ThreatQuotient

Resilient Custom Threat Service (CTS) User Guide

Version 1.1.1

October 02, 2023

ThreatQuotient
20130 Lakeview Center Plaza Suite 400
Ashburn, VA 20147

ThreatQ Supported

Support
Email: support@threatq.com
Web: support.threatq.com
Phone: 703.574.9893

Contents

Warning and Disclaimer

ThreatQuotient, Inc. provides this document “as is”, without representation or warranty of any kind, express or implied, including without limitation any warranty concerning the accuracy, adequacy, or completeness of such information contained herein. ThreatQuotient, Inc. does not assume responsibility for the use or inability to use the software product as a result of providing this information.

Copyright © 2023 ThreatQuotient, Inc.
All rights reserved. This document and the software product it describes are licensed for use under a software license agreement. Reproduction or printing of this document is permitted in accordance with the license agreement.

Support

This integration is designated as **ThreatQ Supported**.

Support Email: support@threatq.com
Support Web: https://support.threatq.com
Support Phone: 703.574.9893

Integrations/apps/add-ons designated as **ThreatQ Supported** are fully supported by ThreatQuotient's Customer Support team.

ThreatQuotient strives to ensure all ThreatQ Supported integrations will work with the current version of ThreatQuotient software at the time of initial publishing. This applies for both Hosted instance and Non-Hosted instance customers.

> ThreatQuotient does not provide support or maintenance for integrations, apps, or add-ons published by any party other than ThreatQuotient, including third-party developers.

Integration Details

ThreatQuotient provides the following details for this integration:

Current Integration Version	1.1.1
Compatible with ThreatQ Versions	>= 4.24.0
Python Version	3.6
Support Tier	ThreatQ Supported

Introduction

The ThreatQuotient for Resilient Custom Threat Service (CTS) integration implements Resilient's Custom Threat Service API, enabling automatic artifact lookups in ThreatQ.

> The Resilient CTS integration is a component of the Resilient App and must be used along with the Resilient Connector and Resilient Functions integrations.

Prerequisites

This section details the prerequisites required to run the ThreatQuotient for Resilient Custom Threat Service.

Python 3.6

In order for this app to run properly, Python 3.6 must be installed. If you are running the app on the Resilient server, Python 3 is pre-installed and located at, `/opt/rh/rh- python36/root/usr/bin/pythonc`. However, if you are running the app on a Resilient Integration Server, follow these steps to install Python 3 if it is not already installed.

1. SSH into your Resilient Server or Resilient Integration Server.
2. Install Python 3.6
   - You can follow this guide to install it: https://phoenixnap.com/kb/how-to- install-python-3-centos-7
   - If you prefer not to use SCL, you can follow this guide: https://www.rosehosting.com/blog/how-to-install-python-3-6-4-on-centos- 7/

Installation

This chapter will show how to install the ThreatQuotient for Resilient Custom Threat Service.

Installing the CTS (Custom Threat Service) Integration

1. Transfer the CTS for ThreatQ integration whl file your Resilient Server (or Resilient Integration Server).
2. Change your user to the user that will be running the resilient integrations.

   > Create the user now if you currently do not have one for running Resilient integrations. While Root can be used, ThreatQuotient does not recommended using it for this process.

3. Create a .resilient folder in your user's home directory, if it has not already been created, with the following command:

   ```
   mkdir ~/.resilient
   ```

4. Navigate into that directory.

   ```
   cd ~/.resilient
   ```

5. Create a Python3 virtual environment to house your CTS integrations. This step can be skipped if you already have a virtual environment set up.

```
python3 -m venv cts-env
```

6. Activate the Python3 virtual environment:

```
source cts-env/bin/activate
```

7. Install the required dependencies for python 3.6:

```
pip install setuptools-scm==6.4.2
```

8. Install the whl file:

> Confirm that you have the python 3.6 SCL environment activated (if you installed Python using SCL).

```
pip install wheel rc_cts_threatq--<version>-py3-none-any.whl
```

9. If you have not created a resilient configuration file, do so now:

> If you run this even though it's already been created, do not over- write the old configuration.

```
resilient-circuits config -c
```

> The default config location is `~/ .resilient/app.config`.

10. Add the new configuration to the config file by running the following:

```
resilient-circuits config -u
```

11. Edit the `app.config` file with your configuration for the integration and Resilient, then save.

    Notes

    - Under the **[resilient]** section, change the following:
      - Comment out `api_key` and `api_key_secret` by prefixing the lines with a `#`.
      - Comment the `email` and `password` back in by removing the `#`.
      - Change the `Org` field to reflect your Resilient Organization name.
      - Uncomment out `cafile` and set the value to `false` if you do not have a certificate file for your Resilient instance. If you do, enter the path here.
    - Under the **[custom_threat_service_threatq]** section, fill out the following fields:

FIELD	DESCRIPTION
Host	Your ThreatQ Hostname
Username	Your ThreatQ Username
Password	The password associated with ThreatQ Username supplied above.
CID	Your ThreatQ Client ID (OAuth). This can be found by viewing you ThreatQ account profile details.

> If you are running this on a an integration server, under the [webserver] section, comment in `server`.

12. Set a location for the lock file:

```
export APP_LOCK_FILE=~/.resilient/cts.lock
```

> This will only persist during your current SSH session.

13. Run resilient-circuits to execute the installed integrations:

```
resilient-circuits run
```

14. Test out the integration by creating a adding a supported artifact to an incident in Resilient.

    Notes

    - You should see a spinning wheel popup next to the artifact. Wait until it disappears and then refresh the page. If the artifact was in ThreatQ, it will turn red and you will be able to view the context from ThreatQ.
    - If you run into networking or authentication issues, make sure that your **app.config file** is configured correctly.
    - Contact support@threatq.com if you are running into any other issues.

15. If you are running this on an integration server, you must open your firewall to allow connections on port 9000 (the configured webserver port):

```
firewall-cmd --zone=public --permanent --add- port=9000/tcp
firewall-cmd --reload
```

Configuring for Reboot

This section will describe how to configure the integration (resilient-circuits) to auto-run on reboot

1. Create a file for the new service.

```
sudo vi /etc/systemd/system/resilient_circuits_ cts.service
```

2. Paste the following into the file:

> Fill out required fields (between < >)
>
> Delete lines 5 and 6 completely (After=... and Required=...) if you are not installing this directly on the Resilient Server.

```
[Unit]
Description=Resilient-Circuits Service for CTS

# Comment out these 2 lines if you are running on an integration server
After=resilient.service
Requires=resilient.service

[Service]
Type=simple
User=<username>
WorkingDirectory=/home/<username>
ExecStart=/home/<username>/.resilient/cts-env/bin/resilient-circuits run
Restart=always
TimeoutSec=10
Environment=APP_CONFIG_FILE=/home/<username>/.resilient/app.config
Environment=APP_LOCK_FILE=/home/<username>/.resilient/cts.lock

[Install]
WantedBy=multi-user.target
```

3. Give the service file the correct privileges:

```
sudo chmod 664 /etc/systemd/system/resilient_circuits_cts.service
```

4. Reload the system daemon and enable the CTS service:

```
sudo systemctl daemon-reload

sudo systemctl enable resilient_circuits_ cts.service
```

5. Start the service:

```
sudo systemctl start resilient_circuits_cts
```

6. You can view the logs for the service by running the following:

```
sudo journalctl -u resilient_circuits_cts -f
```

> This will "follow" the logs in real-time.

Setting HTTPS

This section will show you how to setup the custom threat service with HTTPS. This is an optional step, however, is advised when using an integration server.

1. SSH into your Resilient server (or integration server)
2. Switch to the user that will run the integrations
3. Navigate into your .resilient directory where you installed the app.config file:

```
cd ~/.resilient
```

4. Generate your webserver certificate:

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout
ssl.pem -out ssl.pem
```

5. Uncomment and set the certfile option to the path to your .pem file:

```
certfile=~/.resilient/ssl.pem
```

6. Save your app.config file.
7. Restart the ThreatQ Custom Threat Service:

```
systemctl restart resilient_circuits_cts
```

8. Make sure that the service booted up correctly by checking the logs:

```
journalctl -u resilient_circuits_cts -f
```

> Make sure that when you configure the custom threat service on the Resilient instance, that you use https://.

Adding the Custom Threat Service

1. SSH into your Resilient Server as resadmin.
2. If you installed the integration on an integration server, attempt to ping the your integration server to make sure the servers can communicate:

```
ping <Integration Server IP Address or Host- name>
```

3. Add the custom threat service using `resutil`:

```
sudo resutil threatserviceedit -name "ThreatQ"-resturl "http://<CTS
Server-|127.0.0.1>:9000/cts/threatq_cts"
```

> Change `http://` to `https://` if you did the setup for SSL/HTTPS (on an integrations server).

> If you are using a proxy, include the following flag to bypass it when running this command: `-ignoreProxy true`.

4. If you did the setup for SSL/HTTPS, make sure you add the certificate to Resilient's Trust Manager by following these steps:

```
# Create the certificate (pem)
# Replace <Server Host/IP> with the hostname or IP for your Resilient
integration server (or Resilient host if you are running the webserver
from there)
echo | openssl s_client -connect <Server Host/IP>:9000 | sed -ne '/-BEGIN
CERTIFICATE-/,/-END CERTIFICATE-/p' > threatq_cts.pem

# Generate a keyvault UUID for custom certificates
openssl rand -hex 32 | sudo resutil keyvaultset -name custcerts -stdin

# Add the ThreatQ CTS certificate (pem) to the new key vault
echo yes | sudo keytool -importcert -trustcacerts -keystore /crypt/certs/
custcerts -storepass "$(sudo resutil keyvaultget -name "custcerts")" -file
threatq_cts.pem -alias "ThreatQ CTS"
```

5. Once it is added, you will want to test it using `resutil`:

```
sudo resutil threatservicetest -name "ThreatQ" -v
```

6. Check the running threat service (from Installing the CTS (Custom Threat Service) Integration) to confirm that everything is operating properly.

> Make sure you get a success message on the Resilient Server as well (threatservicetest).

7. Set up is now complete. The last step is to test the threat service.

Troubleshooting

Fixing a Banned IP Address

In the event that the Resilient server blocks an IP address, whether it be to your ThreatQ instance or to your Resilient Integration Server, follow these instructions to unban it.:

1. SSH into your Resilient server using the resadmin user.
2. Run the following command to confirm there's an banned IP:

   ```
   sudo -u postgres psql -c "select * from mon- app.ipban;" co3
   ```

3. Remove the banned IP by running command:

   ```
   sudo -u postgres psql -c "delete from monapp.ipban;" co3
   ```

4. Restart Resilient service by command (for RHEL systems):

   ```
   sudo systemctl restart resilient
   ```

Clearing the Custom Threat Service Lookup Queue

The ThreatQ Custom Threat Service automatically performs lookups for artifacts. If Resilient goes back and performs historical lookups for indicators causing a slow- down/clog in the lookup pipeline, follow these steps to clear the lookup queue. Note: This will clear the lookup queue for all custom threat services, not just ThreatQ lookups.

1. SSH into your Resilient server using the resadmin user.
2. Navigate to **/tmp** directory.
3. Create a file called **delete-cts-jobs.sql**.
4. If you have the script, SCP it over to the Resilient Server, into the **/tmp** directory.
5. Paste the following code into the file and then save.

   ```
   set search_path=monapp;
   begin;
   delete from qrtz_simple_triggers where trigger_name in (select job_name
   from qrtz_job_details where substring(encode(job_data, 'escape') from
   'com.co3.[a-z0-9A-Z\\.]+') = 'com.co3.threat.CustomThreatService');
   delete from qrtz_triggers where trigger_name in (select job_name from
   qrtz_job_details where substring(encode(job_data, 'escape') from 'com.co3.
   [a-z0-9A-Z\\.]+') = 'com.co3.threat.CustomThreatService');
   delete from qrtz_job_details where substring(encode(job_data, 'escape')
   from 'com.co3.[a-z0-9A-Z\\.]+') = 'com.co3.threat.CustomThreatService';
   commit;
   ```

6. Run the script against the Postgres Database.

Shell Script

```
sudo -u postgres psql -f /tmp/delete-cts- jobs.sql co3
```

Known Issues / Limitations

- Some Artifacts, such as Email Subject, Services, Strings, Email Attachment, are not supported.
- Only the first four attributes are parsed for the Password type Artifact. This issue is caused by a bug in Resilient.

Change Log

- **Version 1.1.1**
  - Resolved a ThreatQ instance invalid certificate error.
  - Resolved a compatibility issue with ThreatQ v5 and the Indicator Filter.
  - Resolved an issue where errors would trigger the IBM Security Soar instance to reboot.
  - Added **Known Issues / Limitations** chapter.
  - Updated required python version to 3.6.
- **Version 1.1.0 rev-a (Guide Update)**
  - Updated Install chapter steps to transfer and install .whl instead of tar.gz files.
  - Added a note to the Adding the Custom Threat Service chapter regarding bypassing proxies.
- **Version 1.1.0**
  - Added support for fetching additional relationships (malware, attack patterns, etc.)
  - Minor bug fixes
- **Version 1.0.0**
  - Initial release