CVE_Analysis_3

December 6, 2020

1 Creating CVE, CPE company, CPE SW, CVSS Nodes in Neo4j

[1]:
```
import os
import json
from neo4j import GraphDatabase
import codecs
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme(style="whitegrid")
import glob
from matplotlib.colors import ListedColormap
import numpy as np
from tqdm.notebook import tqdm
```

2 Connect to Neo4j api

[2]:
```
uri = "neo4j://localhost:7687"
userName = "neo4j"
password = "password"
# Connect to the neo4j database server
graph_db_driver  = GraphDatabase.driver(uri, auth=(userName, password))
```

[3]:
```
def run_query(q):
    with graph_db_driver.session() as graph_db_session:
        try:
            graph_db_session.run(q)
        except:
            print(q)
            raise NameError
```

[4]:
```
base_dir = '/Users/janamian/Documents/workstation/ucsd_dse_program/fall_2019/
 ↪docker_vol/saba-ja/workstation/dse_203_2020/project/
 ↪dse_203_final_project_fall_2020/data'
```

[5]:
```
nvdcve_files = sorted(glob.glob(os.path.join(base_dir, 'nvd_data','nvdcve-1.1*.
↪json')), reverse=True)
```

3 Read NVD json files

[6]:
```
for val in nvdcve_files:
    print(val.split('/')[-1])
```

```
nvdcve-1.1-2020.json
nvdcve-1.1-2019.json
nvdcve-1.1-2018.json
nvdcve-1.1-2017.json
nvdcve-1.1-2016.json
nvdcve-1.1-2015.json
nvdcve-1.1-2014.json
nvdcve-1.1-2013.json
nvdcve-1.1-2012.json
nvdcve-1.1-2011.json
nvdcve-1.1-2010.json
nvdcve-1.1-2009.json
nvdcve-1.1-2008.json
nvdcve-1.1-2007.json
nvdcve-1.1-2006.json
nvdcve-1.1-2005.json
nvdcve-1.1-2004.json
nvdcve-1.1-2003.json
nvdcve-1.1-2002.json
```

[7]:
```
# ##############
# Read all CWE data
# Read all NVD CVE Json files
# ##############
with open(os.path.join(base_dir, 'cwe_data', 'cwec_v4.2.json')) as f:
    cwe = json.load(f)

nvd_list = []
for file_addr in nvdcve_files:
    with open(file_addr) as f:
        nvd_list.append(json.load(f))
```

4 Utility functions

[8]:
```
def get_related_cwe(data_list):
    # CVE object
    resultw = []
    if not isinstance(data_list['problemtype']['problemtype_data'], list):
        print(data_list['problemtype']['problemtype_data'])
        raise ValueError

    if len(data_list['problemtype']['problemtype_data']) != 1:
        print(data_list['problemtype']['problemtype_data'])
        raise ValueError

    for val in data_list['problemtype']['problemtype_data'][0]['description']:
        resultw.append(val['value'])
    return resultw

def get_reference_url(data_list):
    result = []
    for val in data_list['references']['reference_data']:
        result.append(val['url'])
    return result

def get_tags(data_list):
    result = []
    for val in data_list['references']['reference_data']:
        for val2 in val['tags']:
            result.append(val2)

    return result

def get_description_data(data_list):
    result = []
    for val in data_list['description']['description_data']:
        if val['lang'] == 'en':
            result.append(val['value'])
    return result

def get_cpe_match(cpe_match_list):
    result = []
    try:
        for val in cpe_match_list['cpe_match']:
            result.append(val['cpe23Uri'])
    except KeyError:
        pass
    return result
```

```
def get_impacted_configuration(data_list):
    result = []
    for val in data_list['nodes']:

        result.extend(get_cpe_match(val))

        if 'children' in val.keys():
            for val2 in val['children']:
                result.extend(get_cpe_match(val2))

    return result

cve_clean_result = []
total_cwes = 0
total_cves = 0
for nvd_obj in nvd_list:
    for cve_obj in nvd_obj['CVE_Items']:
        published_date = cve_obj['publishedDate']
        yy = published_date.split('-')[0]
        if int(yy) < 2000:
            continue

        modified_date = cve_obj['lastModifiedDate']

        cve_id = cve_obj['cve']['CVE_data_meta']['ID']
        total_cves += 1

        related_cwe_list = get_related_cwe(cve_obj['cve'])
        if len(related_cwe_list) == 0:
            related_cwe_list = ['NVD-no-analysis']
            total_cwes += 1
#             print(cve_id)
        else:
            total_cwes += len(related_cwe_list)

        description = get_description_data(cve_obj['cve'])
        reference_url = get_reference_url(cve_obj['cve'])
        tags = get_tags(cve_obj['cve'])

        try:
            cvss_base_score =␣
 ↪cve_obj['impact']['baseMetricV3']['cvssV3']['baseScore']
            cvss_base_severity =␣
 ↪cve_obj['impact']['baseMetricV3']['cvssV3']['baseSeverity']
        except KeyError:
            cvss_base_score = -1
            cvss_base_severity = 'unknown'
```

```
        impacted_config = get_impacted_configuration(cve_obj['configurations'])

        cve_clean_result.append({
            'cve_id': cve_id,
            'related_cwe_list':related_cwe_list,
            'description': description,
            'reference_url':reference_url,
            'tags':tags,
            'cvss_base_score': cvss_base_score,
            'cvss_base_severity':cvss_base_severity,
            'impacted_config': impacted_config,
            'published_date': published_date,
            'modified_date': modified_date
         })
```

5 Create CVE nodes

[9]:
```
for val in tqdm(cve_clean_result):
    desc = ''
    for d in val['description']:
        desc = desc + " " + d.replace('\\','\\\\').replace('"', '\\"').
 ↪replace("'", "\\'")

    cql_create_node = f"""CREATE (:cve {{ cve_id: "{val['cve_id']}",
    description: "{desc}",
    cvss_base_severity: "{val['cvss_base_severity']}",
    cvss_base_score: {val['cvss_base_score']},
    published_date: {val['published_date'].split('-')[0]}
    }})"""
    run_query(cql_create_node)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=152178.0),␣
 ↪HTML(value='')))
```

6 Create CVE to CWE relations

[10]:
```
for val in tqdm(cve_clean_result):
    cve_id = val['cve_id']
    for val2 in val['related_cwe_list']:
        cwe_id = val2
        cql_create_relationship = f"""MATCH (cve1:cve), (cwe1:cwe)
                        WHERE cve1.cve_id = '{cve_id}' AND cwe1.cwe_id =␣
 ↪'{cwe_id}'
```

```
                            CREATE (cve1)-[r:caused_by]->(cwe1)
                            RETURN type(r)"""
        run_query(cql_create_relationship)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=152178.0),␣
 ↪HTML(value='')))
```

7 Create CVSS nodes

[11]:
```
cvss_score_enum = ['CRITICAL', 'HIGH', 'MEDIUM', 'LOW', 'unknown']
for val in cvss_score_enum:
    cql_create_node = f"""CREATE (:cvss {{ cvss_id: "{val}"}})"""
    run_query(cql_create_node)
```

8 Create CVSS relations to CVE

[20]:
```
for val in tqdm(cve_clean_result):
    cve_id = val['cve_id']
    cvss_id = val['cvss_base_severity']

    cql_create_relationship = f"""MATCH (cve1:cve), (cvss1:cvss)
                            WHERE cve1.cve_id = '{cve_id}' AND cvss1.cvss_id =␣
 ↪'{cvss_id}'
                            CREATE (cve1)-[r:has_severity_of]->(cvss1)
                            RETURN type(r)"""
    run_query(cql_create_relationship)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=152178.0),␣
 ↪HTML(value='')))
```

9 Cleanup CPE company and product strings

[37]:
```
company_set = set()
product_set = set()

for val in cve_clean_result:
    impacted_cpe_list = val['impacted_config']
    for val2 in impacted_cpe_list:
        company_set.add(val2.split(':')[3].replace('\\','').replace('"', '').
 ↪replace("'", "").replace("@", "").replace('+','_'))
        product_set.add(val2.split(':')[4].replace('\\','').replace('"', '').
 ↪replace("'", "").replace("@", "").replace('+','_'))
```

[32]:
```
print(len(company_set))
```

22879

10 Create CPE company nodes

[39]:
```
for val in tqdm(list(company_set)):
    cql_create_node = f"""CREATE (:cpe_comp {{ company_id: "{val}"}})"""
    run_query(cql_create_node)
```

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=22879.0),␣
↪HTML(value='')))

11 Creare CPE product nodes

[38]:
```
for val in tqdm(list(product_set)):
    cql_create_node = f"""CREATE (:cpe_prod {{ product_id: "{val}"}})"""
    run_query(cql_create_node)
```

HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=78994.0),␣
↪HTML(value='')))

12 Create CPE company and Product relation to CVE

[41]:
```
for val in tqdm(cve_clean_result):
    cve_id = val['cve_id']
    impacted_cpe_list = val['impacted_config']
    company_already_connected = set()
    product_already_connected = set()
    for val2 in impacted_cpe_list:
        company = val2.split(':')[3]
        product = val2.split(':')[4]
        if company not in company_already_connected:
            cql_create_relationship = f"""MATCH (cve1:cve), (cpe_comp1:cpe_comp)
                        WHERE cve1.cve_id = '{cve_id}' AND cpe_comp1.
↪company_id = '{company}'
                        CREATE (cve1)-[r:applies_to]->(cpe_comp1)
                        RETURN type(r);"""
            run_query(cql_create_relationship)
            company_already_connected.add(company)

        if product not in product_already_connected:
```

```
            cql_create_relationship = f"""MATCH (cve1:cve), (cpe_prod1:cpe_prod)
                        WHERE cve1.cve_id = '{cve_id}' AND cpe_prod1.
↪product_id = '{product}'
                        CREATE (cve1)-[r:applies_to]->(cpe_prod1)
                        RETURN type(r);"""
            run_query(cql_create_relationship)
            product_already_connected.add(product)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=152178.0),␣
↪HTML(value='')))
```

[ ]: