ThreatQuotient

ThreatQuotient for Resilient Custom Threat Service Guide

Version 1.0.1

Friday, January 31, 2020

ThreatQuotient

11400 Commerce Park Dr., Suite 200

Reston, VA 20191

Support

Email: support@threatq.com

Web: support.threatq.com

Phone: 703.574.9893

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.

Last Updated: Friday, January 31, 2020

Contents

Introduction

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

Preface

This guide provides the information necessary to implement the ThreatQuotient for Resilient Custom Threat Service. This document is not specifically intended as a site reference guide. It is assumed that the implementation engineer has experience installing and commissioning the ThreatQuotient Apps and integrations covered within the document, as well as the experience necessary to troubleshoot at a basic level.

Audience

This document is intended for use by the following parties:

1. ThreatQ Security/Engineers.
2. ThreatQuotient Professional Services Project Team & Engineers.

Versioning

- Current integration version: `1.0.1`
- Supported on ThreatQ versions `>= 4.24`

Assumptions

The following criteria is assumed to be in place and functional to allow the implementation of the ThreatQuotient for Resilient Custom Threat Service into the managed estate:

- All ThreatQuotient equipment is online and in service.
- All required firewall ports have been opened.

Prerequisites

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

Python 3.x

In order to run the Resilient Integrations for ThreatQuotient, you must install a Python 3 environment onto your Resilient (or integration) server.

1. SSH into your Resilient Server or Resilient Integration Server.
2. Install Python 3.2+
    - 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 (rc_cts...py3.whl) onto 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 .whl file (and wheel for compatibility):

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

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

8. 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** overwrite the old configuration.

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

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

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

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

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

- Under the `[resilient]` section, change the following:
  - Comment out `api_key` and `api_key_secret` by prefixing the lines with a `#`
  - Uncomment out `email` and `password` 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/IP Address.
Username	Your ThreatQ Username.
Password	Your ThreatQ Password.
CID	Your Client ID (OAuth) found in your ThreatQ profile.

- If you are running this on a an integration server, under the `[webserver]` section, change the following:
  - Uncomment out `server`

11. Set a location for the lock file:

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

This will only persist during your current SSH session

12. 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
```

13. 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
    - If you are running into any other issues, please contact support@threatq.com

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

```
resilient-circuits run
```

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 4 and 5 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 integ-
ration 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-
E=/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 up 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 integration server), change `http://` to `https://`

4. If you did the setup for SSL/HTTPS, make sure you add the certificate to Resilient's TrustManager 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 run-
ning the webserver from there)
Table Row Outside Table:
Table Cell Outside Table:  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 cust-
certs -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.

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.

delete-cts-jobs.sql

```
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
```

Trademarks and Disclaimers

THE SUBJECT AND SPECIFICATIONS INCLUDING ALL INFORMATION REGARDING THE PRODUCTS IN THIS DOCUMENT ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT ARE BELIEVED TO BE ACCURATE AT THE TIME OF WRITING BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE TERMS AND CONDITIONS WHEN PURCHASED. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR THREATQUOTIENT REPRESENTATIVE FOR A COPY.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THIRD PARTY SUPPLIERS ARE PROVIDED “AS IS” WITH ALL FAULTS. THREATQUOTIENT AND THIRD-PARTY SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL THREATQUOTIENT OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF THREATQUOTIENT OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

It is wholly the customers responsibility for any design requirements and the utilization of any recommendations provided by ThreatQuotient. ThreatQuotient recommendations are based upon customer information provided to ThreatQuotient at the time of the services. ThreatQuotient shall not be liable for the accuracy or completeness of the customer information contained in the ThreatQuotient recommendations.

All documentation and deliverables shall be provided in the English language, unless specifically stated otherwise. or agreed before the commencement of any services in writing. Any costs incurred by ThreatQuotient as a result of translations requested by Customer shall be Customer's responsibility.

In the event of any conflict between this English version and the translation(s), the English version will prevail.

ThreatQuotient and the ThreatQuotient Rhino Logo are trademarks of ThreatQuotient, Inc.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.