CVE Manager

by @AntoniosAtlasis

CVE Manager is a Python 3 script that was built to automatically download all NIST NVD (https://nvd.nist.gov/) database, parse them and export them in CSV (Comma Separated Values) files. This data, except from the CVEs and the related CVSS scores, also provide the CPEs (Common Platform Enumeration) and the CWE (Common Weakness Enumeration). However, due to the volume of the data and the requirement for searching (querying) this data, it also provides the capability to create a PostgreSQL (https://www.postgresql.org/) database, and import the NIST NVD data into it. Provided that the PostgreSQL database software is already installed in your system, the CVE Manager automatically creates the required schema and imports the downloaded data into it.

After importing all this data into PostgreSQL, the tool provides you the capability to query it by several useful criteria (e.g. CVE number, CVSS Score, publication date, CPE, CWE) and provides the results to standard output, or, when needed, to CSV files to.

The tool does not require root (admin) access; it requires Internet connection during the download part only. All the rest functionalities can be used in an air-gapped system (provided that Python3 and PostgreSQL are installed).

But let's start from the beginning.

A. Preparation

STEP 1: Downloading NIST NVD Data

The first step is always first to download the NIST NVD database, as follows:

$./cve_manager.py -d

If you want to simply extract them to CSV files, you run:

$./cve_manager.py -p -csv

(results will be stored in ./results folder; this cab be changed by using --*output* switch).

STEP 2: Prepare Postgresql database

Now, provided that you have already installed, initialised and have up and running Postgresql database software, let's create automatically the database that will be used to import and correlate our data, as follows:

$./cve_manager.py -u <postgres user with rights to create> -server <IP or hostname of the database server> -db <name of the database> -ow <postgres user that will be the owner of the database> -cd

And then, with the following command you can create the tables (and views) of the database:

$./cve_manager.py -u <username of database owner> -server <IP or hostname of the database server> -db <name of the database> -ct

STEP 3: Importing NIST NVD data into Postgresql database

Now let's import the downloaded NIST NVD into the database:

./cve_manager.py -u <username of database owner> -server <IP or hostname of the database server> -db <name of the database> -p -csv -idb

NOTE: The data must have been exported in CSV file before importing into db, or while importing them into your postgresql database.

In case you have not downloaded yet the NIST NVD database but you have already prepared the postgresql database as described above, you can simply combine downloading and importing in one command, as follows:

./cve_manager.py -u <username of database owner> -server <IP or hostname of the database server> -db <name of the database> -d -p -csv -idb

Truncating or Dropping NIST NVD imported data

If at some point you need to truncate the imported NIST NVD data (so as to import new ones), you can do it by running the following command:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -tr

You can even completely drop the database (for whatever reason) as follows:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -dd

STEP 4: Importing CWE

Then, to correlate CVEs with the MITRE CWE, you need to import (but only once, since not updated often) a CWE view. The one that is recommended is the 1000 (https://cwe.mitre.org/data/csv/1000.csv.zip – you need to unzip it before importing it), since it is intended to facilitate among others and the vulnerability analysts. You can import it as follows:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -icwe 1000.csv

Hint: You can combine the aforementioned commands in one or to lines to expedite the preparation.

B. Usage of (Querying) the Database

Now that you have imported all the required data, you are ready to query the database.

Querying the CVE database

a) If you want to query the database for a specific CVE number (e.g. CVE-2020-1234), you just run:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -cve <cve number, e.g. 2020-1234>

A sample output is depicted below:

```
CVE:                    CVE-2020-1234
CVSSv3.x Attack vector:  CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
CVSSv3.x Base Score:     7.8 HIGH
CVSSv2.x Attack vector:  AV:N/AC:M/Au:N/C:P/I:P/A:P
CVSSv2.x Base Score:     6.8 MEDIUM
Description:
An elevation of privilege vulnerability exists when Windows Error Reporting improperly handles objects in memory.To exploit this vul
nerability, an attacker would first have to gain execution on the victim system, aka 'Windows Error Reporting Elevation of Privilege
 Vulnerability'.

Pubished Date:          2020-06-09
Last Modified Date:     2020-06-15

Related Common Weakness Enumerations (CWE)
------------------------------------------
CWE-269 Improper Privilege Management

Related Common Platform Enumerations (CPE)
------------------------------------------
cpe:2.3:o:microsoft:windows_10:-:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_10:1607:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_10:1709:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_10:1803:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_10:1903:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_10:1909:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_10:2004:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_server_2016:-:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_server_2016:1803:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_server_2016:1903:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_server_2016:1909:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_server_2016:2004:*:*:*:*:*:*:*
cpe:2.3:o:microsoft:windows_server_2019:-:*:*:*:*:*:*:*
```

As you can see, except from the typical results (e.g. CVSS scores, attack vectors, description, etc.) you also get information about the related CWE(s) and the affected products (CPEs, with the limitations that they have in terms of precision).

b) Finding more information about the affected CWE

If you want to find more information about the related (for the selected CVE) CWE, you can use the following command:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -cwe <cwe number, e.g. 276>

A sample output is depicted below:

```
CWE-269
========
Improper Privilege Management
The software does not properly assign, modify, track, or check privileges for an actor, creating an unintended sphere of control for
that actor.

Modes of Introduction
--------------------
::PHASE:Architecture and Design::PHASE:Implementation:NOTE:REALIZATION: This weakness is caused during implementation of an architect
ural security tactic.::PHASE:Operation::

Common Consequences
--------------------
::SCOPE:Access Control:IMPACT:Gain Privileges or Assume Identity::

Potential Mitigations
--------------------
::PHASE:Architecture and Design Operation:DESCRIPTION:Very carefully manage the setting, management, and handling of privileges. Expl
icitly manage trust zones in the software.::PHASE:Architecture and Design:STRATEGY:Separation of Privilege:DESCRIPTION:Follow the pri
nciple of least privilege when assigning access rights to entities in a software system.::PHASE:Architecture and Design:STRATEGY:Sepa
ration of Privilege:DESCRIPTION:Consider following the principle of separation of privilege. Require multiple conditions to be met be
fore permitting access to a system resource.::
```

As you can see, this can provide some useful information like the modes of introduction (e.g. in the implementation phase, or architecture and design phase), the common consequences, or even (high level) potential mitigations.

c) If you want to have a list of all the CVEs with a CVSS score above a specific threshold, and/or after a specific initial publication date, you can run on the following:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -sc <threshold score> | more

The above command will return a list of all CVEs with any CVSS score equal to or greater than the specified threshold (which has to be between 0 and 10, including).

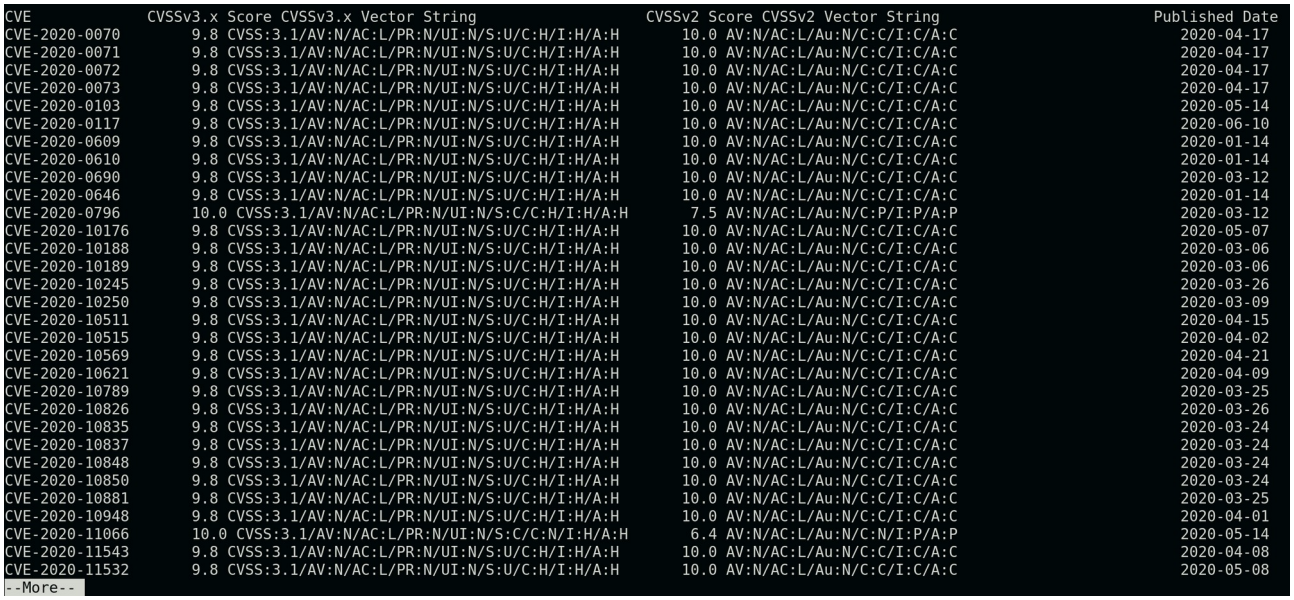
You can run a similar query but using as an additional criterion the publication date. For instance, the following command will list all the CVEs with a publication date after (and including) the specified one (for the selected CVSS score thereshold).

NOTE: The date must be in the format yyyy-mm-dd.

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -sc 0.1 -dt <date in the format yyyy-mm-dd> | more

NOTE: The CVSS score threshold has to be in the query. If you want to list all the CVEs after a date irrespective of the query, just use the value 0.1 for a score threshold.

A sample output of the above command is given below:



CVE	CVSSv3.x Score	CVSSv3.x Vector String	CVSSv2 Score	CVSSv2 Vector String	Published Date
CVE-2020-0070	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-17
CVE-2020-0071	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-17
CVE-2020-0072	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-17
CVE-2020-0073	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-17
CVE-2020-0103	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-05-14
CVE-2020-0117	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-06-10
CVE-2020-0609	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-01-14
CVE-2020-0610	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-01-14
CVE-2020-0690	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-12
CVE-2020-0646	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-01-14
CVE-2020-0796	10.0	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H	7.5	AV:N/AC:L/Au:N/C:P/I:P/A:P	2020-03-12
CVE-2020-10176	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-05-07
CVE-2020-10188	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-06
CVE-2020-10189	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-06
CVE-2020-10245	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-26
CVE-2020-10250	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-09
CVE-2020-10511	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-15
CVE-2020-10515	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-02
CVE-2020-10569	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-21
CVE-2020-10621	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-09
CVE-2020-10789	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-25
CVE-2020-10826	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-26
CVE-2020-10835	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-24
CVE-2020-10837	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-24
CVE-2020-10848	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-24
CVE-2020-10850	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-24
CVE-2020-10881	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-03-25
CVE-2020-10948	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-01
CVE-2020-11066	10.0	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:N/I:H/A:H	6.4	AV:N/AC:L/Au:N/C:N/I:P/A:P	2020-05-14
CVE-2020-11543	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-04-08
CVE-2020-11532	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	10.0	AV:N/AC:L/Au:N/C:C/I:C/A:C	2020-05-08

--More--

This type of lists can be very long, and some times convenient when you can extract them to CSV files (for further processing in other tools). You can do so by simply adding the -csv switch, i.e.:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -sc 0.1-dt <date in the format yyyy-mm-dd> -csv

The file is by default saved under the *results/* folder (you can change it by *-o* switch) with a filename related with the query itself, e.g. *CVEs_score9_2020-01-01.csv*

Querying the CPE database

You can make the aforementioned list of CVEs more specific, depending on the affected product. You can do so by the running the following command:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -sc <threshold score> -dt <date in the format yyyy-mm-dd> -cpe <cpe_to_query>

When querying for CPEs, it is usually useful to start with a generic query (e.g. windows), then make it more specific (e.g. microsoft:windows), then microsoft:windows_10 and finally (e.g. by judging from the results) to make it even more specific, e.g. microsoft:windows_10:1809

A simple output is provided below:

```
CPE                                                 CVE            CVSSv3.x CVSSv2  Published Date
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:x64:* CVE-2020-0662       8.8     9.0     2020-02-11
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0687        8.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0690        9.8       10.0    2020-03-12
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0734        8.8       9.3     2020-02-11
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0738        8.8       9.3     2020-02-11
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0883        8.8       9.3     2020-03-12
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0807        8.8       9.3     2020-03-12
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0809        8.8       9.3     2020-03-12
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0881        8.8       9.3     2020-03-12
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0948        8.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0889        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0907        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0949        8.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0959        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0950        8.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0953        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0988        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0960        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0964        8.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0992        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0994        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-1008        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0995        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-0999        7.8       9.3     2020-04-15
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-1028        7.8       9.3     2020-05-21
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-1061        8.8       9.3     2020-05-21
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-1051        7.8       9.3     2020-05-21
cpe:2.3:o:microsoft:windows_10:1809:*:*:*:*:*:*:* CVE-2020-1067        8.8       9.0     2020-05-21
```

As before, you can export them in a CSV file by adding the *-csv* switch, i.e.:

./cve_manager.py -u <username> -server <IP or hostname of dababase server> -db <name of database> -sc <threshold score> -dt <date in the format yyyy-mm-dd> -cpe <cpe_to_query> -csv

Again, the filename is saved after the query, e.g.:

microsoft:windows_10:1809_9_2020-01-01.csv

Enjoy ;-)