an LDAP server (when configured). Therefore, the certificates that ZeroReveal Client presents to these servers must be signed using one of algorithms with the allowed hashing algorithms (SHA384 or SHA512). While it may seem that this does not place restrictions on ZeroReveal Server or an LDAP server, in actuality, the certificates that each of these servers present to ZeroReveal Client must be signed using an algorithm with an allowed hashing algorithm. Moreover, this applies to any TLS client connection that ZeroReveal Server makes as well. This can occur if a ZeroReveal Server connects to a database using TLS. ZeroReveal Server can establish a TLS client connection with a MySQL Server that is configured via a datasource XML file (see *Compliant TLS Client Connection with MySQL Server*).

4.2.2 Signature Algorithms as a TLS Server

In order to follow Common Criteria, a requirement exists that demands any `CertificateRequest` that originates from ZeroReveal (when acting as a TLS server) presents a restricted set of `signature_algorithms` in the `supported_signature_algorithms` extension. Only algorithms using SHA256 or SHA384 may be used. These extensions are restricted by setting a system property (`jdk.tls.server.SignatureSchemes`). An example of an allowable value contained in this property is `rsa_pss_rsae_sha256` or `ecdsa_secp384r1_sha384`. An example of one that is **not** allowed is `rsa_pss_rsae_sha512`. **This is different from the requirement for a TLS client, which is explained in the previous section.**

How does this requirement apply to ZeroReveal? This should create no issues, which directly contrasts the above restriction for a TLS client. The reason for that is **usually** the default hashing algorithm used in signing is SHA256. However, ZeroReveal Client does interact with ZeroReveal Server with the former acting as a TLS client and the latter acting as a TLS server. Therefore, the certificate that ZeroReveal Client uses as a TLS client needs to be signed with SHA384 and the certificate that ZeroReveal Server uses as a TLS server needs to be signed with SHA384. Note that this is the same conclusion reached in the above requirement. For completeness, ZeroReveal Client behaves as a TLS server when a user connects to it, for example, using a web browser. ZeroReveal Server behaves as a TLS server when ZeroReveal Client connects to it.

4.2.3 Concluding Remark

Please be aware of the above requirements if ZeroReveal Client and ZeroReveal Server are operating according to the Common Criteria.

4.3 ZeroReveal Client LDAP Configuration

In addition to the settings in *Configuring ZeroReveal Client*, the following must occur as well:

For a Common Criteria compliant configuration, ZeroReveal Client can use an LDAP server to authenticate users. The LDAP server must be configured to use TLS and the ZeroReveal Client must contain a TLS client certificate/key pair whose certificate is trusted by the LDAP server. The certificate that a user presents to the ZeroReveal Client must be trusted by the ZeroReveal Client. Moreover, the subject alternative name on the user's certificate can be forwarded to the LDAP server for authorization purposes. The distinguished name of the ZeroReveal Client's certificate can be configured in the LDAP server as an administrator in order to lookup the user ID taken from the subject alternative name of the user's certificate.

Some additional details are provided in the *LDAP Configuration*.

4.4 ZeroReveal Server Authorized Clients File

In addition to the settings in *Configuring ZeroReveal Server*, the following must occur as well:

ZeroReveal Server uses a JSON file to specify the identities of the TLS client certificates corresponding to ZeroReveal Clients allowed to connect. Administrators are encouraged to use the interactive utility script at `/usr/local/enveil/zeroreveal-server/bin/auth-utility` to manage changes to this file.

An example file with a single entry follows:

```
[
  {
    "userId": "aws_enveil_client",
    "certificateIdentifier": "CN=Enveil Example, C=US",
    "permissions": [ "datasource:*" ],
    "publicKeyId": "EC:063f16569b38b2181f3976f77f5a7dd6414adc3cc1564bd2a6cca7144664a461"
  }
]
```

The fields are:

- `userId` — An administrator-selected name for the ZeroReveal Client

- `certificateIdentifier` — The certificate's DN or a SAN as rendered by `keytool`
- `permissions` — A list of permissions. This example has a single permission `datasource:*` that confers access to all data sources the ZeroReveal Server knows about
- `publicKeyId` — A representation of the ZeroReveal Client certificate's public key starting with the algorithm (`RSA` or `EC`) followed by a colon and ending in the lowercase SHA-256 hash of the raw public key as extracted from the certificate

5. Using ZeroReveal to Generate Certificate Signing Requests

Both ZeroReveal Server and ZeroReveal Client include a script called `csr-utility` which uses the ZeroReveal DRBG and cryptographic libraries to generate RSA and Elliptic Curve keys then prepare a basic Certificate Signing Request for signing of a certificate by a Certificate Authority.

5.1 Prerequisites

Both `python3` and `openssl` must be installed to use `csr-utility`.

5.2 Locating `csr-utility`

On ZeroReveal Server, `csr-utility` is located at `/usr/local/enveil/zeroreveal-client/bin/csr-utility`.

On ZeroReveal Client, `csr-utility` is located at `/usr/local/enveil/zeroreveal-client/bin/csr-utility`

5.3 Usage

To generate an RSA 2048-bit private key and Certificate signing request to be placed in `/new-enveil-key/`, use this command:

```
bash$ /usr/local/enveil/zeroreveal-server/bin/csr-utility rsa 2048 \
                  --output-name-base /new-enveil-key/rsa-2048
```

This command would create two files:

- An unencrypted PEM private key at `/new-enveil-key/rsa-2048.key.pem`
- An Certificate Signing Request at `/new-enveil-key/rsa-2048.csr.pem`

Submit the CSR to the appropriate Certificate Authority for signing. By default it will have a nondescript and invalid Distinguished Name. **As a result, the Certificate Authority will need to ensure that issued**

certificates contain Subject Alternative Names of the hostname or the IP address of the ZeroReveal Client or ZeroReveal Server host machine the certificate is intended for.

Once the Certificate Authority has signed the certificate, use the instructions in *Creating a bcfks from a p12* to build a `bcfks` key store containing the certificate and private key, then delete the `PEM` encoded private key from disk.

5.4 Supported key types

RSA Keys

RSA keys of size 2048, 3072, and 4096 are supported and can be generated with invocations of `csr-utility` like the following:

```
bash$ csr-utility rsa <key-size>   # where <key-bits> is 2048, 3072, or 4096
```

Elliptic Curve Keys

NIST Curve P-256 keys are supported and can be generated with invocations of `csr-utility` like the following:

```
bash$ csr-utility ec secp256r1
```

NIST Curve P-384 keys are supported and can be generated with invocations of `csr-utility` like the following:

```
bash$ csr-utility ec secp384r1
```

5.5 Creating a `bcfks` from a `p12`

To configure ZeroReveal Client and ZeroReveal Server in Common Criteria compliant configurations, all TLS key stores and TLS trust stores must be in the Bouncy Castle FIPS Key Store Format, also known as `bcfks`. To convert a key store (in this example, `keystore-in.p12`) to `bcfks` follow these steps on a machine with installed. The steps below ensure that the Bouncy Castle cryptographic library uses `AES-CCM` for protection of stored credentials.

NOTE: Be sure to use the same password for the new key store as was used for the existing key store otherwise `keytool` may produce a corrupted key store

5.5.1 On a machine with ZeroReveal Client installed

To convert `keystore-in.p12` to `keystore-out.bcfks` on a machine with ZeroReveal Client installed, run the following command. You will be prompted for the password to decrypt `keystore-in.p12` and a new password for `keystore-out.bcfks`, be sure to use the same password for both.

```
bash$ keytool -providerclass org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider \
-providerpath /usr/local/enveil/zeroreveal-client/bin/bouncy-castle-fips.jar \
-importkeystore -srcstoretype bcpkcs12 -deststoretype bcfks \
-srckeystore keystore-in.p12 -destkeystore keystore-out.bcfks
```

5.5.2 On a machine with ZeroReveal Server installed

To convert `keystore-in.p12` to `keystore-out.bcfks` on a machine with ZeroReveal Server installed, run the following command. You will be prompted for the password to decrypt `keystore-in.p12` and a new password for `keystore-out.bcfks`, be sure to use the same password for both.

```
bash$ keytool -providerclass org.bouncycastle.jcajce.provider.BouncyCastleFipsProvider \
-providerpath /usr/local/enveil/zeroreveal-server/bin/bouncy-castle-fips.jar \
-importkeystore -srcstoretype bcpkcs12 -deststoretype bcfks \
-srckeystore keystore-in.p12 -destkeystore keystore-out.bcfks
```

5.5.3 Using a GUI

The Free and Open Source program KeyStore Explorer offers a GUI interface to key store management that supports changing a key store's type. It can be downloaded from https://keystore-explorer.org/. To convert an existing key store of any format into a `bcfks`, follow these steps[1]:

1. Open the source key store in KeyStore Explorer.
2. Use the `File > Save As` menu item to save a new copy of the key store (for example `out-keystore.bcfks`)
3. Use `Tools > Change KeyStore Type > BCFKS` menu item to set the key store type as `bcfks`. This will prompt for a new password, be sure to use the same password for the new key store as was

[1] These instructions tested with KeyStore Explorer version 5.4.3

used for the original key store.

4. Save the converted key store. It is now ready for use.

6. Appendix

6.1 Basic Configuration

Example HOCON Configuration File:

```
string.property.to.set = ValueToSetSinglePropertyTo
more.complex.property = Even Properties with spaces are unquoted

# Comments are lines that start with a '#'

# Non-string properties are set just like strings
integer.property = 44
decimal.property = 3.14

# Boolean properties are case sensitive true/false
boolean.property = false

# Time-related properties can have units
time.property = 300s

# Data size properties can have units.
size.property = 1GB
```

Whitespace at the end of lines is ignored by the ZeroReveal Server.

When using strings with special characters, like a password or an LDAP or HTTP URL, the string may need to be wrapped with quotes so that the HOCON format interprets them literally.

For values that denote time, units should be specified. If no units are provided, the value will be interpreted in milliseconds. Valid units include, but are not limited to, `d` (day), `h` (hour), `m` (minute), `s` (second), and `ms` (millisecond). For the full list of valid units, see the section on *Duration format* in the HOCON documentation.

For values that denote data size, units should be specified. If no units are provided, the value will be interpreted in bytes. Valid units include, but are not limited to, `GB` (gigabyte), `MB` (megabyte), and `kB` (kilobyte). For the full list of valid units, see the section on *Size in bytes format* in the HOCON

documentation.

If a setting's value is a path, it may be an absolute path or a relative one. Relative paths are resolved relative to the working directory used to launch ZeroReveal Server; as such, absolute paths are preferred.

The ZeroReveal Server installation comes with a `server.conf` file that has reasonable default values in as many places as possible.

The ZeroReveal Client installation comes with a `client.conf` file that has reasonable default values in as many places as possible.

6.2 LDAP Configuration

Enveil allows system administrators to determine the authentication and authorization stores and mechanisms they want to use. An administrator specifies which store or stores to use by setting the value `enveil.client.auth.mechanisms` property in the file `client.conf`, a settings file used by the client when starting up. Set the value of `enveil.client.auth.mechanisms` to a comma separated list of authentication mechanism names.

6.2.1 Configuring LDAP As An Auth Store

Set the following property in `client.conf` to tell ZeroReveal Client how to connect to LDAP:

enveil.client.auth.mechanisms
: A comma-separated list of authentication and authorization stores to use. For LDAP, must include `ldap`.

enveil.client.auth.ldap.url
: The URL for the LDAP connection.

enveil.client.auth.ldap.ssl.enabled
: `true` if the LDAP connection will use TLS, `false` otherwise. Note that SSL was replaced by TLS, but is still used to refer to TLS in some places.

Note that if `enveil.client.auth.ldap.ssl.enabled` **is set to** `true`**, trust store information for communicating with your LDAP server must be in the trust store configured by the** `enveil.security.tls.truststore.*` **properties.**

If your LDAP server supports mutual TLS, the ZeroReveal Client will attempt to use the TLS key store specified by the `enveil.security.tls.keystore.*` properties when connecting to the LDAP server.

Next you must tell ZeroReveal Client how to authenticate itself to LDAP, and how it will search for users in LDAP.

6.2.1.1 Authenticating ZeroReveal Client to LDAP

ZeroReveal Client can authenticate itself to LDAP with a certificate.

Certificate

If your LDAP system supports the SASL EXTERNAL[2] protocol, you may use a TLS client certificate/key pair contained in ZeroReveal Client's TLS key store to authenticate to the LDAP server with a certificate.

Configure ZeroReveal Client by first obtaining the following:

- A TLS client certificate for ZeroReveal Client, signed by a CA trusted by the LDAP server, and the accompanying private key. Make sure the identity on this certificate is in the format required for LDAP to recognize it when connecting with SASL.
- The certificate for the CA that issued a TLS server certificate to the LDAP server.

Next, ensure that the CA certificate that issued the LDAP server's TLS server certificate is present in ZeroReveal Client's TLS trust store, configured above. Also ensure the TLS client certificate/key pair has been added to the ZeroReveal Client's TLS key store, configured above as well.

Finally, set the following values in `client.properties`:

- `enveil.client.auth.ldap.connect.with.sasl.external` – set to `true`.

ZeroReveal Client will now connect to the LDAP server over mutual TLS using the SASL protocol.

6.2.1.2 Configure LDAP searches

Next you must set the properties that tell ZeroReveal Client how to verify a user's identity and pull their permissions from LDAP. When a user connects to ZeroReveal Client, it does two LDAP searches: the first looks up their Distinguished Name (DN) and verifies their certificate; the second looks up the set of groups the user is associated with. The user's permissions are the union of those we find in the Enveil Permission Attribute (explained *below*) on the user or any of their groups.

[2] See https://tools.ietf.org/html/rfc4422

First, set these properties:

enveil.client.auth.ldap.search.base

The search root in the LDAP tree. Example: `dc=example,dc=org`

enveil.client.auth.ldap.permission.attribute.name

The name of the Enveil Permission Attribute. Example: `enveilPermission`. Defaults to `enveilPlatformPermission`.

Next, you must configure how ZeroReveal Client will look up users in LDAP. This is done using the Subject Alternative Name (SAN) on the certificate. For example, `uid=Bob,ou=Users,dc=example,dc=org` is a valid SAN.

The ZeroReveal Client will then perform DN lookup based on the SAN, using `enveil.client.auth.ldap.user.dn.full.lookup.filter`. This is a templated LDAP search filter[3] that allows fine-grained control over the LDAP search for a given SAN. It must contain "{0}"; when a user attempts to connect to ZeroReveal Client, any "{0}"'s will be replaced with their user ID (uid) and the resulting LDAP search filter will be run. For example, if `client.conf` contains:

```
enveil.client.auth.ldap.search.base=dc=example,dc=org
enveil.client.auth.ldap.user.dn.full.lookup.filter=(& (objectClass=person) (uid={0}))
```

Then if a user Bob attempts to log in to ZeroReveal Client, we will run an LDAP search with base `dc=example,dc=org` and filter:

```
(& (objectClass=person) (uid=Bob))
```

This search must return exactly one object to be considered successful; that object's DN is then treated as Bob's DN. The provided certificate is also verified as the certificate associated with this DN. If this object has an Enveil Permission Attribute, they will be Bob's initial permissions for ZeroReveal Client.

[3] https://www.ldap.com/ldap-filters

Group lookup

Next, you must configure the group lookup using the setting `enveil.client.auth.ldap.user.group.membership.filter`. This has the same format as `enveil.client.enveil.client.auth.ldap.user.dn.full.lookup.filter` explained above, but "{0}" is replaced with a DN, it is used to find groups instead of users, and any number of resulting objects is valid. For example, if `client.conf` contains:

```
enveil.client.auth.ldap.search.base=dc=example,dc=org
enveil.client.auth.ldap.user.group.membership.filter=(& (objectClass=team) (uniqueMember={0}
↪))
```

Then if a user Bob attempts to log in to ZeroReveal Client, and we find Bob's DN to be `cn=Bob,dc=example,dc=org`, we will search for groups using the filter:

```
(& (objectClass=team) (uniqueMember=cn=Bob,dc=example,dc=org))
```

If any of these groups has an Enveil Permission Attribute, those permissions will be added to Bob's permissions for ZeroReveal Client. If no permissions were found on Bob or any of his groups, then Bob will have no permissions in ZeroReveal Client; they will not be able perform any actions.

6.2.1.3 Assign ZeroReveal Client permissions in LDAP

Before any users will be able to perform queries, you must grant them permissions in LDAP. Permissions control which data sources a user is allowed to see and run queries against, and which administrative privileges they have in ZeroReveal Client.

As described above, ZeroReveal Client determines a user's permissions by reading the contents of the Enveil Permission Attribute found on the user and all of their groups. The name of the Enveil Permission Attribute is set by `enveil.client.auth.ldap.permission.attribute.name` in `client.conf`, and defaults to `enveilPlatformPermission`.

Add the Enveil Permission Attribute to LDAP Schema

Before you can assign the Enveil Permission Attribute to users and groups in LDAP, you must add the attribute to your LDAP schema. Contact your LDAP System Administrator for help with adding this attribute.

An example LDIF file for adding a new attribute called `enveilPlatformPermission` is installed with ZeroReveal Client to `/usr/local/enveil/zeroreveal-client/auxiliary/add_enveil_permission_attribute.ldif`. This LDIF file is compatible with OpenLDAP and Microsoft Active Directory. When imported into your LDAP server, this file will add `enveilPlatformPermission` to your LDAP schema, which will allow you to add `enveilPlatformPermission` attributes to users and groups.

6.3 Connecting ZeroReveal Client and ZeroReveal Server

ZeroReveal Client and ZeroReveal Server communicate over a direct network connection. To authenticate and secure HTTP connections between ZeroReveal Client and ZeroReveal Server installations, Enveil uses mutually authenticated TLS: both applications will check the others' TLS certificate to ensure that it has been signed by a Certificate Authority they trust before they will begin to communicate.

Furthermore, ZeroReveal Server uses certificate pinning to keep track of permissions for each ZeroReveal Client it has authorized to run queries. ZeroReveal Server has an explicit list of client certificates that are allowed to connect to it, and what their permissions are, and will reject any connections not using those certificates.

6.3.1 Adding a Gateway Specification File

ZeroReveal Client maintains a gateway specification file for each active ZeroReveal Server connection in its gateway specification directory, specified as `enveil.client.gateway.specification.dir` in `client.conf`. This property defaults to `/etc/enveil/zeroreveal-client/gateways`. It must specify a directory that exists, or ZeroReveal Client will not start.

A gateway specification file is a properties file that configures how ZeroReveal Client connects to ZeroReveal Server. Create a new file ending with `.properties` in the gateway specification directory, and ensure it is only readable by the `enveil` user. A sample file, which you can use as a template, is installed to `/etc/enveil/zeroreveal-client/gateway.sample.properties`.

To begin, set the following properties:

- `enveil.client.gateway.name`: This is the name of the gateway, which users will specify when issuing queries to its data sources. This name must only consist of alphanumeric or underscore characters.
- `enveil.client.gateway.connection`: The type of connection; set to `http`.
- `enveil.client.gateway.rest.uri` The URI of the ZeroReveal Server to connect to; get this from the ZeroReveal Server administrator.

6.3.2 Configuring ZeroReveal Server's TLS trust store

If ZeroReveal Server trusts ZeroReveal Client's CA, it should already have a trust store with the required certificates, so this step can be skipped.

If ZeroReveal Server does not already trust ZeroReveal Client's CA, the ZeroReveal Server administrator must import the new CA certificate into the key store pointed to by the property `enveil.security.tls.truststore.path` in `server.conf`.

6.3.3 Create a TLS trust store for ZeroReveal Client

ZeroReveal Client uses a different TLS trust store for each ZeroReveal Server it is configured with. Therefore, you will always create a new key store containing ZeroReveal Server's CA certificate and use it as the TLS trust store for this connection.

To configure ZeroReveal Client, load the key store file onto its local file system (e.g. in `/etc/enveil/zeroreveal-client/certs/`), ensuring that **only the** `enveil` **user has permission to read or modify it**. Then, in the ZeroReveal Client's gateway specification file for this connection, set the following properties:

- `enveil.security.tls.truststore.path` – Path to the trust store on ZeroReveal Client's local disk.
- `enveil.security.tls.truststore.type` – Type of the trust store: `jks` or `bcfks`.
- `enveil.security.tls.truststore.password` – The key store's password.

6.4 Configure TLS Key Store for ZeroReveal Client

The next step is to create a key store containing the TLS client certificate/key pair created for ZeroReveal Client above; this will serve as ZeroReveal Client's TLS key store for this connection. Save the key store on the ZeroReveal Client's local file system (e.g. in `/etc/enveil/zeroreveal-client/certs`), ensuring that **only the** `enveil` **user has permission to read or modify it**. Then, in the ZeroReveal Client's gateway specification file for this connection, set the following properties:

enveil.security.tls.keystore.path
: Path to the key store on ZeroReveal Client's local disk.

enveil.security.tls.keystore.type
: Type of the key store (`jks`, `pkcs12`, or `bcfks` format).

enveil.security.tls.keystore.password
: The key store's password.

6.5 Set ZeroReveal Client's Permissions on ZeroReveal Server

Finally, ZeroReveal Server also needs to be configured with the ZeroReveal Client's authorizations (i.e. what actions it is allowed to perform). Enveil provides a separate tool for this, `auth-utility`. It is installed to `/usr/local/enveil/zeroreveal-server/bin/auth-utility`. The utility can be used in an interactive CLI mode or in an unattended mode.

6.5.1 Interactive ZeroReveal Server `auth-utility` Use

First the utility will prompt for the location of ZeroReveal Server's auth file. This should be the same file as specified by `enveil.server.authFile` in ZeroReveal Server's configuration file. If that file does not exist, the utility will ask to create it. **If you choose to create the file, check its permissions after finishing with the auth utility**; if the file does not exist and you do not create it, the tool will exit immediately. The auth file must be owned by the `enveil` user and must not have any permissions for other users.

Once the auth file path has been entered, the `auth-utility` will provide a menu that provides options for listing current permissions, editing or deleting existing client entries, or adding a new client.

The dialogue for creating a new client will first request the human-readable name for the entry, the User ID; this name must be unique in the auth file. Next, it will ask for the full path of a public certificate file. Finally, you will be asked to list permissions for the new entry (see *below* for more details on permissions).

In summary, to add a client certificate to ZeroReveal Server auth list, you should:

1. Navigate to the directory where the auth utility is stored, and execute:

   ```
   ./auth-utility
   ```

2. Enter the path to ZeroReveal Server's auth file when prompted.
3. Choose "Add a new authorized client."
4. Enter a system-unique User ID (human readable name for the new record) when prompted.
5. Enter the path to ZeroReveal Client's certificate when prompted.
6. Supply a system-unique, human-readable user identifier for the client.
7. Choose permissions for the client. This controls which data sources that client may query.

The auth utility also allows you to modify or remove clients or permissions. Note that these changes will be picked up by a running ZeroReveal Server; no restart is required to apply them.

6.5.2 Permissions Format

Data source permissions on a ZeroReveal Server are in the form `datasource:[data_source_name]`. `data_source_name` refers to a specific data source provided by the ZeroReveal Server. Note that all permission strings are case-insensitive.

6.5.3 Permission Wildcards

Data source permissions support limited use of asterisks (`*`) for wildcards.

6.5.3.1 Example Permissions

- `datasource:*` — Grants access to all data sources on a ZeroReveal Server.
- `datasource:activeRecords` — Grants access to the single data source named `activeRecords` on a ZeroReveal Server.

ABOUT ENVEIL: Enveil is a pioneering Privacy Enhancing Technology company protecting Data in Use. Enveil's NIAP/CSfC-certified ZeroReveal® solutions allow data to be securely processed with sensitive indicators while remaining in the untrusted domain, extending the boundary of trusted compute. Defining the transformative category of Privacy Enhancing Technologies (PETs), Enveil enables secure cross-domain and tactical edge collaboration by ensuring the content of the search or analytic – and its corresponding results – remain encrypted during processing. Founded by U.S. Intelligence Community alumni with backgrounds in mathematics, algorithmics, and machine learning, Enveil is revolutionizing data security by addressing a Data in Use vulnerability that people have been chasing for more than 40 years. Learn more at www.enveil.com.