Unbound CORE Virtual Enclave for Desktops

User Guide

Version 1.0.2105.41070

August 2021

Table of Contents

1. Revision History

The following table shows the changes for each revision of the document.

Version	Date	Description
1.0.2105.41070	August 2021	Added the section Cryptographic Service Provider. Added the section Certificate Usage.
1.0.2105	June 23, 2021	Updated Apache Configuration.
1.0.2105	June 1, 2021	Initial version.

2. Overview

This document provides instructions for server-side setup and client-side setup for the **Unbound CORE Virtual Enclave technology for Desktops**.

The Virtual Enclave replaces traditional smartcards with virtual ones to authenticate employees and reduce operational strain and total cost of ownership. See here for more information.

2.1. Requirements

The following are required to install CORE.

CORE Server

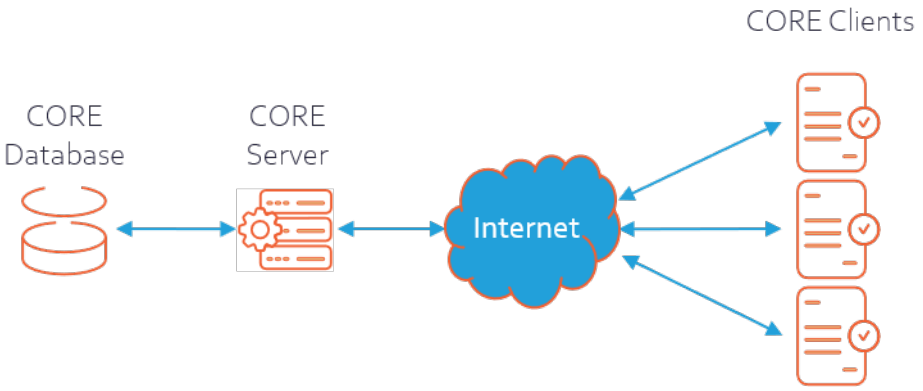
- CentOS/RHEL 7
- Apache server installed
- Java 1.8 or OpenJDK 11 or newer

CORE Client

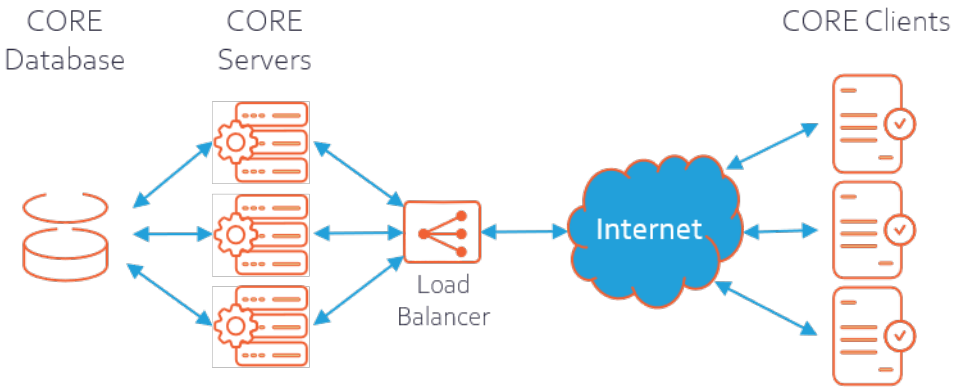
- Windows 10 (version 1703 or newer)
- Supported browsers
    - Google Chrome - 91.0.4472.77 or newer
    - Microsoft Edge - 91.0.864.37 or newer
    - Mozilla Firefox –- 88.0.1 or newer

3. Server Installation

The architecture of the CORE enclave system is as follows:



A high availability system can be configured as follows:



Note
When implementing a high availability system, the *cot.pkcs12* and *cot.conf* files (described later) must be distributed to all CORE servers.

Use the following instructions to install the Unbound CORE Virtual Enclave.

1. Install and configure a PostgreSQL instance. Check that the PostgreSQL CLI is available and usable on the instance used for the CoT server installation. CoT supports both local installation of PostgreSQL as well as PostgreSQL as-a-service (such as AWS RDS).
2. Install CORE using the package file. For example, on Linux, use *cot-<VERSION>.rpm*.
3. Run the following command to generate the self-signed **Certificate Authority** (or alternatively use your own CA certificate).

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias cacert -genkeypair -dname "cn=UNBOUND-CA" -validity 10000 -
keyalg RSA -keysize 2048 -ext bc:c
```

4. Generate the **JWT certificate** signed by CA (from the previous step). This is used for client authorization.
    a. Generate a key pair.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias jwtcert -genkeypair -keyalg RSA -keysize 2048 -dname
"cn=cot-server"
```

> **Note**
> The value for *dname* must be *cot-server*.

    b. Sign the key with the CA.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias jwtcert -certreq -file jwt.req
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias cacert -gencert -infile jwt.req -outfile jwt.cer -
validity <DAYS> -dname "cn=cot-server"
```

    c. Import to the keystore.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias jwtcert -importcert -file jwt.cer
```

5. Generate a **secure channel** certificate signed by CA. This is used for secure communication from the client to server.
    a. Generate a key pair.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias channelcert -genkeypair -keyalg RSA -keysize 2048 -
dname "cn=cot-server"
```

> **Note**
> The value for *dname* must be *cot-server*.

    b. Sign the key with the CA.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias channelcert -certreq -file channel.req
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias cacert -gencert -infile channel.req -outfile
channel.cer -validity <DAYS> -dname "cn=cot-server"
```

    c. Import to the keystore.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-
PASSWORD> -alias channelcert -importcert -file channel.cer
```

6. Generate a **master key** certificate signed by CA. This is used to encrypt the database.
    a. Generate a key pair.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-PASSWORD> -alias rsakey -genkeypair -keyalg RSA -keysize 2048 -dname "cn=cot-server"
```

> **Note**
> The value for *dname* must be *cot-server*.

    b. Sign the key with the CA.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-PASSWORD> -alias rsakey -certreq -file rsakey.req
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-PASSWORD> -alias cacert -gencert -infile rsakey.req -outfile rsakey.cer -validity <DAYS> -dname "cn=cot-server"
```

    c. Import to the keystore.

```
keytool -keystore cot.pkcs12 -storetype pkcs12 -storepass <KEYSTORE-PASSWORD> -alias rsakey -importcert -file rsakey.cer
```

7. Update */etc/unbound/cot.conf*. These changes include the database connection details and details of the keys used for the service.

```
enc.keystore.password=<KEYSTORE-PASSWORD>
enc.keystore.type=pkcs12
enc.keystore.file=/etc/unbound/cot.pkcs12
enc.jwtcert.alias=jwtcert
enc.cacert.alias=cacert
enc.key.alias=rsakey
enc.channelcert.alias=channelcert
db.password=<DATABASE-PASSWORD>
db.userName=sa
db.url=jdbc:postgresql://<DATABASE-URL>:5432/PSQL
db.jar=/etc/unbound/postgresql-42.2.19.jar
db.driver=org.postgresql.Driver
```

8. Run the PSQL command *unbound-postgresql.sql* included with the package. It creates the CoT database schema in the PostgreSQL database.

```
psql -h ' + <PostgreSQL-Server-URL> + ' -p 5432 -U sa -d PSQL -f /home/ubuntu/unbound-postgresql.sql
```

> **Note**
> This step only needs to be run one time for each server. If you are only reconfiguring the product, then it does not need to be run.

9. Start (or restart) the CORE CoT service.

```
sudo service cot start
```

3.1. Apache Configuration

This extra configuration to Apache is required for CORE.

1. Enable Apache httpd connections.

```
sudo /usr/sbin/setsebool -P httpd_can_network_connect 1
```

2. Edit the CORE configuration file */etc/httpd/conf.d/cot-apache.conf*. The file should be modified to match these contents:

```
KeepAlive On
KeepAliveTimeout 1200
TimeOut 4000

<VirtualHost *:80>
 ProxyPreserveHost On
 ProxyTimeout 4000
 ProxyPass /api http://localhost:8080/cot-server/api
 ProxyPassReverse /api http://localhost:8080/cot-server/api
 ProxyPass /mpcengine ws://localhost:9090/mpcengine disablereuse=on
 ProxyPassReverse /mpcengine ws://localhost:9090/mpcengine
</VirtualHost>
```

3. Configure the load balancer to have an idle timeout 4000 seconds. This action is dependent on the type of load balancer that you are using.
4. Run the following command to start Apache.

```
sudo systemctl start httpd
```

3.2. Test the System

Test the enclave by running the following command:

```
curl http://<HOSTNAME>/api/v1/health
```

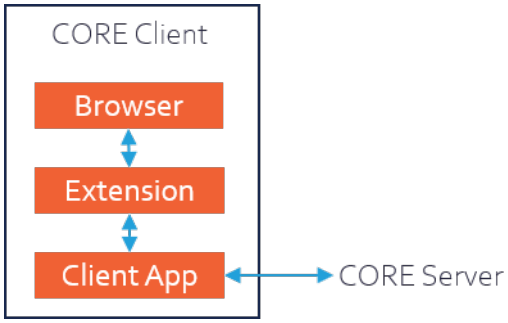
Note
Access via HTTP or HTTPS depends on how it is configured in Apache.

The response should be similar to:

```
{"isMPCEngineInitialized":true,
"isDatabaseConnected":true,
"keystoreCertificates":["cacert","jwtcert","channelcert","rsakey”],
"serverVersion":"",
"serverPort":"8080",
"MPCEnginePort":"9090"}
```

4. Client Installation

The client installation consists of an app that runs on the client device. The browser then uses a browser extension that communicates with the app and subsequently with the CORE Server.



4.1. CORE Client App Installation

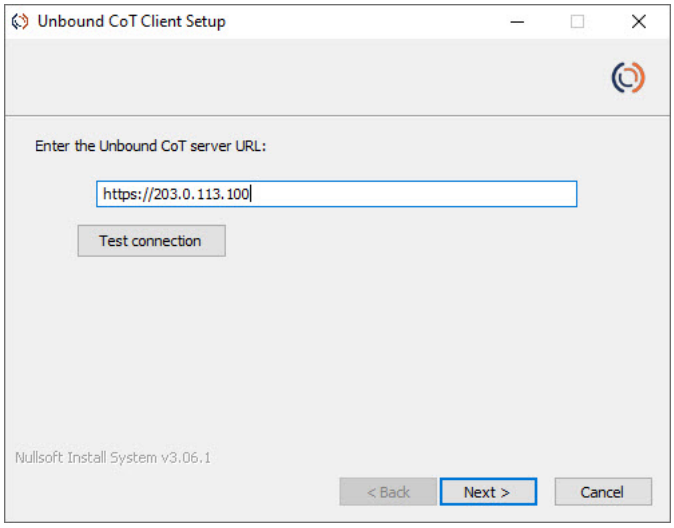
The CORE Client app is installed using the following file:

```
cot-client-<version>.exe
```

> **Note**
> The installation should be run as *Administrator*.

Follow the steps in the installation wizard.

During installation, you need to enter the CORE Server URL. Note that the CORE Server URL is just the IP address **without** any port or sub-path. For example:



> **Note**
> Installation is executed per user on the system.

4.1.1. Cryptographic Service Provider

The Unbound CORE installer includes Unbound's Microsoft Cryptographic Service Provider (CSP). This module provides CSP supported functionality, as well as enabling 2-way TLS using CORE client certificates.

If CORE is installed by a user **with** administrator privileges, the CSP module is automatically installed.

If CORE is installed by a user **without** administrator privileges, the CSP module is not installed. After this type of installation, an administrator can run *csp.reg* (provided with the package), which updates the installation with the CSP module.

> **Warning**
> It is recommended to use the default installation path. If you need to install it in a different path, contact Unbound Support.

4.1.2. CORE Client Settings

The CORE Client provides settings that can be updated in the registry. To access the settings, open the registry and navigate here:

```
[HKEY_CURRENT_USER\SOFTWARE\Unbound\cot]
```

The following entries can be updated:

- **Preactivated keys** – default is 1. Key creation can take some time due to connection delays, and therefore the app prepares a number of keys for usage upon start up. This setting should reflect the maximum number of keys that you anticipate a user to need.

  For example:

  ```
  “preactivatedKeysCount”=dword:00000001
  ```

- **Time interval** – default is 60 seconds. The app creates keys as explained in the previous setting. Since this requires some time, the app periodically checks how many keys are available. If there are no preactivated keys available, then the app will start generating a new preactivated key.

  For example:

  ```
  “checkTimeInterval”=dword:0000003c
  ```

- **CORE server URL** – URL of the CORE server. During installation, you need to enter the CORE Server URL. In case an error was entered for the CORE Server URL, you can update it in the registry.

  For example:

  ```
  “url”=https://<URL>
  ```

4.2. Browser Extension Installation

The browser extension is available for the following browsers:

- Google Chrome, found here.
- Microsoft Edge, found here.
- Mozilla Firefox, found here.

4.3. Certificate Usage

If you are upgrading from a version prior to 1.0.2105.41070, this process needs to be run once after upgrade. Assuming that you have a key and certificate stored in CORE Enclave, the certificate needs to be synced with the Windows certificate store. The rest of the certificates created before the upgrade are copied automatically to the new version.

To do this, run:

```
ucl sync-cert -csp
```

Appendix A. Custom Headers

The web socket handshake header between the client and server includes the following custom headers:

- PEER_FROM
- PEER_TO
- CONNECTION_PARAM

For example:

```
"PEER_FROM: cot-client"
"PEER_TO: cot-server"
"CONNECTION_PARAM: 5a8f9963c012ee5b"
```