VULNERABILITY ANALYSIS USING UNSUPERVISED MACHINE LEARNING METHODS

Submitted By

K. MEGHA SREE (20PIT005)

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

INTRODUCTION

Even ordinary people are now bare to software. Millions of individuals around the sphere use smart devices. Everything is software-based. As per the NIST study, the quantity of IT weaknesses has been expanding throughout the course of recent years. Vulnerabilities allow an attacker to execute code or gain access to the memory of a target machine.This project's purpose is to investigate common vulnerabilities and exposures (CVEs).While there are other vulnerability databases, we'll concentrate on the National Vulnerability Database because it's the most widely utilised (NVD). It is the industry standard database perfectly capture all vulnerabilities revealed publicly.

This analysis emphasizes the extent and tries to impose the vulnerability, and how severe each type is by analyzing it. Data analysis and PCA (Principal Component Analysis) which is a dimensionality reduction technique, and via clustering algorithms such as K-means, K-medoids, and agglomerative hierarchical clustering, are used to do this. This analysis is used to determine the severity of vulnerabilities discovered in the dataset. This project's contribution is the analysis of the severity of vulnerabilities in the dataset and the identification of the top most serious vulnerabilities.

As a result, we suggested a vulnerability analysis model based on clustering techniques in order to analyse vulnerabilities in a very effective and fast manner and to assist organisations in overcoming all of these challenges. Finally, the major goal of this study is to employ clustering to analyse vulnerabilities. The analysis was carried out with the help of a dataset provided by the US government. First and foremost, preprocessing is carried out. Clustering is a machine learning technique that is being employed. Following the evaluation, the performance of clustering methods was compared, and a result was reached. As a result, performance evaluation is a critical component of this project in order to achieve the best outcome across all clustering models.

ABSTRACT

Source code vulnerability is a weakness or a glitch in script used for software development purpose that make a way for an attacker to enter inside a network or system of an individual or a company. The broad usage of software projects has resulted in the possibility of emerging vulnerabilities and potential consequences for their exploits.

Existing code analysis methods are ineffectual at identifying vulnerabilities. This project investigates and presents vulnerabilities, particularly in source code. Vulnerabilities paves a way to businesses and individuals approachable to various kinds like malware and account takeovers.

Vulnerability analysis affords an organisation with the essential information, awareness, and risk background it needs to recognise and respond to threats to its environment. The project's intention is to execute a vulnerability analysis and tool framework.

A complete vulnerability evaluation can assist companies to enhance the safety of their structures. Vulnerability analysis also offers detailed steps for revealing current flaws and preventing future assaults. The analysis can also help improve your company's reputation and goodwill, inspiring greater confidence among customers. It can also assist in safeguarding the integrity of assets in the event of any malicious code being concealed in any of said assets.

The proposed framework consists of five phases, including data acquisition, data preprocessing, feature selection, model building (unsupervised machine learning models) and performance evaluation.According to the Positive Technologies report 2020, 31% of companies dredged endeavor to impose source code vulnerabilities; nearly one-third of discovered risks accommodate software exploit shots.

In this project, vulnerability analysis was done with unsupervised machine learning method using clustering techniques. Examining the vulnerabilities, especially in source code, is done and presented in this project. There are a variety of frequently used techniques, but clustering is the most appropriate. This algorithm focuses on identifying groups of data according to similarities. Hence, the method of clustering allows the data to form clusters. The fact that this is an unsupervised problem with no target class is one of the main reasons for its usage. As a result of the analysis, we are able to equip firms with awareness and knowledge in order to secure their products from becoming vulnerable.

Keywords: Vulnerability, Analysis, K-Means, Silhouette, Clustering

METHODOLOGY

4.1 Dataset Description

Dataset/source: The source of the dataset is fromNational Vulnerability Database (NVD).

The overall methodology for the vulnerability analysis is depicted in **Figure 4.1**.

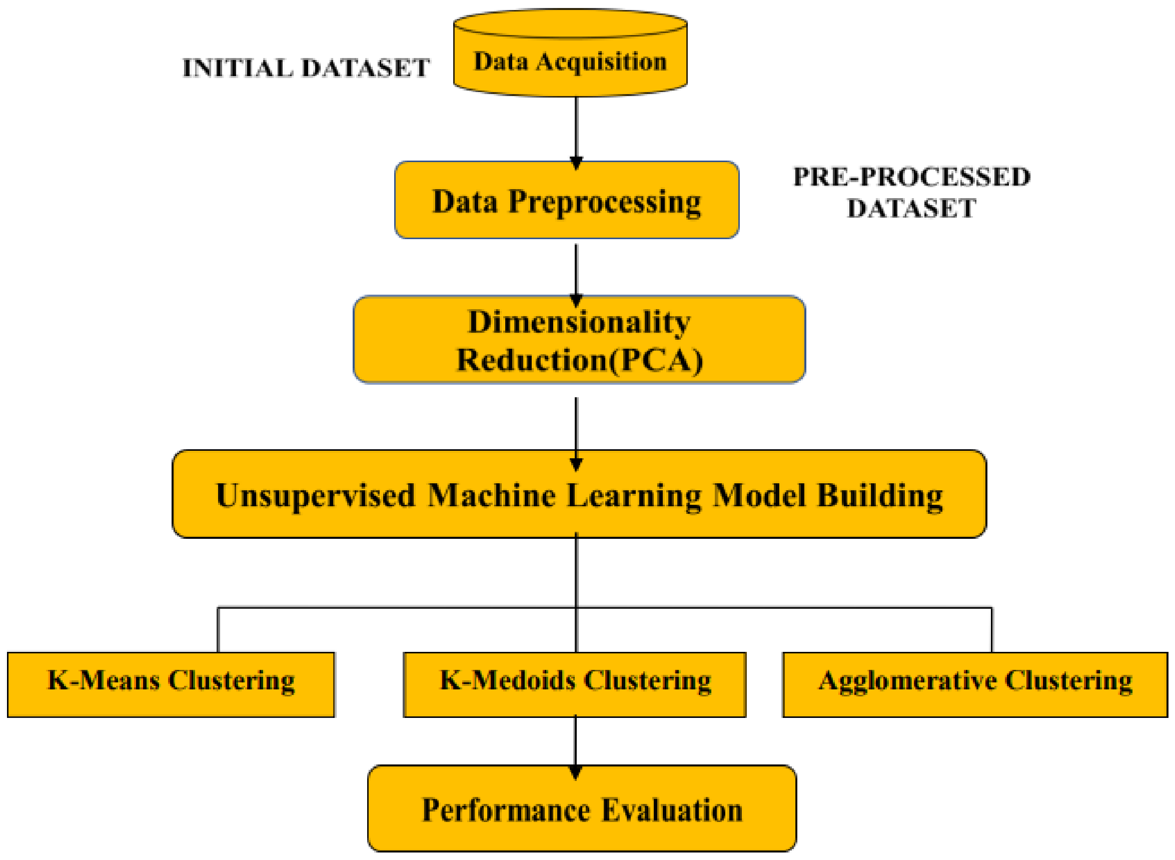


Figure 4.1: Methodology

Tasks performed:

Task 1: Load a dataset from file.

Task 2: Pre-Process a dataset by removing the inessential data and label encoding it.

Task 3: Reduce the elements of the information utilizing Principal Component Analysis (PCA).

Task 4: Applying Clustering calculations like K-Means, K-Medoids and Agglomerative Hierarchical Clustering.

Task 5: Finally, assess its exhibition utilizing Silhouette coefficient and Davies-Bouldin Index

RESULTS AND DISCUSSION

This undertaking depends on the weaknesses gathered from the National Vulnerability Database (NVD), which is a data set of weaknesses. NVD stores weaknesses with CVE values, which incorporate weakness, shortcoming, and openness definitions. The uniqueness of the issues is safeguarded by utilizing one-of-a-kind identifiers (CVE numbers) from the CVE list. CVSS is used to calculate the severity of each vulnerability mentioned in NVD (Common Vulnerability Scoring System).The performance of this project can be done in five phases:

Phase 1: Data Acquisition

Data acquisition is the foundation for the project and most significant step. This is used to collect data from relevant sources before it can be stored, cleaned, pre-processed, and used for further mechanisms. Once acquired the appropriate data, it’s time to prepare it. The accuracy of your model is determined by the quality of the data you provide the machine. The data in this study comes from the National Vulnerability Database (NVD), which is kept up with by NIST.

- The vulnerabilities detected in software products are represented by the samples in the dataset.
- So, in machine learning terms, we're dealing with an unsupervised learning problem.
- There are numerous approaches to creating such a model, but we will focus on clustering techniques.

The National Vulnerability Database (NVD) records C/C++ code defects with labeled Common Vulnerabilities and Exposures (CVEs). For the NVD, announced weaknesses are investigated and included a normalized design. In particular, a dataset passage contains the accompanying:

1) A Common Vulnerability Exposure (CVE) ID number that extraordinarily distinguishes the weakness.
2) The weakness passage's distribution date.
3) The weakness passage's adjusted date.
4) The weakness type/classification, as grouped by the Common Weakness Enumeration (CWE) code and the name related with the CWE code.
5) The name of the vulnerability in the CWE name is related to the CWE code.
6) The CVSS Score is specifically designed for assessing the severity of vulnerability.
7) The summary provides a description of the vulnerability.

ATTRIBUTE	DTYPE	DESCRIPTION
mod_date	Datetime	The date the section was last adjusted
pub_date	Datetime	The date the passage was distributed
Cvss	Float	Normal Vulnerability Scoring System (CVSS) score, a proportion of the seriousness of a weakness
cwe_code	Categorical	Common Weakness Enumeration (CWE) code, identifying the type of weakness
cwe_name	Categorical	The name associated with the CWE code
Summary	Str	A text summary of the vulnerability

access_authentication	Categorical	This measurement measures the times an assailant should verify to an objective to take advantage of weakness.
access_complexity	Categorical	This measurement mirrors the intricacy of the assault expected to take advantage of the weakness.
access_vector	Categorical	This metric reflects on how the vulnerability can be exploiting. The more remote vulnerability can be exploited the higher the rating.
impact_availability	Categorical	Accessibility alludes to on how data assets can be access. This measurement reflects accessibility of an effectively taken advantage of weakness.
impact_confidentiality	Categorical	This metric reflects information confidentiality of a successfully exploited vulnerability.
impact_integrity	Categorical	Integrity refers to the trustworthiness of the information. This metrics reflects integrity of a successfully exploited vulnerability.

Table 4.1: Attributes and its description

The CVSS Score is explicitly intended for surveying weakness to decide the seriousness of the weakness. Score goes from 1 to 10, with the issues having a score of 10 being the most extreme and the ones having a score of 1 being the most un-serious. **Table 4.2** shows the CVSS seriousness level edges.

Label	Score
Low	0.0–3.9
Medium	4.0–6.9
High	7.0–8.9
Critical	9.0–10.0

Table 4.2: Common Vulnerability Scoring System (CVSS)

NVD involves the CVSS standard for rating seriousness. The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for evaluating the seriousness of PC framework security weaknesses. CVSS endeavors to appoint weakness seriousness scores, permitting responders to focus on reactions and assets in view of danger. Each CVSS is made out of three measurement gatherings: Base, Temporal, and Environmental, each comprising of a bunch of measurements.

The Base Metrics

The Base Metrics are made up of variety of components. **Table 4.3** lists these components and their descriptions.

S.No	Base Metrics Component	Explanation	Metric Value & Description
1	**Access Vector (AV)**	This measurement considers how the weakness can take advantage of. The more far off weakness can be taken advantage of the greater the rating.	**Network**: Can be exploit from a distance. **Adjacent**: Can be take advantage of remotely

			yet restricted to the equivalent physical or legitimate organization. **Local**: Not expectorganization to take advantage of the weakness. It additionally can be taking advantage of by actually admittance to weakness.
2	**Access Complexity (AC)**	This measurement mirrors the intricacy of the assault expected to take advantage of the weakness.	**Low**: Specialized condition doesn't exist and an aggressor can anticipate repeatable accomplishment againstthe weak part. **Medium**: Specialized condition to some degree exist and an assailant should spenda measure of work to exploit theweak part. **High**: Specialized that's what condition exist make aggressor should spenda measure of work to exploit theweak part.
3	**Access Authentication (Au)**	This measurement estimates the times an aggressor should validate to an objective to take advantage of weakness.	**None**: Authentication does not expected to take advantage of weakness. **Single**: Authentication expect to take advantage of weakness. **Multiple**: Authentication required at least two in request to take advantage of weakness.

4	**Impact Confidentiality (IC)**	This measurement reflects data classification of an effectively taken advantage of weakness.	**None**: There is no deficiency of data privacy on the framework. **Partial**: There is some deficiency of data classification. **High**: There is all out loss of data classification.
5	**Impact Integrity (II)**	Honesty alludes to the dependability of the data. This measurement reflects honesty of an effectively taken advantage of weakness.	**None**: Modification of framework records is incomprehensible. **Partial**: Modification of some framework records is conceivable. **High**: Modification of whole framework records is conceivable.
6	**Impact Availability (IA)**	Accessibility alludes to on how data assets can be access. This measurement reflects accessibility of an effectively taken advantage of weakness.	**None**: Performance and assets doesn't influence. **Partial**: Modification of some system records is possible. **Complete**: Performance and assets can be control effectively by assailant.

Table 4.3: Base Metric Components

Phase 2: Data Pre-Processing

The data set must be inspected, with data removed and altered as needed. The information is then completely dissected to furnish us with understanding into the informational collection and permit us to decipher the information accurately. Following that, we use PCA to perform dimensionality reduction. We bunch our information utilizing k-means, K-medoids, and the

AHC calculation with a lower aspect. We put our method to the test with clusters by using a silhouette coefficient and the Davies-Bouldin Score. After generating the dataset, it must be cleaned up, i.e., the irrelevant information must be operated.

Originally, all null (NA) rows were eliminated. Categorical variables are present in the dataset. So, I'd like to convert this to numerical form because the machine cannot process categorical variables to produce results. To convert categorical variables to numeric form, label encoding was utilised. The EDA (Exploratory Data Analysis) method was then used to examine the dataset at a high level. The severity distribution of vulnerabilities was investigated using the CVSS score. To gain a complete grasp of the data, each attribute value is reviewed in depth.

First, clean up the data set by deleting any extraneous data and making any necessary changes.The informational collection is shown and the highlights are analyzed in this review.

Phase 3: Dimensionality Reduction (PCA)

K-means, K-medoids, and Agglomerative Hierarchical Clustering algorithms were used to cluster unsupervised data. Cluster formation requires a minimum of two samples.K-means was applied to all of the Principal Component Analysis projections, yielding a most extreme difference with a dimensionality decrease of 2.As a result, the Clusters created utilizing the K-means Algorithm with the Principal Components aspect brought down to 2 were considered awesome. This calculation expects to keep data of interest in the cluster as near one another as conceivable by keeping the number of squared distances between the point and the group's centroid as little as could be expected. The amount of all distances between useful pieces of information and centroids is then processed by the K-means calculation, and every information point is allocated to the closest centroid.To determine the best potential cluster, K-means method uses the Expectation-Maximization approach.

We created clusters based on the cve name and CVSS using K-means. The Primary Components Analysis (PCA) technique was then used to extract principal components. Novel variations are developed using principal component analysis and k-means clustering, and data from the National Vulnerability Database (NVD) is utilised.While settling on the quantity of primary parts, PCA utilized a 95 percent least fluctuation as a measure. To decrease the commotion, 5% of the variety is forgotten about. Out of the 10 segments in the cleaned informational index, the PCA procedure yielded two head parts. The change dissemination is displayed in **figure 5.1**, and the intensity map grid of the vital parts coefficients, i.e., the connection between the primary parts and the cleaned informational index segments, is displayed in **figure 5.2.**

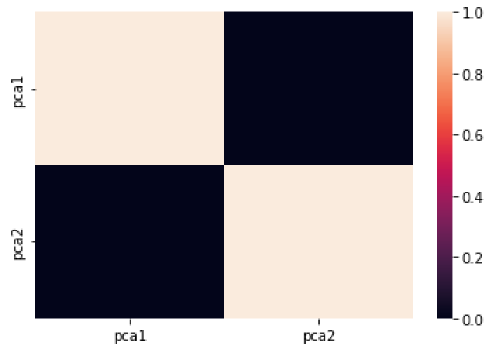


Figure 5.1: Heat map of principal components

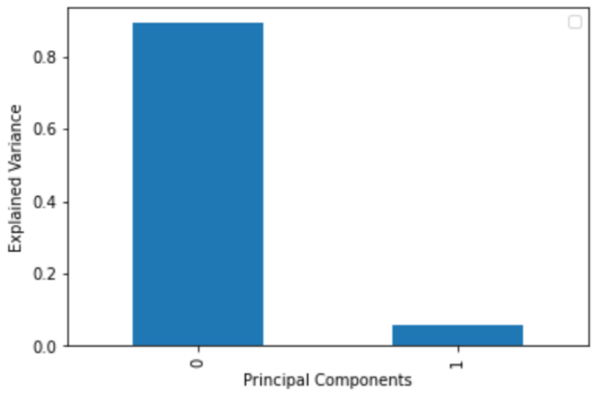


Figure 5.2: Variance in each principal component

Phase 4: Unsupervised Machine Learning Model Building

In this phase, the data is ready for applying algorithms. As it is an unlabelled data, it doesn’t have a target variable so, clustering algorithms is more suitable than other algorithm techniques.As we have an unsupervised issue, clustering is utilized in finding normal gatherings in the component space of information.Clustering works on dataset in which there is no target variable i.e., Unlabelled data. It is more helpful than others as our aim is for analysis purpose, in order to learn more about the problem domain so called pattern discovery or knowledge discovery.Scikit-learn library gives a set-up of various clustering calculations to browse.Among all clustering algorithms, k-means, k-medoids and agglomerative hierarchical clustering were used.

Application of the k-means algorithm

With dataset, the model can be created by initializing the k-means algorithm. But first, an appropriate k needs to be chosen as the k-means algorithm needs this parameter to initialize. The elbow method is a good start for finding the proper k. This elbow method could normally also be used for choosing the number of principal components, In figure 4.3, there is an elbow to choose the number of principal components.

Elbow method

In any unsupervised technique, determining the optimal number of clusters into which the data can be divided is critical. The Elbow Method is one of the most well-known criteria for determining the ideal value of k. The elbow approach is a strategy for finding the best parameter in algorithms like k-means and PCA. The objective is to utilize K-Means to sort out the ideal number of clusters for different group sizes.

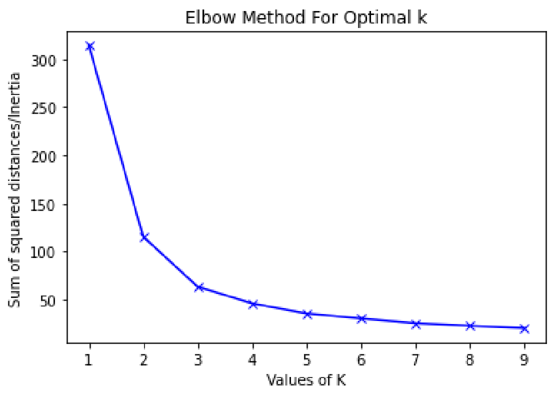


Figure5.3: Elbow method

We initially scaled informational index by normalization so there won't be any predisposition while applying PCA. From that point forward, we applied PCA to decrease the aspects in our informational index. PCA additionally diminished some noise when it leaved out 5% change out of the informational collection.

K-Means Clustering:

The k-means calculation produces bunches. These clusters are grouped by similarities in terms of values for each feature of each data point. This calculation can deliver various results, as this algorithm requires randomization during initialization. Accordingly, the similitude's on which the groups are based might be different for every execution of the calculation. Since the calculation doesn't have any idea what each component implies, how groups are bunched in the way that they are might be wanted or undesirable, expected or unforeseen. It is up to them to determine yet if the clusters are plausible for our given problem, which may necessitate running the grouping calculation on numerous occasions. Accordingly, it is basic to approve the groups created prior to continuing to the subsequent stage. The calculation's means are as per the following:

1. Choose various k clusters to segment n data of interest into.
2. Initialize k cluster centroids arbitrarily by choosing focuses in the space as the data of interest.
3. For every data of interest, process the blunder for each cluster centroid. Appoint the information highlight the group of the cluster centroid with the base mistake.
4. For each cluster, figure the mean of the multitude of useful pieces of information in the group. This implies this is the new bunch centroid.
5. Repeat stages 3 and 4 until the new cluster centroids don't change.

Algorithm: K means Algorithm

Result: Find k clusters using K-means

- ✦ X← {x1, x2, x3...xn} V ← {v1, v2, v3...vk} (a set of centroids chosen at random)
- ✦ Choose k centroids at random.
- ✦ Ascertain the distance between every piece of information while information focuses are reassigned. Appoint information focuses to the centroid with a base distance of
- ✦ Recalculate the new group utilizing: vi = 1 ki Pki j = 1 xi (where ki addresses the quantity of data of interest in the ith cluster)
- ✦ Recalculate the distance between every data of interest and the new centroid end.
- ✦ It shows the clusters acquired utilizing K-means. Examination of our outcomes shows the conveyance of seriousness in various spaces, similar to Memory and Buffer Overflow weaknesses, and Network and Authentication weaknesses.

K-Medoids clustering:

K-Medoids is a grouping calculation that works similarly that K-Means does. The manner in which it picks group focuses shifts fiercely from the K-Means calculation.The former takes the average of a cluster's points as its centre, whereas the latter always chooses the actual data points from the clusters as their centres.Thus, the K-medoids calculation is more commotion lenient than the K-means calculation.

Algorithm

Step1: Initialize k bunches in the given information space D.
Step2: Randomly pick k items from n objects in information and appoint k items to k groups with the end goal that each article is allotted to one and only one bunch. Subsequently, it turns into an underlying medoids for each group.

Step3: For all leftover non-medoid objects, figure the Cost (distance as registered by means of Euclidean, Manhattan, or Chebyshev strategies) from all medoids.

Step4: Now, assign each leftover non-medoid object to that bunch whose medoid distance to that item is least when contrasted with different groups medoid.

Step5: Compute the complete expense for example it is the complete amount of all the non-medoid objects distance from its bunch medoid and appoint it to dj.

Step6: Randomly select a non-medoid object i.

Step7: Now, impermanent trade the article I with medoid j and Repeat Step5 to recalculate complete expense and allocate it to di.

Step8: If di<dj then make the transitory trade in Step7 long-lasting to shape the new arrangement of k medoid. Else fix the brief trade done in Step 7.
Step9: Repeat Step 4, Step 5, Step 6, Step 7, Step 8. Until no change;

Agglomerative Hierarchical Clustering:

In information mining and measurements, progressive grouping investigation is a strategy for bunch examination that looks to fabricate a pecking order of groups i.e., tree-type structure in light of the pecking order. The agglomerative clustering is the most well-known kind of various leveled bunching used to bunch objects in groups in light of their closeness. It's otherwise called AGNES (Agglomerative Nesting). The calculation begins by regarding each item as a singleton bunch. Then, sets of bunches are progressively converged until all groups have been converted into one major group containing all articles. The outcome is a tree-based portrayal of the articles, named dendrogram.

How it works:

1. The procedure begins with determining the dissimilarity between the N objects.
2. Next, two objects that, when clustered together, minimise a specific agglomeration criterion are clustered together, resulting in the creation of a class that includes these two objects.
3. The agglomeration criterion is then used to compute the dissimilarity between this class and the N-2 other objects.
4. The two things or classes of objects whose grouping reduces the agglomeration criterion are subsequently grouped together.
5. Repeat step 4 until all of the objects have been grouped.

Phase 5: Performance Evaluation

In this phase, evaluation was done based on how it performed on applying the algorithms and how well the clusters formed basis. For evaluation, we picked the two most popular methods such as silhouette coefficient, which is most effective for performance evaluation compared to other methods. Another one is Davies-Bouldin index. According to our survey, researchers utilise a variety of evaluation measures to assess the efficacy of various clustering and dimensionality reduction techniques. A selection of commonly used evaluation metrics is presented in **Table 4.4**.

Evaluation Metric	Definition	Formula
Silhouette Coefficient	The Silhouette Coefficient is the most widely recognized method for consolidating the measurements of Cohesion and division in a solitary measure.	$S = \frac{1}{N}\sum_{i=1}^{N} s(i)$ where: **N**: Number of data points in the same cluster, **S(i):** Data point in the

		cluster, i = 1,2, 3…. n,
Davies-Bouldin Index	Davies-Bouldin list is determined as the normal likeness of each bunch with a group generally like it.	$DB \equiv \frac{1}{N}\sum_{i=1}^{N} D_i$ Where: **N**: Total number of clusters **Di**: Similarity measure of each cluster.

Table 4.4: Performance Evaluation Metrics

A crucial element of the clustering data process is evaluating the outcomes of a clustering algorithm.While investigating clustering results, a few angles should be considered for the approval of the calculation results:

- Deciding the clustering propensity in the information.
- Deciding the right number of clusters.
- Evaluating the nature of the clustering results without outer data.
- contrasting the outcomes with outer information; and
- Figuring out which of two arrangements of clusters is prevalent.

Silhouette coefficient

The silhouette coefficient is the most commonly used method for combining cohesion and separation statistics into a single statistic. The three stages for working out the outline coefficient at a particular position are as per the following.The typical distance a(i) between every model and everything different occurrences in a similar cluster is determined:

$$a(i) = \frac{1}{|C_a|} \sum_{j \in C_a, i \neq j} d(i, j)$$

where:

a(i): It is the average distance between i and all the other data points in the cluster to which i belongs. For every model, the base typical distance b(i) between the model and the models contained in each bunch not containing the investigated model:

$$b(i) = \min_{C_b \neq C_a} \frac{1}{|C_b|} \sum_{j \in C_b} d(i, j)$$

where:

b(i): It is the average distance from i to all clusters to which i does not belong. For every model, the outline not entirely set in stone by the accompanying articulation:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where:

s(i): It is the silhouette coefficient of the data point i.

For every model in our informational collection, the outline coefficient is characterized in the stretch [-1, 1]. The worldwide outline coefficient is basically the normal of the outline coefficients for every individual model:

$$S = \frac{1}{n}\sum_{i=1}^{n} s(i)$$

Rather than other consolidated measures, the outline coefficient gives a clear structure to capability. Positive qualities show that groups are very much isolated. Negative qualities show that the groups are combined as one. At the point when the outline coefficient is zero, it shows that the information is conveyed consistently across Euclidean space. Tragically, the outline coefficient's high computational intricacy, O (dn2), makes it unreasonable while managing enormous informational indexes.

This method is used because the ground truth labels are not known; evaluation must be performed using the model itself. This is an example for such evaluation, where a higher silhouette score relates to a model with better clusters. This method is performed with the module Sklearn.metrics.Silhouette_ScoreThis technique is characterized for each example in the cluster and it is made out of two scores. They are,

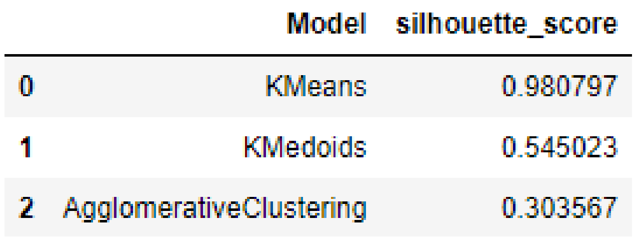
1. The mean distance between an example and different focuses in a similar cluster.
2. The mean distance between an example and any remaining focuses in the following closest cluster.

The silhouette coefficient for a bunch of tests is given as the mean of the outline for each example.

The silhouette score indicates:

- ✦ -1 refers, the clusters are not formed well or it is the indication of incorrect clustering.
- ✦ +1 refers the clusters formed well and it indicates highly dense clustering.
- ✦ 0 refers to the clusters are overlapped.

Through silhouette coefficient, the evaluation results we obtained through k-means is 0.98. which approximately equals to 1. And through k-medoids the score is 0.54. And, through agglomerative clustering we got a score of 0.30.



	Model	silhouette_score
0	KMeans	0.980797
1	KMedoids	0.545023
2	AgglomerativeClustering	0.303567

Figure5.4: Results Obtained through silhouette coefficient

By comparing the three algorithms performance, K-means topped among other two algorithms with a score of 0.98 which indicates it formed highly dense clusters and provides a formation of good clustering. Through the evaluation, we came to a conclusion that, k-means is more suitable algorithm and produce a good result among other two algorithms performed.

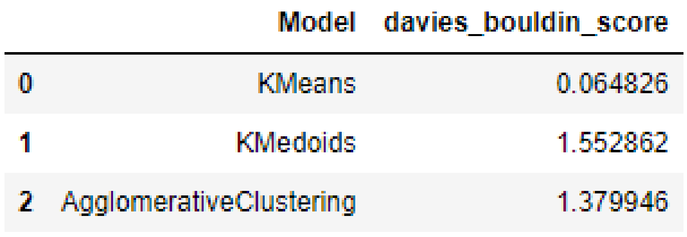
Davies-Bouldin index

A connected inward approval method was proposed in that considers the proportion of intra-cluster dissipate to between bunch detachability across all k gatherings in a clustering. Officially, the DB file is characterized as a component of each group's vicinity to its closest neighbor:

$$DB(\mathcal{C}) = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left\{ \frac{\Delta(C_i) + \Delta(C_j)}{\delta(C_i, C_j)} \right\}$$

As clusters become more conservative and particular, this worth will diminish, making more modest qualities for this list attractive. The DB record has a critical detriment in that it comes up short on fixed range, with a result esteem simply restricted to being non-negative, making understanding troublesome. Moreover, observational proof proposes that while endeavoring to choose k, this file will in general misjudge the quantity of gatherings, particularly for feebly grouped information. DB record can be utilized to assess the model, where lower score connects with a model with better partition between the groups.

0 is the least conceivable score, esteems more like zero demonstrates a superior parcel. Through DB index, the evaluation results we obtained through k-means is 0.064. And through k-medoids the score is 1.55. And, through agglomerative clustering we got a score of 1.37.



	Model	davies_bouldin_score
0	KMeans	0.064826
1	KMedoids	1.552862
2	AgglomerativeClustering	1.379946

Figure 5.5: Results Obtained through Davies-Bouldin Index

By comparing the three algorithms performance, in this also K-means topped among other two algorithms with a score of 0.06 which has a least possible score that indicates the better cluster formation. Through the evaluation, we came to a conclusion that, k-means is more suitable algorithm and produce a good result among other two algorithms performed.

CONCLUSION AND SCOPE FOR FUTURE ENHANCEMENT

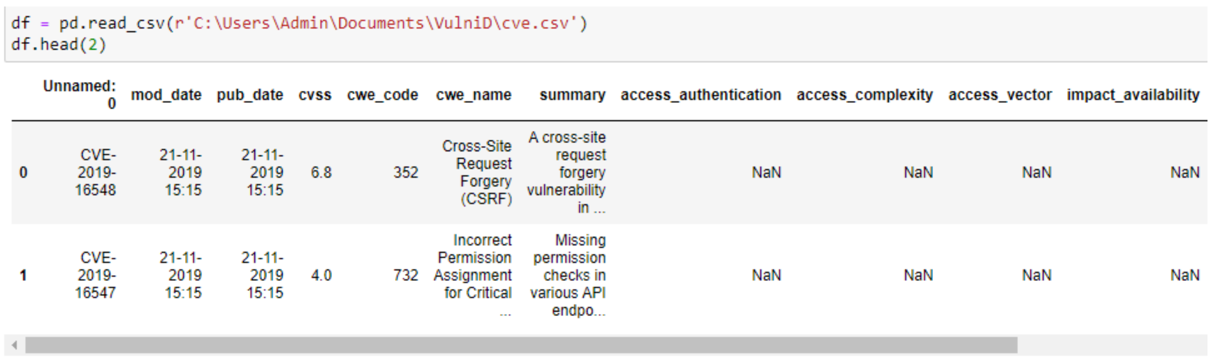
The main intention of this project is to check the quality of the clusters formed through clustering algorithms and it is not an easy problem to solve. The escalating cyber danger was the driving force behind this effort. In this project, we analysed vulnerabilities in source code, we used a few unsupervised clustering techniques. By learning about vulnerabilities in the dataset, this project provides an ideal solution to organisations with the goal of enhancing vulnerability analysis efficacy. As a result, the K-Means algorithm looks to be the most efficient method for analysing vulnerabilities exactly. This project's future enhancements could include a strategy for integrating more and more technologies that will be integrated with new domains in the future to perform automated vulnerability analysis on a large-scale and cross-architecture basis.

REFERENCES

- ∇ Armerding, Taylor. “What Is CVE, Its Definition and Purpose?” CSO Online, CSO, 10 July 2017, www.csoonline.com/article/3204884
- ∇ “About CWE”, Common Weakness Enumeration, September 26, 2007, Available: http://cwe.mitre.org
- ∇ Common Vulnerabilities and Exposures. Available online: https://cve.mitre.org
- ∇ “CWE- Common Weakness Enumeration”, National Vulnerability Database, Available: http://nvd.nist.gov/cwe.cfm
- ∇ “CVSS- A complete Guide to the Common Vulnerability Scoring System Version 2.0”, FIRST: Forum of Incident Response and Security Teams, Available: http://www.first.org/cvss
- ∇ Ghaffarian, S.M.; Shahriari, H.R. Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey. ACM Comput. Surv. 2017, 50, 56. dx.doi.org
- ∇ MITRE, Common Weakness Enumeration. https://cwe.mitre.org
- ∇ National Vulnerability Database. Available online: https://nvd.nist.gov
- ∇ “NVD Common Vulnerability Scoring System Support v2”, National Vulnerability Database, June 20, 2007, Available: http:// nvd.nist.gov
- ∇ “NVD Data Feed and Product Integration”, National Vulnerability Database, Available: http://nvd.nist.gov/download.cfm
- ∇ “OSVDB: The Open Source Vulnerability Database”, OSVDB, Available: http://osvdb.org/
- ∇ S Christey and R. A. Martin, “Vulnerability Type Distributions in CVE”, Common Weakness Enumeration- A Community-Developed Dictionary of Software Weakness Types, May 22, 2007, Available: http://cwe.mitre.org/documents/vuln-trends
- ∇ Y. Y. Chang, P. Zavarsky, R. Ruhl and D Lindskog, “Trend Analysis of Common CVE Vulnerability Types”, Concordia University College of Alberta, May 2011.

ANNEXURE

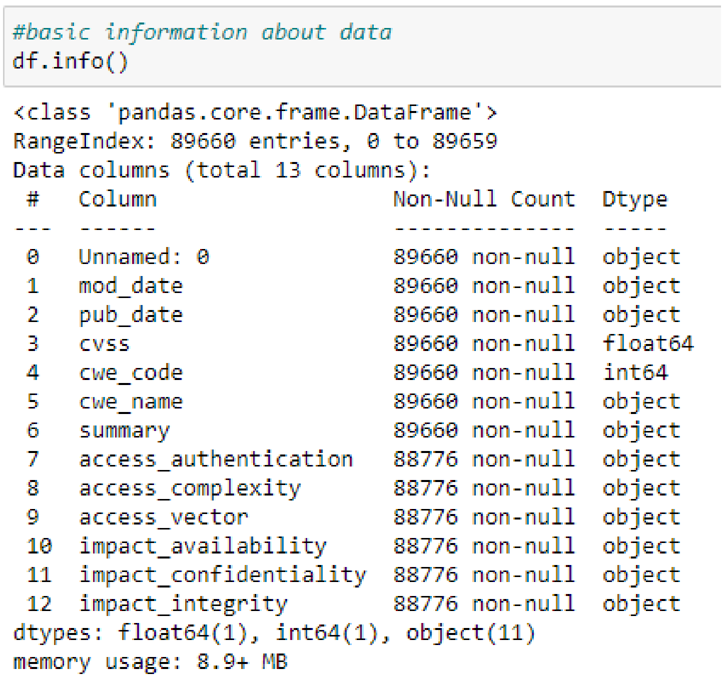
Screenshots



```
df = pd.read_csv(r'C:\Users\Admin\Documents\VulniD\cve.csv')
df.head(2)
```

	Unnamed: 0	mod_date	pub_date	cvss	cwe_code	cwe_name	summary	access_authentication	access_complexity	access_vector	impact_availability
0	CVE-2019-16548	21-11-2019 15:15	21-11-2019 15:15	6.8	352	Cross-Site Request Forgery (CSRF)	A cross-site request forgery vulnerability in ...	NaN	NaN	NaN	NaN
1	CVE-2019-16547	21-11-2019 15:15	21-11-2019 15:15	4.0	732	Incorrect Permission Assignment for Critical ...	Missing permission checks in various API endpo...	NaN	NaN	NaN	NaN

Fig.1: Loading Dataset



```
#basic information about data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89660 entries, 0 to 89659
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Unnamed: 0             89660 non-null  object 
 1   mod_date               89660 non-null  object 
 2   pub_date               89660 non-null  object 
 3   cvss                   89660 non-null  float64
 4   cwe_code               89660 non-null  int64  
 5   cwe_name               89660 non-null  object 
 6   summary                89660 non-null  object 
 7   access_authentication  88776 non-null  object 
 8   access_complexity      88776 non-null  object 
 9   access_vector          88776 non-null  object 
 10  impact_availability    88776 non-null  object 
 11  impact_confidentiality 88776 non-null  object 
 12  impact_integrity       88776 non-null  object 
dtypes: float64(1), int64(1), object(11)
memory usage: 8.9+ MB
```

Fig.2: Attributes Information

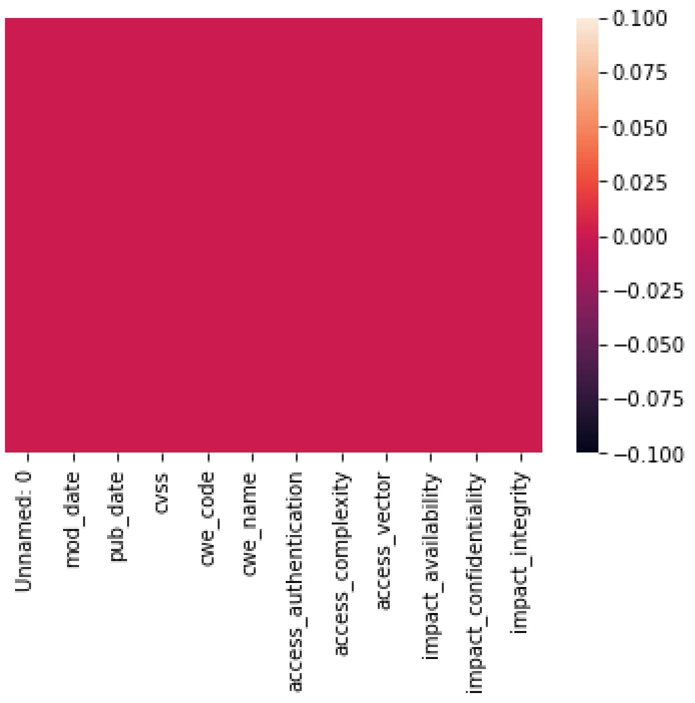


Fig3: Heatmap

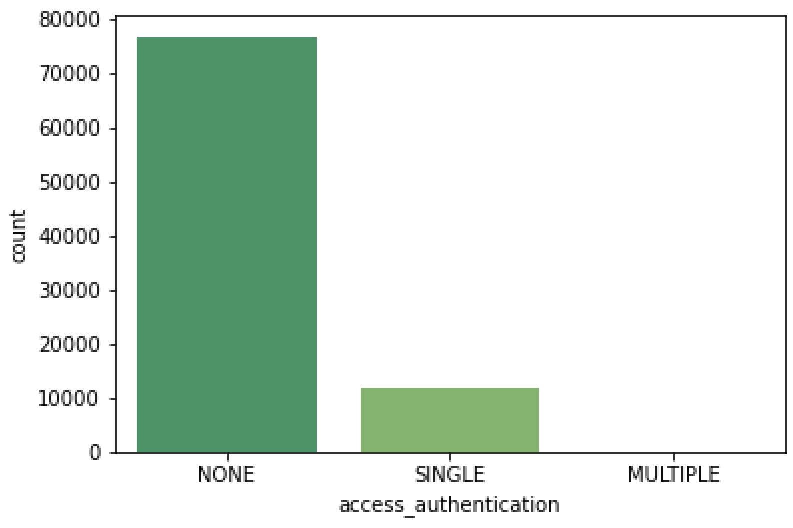


Fig.4: Count Plot for Access_Authentication

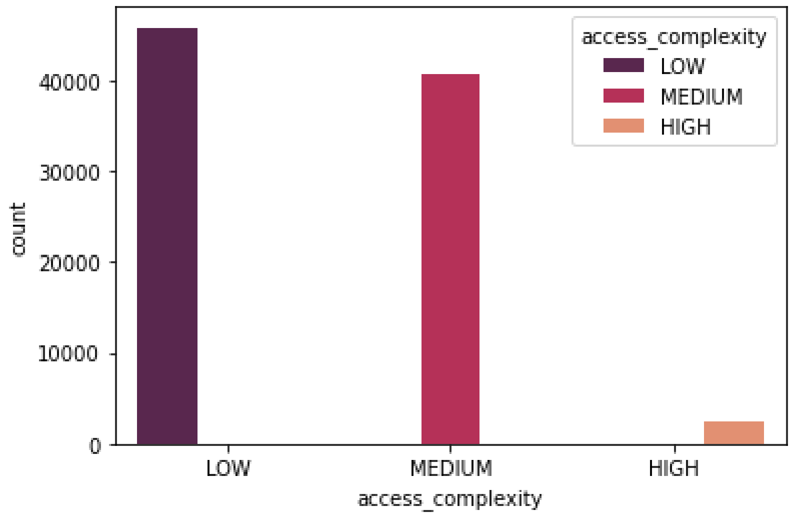


Fig.5: Count Plot for Access_Complexity

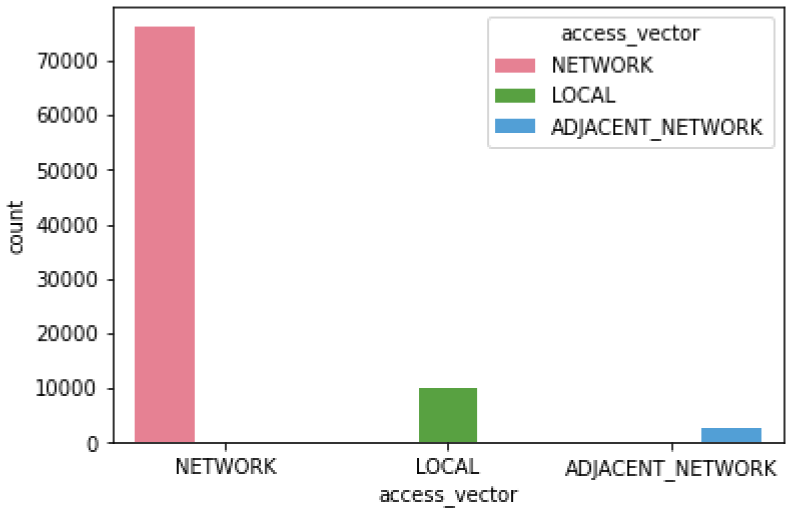


Fig.6: Count Plot for Access Vector

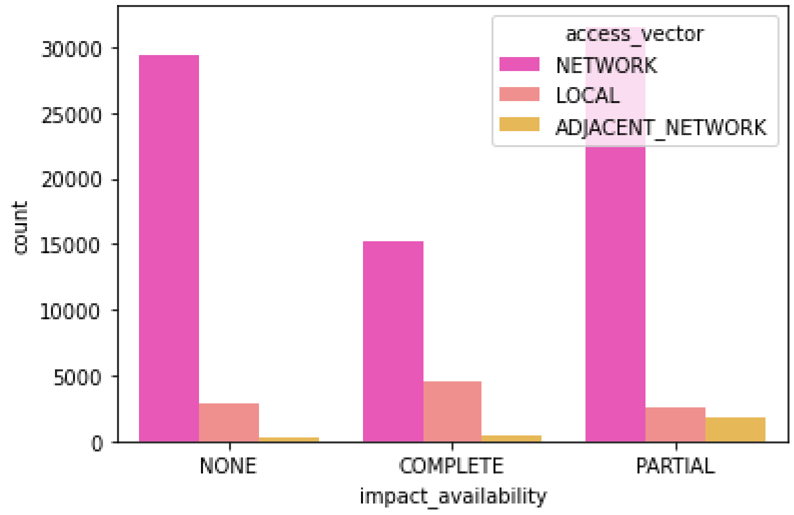


Fig.7: Count Plot for Impact_Availability

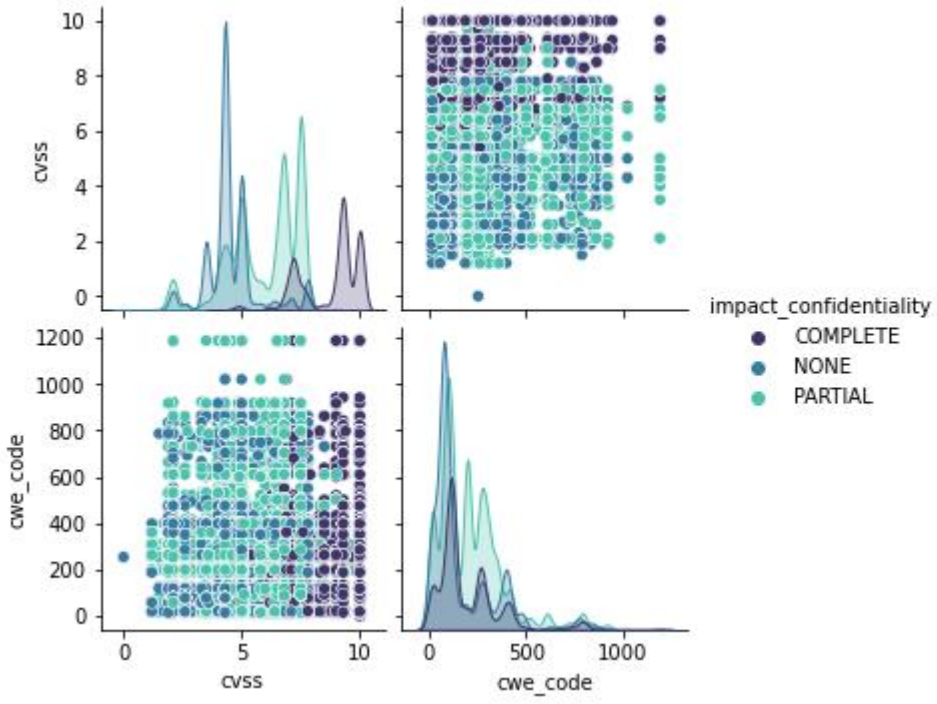


Fig.8: Count Plot for Impact Confidentiality

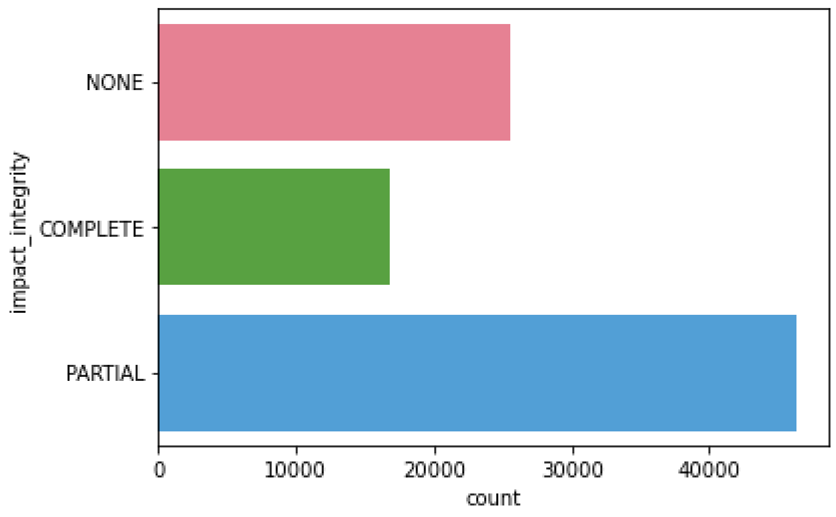


Fig.9: Count Plot for Impact Integrity

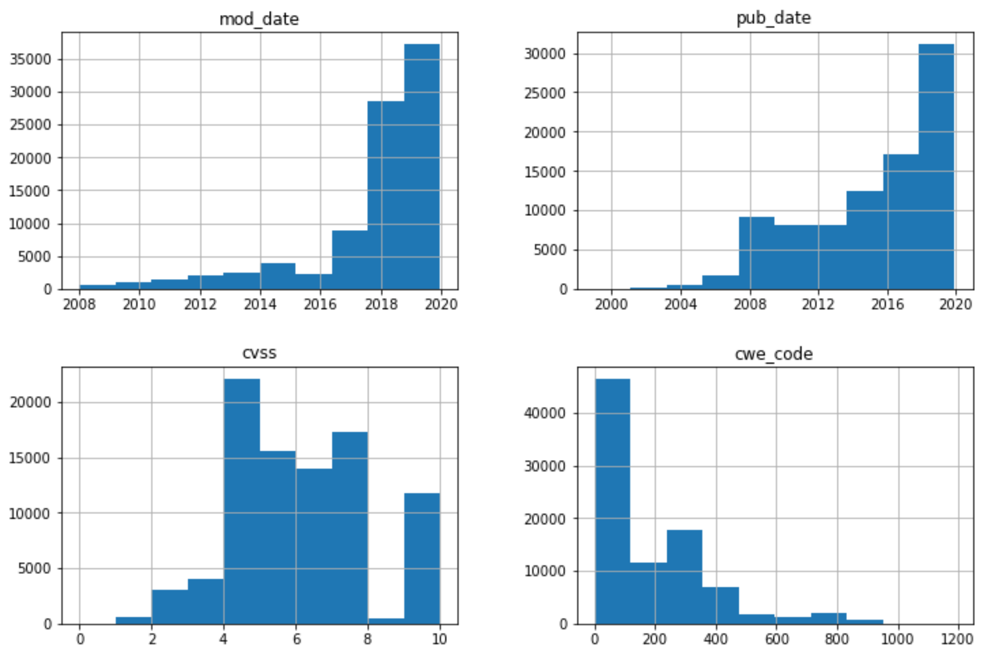


Fig.10: Histogram

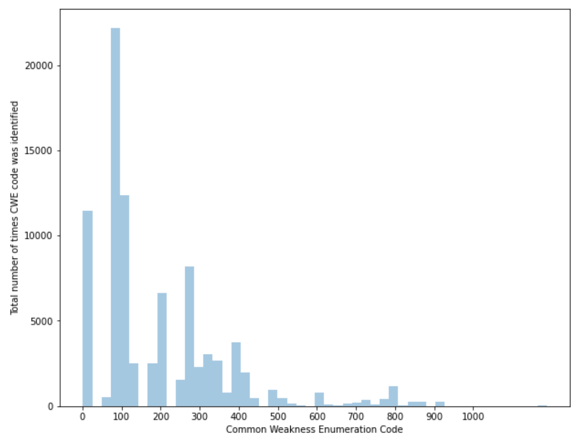


Fig.11: Common Weakness Enumeration Code

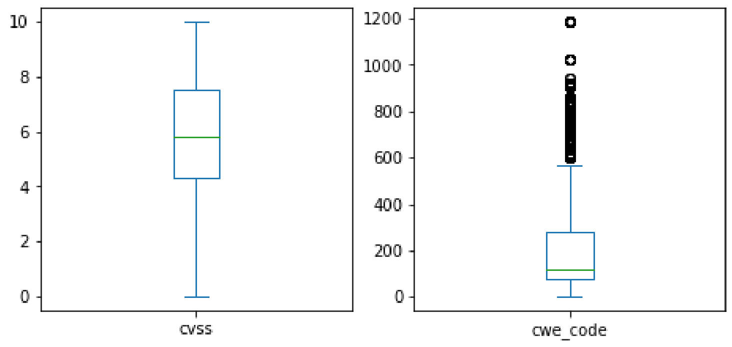


Fig.12: Box Plot

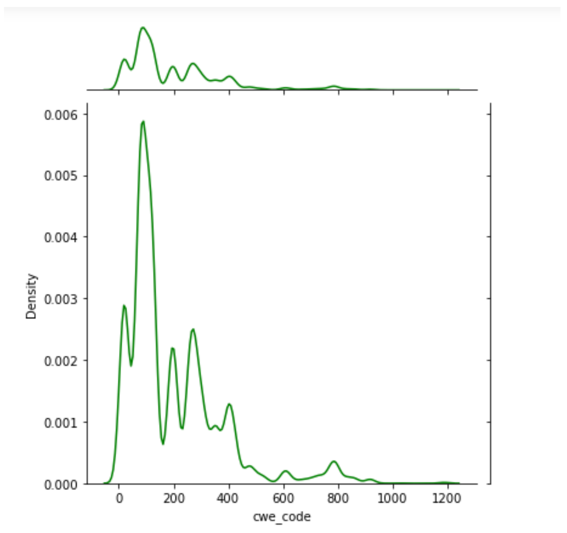


Fig.13: Joint Plot

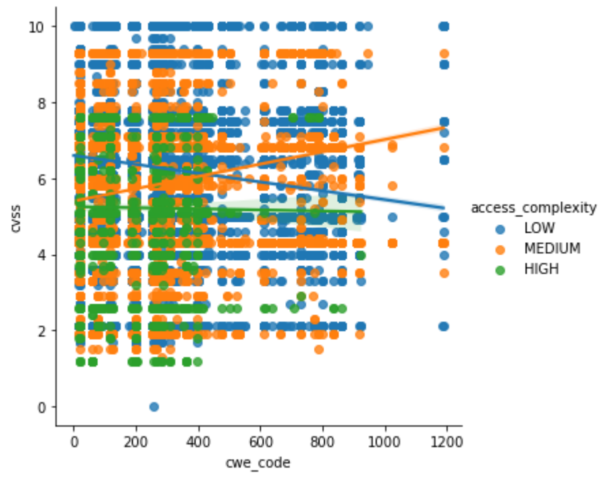


Fig.14: Count plot of cwe_code and cvss

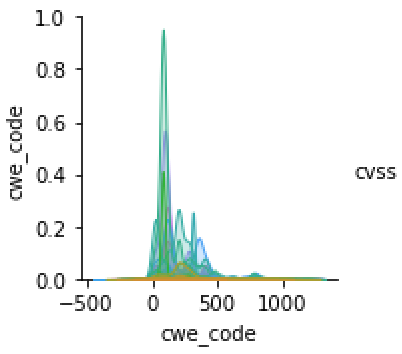


Fig.15: Pair Plot of cvss

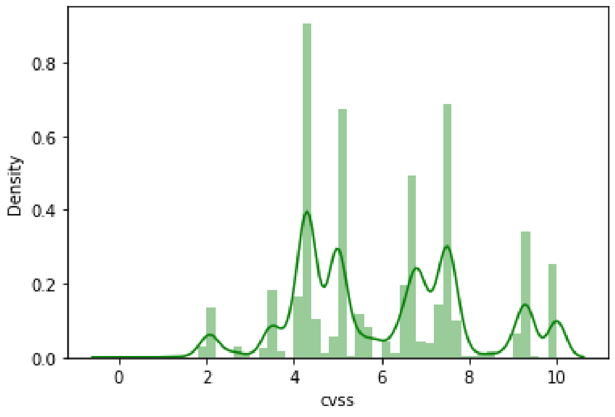


Fig.16: Dist Plot of cvss

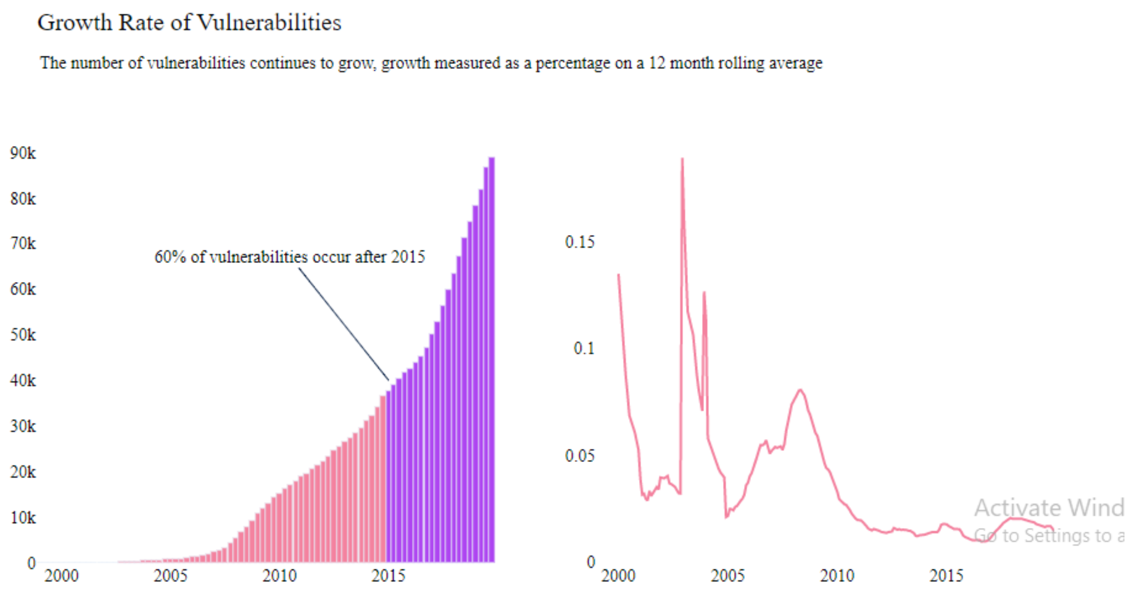


Fig.17: Growth Rate of Vulnerabilities

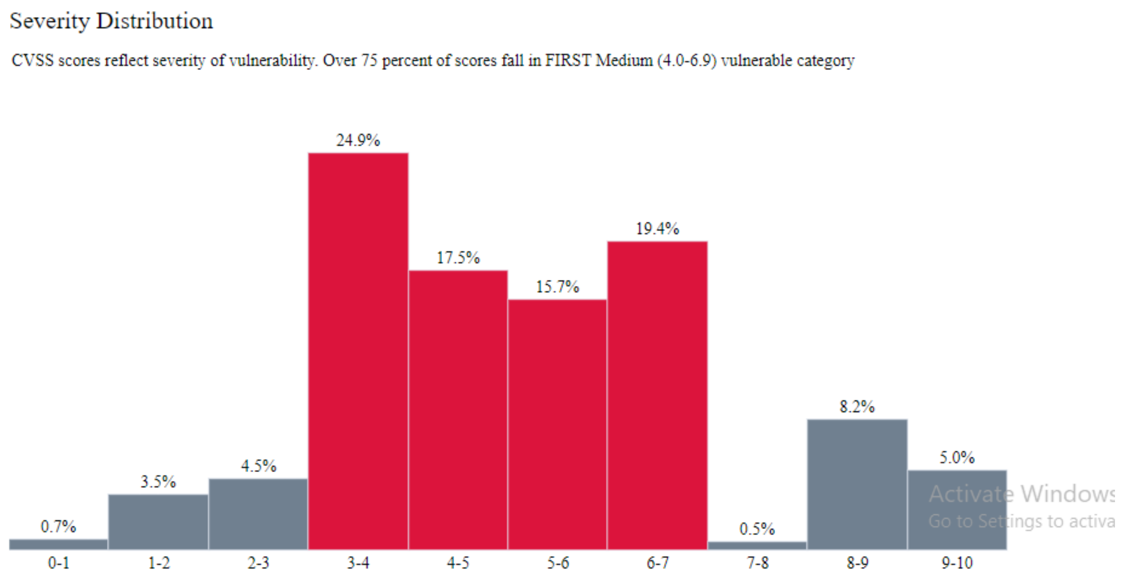


Fig.18: Severity Distribution of Vulnerabilities

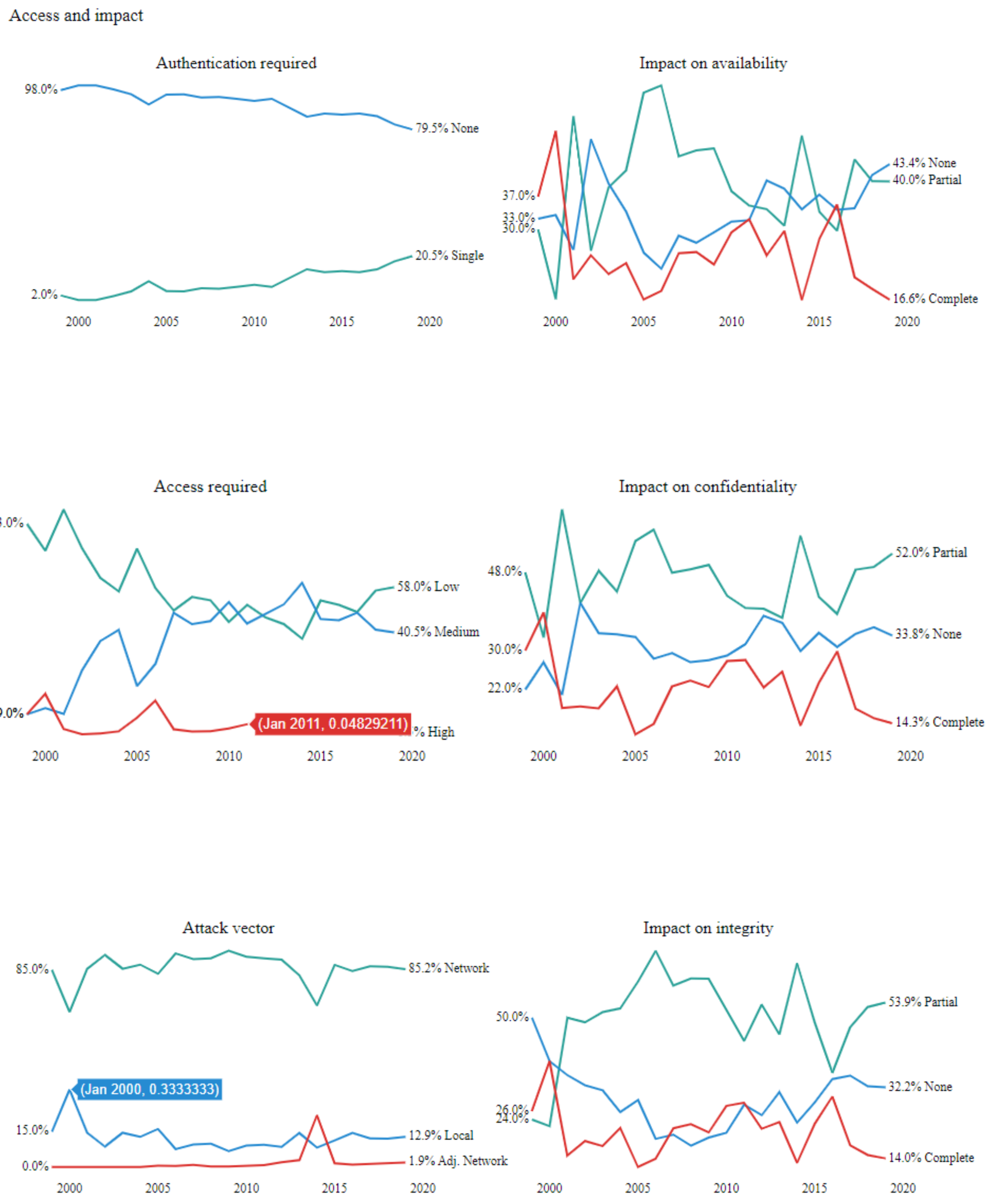


Fig.19: Value Distribution

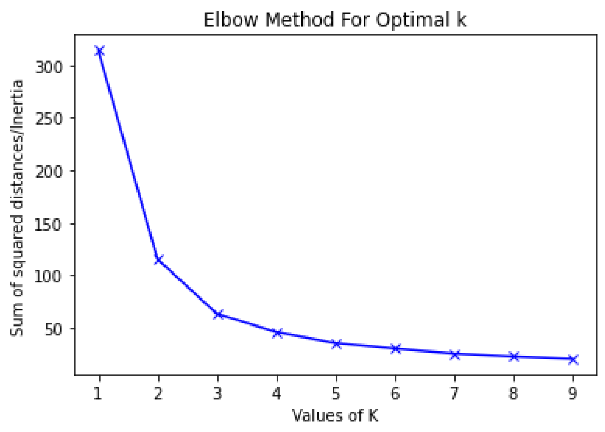
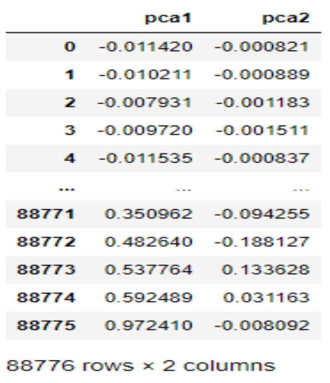


Fig.20: Elbow Method



	pca1	pca2
0	-0.011420	-0.000821
1	-0.010211	-0.000889
2	-0.007931	-0.001183
3	-0.009720	-0.001511
4	-0.011535	-0.000837
...	...	...
88771	0.350962	-0.094255
88772	0.482640	-0.188127
88773	0.537764	0.133628
88774	0.592489	0.031163
88775	0.972410	-0.008092

88776 rows × 2 columns

Fig.21: Principal Components

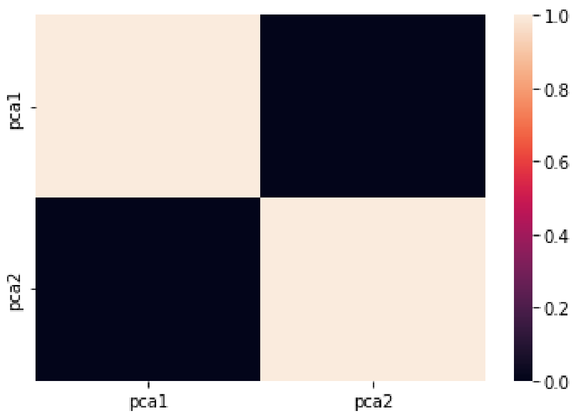


Fig.22: Heatmap after applying PCA

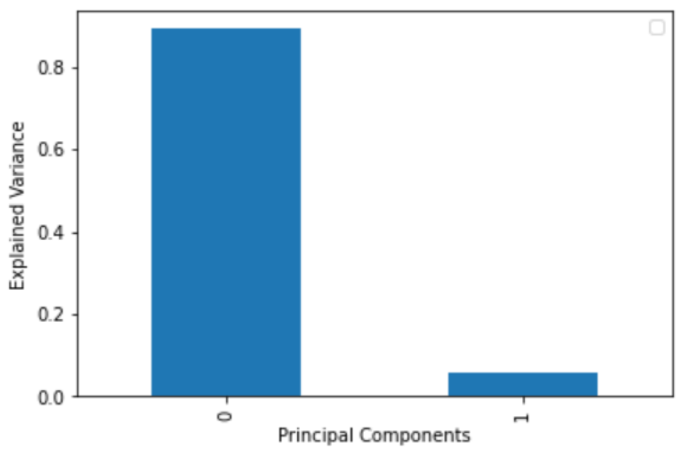


Fig.23: Variance of principal Components

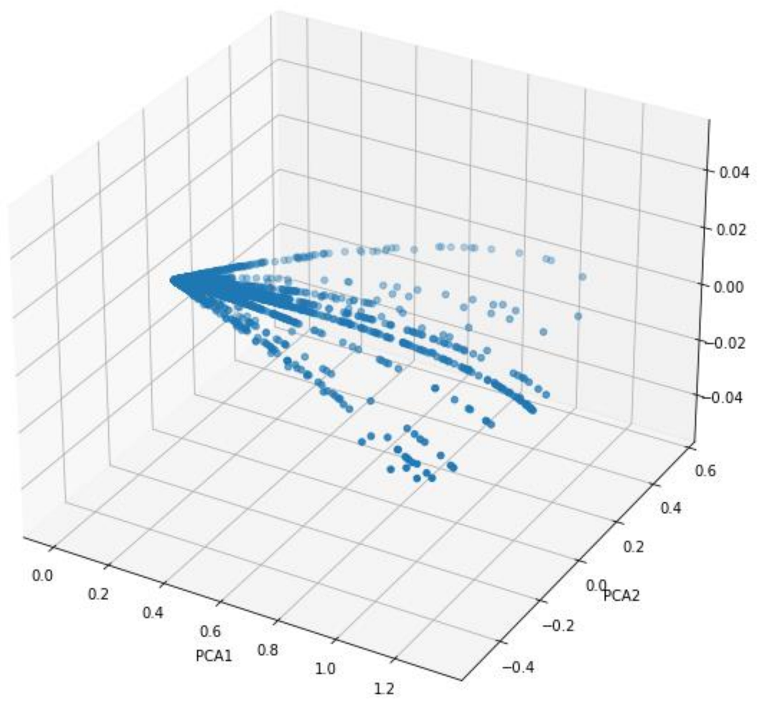


Fig.23: 3D projection of PCA

```
Text(0, 0.5, 'Inertia')
```

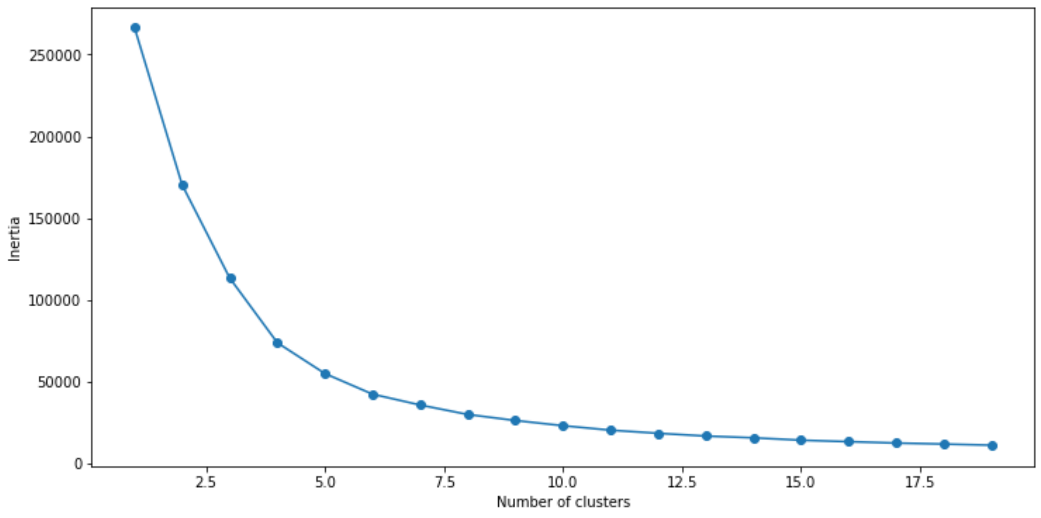


Fig.24: Inertia

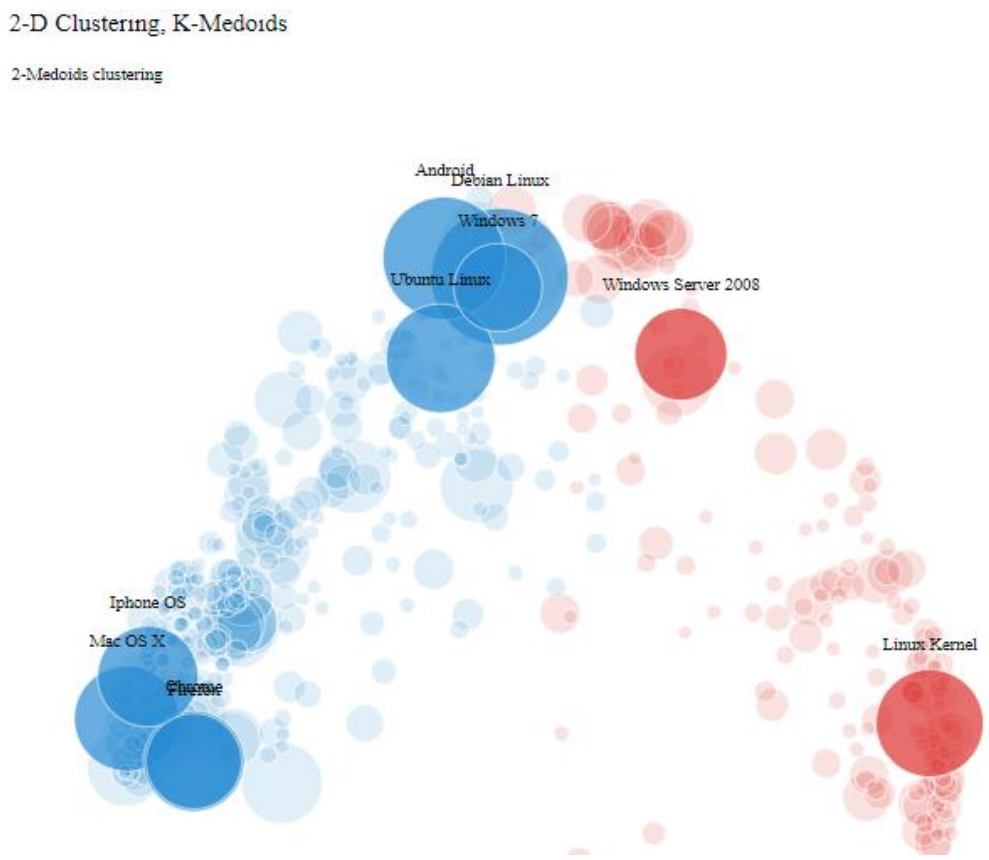


Fig.25: K-Medoid Clustering

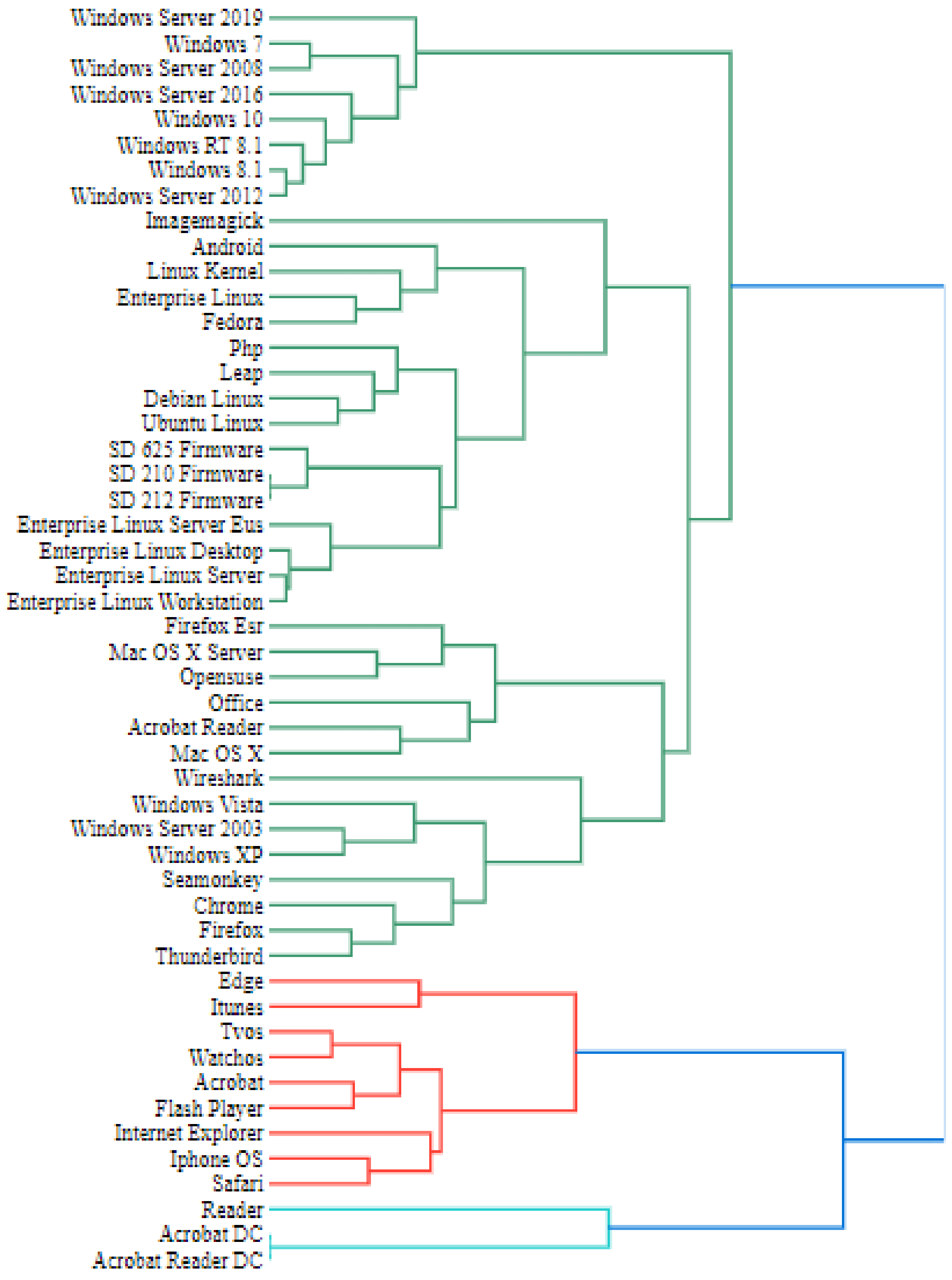
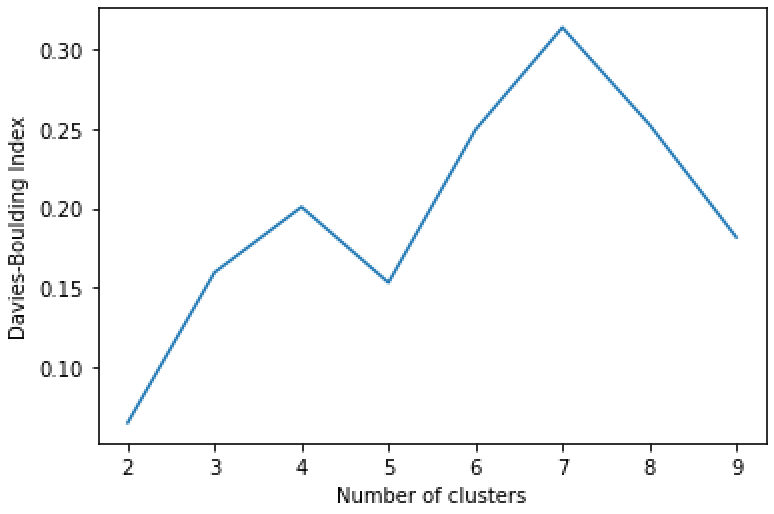
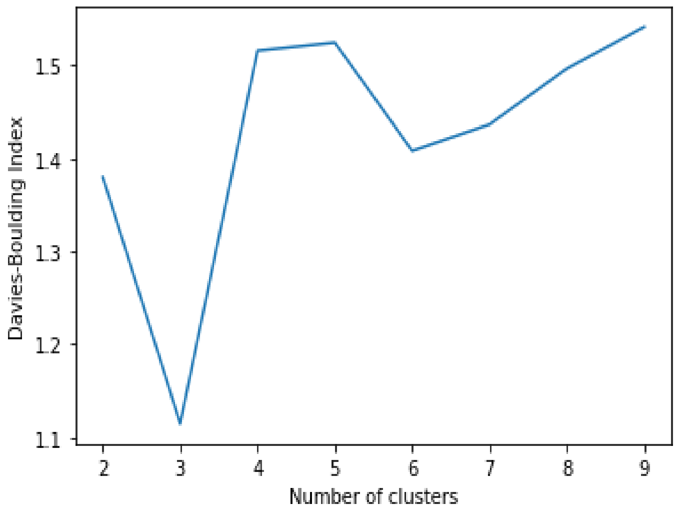


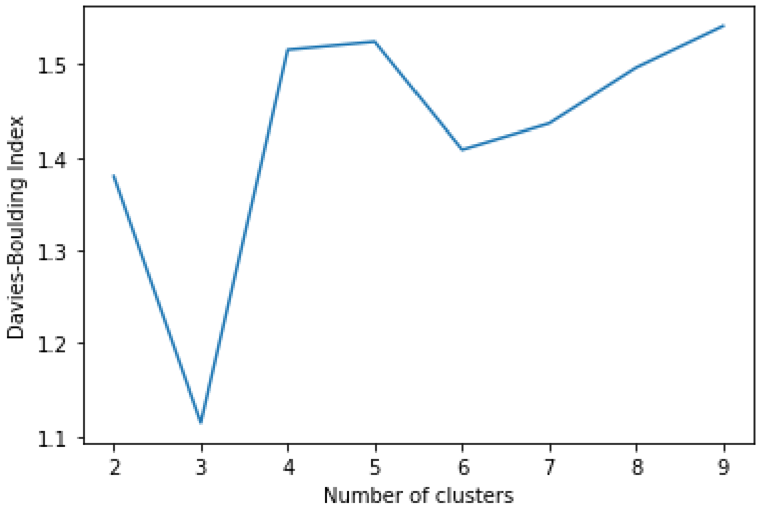
Fig.26: Hierarchical Agglomerative Clustering



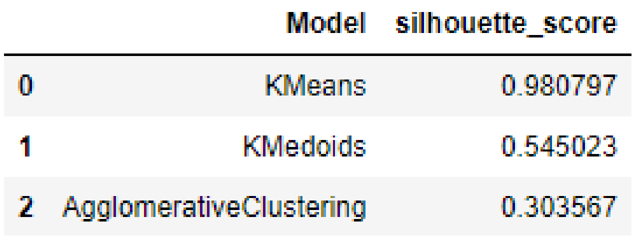
Davies-Boulding Index
0.30
0.25
0.20
0.15
0.10
2
3
4
5
6
7
8
9
Number of clusters



Davies-Boulding Index
1.5
1.4
1.3
1.2
1.1
2
3
4
5
6
7
8
9
Number of clusters

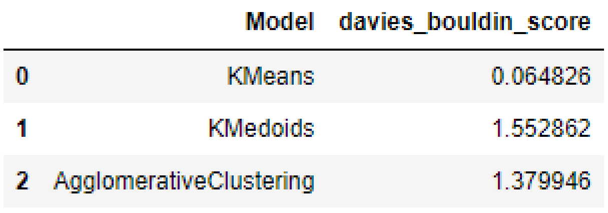


Davies-Boulding Index
1.5
1.4
1.3
1.2
1.1
2
3
4
5
6
7
8
9
Number of clusters



	Model	silhouette_score
0	KMeans	0.980797
1	KMedoids	0.545023
2	AgglomerativeClustering	0.303567

Fig.27: Results obtained through Silhouette Score



	Model	davies_bouldin_score
0	KMeans	0.064826
1	KMedoids	1.552862
2	AgglomerativeClustering	1.379946

Fig.28: Results obtained through Davies Bouldin Index

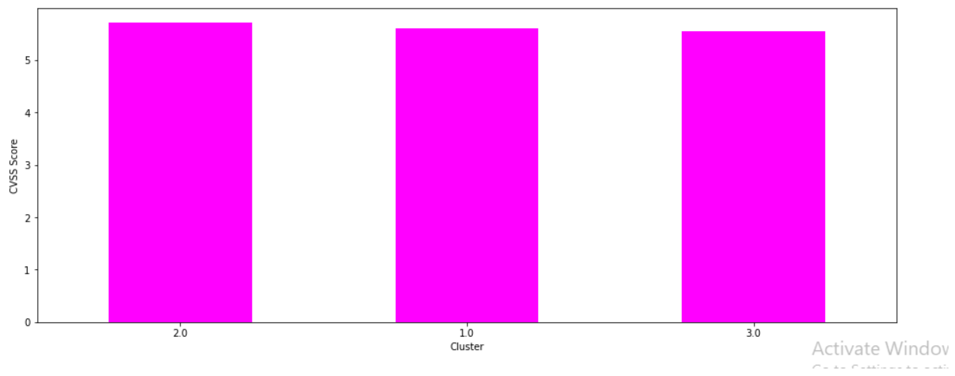


Fig.29: Clusters Formed

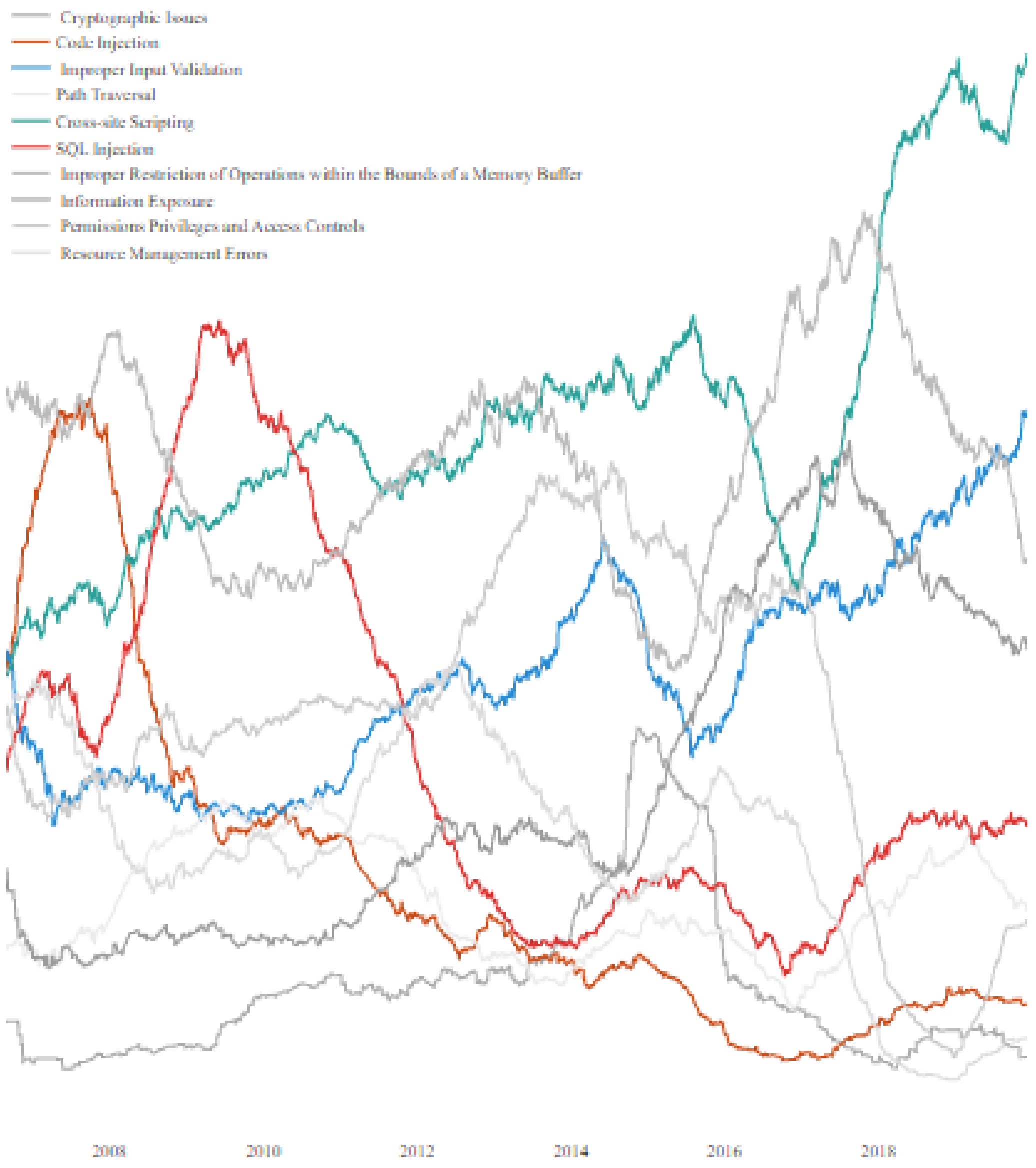


Fig.30: Vulnerabilities Over Time

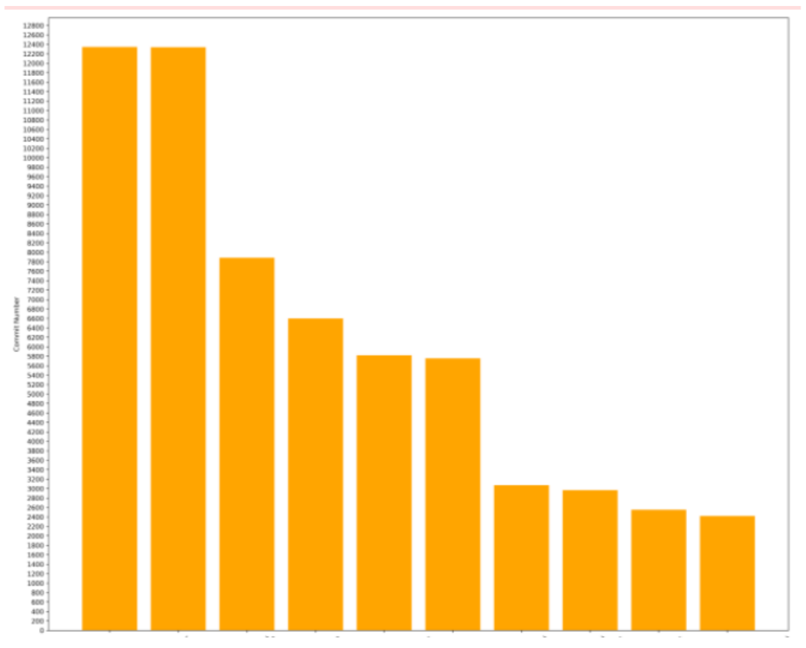


Fig.31: Top Ten Vulnerabilities

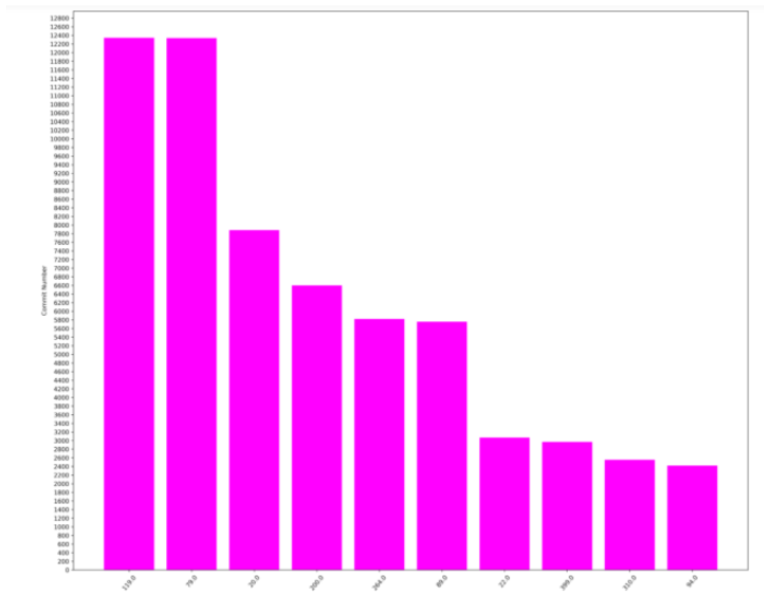


Fig.32: Top CWE Cod

Plotting products using exploit types

The top 10 products are labelled, the top 100 are projected

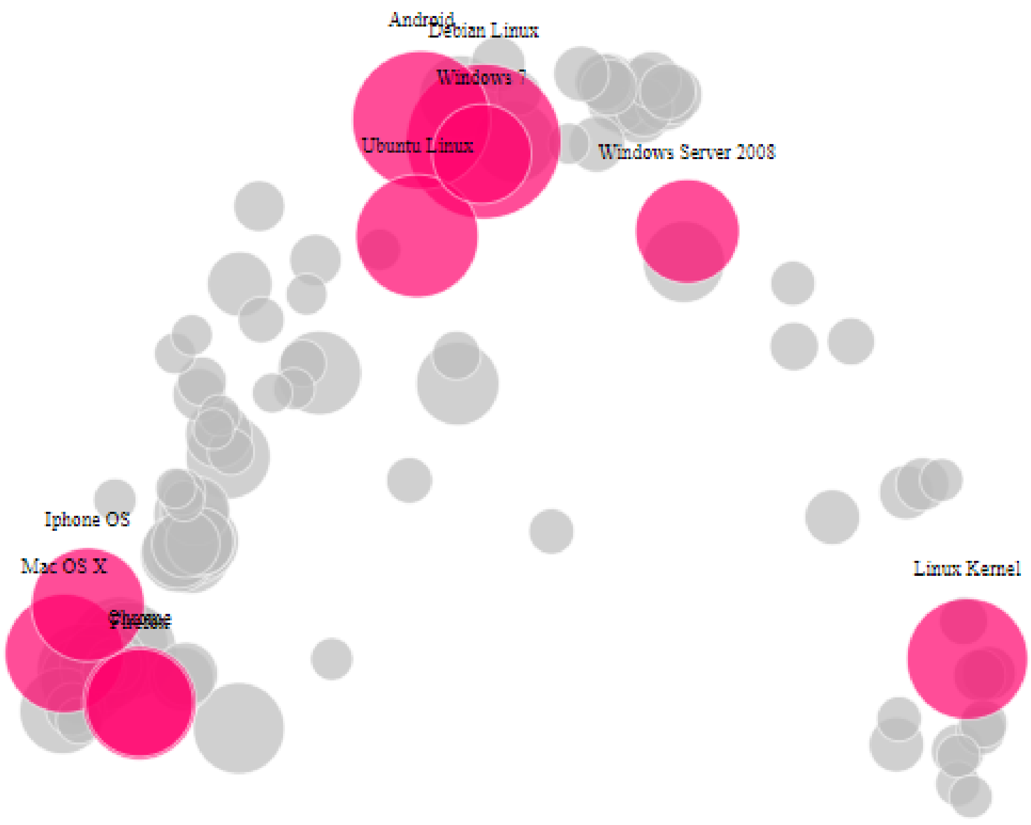


Fig.33: Exploited Products