Abstract

The Red Hat Security Data API exposes a list of endpoints to query security data with certain parameters and retrieve CVRF, CVE and OVAL data easily.
Table of Contents

CHAPTER 1. OVERVIEW ................................................................. 3

CHAPTER 2. CVRF ......................................................................... 4
  2.1. LIST ALL CVRFS ................................................................. 4
  2.2. PARAMETERS ................................................................... 4
  2.3. RETRIEVE A CVRF ............................................................. 5
  2.4. OVAL ................................................................................. 5

CHAPTER 3. CVE ........................................................................... 7
  3.1. LIST ALL CVEs ................................................................. 7
  3.2. PARAMETERS .................................................................. 7
  3.3. RETRIEVE A CVE ............................................................. 8
  3.4. CVE FORMAT .................................................................. 8

CHAPTER 4. OVALSTREAMS .............................................................. 11
  4.1. LIST ALL OVAL STREAMS ................................................. 11
  4.2. PARAMETERS ................................................................ 11
  4.3. RETRIEVE AN OVAL STREAM .......................................... 11

CHAPTER 5. EXAMPLE SCRIPT .......................................................... 13
Red Hat Product Security is committed to providing tools and security data to help you better understand security threats. This data has been available on our Security Data page and will now also be available in a machine-consumable format with the Security Data API. This tool will allow customers to programmatically query the API for data that was previously exposed only through files on our Security Data page.

The data provided by the Security Data API is the same as what is found on the Security Data page: OVAL definitions, Common Vulnerability Reporting Framework (CVRF) documents and CVE data. All data is available in its native XML format or in a representative JSON format.

This effort is a part of Red Hat Product Security's commitment to providing security data to customers in an easy-to-use format.

Please Note: Only one version will be maintained and any changes will be noted in the documentation.

The Security Data API is provided for information and metrics purposes. For any questions or concerns with the API or the data it provides, please contact Red Hat Product Security.

**Base URL**

https://access.redhat.com/hydra/rest/securitydata

**Supported Formats**

The API supports JSON, XML, and HTML formats. The format can be specified as an extension to the url like .json or .xml. If no format is specified, the default HTML format will be rendered.
CHAPTER 2. CVRF

2.1. LIST ALL CVRFS

Abstract

Provides an index to all recent CVRF documents with a summary of their contents, when no parameter is passed. Returns a convenience object as the response with minimal attributes.

NOTE

It does not return an index of published RHSAs as not all RHSA errata have a corresponding CVRF document.

JSON

GET /cvrf.json

XML

GET /cvrf.xml

HTML

GET /cvrf

2.2. PARAMETERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>Index of CVRF documents before the query date. [ISO 8601 is the expected format]</td>
<td>2016-03-01</td>
</tr>
<tr>
<td>after</td>
<td>Index of CVRF documents after the query date. [ISO 8601 is the expected format]</td>
<td>2016-02-01</td>
</tr>
<tr>
<td>rhsa_ids</td>
<td>Index of CVRF documents for RHSA_IDs separated by comma</td>
<td>RHSA-2018:2748,RHSA-2018:2791</td>
</tr>
<tr>
<td>bug</td>
<td>Index of CVRF documents for Bugzilla Ids</td>
<td>1326598,1084875</td>
</tr>
<tr>
<td>cve</td>
<td>Index of CVRF documents for CVEs</td>
<td>CVE-2014-0160,CVE-2016-3990</td>
</tr>
</tbody>
</table>
### Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>severity</td>
<td>Index of CVRF documents for severity</td>
<td>low, moderate, important, critical</td>
</tr>
<tr>
<td>package</td>
<td>Index of CVRF documents which affect package</td>
<td>samba, thunderbird</td>
</tr>
<tr>
<td>page</td>
<td>Index of CVRF documents for page number</td>
<td>Default: 1</td>
</tr>
<tr>
<td>per_page</td>
<td>Number of index of CVRF documents to return per page</td>
<td>Default: 1000</td>
</tr>
<tr>
<td>created_days_ago</td>
<td>Index of CVRF documents created days ago</td>
<td>10</td>
</tr>
</tbody>
</table>

By default, search will return the first page of 1000 results, ordered by date. To change the page size use the 'per_page' param, and then iterate through pages using the 'page' param.

**NOTE**

All the above query parameters can be used in combination with each other to retrieve the desired result.

### 2.3. RETRIEVE A CVRF

**Abstract**

CVRF details for the RHSA.

**JSON**

CVRF documents are in XML format, the JSON view is a representation of the CVRF data in JSON format.

```
GET /cvrf/<RHSA_ID>.json
```

**XML**

```
GET /cvrf/<RHSA_ID>.xml
```

**NOTE**

For more information about the CVRF format see the FAQ.

### 2.4. OVAL

**Abstract**
OVAL details for the RHSA.

**JSON**

OVAL documents are in XML format, the JSON view is a representation of the OVAL data in JSON format.

```plaintext
GET /cvrf/<RHSA_ID>/oval.json
```

**Example:** `/cvrf/RHSA-2016:0685/oval.json`

Returns a JSON representation of the OVAL data for RHSA-2016:0685.

**XML**

```plaintext
GET /cvrf/<RHSA_ID>/oval.xml
```

NOTE: This endpoint has been deprecated, please use OVAL streams instead of the individual files for the most complete product coverage. For more information about the OVAL format see the FAQ.
CHAPTER 3. CVE

3.1. LIST ALL CVES

Abstract
List all the recent CVEs when no parameter is passed. Returns a convenience object as response with very minimum attributes.

JSON
GET /cve.json

XML
GET /cve.xml

HTML
GET /cve

3.2. PARAMETERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>CVEs before the query date. [ISO 8601 is the expected format]</td>
<td>2016-03-01</td>
</tr>
<tr>
<td>after</td>
<td>CVEs after the query date. [ISO 8601 is the expected format]</td>
<td>2016-02-01</td>
</tr>
<tr>
<td>ids</td>
<td>CVEs for Ids separated by comma</td>
<td>CVE-2017-8797, CVE-2014-0161</td>
</tr>
<tr>
<td>bug</td>
<td>CVEs for Bugzilla Ids</td>
<td>1326598, 1084875</td>
</tr>
<tr>
<td>advisory</td>
<td>CVEs for advisory</td>
<td>RHSA-2016:0614, RHSA-2016:0610</td>
</tr>
<tr>
<td>severity</td>
<td>CVEs for severity</td>
<td>low, moderate, important</td>
</tr>
<tr>
<td>package</td>
<td>CVEs which affect the package</td>
<td>samba, thunderbird</td>
</tr>
<tr>
<td>product</td>
<td>CVEs which affect the product. The parameter supports Perl compatible regular expressions.</td>
<td>linux 7, openstack</td>
</tr>
<tr>
<td>cwe</td>
<td>CVEs with CWE</td>
<td>295, 300</td>
</tr>
</tbody>
</table>
By default, search will return the first page of 1000 results, ordered by date. To change the page size use the 'per_page' param, and then iterate through pages using the 'page' param.

NOTE
All the above query parameters can be used in combination with each other to retrieve the desired result.

3.3. RETRIEVE A CVE

Abstract
Retrieve full CVE details.

Path

GET /cve/<CVE>.json

Example: /cve/CVE-2016-3706.json

Returns a JSON representation of the CVE data for CVE-2016-3706.

3.4. CVE FORMAT

Abstract
Unlike CVRF or OVAL, the CVE representation is not a standard. Notes on what fields may exist and what they mean follow.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThreatSeverity</td>
<td>The Severity of the flaw.</td>
<td>See <a href="#">this document</a> for more information.</td>
</tr>
<tr>
<td>PublicDate</td>
<td>When the flaw became public.</td>
<td>ISO 8601 format.</td>
</tr>
<tr>
<td>Bugzilla</td>
<td>Id, URL, and Description of the bug in Red Hat’s Bugzilla.</td>
<td></td>
</tr>
<tr>
<td>CVSS</td>
<td>CVSSv2 score and metrics.</td>
<td>The ’status’ attribute may have a value of ’draft’ or ’verified’, indicating how far along the investigation of the flaw has progressed. See <a href="#">this document</a> for more information.</td>
</tr>
<tr>
<td>CVSS3</td>
<td>CVSSv3 score and metrics.</td>
<td>The ’status’ attribute may have a value of ’draft’ or ’verified’, indicating how far along the investigation of the flaw has progressed. See <a href="#">this document</a> for more information.</td>
</tr>
<tr>
<td>CWE</td>
<td>The CWE chain for this flaw.</td>
<td>See the <a href="#">mitre.org</a> description and our list of possible cwe values.</td>
</tr>
<tr>
<td>Details</td>
<td>Details about the flaw, possibly from Red Hat or Mitre.</td>
<td></td>
</tr>
<tr>
<td>Statement</td>
<td>A statement from Red Hat about the issue.</td>
<td></td>
</tr>
<tr>
<td>References</td>
<td>Links to more information about the issue.</td>
<td></td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>People or organizations that are being recognized.</td>
<td></td>
</tr>
<tr>
<td>Mitigation</td>
<td>A way to fix or reduce the problem without updated software.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Additional Information</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AffectedRelease</td>
<td>A released Erratum that fixes the flaw for a particular product.</td>
<td>Contains product name and CPE, and Erratum link, type, and release date. Optionally also includes &quot;Package&quot; information that describes the name and version of the src.rpm that fixes the issue (will not exist if multiple src.rpms are in the same Erratum).</td>
</tr>
<tr>
<td>PackageState</td>
<td>Information about a package / product where no fix has been released yet.</td>
<td>Contains product name and CPE, package (src.rpm) name, and fix state, which is one of ['Affected','Fix deferred','New','Not affected','Will not fix'].</td>
</tr>
<tr>
<td>UpstreamFix</td>
<td>The version of the upstream project that fixes the flaw.</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 4. OVALSTREAMS

4.1. LIST ALL OVAL STREAMS

Abstract
Provides an index to all OVAL stream files from where they can be downloaded. When no parameter is passed, returns a list of all OVAL stream files.

JSON
GET oval/ovalstreams.json

XML
GET oval/ovalstreams.xml

HTML
GET oval/ovalstreams

4.2. PARAMETERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>after</td>
<td>Index of OVAL stream files modified after the query date. Expected format: ISO 8601.</td>
<td>2016-02-01</td>
</tr>
<tr>
<td>label</td>
<td>Index of OVAL stream files for a product version label.</td>
<td>jboss-eap-6</td>
</tr>
</tbody>
</table>

By default, returned results are ordered by date.

NOTE
All the above query parameters can be used in combination with each other to retrieve the desired result.

4.3. RETRIEVE AN OVAL STREAM

Abstract
Returns the OVAL stream data for a product identified by base name.

JSON
OVAL stream files are in XML format; the JSON view is a representation of the OVAL data in JSON format.

GET oval/ovalstreams/<BASE>.json

Example: oval/ovalstreams/RHEL7.json

Returns a JSON representation of the OVAL streams for Red Hat Enterprise Linux 7.

XML

GET oval/ovalstreams/<BASE>.xml

NOTE

For more information about the OVAL format see the FAQ.
#!/usr/bin/env python

import sys
import requests
from datetime import datetime, timedelta

API_HOST = 'https://access.redhat.com/hydra/rest/securitydata'

def get_data(query):
    full_query = API_HOST + query
    r = requests.get(full_query)

    if r.status_code != 200:
        print('ERROR: Invalid request; returned {} for the following query:
              
              {}'.format(r.status_code, full_query))
        sys.exit(1)

    if not r.json():
        print('No data returned with the following query:
              
              {}'.format(full_query))
        sys.exit(0)

    return r.json()

# Get a list of issues and their impacts for RHSA-2022:1988
endpoint = '/cve.json'
params = 'advisory=RHSA-2022:1988'
data = get_data(endpoint + '?' + params)

for cve in data:
    print(cve['CVE'], cve['severity'])
    print('-----')

# Get a list of kernel advisories for the last 30 days and display the
# packages that they provided.
endpoint = '/cvrf.json'
date = datetime.now() - timedelta(days=30)
params = 'package=kernel&after=' + str(date.date())
data = get_data(endpoint + '?' + params)
kernel_advisories = []

for advisory in data:
    print(advisory['RHSA'], advisory['severity'], advisory['released_on'])
    print('-----', 'n- '.join(advisory['released_packages']))
    kernel_advisories.append(advisory['RHSA'])
print('-----
# From the list of advisories saved in the previous example (as
# `kernel_advisories`), get a list of affected products for each advisory.
endpoint = '/cvrf/

for advisory in kernel_advisories:
    data = get_data(endpoint + advisory + '.json')
    print(advisory)

    product_branch = data['cvrfdoc']['product_tree']['branch']
    for product_branch in data['cvrfdoc']['product_tree']['branch']:

        if product_branch['type'] == 'Product Family':

            if type(product_branch['branch']) is dict:
                print('-', product_branch['branch']['full_product_name']['product_name'])

            else:
                print('-', '
- '.join(pr['full_product_name']['product_name'] for pr in product_branch['branch']))