ThreatQuotient

# Exploit DB Connector Guide

**Version 1.1.0**

July 20, 2021

**ThreatQuotient**
11400 Commerce Park Dr., Suite 200
Reston, VA 20191

**Support**

| | |
|---|---|
| Email: | support@threatq.com |
| Web: | support.threatq.com |
| Phone: | 703.574.9893 |

# Contents

# Warning and Disclaimer

# Versioning

- Current integration version: `1.1.0`
- Supported on ThreatQ versions >= `4.34.0`

There are two versions of this integration:

- Python 2 version
- Python 3 version

# Introduction

The Exploit DB for ThreatQuotient Integration imports Exploit DB exploits into ThreatQ. Included with the exploits are any related CVE indicators, as well as any vulnerable applications.

# Installation

The connector can be installed from the ThreatQuotient repository with YUM credentials or offline via a .whl file.

**Upgrading Users** - Review the Change Log for updates to configuration parameters before updating. If there are changes to the configuration file (new/removed parameters), you must first delete the previous version's configuration file before proceeding with the install steps listed below. Failure to delete the previous configuration file will result in the connector failing.

1. Install the connector using one of the following methods:

   **ThreatQ Repository**

   a. Run the following command:

   ```
   <> pip install tq_conn_exploit_db
   ```

   **Offline via .whl file**

   To install this connector from a wheel file, the wheel file (.whl) will need to be copied via SCP into your ThreatQ instance.

   a. Download the connector whl file with its dependencies:

   ```
   <> mkdir /tmp/tq_conn_exploit_db

      pip download tq_conn_exploit_db -d

      /tmp/tq_conn_exploit_db/
   ```

   b. Archive the folder with the .whl files:

   ```
   <> tar -czvf tq_conn_exploit_db.tgz /tmp/ tq_conn_exploit_db/
   ```

   c. Transfer all the whl files, the connector and all the dependencies, to the ThreatQ instance.

   d. Open the archive on ThreatQ:

   ```
   <> tar -xvf tq_conn_exploit_db.tgz
   ```

   e. Install the connector on the ThreatQ instance.

The example assumes that all the whl files are copied to `/tmp/conn` on the ThreatQ instance.

```
pip install /tmp/conn/ tq_conn_exploit_db-<version>-
<python version>-none-any.whl --no-index --find-links /
tmp/conn/
```

A driver called `tq-conn-exploit-db` will be installed. After installing with `pip` or `setup.py`, a script stub will appear in `/usr/bin/tq-conn-exploit-db`.

2. Once the application has been installed, a directory structure must be created for all configuration, logs and files, using the `mkdir -p` command. Use the commands below to create the required directories:

```
mkdir -p /etc/tq_labs/
mkdir -p /var/log/tq_labs
```

3. Perform an initial run using the following command:

```
tq-conn-exploit-db -v3 -ll /var/log/tq_labs/ -c /etc/tq_labs/
```

4. Enter the following parameters when prompted:

| PARAMETER | DESCRIPTION |
|---|---|
| **ThreatQ Host** | This is the host of the ThreatQ instance, either the IP Address or Hostname as resolvable by ThreatQ. |
| **Client ID** | This is the OAuth id that can be found at Settings Gear → User Management → API details within the user's details. |
| **Email Address** | This is the User in the ThreatQ System for integrations. |
| **Password** | The password for the above ThreatQ account. |
| **Status** | This is the default status for objects that are created by this Integration. |

## Example Output

```
tq-conn-exploit-db -v3 -ll /var/log/tq_labs/ -c /etc/tq_labs/
ThreatQ Host: <ThreatQ Host IP or Hostname>
Client ID: <ClientID>
E-Mail Address: <EMAIL ADDRESS>
Password: <PASSWORD>
Status: Review
Connector configured. Set information in UI
```

You will still need to configure and then enable the connector.

# Configuration

ThreatQuotient does not issue API keys for third-party vendors. Contact the specific vendor to obtain API keys and other integration-related credentials.

To configure the integration:

1. Navigate to your integrations management page in ThreatQ.
2. Select the **Labs** option from the *Category* dropdown (optional).
3. Click on the integration to open its details page.
4. Enter the following parameters under the **Configuration** tab:

| PARAMETER | DESCRIPTION |
|---|---|
| **Import Unverified Exploits** | Setting this to **Yes** will import exploits that are not verified. |
| **Import Vulnerable Apps** | Setting this to **Yes** will import any vulnerable apps related to the exploit. The uploaded app's type will be `Malware Sample`. |

5. Review any additional settings available, make any changes if needed, and click on **Save**.
6. Click on the toggle switch, located above the *Additional Information* section, to enable it.

# Usage

Use the following command to execute the driver:

```
<> tq-conn-exploit-db -v3 -ll /var/log/tq_labs/ -c /etc/tq_labs/
```

## Command Line Arguments

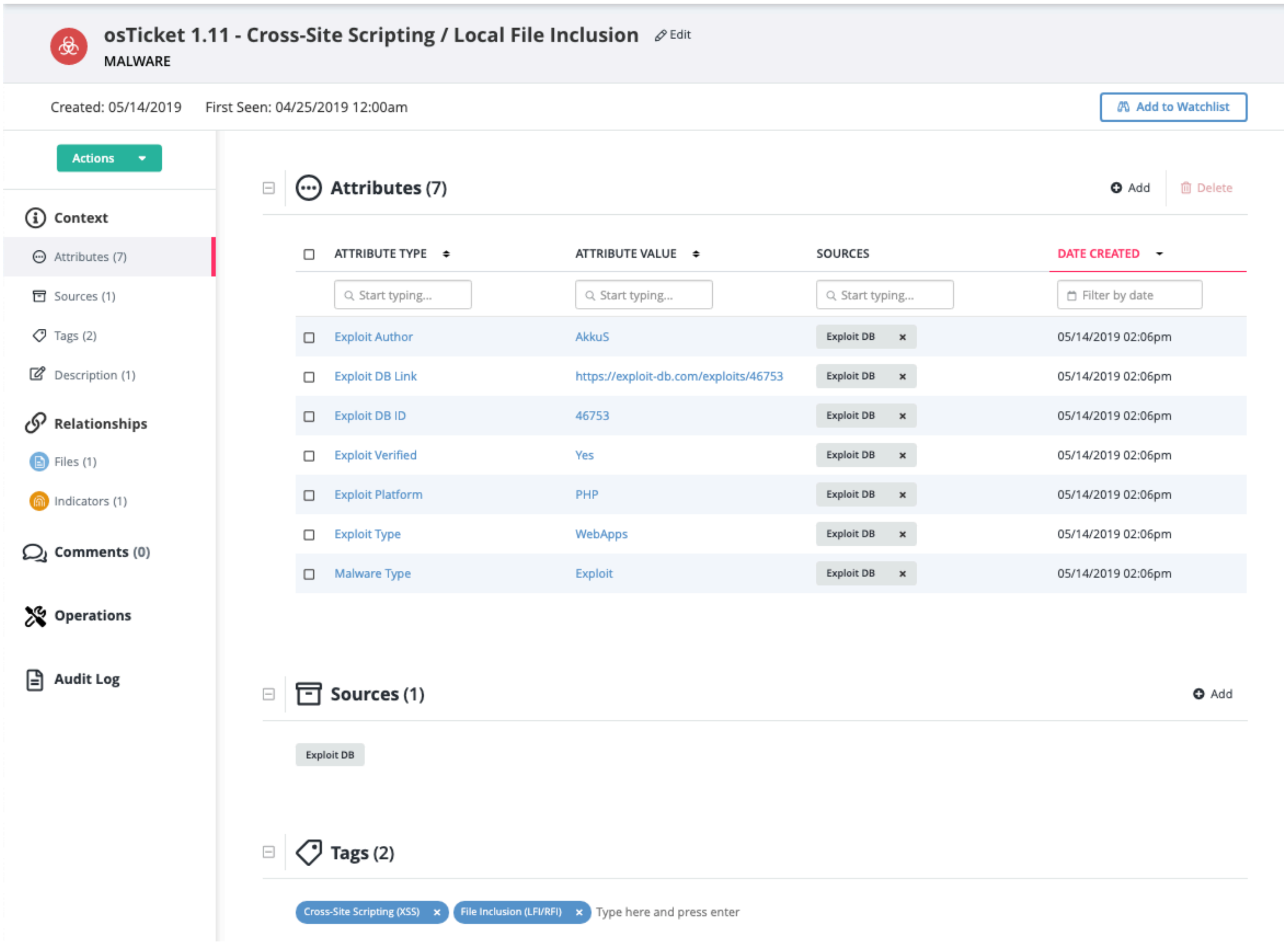
This connector supports the following custom command line arguments:

| ARGUMENT | DESCRIPTION |
|---|---|
| `-h`, `--help` | Shows this help message and exits. |
| `-ll LOGLOCATION`, `--loglocation LOGLOCATION` | Sets the logging location for the connector. The location should exist and be writable by the current. A special value of 'stdout' means to log to the console (this happens by default). |
| `-c CONFIG`, `--config CONFIG` | This is the location of the configuration file for the connector. This location must be readable and writable by the current user. If no config file path is given, the current directory will be used. This file is also where some information from each run of the connector may be put (last run time, private oauth, etc.) |
| `-v {1,2,3}`, `--verbosity {1,2,3}` | This is the logging verbosity level where **3** means everything. The default setting is **1** (Warning). |
| `-n, --name` | This allows you to change the name of the connector. |
| `-d, --no-differential` | If exports are used in this connector, this will turn 'off' the differential flag for the execution. This allows debugging and testing to be done on export endpoints without having to rebuild the |

| ARGUMENT | DESCRIPTION |
|---|---|
| | exports after the test. THIS SHOULD NEVER BE USED IN PRODUCTION. |
| `-hist HISTORICAL, --historical HISTORICAL` | |

# Examples

## Exploit Overview



osTicket 1.11 - Cross-Site Scripting / Local File Inclusion

MALWARE

Created: 05/14/2019 First Seen: 04/25/2019 12:00am

Attributes (7)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE | SOURCES | DATE CREATED |
|---|---|---|---|
| Exploit Author | AkkuS | Exploit DB | 05/14/2019 02:06pm |
| Exploit DB Link | https://exploit-db.com/exploits/46753 | Exploit DB | 05/14/2019 02:06pm |
| Exploit DB ID | 46753 | Exploit DB | 05/14/2019 02:06pm |
| Exploit Verified | Yes | Exploit DB | 05/14/2019 02:06pm |
| Exploit Platform | PHP | Exploit DB | 05/14/2019 02:06pm |
| Exploit Type | WebApps | Exploit DB | 05/14/2019 02:06pm |
| Malware Type | Exploit | Exploit DB | 05/14/2019 02:06pm |

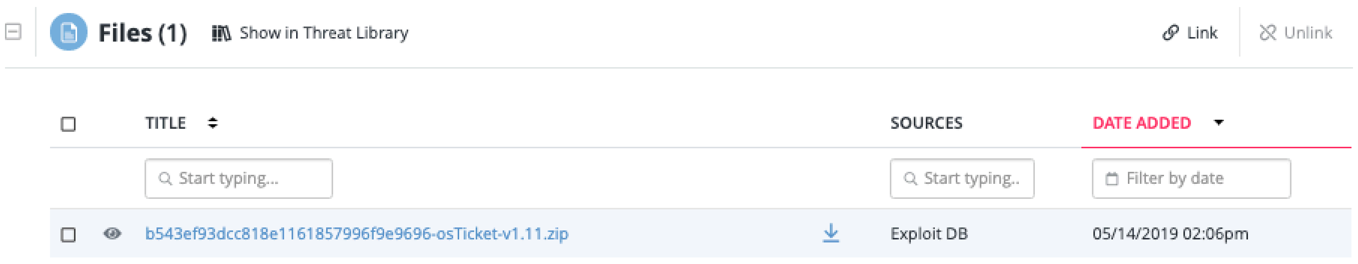
Sources (1)

Exploit DB

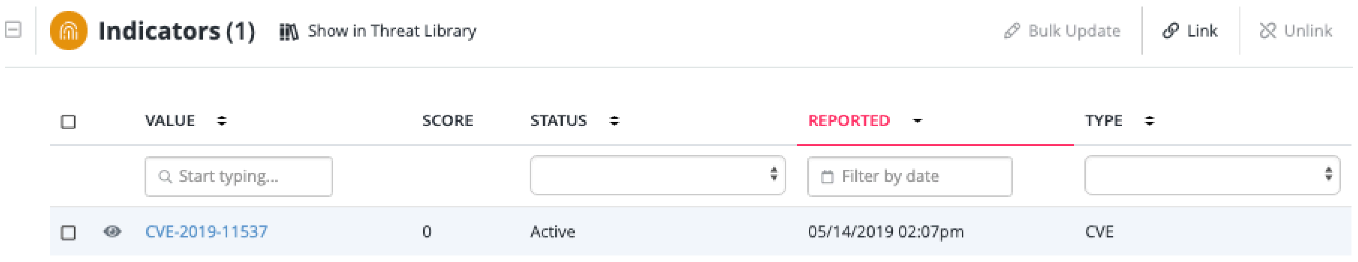
Tags (2)

Cross-Site Scripting (XSS)   File Inclusion (LFI/RFI)   Type here and press enter

## Related CVE and Vulnerable App

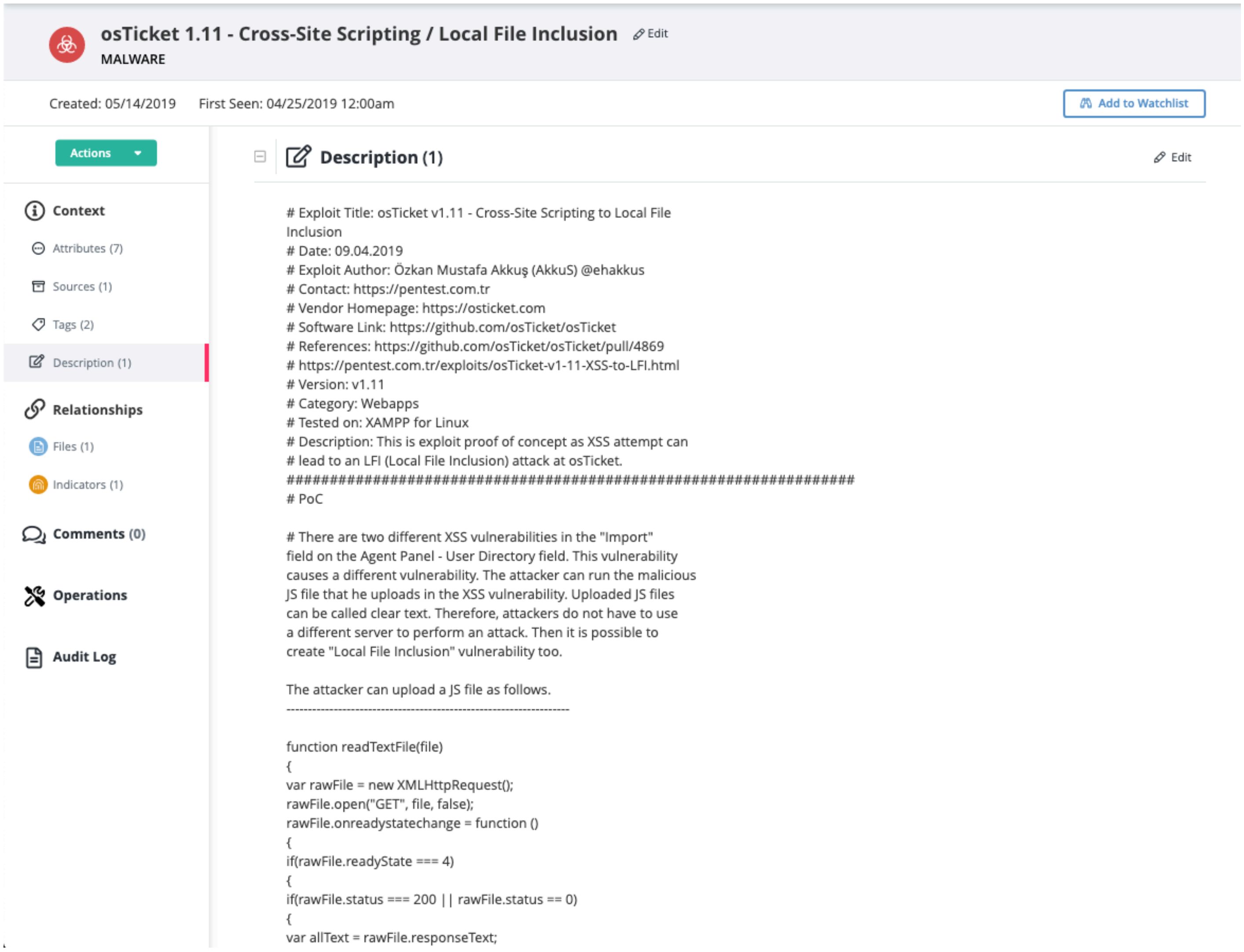


| | TITLE | SOURCES | DATE ADDED |
|---|---|---|---|
| | Start typing... | Start typing.. | Filter by date |
| | b543ef93dcc818e1161857996f9e9696-osTicket-v1.11.zip | Exploit DB | 05/14/2019 02:06pm |



| | VALUE | SCORE | STATUS | REPORTED | TYPE |
|---|---|---|---|---|---|
| | Start typing... | | | Filter by date | |
| | CVE-2019-11537 | 0 | Active | 05/14/2019 02:07pm | CVE |

## Exploit Description



```
# Exploit Title: osTicket v1.11 - Cross-Site Scripting to Local File
Inclusion
# Date: 09.04.2019
# Exploit Author: Özkan Mustafa Akkuş (AkkuS) @ehakkus
# Contact: https://pentest.com.tr
# Vendor Homepage: https://osticket.com
# Software Link: https://github.com/osTicket/osTicket
# References: https://github.com/osTicket/osTicket/pull/4869
# https://pentest.com.tr/exploits/osTicket-v1-11-XSS-to-LFI.html
# Version: v1.11
# Category: Webapps
# Tested on: XAMPP for Linux
# Description: This is exploit proof of concept as XSS attempt can
# lead to an LFI (Local File Inclusion) attack at osTicket.
#################################################################
# PoC

# There are two different XSS vulnerabilities in the "Import"
field on the Agent Panel - User Directory field. This vulnerability
causes a different vulnerability. The attacker can run the malicious
JS file that he uploads in the XSS vulnerability. Uploaded JS files
can be called clear text. Therefore, attackers do not have to use
a different server to perform an attack. Then it is possible to
create "Local File Inclusion" vulnerability too.

The attacker can upload a JS file as follows.
------------------------------------------------------------------

function readTextFile(file)
{
var rawFile = new XMLHttpRequest();
rawFile.open("GET", file, false);
rawFile.onreadystatechange = function ()
{
if(rawFile.readyState === 4)
{
if(rawFile.status === 200 || rawFile.status == 0)
{
var allText = rawFile.responseText;
```

# CRON

Automatic CRON configuration has been removed from this script. To run this script on a recurring basis, use CRON or some other jobs scheduler. The argument in the CRON script must specify the config and log locations.

Add an entry to your Linux crontab to execute the connector at a recurring interval. Depending on how quickly you need updates, this can be run multiple times a day (no more than once an hour) or a few times a week.

In the example below, the command will execute the connector every two hours.

1. Log into your ThreatQ host via a CLI terminal session.
2. Enter the following command:

   ```
   crontab -e
   ```

   This will enable the editing of the crontab, using vi. Depending on how often you wish the cronjob to run, you will need to adjust the time to suit the environment.

3. Enter the commands below:

   **Every 2 Hours Example**

   ```
   0 */2 * * * tq-conn-exploit-db -c /etc/tq_labs/ -ll /var/log/
   tq_labs/ -v3
   ```

4. Save and exit CRON.

# Change Log

- **Version 1.1.0**
    - Added Python 3 support.
- **Version 1.0.0**
    - Initial Release