# ThreatConnect® Installation Guide: Linux® Operating System

## Software Version 7.3.3

Technical Guide

December 19, 2023

10015-35 EN Rev. A

ThreatConnect, Inc.
3865 Wilson Blvd., Suite 550, Arlington, VA 22203

P: 1.800.965.2708 | F: 703.229.4489
www.ThreatConnect.com

# Table of Contents


**System Requirements** ..... **6**

Hardware ..... 6

Software ..... 8

Database ..... 8

SMTP Server ..... 9

Inbound Email ..... 9

Web Browsers ..... 9

Whitelist Requirements ..... 10

**OpenSearch Installation** ..... **10**

Hardware ..... 10

Software ..... 10

Additional Requirements ..... 11

Network Configuration ..... 11

Downloading and Installing OpenSearch 2.6.0 ..... 11

Installing OpenSearch From RPM ..... 11

Verification ..... 13

Configuration ..... 13

Plugin Installation and Starting OpenSearch ..... 15

Plugin Installation ..... 15

Starting the OpenSearch Service ..... 15

**MySQL Installation and Configuration** ..... **17**

MySQL Download and Installation ..... 17

MySQL Configuration Options ..... 19

Port Configuration ..... 19

Creating the Database and Users ..... 20

Creating ThreatConnect Tables ..... 21

**ThreatConnect Installation and Configuration** ..... **22**

Preparing the ThreatConnect Application Server ..... 22

Open File Limits for a Standalone ThreatConnect Application Server ..... 22

Open File Limits for an All-in-One ThreatConnect Application Server ..... 23

JDK ..... 23
User Account Creation ..... 24
Port Forwarding ..... 25
Redis Server Installation and Setup (for RHEL 7) ..... 26
Redis Server Installation and Setup (for RHEL 8) ..... 29
Python 3.6 Installation ..... 32
Python 3.11 Installation ..... 34
Installing Reporting Dependencies ..... 37
Getting Started ..... 38
Downloading the Installer ..... 38
The ThreatConnect Folder ..... 39
The Customer Folder ..... 40
Opening the Installer ..... 40
Unzipping the File ..... 40
Installation Directory Structure ..... 41
Folder Permissions Adjustment ..... 41
TC Exchange Setup ..... 42
Creating “tc-job” User ..... 42
User Privilege Configuration ..... 42
Configuring Sudoers ..... 43
Starting the Installer ..... 43
Installation Basics ..... 43
Time Zone ..... 43
Initial Setup ..... 44
Selecting Appropriate Server Type ..... 44
Entering a Server Name ..... 45
Configuring Database Properties ..... 45
Configuring SSL Properties ..... 46
Configuring SMTP Properties ..... 46
Configuring Memory Properties ..... 47
Configuring a Full Server ..... 47
Configuring a Playbooks/Job Server ..... 47
Configuring the ThreatConnect License ..... 48

Enabling SNI ... 48
**Starting ThreatConnect ... 49**
The Service Script ... 49
Configuring Services ... 50
Starting ThreatConnect as a Service ... 51
**The First Login ... 52**
Entering Initial Login Credentials ... 52
Setting Master Key for Keychain ... 52
Installing the License ... 53
System Settings Checklist ... 55
Configuring OpenSearch on ThreatConnect ... 57
Creating an Organization ... 59
**Appendix A: PostgreSQL Installation and Configuration ... 60**
PostgreSQL Download and Installation ... 60
Port Configuration ... 60
PostgreSQL Access Configuration ... 61
PostgreSQL Configuration Options ... 62
Creating the Users and the Database ... 62
Creating ThreatConnect Tables ... 63
Creating Cast Tables ... 64
**Appendix B: Re-Authenticating the Keystore and SSL Certificate Configuration ... 67**
Re-Authenticating the Keystore ... 67
SSL Certificates ... 67
Generating a Self-Signed SSL Certificate ... 67
Generating a CA-Signed SSL Certificate ... 68
**Appendix C: Configuring SSL Transmission with MySQL and ThreatConnect ... 72**
**Appendix D: Integrating Self-Signed Certificates with Java and Python Apps ... 75**

**Important**: Viewing this document in macOS® Preview may omit and distort some of the formatting. Therefore, it is recommended to view it in a web browser, such as Google Chrome™, or another PDF reader.

# System Requirements

To install an On-Premises instance of ThreatConnect, the requirements in the following sections must be met.

## Hardware

ThreatConnect requires a server, virtual or physical, that meets the specifications listed in Tables 1–5.

**Note**: Multi-server installations are for advanced users only, who should consult with ThreatConnect as to the correct sizing that will meet their needs. See *ThreatConnect System Requirements* for additional information.

**Table 1**

| | Playbooks | Memory Min (GB)[1,2] | Min CPU Cores / vCPUs (2GHz)[2] | Estimated Storage (GB)[3,4] |
|---|---|---|---|---|
| **ThreatConnect Application Server** | No | 16 | 8 | 50 |
| | Yes | 48 | 8 | 150 |

[1] Allocated to TC; OS needs extra. Large single sources (a source with 2+ million Indicators) may require more memory for bulk processing.

[2] Plus cores and memory indicated by installed apps. (This memory is not allocated to TC in the configuration, because apps run in their OS process and require their memory.)

[3] High IOPS, ideally SSDs, are preferred.

[4] ThreatConnect must be installed on an ext4 or XFS partition when running Playbooks.

### Table 2

| | Indicators (Millions) | Memory Min (GB)[1] | Min CPU Cores / vCPUs (2GHz) | Estimated Storage (GB)[1] |
|---|---|---|---|---|
| **Database Server** | 0–2 | 12 | 6 | 20 |
| | 2–5 | 16 | 8 | 40 |
| | 5–10 | 32 | 12 | 60 |
| | 10+[2] | 64 | 16 | 120 |

[1] Allocated to the database; OS needs extra.

[2] Recommended allocations for 10+ million Indicators will work best in a Postgres® database server configuration.

### Table 3

| | Indicators (Millions) | Memory Min (GB) | Min CPU Cores / vCPUs (2GHz) | Estimated Storage (GB) |
|---|---|---|---|---|
| **OpenSearch® Server** | 0–2 | 12 | 6 | 20 |
| | 2–5 | 16 | 8 | 40 |
| | 5–10 | 32 | 12 | 60 |
| | 10+ | 32 | 12 | 60 |

### Table 4

| | |
|---|---|
| **Document Storage** | Equal to the desired capacity of documents stored |

### Table 5

| | Memory Minimum (GB) | Memory Recommended (GB) |
|---|---|---|
| **Swap Space** | 4 | 8 |

**Note**: As the number of users increases, or as the frequency or complexity of automated analysis increases, the need to increase system resources will likely occur.

# Software

ThreatConnect and its supporting packages require the following software to run correctly:

- **Operating System:** Red Hat® Linux variant—either Red Hat Enterprise Linux® (RHEL) 6, 7, or 8 or Community Enterprise Operating System (CentOS™) 6 or 7
- **Java® 11 (OpenJDK or Oracle® version 11):** Access to a local installation of Oracle Java 11 or OpenJDK (JDK version 11)
- **OpenSearch**: OpenSearch Server 2.6.0
- **Python®**: Installation of Python 3.6.*x* and Python 3.11.*x* is required

  **Important**: This requirement refers to CPython. No other type of Python is permitted.

- **Python SDK:** TcEx CLI 1.*x*

  **Important**: Starting with ThreatConnect version 7.2, a specific version of TcEx should no longer be installed. Instead, install the TcEx CLI package, which provides the necessary functionality for App Builder. For instructions on installing the TcEx CLI package with Python 3.6 and Python 3.11, see the "Python 3.6 SDK: TcEx Installation" and "Python 3.11 SDK: TcEx Installation" section, respectively.

- **Redis®**: Installation of Redis 6.2.6
- **Select and install one of the following databases**:
  - MySQL®: Installation of MySQL 8.0.*x* Community or Enterprise Edition
  - SAP HANA®: Installation of SAP HANA 2.0 SPS 02
  - PostgreSQL®: Installation of PostgreSQL v14

  **Important**: Be aware that, depending on the distribution of ThreatConnect to be installed, users may not have the option to use MySQL as the database.

# Database

ThreatConnect requires an available instance of MySQL 8.0, PostgreSQL v14, or SAP HANA 2.0 SPS 02 database backend. For MySQL and PostgreSQL, a client connection requires permissions to create users, databases, and tables within this instance during the installation process. Also, while it is acceptable to run one instance of the database on the same server as ThreatConnect, clients are advised to instantiate another machine for the

replicated database. It is recommended that these machines conform to the hardware specifications provided earlier in this guide.

## SMTP Server

ThreatConnect requires an available Simple Mail Transfer Protocol (SMTP) server to send email alerts and to correspond with users. This server must be routable from the server running the platform, and if SMTP authorization is required, ThreatConnect will need access to a username and password to generate these emails.

During ThreatConnect installation, you will be given the option to select Secure Sockets Layer (SSL) or Transport Layer Security (TLS) in addition to providing credentials for user authentication. These options can be set to **False** if they are not applicable to your SMTP configuration.

## Inbound Email

To allow inbound email functionality to the platform and messages to be routed to the ThreatConnect application host, an MX record pointing to the desired mail domain (e.g., **tcsoar.mydomain.com**) will need to be configured. Doing so will enable functionality for feed mailboxes, phishing mailboxes, and Playbooks that use the Mailbox Trigger. See *Handling Incoming Emails* for more information on configuring email ingestion in ThreatConnect.

When configuring the **mailInboundDomain** system setting for ThreatConnect, set the value to the appropriate host for which the DNS MX record is configured. See *ThreatConnect System Administration Guide* for more information.

## Web Browsers

The ThreatConnect platform supports up to two versions behind the current stable release of the following Web browsers:

- Google Chrome
- Mozilla Firefox®

## Whitelist Requirements

Whitelist **api.threatconnect.com:443**, **broker.threatconnect,com:443**, and **feeds.threatconnect.com:443** to ensure that the ThreatConnect application will be able to communicate with all necessary subdomains.

# OpenSearch Installation

## Hardware

The hardware listed in Table 6 is recommended for a server to run OpenSearch and its supporting plugins.

**Table 6**

| | Indicators (Millions) | Memory (GB) | Min CPU Cores / vCPUs (2GHz) | Estimated Storage (GB) |
|---|---|---|---|---|
| **OpenSearch Server** | 0–2 | 12 | 6 | 20 |
| | 2–5 | 16 | 8 | 40 |
| | 5–10 | 32 | 12 | 60 |
| | 10+ | 32 | 12 | 60 |

## Software

OpenSearch and its supporting plugins require the following software environment to run correctly:

- Operating System (OS): Any OS supported by the vendor
- Java 11 (OpenJDK or Oracle version 11): Access to a local installation of the JDK (version 11)
- OpenSearch: OpenSearch Server 2.6.0

**Note**: ThreatConnect 7.3.*x* has been tested only with OpenSearch version 2.6.0.

# Additional Requirements

Gather the following information in advance:

- The server's IP address
- The total amount of RAM installed on the server

# Network Configuration

By default, OpenSearch will need ports 9200 and 9300 open to allow communication from the network. If OpenSearch is to be installed on the same server as the ThreatConnect application, these ports do not need to be open.

# Downloading and Installing OpenSearch 2.6.0

The following instructions are derived from OpenSearch documentation on installing OpenSearch from RPM; however, you may also install OpenSearch from a tarball.

## Installing OpenSearch From RPM

**Important**: The following installation instructions are for RHEL/CentOS Install only.

Run the following commands to download and install OpenSearch:

```
$ curl -L
"https://artifacts.opensearch.org/releases/bundle/opensearch/2.6.0/opensearch-
2.6.0-linux-x64.rpm" -o opensearch-2.6.0-linux-x64.rpm
$ yum install opensearch-2.6.0-linux-x64.rpm
$ systemctl enable opensearch.service
$ chmod 755 /etc/init.d/opensearch
$ systemctl start opensearch.service
```

**Optional**: To use SSL, you must update the default credentials once OpenSearch is configured and running. Follow these steps during the OpenSearch installation to make the minimum changes necessary to use SSL, if desired:

1. Stop OpenSearch. Alternatively, follow Steps 2–4 before starting OpenSearch.
2. Remove all **.pem** files from **/etc/opensearch**.

3. Place your certification authority (CA) certificate, server certificate **.pem** file, and server key **.pem** file in **/etc/opensearch**.

   > **Note**: Record the Distinguished Name (DN) of your server certificate **.pem** file, as you will use it in Step 4.

4. Make the following changes to **opensearch.yml**:
   a. Update all `transport` and `http` `pem` fields accordingly.
   b. Update the `authcz.admin_dn` field by replacing the existing entry with the DN of your server certificate **.pem** file.
   c. Change `allow_unsafe_democertificates` to `false`.
   d. Save and exit.
5. Start OpenSearch.

   > **Note**: At this point, you should still be able to use the default credentials to utilize curl.

6. Generate a new admin hash:
   a. Run the following commands:

   ```
   cd /usr/share/opensearch/plugins/opensearch-security/tools
   ./hash.sh
   ```

   b. When prompted, enter a new password.
   c. Record the provided hash.
7. Update internal users:
   a. Run the following command:

   ```
   cd /usr/share/opensearch/plugins/opensearch-security/securityconfig
   ```

   b. Make the following changes to **internal_users.yml**:
      - Remove all users except the `admin` user.
      - Replace the existing admin hash with the new one generated in Step 6.
   c. Save and exit.
8. Run **securityadmin.sh**:

```
cd /usr/share/opensearch/plugins/opensearch-security/tools
securityadmin.sh -cd ../securityconfig/ -icl -nhnv -cacert
/etc/opensearch/ca.pem -cert /etc/opensearch/server.crt.pem -key
/etc/opensearch/server.key.pem -h <host IP configured in opensearch.yml>
```

You should now be able to use only the new password to utilize curl.

# Verification

Refer to Table 7 to verify all OpenSearch directories.

**Table 7**

| Type | Directory |
|---|---|
| Base Directory | /usr/share/opensearch |
| Logs | /var/log/opensearch |
| Settings | /etc/opensearch/opensearch.yml |

# Configuration

When allocating memory to OpenSearch, the application should receive half of the total server memory. The other half of the server memory should be left intact for Lucene™ to use as storage. Lucene uses the underlying OS to cache data structures in memory, and its performance relies heavily on its ability to interact with the OS. If it does not have available memory to use, the full-text search will suffer performance impacts. The standard is to give half of the available memory to the OpenSearch heap and leave the other half free for Lucene.

In the configuration file **jvm.options** (default location: **/etc/opensearch/jvm.options**), adjust the RAM to be allocated to OpenSearch.

Table 8 provides an example of memory storage allocation. Please use this table as a reference according to the RAM that has been allocated for the OpenSearch installation.

**Note**: OpenSearch settings may be viewed at https://opensearch.org/docs/latest/opensearch/install/important-settings/.

**Important**: If OpenSearch is placed on a server that has more than 32GB of RAM, it is advised that the **jvm.options** settings are not set above 31GB. Java is built to handle about 31.99GB of RAM optimally. Any allocation over 31.99GB will result in performance degradation.

**Table 8**

| **Physical RAM on System (GB)** | 2 | 4 | 8 |
|---|---|---|---|
| **Min Heap Size** | -Xms1g | -Xms2g | -Xms4g |
| **Max Heap Size** | -Xmx1g | -Xmx2g | -Xmx4g |

Open the configuration file (default location: **/etc/opensearch/opensearch.yml**), and make the following changes:

Uncomment and change the cluster name to **opensearch-cluster**:

```
cluster.name: opensearch-cluster
```

Uncomment **network.host**, and add the hostname/IP of the machine, along with the following entries:

```
network.host: <host assigned IP>
http.host: 0.0.0.0
transport.host: <host assigned IP>
```

**Note**: If OpenSearch is to run on the same server as the ThreatConnect application, `http.host` does not need to be specified. The OpenSearch service will bind to the localhost IP and prevent any remote access to the OpenSearch service.

Uncomment **action.destructive_requires_name: true**:

```
action.destructive_requires_name: true
```

Add the following:

```
plugins.index_state_management.enabled: false
```

Add the discovery type associated with your OpenSearch installation. The typical installation will be single server. In this case, add the following line within the discovery options in the **opensearch.yml** configuration file:

```
discovery.type: single-node
```

Add the location for data to be stored to the **opensearch.yml** file, such as in the following example:

```
path.data: /opensearch-data
```

Save and close the file.

# Plugin Installation and Starting OpenSearch

## Plugin Installation

Install the ingest attachment, which allows for indexing of documents:

```
cd /usr/share/opensearch
bin/opensearch-plugin install --batch ingest-attachment
```

## Starting the OpenSearch Service

1. Run the following commands to install and enable firewalld:

   ```
   yum -y install firewalld
   systemctl enable firewalld
   systemctl start firewalld
   firewall-cmd --permanent --zone=public --add-port=9200/tcp
   firewall-cmd --reload
   ```

2. Start OpenSearch:

   ```
   systemctl start opensearch.service
   ```

3. Check the logs to make sure that the server started successfully:

   ```
   tail -f /var/log/opensearch/opensearch-cluster.log
   ```

**Important**: The name of the **.log** file corresponds to the `cluster.name` defined in **opensearch.yml**.

4. By default, OpenSearch runs in secure mode and listens on port 9200 by default. Run the following command to verify that OpenSearch can be accessed:

```
curl -k -u admin:password https://localhost:9200
```

**Important**: Replace `password` with the password created in Step 6 of the "Downloading and Installing OpenSearch 2.6.0" section.

# MySQL Installation and Configuration

A database instance must be operational before installing ThreatConnect. This section outlines how to configure the MySQL database to prepare it for use with ThreatConnect.

## MySQL Download and Installation

To acquire the latest version of MySQL 8.0, download and install a file that has the distribution repository configured:

```
wget http://repo.mysql.com/mysql80-community-release-sles12-6.noarch.rpm
```

Install the distribution repository:

```
rpm -ivh mysql80-community-release-sles12-6.noarch.rpm
```

**Note**: This file and its location may change, based on the application provider. Please consult the **MySQL Yum Repository** download section at https://dev.mysql.com/downloads/repo/yum/ if the link in the first line of the foregoing code does not function.

With the repository now installed and active, use **yum** to install the MySQL server:

```
yum install mysql-community-server -y
```

Configure the **lower-case-table-names** setting. This setting must be configured before first start, as it cannot be configured after the database is initialized.

```
systemctl set-environment MYSQLD_OPTS="--innodb_use_native_aio=0 --lower-case-
table-names=1"
```

Enable on startup and start the MySQL service:

```
systemctl enable mysqld
systemctl start mysqld
```

On the first start, MySQL will go through its first run process. In doing so, it will create a random password for the root account. To obtain this password, retrieve it from the log file with the following command:

```
grep "temporary password" /var/log/mysqld.log
```

The output string will define the root password. Copy this output to a file for future use.

Once the temporary root password is obtained, run the secure installation process to harden MySQL further:

```
mysql_secure_installation
```

The temporary root password will be set as expired and will require an update. The password complexity requires that the new password be at least four characters long, including one numeric character, one lowercase character, one uppercase character, and one special (non-alphanumeric) character.

Once the password has been changed, the secure install process will continue:

- Enter **N** on "Change the password for root?"
- Enter **Y** on "Remove anonymous users?"
- Enter **Y** on "Disallow root login remotely?"
- Enter **Y** on "Remove test database and access to it?"
- Enter **Y** on "Reload privilege tables now?"

The secure installation is now complete.

# MySQL Configuration Options

ThreatConnect requires the following options to be configured for **[mysqld]** in the MySQL configuration file **/etc/my.cnf**:

```
sql_mode=NO_ZERO_IN_DATE,NO_ZERO_DATE,NO_ENGINE_SUBSTITUTION
lower_case_table_names=1
character_set_server=utf8mb4
collation_server=utf8mb4_unicode_ci
group_concat_max_len=1000000
transaction_isolation=READ-COMMITTED
innodb_flush_log_at_trx_commit=2
innodb_table_locks=0
innodb_autoinc_lock_mode=2
eq_range_index_dive_limit=200
innodb_buffer_pool_size=1G
event_scheduler=1
innodb_lock_wait_timeout = 500
max_connections = 300
```

**Important**: Set **innodb_buffer_pool_size** to a value around 75% of the database memory recommended in (e.g., **innodb_buffer_pool_size=1G**).

It is recommended to add these configuration options to the end of the **my.cnf** file for easier identification.

Save the file, and restart the mysql service to commit the changes made in **my.cnf**:

```
# systemctl restart mysqld
```

## Port Configuration

To allow access to the MySQL database from other servers, the firewall must be set up to allow incoming and outgoing connections:

```
firewall-cmd --permanent --zone=public --add-service=mysql
firewall-cmd --reload
```

Alternative methods may be used for the port configuration, but port 3306 needs to be open to allow servers to connect and relay data to the MySQL database.

# Creating the Database and Users

Once the MySQL database service is installed and running, log in to create a database and user for ThreatConnect. From the command line on the MySQL database server, log in as the root user:

```
mysql -u root –p
```

When prompted, enter the root password chosen upon installation. Enter the following command to create a new database:

```
mysql> CREATE DATABASE threatconnect CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```

**Note**: In this example, the database name **threatconnect** was chosen. Any database name may be used, but note that it is referenced later in the installation and configuration process.

Once the database is created, a separate non-root user should be created for security reasons:

```
mysql> CREATE USER 'tcuser'@'%' IDENTIFIED BY 'Password1!';
```

In this example, the username **tcuser** and password **Password1!** were chosen. The **@'%'** addition allows **tcuser** to authenticate from any host on the network. A hostname can be specified here to limit possible login origins (e.g., localhost).

Grant permissions on the newly created user, allowing them to access the tables within this database. Enter the following commands to log into MySQL and add privileges to the new user:

```
mysql> GRANT ALL PRIVILEGES ON threatconnect.* TO 'tcuser'@'%';
mysql> FLUSH PRIVILEGES;
mysql> quit;
```

**Note**: This example assumes that the database was named **threatconnect** and the user **tcuser**. It also assumes that the access for **tcuser** is not limited to a specific host.

**Note**: Any username and password may be used, but note that they will be referenced later in the installation and configuration process.

# Creating ThreatConnect Tables

Before starting this procedure, download the ThreatConnect binary zip file and have it extracted. The **threatconnect/app/scripts/mysql** directory contains a **.sql** script to create the initial tables for ThreatConnect within the database just defined. Browse to the folder containing the **threatconnect-*<version>*.sql** file, run the following command, and enter the root password again when prompted:

```
mysql -u root -p threatconnect-<version>.sql
```

**Important**: This command requires the use of SUPER permissions to run the script. If the MySQL root account is not to be used to run the script, make sure that the user account running the script has the proper permissions. The SUPER permissions are only needed for the non-root user account in MySQL when running the script. The permissions can be removed once the script import has completed.

**Note**: This command specifies **threatconnect** as the name of the database. If a different name for the database was chosen, use that database name instead.

# ThreatConnect Installation and Configuration

## Preparing the ThreatConnect Application Server

### Open File Limits for a Standalone ThreatConnect Application Server

For a server only running ThreatConnect, edit this file location:

```
/etc/security/limits.conf
```

Add these lines at the bottom:

```
threatconnect - nofile 150000
tc-job - nofile 100000
redis - nofile 100000
```

For the file location /etc/sysctl.conf, add the following line at the bottom:

```
fs.file-max = 150000
```

**Important**: Some CentOS/Red Hat versions come with strict permissions for the number of open files allowed per user (https://access.redhat.com/solutions/61334). The steps outlined adjust the number as needed for ThreatConnect to function. The foregoing commands are based on using an account named **threatconnect** to run the ThreatConnect service, **tc-job**, to be used within ThreatConnect for Job, App, and Playbook executions, and an account named **redis** to run the Redis service. The **fs.file-max** parameter is the maximum number of files that can be open in the OS.

## Open File Limits for an All-in-One ThreatConnect Application Server

For a server that is running ThreatConnect, the database, and OpenSearch, edit the following file location:

```
/etc/security/limits.conf
```

Add these lines at the bottom:

```
threatconnect - nofile 150000
tc-job - nofile 100000
redis - nofile 100000
```

For the file location **/etc/sysctl.conf**, add the following line at the bottom:

```
fs.file-max = 150000
```

**Important**: Some CentOS/Red Hat versions come with strict permissions for the number of open files allowed per user (https://access.redhat.com/solutions/61334). The steps outlined adjust the number as needed for ThreatConnect to function. The foregoing commands are based on using an account named **threatconnect** to run the ThreatConnect service, **tc-job**, to be used within ThreatConnect for Job, App, and Playbook executions, and an account named **redis** to run the Redis service. The **fs.file-max** parameter is the maximum number of files that can be open in the OS. The **fs.file-max** parameter is larger than the standalone configuration to accommodate for the open files needed for OpenSearch and MySQL to function.

## JDK

The system needs access to one of the JDKs as outlined in the "Software" section. Also, the **JAVA_HOME** environment variable needs to be appropriately configured to point to that directory.

Users must execute the following commands to verify that Java is accessible:

```
which java
```

## User Account Creation

It is suggested that the ThreatConnect service be run by a service account. The steps in this section will assist in the creation of the **threatconnect** user account, which is used by default in the ThreatConnect software. The user account that will run the ThreatConnect service can be adjusted to run as another account if needed.

To create the **threatconnect** account, run the following command:

```
adduser threatconnect
```

To capitalize on the Java changes, update the user account's **.bashrc**. First, switch to the new user account:

```
su - threatconnect
```

Access the user's **.bashrc**:

```
vi ~/.bashrc
```

Add this line to the file if using Oracle Java JDK:

```
export JAVA_HOME=/usr/java/latest
```

Add this line to the file if using OpenJDK:

```
export JAVA_HOME=/usr/lib/jvm/jre-11-openjdk-11.0.6.10-1.el7_7.x86_64
```

**Important**: The foregoing location will vary depending on OS version and version of OpenJDK Java 11 downloaded to the system.

Restart the **.bashrc** with the new changes:

```
source ~/.bashrc
```

Verify that the changes are available:

```
echo $JAVA_HOME
```

Ensure that the changes match what was added. If they do, log out of the **threatconnect** user and continue.

# Port Forwarding

For security reasons, the ThreatConnect application runs as an unprivileged user, which does not require root or Administrator permissions. Many operating systems prevent unprivileged applications from binding on commonly used ports, such as ports 80, 443, and 25, which are widely used by web applications such as ThreatConnect.

As a result, it is essential to configure redirection so that incoming web and email traffic on ports 80, 443, and 25 are forwarded to ports used by ThreatConnect (8080, 8443, and 2500, respectively). It is also essential to open the port used in the **appMessageBrokerHost** system setting (e.g., 62000) to allow real-time web socket communication with Playbooks.

Add the following rules to the firewallD:

```
systemctl enable firewalld
systemctl start firewalld
firewall-cmd --permanent --zone=public --add-service=smtp
firewall-cmd --permanent --zone=public --add-service=http
firewall-cmd --permanent --zone=public --add-service=https
firewall-cmd --permanent --zone=public --add-forward-
port=port=25:proto=tcp:toport=2500
firewall-cmd --permanent --zone=public --add-forward-
port=port=80:proto=tcp:toport=8080
firewall-cmd --permanent --zone=public --add-forward-
port=port=443:proto=tcp:toport=8443
firewall-cmd --permanent --direct --add-rule ipv4 nat OUTPUT 0 -p tcp -o lo --dport
443 -j REDIRECT --to-ports 8443
firewall-cmd --permanent --zone=public --add-port=62000/tcp
firewall-cmd --reload
```

Or add the following rules to the iptables:

**Important**: These rules will not persist after an OS reboot. It is suggested that **firewall-cmd** be used for persistent rules.

```
iptables -t nat -A PREROUTING -p tcp -m tcp --dport 443 -j REDIRECT --to-ports 8443
iptables -t nat -A PREROUTING -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 8080
iptables -t nat -A PREROUTING -p tcp -m tcp --dport 25 -j REDIRECT --to-ports 2500
iptables -t nat -A OUTPUT -p tcp -o lo --dport 443 -j REDIRECT --to-ports 8443
```

# Redis Server Installation and Setup (for RHEL 7)

## Installing the Redis Server

**Important**: Redis, like other databases used by ThreatConnect software (e.g., MySQL and OpenSearch), is not supported by ThreatConnect. Therefore, its support is technically independent of the company's product.

**Important**: Redis is installed on the application server.

**Note**: Redis 6.2.6 packages may be downloaded from your desired repository.

It is required to have developer packages installed to compile Redis from source:

```
yum install –y bison byacc cscope ctags cvs diffstat doxygen flex gcc gcc-c++ gcc-gfortran gettext git indent intltool libtool patch patchutils rcs redhat-rpm-config rpm-build tcl
```

Download the source code:

```
curl --output redis-6.2.6.tar.gz http://download.redis.io/releases/redis-6.2.6.tar.gz
```

Decompress the downloaded archive:

```
tar -xvzf redis-6.2.6.tar.gz
```

Navigate to the directory where the decompressed files reside:

```
cd redis-6.2.6
```

Following are the compiled dependencies needed for Redis:

```
cd deps/
make hiredis lua jemalloc linenoise
```

Begin the compile process to ensure there are no errors:

```
cd ..
make && make test
```

Compile all modules needed:

```
make install
```

Issue the following command as a privileged user, and answer the questions to complete the install:

```
cd utils/
./install_server.sh
```

Alternatively, the install can be started via **redis-server** passing in the **redis.conf** file located in **/<*staging-location*>/redis-6.2.6/redis.conf**.

## Setting up the Redis Server

This script will help set up a running Redis server:

```
utils/install_server.sh
```

To operate Redis, configuration settings will need to be specified as required by the script. The default configuration for each line is as follows:

- Redis port for the instance: **6379**
- Redis config file name: **/etc/redis/6379.conf**
- Redis log file name: **/var/log/redis_6379.log**
- Data directory for the instance: **/var/lib/redis/6379**
- Redis executable path: **/usr/local/bin/redis-server**

The selected config is as follows:

```
Port: 6379
Config file: /etc/redis/6379.conf
Log file: /var/log/redis_6379.log
Data dir: /var/lib/redis/6379
Executable: /usr/local/bin/redis-server
Cli Executable: /usr/local/bin/redis-cli
```

When prompted with the question **Is this okay?**, press **Enter** to continue, or press **Ctrl-C** to abort.

The following output from the script will be shown only if the installation is successful:

```
Copied /tmp/6379.conf => /etc/init.d/redis_6379
Installing service...
Successfully added to chkconfig!
Successfully added to runlevels 345!
Installation successful!
```

Once the configuration is completed, edit the configuration file:

```
vi /etc/redis/6379.conf
```

Add these lines at the end of the file:

```
maxmemory 6gb
maxmemory-policy allkeys-lru
maxmemory-samples 5
```

Save and close the file. To better troubleshoot and log any issues, run the Redis service as another user instead of **root**. To do so, create a new user account:

```
adduser redis
```

Edit the Redis service to run as the new Redis user:

```
vi /etc/init.d/redis_6379
```

Add this line after **REDISPORT**:

```
USER=redis
```

Comment out this line:

```
$EXEC $CONF
```

Create a new line after the line that was just commented, and add the following:

```
su - $USER -c "$EXEC $CONF"
```

Save and close the file.

Change ownership of the following file and directories to the Redis user:

```
chown -R redis:redis /var/lib/redis/
chown redis:redis /var/log/redis_6379.log
chown -R redis:redis /etc/redis/
sysctl -w net.core.somaxconn=512
echo 'vm.overcommit_memory = 1' >> /etc/sysctl.conf
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

Restart the Redis service to apply the new changes:

```
/etc/init.d/redis_6379 restart
```

# Redis Server Installation and Setup (for RHEL 8)

## Installing the Redis Server

**Important**: Redis, like other databases used by ThreatConnect software (e.g., MySQL and OpenSearch), is not supported by ThreatConnect. Therefore, its support is technically independent of the company's product.

**Important**: Redis is installed on the application server.

**Note**: Redis 6.2.6 packages may be downloaded from your desired repository.

Install the following Redis dependencies:

```
yum install bison byacc cscope ctags diffstat flex gcc gcc-c++ gcc-gfortran gettext
git intltool libtool patch patchutils rpm-build tcl systemd-devel –y
```

Download the source code:

```
cd /opt
curl --output redis-6.2.6.tar.gz http://download.redis.io/releases/redis-
6.2.6.tar.gz
```

Decompress the archive that was downloaded:

```
tar -xvzf redis-6.2.6.tar.gz
```

Navigate to the directory where the decompressed files reside:

```
cd redis-6.2.6
```

The following compiled dependencies are needed for Redis:

```
cd deps/
make hiredis lua jemalloc linenoise
```

Begin the compile process to ensure that there are no errors:

```
cd ..
make USE_SYSTEMD=yes && make test
```

Compile all modules needed:

```
make install
```

## Setting up the Redis Server

Modify the **redis.conf** file:

```
vi /opt/redis-6.2.6/redis.conf
```

Add the following properties to the **redis.conf** file:

```
bind 0.0.0.0
protected-mode no
supervised systemd
maxmemory 6gb
maxmemory-policy allkeys-lru
maxmemory-samples 5
```

Copy the Redis service to the system:

```
cp utils/systemd-redis_server.service /etc/systemd/system/redis.service
```

Modify the **redis.service** file:

```
vi /etc/systemd/system/redis.service
```

Add the following properties to the **[Service]** section of the **redis.service** file:

```
ExecStart=/usr/local/bin/redis-server /opt/redis-6.2.6/redis.conf
Type=simple
Start the Redis Server (as root)
service redis start
```

Check the Redis version:

```
/usr/local/bin/redis-server --version
Redis server v=6.2.6 sha=00000000:0 malloc=jemalloc-5.1.0 bits=64
build=aa75cf28d09caef0
```

Confirm that Redis is running:

```
/usr/local/bin/redis-cli ping
PONG
```

## Starting the Redis Server as a Non-"root" User

Create the **redis** user:

```
adduser redis
```

Modify the **redis.service** file:

```
vi /etc/systemd/system/redis.service
```

Add the following properties to the **[Service]** section of the **redis.service** file:

```
User=redis
Group=redis
WorkingDirectory=/var/lib/redis
```

Reload **systemd**:

```
systemctl daemon-reload
```

Change the ownership of the following directories and executables to the **redis** user:

```
chown -R redis:redis /var/lib/redis/ && chown redis:redis /var/log/redis_6379.log
&& chown -R redis:redis /etc/redis/
chown redis:redis /usr/local/bin/redis-*
```

Start the Redis service:

```
service redis start
```

Confirm that the Redis service is running:

```
service redis status
```

Confirm that the Redis service was started by the **redis** user:

```
ps -ef | grep redis
```

# Python 3.6 Installation

## Installing Python 3.6

**Note**: The instructions in this section compile and install Python from source code, which is one of the multiple ways to install Python on the application server. If the operating system to be used already has Python 3.6 installed, skip the compile steps of the Python installation.

It is required to have developer packages installed to compile Python from source:

```
yum install zlib-devel bzip2-devel openssl-devel ncurses-devel \
sqlite-devel readline-devel tk-devel gdbm-devel xz-devel expat-devel \
python3-setuptools epel-release openssl11 openssl11-libs openssl11-devel -y
```

Download the source code to **/tmp**:

```
cd /tmp
wget https://www.python.org/ftp/python/3.6.15/Python-3.6.15.tar.xz
```

Decompress the downloaded archive:

```
tar -xf Python-3.6.15.tar.xz
```

Navigate to the directory where the decompressed files reside:

```
cd Python-3.6.15
```

Run the following commands to configure Python:

```
mkdir -p /opt/python3.6.15/lib &&\
./configure --prefix=/opt/python3.6.15 \
--with-ensurepip=install --enable-optimizations \
--enable-shared LDFLAGS="$LDFLAGS -Wl,-rpath /opt/python3.6.15/lib"
```

Begin the compile process to ensure there are no errors:

```
make && make altinstall
```

Set up a symbolic link:

```
ln -s /opt/python3.6.15/bin/python3.6 /opt/python3.6.15/bin/python
```

## Python 3.6 SDK: TcEx Installation

**Note**: The instructions in this section need to be run after Python has been installed on the application server.

**Note**: The instructions in this section are based on the Python installation method discussed in this guide. Adjust directories as needed if Python is installed in another location.

To ensure that no permissions issues arise from the use of Python packages, use the following commands to update permissions:

```
chmod -R 755 /opt/python3.6.15/lib/python3.6/site-packages
chmod -R 755 /opt/python3.6.15/lib/python3.6/lib2to3
```

To install TcEx using pip, execute the following command:

```
/opt/python3.6.15/bin/pip3.6 install tcex-cli
```

# Python 3.11 Installation

## Installing Python 3.11

**Note**: The instructions in this section compile and install Python from source code, which is one of the multiple ways to install Python on the application server. If the operating system to be used already has Python 3.11 installed, skip the compile steps of the Python installation.

### CentOS 7 and RHEL 7

Run one of the following commands to install additional software collections (SCLs):

- **CentOS 7**

```
yum install centos-release-scl -y
```

- **RHEL 7**

```
subscription-manager repos --enable rhel-server-devtools-7-rpms
subscription-manager repos --enable rhel-server-rhscl-7-rpms
```

Install the GNU Compiler Collection (GCC) and set it as the default:

```
yum install devtoolset-7
echo "source scl_source enable devtoolset-7" >> ~/.bash_profile
. ~/.bash_profile
```

Install dependencies specific to Python 3.11:

```
yum install -y epel-release --enablerepo=extras
yum install -y openssl11 openssl11-libs openssl11-devel lcms2-devel
```

It is required to have developer packages installed to compile Python from source:

```
yum install -y yum-utils \
    make gcc \
    openssl openssl-devel \
    postgresql-devel \
    libtiff-devel libjpeg-devel libzip-devel freetype-devel \
    libwebp-devel tcl-devel libxslt-devel libxml2-devel \
    bzip2-devel \
    gdbm-devel \
    libffi-devel \
```

```
    sqlite-devel \
    ncurses-devel \
    readline-devel \
    tk-devel \
    xz-devel \
    zlib-devel \
    wget ;\
    yum clean all
```

Download, build, and install Python 3.11:

```
mkdir /tmp/python3.11.1-build && \
    cd /tmp/python3.11.1-build && \
    curl https://www.python.org/ftp/python/3.11.1/Python-3.11.1.tgz > python-
3.11.1.tgz && \
    tar xzf python-3.11.1.tgz && \
    cd Python-3.11.1 && \
    mkdir -p /opt/python3.11.1/lib && \
    export CFLAGS="$CFLAGS $(pkg-config --cflags openssl11)" && \
    export LDFLAGS="$LDFLAGS $(pkg-config --libs openssl11)" && \
    ./configure --prefix=/opt/python3.11.1 \
        --with-ensurepip=install \
        --enable-optimizations \
        --enable-shared LDFLAGS="$LDFLAGS -Wl,-rpath /opt/python3.11.1/lib" && \
    make -j$(nproc)
```

Begin the compile process to ensure there are no errors:

```
cd /opt/python3.11.1/lib/python3.11
make install
```

Set up a symbolic link:

```
ln -s /opt/python3.11.1/bin/python3.11 /opt/python3.11.1/bin/python
```

## RHEL 8

It is required to have developer packages installed to compile Python from source:

```
yum install -y yum-utils \
    make gcc \
    openssl openssl-devel \
    postgresql-devel \
    libtiff-devel libjpeg-devel libzip-devel freetype-devel \
    libwebp-devel tcl-devel libxslt-devel libxml2-devel \
    bzip2-devel \
    gdbm-devel \
    libffi-devel \
    sqlite-devel \
    ncurses-devel \
    readline-devel \
    tk-devel \
    xz-devel \
    zlib-devel \
    wget ;\
    yum clean all
```

Download, build, and install Python 3.11:

```
mkdir /tmp/python3.11.1-build && \
    cd /tmp/python3.11.1-build && \
    curl https://www.python.org/ftp/python/3.11.1/Python-3.11.1.tgz > python-
3.11.1.tgz && \
    tar xzf python-3.11.1.tgz && \
    cd Python-3.11.1 && \
    mkdir -p /opt/python3.11.1/lib && \
    export CFLAGS="$CFLAGS $(pkg-config --cflags openssl11)" && \
    export LDFLAGS="$LDFLAGS $(pkg-config --libs openssl11)" && \
    ./configure --prefix=/opt/python3.11.1 \
        --with-ensurepip=install \
        --enable-optimizations \
        --enable-shared LDFLAGS="$LDFLAGS -Wl,-rpath /opt/python3.11.1/lib" && \
    make -j$(nproc)
```

Begin the compile process to ensure there are no errors:

```
cd /opt/python3.11.1/lib/python3.11
make install
```

Set up a symbolic link:

```
ln -s /opt/python3.11.1/bin/python3.11 /opt/python3.11.1/bin/python
```

### Python 3.11 SDK: TcEx Installation

**Note**: The instructions in this section need to be run after Python has been installed on the application server.

**Note**: The instructions in this section are based on the Python installation method discussed in this guide. Adjust directories as needed if Python is installed in another location.

To ensure that no permissions issues arise from the use of Python packages, use the following commands to update permissions:

```
chmod -R 755 /opt/python3.11.1/lib/python3.11/site-packages
chmod -R 755 /opt/python3.11.1/lib/python3.11/lib2to3
```

To install TcEx using pip, execute the following command:

```
/opt/python3.11.1/bin/pip3 install --upgrade pip
/opt/python3.11.1/bin/pip3 install tcex-cli
```

## Installing Reporting Dependencies

Use the following commands to install dependencies for the ThreatConnect reporting feature:

```
yum install fontconfig libXrender libXext -y
curl -L https://dl.google.com/linux/direct/google-chrome-stable_current_x86_64.rpm
-o google-chrome.rpm && \
yum install ./google-chrome.rpm -y && \
rm -f google-chrome.rpm
echo "export CHROME_BIN=/usr/bin/google-chrome" >> /home/threatconnect/.bashrc
```

# Getting Started

ThreatConnect clients can use this guide to configure and install their instance of ThreatConnect. This guide assumes a moderate level of systems-administration expertise and an operating environment that satisfies the requirements detailed earlier in this document.

**Important**: All references to **threatconnect-‹ *version*›** should not be taken literally; instead, replace the **‹ *version*›** string with the most recent version of the software (e.g., **threatconnect-7.3**).

# Downloading the Installer

Follow these steps to download the files to install ThreatConnect:

**Important**: To download the files, users will need to obtain Onehub® credentials from their ThreatConnect Customer Success team. If users do not have a Onehub account, an alternative repo location containing the files can be provided.

1. Navigate to https://ws.onehub.com/workspaces/571735/signin to access the ThreatConnect Workspace **Sign In** screen (Figure 1).


**Figure 1**

2. Enter the sign-in credentials, and then click the **SIGN IN** button. The **ThreatConnect Onehub Workspace** screen will be displayed (Figure 2).

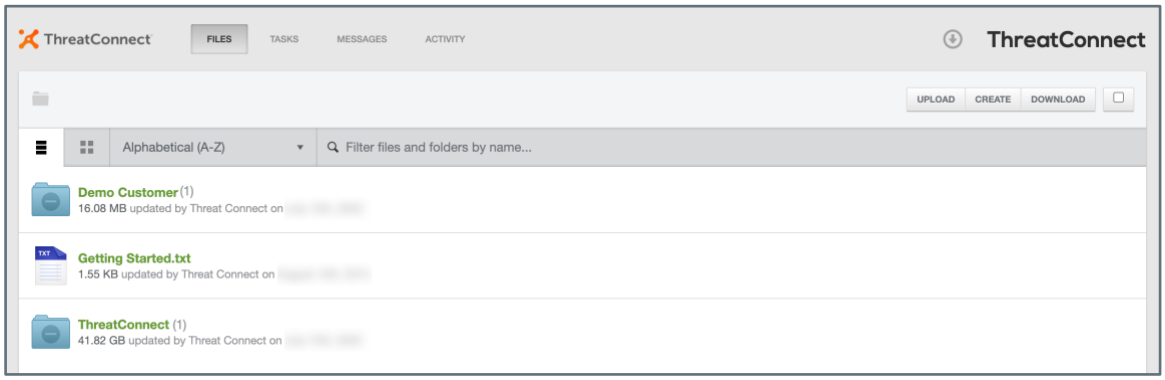


**Figure 2**

The screen displays two directories: a **ThreatConnect** folder and a private **Customer** folder that is specific to each client's account.

## The ThreatConnect Folder

The **ThreatConnect** folder contains all the files required to install each version of ThreatConnect. The directory structure for this folder is as follows:

```
/ThreatConnect/
    Release/
        <version> - (Date)/
            Documents/
            threatconnect-<version>.zip
```

To download a file, click its corresponding checkbox (on the right side of the screen), and then click the **DOWNLOAD** button at the top right (Figure 3).

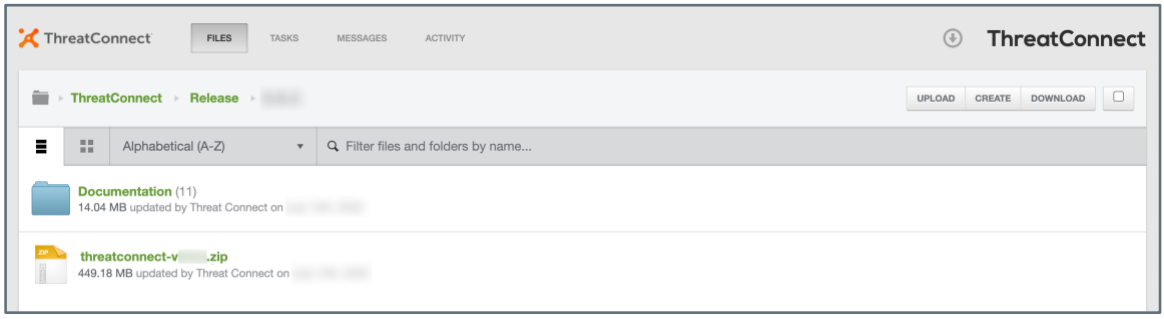


**Figure 3**

## The Customer Folder

The **Customer** folder (in this example, **Demo Customer**; see Figure 4) is private between clients and their ThreatConnect Customer Success team; other clients cannot access this folder or even know of its existence. From this folder, clients can download their license key. In addition, the folder contains a sub-folder named **Upload**, which clients can use to safely and securely share files, such as screen captures or log files, which facilitates remote assistance from ThreatConnect's Customer Success team.


**Figure 4**

# Opening the Installer

## Unzipping the File

The ThreatConnect **.zip** file serves as an archive of all the necessary files and folders needed to install the platform.

1. Copy the **.zip** file to the desired directory on the ThreatConnect Application server. By default, this directory is **/opt**, which will result in an installation directory of **/opt/threatconnect**.
2. Unzip the file from the command-line interface with the following command:

```
unzip threatconnect-<version>.zip
```

## Installation Directory Structure

After extracting the archive, the new folder will contain the following directory structure:

```
threatconnect/
      app/
      config/
      exchange/
```

The **app** directory contains all of the binaries and supporting libraries needed to run ThreatConnect, and it also contains the scripts required to configure and administer a user's ThreatConnect instance.

The **config** directory contains all of the customized configuration information for a user's ThreatConnect instance. It is populated with keystore, XML, and properties files after the installer is run. The **app** and **config** directories are referenced throughout the rest of this document.

**Important**: Users patching their ThreatConnect instance to a newer version will need to replace the **app** directory with one from the latest version, but will not need to update their **config** directory after it is established from the initial installation.

## Folder Permissions Adjustment

Once the ThreatConnect binary has been unzipped and put in the location from where it will run, the permissions will need to be adjusted to allow the **threatconnect** user ownership of the files:

```
chown -R threatconnect:threatconnect /opt/threatconnect
```

**Note**: The installation commands in this guide use the default installation path of **/opt/threatconnect** and the default user account of **threatconnect**. Adjust the commands as needed.

# TC Exchange Setup

## Creating “tc-job” User

To provide additional security, it is recommended that a separate user be created to run the TC Exchange™ jobs. It is also recommended that read and write groups be created to control the permission to these files:

```
useradd tc-job
echo "tc-job-pass123" | passwd tc-job --stdin
```

Update permissions as follows:

```
chgrp -R threatconnect /opt/threatconnect/exchange/

# correct octal permissions
find /opt/threatconnect/exchange/ -type f -exec chmod 644 -- {} +
find /opt/threatconnect/exchange/ -type d -exec chmod 755 -- {} +

# set new default ACLs
setfacl -Rdm u:tc-job:rx /opt/threatconnect/exchange/programs/
setfacl -Rdm u:tc-job:rwx /opt/threatconnect/exchange/jobs/
setfacl -Rdm u:threatconnect:rwx /opt/threatconnect/exchange/jobs/
```

## User Privilege Configuration

Add the following lines to the **pam** configuration above the top **auth** command:

```
/etc/pam.d/su
auth    sufficient pam_rootok.so
auth    [success=ignore default=1] pam_succeed_if.so user = tc-job
auth    sufficient pam_succeed_if.so use_uid user = threatconnect
```

## Configuring Sudoers

Add the following to the sudoers configuration to allow the **threatconnect** user to run the jobs as the **tc-jobs** user:

```
/etc/sudoers.d/threatconnect
Defaults:threatconnect !requiretty
threatconnect ALL=(tc-job) NOPASSWD: ALL
```

# Starting the Installer

Open a terminal window and browse to the **app** directory within the **threatconnect** folder. Run the **setup.sh** file as the ThreatConnect user defined for the application.

```
./setup.sh
```

**Warning**: The **threatconnect.xml** file can be corrupted if run as any other user.

**Important**: Attempting to run the **start.sh** script before the respective **setup.sh** script will result in an error.

# Installation Basics

The installer prompt allows for configuration of the database and SMTP servers used by ThreatConnect. Clients will be prompted to conduct an initial setup, and subsequent runs of the installer will present additional configuration options, as detailed in the following sections.

## Time Zone

**Important**: Users MUST install the ThreatConnect instance in the UTC (Coordinated Universal Time) time zone for all servers, as the timestamps will be sent in UTC.

Use the following commands to change the time zone for CentOS 6 or RHEL 6:

```
mv /etc/localtime /etc/localtime_backup ln -s /usr/share/zoneinfo/UTC
/etc/localtime
```

For CentOS 7, RHEL 7, or RHEL 8, use the following command instead:

```
timedatectl set-timezone UTC
```

Once the change has been implemented, it can be validated via the following command:

```
ls -l /etc/localtime
```

The expected output will look like the following:

```
/etc/localtime -> /usr/share/zoneinfo/UTC
```

## Initial Setup

The initial setup runs the database, SMTP, and memory-configuration options simultaneously. It will reflect a user's input in the appropriate files within the **config** directory. After running the initial setup, additional advanced options will appear when running the setup script.

## Selecting Appropriate Server Type

Select the appropriate server type to configure, which will construct the correct corresponding **threatconnect.xml** file to use on that server.

There are four options from which to choose:

- Messaging server
- Playbooks/Job server
- Web/API server
- Full server

**Note**: The selection will always be the **Full server** option unless specified by the Deployment Engineer.

## Entering a Server Name

Enter a server name that will be applied to the **tc.serverName** system property in the **threatconnect.xml** file.

> **Note**: To use the default name that is displayed in the prompt, do not enter any value. Otherwise, type a sensible, short name without spaces that describes your ThreatConnect instance properly.

## Configuring Database Properties

This option allows for the configuration of the database properties for ThreatConnect. To fulfill its role as an intelligence platform, ThreatConnect requires access to a database instance for storing Indicators, user data, and more. Clients will be prompted to enter the information in Table 9.

**Table 9**

| Property | Description | Example Value |
|---|---|---|
| Database Name | The name of the database instance | threatconnect |
| Database Port | The port used by the database server | 3306 (MySQL)<br>5432 (PostgreSQL) |
| Database Host | The hostname of the server hosting the database | localhost |
| Username | The username used for authentication with the database | tcuser |
| Password | The password used for authentication with the database | Password1! |

The installer will offer the user the option of testing the database connection with the values as configured using Java Database Connectivity (JDBC). When prompted, save the database configuration changes if satisfied with the configuration results.

> **Important**: If a valid database connection cannot be reached, then the installer will not proceed with the installation and configuration.

## Configuring SSL Properties

This option allows the user to configure the following SSL settings for ThreatConnect:

- Enable the Server Name Indication (SNI) extension by setting its value to `true`. See the "Enabling SNI" section for more information.
- ThreatConnect is pre-configured with a keystore containing a self-signed SSL certificate, and the password for this keystore is `password`. It is recommended to use your own keystore and certificate. See "Appendix B: Re-Authenticating the Keystore and SSL Certificate Configuration" for more information.

## Configuring SMTP Properties

This option allows the user to set the SMTP server settings for ThreatConnect. The platform will use this server to send email alerts, communicate with users, and more. When setting the SMTP properties, users will be prompted for the information in Table 10.

**Table 10**

| Property | Description | Example Value |
|---|---|---|
| Mail SMTP Host | The hostname of the SMTP server used by ThreatConnect | smtp.acme.net |
| Mail SMTP Port | The port used by the SMTP server | 25 |
| SSL Required | Indicates whether the SMTP server requires encrypted SSL | true |
| SMTP User | The SMTP username, if authentication is required | tcsender |
| TLS Required | Indicates whether the SMTP server requires TLS | true |
| SMTP Password | The password used to authenticate the username | smtppassword |

**Note**: If user authentication is not needed for the SMTP settings, select **false** when prompted to select whether user authentication is required.

## Configuring Memory Properties

This option allows the user to set the memory settings for the server. The ThreatConnect application will be generated using the parameters specified in Table 11. Once entered, the installer will confirm the settings.

**Table 11**

| Property | Description | Example Value (GB) |
|---|---|---|
| Heap Size Limit | The memory limit for the heap | 8 |
| Initial Heap Size | The initial heap size | 8 |

**Important**: Setting the heap size limit to less than 6 gigabytes (represented as 6G or 6000M) or higher than 31 gigabytes (represented as 31G or 31000M) may result in adverse performance. Use the same value for the initial heap size.

## Configuring a Full Server

When configuring a **Full server**, the user will be prompted for the **Playbooks** server Worker size (e.g., 4). This value can be changed later in the UI if additional workers are allocated, per the ThreatConnect license agreement.

## Configuring a Playbooks/Job Server

When configuring a **Playbooks/Job server**, *only* the ThreatConnect server has to be configured, according to the steps outlined in the "ThreatConnect Installation and Configuration" section.

The user will be prompted for the **Playbooks** server Worker size (e.g., 4). This value can be changed later in the UI if additional workers are allocated, per the ThreatConnect license agreement.

The user will also be prompted for the location for the **keystore.jks** file. The default location is **/opt/threatconnect/config/keystore.jks**.

## Configuring the ThreatConnect License

This option allows the user to import a valid ThreatConnect license straight from the installer. Users will be asked if they wish to apply a license at this point. If so, the absolute path of the ThreatConnect license file must be provided (e.g., **/opt/threatconnect/license.xml**). The license may also be applied once ThreatConnect is installed and running.

If you are using Java version 11.0.11 or above, you will need to modify the threatconnect.xml file afterwards and make the following changes to each **"jdbc"** database connection entry:

1. Search for "**jdbc**" in the **threatconnect.xml** file within your ThreatConnect installation folder/configuration.
2. In the two **jdbc** entries, add the following string to each jdbc property, at the end of each string:

```
&amp;useSSL=false
```

3. Save the changes, and quit your file editor. Then proceed with the steps detailed in the "Starting ThreatConnect" section.

## Enabling SNI

Enable SNI in the **threatconnect.xml** file to ensure that Shodan® API keys of all access levels can be used to configure the Shodan built-in enrichment functionality:

1. Place the following property inside the **<system-properties/>** tag of the **threatconnect.xml** file:

```
<property name="jsse.enableSNIExtension" value="true"/>
```

2. Restart the ThreatConnect application server.

# Starting ThreatConnect

After completing the installation and configuration procedures, it is time to start ThreatConnect. The options in the previous sections allow a user to run ThreatConnect in a single session. This capability presents some limitations: The platform will need to be started manually after each reboot, or a terminal window or Secure Shell (SSH) session may have to be left open. This section details the file configuration to run ThreatConnect as a service in Linux.

**Important**: If FIPS is enabled for RHEL 8, add `-Dcom.redhat.fips=false` to the `JAVA_OPTS` variable in the **setup.sh** and **start.sh** scripts.

Open a terminal window, and browse to the **app** directory within the ThreatConnect directory. Run the **start.sh** file as the ThreatConnect user defined for the application:

```
# ./start.sh
```

**Note**: It is suggested that the application be set up to run as a service. This setup limits any issues in configuration, as well as allows the application to run in the background.

## The Service Script

Within the ThreatConnect installation, there is a script used for running ThreatConnect as an initialized service:

```
/opt/threatconnect/app/service/threatconnect
```

This script must be copied into the **/etc/init.d** directory for it to be recognized as a system service. Note that users may need privileges to copy to this directory:

```
cp /opt/threatconnect/app/service/threatconnect /etc/init.d
chmod 755 /etc/init.d/threatconnect
```

# Configuring Services

The service script requires proper permissions and paths to be set.

1. Specify the **BASEDIR** variable within the script to point to the path where the ThreatConnect installation exists. By default, this path is **/opt/threatconnect**.
2. Specify the **USER** variable within the script to identify the user that owns the files for the ThreatConnect application. It is advisable, for security reasons, that the root user not be employed. By default, the username is assumed to be **threatconnect**.
3. Optionally, un-comment the **iptables** lines below within the script file. Choosing to do so will redirect ports 443 and 80 to run on the specified ports. By default, they redirect to 8443 and 8080, although these may be modified, as needed, depending on how the ports were configured according to the "Port Forwarding" section. The lines in the script are initially as follows:

```
iptables -t nat -A PREROUTING -p tcp -m tcp --dport 443 -j REDIRECT --to-ports
8443
iptables -t nat -A PREROUTING -p tcp -m tcp --dport 80 -j REDIRECT --to-ports
8080
iptables –t nat –A OUTPUT –p tcp –o lo –dport 443 –j REDIRECT –to-ports 8443
iptables -t nat -A PREROUTING -p tcp -m tcp --dport 25 -j REDIRECT --to-ports
2500
```

**Note**: If the **firewall-cmd** service is not allowed on the application server, it is suggested that the above rules be uncommented, so that network traffic is allowed through anytime the ThreatConnect service is online.

# Starting ThreatConnect as a Service

Once the services have been configured, ThreatConnect can be started as a service. To do so, enter one of the following commands as the root user:

```
service threatconnect start
/etc/init.d/threatconnect start
systemctl start threatconnect
```

To stop the service, enter any of the following commands:

```
service threatconnect stop
/etc/init.d/threatconnect stop
systemctl stop threatconnect
```

To have ThreatConnect start on system startup, issue the following commands after the script is configured in the **/etc/init.d** directory:

```
chkconfig --add threatconnect
chkconfig threatconnect on
```

For CentOS 7, RHEL 7, or RHEL 8, execute the following command instead:

```
systemctl enable threatconnect
```

# The First Login

Once the ThreatConnect instance is installed, configured, and running, log in to begin customization. Most of the administrative functions within ThreatConnect are outside the scope of this document and can be found in the ThreatConnect administration guides. Before initiating any customizations, complete the tasks in the following sections.

## Entering Initial Login Credentials

To conduct the initial login, and to access a system account to begin customization, go to the ThreatConnect instance in a Web browser and log in with the username **admin** and the password **password1**. Change these credentials after the initial installation to enhance security.

## Setting Master Key for Keychain

If the **keychainEnabled** property is set to **true**, a System Administrator will need to log out and log back into ThreatConnect to set the master key. When prompted, enter a Master Secret Key in the fields provided (Figure 5).

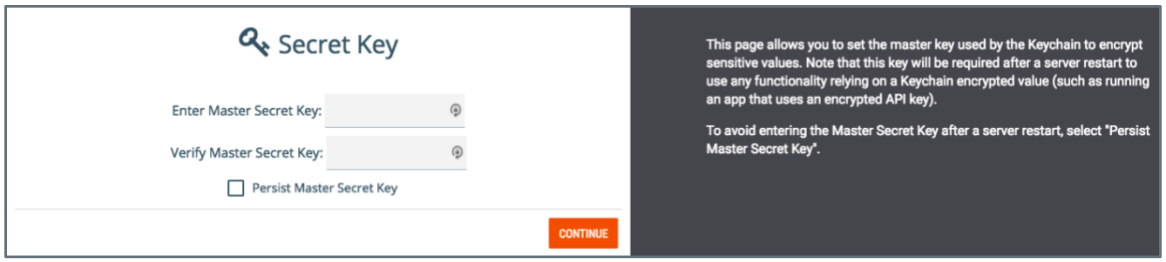


**Figure 5**

The Master Secret Key is used to encrypt sensitive values and is required on every server restart unless the **Persist Master Secret Key** box is checked. Selecting the checkbox is recommended to avoid entering the Master Secret Key on each ThreatConnect service restart.

# Installing the License

**Important**: This section is applicable only if a license was not correctly inserted into ThreatConnect during the installer's configuration.

1. Log into ThreatConnect with a System Administrator account.

2. On the top navigation bar, hover the cursor over **Settings** and select **System Settings**. The **System Settings** screen will be displayed (Figure 6).

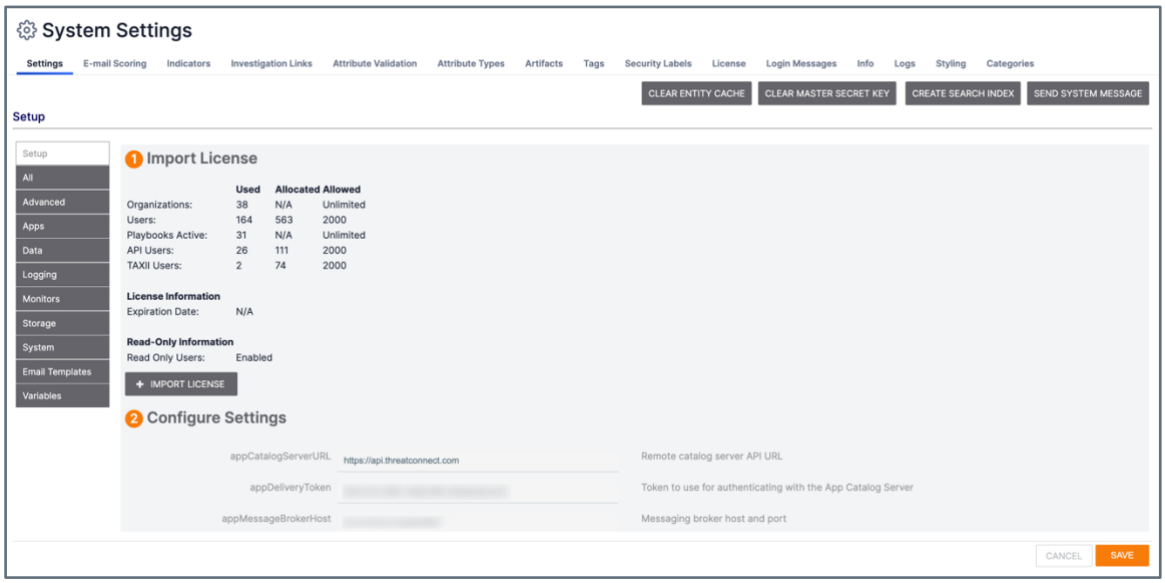


**Figure 6**

3. Click the **License** tab. The **License** screen will be displayed with the **License Config** subtab highlighted (Figure 7), showing the current allocations of Organizations and users. From this screen, the user can also import a license.

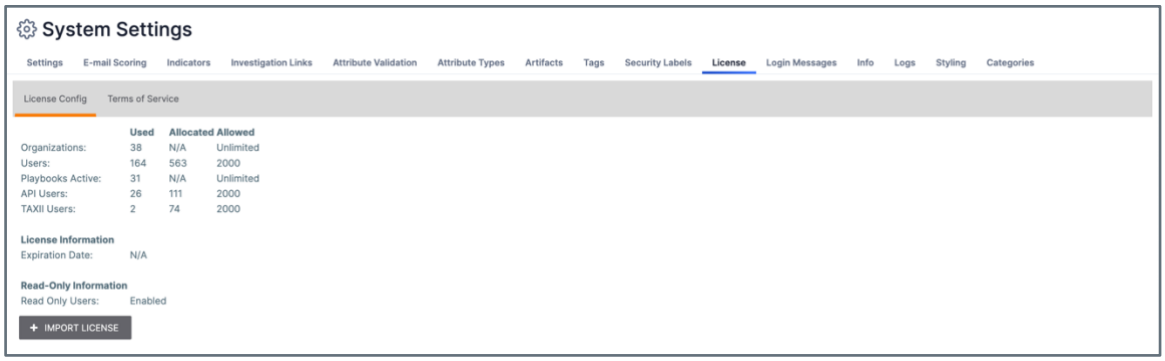


**Figure 7**

> **Note**: Certain users may be required to re-accept the terms of the License Agreement upon next login.

4. Click the **+ IMPORT LICENSE** button, and choose the License file from the directory.
5. Specific users may then be prompted to accept the license terms.
6. Restart ThreatConnect.

> **Warning**: Without a valid license, ThreatConnect will not function properly.

# System Settings Checklist

Users should review the settings of their instance to ensure that it is configured according to their needs. Table 12 is a checklist showing the core properties and their values, assuming a standard ThreatConnect setup.

**Table 12**

| Property | Common Value | Required For |
|---|---|---|
| appsApiUrl | https://threatconnect.customer.com/api | API URL for App Jobs |
| appsJavaHome | /usr/java/latest | TC Exchange Apps |
| appsPythonHome | /opt/python3.6.15/bin | TC Exchange Apps |
| appsPythonHome311 | /opt/python3.11.1/bin | TC Exchange Apps |
| batchApiEnabled | true | Batch API |
| bulkIndicatorEnabled | true | Bulk Indicator Export |
| CALEnabled<br>(four different CAL™ settings) | true | Collective Analytics Layer (CAL) |
| dnsEnabled | true | DNS Data Services |
| documentStorageType | LOCAL or AWS<br>**Note**: As of ThreatConnect version 7.2, ThreatConnect supports the Amazon Web Services® (AWS) Default Credential Provider Chain. This allows users to set the **documentStorageType** setting to **AWS** | Document Storage |

| | | |
|---|---|---|
| | without needing to enter AWS credentials on the **System Settings** screen. | |
| searchEnabled | true (Refer to the "Configuring OpenSearch on ThreatConnect" section.) | OpenSearch |
| emailEnabled | true | Email Notifications and Alerts |
| ipGeoEnabled | true | IP GEO |
| keychainEnabled | true (Refer to the "Setting Master Key for Keychain" section.) | App Credential Encryption |
| logToFile | true | App Logging |
| mailInboundEnabled | true | Feed and Phishing Mailboxes |
| reverseWhoisEnabled | true | Reverse Whois Data Services |
| secureSystemUrl | https://hostname | Images and Links in Emails and Posts |
| sourceFeedMonitorEnabled | true | HTTP Source Feeds |
| systemDisplayName | your_hostname | Email Recognition |
| systemEmailAddressAccount | your_sysadmins_email | Account Questions |
| systemEmailAddressNotification | your_sysadmins_email | Notification Questions |

| | | |
|---|---|---|
| systemSubjectName | your_instance_name | Instance Recognition |
| systemUrl | http://hostname | Images and Links in Emails and Posts |
| taskEmailMonitorEnabled | true | Workflow Task Emails |
| taxiiExchangeMonitorEnabled | true | STIX™/TAXII™ Feeds |
| threatAssessMonitorEnabled | true | ThreatAssess |
| threatDeprecationMonitor | true | Indicator Deprecation |
| whoisEnabled | true | Whois Data Services |

# Configuring OpenSearch on ThreatConnect

1. Log into ThreatConnect with a System Administrator account.
2. On the top navigation bar, hover the cursor over **Settings** and select **System Settings**. The **System Settings** screen will be displayed (Figure 6).
3. Select **ALL** in the vertical column on the left. Configure the settings ***in this exact order***:
   a. **documentStorageType**: Value = AWS or Local. (Review information on document storage in *ThreatConnect Account Administration Guide* and *ThreatConnect System Administration Guide*.)
   b. **searchUrl**: Value = URL
   c. Scroll down and click **Save Settings.**
   d. **logToSearchCluster**: Value = true (checkbox is selected)
   e. Scroll down and click **Save Settings.**
   f. **searchEnabled**: Value = true (checkbox is selected)

g. Scroll down and click **Save Settings.**

4. Click the **CREATE SEARCH INDEX** button. The **Search Index Configuration** screen will be displayed with the **Setup** tab highlighted (Figure 8). The **Perform search indexing on database source** checkbox will index items currently existing in the ThreatConnect database when selected. The **Load file contents into search index** checkbox will index objects currently existing in document storage when selected.

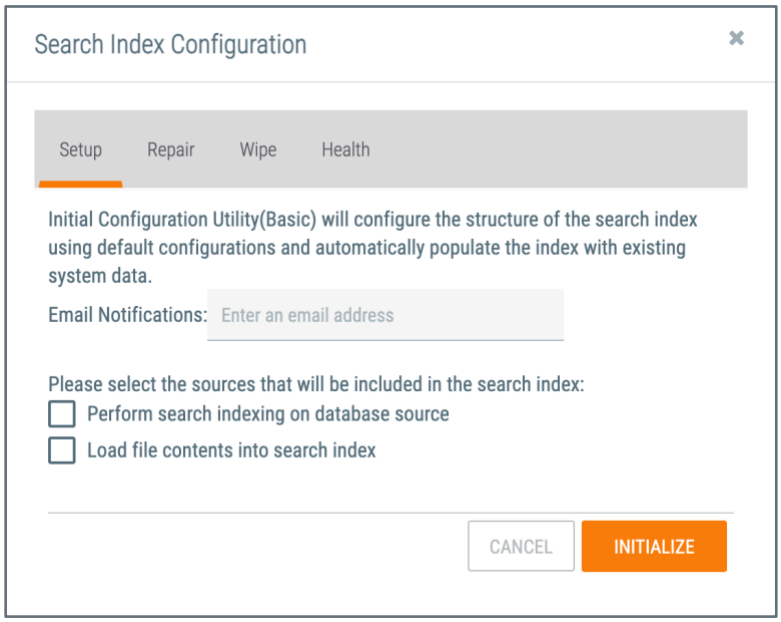


**Figure 8**

**Note**: The high availability of the index is not critical due to the amount of data. The index can be re-created at a rate of ~300,000 records/10 minutes.

5. The **Repair** and **Wipe** tabs should be accessed only by *advanced users*, as these utilities will delete existing data.

6. Click the **Health** tab to view the health and status of the OpenSearch nodes, index, and cluster.

7. Return to the **Setup** screen, select both checkboxes, and click the **INITIALIZE** button.

8. The indexing status will be logged in the log file specified for ThreatConnect. Review the log by accessing the **Logs** tab of the **System Settings** screen.

# Creating an Organization

The System Organization account cannot be used for creating Indicators or performing analyses. Its sole job is system configuration and user administration. To generate additional users, add Indicators, etc., an Organization (Org) must be created. See *ThreatConnect Organization Administration Guide* for information on creating and customizing Organizations.

# Appendix A: PostgreSQL Installation and Configuration

This section describes how to configure the PostgreSQL server and database to prepare it for usage with ThreatConnect, instead of using the MySQL database.

## PostgreSQL Download and Installation

The most recent version of PostgreSQL can be downloaded and installed directly from https://www.postgresql.org/download/linux/redhat/. With the repository now installed and active, utilize **yum** to install the PostgreSQL server:

```
yum install postgresql14 postgresql14-server -y
```

Once the installation is complete, initialize the database setup:

```
/usr/pgsql-14/bin/postgresql-14-setup initdb
```

Once the database setup is complete, enable on startup and start the PostgreSQL service:

```
systemctl enable postgresql-14
systemctl start postgresql-14
```

### Port Configuration

To allow access to the PostgreSQL database from other servers, the firewall must be set up to allow incoming and outgoing connections:

```
firewall-cmd --permanent --zone=public --add-service=postgresql
firewall-cmd --reload
```

Alternative methods may be used for the port configuration, but port 5432 needs to be open to allow servers to connect and relay data to the PostgreSQL database.

# PostgreSQL Access Configuration

By default, PostgreSQL is not set up with a password. A password will be needed in order to change the authentication setting to force a password prompt for any access to the PostgreSQL console or run scripts. To add a password to the postgres account, run the following commands:

```
su - postgres
psql -c "ALTER USER postgres WITH PASSWORD '<new-password>'";
```

This set of commands will change the password for the **postgres** account in the PostgreSQL application to what was inserted in between the single quotes.

To allow access to the PostgreSQL server and databases, incoming connections need to be defined. The settings suggested in the following code block will force all local connections to the PostgreSQL to use MD5 authentication (i.e., passwords), as well as define the same settings for the IP address(es) that will connect to the PostgreSQL server:

```
#Line add to file:
host    all         <IP-address-or-CIDR-block> md5

#Line change in file:
local   all     all                         peer

to:
local   all     all                         md5

#Line change in file:
host    all     all             127.0.0.1/32            ident

to:

host    all     all             127.0.0.1/32            md5
```

The default location of the file is **/var/lib/pgsql/14/data/pg_hba.conf**.

## PostgreSQL Configuration Options

ThreatConnect requires the following options to be configured in the PostgreSQL configuration file that is located by default at **/var/lib/pgsql/14/data/postgresql.conf**:

```
listen_addresses = '<ip-of-PostgreSQL-server>'
synchronous_commit = off
effective_cache_size = 75% total memory available on server
shared_buffers = 25% of the effective_cache_size variable
work_mem = 32MB
max_parallel_workers = total CPU cores installed on the server -1
max_parallel_workers_per_gather = 3
max_worker_processes = total CPU cores installed on the server -1
max_connections = 300
```

Save the file and restart the postgresql-14 service to commit the changes made in both configuration files:

```
systemctl restart postgresql-14
```

## Creating the Users and the Database

Once the PostgreSQL database service is installed and running, log in to create a database and user for ThreatConnect. From the command line on the PostgreSQL database server, enter the following commands:

```
su - postgres
psql -U postgres -W
```

When prompted, enter the postgres password chosen upon creation.

A separate non-root user should be created for security reasons. In this example, the username **tcuser** and password **Password1!** were chosen. Enter the following command to create a new user:

```
CREATE ROLE tcuser WITH LOGIN PASSWORD 'Password1!';
```

Enter the command shown next to create a new database. In this example, the database name **“threatconnect”** and the user account **“tcuser”** that the ThreatConnect application will use  were chosen.

```
CREATE DATABASE "threatconnect" WITH OWNER "tcuser" ENCODING 'UTF8' LC_COLLATE =
'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8';
```

**Note**: Any database name and user account may be used, but they will be referenced later in the installation and configuration process.

Grant permissions on the newly created user, allowing it to access the tables within this database. Enter the following commands to log in to PostgreSQL and add privileges to the new user:

```
GRANT ALL PRIVILEGES ON DATABASE threatconnect TO tcuser;
\c threatconnect
ALTER TYPE text OWNER to tcuser;
ALTER TYPE bool OWNER to tcuser;
```

**Important**: The foregoing example assumes that the database was named **threatconnect** and the user was named **tcuser**.

## Creating ThreatConnect Tables

Before starting this process, download the ThreatConnect binary zip file and have it extracted. The **threatconnect/app/scripts/postgres** directory contains a **.sql** script to create the initial tables for ThreatConnect within the database just defined. Browse to the folder containing the threatconnect-***<version>*.sql** file, run the command shown next, and enter the root password again when prompted.

**Note**: The install .sql script will use the default "public" schema under the database.

```
su - postgres
psql -U tcuser -d threatconnect -f
</opt/threatconnect/app/scripts/postgres/ThreatConnect-<version>.sql
```

**Note**: This command specifies **threatconnect** as the name of the database. If a different name for the database was chosen, use that instead.

## Creating Cast Tables

ThreatConnect includes and expects both smallint and Boolean values. The postgres user must be the owner of the smallint and Boolean data types to ensure that the column "hidden" in the table "blueprint" can use both smallint and Boolean values.

Log into postgres as the postgres user (i.e., the current owner of the types). Execute the following commands separately:

```
psql -U postgres -d threatconnect -W
```

```
ALTER TYPE text OWNER TO tcuser;
```

```
ALTER TYPE bool OWNER TO tcuser;
```

**Note**: These commands assume that the name of the database user is **tcuser**. If another name is being used, replace tcuser with that name in the command.

Log in to postgres as **tcuser** (or the defined database user) and create the missing casts by executing the following commands:

**Important**: An error may occur if the objects already exist.

```
psql -U tcuser -d threatconnect -W

CREATE FUNCTION public.boolean1(i smallint) RETURNS boolean
    LANGUAGE sql IMMUTABLE STRICT
    AS $$SELECT (i::smallint)::int::bool;$$;
```

```
CREATE FUNCTION public.inttobool(val boolean, num integer) RETURNS boolean
    LANGUAGE plpgsql
    AS $$
begin
    return public.inttobool(num,val);
end;
$$;
```

```
CREATE FUNCTION public.inttobool(num integer, val boolean) RETURNS boolean
    LANGUAGE plpgsql
    AS $$
begin
if num=0 and not val then
    return true;
elsif num=1 and val then
    return true;
else return false;
end if;
end;
$$;
```

```
CREATE FUNCTION public.notinttobool(val boolean, num integer) RETURNS boolean
    LANGUAGE plpgsql
    AS $$
begin
    return not public.inttobool(num,val);
end;
$$;
```

```
CREATE FUNCTION public.notinttobool(num integer, val boolean) RETURNS boolean
    LANGUAGE plpgsql
    AS $$
begin
    return not public.inttobool(num,val);
end;
$$;
```

```
CREATE FUNCTION public.text(date) RETURNS text
    LANGUAGE sql IMMUTABLE STRICT
    AS $_$SELECT textin(date_out($1));$_$;
```

```
CREATE FUNCTION public.text(smallint) RETURNS text
    LANGUAGE sql IMMUTABLE STRICT
    AS $_$SELECT textin(int2out($1));$_$;
```

```
CREATE FUNCTION public.text(integer) RETURNS text
    LANGUAGE sql IMMUTABLE STRICT
    AS $_$SELECT textin(int4out($1));$_$;
```

```
CREATE FUNCTION public.text(oid) RETURNS text
    LANGUAGE sql IMMUTABLE STRICT
    AS $_$SELECT textin(oidout($1));$_$;
```

```
CREATE CAST (date AS text) WITH FUNCTION public.text(date) AS IMPLICIT;
```

```
CREATE CAST (smallint AS boolean) WITH FUNCTION public.boolean1(smallint) AS
IMPLICIT;
```

```
CREATE CAST (smallint AS text) WITH FUNCTION public.text(smallint) AS IMPLICIT;
```

```
CREATE CAST (integer AS text) WITH FUNCTION public.text(integer) AS IMPLICIT;
```

```
CREATE CAST (oid AS text) WITH FUNCTION public.text(oid) AS IMPLICIT;
```

To verify that everything has been done correctly, run the following **select** command, which should return results without any errors:

```
select * from role where active = true;
```

# Appendix B: Re-Authenticating the Keystore and SSL Certificate Configuration

## Re-Authenticating the Keystore

ThreatConnect requires the use of HTTPS and uses Secure Sockets Layer (SSL) certificates to guarantee security to users. By default, ThreatConnect is pre-configured with a keystore containing a self-signed SSL certificate. For the sake of authenticity, users may wish to replace this certificate with their SSL certificate. This option in the installer allows a user to re-authenticate if the keystore is changed. Please note that ThreatConnect requires that the following conditions are met:

- There is a Java keystore **.jks** file located at **config/keystore.jks**.
- This keystore contains a key pair with alias labeled as **tc**.
- The password for this keystore has been entered via the **Re-authenticate SSL keystore password** option in the setup script.
- The keystore password *and* the key password are the same.

Directions for configuring an additional keystore are provided in the next section. Note that there are many ways to generate a keystore. The examples in the next section use **keytool**, as it is included with the JDK software listed in the "Software" section.

## SSL Certificates

### Generating a Self-Signed SSL Certificate

To use a new keystore with a self-signed certificate, refer to the **keytool** commands in this section. First, rename the current **keystore.jks** file in the directory in case it is needed later. Then run the following command in the **config** directory:

```
keytool -genkeypair -alias tc -keyalg RSA -keystore keystore.jks -storetype JKS -
storepass yourPassword --dname "CN=threatconnect"
```

Do not enter a key password for the **tc** alias when prompted. Leave this field blank, and press **Enter** to confirm the use of the keystore password. (ThreatConnect and its dependent packages rely upon this behavior in the keystore.) This action will generate a keystore named **keystore.jks**, with an alias of **tc** and a password of **yourPassword**. This password will need to be entered into the setup script as detailed in the previous section to meet the three specified invariants.

## Generating a CA-Signed SSL Certificate

If a certification authority (CA) certificate will be installed on the ThreatConnect Application server, a new keystore must be created. To create a new keystore, log into the ThreatConnect Application server, and access a directory where these files can be stored without impacting ThreatConnect (e.g., **/home** or **/opt**). Run the following command to create a new keystore:

```
keytool -genkey -alias server -keyalg RSA -keysize 2048 -keystore keystore.jks
```

A series of prompts will be given:

- **Enter keystore password**: *‹entered by the client›*
- **Re-enter new password**: *‹entered by the client›*
- **What is your first and last name?**: *‹The FQDN or DNS name that will be used to access the server should be entered here.›*
- **What is the name of your organizational unit?**: *‹entered by the client›*
- **What is the name of your organization?**: *‹entered by the client›*
- **What is the name of your city or locality?**: *‹entered by the client›*
- **What is the name of your state or province?**: *‹entered by the client›*
- **What is the two-letter country code for this unit?**: *‹entered by the client›*

A prompt will appear to ensure that all data entered are correct. Type **yes** if no changes need to be made.

- **Enter key password for ‹*server*›**: *‹Press **Return** if this password is the same as the keystore password.›*
- *‹Press **Enter**.›*

Once the keystore is made, run the following command to create the alias needed for ThreatConnect:

```
keytool -changealias -alias server -destalias tc -keystore keystore.jks
```

To obtain a keystore that contains a certificate signed by a CA, generate a certificate-signing request:

```
keytool -certreq -keyalg RSA -alias tc –keystore keystore.jks -file certreq.csr
```

This command will generate a certificate-signing request file, **certreq.csr**. Present this file to the CA, who will return a signed file.

## Option 1: CA returns with .crt file

The user may be required to import the root CA certificate as well, to maintain the trust. To import root certificates, request them from a CA, and import them using the following command, where **Root.crt** is the name of the certificate file provided by the CA:

```
keytool -import -keystore keystore.jks -alias RootCA -file Root.crt
```

The user may be required to import the intermediate CA certificate as well, to maintain the trust. To import intermediate certificates, request them from a CA, and import them using the following command, where **intermediate.crt** is the name of the certificate file provided by the CA:

```
keytool -import -keystore keystore.jks -alias intermediateCA -file intermediate.crt
```

Once the other certificates are imported, import the file back into the keystore using the following command, where **server.crt** is the name of the file returned by the CA:

```
keytool -import -alias tc -keystore keystore.jks -file server.crt
```

## Option 2: CA returns with .pem file

If the CA returns with **.pem** files, convert them for use in the keystore. Upload the certificates to the ThreatConnect Application server, and import them using the following command, where **intermediate_cert.pem** is the name of the certificate file provided by the CA and **signed_cert.pem** is the name of the server certificate:

```
openssl crl2pkcs7 -nocrl -certfile signed_cert.pem -out cert.p7b -certfile
intermediate_cert.pem
```

This action will create a new file named **cert.p7b**. This file will be used to import the certificates into the keystore with the following command:

```
keytool -import -alias tc -keystore keystore.jks -trustcacerts -file cert.p7b
```

## Option 3: CA returns with .cer file

If the CA returns with **.cer** files, ensure that the root CA and intermediate CA files have been returned for import. If the root and intermediate certificates are not available, the keytool cannot maintain the trust on the signed certificate. Start by copying all certificate files to the ThreatConnect Application server and importing the root CA certificate via the following command, where **rootca.cer** is the name of the root certificate file:

```
keytool -import -alias RootCA -keystore keystore.jks -trustcacerts -file rootca.cer
```

Use the following command to continue with the intermediate CA certificate, where **intermediateca.cer** is the name of the intermediate CA certificate file:

```
keytool -import -alias IntermediaCA -keystore keystore.jks -trustcacerts -file
intermediateca.cer
```

Finally, use the following command to add the certificate for the ThreatConnect Application server, where **tc_cert.cer** is the name of the server certificate:

```
keytool -import -alias tc -keystore /path/to/keystore.jks -trustcacerts -file
tc_cert.cer
```

After one of the options is completed, a new keystore will be ready to be used by ThreatConnect. To use the new keystore, first stop the ThreatConnect service. Rename the **keystore.jks** file in **/opt/threatconnect/config** to **keystore.jks.original**. Copy the newly created **keystore.jks** file to **/opt/threatconnect/config**/. Be sure to change ownership on the new **keystore.jks** file by running the following command:

```
chown threatconnect:threatconnect /opt/threatconnect/config/keystore.jks
```

Next, the certificates need to be added to the JRE keystore located at **/usr/java/jdk<version>/jre/lib/security/**:

```
cd /usr/java/jdk<version>/jre/lib/security/
```

For a root CA, enter the following command:

```
keytool -import -alias RootCA -keystore cacerts -trustcacerts -file
<rootCA_filepathandname>
```

For an intermediate CA, enter the following command:

```
keytool -import -alias IntermediateCA -keystore cacerts -trustcacerts -file
<intermediateCA_filepathandname>
```

For a server CA, enter the following command:

```
keytool -import -alias tc -keystore cacerts -trustcacerts -file
<serverCA_filepathandname>
```

Next, log in as the **threatconnect** user:

```
su – threatconnect
```

Change the current location to **/opt/threatconnect/app/**:

```
cd /opt/threatconnect/app/
```

Run **setup.sh**:

```
./setup.sh
```

Select the option to change the SSL password. Enter the password that was created for the new keystore two times. Log out as the **threatconnect** user, and start the **threatconnect** service. Once the **threatconnect** service is fully started, log into the Web interface of ThreatConnect. Verify that the certificate is found by the browser and that SSL is working.

For instructions on ensuring all certificate requirements are met for the ThreatConnect framework and all Apps, regardless of whether the App is Java or Python based, see the "Appendix D: Integrating Self-Signed Certificates with Java and Python Apps" section.

# Appendix C: Configuring SSL Transmission with MySQL and ThreatConnect

This section will guide the user through enabling SSL transmission between the MySQL server and the ThreatConnect application. This configuration is an optional setting and is not required for ThreatConnect to function.

First, the SSL certificates for the MySQL server will need to be located and utilized. Based on the default installation of MySQL defined in this document, the installation process will have autogenerated the SSL certificates for MySQL to use. The certificates must be saved as a **.pem** format for MySQL.

Update MySQL server configuration (**/etc/my.cnf**) with the following lines:

```
#SSL Settings
ssl-ca=/var/lib/mysql/ca.pem
ssl-cert=/var/lib/mysql/server-cert.pem
ssl-key=/var/lib/mysql/server-key.pem

[client]
ssl-ca=/var/lib/mysql/ca.pem
ssl-cert=/var/lib/mysql/server-cert.pem
ssl-key=/var/lib/mysql/server-key.pem
```

**Important**: These settings need to be placed at the end of the **[mysql]** section.

The foregoing paths and keys are the autogenerated ones from the installation of MySQL. If they are not to be used, simply substitute the path and filenames.

Once the configuration is updated, save and close the file. The service will need to be restarted to apply the settings:

```
systemctl restart mysqld
```

Log into the MySQL console, and run the following command:

```
status;
```

The status output will show the settings used in the connection. SSL will be defined, and the SSL settings that are default will be set.

The file defined as **ssl-cert** in the **[client]** section of the SSL Settings will need to be copied to the ThreatConnect application server. The certificate will need to be installed in the ThreatConnect keystore. The command to import the file is as follows:

```
keytool -import -alias <IP-or-FQDN-of-MYSQL-server> -keystore
/opt/threatconnect/config/keystore.jks -file <path-to-server-cert-file>
```

**Note**: This command assumes the default installation of ThreatConnect in **/opt/threatconnect**. Adjust if the installation path is different.

There will be a prompt for the password to the keystore. Once the password is entered, type **yes** to save the certificate into the keystore.

The process will need to be duplicated into the Java keystore on the ThreatConnect application server. Run the following command:

```
keytool -import -alias <IP-or-FQDN-of-MYSQL-server> -keystore
/usr/java/latest/jre/lib/security/cacerts -file <path-to-server-cert-file>
```

The password for the keystore is **changeit** . Type **yes** to save the password into the keystore.

Once the certificate has been imported, the ThreatConnect configuration will need to be updated to use the SSL connection when communicating with MySQL. The default location for the file is **/opt/threatconnect/config/threatconnect.xml.**

Change two lines in the config. The original lines look like this:

```
<connection-
url>jdbc:mysql://mysql.example.com:3306/threatconnect?autoReconnect=true&amp;rewrit
eBatchedStatements=true</connection-url>
```

Update the lines to include the SSL setting, such as in the following example:

```
<connection-
url>jdbc:mysql://192.168.1.22:3306/threatconnect?autoReconnect=true&amp;rewriteBatc
hedStatements=true&amp;useSSL=true</connection-url>
```

Save the configuration file.

Start the ThreatConnect service, or continue the installation process defined in the ThreatConnect installation section.

Verify that the application is using SSL to connect by connecting to the MySQL server and running the following command:

```
tcpdump -l -i <name-of-nic> -w - src or dst port 3306 | strings
```

The output from the foregoing command should just be garbled text. If the transmission between the servers is legible, run through the procedure in Appendix C again to ensure no configurations or commands were missed.

# Appendix D: Integrating Self-Signed Certificates with Java and Python Apps

Follow the steps in this section to ensure all certificate requirements are met for the ThreatConnect framework and all Apps, regardless of whether an App is Java or Python based.

1. Collect all necessary certificates in **.pem**, **.der**, or **.cer** format.
2. Copy all intermediate and root self-signed certificates and CAs into the following directory:

   ```
   /usr/share/pki/ca-trust-source/anchors
   ```

3. Run the following command:

   ```
   update-ca-trust
   ```

   **Important**: This command updates multiple files to include the ones mentioned in Steps 4 and 5 and the Java cacerts keystore.

4. **Optional**: Run the following command and verify that the date is today's date to validate that the appropriate files were updated:

   ```
   ll /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem
   ```

5. Add the following to the **/etc/environment** file:

   ```
   REQUESTS_CA_BUNDLE=/etc/ssl/certs/ca-bundle.crt
   SSL_CERT_FILE=/opt/threatconnect/certificate/server.pem
   ```

   **Note**: The value in **SSL_CERT_FILE** should be the full path to the self-signed certificate for the ThreatConnect instance. It is recommended to place this file in a **certificate** or **cert** directory under the ThreatConnect installation directory.

6. Stop and then restart the ThreatConnect service.