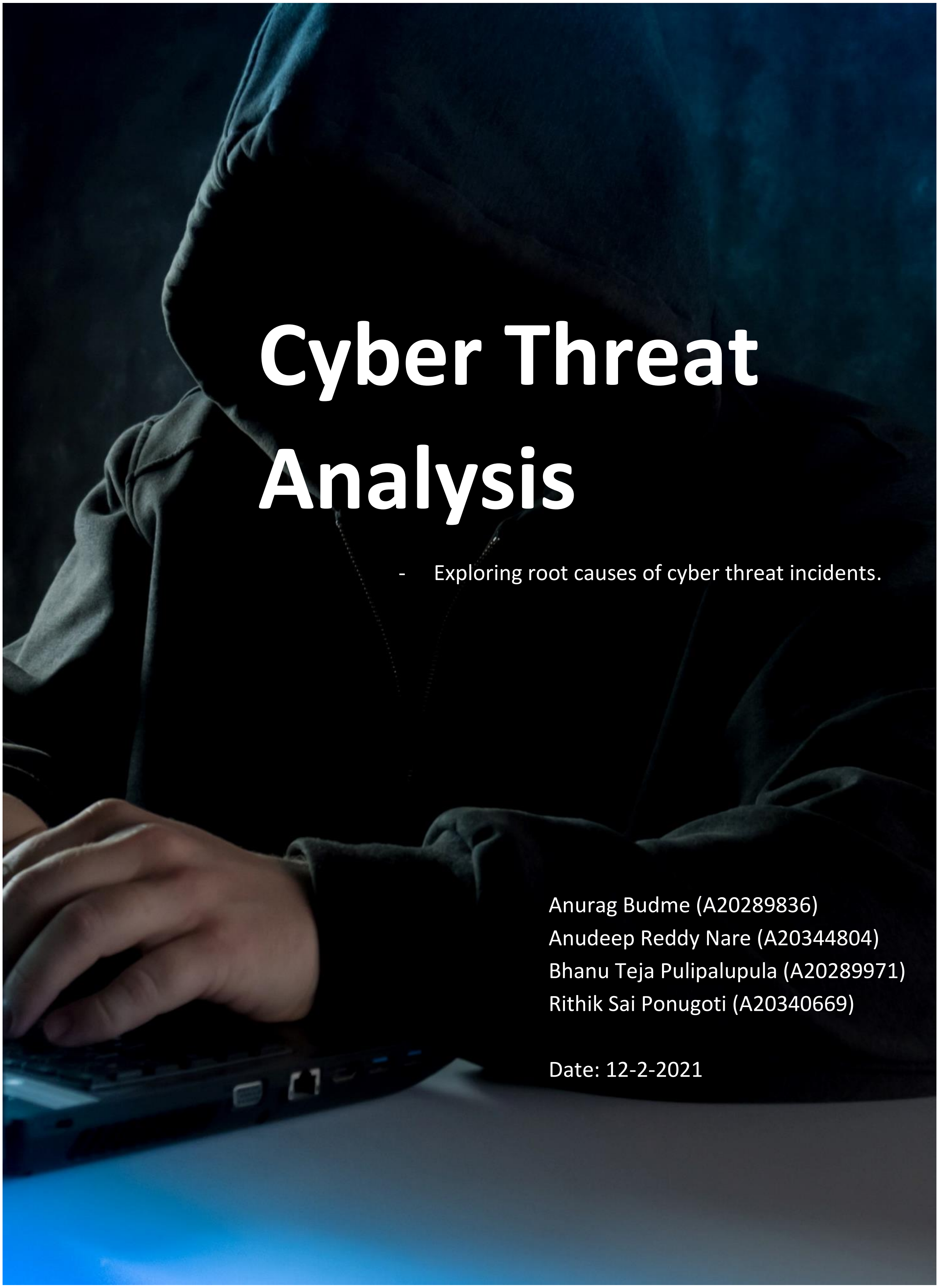


Cyber Threat Analysis

- Exploring root causes of cyber threat incidents.

Anurag Budme (A20289836)
Anudeep Reddy Nare (A20344804)
Bhanu Teja Pulipalupula (A20289971)
Rithik Sai Ponugoti (A20340669)

Date: 12-2-2021

Table of Contents

Executive Summary

Global losses from cybercrime skyrocketed to nearly $1 trillion in 2020. The damage caused by cyber-attacks goes far beyond financial loss, impacting businesses' finances, reputation, operation, valuation, and staff. With people getting technology-enabled and companies rapidly developing their digital platforms, security is the top priority to stay in business and gain customer trust. Business leaders should prioritize making policies and implementing procedures to prevent cyber-attacks and effectively respond if one occurs. By knowing the root causes of cyber-attacks, companies can shape the security systems that help them drive business growth, strengthen customer trust, generate new competitive advantages, and minimize financial loss. Using this analysis, organizations with a digital presence and users can improve their products and services. Companies can be aware of the past trends in security attacks and avoid cyber-attacks/data breaches by taking necessary measures based on the type of attack they're prone to or are currently trending in the internet world.

Statement of Scope

The main goal is to identify the principal root causes and types of cyber threats that hamper the organizational digital transformation journey. This project's scope is limited to scraping information from websites that contain product and vendor-specific vulnerability information relating to cyber security, transforming, and analyzing the collected data.

Our main objectives in the scope of this project are:

- To extract the list of top 20 technology products and their vendors in terms of vulnerability frequency.
- To analyze the data and identify the root causes, types of vulnerabilities.
- To perform exploratory data analysis using the frequency of vulnerabilities by type.
- To perform topic analysis on descriptions of CWE(Common Weakness Enumeration) definitions to find the most common topics or types of vulnerabilities.
- To compare similar technology solutions using vulnerability per product ratio for a specific business need.

This report helps enterprises understand the types of vulnerabilities associated with the products they choose for their business.

Unit of Analysis

Our project analyzes every record of vulnerability specific to the product and vendor that could lead to a cyber-attack. For textual analysis, we used the description of each CWE definition entry.

Variables

We have extracted data from 4 categories, i.e., ‘Vendors’ (526 tables), ‘Products’ (550 tables), ‘Vulnerability Trends Over Time’ (26769 tables) for each product, and ‘Description’ (14 tables) for each CWE definition from CVEDetails.com.

- Vendors Table contains Vendor name, Products, and Number of vulnerabilities.
- Products Table contains the Product name, Vendor name, Number of incidents, Product Type, Vulnerability count, Patch count, Compliance count, Inventory count.
- The table ‘Vulnerability Trends Over Time’ contains Number of Vulnerabilities, DoS, Code Execution, Overflow, Memory Corruption, SQL Injection, XSS, Directory Traversal, HTTP Response Splitting, Bypass something, Gain Information, Gain Privileges, CSRF, File Inclusion, Number of exploits for each product.
- The ‘CWE Definitions’ table contains the CWE Definition URL, Number of Vulnerabilities, Description, Background Details, Other Notes.

Project Schedule

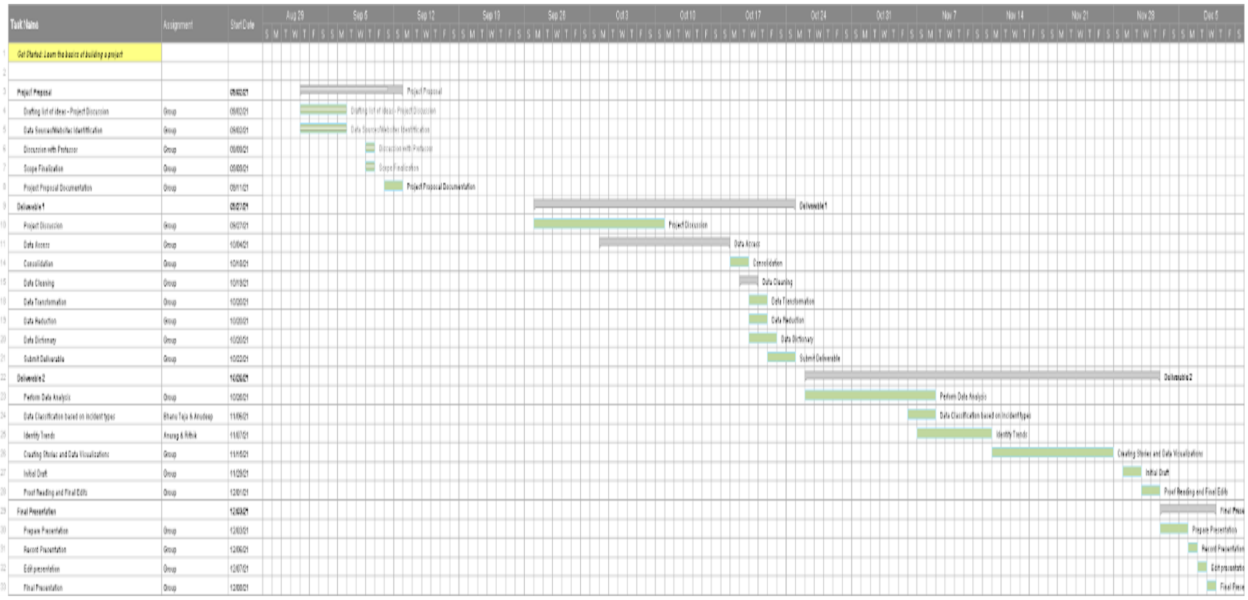
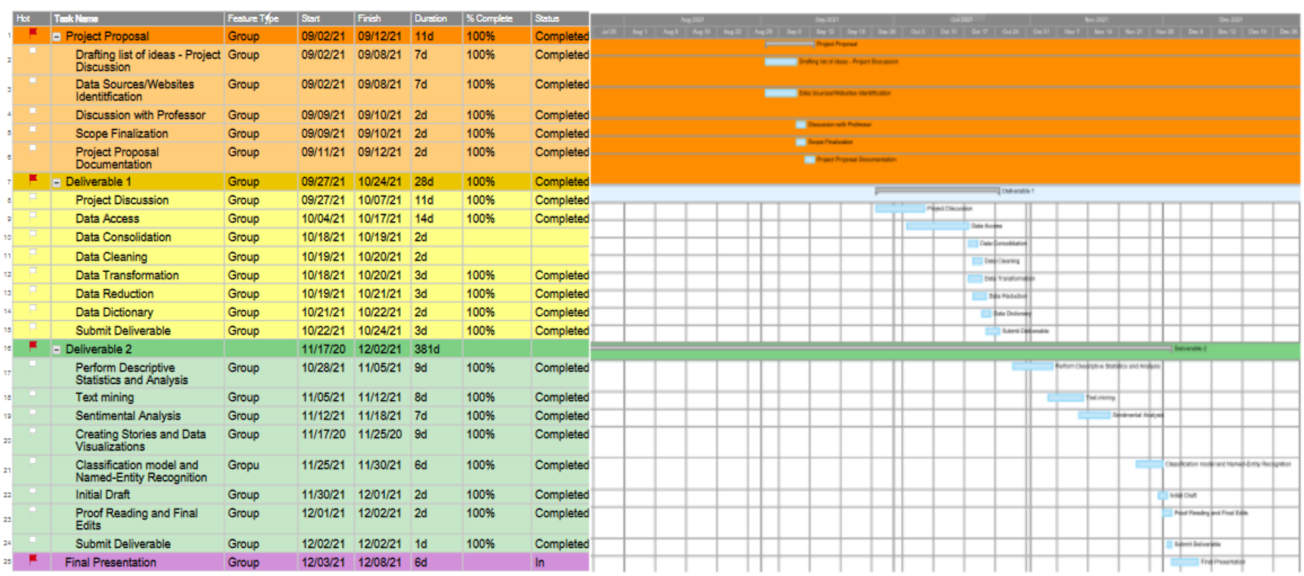


FIGURE 1:DELIVERABLE 1 INITIAL GANTT CHART SCHEDULE



#	Hot	Task Name	Feature Type	Start	Finish	Duration	% Complete	Status
1		Project Proposal	Group	09/02/21	09/12/21	11d	100%	Completed
2		Drafting list of ideas - Project Discussion	Group	09/02/21	09/08/21	7d	100%	Completed
3		Data Sources/Websites Identitfication	Group	09/02/21	09/08/21	7d	100%	Completed
4		Discussion with Professor	Group	09/09/21	09/10/21	2d	100%	Completed
5		Scope Finalization	Group	09/09/21	09/10/21	2d	100%	Completed
6		Project Proposal Documentation	Group	09/11/21	09/12/21	2d	100%	Completed
7		Deliverable 1	Group	09/27/21	10/24/21	28d	100%	Completed
8		Project Discussion	Group	09/27/21	10/07/21	11d	100%	Completed
9		Data Access	Group	10/04/21	10/17/21	14d	100%	Completed
10		Data Consolidation	Group	10/18/21	10/19/21	2d		
11		Data Cleaning	Group	10/19/21	10/20/21	2d		
12		Data Transformation	Group	10/18/21	10/20/21	3d	100%	Completed
13		Data Reduction	Group	10/19/21	10/21/21	3d	100%	Completed
14		Data Dictionary	Group	10/21/21	10/22/21	2d	100%	Completed
15		Submit Deliverable	Group	10/22/21	10/24/21	3d	100%	Completed
16		Deliverable 2		11/17/20	12/02/21	381d		
17		Perform Descriptive Statistics and Analysis	Group	10/28/21	11/05/21	9d	100%	Completed
18		Text mining	Group	11/05/21	11/12/21	8d	100%	Completed
19		Sentimental Analysis	Group	11/12/21	11/18/21	7d	100%	Completed
20		Creating Stories and Data Visualizations	Group	11/17/20	11/25/20	9d	100%	Completed
21		Classification model and Named-Entity Recognition	Gropu	11/25/21	11/30/21	6d	100%	Completed
22		Initial Draft	Group	11/30/21	12/01/21	2d	100%	Completed
23		Proof Reading and Final Edits	Group	12/01/21	12/02/21	2d	100%	Completed
24		Submit Deliverable	Group	12/02/21	12/02/21	1d	100%	Completed
25		Final Presentation	Group	12/03/21	12/08/21	6d		In

FIGURE 2:DELIVERABLE 2 INITIAL GANTT CHART SCHEDULE

Data Preparation

Data Access

Data Sources :

- CVE details website: https://www.cvedetails.com/
    - Products list: https://www.cvedetails.com/product-list.php
    - Vendors list: https://www.cvedetails.com/vendor.php
    - Vulnerabilities by Type for Product: https://www.cvedetails.com/product-search.php
    - CWE Definitions: https://www.cvedetails.com/cwe-definitions.php

Initially, we planned to extract the data from the National Vulnerability Database that reports the cyber vulnerabilities. However, after discussing with the professor about the data sources, we learned about the CVE details repository that fetches data from the source we planned to use before.

As there's no specific API to fetch data from cvedetails.com, we've scraped the data using the BeautifulSoap package. The mentioned website comprises the data of variables listed above in multiple tables, covering products and vendors. The web crawler fetches the data from tables row by row, scraping from one web page to another.

The URL format below is the same for multiple pages except for the page number. The data is accessed from multiple pages by using this distinction.

Products:

http://www.cvedetails.com/product-list/product_type-/vendor_id-0/firstchar-A/page-{}/products.html?sha=6e3421fae68ad74b2f60561745bc909432a34f76&trc=6125&order=1

Vendors:

https://www.cvedetails.com/vendor/firstchar-A/{}/?sha=50b75327da93bc47b6520cb791181fdbc5f13d9b&trc=1841&order=1

CWE Definitions:

https://www.cvedetails.com/cwe-definitions/{}/cwelist.html?order=2&trc=668&sha=0427874cc45423ccb6974ee25935fbfceac76fcb

Data Consolidation

We created the consolidated data set by joining three tables: Products, Vendors, and Vulnerabilities by Type for each product.

In the first step, we merged ‘Products’ and ‘Vendors’ data, fetched from the individual pages with respective page links in Products and Vendors web pages on the ‘VendorName’ column. This step provided the output file containing all the columns from the Products and Vendors tables.

In the second step, the above-created data set is merged with the ‘Vulnerabilities by Type for Product’ table on the ‘ProductName’, ‘VendorName’ columns. As there are multiple products with the same name associated with a different vendor, we had to join these two tables on the combination of ProductName, VendorName as this will be unique. This step gave us the final consolidated file with the columns mentioned in the data dictionary.

While merging the tables, we observed some leading and trailing spaces for the ‘Product Name’ column in the table fetched in the first step. We eliminated these spaces before moving on to the second step.

Before merging the 'Products-Vendors' and 'Vulnerabilities by Type for Products' tables, we filtered out the rows containing aggregated counts for each vulnerability type.

We extracted each table from the CWE definitions web page for textual data. These tables are fetched from individual pages using the ‘CWE Number’ hyperlink in the 'CWE Definitions table'. We transposed the tables to get a resulting table with columns 'CWE Definition', ‘Number of Vulnerabilities’, Description, ‘Background Details’, ‘Other Notes’. Finally, we consolidated all these individual transposed tables into a single CSV file.

Data Cleaning

After consolidation, we have observed multiple records that do not have any information for the total count of vulnerabilities. Since the missing values represent factual data related to security vulnerabilities, we did not assume or impute any values based on statistical calculations. We dropped the records for which the column ‘Vulnerabilities_y’ (Number of Vulnerabilities specific to Vendor of the product) has a value blank or 0. Also, the column ‘Vulnerabilities_y’ is renamed to ‘Total_No_of_Vulnerabilities_by_Vendor’.

We found special characters like '//', '/' in Product and Vendor names while observing the consolidated data. As these characters are not appropriate for a Product/Vendor name, we replaced these characters with an empty string.

For textual data, we chose the 'Description' column for analysis. We converted the upper-case characters of this column to lowercase, removed numerical values, punctuations, white spaces, and Stop words from the text. We took these steps to get better insights out of our analysis.

Data Transformation

We've constructed a new attribute that calculates the contribution of product vulnerability to a vendor (Vulnerability count of a Product / Total Vulnerability count of that Product's Vendor).

This attribute identifies the contribution of a product to the vulnerabilities of its vendor. When there are multiple products from a specific vendor, and if one wants to know the impact of particular product vulnerabilities on the vendor, the above newly built attribute will provide such information.

We've constructed another attribute, 'vulnerabilities_to_product_ratio', that calculates the average number of vulnerabilities per product. For a newly launched product from a vendor, this attribute gives a rough estimate of the number of vulnerabilities.

Data Reduction

From the consolidated table obtained by merging Products table, Vendors table, and Vulnerabilities Trends Over Time with Type, we've dropped the columns: 'Year', 'Patches', 'Compliance', 'Inventory' as we planned to consider only vulnerability data in our scope of this project.

For textual data, we have dropped the columns of 'CWE Definition', 'Background Details', 'Other Notes' as we planned to perform textual analysis only on the 'Description' column that comprises CWE(Common Weakness Enumeration) descriptions.

Data Dictionary

Attribute Name	**Description**	**Data Type**	**Source**
ProductName	Name of the product	char	https://www.cvedetails.com/product-list.php
VendorName	Name of the vendor that owns the product	char	https://www.cvedetails.com/product-list.php
No.ofCVEEntries	Number of common vulnerabilities of a product	integer	https://www.cvedetails.com/product-list.php
ProductType	Type of Product (Ex: Hardware, OS, Application)	char	https://www.cvedetails.com/product-list.php
Vulnerabilities_ x	Number of Vulnerabilities specific to a product	integer	https://www.cvedetails.com/product-list.php
Products	Number of products	char	https://www.cvedetails.com/vendor.php
Total_No_of_Vulnerabilities_by_Vendor	Number of vulnerabilities specific to Vendor of the product	char	https://www.cvedetails.com/vendor.php

# of Vulnerabilities	Total number of vulnerabilities specific to a product	integer	https://www.cvedetails.com/product-search.php
DoS	Number of vulnerabilities caused by DoS for a specific product	integer	https://www.cvedetails.com/product-search.php
Code Execution	Number of vulnerabilities caused by Code Execution for a specific product	integer	https://www.cvedetails.com/product-search.php
Overflow	Number of vulnerabilities caused by Overflow for a specific product	integer	https://www.cvedetails.com/product-search.php
Memory Corruption	Number of vulnerabilities caused by Memory Corruption for a specific product	integer	https://www.cvedetails.com/product-search.php
SQL Injection	Number of vulnerabilities caused by SQL Injection for a specific product	integer	https://www.cvedetails.com/product-search.php
XSS	Number of vulnerabilities caused by XSS for a specific product	integer	https://www.cvedetails.com/product-search.php
Directory Traversal	Number of vulnerabilities caused by Directory Traversal for a specific product	integer	https://www.cvedetails.com/product-search.php
HTTP Response Splitting	Number of vulnerabilities caused by HTTP Response Splitting for a specific product	integer	https://www.cvedetails.com/product-search.php
Bypass something	Number of vulnerabilities caused by Bypass something for a specific product	integer	https://www.cvedetails.com/product-search.php
Gain Information	Number of vulnerabilities caused by Gain Information for a specific product	integer	https://www.cvedetails.com/product-search.php
Gain Privileges	Number of vulnerabilities caused by Gain Privileges for a specific product	integer	https://www.cvedetails.com/product-search.php

CSRF	Number of vulnerabilities caused by CSRF for a specific product	integer	https://www.cvedetails.com/product-search.php
File Inclusion	Number of vulnerabilities caused by File Inclusion for a specific product	integer	https://www.cvedetails.com/product-search.php
# Of exploits	Number of vulnerabilities caused by # of exploits for a specific product	integer	https://www.cvedetails.com/product-search.php
contribution_of_product_vulnerability_to_vendor	Numeric value depicting the contribution of product vulnerability to a vendor.	float	derived
vulnerabilities_to_product_ratio	Average number of vulnerabilities per product w.r.t vendor	float	derived
Description	Description of the CWE entry	char	https://www.cvedetails.com/cwe-definitions.php

TABLE 1.DATA DICTIONARY

Descriptive Statistics and Analysis

For application-type products, here's the bar plot for the top 20 products with respect to the number of vulnerabilities:

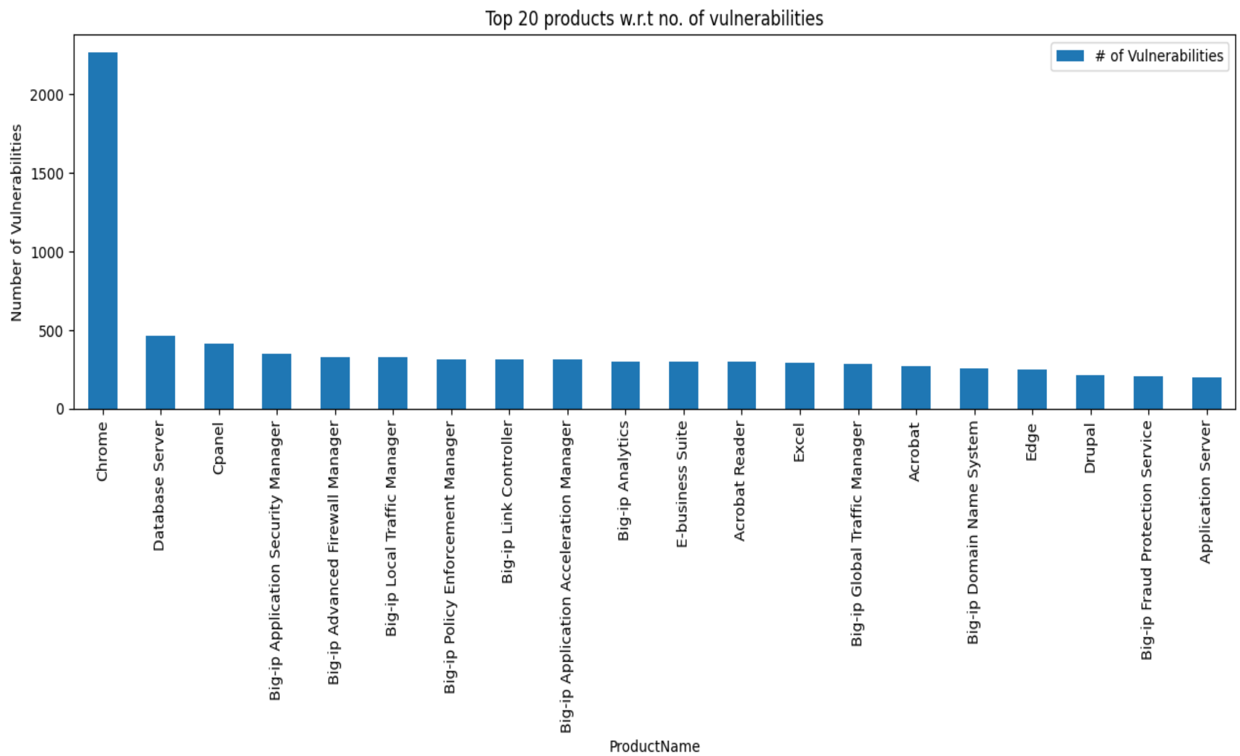


FIGURE 3: Top 20 Products with respect to the number of vulnerabilities

From the above graph, we interpret the following:

1. The top product is the web browser Chrome. Interestingly, another web browser Microsoft Edge is on the top twenty list of products.
2. cPanel, Drupal and Adobe have content management systems (CMS) products under the web applications category.

3. Application Server, Database Server are products from multiple vendors like Oracle, SAP, HP. The platforms that host web applications will generally contain an application server and database server.
4. Products with names ‘Big IP’ are vendor F5’s proprietary products. F5 is a cloud management & application security company. As this company (F5) has a wide range of network & web servers, cloud products, the user/client base might be high, which in turn means that many end-users might get affected by the vulnerabilities of ‘Big IP’ products from the vendor ‘F5’. Notably, top companies like Microsoft, Oracle, and Facebook use the products of F5.

Overall, the top 20 products majorly fall under the content management systems or are web application-based, coming from vendors like Microsoft, Adobe, Oracle, F5, Google, Drupal, cPanel.

Here’s the frequency plot of overall type of vulnerability statistics:

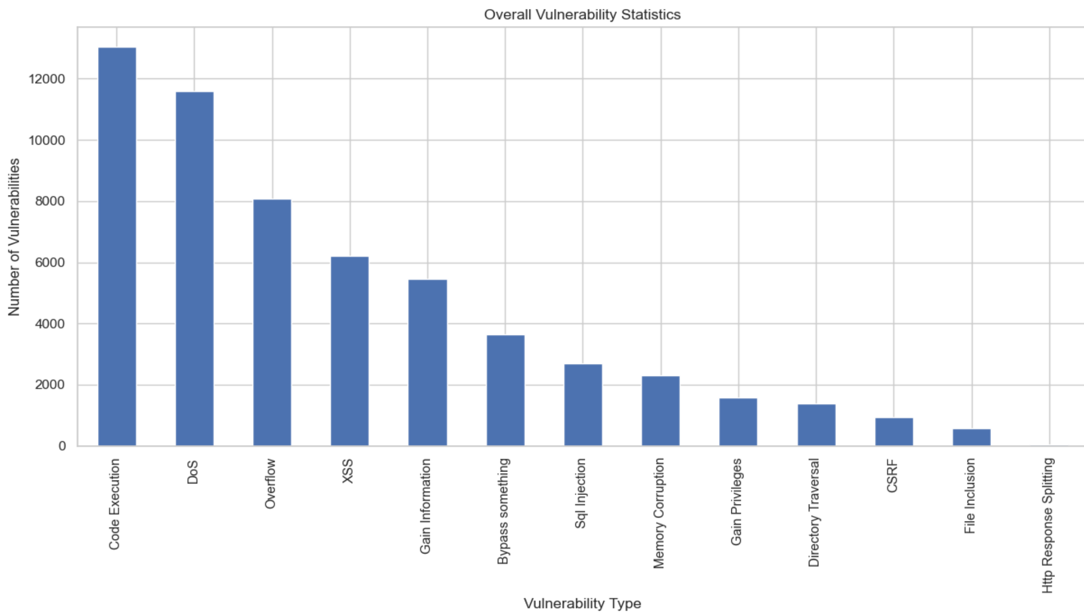


FIGURE 4: Overall vulnerability statistics

The top 4 type of vulnerabilities are,

1. Code Execution
2. DoS
3. Overflow
4. XSS (Cross site scripting)

As we observe that code execution is the top reason for most cyberattacks, we suggest that application developers write database queries /server-side code by considering the possible attacks due to flawed code.

Apart from code execution, XSS and Overflow are other two crucial vulnerabilities that are also found from the text analysis of description from CWE definitions data.

Here's the frequency plot for top 20 vendors with respect to number of vulnerabilities,

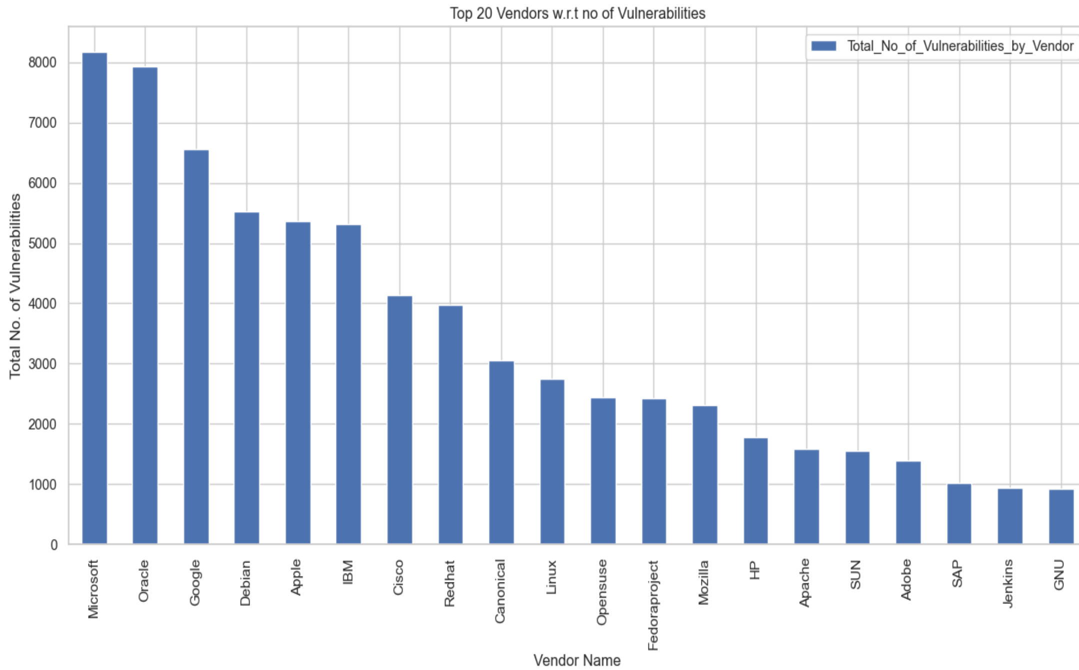


Figure 5: Top 20 vendors with respect to the number of vulnerabilities

For application-type products, the top 10 vendors with respect to vulnerabilities are Microsoft, Oracle, Google, Debian, Apple, IBM, Cisco, Redshift, Canonical, Linux.

Based on our previous analysis, our top 20 products with respect to vulnerabilities belong to only seven vendors, and of these seven vendors, four are on the top 10 vendor list for vulnerabilities.

Here's our analysis based on our derived attribute vulnerabilities_to_product_ratio for a vendor.

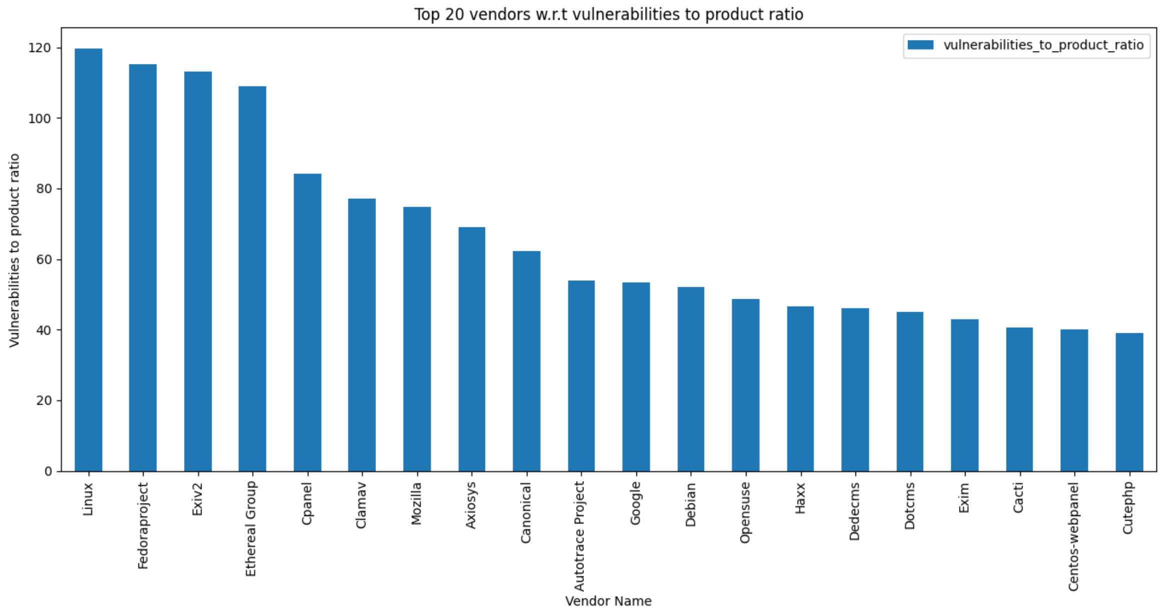


Figure 6: Top 20 vendors with respect to the vulnerabilities to product ratio

The above bar plot denotes the average number of vulnerabilities of a product that belongs to a specific vendor. For example, for the vendor Linux, the average number of vulnerabilities for a product is close to 120.

Text Mining & Sentiment Analysis

Topic Analysis:

From the CWE Definitions table, we've performed the text analysis on the column Description, which contains the textual data.

Before performing topic modeling, we cleaned the text description. On the cleaned text, we've used PorterStemmer to stem the words in the sentences to reduce the redundancy of words in the sentence.

We've performed topic analysis using the LDA topic model resulting in four topics.

```
Top 10 words for topic #0:
['receiv', 'data', 'neutral', 'handl', 'input', 'compon', 'special', 'element', 'incorrectli', 'softwar']


Top 10 words for topic #1:
['allow', 'access', 'path', 'directori', 'product', 'function', 'file', 'attack', 'softwar', 'use']


Top 10 words for topic #2:
['oper', 'access', 'applic', 'user', 'use', 'inform', 'file', 'softwar', 'attack', 'resourc']


Top 10 words for topic #3:
['protect', 'attack', 'applic', 'buffer', 'memori', 'input', 'softwar', 'valid', 'use', 'data']
```

From the topics listed above, we've inferred the following,

Topic 1 (#0): When software receives input from a component, it incorrectly interprets special elements as input, resulting in incorrect output. We can call this XSS (Cross-Site Scripting).

Topic 2 (#1): Attack on software product allowed access to the secured file directory, resembling Directory Traversal Attack.

Topic 3 (#2): Attacker users gain access to intrude into applications to access file resources, indicating 'File inclusion' type of vulnerability.

Topic 4 (#3): We anticipate memory buffer overflow where attackers attack the software (web page) by providing too much information that causes a buffer overflow. It happens when the size of input data is greater than the memory allocated to that attribute, coming under the 'Overflow' type of vulnerability.

The following figure shows the distribution of text descriptions in topics using LDA:

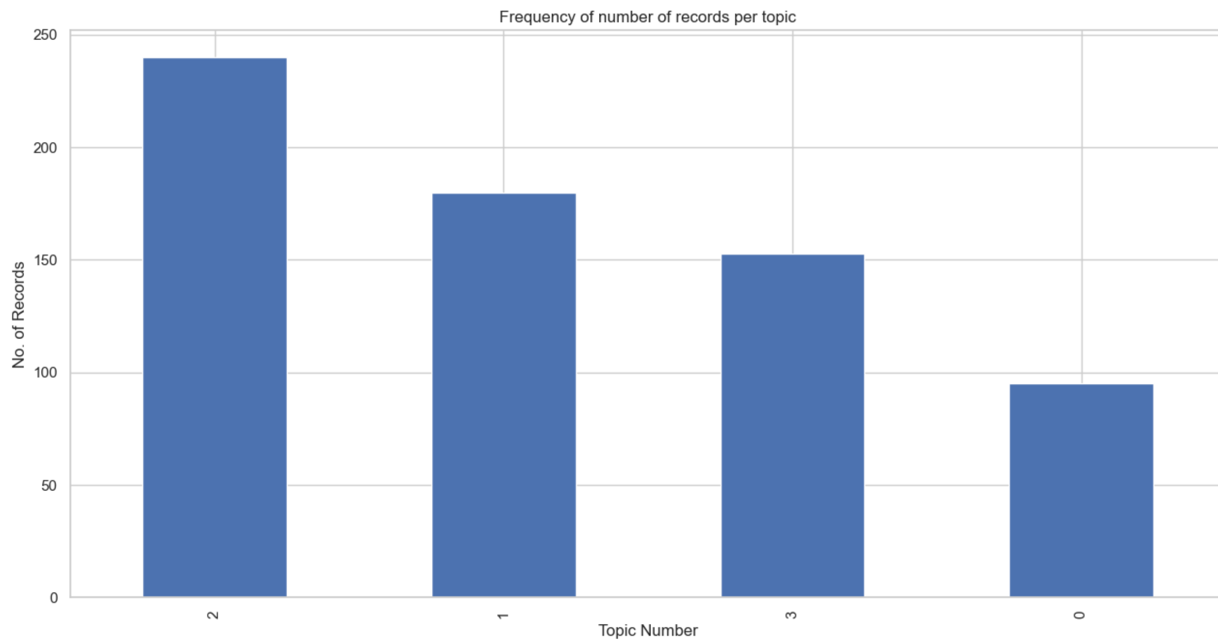


Figure 7: Frequency of number of records per topic

Emotion Analysis and Visualizations

Though it seems challenging to measure emotions related to cyber vulnerabilities, after analyzing the description text in our data, we've decided to check the text for pairs of emotions 'trust and fear', 'positive and negative'. The NRC lexicon is a unigrams emotion lexicon. It has a dictionary of the words and their corresponding sentiment/emotion.

Trust & Fear

We found that 813 words were associated with trust emotion. Below is the word cloud interpreting the same.

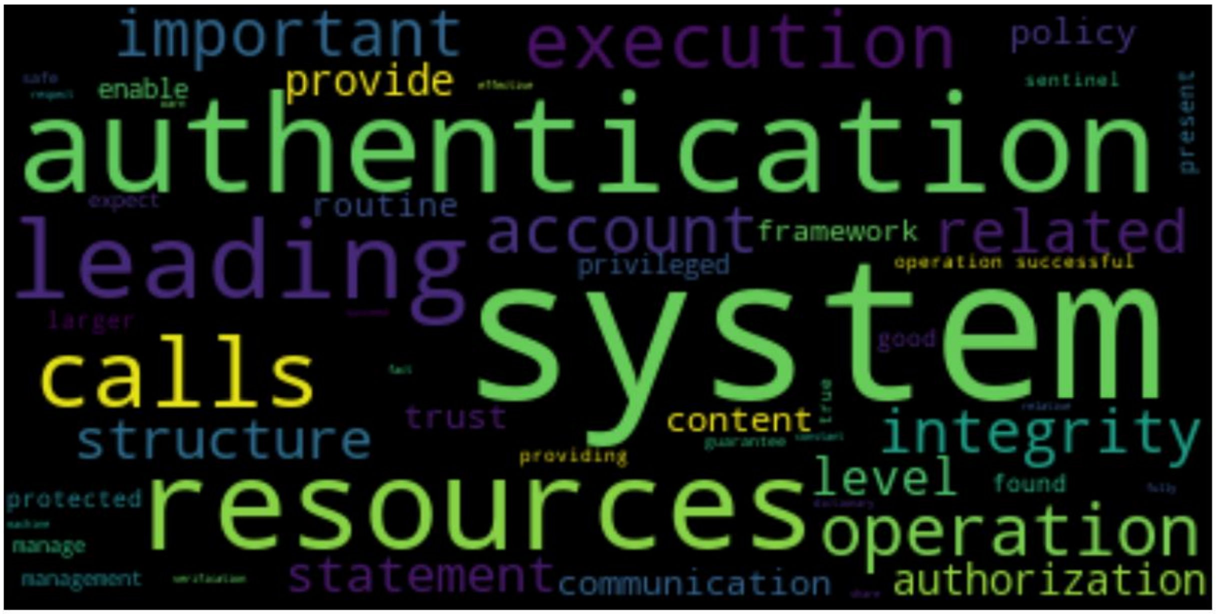


Figure 8: Word Cloud for trust emotion

On the other hand, 585 words were assigned to the fear emotion. Below is the word cloud interpreting the same.

Figure 9: Word Cloud for fear emotion

Positive & Negative

We found that 1213 words were associated with positive emotion. Below is the word cloud interpreting the same.

Figure 10: Word Cloud for positive emotion

On the other hand, 982 words were assigned to the negative emotion. Below is the word cloud interpreting the same.

Figure 11: Word Cloud for negative emotion

Classification Model:

Due to a lack of data labeling, most business problems do not reach the solution stage. The process of data labeling is generally carried out by a team of data professionals who use business logic or pre-defined generic packages and industry professionals with domain expertise. Once they get a certain amount of classified data, they use it for model training and testing. Data labeling is crucial for any classification problem.

Here we have a similar problem with our data set. The description column from CWE definitions is not classified data. To solve this, we used a package called "Textblob" for processing textual data to generate sentiment analysis. This package provides the polarity of the statement by considering the positive and negative sentiment in the sentence. The drawback with this package is that it is generic and does not consider any industry-specific terms. However, for initial analysis, this package helped us to get started.

Here is a bar plot that shows how the sentiment is distributed across the CWE description text records.

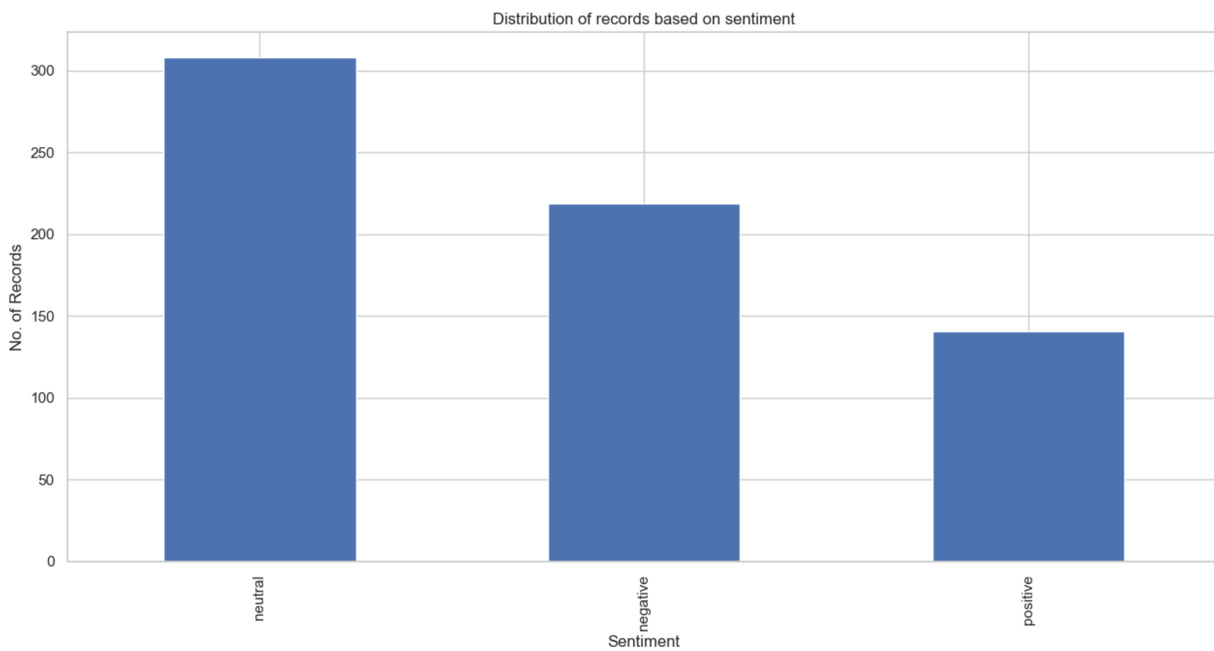


Figure 12: Distribution of record based on sentiment

Text description from the CWE definitions table was input as features to predict the emotion of the description. After cleaning the text, a TF-IDF matrix is produced with 2500 maximum features for the description column. The original data got split into 75% for training and 25% for testing, and the random forest classifier model was used to build the classification model. The overall accuracy was 0.68 on the test data. In the classification report below, precisions of positive and negative are 0.75 and 0.81. The F1-score for positive is 0.36, and the F1-score for negative is 0.72.

```
              precision    recall  f1-score   support

    negative       0.81      0.66      0.72        58
     neutral       0.61      0.93      0.74        71
    positive       0.75      0.24      0.36        38

    accuracy                           0.68       167
   macro avg       0.72      0.61      0.61       167
weighted avg       0.71      0.68      0.65       167
```

Here’s the model confusion matrix for the positive, neutral, and negative emotions,

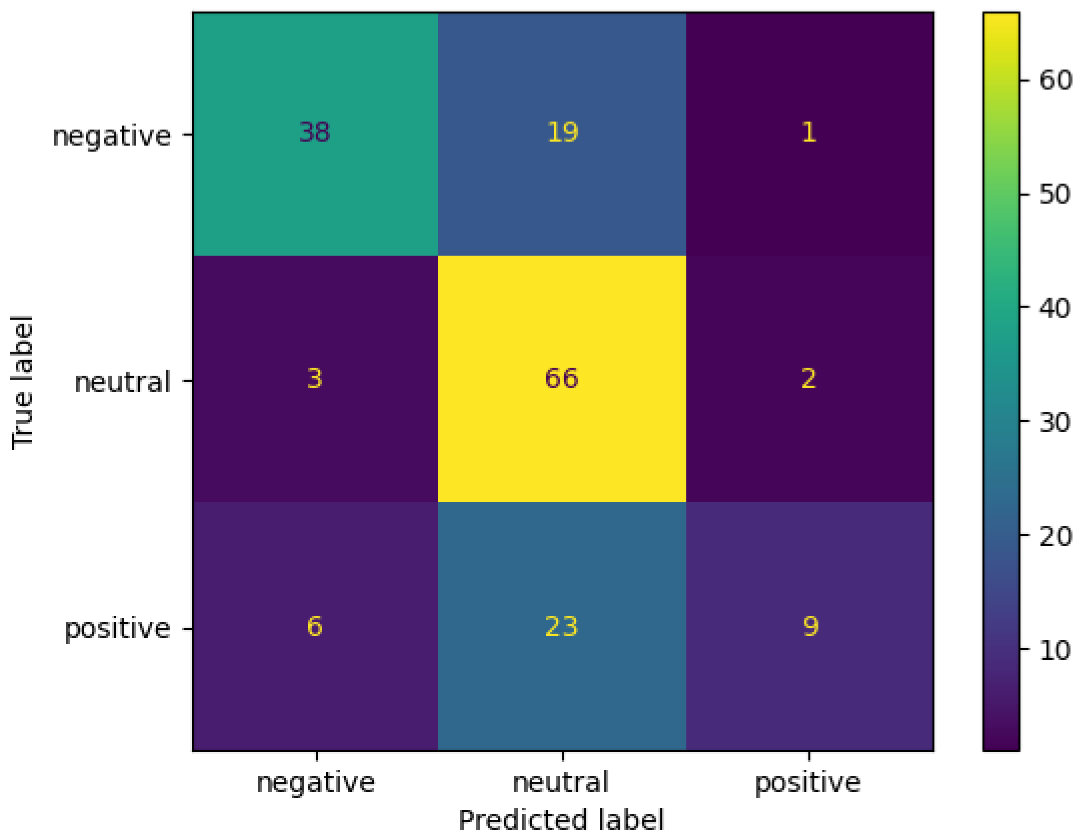


Figure 13: Confusion matrix for positive, neutral, and negative emotions

As this description is about vulnerabilities and attacks it contains negative, neutral sentiment words. By using the model confusion matrix, it is found that the model performs better for neutral and negative sentiment than for positive sentiment, making the model promising.

Named-Entity Recognition

To extract entities from the text in the Description column, we used the nltk library. It is required to tokenize the text before passing it via a POST filter and a chunking process. We passed the tagged items to the named entity chunker after the POST process. The nltk library contains several chunks: LOCATION, ORGANIZATION, PERSON, DURATION, DATE, CARDINAL, PERCENT, MONEY, and MEASURE. In our case, as expected, for PERSON, LOCATION, we did not get any results.

The description of vulnerabilities yields an interesting result: we expected the column ORGANIZATION to have some information about the companies/vendors that have these vulnerabilities, but to our surprise, the column did not contain any information. After digging deep, as we checked a few descriptions, we learned that the CWE descriptions were generic and did not mention any specific vendor/organization names.

Conclusion

In conclusion, the datasets were able to generate insights about the vulnerability types related to application products. Products in the web applications, content management systems (CMS) category have the most vulnerabilities. Code Execution is the most occurring type of vulnerability for all vendors. Surprisingly, internet giants like Google, Microsoft, Apple, Oracle are in the top list of vendors by vulnerabilities. For the future scope of this project, it would be beneficial to consider data for a more large-scale range of products, collecting data from multiple sources apart from the CVE details website and the latest cyber security news from authentic journals and websites.

References:

- https://www.washingtonpost.com/politics/2020/12/07/cybersecurity-202-global-losses-cybercrime-skyrocketed-nearly-1-trillion-2020/
- https://www2.deloitte.com/uk/en/pages/consumer-business/articles/accelerated-digitalisation-leave-businesses-susceptible-to-cyberattacks.html
- https://www.dqindia.com/failure-prioritize-cybersecurity-hampering-digital-transformation-journey-organization-report/
- https://nvd.nist.gov/vuln/full-listing
- https://www.cvedetails.com/product-list.php
- https://www.cvedetails.com/cwe-definitions.php

Appendix

GitHub Files

https://github.com/anurag-osu/cyber_threat_analysis/

Code

products_data_scraping.py

```
#Data Extraction
#CVE Details website Products Data Extraction

#Importing packages
from bs4 import BeautifulSoup
import requests
import logging
import random
import time
import pandas as pd
import os

#Configuring logging information in a log file
logging.basicConfig(filename='logfile.log', filemode='w',
format='%(asctime)s - %(message)s', level=logging.INFO)
logging.info("This is the first logging message")

#Changing directory to save files
os.chdir(r'C:\Users\budme\Documents\GitHub\cyber_threat_analysis\products_
cve\C')

#Defining a function that takes product page url, uses beautiful soup to
find tables
#Tables fetched are put into data frames
def table_extraction(url,page_number):
    logging.info("working on the url: %s",url)
    req = requests.get(url)
    soup=BeautifulSoup(req.content,'html.parser')
    table=soup.find("table",attrs={"class":"listtable"})

    try:
        final_data=[]
        for row in table.find_all("tr")[2:]:
            data=[t.get_text() for t in row.find_all("td")]
            final_data.append([i.replace('\t','').replace('\n','') for i
in data])
```

```
        final_data_dataframe=pd.DataFrame(final_data)
        #Saving products with starting character 'C' and using page number
in the file name
        #this was changed while running the script for products starting
with each alphabet
        filename_csv="table_C_{}.csv".format(page_number)
        final_data_dataframe.to_csv(filename_csv)
        logging.info("Success on the page: %s",page_number)
    except Exception as e:
        #Logging failure with page number and exception message
        logging.info('Failed on the page: %s', page_number)
        logging.info('Exception Message: %s',str(e))

#below url has the option to change the range
#Looping through the URL by changing page number and calling the function
for i in range(1,174):
    url="http://www.cvedetails.com/product-list/product_type-/vendor_id-
0/firstchar-C/page-
{}/products.html?sha=6e3421fae68ad74b2f60561745bc909432a34f76&trc=6125&ord
er=1".format(i)

    if i%2==0:
        table_extraction(url,i)
        random_number=random.randint(1,3)
        time.sleep(random_number)
    else:
        table_extraction(url,i)
        random_number=random.randint(3,5)
        time.sleep(random_number)
```

vendors data scraping.py

```
#Data Extraction
#CVE Details website Vendors Data Extraction

#Importing packages
from bs4 import BeautifulSoup
import requests
import logging
import random
import time
import pandas as pd
import os

#Configuring logging information in a log file
logging.basicConfig(filename='logfile.log', filemode='w',
format='%(asctime)s - %(message)s', level=logging.INFO)
logging.info("This is the first logging message")

#Changing directory to save files
```

```
os.chdir(r'C:\Users\budme\Documents\GitHub\cyber_threat_analysis\vendors_c
ve\B')

#Defining a function that takes vendor page url, uses beautiful soup to
find tables
#Tables fetched are put into data frames
def table_extraction(url,page_number):
    logging.info("working on the url: %s",url)
    req = requests.get(url)
    soup=BeautifulSoup(req.content,'html.parser')
    table=soup.find("table",attrs={"class":"listtable"})

    try:
        final_data=[]
        for row in table.find_all("tr")[1:]:
            data=[t.get_text() for t in row.find_all("td")]
            final_data.append([i.replace('\t','').replace('\n','') for i
in data])

        final_data_dataframe=pd.DataFrame(final_data)
        filename_csv="table_B_{}.csv".format(page_number)
        final_data_dataframe.to_csv(filename_csv)
        logging.info("Success on the page: %s",page_number)
    except Exception as e:
        #Logging failure with page number and exception message
        logging.info('Failed on the page: %s', page_number)
        logging.info('Exception Message: %s',str(e))

#below url has the option to change the range
#Looping through the URL by changing page number and calling the function
for i in range(1,23):
    url="https://www.cvedetails.com/vendor/firstchar-
B/{}/?sha=50b75327da93bc47b6520cb791181fdbc5f13d9b&trc=1841&order=1".forma
t(i)

    if i%2==0:
        table_extraction(url,i)
        random_number=random.randint(1,3)
        time.sleep(random_number)
    else:
        table_extraction(url,i)
        random_number=random.randint(3,5)
        time.sleep(random_number)
```

vulnerabilities_by_product.py

```
#Loading Packages
from bs4 import BeautifulSoup
import requests
```

```
import logging
import random
import time
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common import keys
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import random
import os

#Changing the Working Directory
os.chdir(r'C:\Users\budme\Desktop\Assignments\5193_ProgrammingForDS\cyber_
threat_analysis_v2')

#LogFile Configuration to record logs.
logging.basicConfig(filename='logfile.log', filemode='w',
format='%(asctime)s - %(message)s', level=logging.INFO)

#Initialize the chromedriver selenium.
driver=webdriver.Firefox(executable_path=r'C:\Users\budme\Documents\GitHub
\geckodriver.exe')
time.sleep(5)

#Dictionary which contains the ProductName and URL
final_records=[]

def ProductName_VendorName_and_URL_mapper(startpage,endpage):

    #Avoid Continuous Hits on webpage.
    for i in range(startpage,endpage):
        if i%2==0:
            pass
            #time.sleep(random.randint(1,2))
        else:
            time.sleep(random.randint(1,2))


        url="https://www.cvedetails.com/product-list/product_type-
/vendor_id-0/firstchar-E/page-
{}/products.html?sha=eaa58145d4fcdef9eb08e9ce6c33792e6c27ce44&trc=8892&ord
er=1".format(i)
        driver.get(url)

        for j in range(3,53):
            try:
                final_dic={}
                xpath='/html/body/table/tbody/tr[2]/td[2]/div/table/tbody/
tr/td[1]/form/table/tbody/tr[{}]/td[1]/a'.format(j)
                vendor_link=driver.find_element_by_xpath(xpath)
                vendor_link.click()
                final_url=driver.current_url
```

```
                vendor_name=driver.find_element_by_xpath('/html/body/table
/tbody/tr[2]/td[2]/div/h1/a[1]')
                vendor_name=vendor_name.text

                product_name=driver.find_element_by_xpath('/html/body/tabl
e/tbody/tr[2]/td[2]/div/h1/a[2]')
                product_name=product_name.text

                final_dic['vendor_name']=vendor_name
                final_dic['product_name']=product_name
                final_dic['product_url']=final_url
                final_records.append(final_dic)
                driver.get(url)
                logging.info("Data Loaded Succssefully for Page:
%s".format(j))

            except Exception as e:
                logging.info("Exception for page:%s".format(j))
                logging.info("Exception Message %s".format(str(e)))


ProductName_VendorName_and_URL_mapper(70,99)
df=pd.DataFrame(final_records)
df.to_csv("tableE_data_70to98.csv",index=None)
```

textanalysis.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import nltk
from wordcloud import WordCloud

#nltk.download('stopwords')
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import LancasterStemmer, WordNetLemmatizer, PorterStemmer
from nltk.featstruct import FeatStructReader

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import NMF
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, plot_confusion_matrix

from nrclex import NRCLex
from textblob import TextBlob
```

```
from nltk import word_tokenize, pos_tag, ne_chunk
from nltk.chunk import conlltags2tree, tree2conlltags

#Loading Dataset
ta_df =
pd.read_csv(r'C:\Users\budme\Desktop\Assignments\5193_ProgrammingForDS\cyb
er_threat_analysis_v2\Deliverable2Work\cyber_threat_analysis-
branch_rithik\cwedefinitions_text_consolidated.csv',index_col=
None,header=0)

#Data Cleaning
#Converting the upper case to lower case for the column Description
ta_df['Description'] = ta_df['Description'].apply(lambda x: "
".join(x.lower() for x in x.split()))

#Removing numbers for the column Description
digitspattern = '\\b[0-9]+\\b'
ta_df['Description'] = ta_df['Description'].str.replace(digitspattern,'')

#Removing punctuation and space for the column Description
puncpattern = '[^\w\s]'
ta_df['Description'] = ta_df['Description'].str.replace(puncpattern,'')

#Remove stop words
stop = stopwords.words('english')
ta_df['Description'] = ta_df['Description'].apply(lambda x: " ".join(x for
x in x.split() if x not in stop))

#Stemming the words
porstem = PorterStemmer()
ta_df['Description_Stemmed'] = ta_df['Description'].apply(lambda x: "
".join([porstem.stem(word) for word in x.split()]))


################################ TOPIC MODELLING
##################################################


vectorizer = CountVectorizer(max_df=0.8, min_df=4, stop_words='english')
term_matrix =
vectorizer.fit_transform(ta_df['Description_Stemmed'].values.astype('U'))
term_matrix.shape
LDA = LatentDirichletAllocation(n_components=4, random_state=25)
LDA.fit(term_matrix)

for i,topic in enumerate(LDA.components_):
    print(f'Top 10 words for topic #{i}:')
    print([vectorizer.get_feature_names()[i] for i in topic.argsort()[-
10:]])
    print('\n')
```

```
topic_values = LDA.transform(term_matrix)
ta_df['topic'] = topic_values.argmax(axis=1)

#Distribution of Topic with no of records
ta_df['topic'].value_counts().plot(kind="bar")
plt.title("Frequency of number of records per topic")
plt.xlabel("Topic Number")
plt.ylabel("No. of Records")
plt.show()

################################### Emotion/Feeling Analysis
#####################################

list_of_text = ta_df['Description'].values.tolist()

trust=[]
fear=[]
joy=[]
sadness=[]
positive=[]
negative=[]

for i in range(len(list_of_text)):
    emotion = NRCLex(list_of_text[i])
    trust.append(emotion.affect_frequencies['trust'])
    fear.append(emotion.affect_frequencies['fear'])
    joy.append(emotion.affect_frequencies['joy'])
    sadness.append(emotion.affect_frequencies['sadness'])
    joy.append(emotion.affect_frequencies['positive'])
    joy.append(emotion.affect_frequencies['negative'])

trust_freq=[]
fear_freq=[]

joy_freq=[]
sadness_freq=[]

pos_freq=[]
neg_freq=[]

for i in range(len(list_of_text)):
    wordlist=list_of_text[i].split()

    for word in wordlist:
        word_emotion=NRCLex(word)

        trust_value=word_emotion.affect_frequencies['trust']
        fear_value=word_emotion.affect_frequencies['fear']

        joy_value=word_emotion.affect_frequencies['joy']
        sadness_value=word_emotion.affect_frequencies['sadness']
```

```
        positive_value=word_emotion.affect_frequencies['positive']
        negative_value=word_emotion.affect_frequencies['negative']

        if trust_value>0:
            trust_freq.append(word)
        if fear_value >0:
            fear_freq.append(word)

        if joy_value > 0:
            joy_freq.append(word)
        if sadness_value > 0:
            sadness_freq.append(word)

        if positive_value > 0:
            pos_freq.append(word)
        if negative_value > 0:
            neg_freq.append(word)

fear_df=pd.DataFrame()
fear_df['fear']=fear_freq
fear_df['fear'].value_counts().head(10).plot(kind="bar")
plt.show()

#WordCloud
fearcloud=WordCloud(max_words=50,background_color='black',contour_color='b
lack').generate(' '.join(fear_freq))
plt.imshow(fearcloud,interpolation='bilinear')
plt.axis("off")
plt.show()

fearcloud=WordCloud(max_words=50,background_color='black',contour_color='b
lack').generate(' '.join(trust_freq))
plt.imshow(fearcloud,interpolation='bilinear')
plt.axis("off")
plt.show()

fearcloud=WordCloud(max_words=50,background_color='black',contour_color='b
lack',collocations=False).generate(' '.join(pos_freq))
plt.imshow(fearcloud,interpolation='bilinear')
plt.axis("off")
plt.show()

fearcloud=WordCloud(max_words=50,background_color='black',contour_color='b
lack',collocations=False).generate(' '.join(neg_freq))
plt.imshow(fearcloud,interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
############################################## Classification Model
####################################
Description_Stemmed=ta_df['Description_Stemmed'].values.tolist()

polarity_values=[]

for sentence in Description_Stemmed:
    polarity_value= TextBlob(sentence).sentiment.polarity

    if polarity_value==0:
        polarity_text_value="neutral"
    elif polarity_value > 0:
        polarity_text_value="positive"
    else:
        polarity_text_value="negative"

    polarity_values.append(polarity_text_value)

sentimentanalysis_df=pd.DataFrame()
sentimentanalysis_df['Description_Stemmed']=Description_Stemmed
sentimentanalysis_df['Sentiment']=polarity_values

sentimentanalysis_df['Sentiment'].value_counts().plot(kind='bar')
plt.xlabel("Sentiment")
plt.ylabel("No. of Records")
plt.title("Distribution of records based on sentiment")
plt.show()

features = sentimentanalysis_df['Description_Stemmed']
vectorizer = TfidfVectorizer(max_features=2500, min_df=7, max_df=0.8,
stop_words=stop)
processed_features = vectorizer.fit_transform(features).toarray()
labels = sentimentanalysis_df['Sentiment']
X_train, X_test, y_train, y_test = train_test_split(processed_features,
labels, test_size=0.25, random_state=0)

text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)
predictions = text_classifier.predict(X_test)
cm = confusion_matrix(y_test,predictions)
print(cm)
plot_confusion_matrix(text_classifier, X_test, y_test)
plt.show()

print(classification_report(y_test,predictions))



############################################ NAMED ENTITY RECOGNITION
####################################
ner_df = ta_df
ner_df['NN'] = ''
```

```
ner_df['JJ'] = ''
ner_df['VB'] = ''
ner_df['GEO'] = ''


def desc_ner(chunker):
    treestruct = ne_chunk(pos_tag(word_tokenize(chunker)))
    entityp = []
    entityo = []
    entityg = []
    entitydesc = []

    for x in str(treestruct).split('\n'):
        if 'PERSON' in x:
            entityp.append(x)
        elif 'ORGANIZATION' in x:
            entityo.append(x)
        elif 'GPE' in x or 'GSP' in x:
            entityg.append(x)
        elif '/NN' in x:
            entitydesc.append(x)

    stringp = ''.join(entityp)
    stringo = ''.join(entityo)
    stringg = ''.join(entityg)
    stringdesc = ''.join(entitydesc)
    return stringp, stringo, stringg, stringdesc



i = 0
for x in ner_df['Description']:
    entitycontainer = desc_ner(x)
    ner_df.at[i,'PERSON'] = entitycontainer[0]
    ner_df.at[i,'ORGANIZATION'] = entitycontainer[1]
    ner_df.at[i,'GEO'] = entitycontainer[2]
    ner_df.at[i,'NOUN'] = entitycontainer[3]
    i += 1


person=ner_df['PERSON'].tolist()
organization=ner_df['ORGANIZATION'].tolist()
geo=ner_df['GEO'].tolist()
```

cwe_definition_scraping1.py

```
#Loading Packages
from bs4 import BeautifulSoup
import requests
import logging
```

```
import random
import time
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common import keys
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import random
import os

#Changing the Working Directory
os.chdir(r'C:\Users\budme\Desktop\Assignments\5193_ProgrammingForDS\cyber_
threat_analysis_v2')

#LogFile Configuration to record logs.
logging.basicConfig(filename='logfile.log', filemode='w',
format='%(asctime)s - %(message)s', level=logging.INFO)

#Initialize the chromedriver selenium.
driver=webdriver.Firefox(executable_path=r'C:\Users\budme\Documents\GitHub
\geckodriver.exe')
time.sleep(5)

#Dictionary which contains the ProductName and URL
final_records=[]

def ProductName_VendorName_and_URL_mapper(startpage,endpage):

    #Avoid Continuous Hits on webpage.
    for i in range(startpage,endpage):
        if i%2==0:
            pass
            #time.sleep(random.randint(1,2))
        else:
            time.sleep(random.randint(1,2))

        url="https://www.cvedetails.com/cwe-
definitions/{}/cwelist.html?order=2&trc=668&sha=0427874cc45423ccb6974ee259
35fbfceac76fcb".format(i)

        driver.get(url)

        for j in range(2,52):
            try:
                final_dic={}
                xpath='/html/body/table/tbody/tr[2]/td[2]/div/table/tbody/
tr/td[1]/table/tbody/tr[{}]/td[1]/a'.format(j)
                vendor_link=driver.find_element_by_xpath(xpath)
                vendor_link.click()
                final_url=driver.current_url

                cwenumber_name=driver.find_element_by_xpath('/html/body/ta
ble/tbody/tr[2]/td[2]/div/h1')
```

```
                cwenumber_name=cwenumber_name.text


                final_dic['cwenumber_name']=cwenumber_name
                final_dic['product_url']=final_url
                final_records.append(final_dic)
                driver.get(url)
                logging.info("Data Loaded Succssefully for Page:
%s".format(j))

            except Exception as e:
                logging.info("Exception for page:%s".format(j))
                logging.info("Exception Message %s".format(str(e)))


ProductName_VendorName_and_URL_mapper(1,15)
df=pd.DataFrame(final_records)
df.to_csv("cwedefinitions.csv",index=None)
```

cwe_definition_scraping2.py

```
from numpy import product
import pandas as pd
import requests
import concurrent.futures
import os
import re

#Loading the Table.
mapping_table=pd.read_csv(r'C:\Users\budme\Desktop\Assignments\5193_Progra
mmingForDS\cyber_threat_analysis_v2\text\cwedefinitions.csv')
mapping_table.columns
mapping_table_nrow=mapping_table.shape[0]
mapping_table.head()

#List of Products and Links
def extract_tables(url,i):

    header = {
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/50.0.2661.75 Safari/537.36",
    "X-Requested-With": "XMLHttpRequest"
    }
    r = requests.get(url, headers=header)
    dfs = pd.read_html(r.text)

    mydataframe=dfs[5].T
```

```
    output_name=str(i)+".csv"
    output_path='C:/Users/budme/Desktop/Assignments/5193_ProgrammingForDS/
cyber_threat_analysis_v2/text/tables/'+output_name
    mydataframe.to_csv(output_path,index=None)


list_of_links=mapping_table['product_url'].values.tolist()

for i in range(0,len(list_of_links)):
    try:
        url=list_of_links[i]
        extract_tables(url,i)
    except Exception as e:
        print(i,str(e))
```

final_consolidation.py

```
#loading packages
from typing import final
import pandas as pd
import os
import glob
import re

#change the working directory to save the intermediate files
os.chdir(r'C:\Users\budme\Desktop\Assignments\5193_ProgrammingForDS\cyber_
threat_analysis_v2\Deliverable2Work\Consolidated Data_after getting full
Vendor Data')

#Load Products table
products_dataframe=pd.read_csv(r'products_consolidated.csv',index_col=None
)

#Load Vendors table
vendors_dataframe=pd.read_csv(r'vendors_consolidated.csv',index_col=None)

#Merged the Products and Vendors table on column: VendorName and create
staging table.
joined_dataframe=pd.merge(products_dataframe,vendors_dataframe,on="VendorN
ame",how="left")
joined_dataframe.to_csv("staging1.csv",index=False)

#Below Function Merges the staging tables
def merging_tables(staging1,staging2):

    #Remove trailing and Leading Spaces from the column name: ProductName
    staging1['ProductName']=staging1[['ProductName']].applymap(str.strip)
    staging2['product_name']=staging2[['product_name']].applymap(str.strip
)
    staging1['VendorName']=staging1[['VendorName']].applymap(str.strip)
    staging2['vendor_name']=staging2[['vendor_name']].applymap(str.strip)
```

```python
    #Both the dataframes are merged on column ProductName.
    staging3=pd.merge(staging1,staging2,how="left",left_on=['ProductName',
'VendorName'],right_on=['product_name','vendor_name'])

    #creating final consolidation file.
    staging3.to_csv("staging3.csv",index=None)

#load staging1 table
staging1=pd.read_csv(r'staging1.csv',index_col=None)

#loading staging2 table
staging2=pd.read_csv(r'staging2.csv',index_col=None)

#Merging Tables
merging_tables(staging1,staging2)

####### Loading DataFrame #############
final_df=pd.read_csv(r'staging3.csv')

#Drop Columns
final_df.drop('Year',axis=1,inplace=True)
final_df.drop('Patches',axis=1,inplace=True)
final_df.drop('Compliance',axis=1,inplace=True)
final_df.drop('Inventory',axis=1,inplace=True)

#Rename Column
final_df.rename(columns={'Vulnerabilities_y':'Total_No_of_Vulnerabilities_
by_Vendor'},inplace=True)

#Filter Column
final_df=final_df[final_df['Total_No_of_Vulnerabilities_by_Vendor']>0]

#Data Transformation
final_df['contribution_of_product_vulnerability_to_vendor']=final_df['# of
Vulnerabilities']/final_df['Total_No_of_Vulnerabilities_by_Vendor']

final_df.to_csv("final_consolidated_data.csv",index=None)


####################### Analysis - Story 1  #######################
from matplotlib import pyplot as plt
final_dataframe=pd.read_csv(r'final_consolidated_data.csv')
top_20products=final_dataframe[final_dataframe['ProductType']=="Applicatio
n"].sort_values(by='# of
Vulnerabilities',ascending=False)[['ProductName','# of
Vulnerabilities','VendorName']].head(20)
top_20products.set_index('ProductName').plot(kind="bar")
plt.title("Top 20 products w.r.t no. of vulnerabilities")
plt.xlabel('ProductName')
plt.ylabel('Number of Vulnerabilities')
```

```
plt.show()

######################################## Analysis Story 2
##########################################
filtered_columns=['DoS','Code Execution','Overflow','Memory
Corruption','Sql Injection','XSS','Directory Traversal','Http Response
Splitting',
'Bypass something','Gain Information','Gain Privileges','CSRF','File
Inclusion']
vulnerabilities_df=final_dataframe[filtered_columns]
vulnerabilities_df.sum().sort_values(ascending=False).plot(kind="bar")
plt.title("Overall Vulnerability Statistics")
plt.xlabel('Vulnerability Type')
plt.ylabel('Number of Vulnerabilities')
plt.show()

### Number of vulnerabilities by number of products ###
vulnerabilities_by_products=final_dataframe[['VendorName','Products','Tota
l_No_of_Vulnerabilities_by_Vendor']]
vulnerabilities_by_products['avg_number_of_vulnerabilities_per_product']=v
ulnerabilities_by_products['Total_No_of_Vulnerabilities_by_Vendor']/vulner
abilities_by_products['Products']
vulnerabilities_by_products.drop_duplicates(inplace=True)
vulnerabilities_by_products

########### story 4 ################
final_dataframe[final_dataframe['ProductType']=="Application"]['VendorName
'].value_counts().sort_values(ascending=False).head(20).plot(kind="bar")
plt.title("Top 20 Vendors w.r.t no of application products")
plt.xlabel('Vendor Name')
plt.ylabel('Number of Products')
plt.show()


final_dataframe.columns
final_dataframe[final_dataframe['ProductType']=="Application"][['VendorNam
e','Total_No_of_Vulnerabilities_by_Vendor']].set_index('VendorName').sort_
values(ascending=False,by='Total_No_of_Vulnerabilities_by_Vendor').drop_du
plicates().head(20).plot(kind="bar")
plt.title("Top 20 Vendors w.r.t no of Vulnerabilities")
plt.xlabel('Vendor Name')
plt.ylabel('Total No. of Vulnerabilities')
plt.show()


final_dataframe['vulnerabilities_to_product_ratio']=final_dataframe['Total
_No_of_Vulnerabilities_by_Vendor']/final_dataframe['Products']
final_dataframe[final_dataframe['ProductType']=="Application"][['VendorNam
e','vulnerabilities_to_product_ratio']].drop_duplicates().set_index('Vendo
rName').sort_values(by="vulnerabilities_to_product_ratio",ascending=False)
.head(20).plot(kind="bar")
plt.title("Top 20 vendors w.r.t vulnerabilities to product ratio")
plt.xlabel('Vendor Name')
```

```
plt.ylabel("Vulnerabilities to product ratio")
plt.show()
```

vulnerabilities_by_product_table_extraction.py

```
from numpy import product
import pandas as pd
import requests
import concurrent.futures
import os
import re

#Loading the Table.
mapping_table=pd.read_csv(r'C:\Users\budme\Desktop\Assignments\5193_ProgrammingForDS\cyber_threat_analysis_v2\tableB_data_1to56.csv')
mapping_table.columns
mapping_table_nrow=mapping_table.shape[0]
mapping_table.head()

def extract_tables(url):

    header = {
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.75 Safari/537.36",
    "X-Requested-With": "XMLHttpRequest"
    }
    r = requests.get(url, headers=header)
    dfs = pd.read_html(r.text)

    filter_df=mapping_table[mapping_table['product_url']==url]
    vendor_name=filter_df['vendor_name'].values.tolist()[0]
    product_name=filter_df['product_name'].values.tolist()[0]

    vendor_name_cleaned=re.sub(r'[\/\\\|\? ]',"_",vendor_name)
    product_name_cleaned=re.sub(r'[\/\\\|\? ]',"_",product_name)
    filename=product_name_cleaned+vendor_name_cleaned

    mydataframe=dfs[5]
    mydataframe['product_name']=[product_name]*mydataframe.shape[0]
    mydataframe['vendor_name']=[vendor_name]*mydataframe.shape[0]

    output_name=filename+".csv"
    output_path='C:/Users/budme/Desktop/Assignments/5193_ProgrammingForDS/tableextraction/B/'+output_name
    mydataframe.to_csv(output_path,index=None)


list_of_links=mapping_table['product_url'].values.tolist()


for i in list_of_links:
    try:
```

```
        extract_tables(i)
    except Exception as e:
        print(i,str(e))
```