



# Palo Alto Networks Common Criteria Evaluated Configuration Guide (CCECG) Cortex XSOAR Server and Engine 6.6

Revision Date: September 16, 2022

Palo Alto Networks, Inc.  
[www.paloaltonetworks.com](https://www.paloaltonetworks.com)

© 2022 Palo Alto Networks, Inc. Palo Alto Networks is a registered trademark of Palo Alto Networks. A list of our trademarks can be found at <https://www.paloaltonetworks.com/company/trademarks.html>. All other marks mentioned herein may be trademarks of their respective companies.



# XSOAR

Palo Alto Networks, Inc.

[www.paloaltonetworks.com](https://www.paloaltonetworks.com)

© 2022 Palo Alto Networks, Inc. Palo Alto Networks is a registered trademark of Palo Alto Networks. A list of our trademarks can be found at <https://www.paloaltonetworks.com/company/trademarks.html>. All other marks mentioned herein may be trademarks of their respective companies.

# Table of Contents

- Introduction .....4
  - Document Purpose and Scope .....4
  - Scope of Evaluation .....4
- Software Download and Installation .....5
  - Download .....5
  - Server Installation .....5
  - Engine Installation.....7
- TOE Operations ..... 10
  - Viewing the Current TOE Version ..... 10
  - User Management ..... 11
  - Password Change..... 12
  - Password Complexity Management ..... 13
  - TLS/X509 Configuration ..... 14
  - Configure Mutual Authentication for the Web UI ..... 26
  - Configure the Sensitive Data to be Encrypted ..... 31
  - Update the TOE Version ..... 32

## Introduction

The Cortex XSOAR is the most comprehensive SOAR (Security Orchestration, Automation and Response) platform in the market today, orchestrating across hundreds of security products to help your SOC customers standardize and automate their processes for faster response times and increased team productivity.

## Document Purpose and Scope

This document provides administrative guidance for the Cortex XSOAR Server and Engine and is a supplement to the Palo Alto Networks Cortex XSOAR Administrator's Guide Version 6.6. This document describes procedures on how to operate and prepare the Cortex XSOAR to meet its Common Criteria evaluated configuration and is referred to as the operational user guide in the Application Software Protection Profile v1.4 [APPSW] and Functional Package for Transport Layer Security v1.1 [PKG TLS] that meets all the required guidance assurance activities from the APPSW and PKG TLS.

The Palo Alto Networks System documentation set includes online help and PDF files.

The following product guidance documents are provided online or by request:

- Cortex XSOAR Administrator's Guide Version 6.6, Last Revised: See Link Below  
<https://docs.paloaltonetworks.com/cortex/cortex-xsoar/6-6/cortex-xsoar-admin.html>
- Palo Alto Networks Common Criteria Evaluated Configuration Guide (CCECG) for Cortex XSOAR 6.6 [This Document]

The most up-to-date versions of the documentation can be accessed on the Palo Alto Networks Support web site (<https://support.paloaltonetworks.com>) or Technical Documentation (<https://docs.paloaltonetworks.com/>).

## Scope of Evaluation

The scope of evaluation only covers security functionalities specified by *Protection Profile for Application Software* and *Functional Package for Transport Layer Security*. The security functions tested are outlined in the Security Target (ST), and these functions include TLS trusted channels, X509 authentication, certificate validation and signature checking.

### TOE Identification –

- Cortex XSOAR Server 6.6.0 running on RedHat 8<sup>1</sup>.
- Cortex XSOAR Engine 6.6.0 running on RedHat 8.

---

<sup>1</sup> While the TOE was tested on RHEL 8, additional Linux environments such as RHEL 7, Ubuntu (18.04, 20.04), Oracle Linux 7, and Amazon Linux 2 are supported as well. This is vendor affirmed. The minimum hardware requirements on those system are 16 CPU cores, 32 GB of memory, and 1 TB SSD hard disk.

# Software Download and Installation

## Download

Customer Support will send purchasing customers an email with a download link. They will also send license key and instructions in the email. The link is only valid for a limited number of downloads and time. Please download the signed installer. You will also need to import the public key to the operating system.

## Server Installation

By default, Cortex XSOAR Server is installed in /root folder, but you can change the default folder, if necessary.

Asset	Path
Binaries	/usr/local/demisto
Data	/var/lib/demisto
Logs	/var/log/demisto
Configurations	/etc/demisto.conf
Reports	/tmp/demisto_install.log
Install Log	/tmp/demisto_install.log

## Prerequisites

Verify the following information and requirements before you install Cortex XSOAR.

- Your deployment meets the minimum system requirements.

Component	DEV Minimum	Production Minimum
CPU	8 CPU cores	16 CPU cores
Memory	16GB RAM	32GB RAM
Storage	500GB SSD	1TB SSD

- You have root access.
- Download Cortex XSOAR from the link that you received from Cortex XSOAR Support by running the following command. For example,  
  
**wget -O demisto.sh "<downloadLink>"**
  - If you are deploying Cortex XSOAR using a signed installer (GPG), you need to import the GPG public key that was provided with the signed installer.  
  
For example, you can use the **rpm -import <public.key>** command to import the public key into the local GPG keyring.
  - If you are deploying Cortex XSOAR using a signed installer (GPG) you might need to manually install the makeself package by running the **yum install makeself** command.

4. Run the **chmod +x demisto.sh** command to convert the file to an executable file. (If Installer is signed, use **rpm -K** to verify signature before installing.)
5. Execute the command **sudo ./demisto.sh**.
6. Accept the EULA.
7. Enter the Server HTTPS port, if desired. Default port is 443 (recommended).
8. Enter the admin name (default is admin).
9. Enter a secure password<sup>2</sup> for admin.

**NOTE:** The TOE enforces a password that is at least 8 characters and a combination of upper and lower characters, numbers, and/or special characters.

**WARNING:** Do not use the “-y” parameter. See footnote below.

---

<sup>2</sup> The default "admin" password will only be set if the "-y" (non-interactive installation mode) flag has been passed to the installer, during a clean installation. Even with this default password, the admin account is forced to change the password before logging in to the system for the first time. During this password change, default password complexity is enforced.

## Engine Installation

If you need a signed RPM file for installing an engine, you need to download the engine file from the download server. When you download Cortex XSOAR for the first time, you are sent a link to the download server. The download server includes a signed Cortex XSOAR installation file and a signed engine file. After you download and install the signed engine, you need to replace the conf file by creating an engine from the Cortex XSOAR server.

1. Download the signed engine file from the link you received from Cortex XSOAR support by running the following command.

```
wget -content-disposition '<download link>'
```

Use the original URL that was sent to you when installing Cortex XSOAR, by replacing the **downloadName** value with **<build-number>\_signed\_engine.rpm**. For example:

```
wget --content-disposition  
'https://download.demisto.com/downloadparams?token=aBCiXjNoSSxy&email=user@demisto.com&downloadName=  
1578666_signed_engine.rpm&eula=accept'
```

2. Verify the signature on the signed installer.

```
rpm -K <file-name.rpm>
```

3. If the verification fails, either import the correct public key or contact Support for new signed installer.
4. If verification passes, install the signed installer.

```
rpm -i <file-name.rpm>
```

5. In Cortex XSOAR Server, create a configuration file.
  - a. Select **Settings > Engines > Create New Engine**.
  - b. Type a meaningful name for the engine.
  - c. In the Installer type field, select **Configuration**.
  - d. Click **Create New Engine**.


In the machine you installed the Engine, replace the file `/usr/local/demisto/d1.conf` with the file you created in step 5.

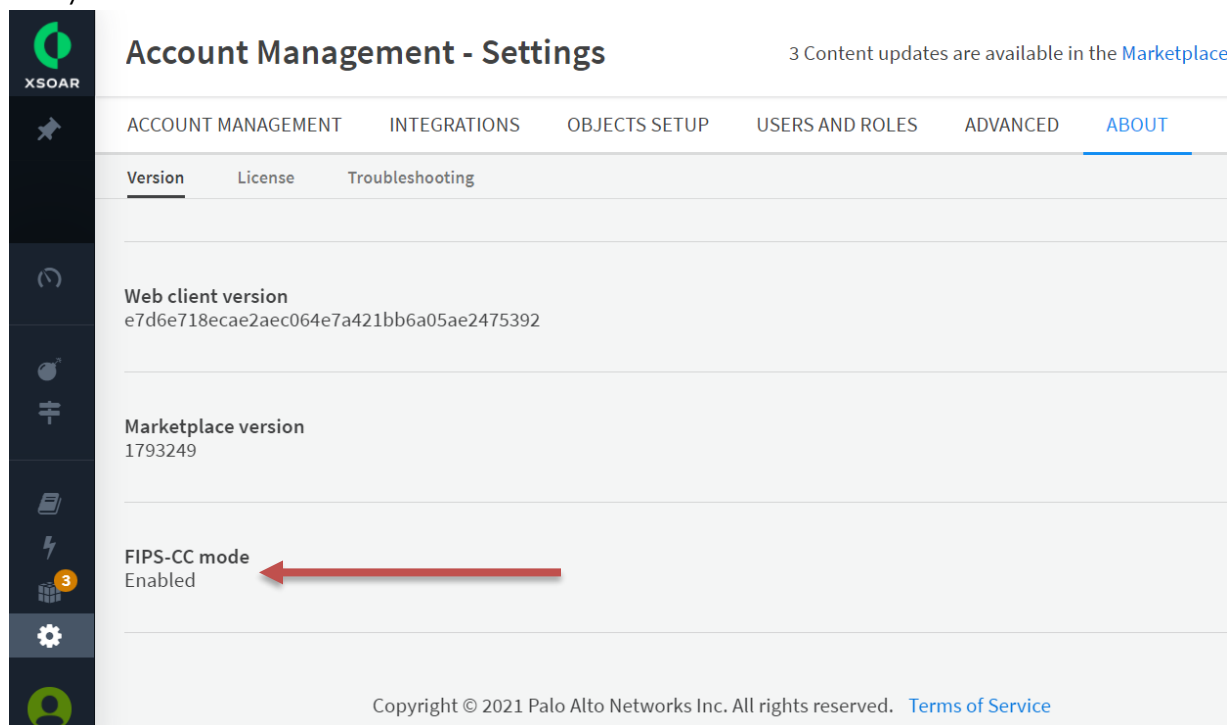
6. Restart the D1 service.

## Verify FIPS-CC Mode

Verify that FIPS-CC mode is enabled.

**NOTE:** While FIPS-CC mode cannot be disabled, it requires all power-up self-tests to pass. If any of the self-tests fails, the system will not start up (i.e., in FIPS error state). If this occurs, try uninstalling and re-installing the application.

1. Log into the Web UI (Cortex XSOAR Server only).
2. Go to Settings  Icon.
3. Go to **About** tab.
4. Verify that FIPS-CC mode is enabled.



On the Engine, if deployed, there is no web UI. However, you can look at the log to verify. For example,

```
sudo less /var/log/demisto/d1.log | grep FIPS-CC
```

```
...  
FIPS-CC self-tests passed. FIPS-CC mode enabled successfully.  
...
```

**WARNING:** The TOE is using BoringCrypto cryptographic module which uses NIST-validated and Approved algorithms as specified in [#A2517](#). The cryptographic engine cannot be changed by the security administrator or user.



## Confirm Connection to the Engine

Go to [Settings > Integrations > Engines](#). Make sure the Connected column is set to [True](#).

## Viewing the System Logs

```
sudo less /var/log/demisto/d1.log
```

```
sudo less /var/log/demisto/server.log
```

### 'tail' (instead of 'less') to see the latest records.

## Troubleshooting Errors


Error Message	Description
invalid signature by the server certificate	You use the wrong server key for the server certificate or copy the wrong key to cert.key.
certificate signed by unknown authority	You forgot to put CA certificate in the peer's trust anchor.
certificate has expired or is not yet valid	You use expired certificates.
warning error checking revocation via OCSP	The OCSP responder is not working, or the wait is too long.
certificate is revoked.	The peer certificate is revoked.
Could not check OCSP revocation [error 'ExtKeyUsageOCSPSigning not found on the OCSP responder']	OCSP responder certificate is missing OCSPsign in EKU field.
certificate signed by unknown authority (possibly because of "x509: invalid signature: parent certificate cannot sign this kind of certificate" while trying to verify candidate authority certificate	CA certificate in chain is missing basicConstraint cA flag or cA flag is set to FALSE.
certificate specifies an incompatible key usage	Server certificate is missing serverAuth or client certificate is missing clientAuth. Other EKU mismatch can trigger this as well such as CA certificate with OCSPsign in the EKU.
certificate is not valid for any names, but wanted to match <expected identifier>	Fail to match the expected identifier with the referenced identifier.
certificate is valid for <referenced identifier>, not <expected identifier>	Fail to match the expected identifier with the referenced identifier.
certificate relies on legacy Common Name field, use SANs or temporarily enable Common Name matching with GODEBUG=x509ignoreCN=0	Certificate is missing the SAN field. The peer certificate must contain a SAN field, or the TLS connection will be terminated.
fatal Failed load client certificate	Client certificate is in wrong format, corrupted, or tampered with. Also check the permission and ownership.

## TOE Operations

Once the TOE is in FIPS-CC mode, it will enforce the necessary CC requirements<sup>3</sup>. The administrator must still configure certification validation check and revocation check. In addition, the administrator must also enable and configure the SAN trust list for the Server.

**NOTE:** Web UI is only available on the XSOAR Server.

### Viewing the Current TOE Version

To view the current TOE version, the administrator can first navigate to the settings by clicking on (  ) and the [About](#) tab. The version is identified by the [Version](#) field.

On the Engine, if deployed, there is no web UI. However, you can look at the log to identify the version. For example,

**`sudo less /var/log/demisto/d1.log | grep version`**


`2021-10-18 18:35:27.2793 info Starting d1 server, version <version number> run id: 133e2886-81ae-4641-816a-6117fa8ccab5 (source: /builds/GOPATH/src/code.pan.run/xsoar/server/sensor/d1/engine/connection.go:71)`

---

<sup>3</sup> Restrict TLS version to 1.2 and Approved cipher suites only, RFC 6125 strict checking, X509 Key Usage and Extended Key Usage strict checking.

## User Management

Cortex XSOAR uses role-based access control (RBAC) for controlling user access. RBAC helps manage access to Cortex XSOAR components, so that users, based on their roles, are granted minimal access required to accomplish their tasks. View and manage different users in the [Users](#) tab. You can view the user's details such as name, email address, last log in, whether they have been locked out, and so on. You can also manage the user's password, unlock their account, disable, enable, and remove their account. You can add as many roles as required and change their permission levels (see [Roles](#) tab).

To manage users and roles, the administrator can first navigate to the settings by clicking on (  ) and the [Users and Roles](#) tab.

You can manage the following settings/roles in the [Users and Roles](#) tab:

- View and manage different roles and access permissions in the [Roles](#) tab. You can add as many roles as required and change their permission levels.
- View and manage different users in the [Users](#) tab. You can view the user's details such as name, email address, last log in, whether they have been locked out, and so on. You can also manage the user's password, unlock their account, disable, enable, and remove their account.
- Invite users and manage invitations. After the user accepted the invitation, you can manage their role.
  - Before inviting users to your Cortex XSOAR environment, you need to add an email integration instance, such as EWS, Gmail, EWS, Mail Sender, and so on.
- View details of actions taken in the [Audit trail](#).

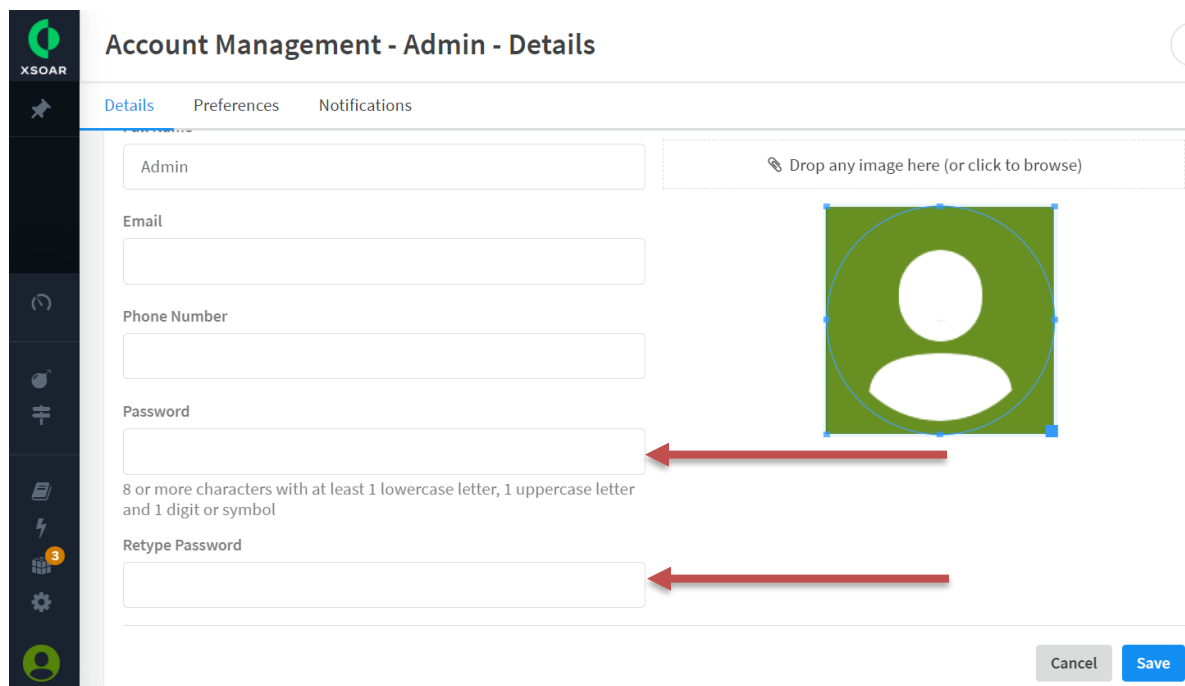
A role is a set of permissions that determine which actions and resources users within that role are granted access. Users are assigned to at least one role, depending on their required level of access.

1. Click on the [Roles](#) tab.
2. You can add as many roles as you require, by clicking [New](#).
3. The following roles are pre-defined:
  - a. Administrator – Read/write permissions for all components and access to all pages.
  - b. Analyst – Mix of read and read/write permissions for all components and access to all pages.
  - c. Read-Only – Read permission for all components and access to all pages.
4. You can view and change the following permission levels as required.
  - a. None – No access to the specified component.
  - b. Read – Can view but not edit the specified component.
  - c. Read/Write – Can view and edit the specified component.

**NOTE:** There is no concept of user for Engine as it does not have a web UI. The Engine is managed entirely by the Server. However, OS privileged user can modify its configuration file as well.

## Password Change

In the lower left-hand corner of the web UI, click on the username. Enter the password twice and click [Save](#).



The screenshot shows the 'Account Management - Admin - Details' page in the XSOAR web UI. The page has a dark sidebar on the left with various icons, including a user icon at the bottom. The main content area has tabs for 'Details', 'Preferences', and 'Notifications'. The 'Details' tab is active, showing a form with fields for 'Admin', 'Email', 'Phone Number', 'Password', and 'Retype Password'. A red arrow points from the 'Password' field to the 'Retype Password' field. To the right of the form is a profile picture placeholder with a green background and a white circle. Below the form are 'Cancel' and 'Save' buttons.

Account Management - Admin - Details

Details Preferences Notifications

Admin

Email

Phone Number

Password

8 or more characters with at least 1 lowercase letter, 1 uppercase letter and 1 digit or symbol

Retype Password

Drop any image here (or click to browse)

Cancel Save

## Password Complexity Management

You can set a password policy for all internal users. The password policy enables you to do the following:

- Set password complexity requirements.
- Set the password expiry and prevent repetition passwords (remembers up to the last 30 passwords).
- Brute-force prevention (user lockout after a number of attempts).

To create a password policy,

1. Go to [Settings > USERS AND ROLES > Password Policy](#).
2. In the [Enable Password Policy](#) section, select **On**.
3. Add the password requirement, as necessary. The 0 value disables the settings.
4. When selecting **unlock** choose one of the following options to unlock the user's account:
  - a. **By Admin only**: only administrators can manually unlock user accounts.
  - b. **Automatically**: users can unlock themselves after a specified period of time.
5. Click **Save**.

The screenshot displays the 'Account Management - Settings' interface for XSOAR. The 'USERS AND ROLES' tab is selected, and the 'Password Policy' sub-tab is active. The 'Enable Password Policy' toggle is turned 'ON'. Below this, several settings are configured:

- Minimum number of characters:** 8
- Minimum number of lowercase letters:** 1
- Minimum number of uppercase letters:** 1
- Minimum number of digits or symbols:** 1
- Password must be changed every:** 0 Months (Password never expires)
- Prevent password repetition:** ON
- Lock out user:** After 10 failed logins (within one minute)
- Unlock:** By Admin only

At the bottom right of the settings area are 'Cancel' and 'Save' buttons.

## TLS/X509 Configuration

By default, certificate path validation and revocation checking are disabled for both Server and Engine. This setting can be added in the web UI from the [About > Troubleshooting](#) tab and clicking on [Add Server Configuration](#). Enter key and value and click [Save](#).

The screenshot shows the 'Account Management - Settings' page in the XSOAR web UI. The 'ABOUT' tab is selected, and the 'Troubleshooting' sub-tab is active. The 'Server Configuration' section contains several input fields: 'https\_proxy', 'https\_proxy', 'Base URL (for D2 Agents and Engines)' with a default of '10.8.48.90:443, 172.17.0.1:443' and a value of 'server.test.local:443', 'External Host Name (for notifications such as e-mail and Slack)' with a default of 'server:443' and a value of '10.8.48.90:443', and a key-value pair 'security.server.certificate.donotverifyclientcertificates' with a value of 'false'. A '+ Add Server Configuration' link is present. The 'Time Configuration' section shows 'Date format' set to 'Default' with a dropdown, and '12 hours' selected with a radio button, and 'Display timezone' set to 'Local timezone' with a dropdown. A 'Cancel' button and a 'Save' button are at the bottom right. A notification at the top right states '3 Content updates are available in the Mar'.

Account Management - Settings 3 Content updates are available in the Mar

ACCOUNT MANAGEMENT INTEGRATIONS OBJECTS SETUP USERS AND ROLES ADVANCED **ABOUT**

Version License **Troubleshooting**

### Server Configuration

https\_proxy

https\_proxy

Base URL (for D2 Agents and Engines)  
Default: 10.8.48.90:443, 172.17.0.1:443  
server.test.local:443

External Host Name (for notifications such as e-mail and Slack)  
Default: server:443  
10.8.48.90:443

security.server.certificate.donotverifyclientcertificates false

+ Add Server Configuration Cancel Save

### Time Configuration

Date format **Default** 12 hours 24 hours

Display timezone **Local timezone**

You can also configure these values in the configuration files on Server and Engine. Here are the X509v3 certificates settings.

Key	Description	Default Value	Possible Values
security.certificate.override.donotverifyservercertificates	Globally enable/disable server certificate verification in client connections	true	true, false
security.server.certificate.donotverifyclientcertificates	Globally enable/disable client certificate verification in server connections	true	true, false
server.certificate.clientauthpolicy	Should the server require/verify client certificates*	""	"", "require", "verify", "requireandverify"
security.certificate.verifyclientsan	Verify that the SAN field in client certificate matches trusted list of SANs defined below	false	true, false
security.certificate.santrustlist	Defines trusted list** of SANs	""	comma separated list (for example: "localhost,test.abc.com")
security.certificate.verifyrevocation	Verify that the certificate is not revoked (supports for OCSP only)	false	true, false
security.certificate.failonunknownrevocationstatus	Block the connection if the certificate revocation status cannot be determined	true	true, false
security.client.enabled	Enable use of client certificate	false	true, false
security.client.certfile	Location of the client certificate	"/usr/local/demisto/client.pem"	"/<filepath>"
security.client.keyfile	Location of the client private key	"/usr/local/demisto/client.key"	"/<filepath>"
security.certificate.client.keypass <sup>4</sup>	Passphrase (in cleartext) of the client private key	""	string
security.keypass.encrypt	Encrypt the passphrase in the DB	false	true, false
security.certificate.verifycasign	Verify CA certificates have caSign	false	true, false
security.certfile	Location of the server certificate	"/usr/local/demisto/cert.pem"****	"/<filepath>"

<sup>4</sup> Sometime the private key is encrypted. For example, using an export password. If that is the case, the export password must be stored in the keypass.

security.keyfile	Location of the server private key	"/usr/local/demisto/cert.key"****	"/<filepath>"
security.keypass	Passphrase of the server private key***	""	string
engine.connection.trust_any_certificate	Tell engine to trust any server certificate.	true	true, false

\* "" = does not require client certificate, "require" = require client certificate but do not verify it, "verify" = do not require client certificate but if one is presented, verify it, "requireandverify" = require client certificate and verify it

\*\* - Blank santrustlist means nothing is allowed. No space between comma. String must be in double quotes but a Boolean does not.

\*\*\* - Engine certificate private key (server side) is configured from the engine configuration page in the WebUI:

The screenshot shows the 'Settings' page in the Cortex XSOAR WebUI. The 'Engines' tab is selected under the 'INTEGRATIONS' section. A table lists engines, with 'eng' selected. A red box highlights the 'eng' engine, and a red arrow points to the 'Edit Configuration' button. A modal window titled 'Edit Engine Configuration' is open, showing the 'JSON formatted configuration' field and a checkbox labeled 'Use a passphrase for the engine certificate private key' which is checked. Below this, there is a field for 'Passphrase for the certificate private key' with a red box around it.

\*\*\*\* - Default server certificate filenames in Engine are "d1.pem" and "d1.key" respectively.



Here are some examples of Engine and Server configuration files, respectively. Please note this is for demonstration of syntax only<sup>5</sup>. The keys and values may differ based on customers' environment and requirement needs.

### Engine configuration file

**sudo less /usr/local/demisto/d1.conf**

```
{
  "AgentURLs": [
    "wss://server.test.local1:443/d2ws"
  ],
  "ArtifactsFolder": "",
  "BindAddress": ":443",
  "EngineID": "ff1dea73-8b22-437a-adb8-391514c05cfa",
  "EngineURLs": [
    "wss://server.test.local1:443/d1ws"
  ],
  "LogFile": "/var/log/demisto/d1.log",
  "LogLevel": "info",
  "Server.ExternalHostName": "server.test.local1:443",
  "security.certificate.override.donotverifyservercertificates": false,
  "security.certificate.verifyrevocation": true,
  "security.certificate.failonunknownrevocationstatus": true,
  "security.client.enabled": true,
  "engine.connection.trust_any_certificate": false,
  "ServerPublic":
  "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAXyezs/7WeffS26YUzMNxVOYKMEZ0urLfFVOj/1HzOBw3+215qlh4
  iWJpgV5UHGXpSYxo4mzxwDtvle+P/ZQTk89kIJ5xQ6QuGbU8tPjNiQZ0aLQMWmn79+Emj3OZMo9gFjMJF16LSOAZPZQZ6P8
  RDbdDED/c4/wzJ1dfKLKnadwNcd04ygvP9gYNQNhxVLoDM2Sy0OD+mKEtCiKraF1NJxEba+EV04L3kTwHXsJtYJVOaYrSlrG/+j
```

---

<sup>5</sup> Misspelled, typo, or missing quotes words will ensure the default behavior.

3LDHKiko8+pZX8OWJd/0TCUYzj8Cn0orINqPIKXH4pM885DXvpnoAQfgOi6O9kmp1RvjTiHTyit5EhBSQTd7I50w/KuRyHDPwl  
DAQAB",

"TempFolder": "",

"container": {

"engine": {

"type": "docker"

}

},

"powershell": {

"engine": {

"docker": true

}

},

"python": {

"engine": {

"docker": true

}

}

}

### Server configuration file

***sudo less /etc/demisto.conf***

```
{
  "Server": {
    "HttpsPort": "443",
    "ProxyMode": true
  },
  "security.server.certificate.donotverifyclientcertificates": false,
  "server.certificate.clientauthpolicy": "verify",
  "security.certificate.verifyrevocation": true,
  "security.certificate.failonunknownrevocationstatus": true,
  "security.certificate.verifyclientsan": true,
  "security.certificate.santrustlist": "engine.test.local1,good.test.local1",
  "container": {
    "engine": {
      "type": "docker"
    }
  },
  "custom": {
    "fields": {
      "validate": {
        "grid": {
          "values": true
        }
      }
    }
  }
}
```

```

},
"db": {
  "index": {
    "entry": {
      "disable": true
    }
  }
}
}

```

**NOTE:** All strings are in double quotes and there is a space between the colon (:) and value.

### **Cannot Establish Connection to Determine Validity of Certificate**

By default, if the TOE cannot establish a connection to determine the validity of the certificate, the TLS connection will fail. However, this is configurable as specified in the table above (see failonunknownrevocationstatus).

### **Expected Reference Identifiers**

On the Engine,

"Server.ExternalHostName": "<Expected Server Name>:443",

On the Server,

"security.certificate.verifyclientsan": true,

"security.certificate.santrustlist": "<List of Expected Trusted Clients>",

## Storing CA certificates in Linux

The Root CAs are stored in Linux keypair (its trust CA anchor store). Extension should be .crt as recommended.

Function	Method
Add	<ol style="list-style-type: none"><li>1. Copy your CA to dir /usr/local/share/ca-certificates/<sup>6</sup> Use command: <code>sudo cp foo.crt /usr/local/share/ca-certificates/foo.crt</code></li><li>2. Update the CA store: <code>sudo update-ca-certificates</code></li></ol>
Remove	<ol style="list-style-type: none"><li>1. Remove your CA.</li><li>2. Update the CA store: <code>sudo update-ca-certificates --fresh</code></li></ol>

**NOTE:** All the CA certificates must be stored in the Linux trust anchor. For example, if the server certificate is signed and issued by Root CA -> Sub CA. Both the CA certificates must be stored in the Linux trust anchor.

## Configuration Tricks and Tips

- Every configuration change requires a restart of the Server or Engine (e.g., on Ubuntu, `sudo service demisto restart` or `sudo service d1 restart`) depending on where the change is made.
- All x509v3 certificates must contain SAN field or TLS connection will be terminated. CN field is not checked, and wildcard is not supported.
- Server key and certificate (cert.key and cert.pem) must have 400 permission and demisto ownership. If you replace the cert or keys, make sure permission is 400 with demisto ownership, and restarts the server/engine.
  - `sudo chown demisto:demisto /usr/local/demisto/cert.*`
  - `sudo chmod 400 /usr/local/demisto/cert.*`
- Engine client key and certificate (client.key and client.pem) must have the same as above.
- The whole chain must be in the certificate. For example, `cat server.cert.pem subca.cert.pem Root.cert.pem > cert.pem`
- For revocation checking, the calssuers must be included in the CA certificate's AIA field along with OCSP information. For example,
  - `calssuers;URI.0 = http://<hostname or IP>/<CA Issuer certificate>`
- The OCSP responder certificate must have the OCSPsign in the Extended Key Usage (EKU) field.
- A certificate must have a Subject Alternative Name SAN field, or the TLS connection will be terminated.

---

<sup>6</sup> On Redhat 8 or later, the directory is /usr/share/pki/ca-trust-source/anchors and the command is "update-ca-trust"

- The Root certificate MUST NOT be an OSCP responder. CA certificates with OCSPsign in the EKU field will be rejected.
- CA certificates must include caSigning purpose in the Key Usage (KU) field.
- Server certificates must have ServerAuth in the EKU field and client certificates must have ClientAuth in the EKU field.
- Certificate with MD5 hash is not supported. SHA-1 is supported for verification only. Recommend using SHA-256 or higher.
- Signature algorithms RSA 2048 or higher and ECDSA 256 or higher are supported.
- When generating ECDSA key, you must use “-noout” to prevent explicit parameters from being included in the ECDSA private key. Otherwise, the private key cannot be parsed, and the service will fail.
  - `openssl ecparam -name secp384r1 -noout -genkey -out private/ECDSAserver.key.pem`

### Example of client certificate ###

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 4096 (0x1000)

Signature Algorithm: sha256WithRSAEncryption

Issuer: CN = RSA-SubCA

Validity

Not Before: Oct 16 14:27:24 2021 GMT

Not After : Oct 14 14:27:24 2031 GMT

Subject: CN = engine.test.local1

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public-Key: (3072 bit)

Modulus:

00:c1:16:fb:36:23:01:2e:70:34:78:83:f4:9c:4d:

b5:21:38:be:de:7e:30:72:6e:f2:48:1a:13:69:34:

2b:81:18:26:8f:a4:e3:3d:ea:c6:94:40:10:7a:a8:

9f:76:1c:7c:0f:47:77:35:69:26:47:3e:8c:6c:cb:

97:23:af:5d:82:2b:f0:f2:2b:40:dd:d8:ff:f9:d9:

a5:ba:a5:05:99:a2:57:5a:bb:02:61:8e:6e:4d:d9:  
44:7c:18:19:9d:30:94:70:56:0a:c6:2b:78:8c:5a:  
8b:e3:8e:2f:82:7f:39:ab:f7:3b:11:02:c8:26:de:  
ee:3c:7a:30:1e:05:35:18:61:b3:90:4a:fc:5d:c9:  
ae:f1:5b:58:bf:a7:49:6c:8c:f4:59:b7:d6:d1:9e:  
77:9c:64:3e:3b:15:14:0f:e2:90:2b:c2:0e:bb:ca:  
59:5f:f0:b6:0d:7c:bf:58:c5:ce:41:f1:0d:87:d9:  
92:7c:ac:61:5a:6a:db:96:d7:89:9c:0b:a9:1f:6c:  
6a:58:a3:ac:2d:ad:30:37:54:80:4f:64:79:9b:49:  
73:bf:0a:58:28:f9:b0:b6:35:02:df:68:79:80:e3:  
bd:ca:6e:5d:dd:21:a1:0e:88:01:6a:b6:7d:35:41:  
d3:07:b4:ee:cb:ef:5a:b9:d3:d0:8a:e9:e0:fc:e8:  
bc:e3:54:1e:2f:8b:14:f9:d8:b9:8d:c1:cf:35:a4:  
29:b5:e5:5a:dd:99:2d:c9:4e:e1:31:e3:19:30:af:  
40:52:4a:9f:0a:8f:5b:fe:a2:ef:46:13:0a:5b:f2:  
90:21:1f:0f:ae:fe:b2:cf:8d:c5:32:b0:31:8a:65:  
6f:4f:d2:1f:1b:6b:0a:0e:c8:15:5c:4d:1d:8f:b5:  
df:7d:77:7d:85:d1:96:8f:cf:3e:ba:d5:5c:c9:ef:  
44:d1:40:a6:5c:81:29:e0:c3:f3:a3:af:0d:b2:25:  
41:eb:33:95:a1:49:89:78:ea:be:8d:d5:b2:d2:d9:  
cd:9e:88:51:35:83:75:4d:22:41

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Basic Constraints:

CA:FALSE

Netscape Cert Type:

SSL Client

Netscape Comment:

OpenSSL Generated Server Certificate

X509v3 Subject Key Identifier:

F9:20:94:9F:C8:F7:8F:87:6D:B2:33:7B:F0:C5:60:43:F1:A8:D3:B8

X509v3 Authority Key Identifier:

keyid:44:9C:4F:C2:2C:35:CD:5D:24:AB:6B:A9:3D:8C:E2:F6:22:29:2D:AA

X509v3 Extended Key Usage:

[TLS Web Client Authentication](#)

X509v3 Key Usage:

Digital Signature, Non Repudiation, Key Encipherment

Authority Information Access:

[CA Issuers - URI:http://10.8.48.90/RSAsubca.cert.pem](#)

OCSP - URI:http://10.8.48.90:8888

X509v3 Subject Alternative Name:

[DNS:engine.test.local1](#)

Signature Algorithm: sha256WithRSAEncryption

b9:25:87:e9:30:e5:ff:78:91:f1:35:23:9e:9d:eb:4c:b9:f8:  
62:fd:9b:3f:c9:37:75:00:70:16:fc:e1:58:63:19:cb:a2:05:  
c4:aa:40:f4:ed:7e:ff:6d:8c:dc:ff:26:eb:68:47:a0:ba:1e:  
6d:82:d9:7d:26:9a:d1:9a:13:bc:26:55:5a:23:dd:a1:f9:40:  
19:6b:b2:50:9b:bf:bd:88:fd:80:81:f5:2d:d4:7e:87:20:9d:  
c4:39:3b:85:fa:3c:3a:b4:90:a9:97:97:02:5c:b8:7a:2f:82:  
3f:db:cf:d8:e3:12:6a:86:2f:eb:a1:2b:2a:3b:2b:67:a4:9b:  
ec:48:b7:8d:de:b6:b1:ac:65:c1:de:de:b5:38:c4:59:9e:5f:  
ed:3d:23:6b:f6:d9:df:ef:fd:cb:69:1c:1c:38:e9:e8:28:f3:



bc:31:9b:ee:28:13:1f:39:b5:74:f7:86:80:59:a8:15:a8:75:  
51:c2:42:23:7a:7e:02:0b:34:4d:c5:ff:cf:9e:61:a2:91:b6:  
a9:f0:b2:13:af:eb:bf:b9:c9:5f:08:a4:44:4d:9a:cd:67:6a:  
44:48:e3:0d:0c:7e:38:ba:7e:e5:cf:76:43:21:98:60:7d:83:  
07:1a:7b:33:50:90:e1:c6:ac:a9:7c:1c:60:e3:db:89:5b:8f:  
8b:b8:40:58:8e:dc:e7:63:36:6d:90:99:4f:46:96:37:e6:26:  
10:53:4b:f7:e1:0b:85:b5:cc:b4:23:d3:4c:b8:95:af:c4:b8:  
da:d2:f1:cf:3c:e5:df:96:de:10:89:23:6a:07:01:7c:fc:76:  
ba:8a:0c:02:77:fd:31:29:81:c4:42:fb:36:1f:b7:85:04:aa:  
bc:a2:a9:58:25:1e:47:06:9e:4b:16:f5:85:9b:42:11:7c:9f:  
68:c5:3f:df:a8:4e:2c:5f:e6:99:78:69:85:33:f2:a9:46:2e:  
13:de:b3:d1:88:9a:0e:ad:cf:8a:28:7c:13:a1:d1:0c:16:fd:  
5c:b2:23:c2:2d:07

## Configure Mutual Authentication for the Web UI

By default, the web UI supports server-side authentication only. To configure mutual authentication for the web UI, the client certificate must be imported into the web browser.

In the `/etc/demisto.conf`, set

```
"security.certificate.override.donotverifyclientcertificates": false,  
"server.certificate.clientauthpolicy": "requireandverify",
```

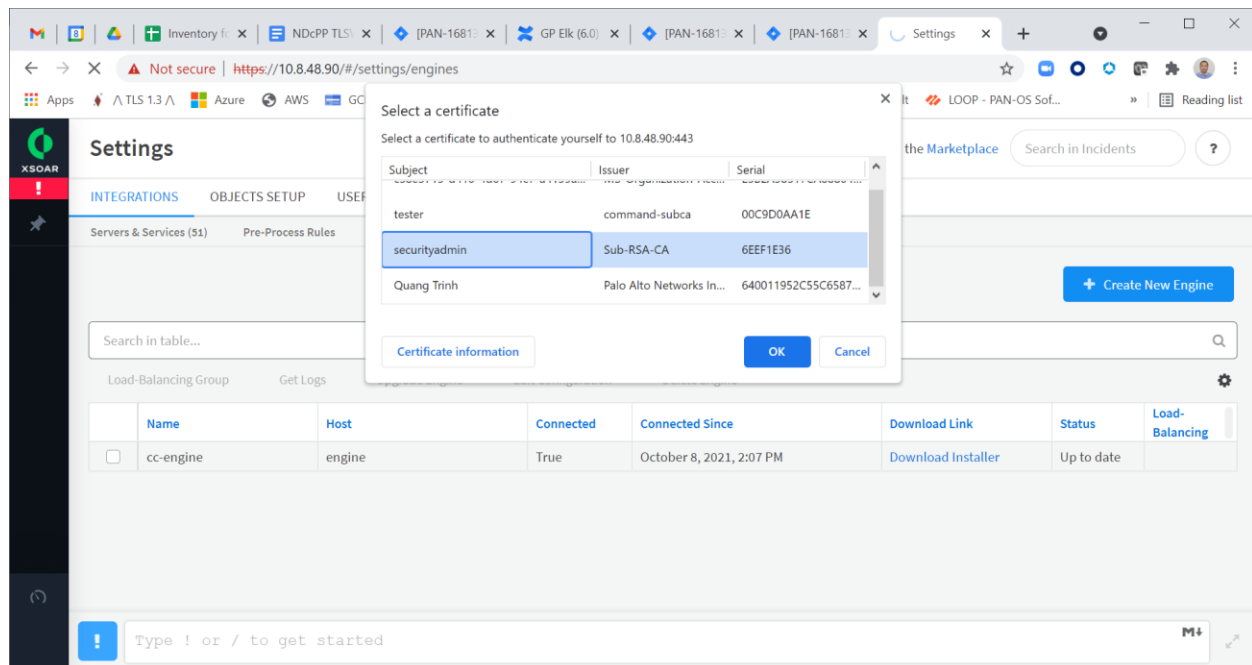
Restart the demisto server

Move the CA certificates into the Linux trust anchor.

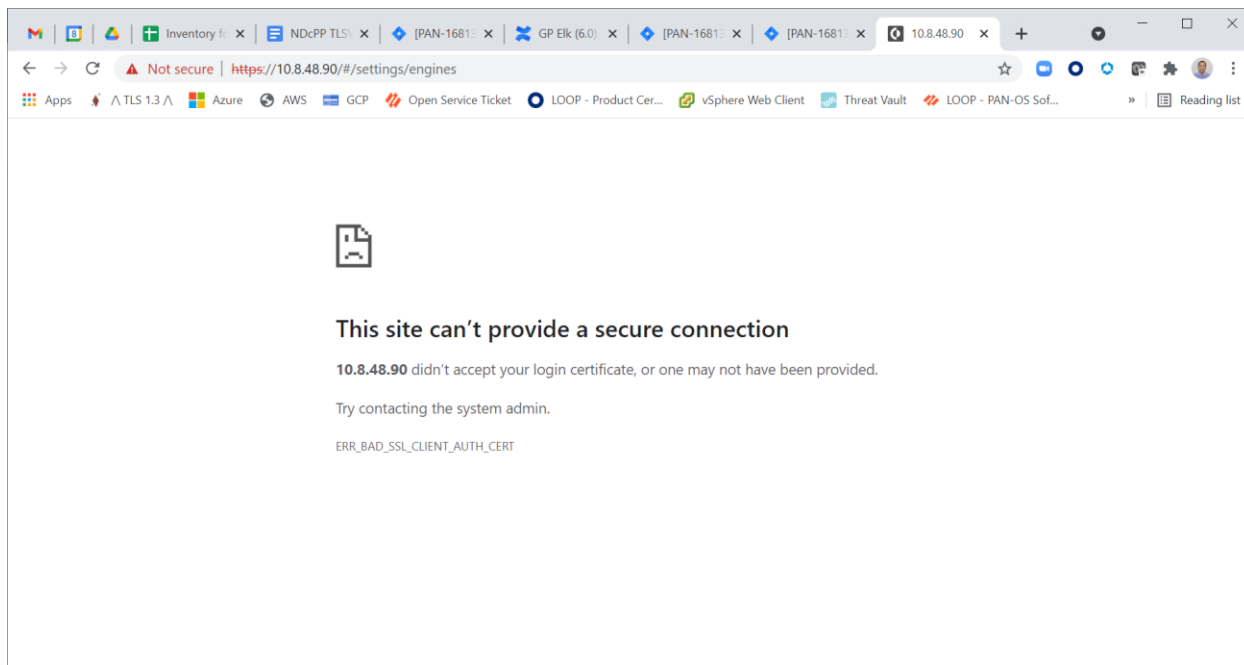
**NOTE:** This applies to the web UI connection and the Engine connection as well.

##### On the web UI, you are asked to provide a client certificate #####

Provide the wrong client cert.



The connection fails as expected.



### Now import the correct certificate and key into the web browser

### We need to convert the certificate and key into PKCS12 format (from pem format)

***openssl pkcs12 -export -clcerts -in <client.cert.pem> -inkey <client.key.pem> -out client.p12***

*Enter Export Password:*

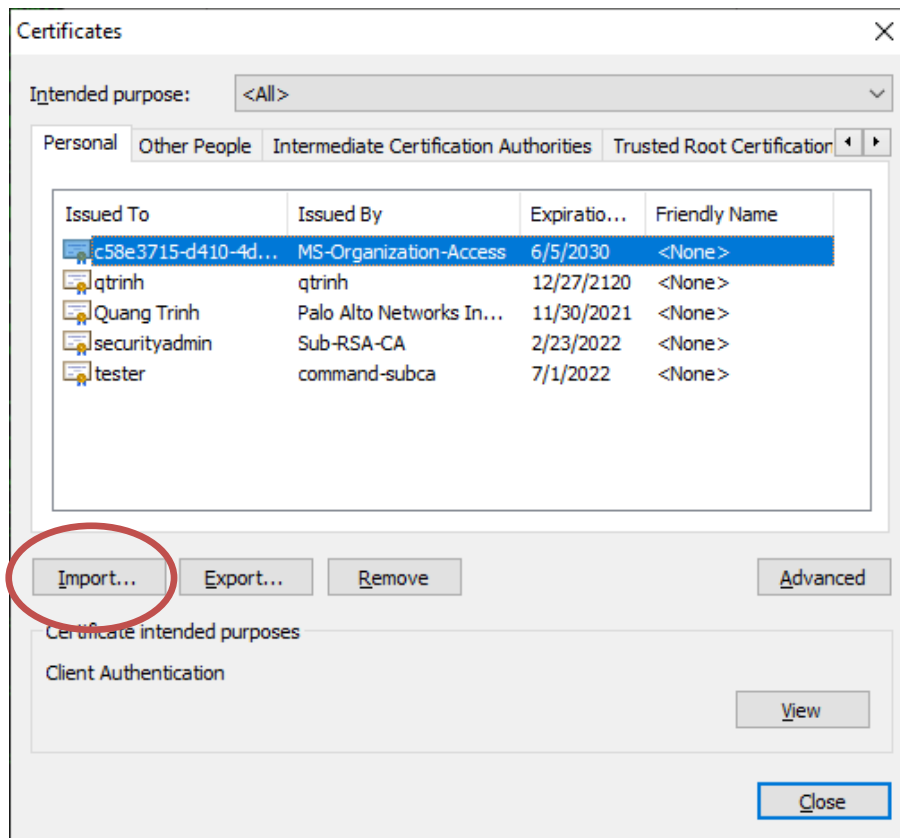
*Verifying - Enter Export Password:*

### SCP the client.p12 to the desktop and import into web browser of your choice

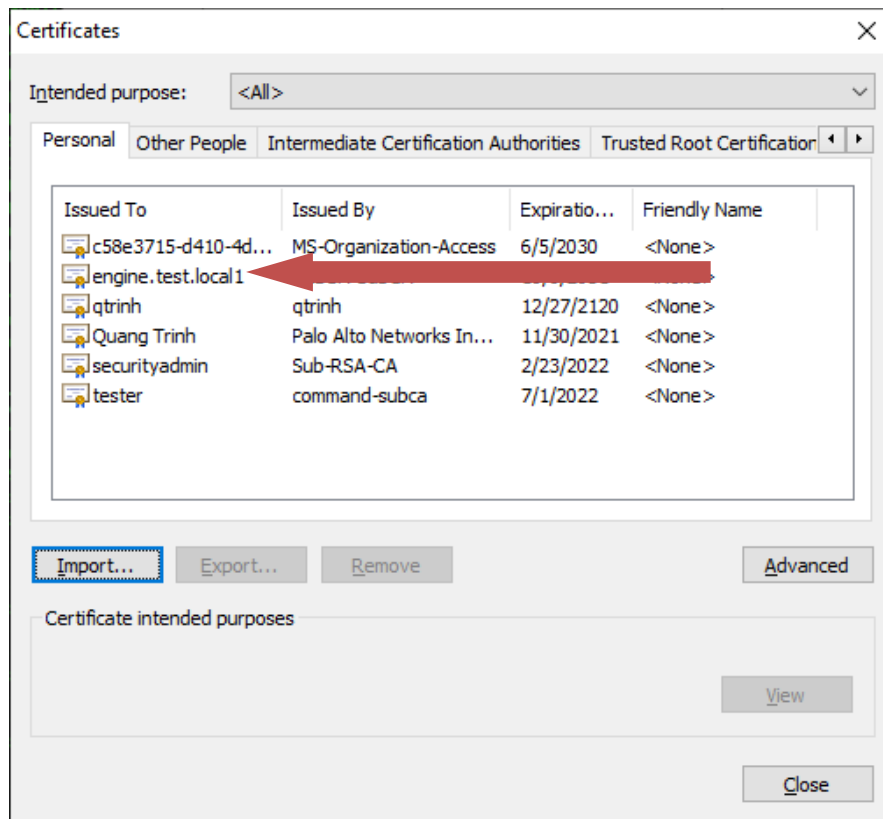
Using Chrome as the web browser.

1. Go to

**Settings > Privacy & Security > Security > Manage Certificates > Import...**

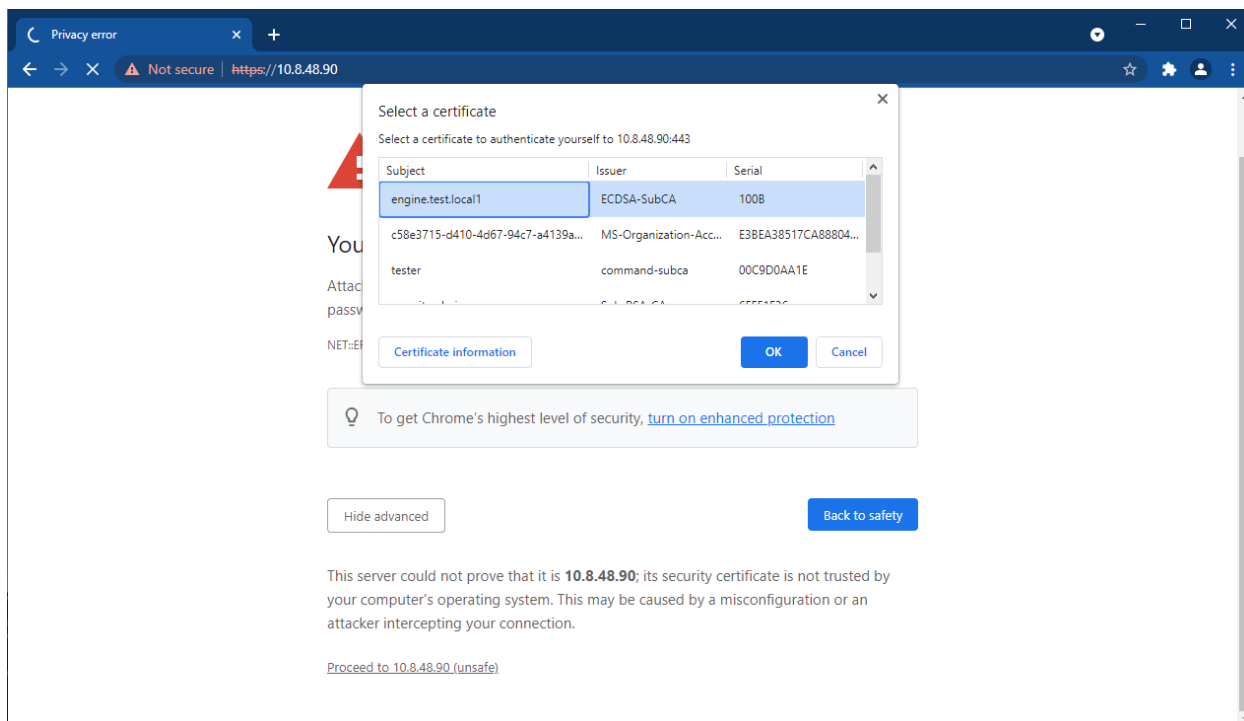


2. Click **Next**  
Select the file to import, for example client.p12 and enter the export password.
3. Click **Next**  
Place the certificate in the Personal Certificate store.
4. Click **Next**
5. Click **Finish**. Should get message "The import was successful". Click **OK**.

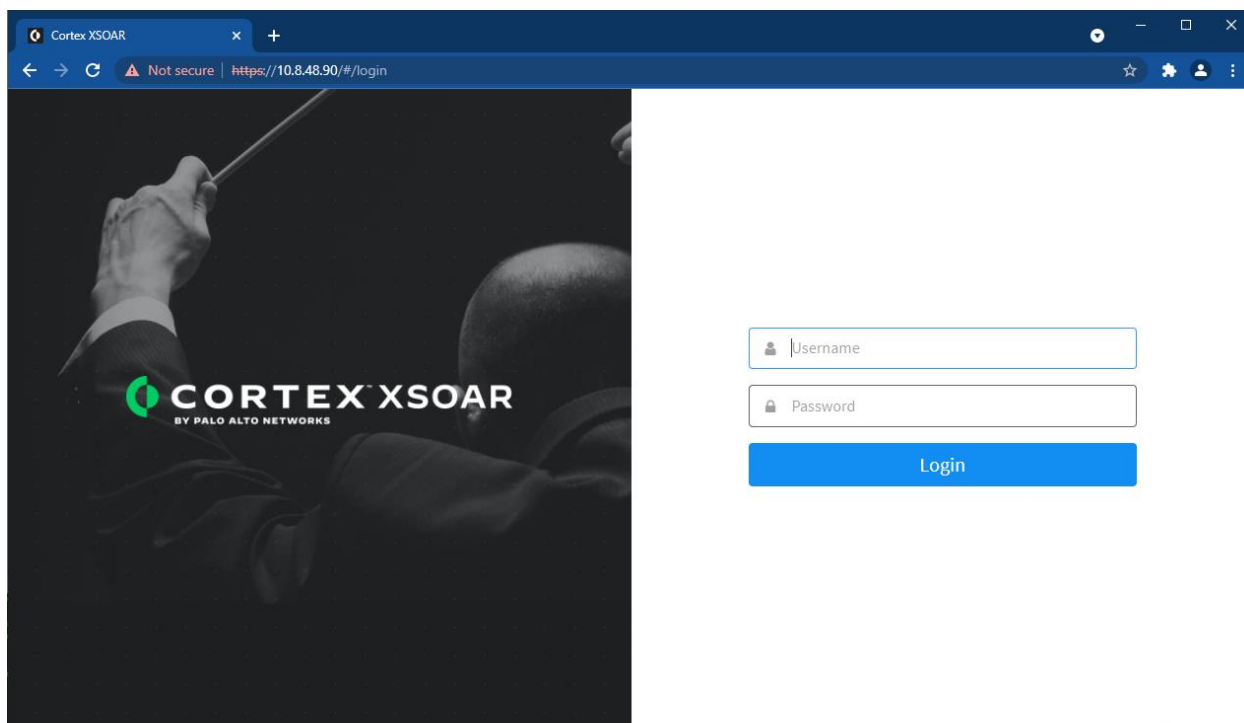


6. Click [Close](#).

#### Try to connect with that client certificate



The connection should be successful.



## Configure the Sensitive Data to be Encrypted

To set the API key value to be encrypted.

1. Connect to the XSOAR web UI.
2. Go to [Settings > Integrations > +BYOI](#).
3. Change the integration name.
4. Save the integration. Click [Save Version](#).
5. Click [Update Integration](#).
6. Click [Add Instance](#).
7. Set API key to any value.

HelloWorldsfgbsgd

Mapper (incoming) [Select](#)

Server URL (e.g. <https://soar.monstersofhack.com>) \*

<https://soar.monstersofhack.com>

Maximum number of incidents per fetch

10

API Key \*

.....

Score threshold for IP reputation command ?

65

Score threshold for domain reputation command ?

65

Fetch alerts with status (ACTIVE, CLOSED)

ACTIVE x

☐ Delete

[Test](#) [Cancel](#) [Save & exit](#)

[Help](#) [Test results](#)

### Community Contributed Integration

Integration Author: Cortex XSOAR

No support or maintenance is provided by the author. Customers are encouraged to engage with the user community for questions and guidance at the [Cortex XSOAR Live Discussions](#).

### Hello World

- This section explains how to configure the instance of HelloWorld in Cortex XSOAR.
- You can use the following API Key: `43ea9b2d-4998-43a6-ae91-aba62a26868c`

[View Integration Documentation](#)

8. [Save & exit](#).

**NOTE:** To encrypt the passphrase used to encrypt the private key, set `security.keypass.encrypt` to [true](#).

## Update the TOE Version

The installer automatically detects the existing configurations and applies them to the upgraded TOE. Optionally, take a snapshot of the TOE or back up your content by selecting [Settings > About > Troubleshooting > Export](#).

1. Download the new installer and copy it to all the Linux servers that will be upgraded.  
**wget -O demisto.sh "<downloadLink>"**
2. If you are deploying Cortex XSOAR using a signed installer (GPG), you need to import the GPG public key that was provided with the signed installer.

For example, you can use the **rpm -import <public.key>** command to import the public key into the local GPG keyring.

3. If you are deploying Cortex XSOAR using a signed installer (GPG) you might need to manually install the makeself package by running the **yum install makeself** command.
4. Run the **chmod +x demisto.sh** command to convert the file to an executable file. (If Installer is signed, use **rpm -K** to verify signature before installing.)
5. Execute the command **sudo ./demisto.sh**.