NVDLib

Release 0.5.2

Vehemont

Feb 15, 2022

CONTENTS

NVDLib is a Python API wrapper utilizing the REST API provided by NIST for the National Vulnerability Database (NVD).

Demo:

```
>>> import nvdlib
>>> r = nvdlib.getCVE('CVE-2021-26855')
>>> print(r.v3severity + ' - ' + str(r.v3score))
CRITICAL - 9.8
>>> print(r.cve.description.description_data[0].value)
Microsoft Exchange Server Remote Code Execution Vulnerability This CVE ID is unique from␣
↪CVE-2021-26412,
CVE-2021-26854, CVE-2021-26857, CVE-2021-26858, CVE-2021-27065, CVE-2021-27078.
>>> print(r.v3vector)
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
```

NVDLib is able to pull all data on known CVEs, search the NVD for CVEs or Common Platform Enumeration (CPE) names.

CHAPTER

ONE

FEATURES:

- Pull data on individual CVEs:
- CVE ID, description, reference links, CWE.
- CPE applicability statements and optional CPE names.
- CVSS severity scores.
- CVE publication date.
- CVE modified date.
- Search the NVD for CVEs by:
- Keywords
- Publish or modification start/end dates
- cweID
- CVSS V2/V3, score, severity, or metrics.
- CPE match string
- CPE name.
- Search the NVD for CPE names by:
- Modification start/End dates
- Keywords
- CPE match string.
- Dump data into objects to be accessible easily as class attributes.

For more information on the NIST NVD API for CPE and CVEs, see the documentation here: https://nvd.nist.gov/General/News/New-NVD-CVE-CPE-API-and-SOAP-Retirement

Note: NVDLib version 0.5.3 now includes NIST NVD rate limiting recommendations. Requests will now sleep for 6 seconds to complete with no API key. Requests with an API key will sleep 0.6 seconds. Get an API key here (free): https://nvd.nist.gov/developers/request-an-api-key

CHAPTER

TWO

NAVIGATION:

2.1 Getting started

2.1.1 Installation

To use NVDLib, first install it using pip:

```
(.venv) $ pip install nvdlib
```

This will also install the requests package if you do not already have it installed.

2.1.2 Importing NVDLib

Before you begin utilizing NVDLib make sure you import the nvdlib.py module:

```
import nvdlib
```

2.2 CVE

2.2.1 Single CVE

NVDLib allows you to grab data on a single CVE if the CVE ID is known. This is useful if you know the CVE but you need to know something about it such as the score, publish date, etc.

You can also use this to iterate through a list of CVE IDs if you have a list of known CVE IDs.

Begin by importing NVDLib:

```
>>> import nvdlib
```

Lets grab CVE-2017-0144.

```
>>> r = nvdlib.getCVE('CVE-2017-0144')
```

Example with an API key (insert your own API key).

```
>>> r = nvdlib.getCVE('CVE-2017-0144', key='xxxxxx-xxxxx-xxxx-xxxx-xxxxxxxxxx')
```

Note:

Due to rate limiting restrictions by NIST, a request will take 6 seconds with no API key.
Requests with an API key take 0.6 seconds per request.
Get a NIST NVD API key here (free): https://nvd.nist.gov/developers/request-an-api-key

From this point you are able to retrieve any information on the CVE. Here is a method to print the version 3 CVSS severity.

```
>>> print(r.v3severity)
HIGH
```

If you just need a score and severity from a CVE, you can use the *score* attribute that contains a list. This exists on all CVE objects and will prefer version 3 scoring. If version 3 scoring does not exist, it will use version 2. If no scoring exists for the CVE, it will set all values to *None*. The first element is the CVSS version, then score, and severity.

```
>>> print(r.score)
['V3', 8.8, 'HIGH']
```

nvdlib.cve.**getCVE**(*CVEID*, *cpe_dict=False*, *key=False*, *verbose=False*)
Build and send GET request for a single CVE then return object containing CVE attributes.

Parameters

- **CVEID** (*str*) – String of the CVE ID of the vulnerability to retrieve more details.
- **cpe_dict** (*Bool True*) – Set this value to true to control whether matching CPE names from the Official Dictionary are included in the response.
- **key** (*str*) – NVD API Key. Allows for a request every 0.6 seconds instead of 6 seconds.
- **verbose** (*bool*) – Prints the URL request for debugging purposes.

Below are all of the accessible variables within a CVE. Since these are assigned as is from the response of the API, I recommend printing some of the values to get an idea of what they will return. You can see what the JSON API response looks like and more details here https://nvd.nist.gov/developers/vulnerabilities

class nvdlib.classes.**CVE**(*dict*)
JSON dump class for CVEs

Variables

- **cve** (*dict*) – CVE ID, description, reference links, CWE.
- **configurations** (*dict*) – CPE applicability statements and optional CPE names.
- **impact** (*dict*) – CVSS severity scores
- **publishedDate** (*ISO 8601 date/time format including time zone.*) – CVE publication date
- **lastModifiedDate** (*ISO 8601 date/time format including time zone.*) – CVE modified date
- **id** (*str*) – CVE ID
- **cwe** (*str*) – Common Weakness Enumeration Specification (CWE)
- **url** (*str*) – Link to additional details on nvd.nist.gov for that CVE.

- **v3score** (*list*) – List that contains V3 or V2 CVSS score (float 1 - 10) as index 0 and the version that score was taken from as index 1.
- **v2vector** (*str*) – Version two of the CVSS score represented as a vector string, a compressed textual representation of the values used to derive the score.
- **v3vector** (*str*) – Version three of the CVSS score represented as a vector string.
- **v2severity** (*str*) – LOW, MEDIUM, HIGH (Critical is only available for v3).
- **v3severity** (*str*) – LOW, MEDIUM, HIGH, CRITICAL.
- **v2exploitability** (*float*) – Reflects the ease and technical means by which the vulnerability can be exploited.
- **v3exploitability** (*float*) – Reflects the ease and technical means by which the vulnerability can be exploited.
- **v2impactScore** (*float*) – Reflects the direct consequence of a successful exploit.
- **v3impactScore** (*float*) – Reflects the direct consequence of a successful exploit.
- **score** (*list*) – Contains the v3 CVSS score (v2 if v3 isn't available) [score, severity, version]. Where score is an int, severity is a string('LOW','MEDIUM','HIGH','CRITICAL'), and version is a string (V3 or V2).

2.2.2 Searching CVEs

Searching for CVEs will return a list containing the objects of all of the CVEs the search had found.

Example search for all vulnerabilities for Microsoft Exchange 2013, cumulative_update_11 and a limit of two:

```
>>> r = nvdlib.searchCVE(cpeName = 'cpe:2.3:a:microsoft:exchange_
↪server:2013:cumulative_update_11:*:*:*:*:*:*:*', limit = 2)
```

Now we have the results of the search in a list containing each CVE. Each CVE use the same schema as the CVEs retrieved as used in *getCVE*.

```
>>> type(r)
<class 'list'>
>>> for eachCVE in r:
... print(eachCVE.id)
CVE-1999-1322
CVE-2016-0032
```

nvdlib.cve.**searchCVE**(*keyword=False*, *pubStartDate=False*, *pubEndDate=False*, *modStartDate=False*, *modEndDate=False*, *includeMatchStringChange=False*, *exactMatch=False*, *cvssV2Severity=False*, *cvssV3Severity=False*, *cvssV2Metrics=False*, *cvssV3Metrics=False*, *cpeMatchString=False*, *cpeName=False*, *cpe_dict=False*, *cweId=False*, *limit=False*, *key=False*, *verbose=False*)

Build and send GET request then return list of objects containing a collection of CVEs.

Parameters

- **pubStartDate** (*ISO 8601 date/time*) – The pubStartDate and pubEndDate parameters specify the set of CVE that were added to NVD (published) during the period. Maximum 120 day range. It is not necessary to provide both start and end dates if your goal is to retrieve all CVE after a certain date, or up to a certain date. All times are in UTC 00:00. Example: '2020-06-28 00:00'

- **pubEndDate** (*ISO 8601 date/time*) – Publish end date. Can be used to get all vulnerabilities published up to a specific date and time. All times are in UTC 00:00. Example: '2020-06-28 00:00'
- **modStartDate** (*ISO 8601 date/time*) – The modStartDate and modEndDate parameters specify CVE that were subsequently modified. All times are in UTC 00:00. Example: '2020-06-28 00:00'
- **modEndDate** (*ISO 8601 date/time*) – Modifified end date. Can be used to get all vulnerabilities modfied up to a specific date and time. All times are in UTC 00:00.
- **includeMatchStringChange** (*bool True*) – Retrieve vulnerabilities where CPE names changed during the time period. This returns vulnerabilities where either the vulnerabilities or the associated product names were modified.
- **keyword** (*str*) – Word or phrase to search the vulnerability description or reference links.
- **exactMatch** (*bool True*) – If the keyword is a phrase, i.e., contains more than one term, then the isExactMatch parameter may be used to influence the response. Use exactMatch to retrieve records matching the exact phrase. Otherwise, the results contain any record having any of the terms.
- **cvssV2Severity** (*str*) – Find vulnerabilities having a 'LOW', 'MEDIUM', or 'HIGH' version 2 score.
- **cvssV3Severity** (*str*) – – Find vulnerabilities having a 'LOW', 'MEDIUM', 'HIGH', or 'CRITICAL' version 3 score.
- **cvssV3Metrics** (*cvssV2Metrics /*) – – If your application supports CVSS vector strings, use the cvssV2Metric or cvssV3Metrics parameter to find vulnerabilities having those score metrics. Partial vector strings are supported.
- **cpeMatchString** (*str*) – – Use cpeMatchString when you want a broader search against the applicability statements attached to the Vulnerabilities (e.x. find all vulnerabilities attached to a specific product).
- **cpeName** (*str*) – – Use cpeName when you know what CPE you want to compare against the applicability statements attached to the vulnerability (i.e. find the vulnerabilities attached to that CPE).
- **cpe_dict** (*bool True*) – – Set this value to true to control whether matching CPE from the Official Dictionary for each CVE are included in the response.

  Warning: If your search contains many results, the response will be very large as it will contain every CPE that a vulnerability has, thus resulting in delays.
- **limit** (*int*) – – Custom argument to limit the number of results of the search. Allowed any number between 1 and 2000.
- **key** (*str*) – NVD API Key. Allows for a request every 0.6 seconds instead of 6 seconds.
- **verbose** (*bool*) – Prints the URL request for debugging purposes.

2.2.3 SearchCVE Examples:

The arguments are not positional. SearchCVE will build the request based on what is passed to it. All of the parameters can be mixed together in any order. If a value is not passed to the function, it is assumed to be false and will not be added to the filter.

Filter by publication start and end date, keyword, version 3 severity of critical, and an API key to allow for faster requests:

Note: There is a maximum 120 day range when using date ranges.

```
>>> r = nvdlib.searchCVE(pubStartDate = '2021-09-08 00:00', pubEndDate = '2021-12-01␣
↪00:00', keyword = 'Microsoft Exchange', cvssV3Severity = 'Critical', key='xxxxx-xxxxxx-
↪xxxxxxx')
```

Filter for publications between 2019-06-02 and 2019-06-08:

```
>>> r = nvdlib.searchCVE(pubStartDate = '2019-06-08 00:00', pubEndDate = '2019-06-08␣
↪00:00')
```

Filter by CPE name and keyword with exact match enabled:

```
>>> r = nvdlib.searchCVE(cpeName = 'cpe:2.3:a:microsoft:exchange_server:2013:cumulative_
↪update_11:*:*:*:*:*:*:*', keyword = '1ArcServe', exactMatch = True)
```

Filter by CPE name, keyword, exact match enabled, and cpe_dict enabled:

```
>>> r = nvdlib.searchCVE(cpeName = 'cpe:2.3:a:microsoft:exchange_server:2013:cumulative_
↪update_11:*:*:*:*:*:*:*', keyword = '1ArcServe', exactMatch = True, cpe_dict = True)
```

Get the CVE IDs, score, and URL of a specific CPE name:

```
r = nvdlib.searchCVE(cpeName = 'cpe:2.3:a:microsoft:exchange_server:5.0:-:*:*:*:*:*:*')
for eachCVE in r:
   print(eachCVE.id, str(eachCVE.score[0]), eachCVE.url)
```

Grab the CPE names that match a CVE.

Note: CPE names will only be returned if 'cpe_dict = True' is passed to the search as a parameter.

```
r = nvdlib.searchCVE(cpeName = 'cpe:2.3:a:microsoft:exchange_server:2013:cumulative_
↪update_11:*:*:*:*:*:*:*', keyword = '1ArcServe', exactMatch = True, cpe_dict = True)
for eachCVE in r:
   config = eachCVE.configurations.nodes
   for eachNode in config:
      for eachCpe in eachNode.cpe_match:
            print(eachCpe.cpe23Uri)
```

2.3 CPE

2.3.1 Search CPE

Searching for CPEs is similar to searching for CVEs albeit less parameters. CPE match strings are allowed, meaning if partial strings are known, you can search for all possible CPE names. Like searching CVEs, the parameters are not positional.

Here is an example search with a keyword and a limit of 2 results then iterate through said CPE names.

Note:

Due to rate limiting restrictions by NIST, a request will take 6 seconds with no API key.
Requests with an API key take 0.6 seconds per request.
Get a NIST NVD API key here (free): https://nvd.nist.gov/developers/request-an-api-key

---

```
import nvdlib

r = nvdlib.searchCPE(keyword = 'Microsoft Exchange', limit = 2)
for eachCPE in r:
    print(eachCPE.name)


'cpe:2.3:a:microsoft:exchange_instant_messenger:-:*:*:*:*:*:*:*:*'
'cpe:2.3:a:microsoft:msn_messenger_service_for_exchange:4.5:*:*:*:*:*:*:*'
```

nvdlib.cpe.**searchCPE**(*modStartDate=False*, *modEndDate=False*, *includeDeprecated=False*, *keyword=False*, *cpeMatchString=False*, *cves=False*, *limit=False*, *key=False*, *verbose=False*)

Build and send GET request then return list of objects containing a collection of CPEs.

Parameters

- **modStartDate** (*ISO 8601 date/time Example: '2020-06-28 00:00' Maximum 120 day range*) – CPE modification start date
- **modEndDate** (*ISO 8601 date/time Example: '2020-06-28 00:00'*) – CPE modification end date
- **includeDeprecated** (*Bool True*) – Include deprecated CPE names that have been replaced.
- **keyword** (*str*) – Free text keyword search.
- **cpeMatchString** (*str*) – CPE match string search.
- **cves** (*bool True*) – Return vulnerabilities. **Warning**: This parameter may incur large amounts of results causing delays.
- **limit** (*int*) – Limits the number of results of the search.
- **key** (*str*) – NVD API Key. Allows for a request every 0.6 seconds instead of 6 seconds.
- **verbose** (*bool*) – Prints the URL request for debugging purposes.

class nvdlib.classes.**CPE**(*dict*)

JSON dump class for CPEs

Variables

- **name** (*str*) – CPE URI name
- **title** (*str*) – The first title result of the CPE.
- **deprecated** (*bool*) – Indicates whether CPE has been deprecated
- **cpe23Uri** (*str*) – The CPE name
- **lastModifiedDate** – CPE modification date
- **titles** (*dict*) – Human-readable CPE titles
- **refs** (*dict*) – Reference links.
- **deprecatedBy** – If deprecated=true, one or more CPE that replace this one
- **vulnerabilities** (*list*) – Optional vulnerabilities associated with this CPE. Must use 'cves = true' argument in searchCPE.

2.3.2 CPE Search Examples

Filter for a partial cpeMatchString for Microsoft Exchange 2013, return all the vulnerabilities for said matching CPEs, and print their CVE IDs.

Note: CVEs returned using searchCPE do *not* contain details and are only the ID.

```
r = nvdlib.searchCPE(cpeMatchString='cpe:2.3:a:microsoft:exchange_server:2013:',␣
↪cves=True, key='xxxxxx-xxxxx-xxxx-xxxx-xxxxxxxxxx')
for eachCPE in r:
    for eachVuln in eachCPE.vulnerabilities:
        print(eachVuln)
```

Filter for CPE names modfied between 2019-01-01 and 2021-01-01 with the keyword of PHP.

Note:

Maximum 120 days between a date.
len(r) will return how many CPE entries were found in the result.

```
r = nvdlib.searchCPE(modStartDate='2019-01-01 00:00', modEndDate='2020-01-01 00:00',␣
↪keyword='PHP')
print(len(r))

5992
```

2.4 Updates

Get updates on the GitHub repo.

INDEX