LimberDuck

Release 0.0.1

Damian Krawczyk

Apr 16, 2024

CONTENTS

LimberDuck (pronounced lm.b dk) is a project initiated on November 26, 2018. The main goal of this project is to create an array of free and Open Source[1] tools dedicated for Security Engineers who wants to automate their work, decrease their workload and focus on data analysis.

nessus file analyzer

This is a GUI (Graphical User Interface) tool that allows you to analyze nessus files containing the results of scans performed by using *Nessus* or *Tenable.sc* by © Tenable, Inc. used for VA (Vulnerability Assessment)[2] process.

read more

nessus file reader

This is a CLI (command-line interfaces) tool and python module that allows you to quickly parse nessus files containing the results of scans performed by using *Nessus* or *Tenable.sc* by © Tenable, Inc. used for VA[2] process.

read more

converter csv

This is a GUI tool that allows you to convert multiple large CSV (comma-separated value) files to XLSX (Microsoft Excel Open XML Spreadsheet) files keeping your operational memory usage at a low level.

read more

---

[1] read more about *Open Source* in glossary
[2] read more about *Vulnerability Assessment* in glossary

testimonials

nessus file analyzer:

> I'm grateful for your software…
>
> —User

> I love the Nessus File Analyzer, so thank you so much for sharing and maintaining.
>
> —User

> Tested everyday. Works perfect.
>
> —User

> Brilliant work!
>
> —User

> This tool is really helpful! Thanks for sharing this.
>
> —User

> I found nessus file analyzer to be an excellent tool.
>
> —User

> First of all… Great tool! You did a really great job! Thanks for developing such a wonderful tool!
>
> —User

CHAPTER

ONE

TOOLS

1.1 nessus file analyzer

This is a GUI tool which enables you to parse multiple nessus files containing the results of scans performed by using *Nessus* or *Tenable.sc* by © Tenable, Inc. used for VA[1] process. Parsed scan results are exported to a Microsoft Excel Workbook for effortless analysis.

Operational memory usage will be kept low while parsing even the largest of files. You can run it on your favorite operating system, whether it is Windows, macOS or GNU Linux. As a parsing result, you will receive spreadsheets with a summary view of the whole scan and/or all reported hosts. You will also be able to generate spreadsheets with a detailed view of all reported vulnerabilities[2] and/or noncompliance. It's free and Open Source[3] tool.

source code

release notes

discussions

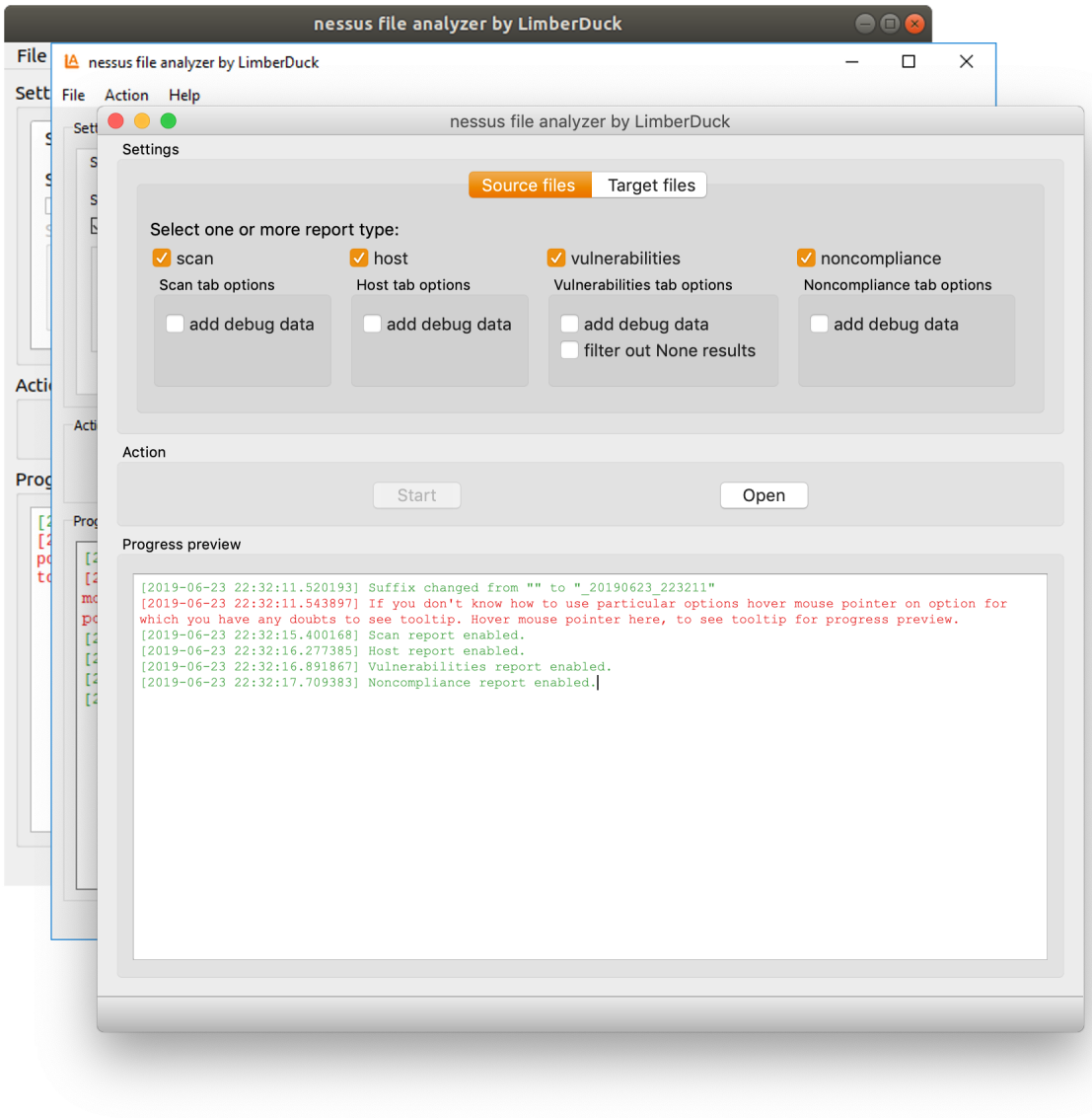
issues

docs

[1] read more about *Vulnerability Assessment* in glossary
[2] read more about *vulnerability* in glossary
[3] read more about *Open Source* in glossary

Listing 1: Install nessus-file-analyzer

```
pip install nessus-file-analyzer
```



1.1.1 technology stack

1.1.2 testimonials

I'm grateful for your software…

—User

I love the Nessus File Analyzer, so thank you so much for sharing and maintaining.

—User

Tested everyday. Works perfect.

—User

Brilliant work!

—User

This tool is really helpful! Thanks for sharing this.

—User

I found nessus file analyzer to be an excellent tool.

—User

First of all… Great tool! You did a really great job! Thanks for developing such a wonderful tool!

—User

1.1.3 stargazers over time

1.2 nessus file reader

This is a CLI tool and python module which enables you to quickly parse nessus files containing the results of scans performed by using *Nessus* or *Tenable.sc* by © Tenable, Inc. used for VA[1] process. This module will let you get data

[1] read more about *Vulnerability Assessment* in glossary

through functions grouped into categories like file, scan, host, and plugin to get specific information from the provided nessus scan files e.g. file size, report name, report hosts names, the number of target hosts, the number of hosts scanned with credentialed checks, the number of reported plugins per Risk Factor, exact host scan times, outputs of particular plugins and a lot more. It's free and Open Source[2] tool.

source code

release notes

discussions

issues

docs

Listing 2: Install nessus-file-reader

```
pip install nessus-file-reader
```

1.2.1 technology stack

1.2.2 stargazers over time

1.3 converter csv

This is a GUI tool which lets you convert multiple large CSV files to XLSX files keeping your operational memory usage at a low level. You can run it on your operating system no matter if it is Windows, MacOS or GNU Linux. It's free and Open Source[1] tool.

source code

[2] read more about *Open Source* in glossary
[1] read more about *Open Source* in glossary

release notes

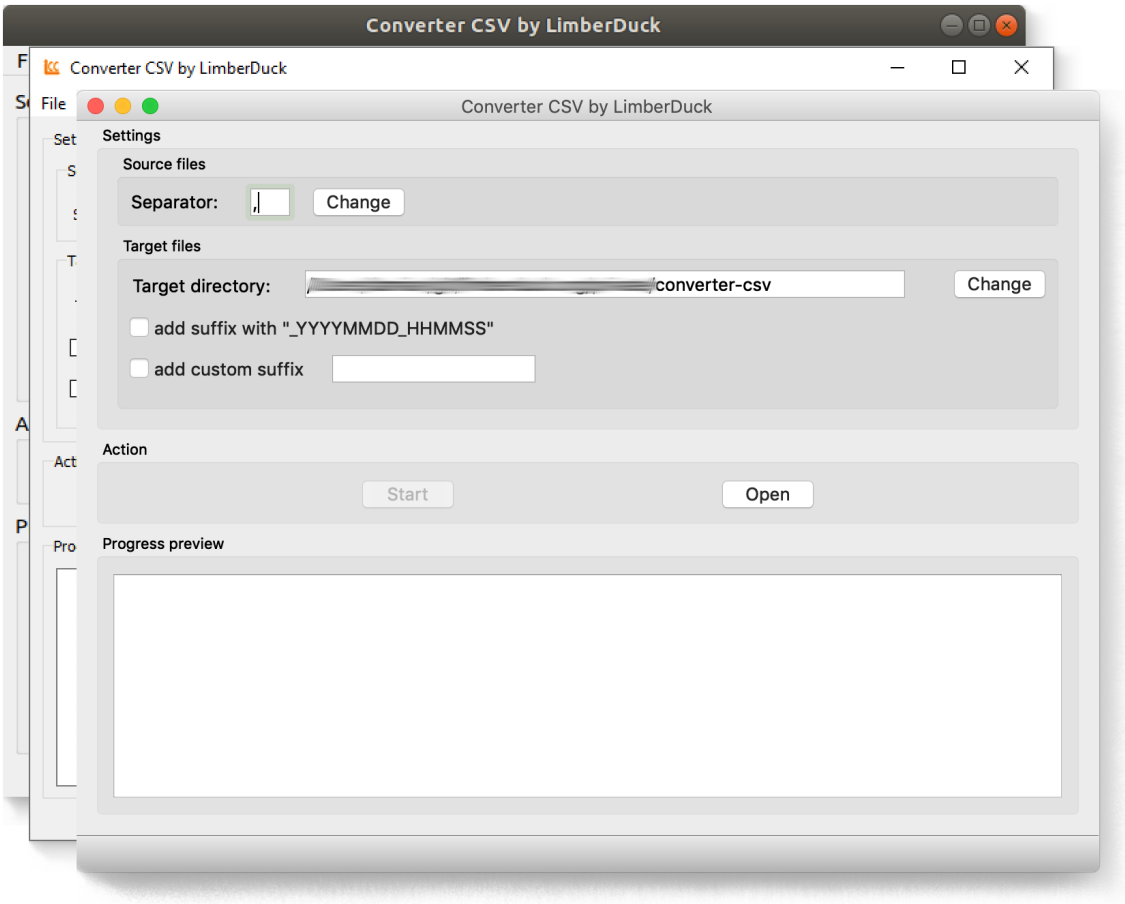
discussions

issues

docs

Listing 3: Install converter-csv

```
pip install converter-csv
```



1.3.1 technology stack


Qt

CHAPTER

TWO

NOTEBOOKS

2.1 CPE

Common Platform Enumeration (CPE) is a structured naming scheme for information technology systems, software, and packages. Based upon the generic syntax for Uniform Resource Identifiers (URI), CPE includes a formal name format, a method for checking names against a system, and a description format for binding text and tests to a name. This method of naming is known as a well-formed CPE name (WFN)

source: cpe.mitre.org/specification

You can see this notebook directly via:

- GitHub
- Jupter nbviewer

2.1.1 Generation time

```
from datetime import datetime, timezone, timedelta

timezone_offset = 0.0
tzinfo = timezone(timedelta(hours=timezone_offset))
generation_time = datetime.now(tzinfo).strftime('%Y-%m-%d %H:%M:%S %z')
print(generation_time)
```

```
2024-04-16 06:01:44 +0000
```

2.1.2 Creative Commons

This notebook and generated diagrams are released with Creative Commons liecense (CC BY 4.0).

```
import requests
import urllib3

urllib3.disable_warnings()

urls = ['https://mirrors.creativecommons.org/presskit/icons/cc.xlarge.png',
        'https://mirrors.creativecommons.org/presskit/icons/by.xlarge.png']
```

(continues on next page)

(continued from previous page)

```
for url in urls:
    file_name = url.split("/")[-1:][0]
    print(file_name)

    file = requests.get(url, verify=False)
    open(file_name, 'wb').write(file.content)
```

```
cc.xlarge.png
by.xlarge.png
```

2.1.3 CPE data downloading

All CPE stats are taken from nvd.nist.gov/products/cpe/statistics

```
from urllib.request import urlopen
import ssl
from bs4 import BeautifulSoup, SoupStrainer

def get_data(url):

    ctx = ssl.create_default_context()
    ctx.check_hostname = False
    ctx.verify_mode = ssl.CERT_NONE

    page = urlopen(url, context=ctx)
    html = page.read().decode("utf-8")

    product = SoupStrainer('table')
    soup = BeautifulSoup(html, "html.parser", parse_only=product)

    return soup

url = "https://nvd.nist.gov/products/cpe/statistics"
data = get_data(url)

print(len(data))
```

```
16
```

2.1.4 New CPE entries

CPE data parsing

```
import pandas as pd

def pars(data):
    data_table = []
```

(continues on next page)

(continued from previous page)

```
    for table in data:
        table_id = table['id']
        table_year = table_id[-4:]
        table_rows = table.find_all("tr")
        number_of_new_cpe_entries_yearly = 0
        number_of_new_cpe_entries_list = []
        data_row = []
        for table_row in table_rows:
            data = table_row.find_all("td")

            if data:
                number_of_new_cpe_entries = int(data[1].string.replace(",",""))

                number_of_new_cpe_entries_list.append(number_of_new_cpe_entries)

                number_of_new_cpe_entries_yearly += number_of_new_cpe_entries

        while len(number_of_new_cpe_entries_list) < 12:
            number_of_new_cpe_entries_list.append(0)

        data_row.append(table_year)
        data_row.append(number_of_new_cpe_entries_yearly)
        data_row = data_row + number_of_new_cpe_entries_list

        data_table.append(data_row)

    data_columns = ['Year', 'Summary', 'January', 'February', 'March', 'April', 'May',
↪'June', 'July', 'August', 'September', 'October', 'November', 'December']
    df = pd.DataFrame (data_table, columns = data_columns)
    df.sort_values(by=['Year'], inplace=True)
    df.reset_index(drop=True, inplace=True)
    df.index += 1
    return df

parsed_data = pars(data)

parsed_data.style.bar(subset=['Summary'], color='#FF6200')
```

```
Matplotlib is building the font cache; this may take a moment.
```

```
<pandas.io.formats.style.Styler at 0x7ff978208790>
```

CPE data saving

CSV file is available in GitHub repository, see:

- file via GitHub
- file directly

```
csv_filename = 'cpe-number-of-new-entries.csv'

parsed_data.to_csv(csv_filename, index=False)
```

CPE data ploting

PNG files are available in GitHub repository with two background versions, see:

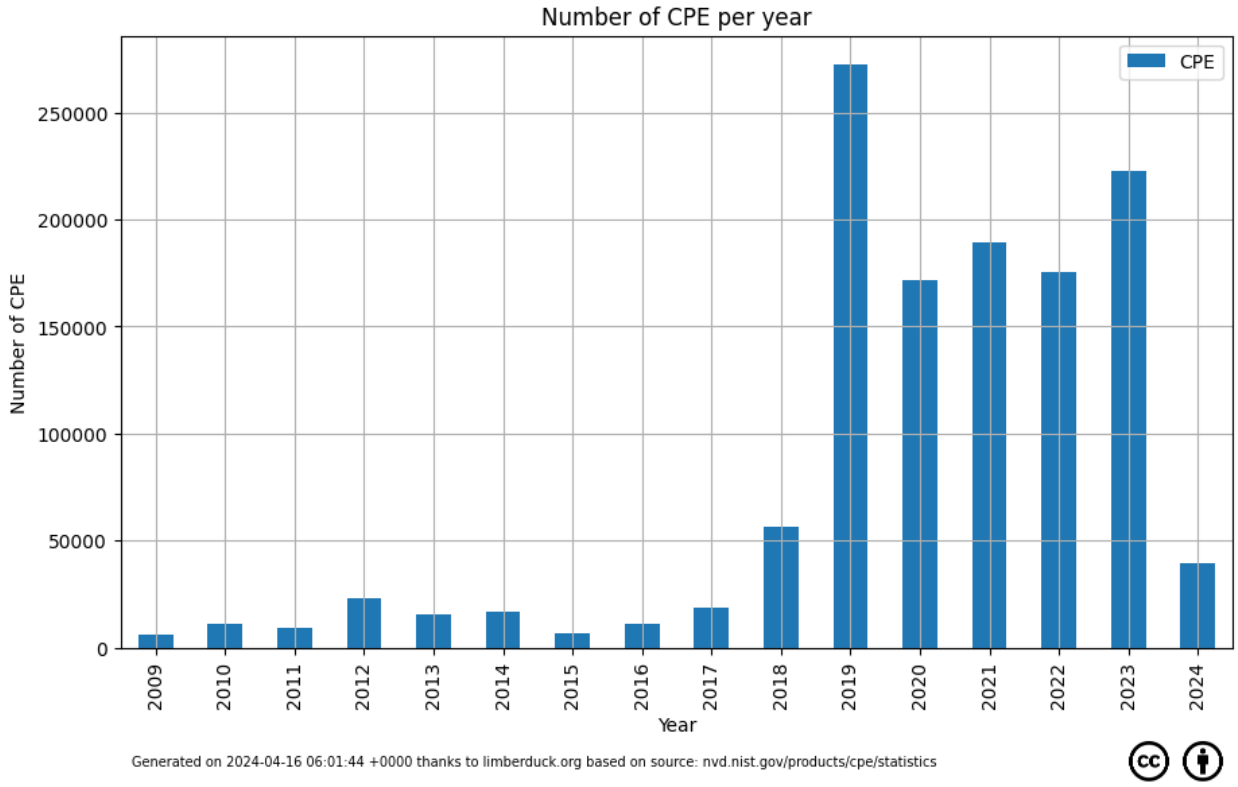
- file via GitHub (white background)
- file via GitHub (transparent background)
- file directly (white background)
- file directly (transparent background)

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

import urllib

df = pd.read_csv(csv_filename)

df.plot(x='Year',
        xlabel='Year',
        y='Summary',
        ylabel='Number of CPE',
        kind='bar',
        title='Number of CPE per year')
plt.tight_layout()
plt.legend(['CPE'])
plt.figtext(0.16, 0.02, f"Generated on {generation_time} thanks to limberduck.org based␣
↪on source: nvd.nist.gov/products/cpe/statistics", ha="left", fontsize=7)
fig = plt.gcf()
fig.set_size_inches(10,6)
fig.patch.set_facecolor('white')
plt.grid(True)

img_cc = plt.imread('cc.xlarge.png')
newax_cc = fig.add_axes([0.88, 0.0, 0.05, 0.05], anchor='NE', zorder=-1)
newax_cc.imshow(img_cc)
newax_cc.axis('off')
img_by = plt.imread('by.xlarge.png')
newax_by = fig.add_axes([0.92, 0.0, 0.05, 0.05], anchor='NE', zorder=-1)
newax_by.imshow(img_by)
newax_by.axis('off')
```

(continues on next page)

(continued from previous page)

```
plt.savefig('cpe-number-of-new-entries-bg-white.png', dpi = 300, facecolor = 'white')
plt.savefig('cpe-number-of-new-entries-bg-transparent.png', dpi = 300, transparent =␣
↪True)
```



2.2 CVE

Common Vulnerabilities and Exposures Identifier (CVE ID) is a unique, alphanumeric identifier assigned by the CVE Program. Each identifier references a specific vulnerability. A CVE ID enables automation and multiple parties to discuss, share, and correlate information about a specific vulnerability, knowing they are referring to the same thing

source: www.cve.org

You can see this notebook directly via:

- GitHub
- Jupter nbviewer

2.2.1 Generation time

```
from datetime import datetime, timezone, timedelta

timezone_offset = 0.0
tzinfo = timezone(timedelta(hours=timezone_offset))
generation_time = datetime.now(tzinfo).strftime('%Y-%m-%d %H:%M:%S %z')
print(generation_time)
```

```
2024-04-16 06:02:08 +0000
```

2.2.2 Creative Commons

This notebook and generated diagrams are released with Creative Commons liecense (CC BY 4.0).

```
import requests
import urllib3

urllib3.disable_warnings()

urls = ['https://mirrors.creativecommons.org/presskit/icons/cc.xlarge.png',
        'https://mirrors.creativecommons.org/presskit/icons/by.xlarge.png']
for url in urls:
    file_name = url.split("/")[-1:][0]
    print(file_name)

    file = requests.get(url, verify=False)
    open(file_name, 'wb').write(file.content)
```

```
cc.xlarge.png
by.xlarge.png
```

2.2.3 CVE data downloading

All CVE IDs are taken from cve.mitre.org/data/downloads/index.html

```
url = 'https://cve.mitre.org/data/downloads/allitems.xml.Z'
file_name = url.split("/")[-1:][0]
print(file_name)
```

```
allitems.xml.Z
```

```
import requests
import urllib3

urllib3.disable_warnings()
```

(continues on next page)

(continued from previous page)

```
file = requests.get(url, verify=False)
open(file_name, 'wb').write(file.content)
```

```
69886215
```

```
import unlzw3
from pathlib import Path

uncompressed_data = unlzw3.unlzw(Path(file_name))
```

```
with open(file_name[:-2], 'wb') as file:
    file.write(uncompressed_data)
```

```
import glob

file_name = glob.glob('*.xml')[-1]
print(file_name)
```

```
allitems.xml
```

2.2.4 CVE data parsing

```
import pandas as pd
import xml.etree.ElementTree as et

tree = et.parse(file_name)
root = tree.getroot()
df_cols = ["number", "year"]
rows = []

for item in root:
    item_name = item.attrib.get("name")
    item_year = item_name[4:8]
    rows.append({"number": item_name, "year": item_year})

df = pd.DataFrame(rows, columns = df_cols)

print(df)
```

```
                number  year
0        CVE-1999-0001  1999
1        CVE-1999-0002  1999
2        CVE-1999-0003  1999
3        CVE-1999-0004  1999
4        CVE-1999-0005  1999
...                ...   ...
311256  CVE-2024-30266  2024
311257  CVE-2024-30267  2024
311258  CVE-2024-30268  2024
```

(continues on next page)

(continued from previous page)

```
311259  CVE-2024-30269  2024
311260  CVE-2024-30270  2024

[311261 rows x 2 columns]
```

```
df = df.groupby(['year'], as_index=False)[['number']].count()
df.reset_index(drop=True, inplace=True)
df.index += 1

df.style.bar(subset=['number'], color='#FF6200')
```

```
<pandas.io.formats.style.Styler at 0x7fe07d71abd0>
```

2.2.5 CVE data saving

CSV file is available in GitHub repository, see:

- file via GitHub
- file directly

```
csv_filename = 'cve-number-of-entries.csv'

df.to_csv(csv_filename, index=False)
```

2.2.6 CVE data ploting

PNG files are available in GitHub repository with two background versions, see:

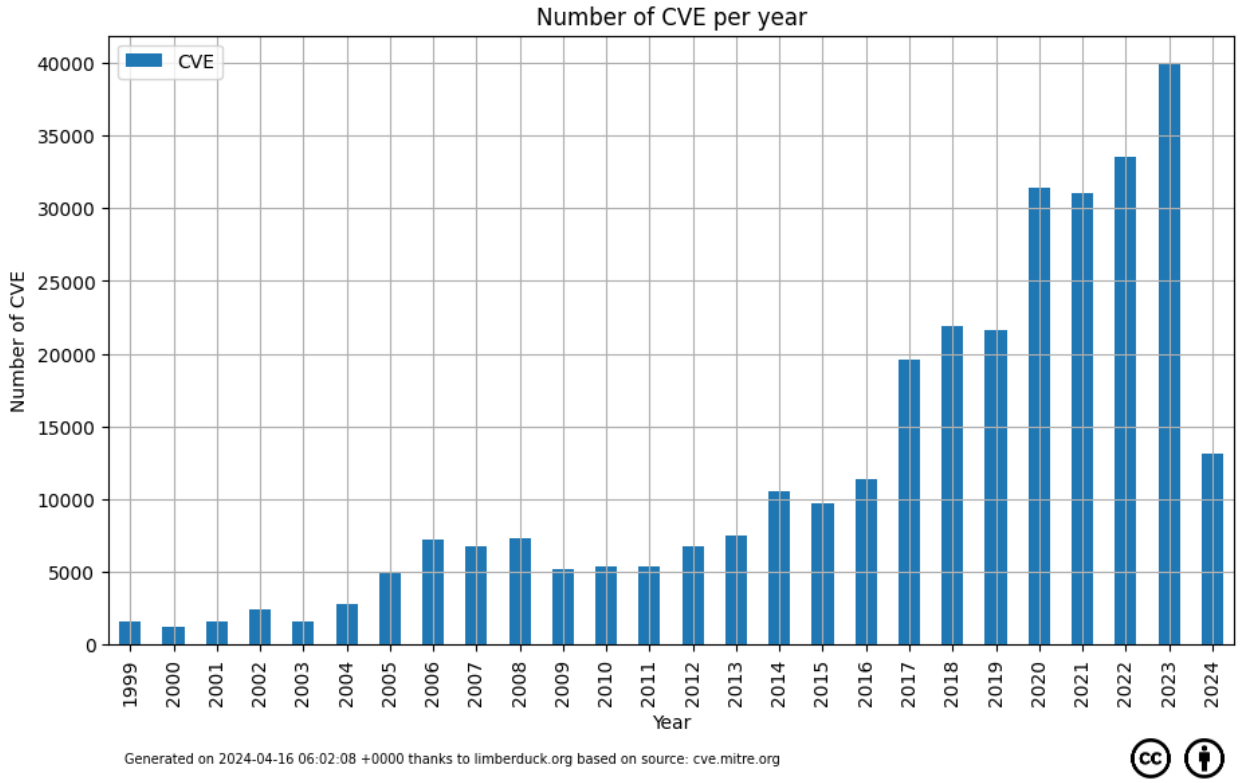
- file via GitHub (white background)
- file via GitHub (transparent background)
- file directly (white background)
- file directly (transparent background)

```
import pandas as pd
import matplotlib.pyplot as plt
import datetime

df = pd.read_csv(csv_filename)

df.plot(x='year',
        xlabel='Year',
        y='number',
        ylabel='Number of CVE',
        kind='bar',
        title='Number of CVE per year')
plt.tight_layout()
plt.legend(['CVE'])
plt.figtext(0.15, 0.02, f"Generated on {generation_time} thanks to limberduck.org based␣
```

(continues on next page)

(continued from previous page)

```
↪on source: cve.mitre.org", ha="left", fontsize=7)
fig = plt.gcf()
fig.set_size_inches(10,6)
fig.patch.set_facecolor('white')
plt.grid(True)

img_cc = plt.imread('cc.xlarge.png')
newax_cc = fig.add_axes([0.88, 0.0, 0.05, 0.05], anchor='NE', zorder=-1)
newax_cc.imshow(img_cc)
newax_cc.axis('off')
img_by = plt.imread('by.xlarge.png')
newax_by = fig.add_axes([0.92, 0.0, 0.05, 0.05], anchor='NE', zorder=-1)
newax_by.imshow(img_by)
newax_by.axis('off')

plt.savefig('cve-number-of-entries-bg-white.png', dpi = 300, facecolor = 'white')
plt.savefig('cve-number-of-entries-bg-transparent.png', dpi = 300, transparent = True)
```



2.3 CWE

Common Weakness Enumeration (CWE™) is a formal list or dictionary of common software and hardware weaknesses that can occur in architecture, design, code, or implementation that can lead to exploitable security vulnerabilities. CWE was created to serve as a common language for describing security weaknesses; serve as a standard measuring stick for security tools targeting these weaknesses; and to provide a common baseline standard for weakness identification, mitigation, and prevention efforts. "Weaknesses" are flaws, faults, bugs, and other errors in software and hardware design, architecture, code, or implementation that if left unaddressed could result in systems and networks, and hardware being vulnerable to attack

> source: cwe.mitre.org

You can see this notebook directly via:

- GitHub
- Jupter nbviewer

2.3.1 Generation time

```
from datetime import datetime, timezone, timedelta

timezone_offset = 0.0
tzinfo = timezone(timedelta(hours=timezone_offset))
generation_time = datetime.now(tzinfo).strftime('%Y-%m-%d %H:%M:%S %z')
print(generation_time)
```

```
2024-04-16 06:03:12 +0000
```

2.3.2 Creative Commons

This notebook and generated diagrams are released with Creative Commons liecense (CC BY 4.0).

```
import requests
import urllib3

urllib3.disable_warnings()

urls = ['https://mirrors.creativecommons.org/presskit/icons/cc.xlarge.png',
        'https://mirrors.creativecommons.org/presskit/icons/by.xlarge.png']
for url in urls:
    file_name = url.split("/")[-1:][0]
    print(file_name)

    file = requests.get(url, verify=False)
    open(file_name, 'wb').write(file.content)
```

```
cc.xlarge.png
by.xlarge.png
```

2.3.3 CWE data downloading

All CWE IDs are taken from cwe.mitre.org/data/downloads.html

```
url = 'https://cwe.mitre.org/data/xml/cwec_latest.xml.zip'
file_name = url.split("/")[-1:][0]
print(file_name)
```

```
cwec_latest.xml.zip
```

```
import requests
import urllib3

urllib3.disable_warnings()

file = requests.get(url, verify=False)
open(file_name, 'wb').write(file.content)
```

```
1720673
```

```
import zipfile

with zipfile.ZipFile(file_name, 'r') as zip_ref:
    zip_ref.extractall()
```

```
import glob

file_name = glob.glob('*.xml')[-1]
print(file_name)
```

```
cwec_v4.14.xml
```

2.3.4 CWE data parsing

Updated to pars `cwec_v4.14.xml`.

```
import pandas as pd
import xml.etree.ElementTree as et

tree = et.parse(file_name)
root = tree.getroot()
df_cols = ["number", "year"]
rows = []

if root.findall('{http://cwe.mitre.org/cwe-7}Weaknesses'):
    weeknesses = root.find('{http://cwe.mitre.org/cwe-7}Weaknesses')
    for weekness in weeknesses:
        weekness_id = weekness.get("ID")
        weekness_content_history = weekness.find("{http://cwe.mitre.org/cwe-7}Content_
↪History")
```

(continues on next page)

(continued from previous page)

```
        weekness_content_submission = weekness_content_history.find("{http://cwe.mitre.
↪org/cwe-7}Submission")
        weekness_content_submission_date = weekness_content_submission.find("{http://cwe.
↪mitre.org/cwe-7}Submission_Date").text
        weekness_content_submission_year = weekness_content_submission_date[0:4]

        rows.append({"number": weekness_id, "year": weekness_content_submission_year})

df = pd.DataFrame(rows, columns = df_cols)

print(df)
```

```
    number  year
0     1004  2017
1     1007  2017
2      102  2006
3     1021  2017
4     1022  2017
..     ...   ...
958     95  2006
959     96  2006
960     97  2006
961     98  2006
962     99  2006

[963 rows x 2 columns]
```

```
df = df.groupby(['year'], as_index=False)[['number']].count()
df.reset_index(drop=True, inplace=True)
df.index += 1

df.style.bar(subset=['number'], color='#FF6200')
```

```
<pandas.io.formats.style.Styler at 0x7f544f3897d0>
```

2.3.5 CWE data saving

CSV file is available in GitHub repository, see:

- file via GitHub
- file directly

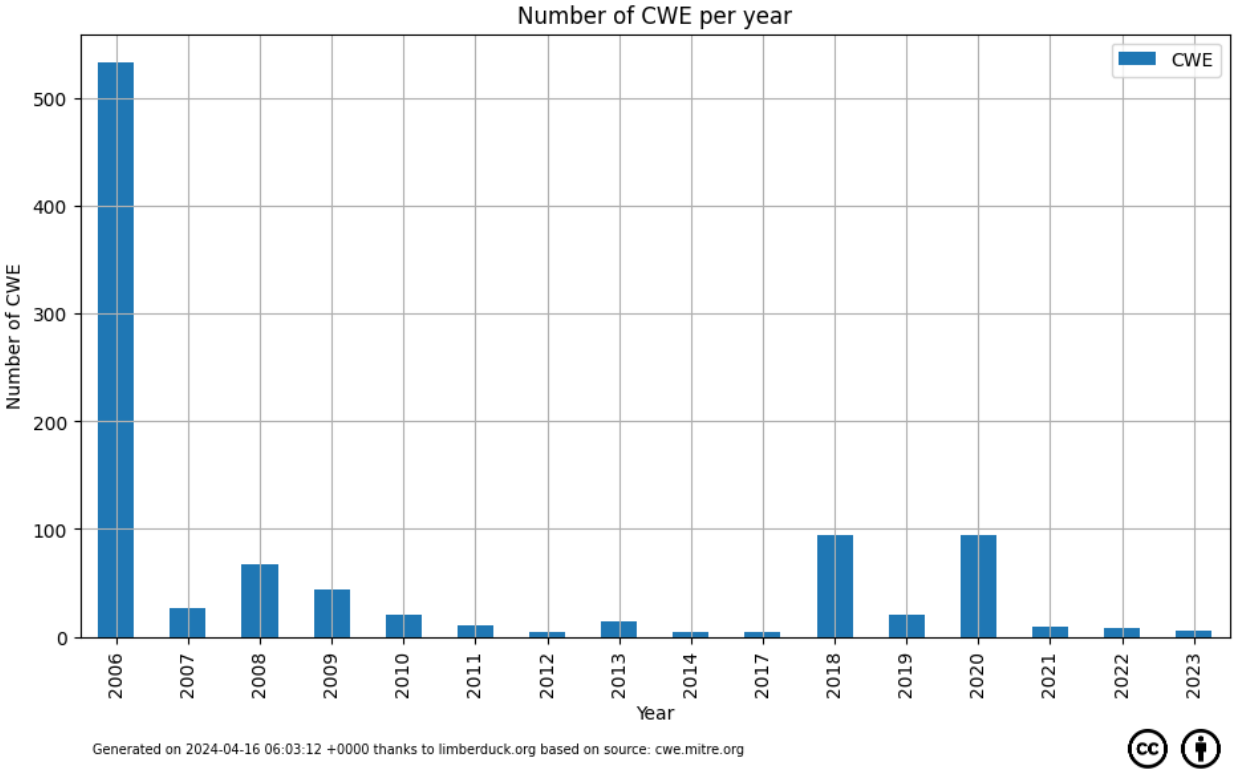
```
csv_filename = 'cwe-number-of-entries.csv'

df.to_csv(csv_filename, index=False)
```

2.3.6 CWE data ploting

PNG files are available in GitHub repository with two background versions, see:

- file via GitHub (white background)
- file via GitHub (transparent background)
- file directly (white background)
- file directly (transparent background)

```
import pandas as pd
import matplotlib.pyplot as plt
import datetime

df = pd.read_csv(csv_filename)

df.plot(x='year',
        xlabel='Year',
        y='number',
        ylabel='Number of CWE',
        kind='bar',
        title='Number of CWE per year')
plt.tight_layout()
plt.legend(['CWE'])
plt.figtext(0.12, 0.02, f"Generated on {generation_time} thanks to limberduck.org based␣
↪on source: cwe.mitre.org", ha="left", fontsize=7)
fig = plt.gcf()
fig.set_size_inches(10,6)
fig.patch.set_facecolor('white')
plt.grid(True)

img_cc = plt.imread('cc.xlarge.png')
newax_cc = fig.add_axes([0.88, 0.0, 0.05, 0.05], anchor='NE', zorder=-1)
newax_cc.imshow(img_cc)
newax_cc.axis('off')
img_by = plt.imread('by.xlarge.png')
newax_by = fig.add_axes([0.92, 0.0, 0.05, 0.05], anchor='NE', zorder=-1)
newax_by.imshow(img_by)
newax_by.axis('off')

plt.savefig('cwe-number-of-entries-bg-white.png', dpi = 300, facecolor = 'white')
plt.savefig('cwe-number-of-entries-bg-transparent.png', dpi = 300, transparent = True)
```

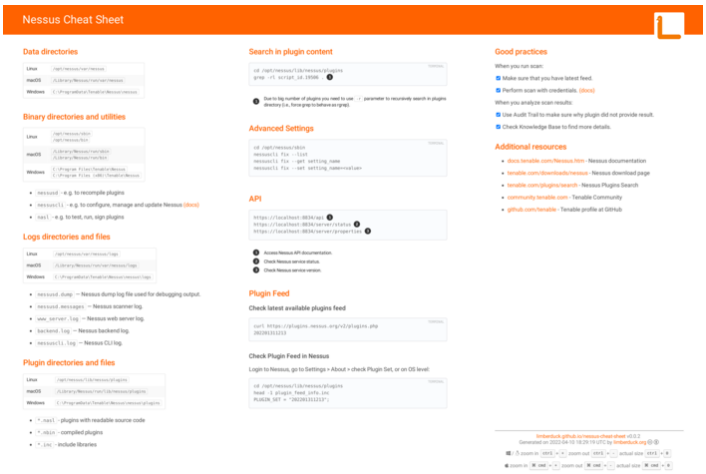

Number of CWE per year
CWE
Number of CWE
0
100
200
300
400
500
2006
2007
2008
2009
2010
2011
2012
2013
2014
2017
2018
2019
2020
2021
2022
2023
Year
Generated on 2024-04-16 06:03:12 +0000 thanks to limberduck.org based on source: cwe.mitre.org

CHAPTER

THREE

CHEAT SHEETS

3.1 Nessus Cheat Sheet



source code

project page

CHAPTER

FOUR

GLOSSARY

Open Source
: Generally, Open Source software is software that can be freely accessed, used, changed, and shared (in modified or unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with the Open Source Definition.

  Source: https://opensource.org/faq#osd

vulnerability
: A vulnerability /vlnrblti/ is a weakness in a system that allows a threat source to compromise its security. It can be a software, hardware, procedural, or human weakness that can be exploited. A vulnerability may be a service running on a server, unpatched applications or operating systems, an unrestricted wireless access point, an open port on a firewall, lax physical security that allows anyone to enter a server room, or unenforced password management on servers and workstations.

  Source: CISSP All-in-One Exam Guide, 8th Edition, 2018, by Shon Harris, Fernando Maymi, page 6

VA
Vulnerability Assessment
: A vulnerability assessment identifies a wide range of vulnerabilities in the environment. This is commonly carried out through a scanning tool. The idea is to identify any vulnerabilities that potentially could be used to compromise the security of our systems. By contrast, in a penetration test, the security professional exploits one or more vulnerabilities to prove to the customer (or your boss) that a hacker can actually gain access to company resources.

  Source: CISSP All-in-One Exam Guide, 8th Edition, 2018, by Shon Harris, Fernando Maymi, page 878

CHAPTER

FIVE

CONTACT

github.com/limberduck

damian.krawczyk@limberduck.org

INDEX