

CLUSTERING SOFTWARE VULNERABILITIES USING SELF-ORGANIZING MAPS:
OBSERVATIONS AND ANALYSIS

By

KHYATI HARDIKKUMAR PANCHAL

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

WASHINGTON STATE UNIVERSITY
School of Engineering and Applied Sciences

MAY 2022

© Copyright by KHYATI HARDIKKUMAR PANCHAL, 2022
All Rights Reserved

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of
KHYATI HARDIKKUMAR PANCHAL find it satisfactory and recommend that it be ac-
cepted.

John Miller, Ph.D., Co-Chair

Luis De La Torre, Ph.D., Co-Chair

Mahantesh Halappanavar, Ph.D.

ACKNOWLEDGMENT

I am extremely grateful to my supervisor, Dr. John Miller, for his continued guidance and an endless supply of fascinating suggestions on how I can progress my research. His unassuming approach to research and science is a source of inspiration. This approach is reflected by his simple but clear writing style, which is something I hope to carry forward throughout my career.

Thank you to my co-chair, Dr. Luis DeLaTorre, for your encouraging efforts and thoughtful, detailed explanation on code have been very important to me. I am grateful to, Dr. Mahantesh Halappanavar, for providing me with this project. Your valuable suggestions, ever encouraging and motivating guidance was very useful for me.

Thank you to my life partner, Hardik Panchal, for constantly listening to me rant and talk things out (even after long days at work and during difficult times), for cracking jokes when things became too serious, and for the sacrifices you have made in order for me to pursue a Master's degree.

Most importantly, I am thankful for unconditional, unequivocal, and loving support from my father and mother.

CLUSTERING SOFTWARE VULNERABILITIES USING SELF-ORGANIZING MAPS: OBSERVATIONS AND ANALYSIS

ABSTRACT

by Khyati Hardikkumar Panchal, M.S.
Washington State University
May 2022

Co-Chairs: John Miller and Luis De La Torre

The goal of this research is to analyze patterns in the annotation of common vulnerabilities and exposures (CVE). One way to express these patterns is to relate CVEs to classes in Common Weakness Enumeration(CWE). Our research aims to improve this using the extensive annotation in the National Vulnerability Database (NVD). To this end, we process information from the NVD using the natural language processing model V2W-BERT, which generates a large tabular database of approximately 137,226 records each characterizing an annotation by a vector with 768 numerical attributes. Given the data in vector form, we are using the unsupervised machine learning tools to discover patterns through clustering. One of the tool we are using is Self-Organizing Maps(SOM), a well-established technique of data compression.

We expect at least a 10-fold data compression, which means a SOM output array of 6417 nodes from the full dataset. We are investigating the most informative way to interpret the SOM output array. For example, we have investigated how we can use the SOM generated codebooks of the output array to suggest the number of clusters in a K-means representation of the tabular data, followed by trace-back to the annotation to assign labels to the clusters.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENT	iii
ABSTRACT	iv
LIST OF FIGURES	ix
1 INTRODUCTION	1
1.1 Background	1
1.2 Vector data set using V2W-BERT	3
1.3 Implementation tool	5
2 CLUSTERING USING SOM	8
2.1 K-means clustering	8
2.2 Self-organizing Map (SOM)	9
2.3 SOM functions for preliminary setup	11
2.4 SOM visualization using U-matrix	13
2.5 Using K-means with SOM	16
3 CLUSTERING QUALITY INDEX: DAVIES-BOULDIN INDEX(DBI)	18
4 CLUSTERING RESULTS AND ANALYSIS	24
4.1 Clustering results	24
4.1.1 Bar graph representation of clusters	24
4.2 Contribution of CWEs to BMUs of Cluster 6	28
5 CASE STUDY OF CLUSTERS USING U-MATRIX	32
5.1 Understanding structure of supercluster CWE-79	34
6 FUTURE RESEARCH WORK	47
7 CONCLUSIONS	49

8 Matlab Code	51
8.1 alldataClustering138.m file	51
8.2 graph.m file	53
8.3 percentageData.m file	54
REFERENCES	57

LIST OF TABLES

1.1	Sample of original CVE dataset	4
1.2	Sample of original CWE dataset	6
2.1	Sample of placement of 6417 Cell numbers into first 12 vectors	14
2.2	Sample of placement of 138 clusters into first 12 cell numbers	17
3.1	Sample of DBI matrix with index values using K-means from 131 to 140 . .	20
3.2	Result label matrix with first 20 records	22
4.1	Data distribution for cluster 6(first 20 records)	30
5.1	Total percentage computation based on data distribution in two prominent peaks(first 15 records in descending order)	33
5.2	Sample of distance matrix of adjacent clusters to cluster 9(in ascending order top five based on minimum distance)	36
5.3	Snippet of cluster 9 with CWE-79 and related CVEs	37
5.4	Snippet of cluster 104 with CWE-79 and related CVEs	37
5.5	Snippet of cluster 138 with CWE-352 and related CVEs	40
5.6	Sample of distance matrix of adjacent clusters to cluster 39(in ascending order top five based on minimum distance)	41
5.7	Sample of distance matrix of adjacent clusters to cluster 60(in ascending order top three based on minimum distance)	43

5.8	Sample of distance matrix of adjacent clusters to cluster 81(in ascending order top five based on minimum distance)	44
5.9	Sample of distance matrix of adjacent clusters to cluster 127(in ascending order top five based on minimum distance)	46

LIST OF FIGURES

2.1	Overview of process from NVD dataset to SOM Clustering	10
2.2	1×1 matrix sD for data	11
2.3	1×1 structure sM for data	12
2.4	(a) U-matrix using SOM, (b) U-matrix with hits	15
2.5	U-matrix using K-means with 138 clusters	16
3.1	K-means clustering using Davise-Bouldin's Index with an extended range search suggested best value for k	21
3.2	Squared sum of errors for selecting best k	23
4.1	Bar graph for cluster 6	25
4.2	Bar graph for first 10 clusters	26
4.3	Bar graph for cluster 11 to 20	27
4.4	Bar graph for cluster 121 to 130	28
4.5	Role of CWEs 59 and 264 in the BMUs of members of cluster 6	29
5.1	U-matrix of clusters only containing CWE-79 for prominent peak percentage more than 50%	35
5.2	U-matrix of clusters only containing CWE-352 for prominent peak percentage more than 50%	39

5.3	U-matrix of clusters only containing CWE-89 for prominent peak percentage more than 50%	40
5.4	U-matrix of clusters only containing CWE-22 for prominent peak percentage more than 50%	42
5.5	U-matrix of clusters only containing CWE-200 for prominent peak percentage more than 50%	43
5.6	U-matrix of clusters only containing CWE-119 for prominent peak percentage more than 50%	45

Dedication

This thesis is dedicated to my husband, mother and father who have given me courage to start this journey and providing emotional support.

Chapter One

INTRODUCTION

1.1 Background

With emerging technologies and digitization people sometimes forget that they are leaving traces on digital devices and communications over the Internet. These traces could result in potential threat on privacy and security of our devices. Cybercrime has been rapidly growing over the years which leads our attention towards those security incidents and potential threats, integrating security aspects during the development and testing phase of software applications has been very important lately. Generally, attacks are detected from software applications while exploiting vulnerabilities or weaknesses and leading us to major defects. "A software vulnerability is a flaw in the system that allows an attacker to breach the security measures implemented." [1]

In 1999, National Vulnerability Database (NVD) which is a public data repository for maintaining standardized information, started publishing information on software vulnerabilities and configuration settings that are affecting software applications. The purpose of this database was to provide valuable information to help organizations with future mitigation processes and understand trends and patterns of software vulnerabilities. To allow this information for public usage and understand it so programmers can identify and re-

solve vulnerabilities, the co-creators of ¹MITRE Corporation came up with the Common Enumeration of Vulnerabilities (CVE) list also known as dictionary of publicly available vulnerabilities and exposures in 1999.

Based on ²CVE's FAQ, "CVE is a list of information security vulnerabilities and exposures that aims to provide common names for publicly known problems. The goal of CVE is to make it easier to share data across separate vulnerability capabilities (tools, repositories, and services) with this 'common enumeration.'" Each CVE contains a CVE ID, a detailed description of the CVE, and references to vulnerability reports. Currently, we have 172,659 CVE records. Another important community-developed list from MITRE is Common Weakness Enumeration (CWE). Based on About CWE, ³ "CWE is the list of common software and hardware weakness types developed by a community that has security ramifications." It is also defined as a hierarchically designed dictionary of software weaknesses that allows understanding software flaws, potential impact of their exploitation to mitigate these flaws. This list contains 916 CWEs in the MITRE database which primarily focuses on how and why the vulnerabilities can be exploited. CVE instances are classified into CWEs which are ordered in a hierarchical structure to help develop countermeasure techniques and get a better understanding of CVE implications.

NVD integrates CWE into the scoring of Common Vulnerabilities and Exposures (CVE®) entries, upon which NVD is built, by providing a cross section of the overall CWE structure. NVD analysts score CVEs using CWEs from different levels of the hierarchical structure. This allows analysts to score CVEs at both a fine and coarse granularity, which is necessary due to the varying levels of specificity possessed by different CVEs. For example, tracing the path from the root node, CWE-707, to a node CWE-892, reveals that SQL injection (CWE-89) is caused by improper neutralization of special elements in data query logic (CWE-943),

¹<http://cve.mitre.org/>

²<http://cve.mitre.org/about/faqs.html>

³<http://cwe.mitre.org/>

which in turn is caused by injection (CWE-74) or sent to a downstream component (CWE-707). This insight provides a means to design countermeasures even when a specific CWE node is not available.[2] Based on the mapping we can associate impact of specific vulnerability to weakness and develop steps to mitigate them. Developers most of the time rely on CVEs and CWEs to learn about weaknesses and vulnerabilities for privacy concerns. Thus, analyzing the correct path from a root to lower level nodes provides valuable insight and functional directions to learn more about specific weakness. We are observing all the CVE’s from year 1997 to 2020 and mapping them to related CWE’s based on hierarchical classification. Original data set without vector representation is shown in table 1.1 where CVE’s are mapped to related CWE label after the training from V2W-BERT model.

1.2 Vector data set using V2W-BERT

We are using the data set that has been transformed into vector representation using natural language processing with the V2W-BERT framework. Bidirectional Encoder Representation from Transformer(BERT) is a bidirectional model which understands context based on its surroundings. V2W-BERT framework is designed to classify CVEs to CWEs hierarchically using different training and optimization processes. It has 3 steps: The first step is called unsupervised pre-training. In this step, CVE and CWE descriptions are given to the pre-trained BERT language model. In the second step, also referred to as the Link prediction (LP) component, the trained BERT model transforms CVE/CWE description and tries to establish a link between them. If a particular CVE belongs to CWE then it adds the confidence value to it. In the third step Reconstruction Decoder (RD) comes into the picture. It helps preserve the context of the corresponding CVEs and CWEs bypassing the last hidden state to the Masked Language Model (LM) and optimizes tokens while the link prediction component is trying to predict the link. LP and RD share BERTs’ hidden states, updates are made on trainable layers for loss in link classification and reconstruction. This is how

CVE Dataset			
publishedDate	CVE ID	CVE Description	CWE Code
1999-12-30 05:00:00+00:00	CVE-1999-0001	ip_input.c in BSD-derived TCPIP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.	['CWE-20']
1998-10-12 04:00:00+00:00	CVE-1999-0002	Buffer overflow in NFS mountd gives root access to remote attackers, mostly in Linux systems.	['CWE-119']
1998-06-26 04:00:00+00:00	CVE-1999-0007	Information from SSL-encrypted sessions via PKCS 1.	['CWE-327']
1997-07-16 04:00:00+00:00	CVE-1999-0027	Root privileges via buffer overflow in eject command on SGI IRIX systems.	['CWE-119']

Table 1.1 Sample of original CVE dataset

V2W-BERT preserves context while training for link classification as per Siddhartha Das's discussion.[3]

We are using a V2W-BERT model trained word2vec transformed data set which has approximately 137,226 records each characterizing an annotation by a vector with 768 numerical attributes. From all those records we are removing the records which are categorized as 'NVD-CWE-Other', 'NVD-CWE-noinfo', and 'NONE'. After filtering all the records containing the above categorization we retrieve a database with 99,950 records and 768 attributes. In table 1.2 Table 1.2 shows a brief description of CWE-120. It explains how buffer flow can leads to a security breach.

1.3 Implementation tool

We are using MATLAB which is an acronym for "Matrix Laboratory" programming language. It was developed by Cleve Moler for resolving issues of his students with calculations of algebra and numerical analysis at the University of New Mexico.

We are using MATLAB with add-On toolboxes such as Self Organizing Map (SOM) toolbox, deep learning toolbox, Computer Vision toolbox, and Optimizing toolbox to obtain the visual representations and perform analysis. One of the main toolboxes we are using is the SOM toolbox which is a magnificent tool to overcome some of the features that were not provided by the neural networks toolbox such as training SOM with different learning parameters and network technologies, computing principal component analysis, quality and measures for SOM, data processing tool, etc. based on Toolbox Home ⁴. It was developed by the Laboratory of Computer Science(CIS) at Helsinki University of Technology.

The deep learning toolbox allows MATLAB to add additional features for designing and implementing deep neural networks with algorithms, pre-trained models, and applications without having an understanding of advanced computer vision algorithms or neural networks.

⁴<http://www.cis.hut.fi/projects/somtoolbox/>

Example of CWE		
CWE Description	CWE Code	Name
<p>Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow. A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer. The simplest type of error, and the most common cause of buffer overflows, is the classic case in which the program copies the buffer without restricting how much is copied. Other variants exist, but the existence of a classic overflow strongly suggests that the programmer is not considering even the most basic of security protections. Execute Unauthorized Code or Commands:NOTE:Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. This can often be used to subvert any other security service.:SCOPE:Availability:IMPACT:DoS: Crash, Exit, or Restart:IMPACT:DoS: Resource Consumption (CPU):NOTE:Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop.:</p>	['CWE-120']	CWE-120

Table 1.2 Sample of original CWE dataset

We are also using the Computer Vision toolbox which provides capabilities for 3D vision, video processing, developing algorithms, and functions for the design and test of computer vision.

Chapter Two

CLUSTERING USING SOM

2.1 K-means clustering

"Data clustering is a data exploration technique in which data objects being processed are partitioned into groups based on some similarity criteria." [4] One of the first methods that have been used by social scientists and biologists was hierarchical clustering but the analysis of clusters comes under the unsupervised learning branch for statistical multivariate analysis. From a statistical point of view, it has been divided into two parts: 1) probability model-based approaches where the main focus is on utilizing a mixture likelihood approach for clustering data points [5] and 2) non-parametric approaches where clustering methods based on similarity and dissimilarity of the objective function that is being used [6].

Furthermore, non-parametric approaches can be divided into two parts: hierarchical methods and partitional methods (which are being used in higher proportion). Usually, for applying partitional methods we need finite cluster prototypes by defining the distance between a particular point and the cluster prototype. The most well know and oldest partitioning method is the K-means algorithm. In K-means algorithms we as a user try to identify an appropriate number of clusters using the trial-and-error method. Pseudocode for the k-means clustering algorithm is as shown in Algorithm 1. [7]

Given the size of dataset, data compression is needed and it's very critical to select

Algorithm 1 K-means clustering algorithm

Require: Data = $a_1, a_2, a_3, \dots, a_i, \dots, a_n$ // Set of n data points. k // Number of desired clusters

Ensure: A set of k clusters

- 1: Choose k data points from Data as initial centroids;
 - 2: Repeat: Assign each point a_i to the cluster which has the closest centroid; Calculate the new mean for each cluster; Until convergence criteria is met.
-

initialization for the value of K , and sometimes it's hard to choose considerable clusters needed. The benefits of using the K-means algorithm are easy to implement and required execution time very less. The disadvantage of K-means is it selects the initial centroid randomly, so in case we select a poor initial centroid then the obtained clustering results are also not well. However, there are multiple indices developed by different researchers for obtaining the best value of K . For example, Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC), Dunn's index, Davies-Bouldin index (DBI), Silhouette Width (SW), Calinski and Harabasz index (CH), Gap statistic, generalized Dunn's index (DN_g), and modified Dunn's index (DN_s). We will be using DBI for determining the best value for K .

2.2 Self-organizing Map (SOM)

SOM is a data visualization paradigm introduced by Professor Teuvo Kohonen. SOM clusters similar data and performs data mining using topology-preserving mapping and training through data distribution to achieve data compression. SOM provides visualization of the structure of higher-dimensional data space in lower dimensions by allowing neighborhood relations on a rectangular or hexagonal lattice.[8] It gives dimensionality reduction which allows people to visualize and interpret what would otherwise be, indecipherable data. SOMs generate subspaces with an unsupervised learning neural network trained with a competitive

learning algorithm. For clustering, neuron weights are fine-tuned based on their closeness to "winning" neurons (i.e. neurons that most closely resemble a sample input). When we train input data sets on multiple iterations as an outcome we achieve the association of similar neurons and vice versa.

A SOM consists of neurons, which are connected to adjacent neurons by neighborhood relations. In the training step, one vector x from the input set is chosen, and all the weight vectors of the SOM are calculated using some distance measure such as the Euclidian distance (Kohonen, 2001). The neuron, whose weight vector is closest to the input x is called the best-matching unit (BMU). After finding the BMU, the weighting vectors of the SOM are updated so that the BMU is moved closer to the input vector.

In this proposal, we used the information from the NVD database for all the CVEs from the year 1999 to 2020, and we processed it with the natural language processing model V2W-BERT. Once we received the processed data from the natural language processing tool, it is converted into a vector format. Then we utilize codebooks generated using *Som_make()* function for performing K-means clustering using the SOM toolbox in MATLAB.

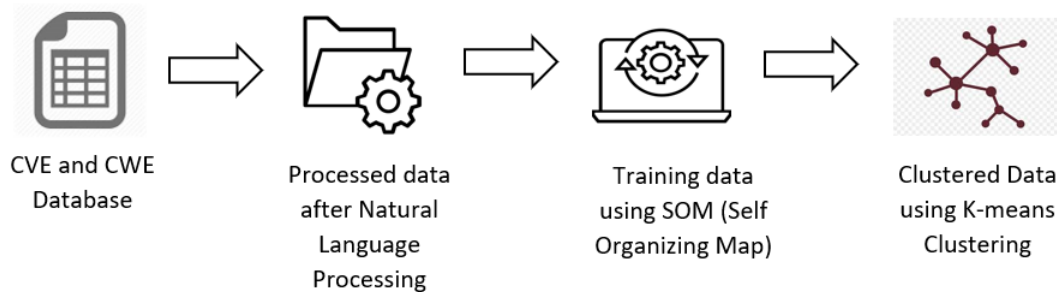


Figure 2.1 Overview of clustering of NVD dataset using K-means clustering with SOM.

As we can see in figure 2.1, the first step in this process is to use the data set derived from NVD, and then the data is sent for natural language processing using the V2W-BERT model. After the processing, we receive the data set in vector format that we use with the

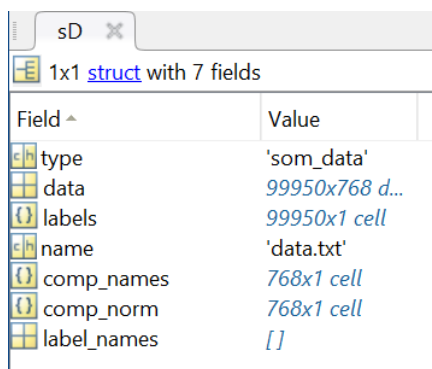
SOM toolbox. Then, we load data without 'NVD-CWE-Other', 'NVD-CWE-noinfo', and 'NONE' CVE labels in MATLAB using the SOM toolbox function.

2.3 SOM functions for preliminary setup

We are using functions provided by the SOM toolbox for data compression and visualization. To read the vectored data using SOM, we use the `som_read_data()` function. The purpose of this function is to read data from an ASCII file in SOM_PAK format. The specifications for the SOM_PAK data file format is that the first line must contain the input space dimension. The following lines are comment lines, empty lines, or data lines. Because this function can also determine the input dimension from the first data lines if the input space dimension line is missing it won't generate the intended output. It loads 99,950 records with 786 attributes. Based on the SOM toolbox website syntax is as follows:

¹ `sD = som_read_data(filename);`

It generates 1×1 structure in MATLAB work-space as shown in figure 2.2.



Field ^	Value
type	'som_data'
data	99950x768 d...
labels	99950x1 cell
name	'data.txt'
comp_names	768x1 cell
comp_norm	768x1 cell
label_names	[]

Figure 2.2 1×1 matrix sD for data set with 99950 elements

Now, we are going to see the next function, which is `som_make()`. It creates, initializes, and trains vector datasets. For example, first step is to determine number of map units using

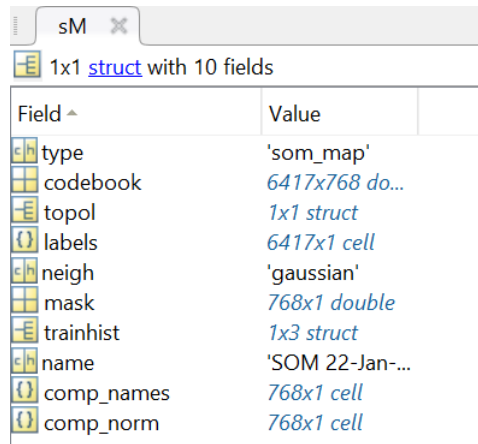
¹http://www.cis.hut.fi/projects/somtoolbox/package/docs2/som_read_data.html

heuristic formula of $units = 5 * dlen^{0.54321}$.

The next step is to determine map size based on the map unit. Then SOM is initialized randomly. Then it's trained in two phases: rough training and fine-tuning which is performed using an either sequential or batch algorithm. Once the training is done final map is calculated. The syntax for `som_make` is as follows:

```
sM=som_make(sD, 'shape', 'toroid', 'mapsize', 'big', 'training', 'long', 'tracking', 0);
```

Argument 'shape' describes the shape of a map such as 'sheet', 'cyl' or 'toroid'. 'mapsize' argument defines the size of a map such as 'small', 'big', or 'normal'. 'training' describes the length of rough training and length of fine-tuning in epochs. 'tracking' usually defines the amount of reporting during training. It generates 1×1 structure in MATLAB workspace as shown in figure 2.3



Field ^	Value
type	'som_map'
codebook	6417x768 do...
topol	1x1 struct
labels	6417x1 cell
neigh	'gaussian'
mask	768x1 double
trainhist	1x3 struct
name	'SOM 22-Jan-...
comp_names	768x1 cell
comp_norm	768x1 cell

Figure 2.3 1×1 structure sM for data set with 99950 elements

After we process data with the `som_make()` function 99950 records get compressed to 6417 elements. As we can see from figure 2.3, one of the fields is 'codebook' that allows a two-dimensional representation of the input space prototypical vectors called codebooks associated with each node of a rectangular grid. Codebooks are expected to be similar to the centroids that would result from K-means applied directly to the input data. Usually, the setup of SOM is done with a rectangular grid which is an arrangement of units and each

one of those units contains a codebook.

Another important function is `som_bmus()`. It is used to find Best-Matching-Unit (BMU) in the SOM output array for a particular input vector. As the result of this function index of BMU is returned by default but using the 'which' argument we can retrieve second and third best matching units as well. usually, we calculate the distance between the sample vector to each weight vector. Weight with the shortest distance gets selected as BMU. The default output is the node of the SOM output array with codebook that is most like the input vector in the Euclidian distance sense.

Clustering follows the partitioning process as follows: 1) If the sub-cluster represents less than half of the parent clusters sample then the parent cluster is considered. Otherwise, a sub-cluster is used. 2) Remaining units are partitioned based on distance, whoever is closest gets placed in the nearest cluster. 3) Now in the last step Partitioned original data sample gets distributed similar way as its BMU.[9] From 99950 records `som_bums()` function assigns each of the values from 1 to 6417 as seen in table 2.1. Syntax of BMU function is as below:

```
Bmus = som_bmus(sM, sD);
```

2.4 SOM visualization using U-matrix

There are multiple techniques to display clusters on SOM. One of them is the distance matrix. A unified distance matrix also known as U-matrix is used to visualize the similarity of the codebooks of adjacent nodes by using a different coloring of the connection between nodes. Generally, dark color between nodes represents a large distance gap which is seen as a border between clusters for separation and light color represents a small distance gap which is usually seen as clusters. Viewing the U-matrix as a topological map allows graph-theory method to be used in clustering of BMUs" (reference starburst method).

An example of a U-matrix is shown in figure 2.4(a). The light blue dots indicate the location of map units and hexagons between them show the actual values of the U-matrix.

Examples of BMU	
Index of input vector	BMU
1	5249
2	5723
3	1740
4	6286
5	1990
6	5531
7	4330
8	4144
9	5076
10	6003
11	2945
12	7

Table 2.1 Sample of placement of 6417 Cell numbers into first 12 vectors

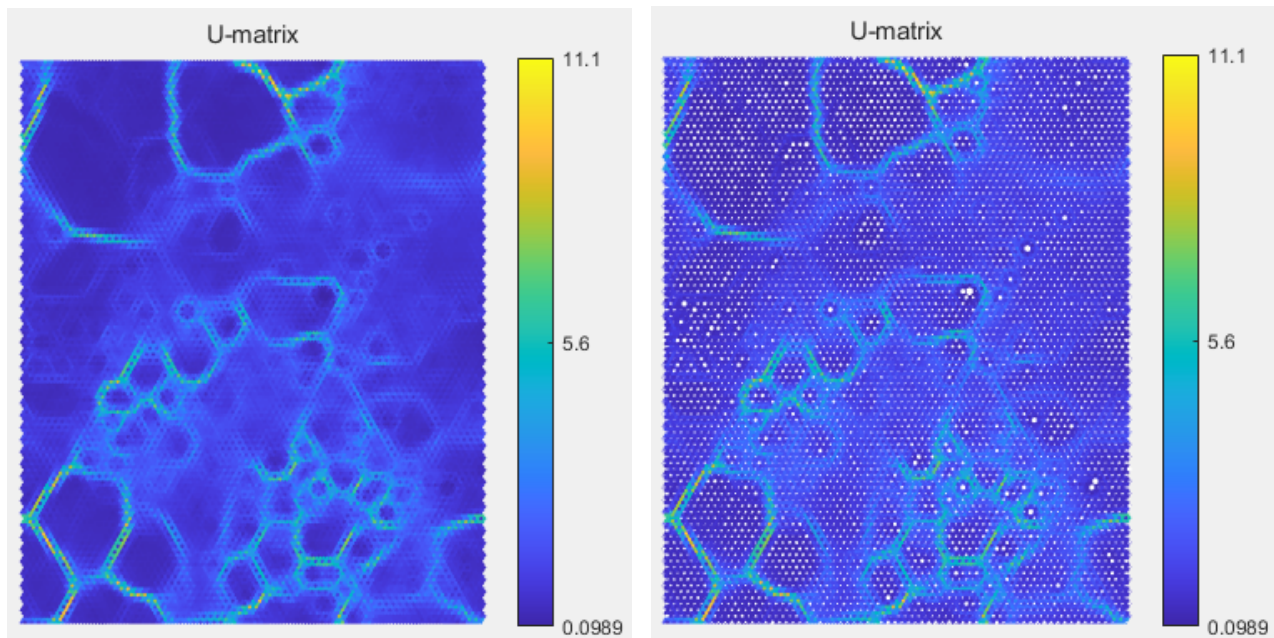


Figure 2.4 (a) U-matrix using SOM, (b) U-matrix with hits

The darker the shade, the smaller the distance. UMAT is related to difference in codebooks. Similar codebooks (smaller UMAT) make up clusters. Lighter strings In fig 2.4(b) larger dots show greater number of hits. Boundaries between cluster are the same as in 2.4(a). Information on 'hits' can be specifically used in clustering using SOM by indicating cluster borders with zero-hit units. The hits or size markers represent the distribution of each neuron.

We can visualize U-matrix using `som_show()` function in figure 2.4. It is used for showing basic SOM visualization such as U-matrix(using 'umat' argument with value 'all' or ''), component planes (using 'comp' argument with value 'all' or ''), empty map for histogram visualization (using 'empty' argument) and showing color coding or clustering (using 'color' argument). Argument 'norm' defines if values are normalized (represented by 'n') or denormalized (represented with 'd'). Syntax for `som_show()` is given below:

```
som_show(sM, 'umat', 'all', 'empty', 'Labels', 'norm', 'd');
```

2.5 Using K-means with SOM

When we are using large data applications performance slows down based on the value of the initial weights that are randomly assigned while using SOM. Using SOM with the K-means algorithm provides accuracy and consistency because it only works with local optimum with random initial center points. Using batch K-means algorithm partitive clustering of data set and SOM's occurs. Since using the K-means initialization process is sensitive, eg. for each K with random initialization K-means runs 100 times and best partitioning is selected based on error criteria.

SOM toolbox has function called `som_kmeans()` to implement k-means algorithm. Syntax of `som_kmeans()` is given as below:

```
[codes,clusters,errors]=som_kmeans('batch',sM,k);
```

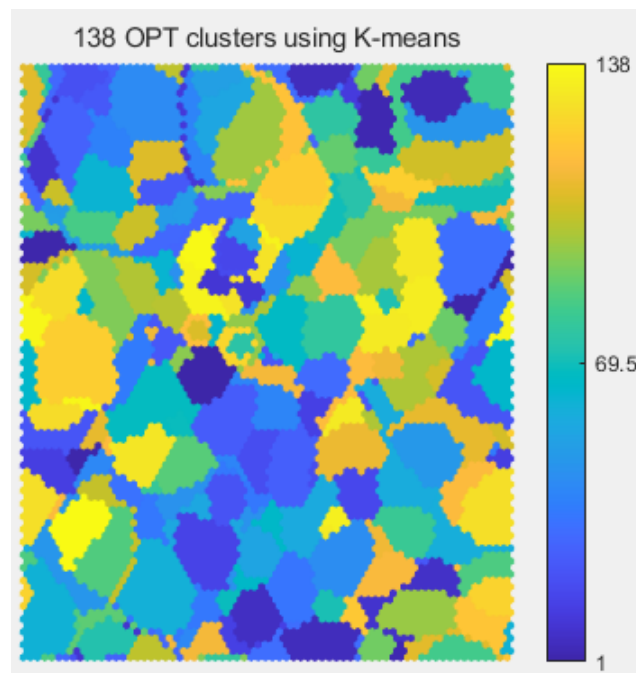


Figure 2.5 U-matrix using K-means with 138 clusters

Here, output arguments are codes that represent codebook vectors, clusters represent cluster numbers related to each cluster, and errors represent total quantization error for the

BaseM matrix	
Cell Number	Cluster Number
1	109
2	109
3	109
4	109
5	81
6	109
7	109
8	109
9	109
10	109
11	105
12	109

Table 2.2 Sample of placement of 138 clusters into first 12 cell numbers

data set. The first input argument is a method that we select for the type of algorithm. It could be either 'batch' or 'seq'. The second input argument is the data matrix and the third argument is k which is the total number of centroids. Figure 2.5 represents U-matrix using k-means clustering with 138 clusters. After selecting the best value for k, now it assigns values between 1 to 138 to all the selected BMU as shown in table 2.2.

Chapter Three

CLUSTERING QUALITY INDEX:

DAVIES-BOULDIN INDEX(DBI)

Yudhistira Arie Wijaya [10] proposed using the Davies-Bouldin Index (DBI) to find the best value of K for K-means. The rational of using DBI to choose K is to find the set of clusters least like each other. A simple definition of the similarity of clusters i and j is $R_{ij} = (S_i + S_j)/M_{ij}$, where S_n is the intra-dispersion of cluster n, (the average separation of its members from the centroid) and M_{ij} is the distance between the centroids of clusters i and j. Small dispersion and wide separation implies low similarity. With K clusters, calculate R_{ij} for the $K(K - 1)/2$ distinct ordered pairs. For each cluster i, find $R_{imax} = \max(R_{ij})$. DBI can be defined as the simple average of R_{imax} over the K clusters. Algorithm 2 is an implementation of this idea.

SOM has an inbuilt function to calculate DBI for clustering. The syntax of that function is as below:

```
[c,p,err,ind] = kmeans_clusters(sD, [n_max], [c_max]);
```

Where input argument 'sD' is a data structure for a map, '[n_max]' specifies the maximum number of clusters (it is an optional argument), and '[c_max]' for a maximum number of k-means runs. As output parameters we receive 'c' which is centroids for each cluster, 'p' contains cluster indices for each cluster, 'err' argument gives the squared sum of errors for

Algorithm 2 Davise-Bouldin's Index

Require: $R_{i,j}$ // measure of good clustering scheme,

$M_{i,j}$ // separation between the ith and the jth cluster

S_i // the within cluster scatter for cluster i

Ensure: $R_{i,j} \geq 0$

$$R_{i,j} = R_{j,i}$$

When $s_j \geq s_k$ and $M_{i,j} = M_{i,k}$ then $r_{i,j} \geq r_{i,k}$

When $s_j = s_k$ and $M_{i,j} \leq M_{i,k}$ then $r_{i,j} \geq r_{i,k}$

$$R_{i,j} = \frac{s_i s_j}{M_{i,j}}$$

$$D_i \equiv \max_{j \neq i} R_{i,j}$$

if N is total number of clusters:

$$DB = 1/N \sum_{i=1}^N D_i$$

each value of cluster, and 'ind' specifies Davies-Bouldin index value for each cluster.

Davies-Bouldin's validity index is utilized for partitioning with a different number of clusters to choose the best clustering. Usually, by using DBI we try for reducing the distance between a point in a cluster, and at a similar time, we also try to increase the inner-cluster distance. Based on the proposed centroids DBI method try to evaluate clustering outcomes. In practice, it is being used as a clustering metric and it subdivides the validation into two categories: 1) internal validation and 2) external validation.

Figure 3.1 shows the DBI for up to K=160 in the K-means clustering or the 6417 code-books in the application of SOM to the data set of vectors from natural language processing of CVEs discussed in Chapter 2. (insert Figure 3.1) Table 3.1 gives values of DBI for individual K values between 130 and 140, which show that the smallest DBI occurs for k=138.

The Sum of squared differences between each observation and its group's mean is called the Error Sum of Squares(SSE) also called the Squared sum of errors. It can be utilized to measure variation within a cluster. A smaller value for SSE indicates a tight fit between

Ind matrix	
Value for k	DBI index
131	1.1195
132	1.1403
133	1.1398
134	1.1282
135	1.1317
136	1.1308
137	1.1286
138	1.0798
139	1.1500
140	1.1443

Table 3.1 Sample of DBI matrix with index values using K-means from 131 to 140

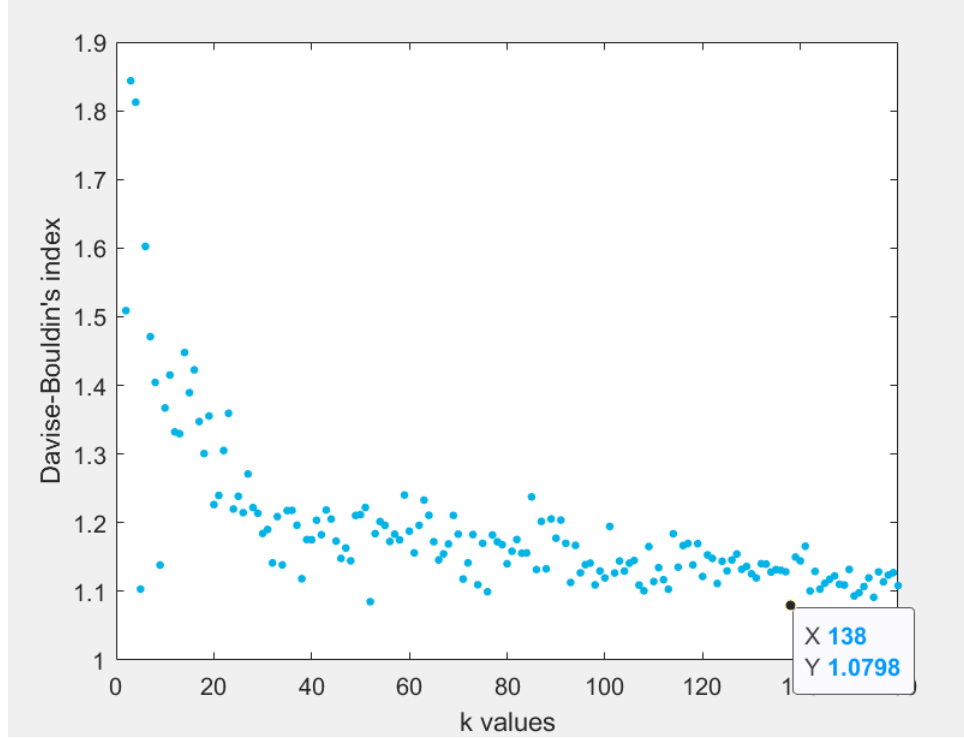


Figure 3.1 K-means clustering using Davise-Bouldin's Index with an extended range search suggested best value for k

model and data. Figure 3.2 shows the value of k on the X-axis and the Squared sum of errors on the Y-axis. By plotting values received in the 'err' output argument of the `kmeans_clusters()` function, we can visualize the Squared sum of errors highlighted for each cluster.

After selecting best k using DBI, we mapped the CVE label with matching CWE Id, cluster number, and Best Matching Unit (BMU). Table 3.1 shows a little snippet of matrix `Result_label`.

Result label matrix			
CVE ID	CWE label	Cluster Number	Cell Number
CVE-1999-0001	CWE-20	133	5249
CVE-1999-0002	CWE-119	22	5723
CVE-1999-0007	CWE-327	79	1740
CVE-1999-0027	CWE-119	22	6286
CVE-1999-0179	CWE-17	101	1990
CVE-1999-0226	CWE-19	89	5531
CVE-1999-0227	CWE-264	116	4330
CVE-1999-0265	CWE-20	30	4144
CVE-1999-0284	CWE-120	99	5076
CVE-1999-0332	CWE-119	22	6003
CVE-1999-0344	CWE-264	19	2945
CVE-1999-0348	CWE-200	109	7
CVE-1999-0349	CWE-119	64	5909
CVE-1999-0366	CWE-287	65	3233
CVE-1999-0372	CWE-200	82	4930
CVE-1999-0385	CWE-120	99	5074
CVE-1999-0387	CWE-255	86	2018
CVE-1999-0453	CWE-200	105	6335
CVE-1999-0491	CWE-94	126	3746
CVE-1999-0496	CWE-264	19	3227

Table 3.2 Result label matrix with first 20 records

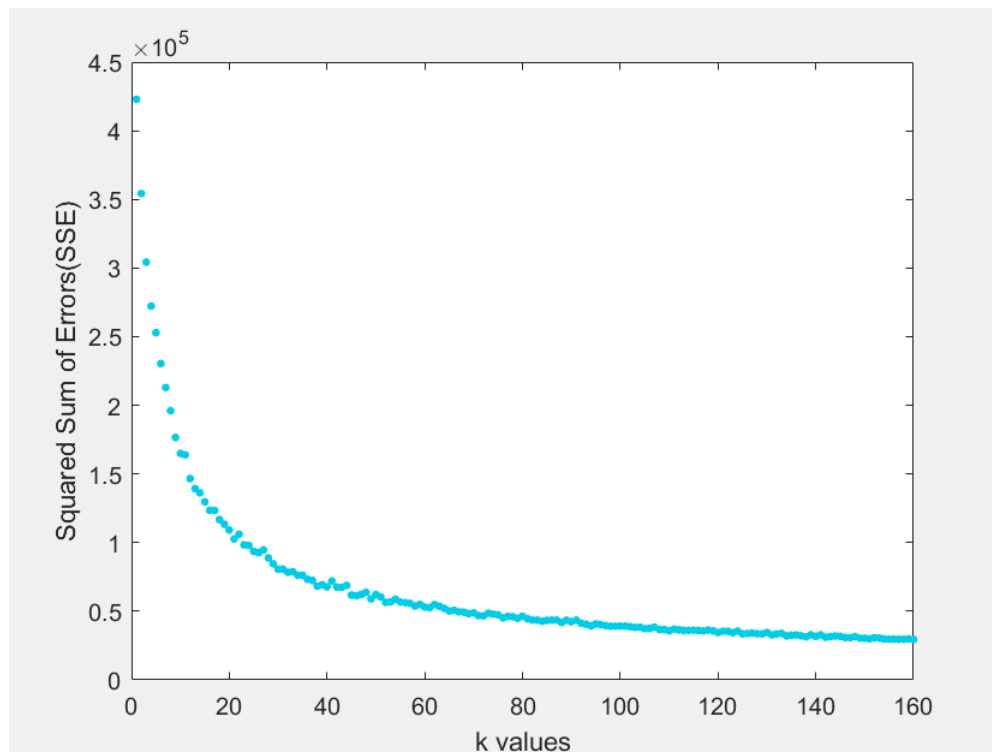


Figure 3.2 Squared sum of errors for selecting best k

Chapter Four

CLUSTERING RESULTS AND ANALYSIS

As mentioned by Kristina P. Sinaga, clustering is the best method to characterize similarity and dissimilarity among data.[11] Cluster analysis is the process of combining data into subgroups in a way that a particular node is closer to the node with similar characteristics to other dissimilar nodes which are outside of the cluster.[12] Generally, the distance between nodes is calculated based on measures such as Euclidean distance, and Manhattan distance to evaluate whether data points are similar or not.

4.1 Clustering results

Based on the Davies-Bouldin index ref, we chose 138 as the best value of k to use in k-means clustering of nodes in the SOM output array. On the basis of their characteristics CVEs are distributed into multiple clusters based on similarity or dissimilarity.

4.1.1 Bar graph representation of clusters

A bar chart is a graphical way to represent data point distribution or sometimes it is used to view differently grouped data based on their characteristics and perform a comparison

between the value of the metric. It helps in understanding frequency counts for multiple levels of variables. Generally, histograms are used to represent continuous data whereas bar charts are used to show categorical or nominal data point distribution. Because the bar chart shows counts of categories for our data, they are not affected by extreme values. It can also point to incorrect values in our data. To create bar graphs MATLAB uses a function called `bar(x,y)`. It usually has a minimum 1 parameter, it draws the bar at a location specified by X-axis. We can also add different properties such as width, style, color, etc.

Figure 4.1 shows an example of a bar chart for cluster 6 data points where the X-axis shows the CWE labels that belong to cluster 6 and the Y-axis shows the total count of data points that belongs to a particular CWE label. Some of the CWE labels that are included in cluster 6 are CWE-16, CWE-18, CWE-22, CWE-264, CWE-269, CWE-284, CWE-310, CWE-362, CWE-59, CWE-706. In the figure some of the bar graphs as multiple CWE labels; for example, CWE-362 and CWE-264 are combined because in the CWE hierarchy it is possible that based on characteristics single CVE has more than one CWE label assigned to it. Based on the bar graph, We can derive that CWE-59 has more than 550 CVE data points that belong to cluster 6.

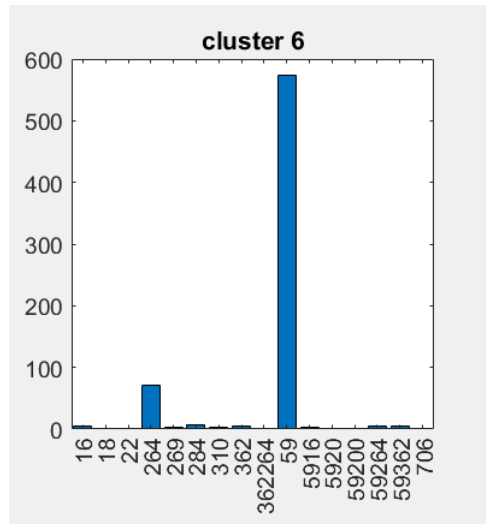


Figure 4.1 Bar graph for cluster 6

Figure 4.2 shows the distribution of CVEs based on CWE labels in the first 10 clusters. As highlighted in the figure we can determine that cluster 5 has more than 98% of the total population it belongs to CWE-416. All other elements are considered in the 2% remaining population of cluster 5. Similar way, in cluster 6 94% population from CVE data points goes to prominent peak CWE-59 and all other CWE labels are part of the remaining 6% population. From the distribution of counts over CWE label, one can reasonably conclude that association with CWEs 416 and 59 are the primary reasons for CVE membership in clusters 5 and 6, respectively. The secondary peaks, CWE 399 and 264 in clusters 5 and 6 respectively, might play some role in the assignment of CVEs to clusters 5 and 6, but the remaining peaks can be dismissed as random noise from natural language processing of the text of CVEs. In section 4.2 we examine individual BMUs of cluster in terms of primary, secondary and noise contributions. Cluster 9 has only one label that is CWE-79 which shows that there is no noise contribution involved in the distribution of CVE for this specific cluster.

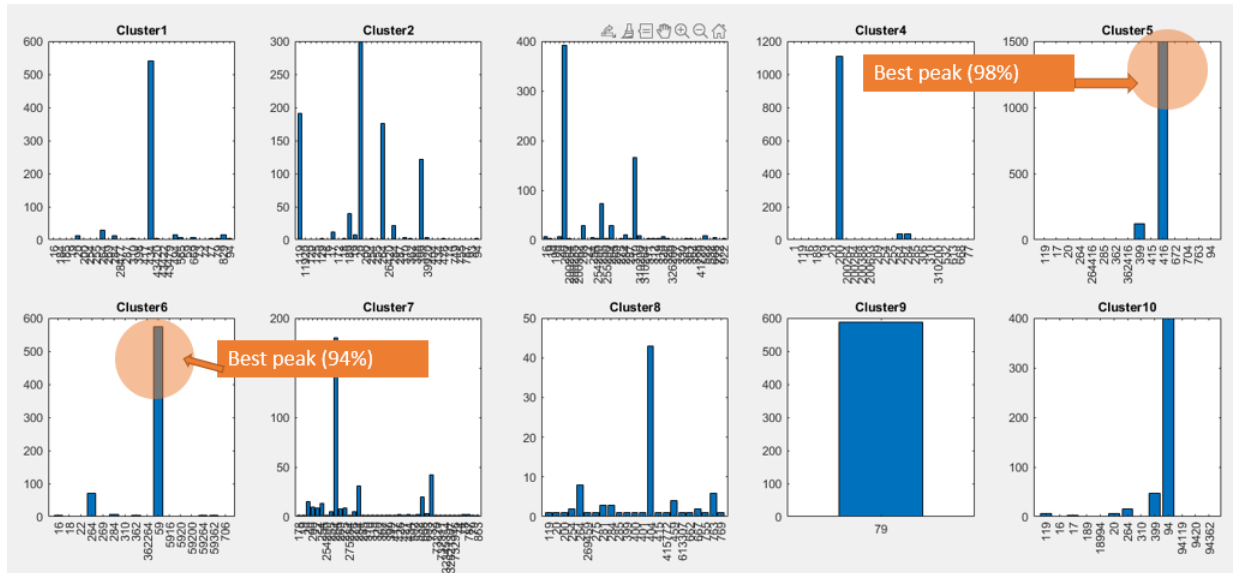


Figure 4.2 Bar graph for first 10 clusters

Figure 4.3 shows the distribution of CVEs based on CWE labels from clusters 11 to 20.

In the figure, it is emphasized that cluster 12 has 92% of the total population of it belongs to CWE-416 with more than 50 counts of elements. All other elements are considered in the remaining 8% population. Similar way, in cluster 16 99% population from CVE data points which is close to 500 elements that go to the dominant peak in this case CWE-94 and all other CWE labels are considered under the remaining 1% population.

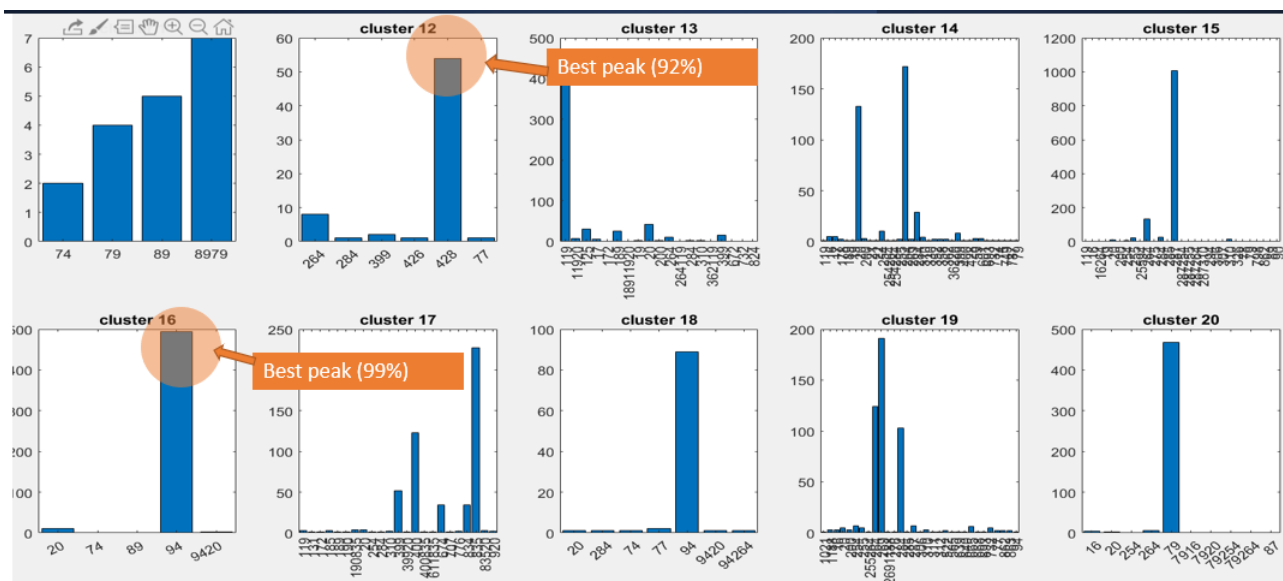


Figure 4.3 Bar graph for cluster 11 to 20

Figure 4.4 shows a bar graph for clusters 121 to 130. If we look closely at cluster 124, CWE-22 has a peak of almost 99%. The CWE's seen in that bar graph's are CWE-200, CWE-21, CWE-22, CWE-264, and CWE-79. CWE-200 is a weakness where exposure of sensitive information to an unauthorized actor can occur. CWE-21 had been deprecated but it was being used for pathname traversal and equivalence errors. CWE-22 is also considered a path traversal weakness which is an improper limitation of a pathname to a restricted directory. CWE-264 is defined as a weakness for permissions, privileges, and access controls. CWE-79 is utilized for improper neutralization of input during web page generation which is also called Cross-site scripting. The above-mentioned CWEs for cluster 124 is a members of CWE-635 which includes weaknesses originally used by NVD from 2008 to 2016.

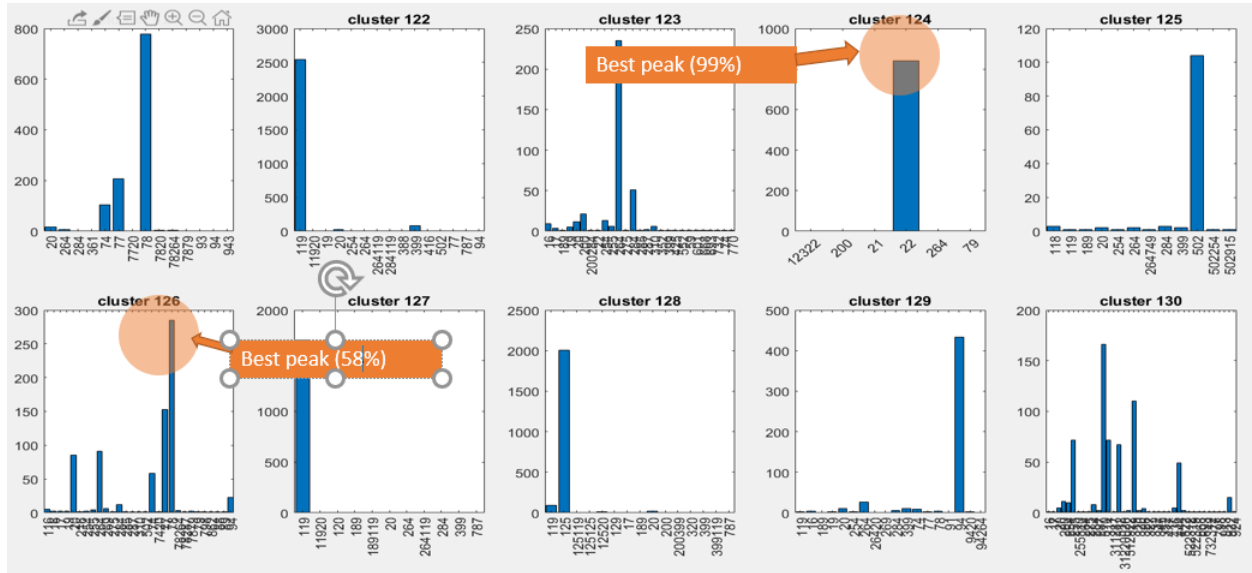


Figure 4.4 Bar graph for cluster 121 to 130

4.2 Contribution of CWEs to BMUs of Cluster 6

The main purpose behind cluster analysis is to derive observations from every group concerning objects or attributes of interest. It is used to combine different variables in different and homogeneous groups. For example, on basis of responses retrieved in the questionnaire draft revising questionnaire. With cluster analysis, we group questions to verify repeated questions and remove them to achieve a good response rate for the final questionnaire.

Figure 4.5 shows the contributions of CWEs 59 and 246 to the BMUs of the CVEs that are members of cluster 6. The X-axis of Figure 4.5 show all the BMUs of CVEs that became members of cluster 6. The Y axis shows the proportion of CVEs with labels CWE 59 and 264 that became members of cluster 6 via the indicated BMU. The 10 full blue bars (0 to 1) show BMUs for CWE 59-labeled CVEs only. Similarly, the 2 full orange bars were BMUs for CWE 264-labeled CVEs only. Comparing the text of CVEs in these 2 groups might reveal aspects captured by natural language processing that distinguish them. Most cells in this region of the SOM output were BMUs for a combination of CWE 59- and 264-labeled CVEs, with a predominance of CWE 59 as revealed by the bar graph of cluster 6 in Figure 4.2.

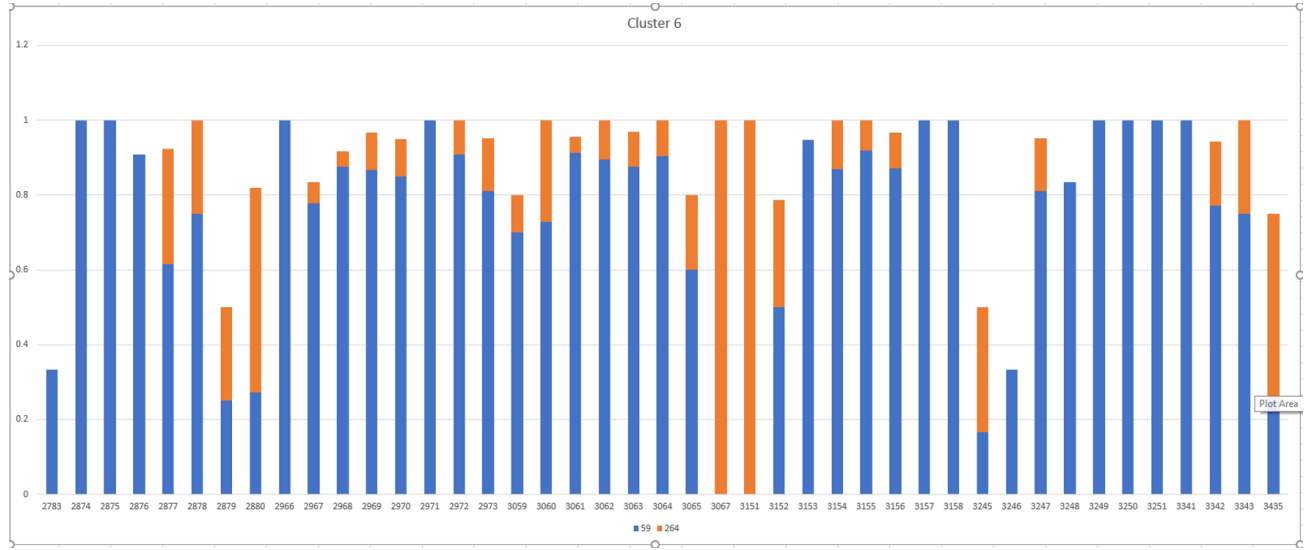


Figure 4.5 Role of CWEs 59 and 264 in the BMUs of members of cluster 6

Many of the bars in the bar graph are incomplete but with a level of incompleteness that we can associate with a noise component in the instance. In those cases where the incompleteness is significant, close to four in this cluster, it would be important to figure out what CWEs are involved in completing that particular bar. This is specifically interesting when the completion is by a CWE that is dominant in a different cluster. It suggests a relationship between that CWE and the dominant CWEs of cluster 6 that would cause the instance to be classified in cluster number 6 rather than be classified in the cluster where the completing CWE is dominant. The wording of the CVE for that instance might be able to give an insight into the nature of the relationship.

Table 4.2 represents all the CVE-Ids and BMU-index involved in constructing the bar graph shown in figure 4.2. As per MITRE website ¹ CWE-59 is defined as improper link resolution before file access('link following') in short its also called insecure temporary file. That means the File is accessed through software using filename but it does not resolve the resource of the file. In a hierarchical relationship, it is a member of CWE ID 1219 which is

¹<https://cwe.mitre.org/data/definitions/59.html>

Data of cluster 6			
CVE-Id	Cluster Number	CWE-label	Cell Number
CVE-2008-4104	6	CWE-59	2783
CVE-2007-2978	6	CWE-59	2874
CVE-2008-3227	6	CWE-59	2874
CVE-2008-4694	6	CWE-59	2875
CVE-2008-3456	6	CWE-59	2876
CVE-2008-7247	6	CWE-59	2876
CVE-2009-1867	6	CWE-59	2876
CVE-2010-1160	6	CWE-59	2876
CVE-2011-0402	6	CWE-59	2876
CVE-2014-1639	6	CWE-59	2876
CVE-2014-4996	6	CWE-59	2876
CVE-2014-6407	6	CWE-59	2876
CVE-2014-7206	6	CWE-59	2876
CVE-2015-1038	6	CWE-59	2876
CVE-1999-0981	6	CWE-59	2877
CVE-2007-5969	6	CWE-264	2877
CVE-2007-6692	6	CWE-59	2877
CVE-2008-2936	6	CWE-264	2877
CVE-2008-3329	6	CWE-59	2877
CVE-2008-4162	6	CWE-59	2877

Table 4.1 Data distribution for cluster 6(first 20 records)

"File handling issues".² CWE-264 is related to permissions, privileges and access controls. It is a member of CWE-635 which contains weaknesses used by NVD from 2008 to 2016. This CWE has been made obsolete because other sources already cover appropriate weakness that maps CWE-264 into them.

Now, the reason for both CWE-59 and CWE-264 being included in the same cluster is essential to find out. Based on our discussion above we can observe that both the CWEs are members of CWE-635 and CWE-1345. CWE-1345 comes under the "Broken Access Control" category which is part of the Open Web Application Security Project (OWASP) top ten 2021 category. OWASP is a foundation that works for protecting web application security. We already discussed in the previous paragraph that CWE-635 has been made obsolete and it's no longer under use.

²<https://cwe.mitre.org/data/definitions/264.html>

Chapter Five

CASE STUDY OF CLUSTERS USING U-MATRIX

To learn structure of each cluster and visualize their distribution on a map we have introduced a U-matrix representation for some of the clusters. It shows the location of a singular cluster on the U-matrix. Based on that we can determine the adjacent clusters and also the reason behind why they are adjacent to each other or far from each other. We have analyzed 138 clusters and based on their 2 prevalent peak percentage value as shown in Table 5.1. Clusters are ranked by the percentage of membership in the most dominate peak. The 1st 10 entries in Table 1 are cluster with only one peak, 7 of which are composed of CVEs with label 79 and 3 are composed of CVEs with label 89. Table 5.1 ends with 5 examples of clusters with nearly pure membership; 3 more examples of 79, 1 more example of 89, a cluster 85 with all but 2 members with label CWE 352. Hence, we find at least 10 clusters with essentially pure membership of CVEs labeled by CWE 79. In section 5.1 we explore the idea that these clusters are adjacent in the SOM output array, justifying the idea of calling them a “supercluster”

After we received the optimal value for k which is 138 we picked the two prominent peaks from the total number of CWEs that belongs to a specific cluster.

table 5.1 shows percentage calculations based on two dominant peaks represented in the

Percentage-data matrix						
Cluster no.	Total-num-CVE	1st CWE label	Count - label1	2nd CWE label	Count-label2	Percentage
9	587	CWE-79	587	0	0	100%
27	1996	CWE-79	1996	0	0	100%
38	900	CWE-79	900	0	0	100%
39	610	CWE-89	610	0	0	100%
51	584	CWE-89	583	CWE-264	1	100%
54	1291	CWE-89	1289	CWE-89, CWE-20	2	100%
61	1745	CWE-79	1744	CWE-352	1	100%
73	783	CWE-79	781	CWE-352	2	100%
83	657	CWE-79	657	0	0	100%
108	853	CWE-79	853	0	0	100%
42	2769	CWE-79	2764	CWE-20	2	99.89%
97	2998	CWE-89	2991	CWE-89, CWE-20	2	99.83%
102	472	CWE-79	470	CWE-20	1	99.79%
85	1319	CWE-352	1314	CWE-264	2	99.77%
104	778	CWE-79	774	CWE-20	2	99.74%

Table 5.1 Total percentage computation based on data distribution in two prominent peaks(first 15 records in descending order)

bar graph for each cluster. Table's first column represents the cluster number, the second column for the total number of CVEs that are contributing to that cluster, column three has the CWE label with the first predominant peak, column four has the total count of elements for the first peak label, column five has second CWE label with predominant peak, column six has a total count for CWE label 2 with dominant peak and last column shows percentage calculation based on a total count for label 1 and label 2 divided by the total number of CVE for each cluster. Percentage calculation is sorted in descending order to view pure clusters.

We can derive from table 5.1 that there is a total of 11 clusters which are pure clusters with only one dominant CWE label. For example, if we see from the column the total number of CVEs that belongs to clusters 9, 27, 38, 39, 51, 54, 61, 73, 83, 108 all of them have most of their elements contributing to only one CWE label.

5.1 Understanding structure of supercluster CWE-79

A Supercluster is the grouping of adjacent clusters based on the likeness of features in between these clusters. In Figure 5.1 we can see all the clusters for example 9, 24, 27, 38, 42, 49, 61, 73, 83, 102, 104, 108, 20, 87, 29, 50 which have CWE-79 as dominant peak as we've seen in table 5.1 of chapter 5. Together the entire cluster can be called a supercluster for CWE-79. When we calculated the space between all the adjacent clusters from cluster 9 (demonstrated by the dark blue color in figure 5.1) as shown in table 5.2 it is visible how close they are to each other. For example, the distance from the centroid of cluster 9 to the centroid of cluster 24 is 2.2365, the distance from cluster 9 to cluster 27 is 1.2940, and the distance from cluster 9 to adjacent cluster 61 is 2.0085, etc. So based on distance measures we can derive that cluster 27 is relatively more close to cluster 9. If we check the distance between the centroid of cluster 9 and cluster 138 it is 13.9453. But the most adjacent cluster to cluster 9 in a group of CWE-79 clusters is cluster 104 since it has the lowest distance.

CWE-79 is the first one in the list of 2020 top 25 CWEs with a score of 46.82 based on the

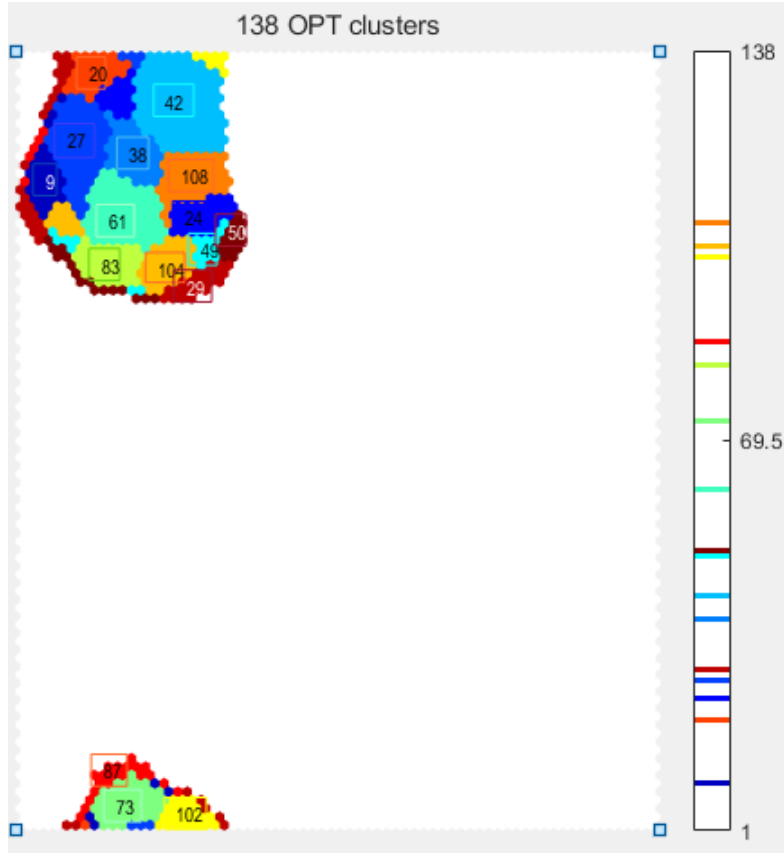


Figure 5.1 U-matrix of clusters only containing CWE-79 for prominent peak percentage more than 50%

Common Vulnerability Scoring System (CVSS) score. CWE-79 is described as "Improper neutralization of input during web page generation" which is also considered as cross-site scripting[13]. Some of the justifications why cross-site scripting vulnerability can be exploited for attack are as below:

- 1) During web requests some of the untrusted data evolve into part of a web application.
- 2) Untrusted data becomes available through a dynamically developed web application.
- 3) If while we are generating page application does not prevent data with a content executable with a web browser it becomes challenging to handle.
- 4) Accidentally user accesses a web page containing a malicious script that permits injecting untrusted data into the system.

Distance matrix for CWE-79	
Cluster number	Neighbouring Distance
104	1.0781
73	1.1087
27	1.2940
20	1.6427
49	1.6471

Table 5.2 Sample of distance matrix of adjacent clusters to cluster 9(in ascending order top five based on minimum distance)

5) webserver transmits a script to a web browser, a user executes a malicious script from the webserver domain and compromises the security of the system.

6) Web server's same-origin policy is violated and the user can not run scripts from one domain to another domain to access resources.

Now, if we explore these clusters we can pinpoint the CVEs that are part of CWE-79 and at peak in clusters 9, 24, 27, 38, 42 are adjacent according to distance. For example, as we can witness in table 5.3 the CWEs that are part of cluster 9 are CVE-2002-2364, CVE-2003-0310, CVE-2003-0712, CVE-2003-1334, and CVE-2003-1539. If we closely look at the facts of the first CVE in a table which is 'CVE-2002-2364' it is described as Cross-site scripting (XSS) vulnerability in PHP and it allows remote attackers to inject arbitrary web script or HTML via a help ticket. Now, the second CVE in the table is 'CVE-2003-0310' and it is illustrated as Cross-site scripting (XSS) vulnerability in articleview.php which allows remote attackers to insert arbitrary web script. Based on both of the above depictions we can decide that they both are related to Cross-site scripting (XSS) vulnerability and it is being injected arbitrarily through a script.

Now if we will closely understand CVEs for cluster 104 because it has the most adjacent

Sample CVE's of cluster 9			
CVE-Id	CWE-label	Cluster Number	BMU-index
CVE-2002-2364	CWE-79	9	111
CVE-2003-0310	CWE-79	9	293
CVE-2003-0712	CWE-79	9	743
CVE-2003-1334	CWE-79	9	297
CVE-2003-1539	CWE-79	9	204

Table 5.3 Snippet of cluster 9 with CWE-79 and related CVEs

Sample CVE's of cluster 104			
CVE-Id	CWE-label	Cluster Number	BMU-index
CVE-2002-1852	CWE-79	104	1329
CVE-2002-2422	CWE-79	104	486
CVE-2003-0801	CWE-79	104	1422
CVE-2004-0067	CWE-79	104	1421
CVE-2005-2022	CWE-79	104	1608

Table 5.4 Snippet of cluster 104 with CWE-79 and related CVEs

cluster corresponding to other clusters to identify similarities between cluster 9 and cluster 104. The very first CVE in table 5.4 is 'CVE-2002-1852' which is described as Cross-site scripting (XSS) vulnerability in an individual app that allows remote attackers to inject arbitrary web script or HTML via either the URL or a parameter used to test. The second CVE shown in table 5.4 is 'CVE-2002-2422' which is also Cross-site scripting (XSS) vulnerability but in various applications that authorize remote attackers to inject arbitrary web script or HTML via a URL, which inserts the script into the resulting error message. Both of the above-mentioned CVEs are for Cross-site scripting (XSS) vulnerability and it injects malicious scripts into applications and violates the privacy and security of web applications.

Case study of supercluster CWE-352

CWE-352 is used to express the web application vulnerability Cross-Site Request Forgery (CSRF) in which a web application is incapable to verify a valid request made by a user. The attacker can easily take advantage of this situation and trick a client to make an unintentional request. It can result in a user can compromise the exposure of data or obligatory code execution and generating an exploit into an attack.

In Figure 5.2 we can see all the clusters for example 138, 85, and 47 which have CWE-352 as the prevailing peak. When we calculated the distance between all the adjacent clusters it was visible that all of them are nearby. For example, the distance from the centroid of cluster 138 to the centroid of cluster 85 is 1.81. The distance between the centroid of cluster 138 to the centroid of cluster 47 is 2.3364.

Table 5.4 shows some of the CVEs that are part of cluster 138 and have the CWE-352 CWE label. The first CVE in the table is ¹ 'CVE-2007-4822' that is described as CSRF vulnerability in the device managing interface in an application that allows remote attackers to modify configuration changes as an administrator via HTTP requests to particular HTML pages in the res parameter with an inp req parameter. if we look at the second CVE in the

¹<https://nvd.nist.gov/vuln/detail/CVE-2007-4822>

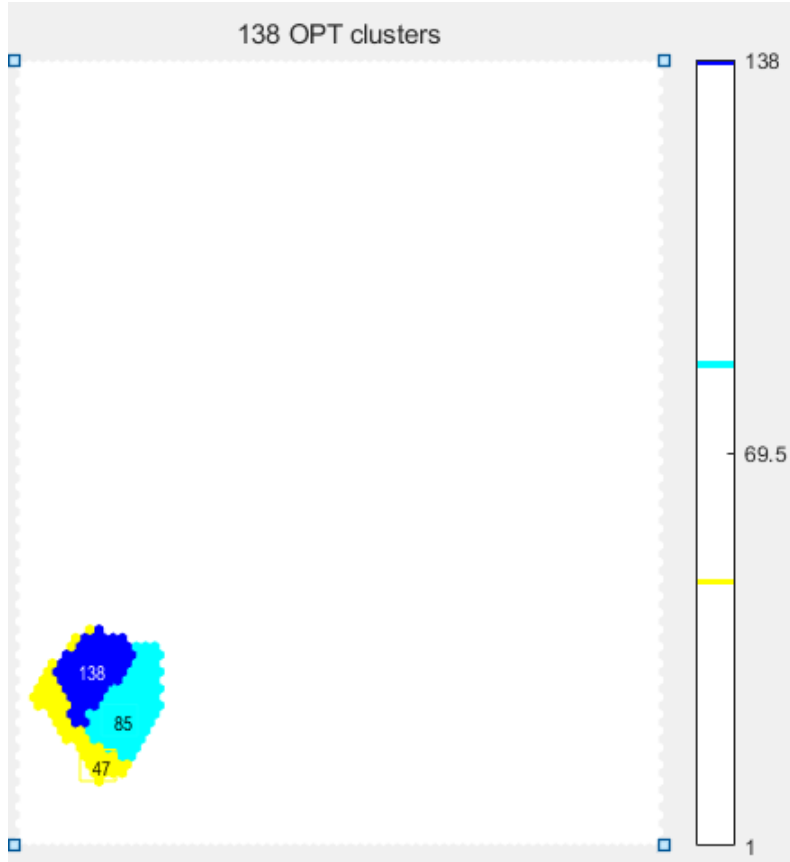


Figure 5.2 U-matrix of clusters only containing CWE-352 for prominent peak percentage more than 50%

table 'CVE-2007-5384' it is used for Multiple CSRF vulnerabilities in the router which was used for Hub that allowed remote attackers to perform actions as administrators via unknown POST requests, as shown by enabling an inbound remote-assistance HTTPS session on TCP port 51003. Both of the above-mentioned CVEs are mainly CSRF vulnerabilities with some kind of device and are allowing to change configuration parameters of that device unintentionally.

Case study of supercluster CWE-89

CWE-89 is generally used for improper neutralization of special elements used in an SQL Command also called 'SQL Injection'. This weakness describes if SQL syntax with user-

Sample CVE's of cluster 138			
CVE-Id	CWE-label	Cluster Number	BMU-index
CVE-2007-4822	CWE-352	138	1004
CVE-2007-5384	CWE-352	138	908
CVE-2007-5828	CWE-352	138	820
CVE-2008-0575	CWE-352	138	818
CVE-2008-1260	CWE-352	138	912

Table 5.5 Snippet of cluster 138 with CWE-352 and related CVEs

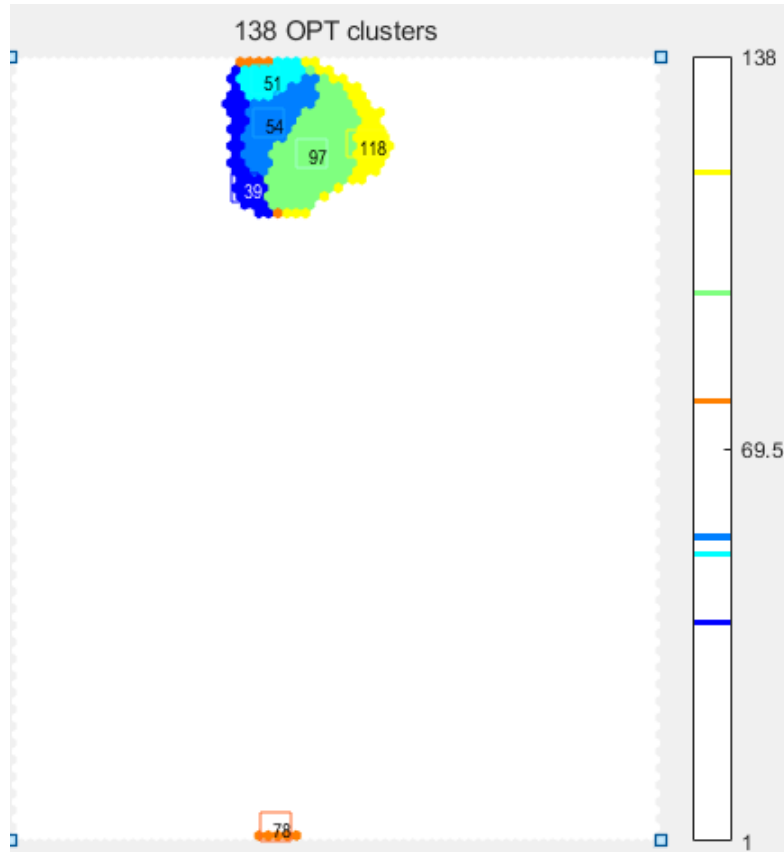


Figure 5.3 U-matrix of clusters only containing CWE-89 for prominent peak percentage more than 50%

Distance matrix for CWE-89	
Cluster number	Neighbouring Distance
54	1.4698
51	1.6966
97	1.8940
78	2.5402
118	2.9564

Table 5.6 Sample of distance matrix of adjacent clusters to cluster 39(in ascending order top five based on minimum distance)

controllable inputs does not contain enough quoting or removal in SQL query then those inputs are interpreted as SQL, not as user data. It can be done easily by modifying query logic to disregard security checks or maybe inserting extra statements that can update the back-end database. It is one of the common weaknesses of data-driven websites.

In Figure 5.3 we can see the clusters for example 39, 54, 51, 97, 118, and 78 which have CWE-89 as the dominant peak. When we computed the distance between all the neighboring clusters it was visible that all of them are close to each other. The distance between the centroid of cluster 39 to the centroid of cluster 54 is 1.4698 which is displayed in table 5.6. The distance between the centroid of cluster 39 to the centroid of cluster 51 is 1.6966. According to all the adjacent distances mentioned in the table, we can conclude that cluster 54 is closest to cluster 39.

Case study of supercluster CWE-22

CWE-22 is depicted as improper limitation of a pathname to a restricted directory also known as 'Path Traversal' weakness. Exterior input is used by software to create a pathname that is going to figure out the file or directory location under the restricted parent directory but

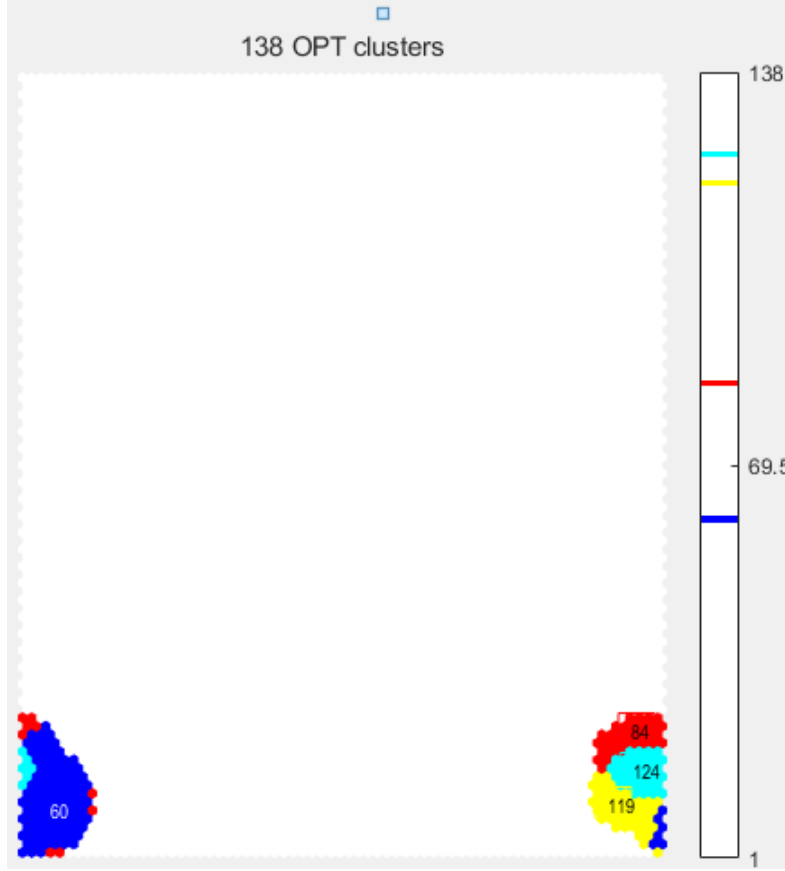


Figure 5.4 U-matrix of clusters only containing CWE-22 for prominent peak percentage more than 50%

due to software being unable to decide the correct location of the file or directory it goes to a location outside of the restricted directory.

Figure 5.4 represents clusters 60, 124, 119, and 84 which have CWE-22 as the dominant peak. As we can see because the shape of the map is toroid so half of the cluster is shown on the bottom left-hand side of the U-matrix and wraps around another half on the bottom right-hand side. When we calculated the adjacent cluster distance between all of them it was observed that all of them are close to each other. The distance between the centroid of cluster 60 to the centroid of cluster 124 is 2.3560 as shown in table 5.7. The distance between the centroid of cluster 60 to the centroid of cluster 119 is 1.9012.

Distance matrix for CWE-22	
Cluster number	Neighbouring Distance
119	1.9012
124	2.3560
84	2.6380

Table 5.7 Sample of distance matrix of adjacent clusters to cluster 60(in ascending order top three based on minimum distance)

Case study of supercluster CWE-200

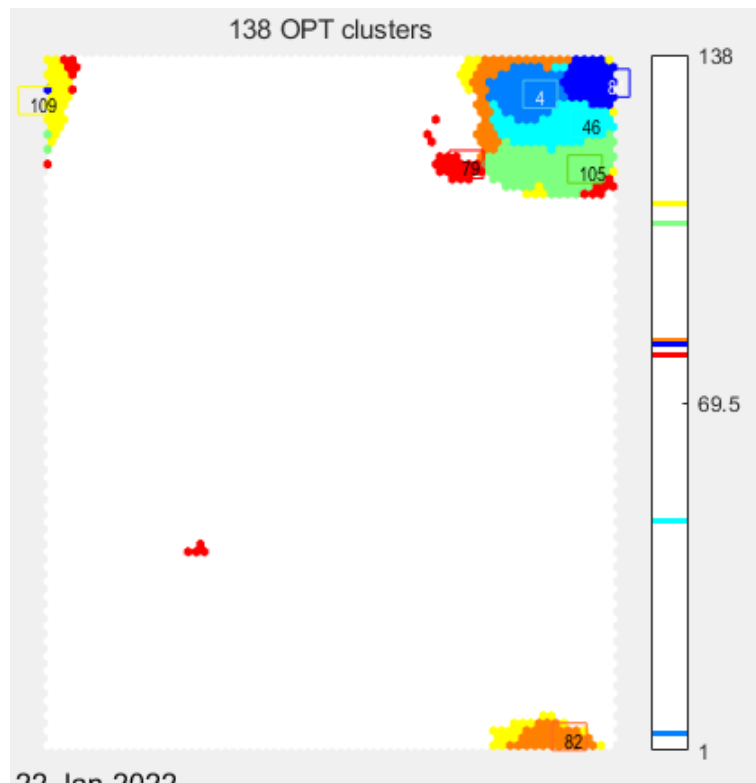


Figure 5.5 U-matrix of clusters only containing CWE-200 for prominent peak percentage more than 50%

CWE-200 is portrayed as exposure of sensitive information to an unauthorized actor's

Distance matrix for CWE-200	
Cluster number	Neighbouring Distance
46	2.0570
4	2.1223
82	2.5955
105	3.1769
109	3.2827

Table 5.8 Sample of distance matrix of adjacent clusters to cluster 81(in ascending order top five based on minimum distance)

weakness. In this weakness, users get involuntary access to the susceptible product information that he/she is not authorized to have. It includes private or personal information such as personal text messages, and financial, health, or geographical information. Somehow business secrets or network configuration gets compromised.

Figure 5.5 represents clusters 81, 4, 46, 105, 109, 82, 3, and 79 which have CWE-200 as the dominant peak. Upon calculating the adjacent cluster distance between them it is seen that all of them are close to each other. The distance between the centroid of cluster 81 to the centroid of cluster 4 is shown in table 5.8 which is 2.1223. The distance between the centroid of cluster 81 to the centroid of cluster 46 is 2.0570.

Case study of supercluster CWE-119

CWE-119 weakness is induced by improper restriction of procedures within the bounds of a memory buffer. A memory buffer is used to read and write information to a memory location for specific software. But in this weakness software performs read and write operations outside of the intended boundary of the memory buffer. When languages allow direct addressing of memory locations they automatically verify the location of the memory buffer

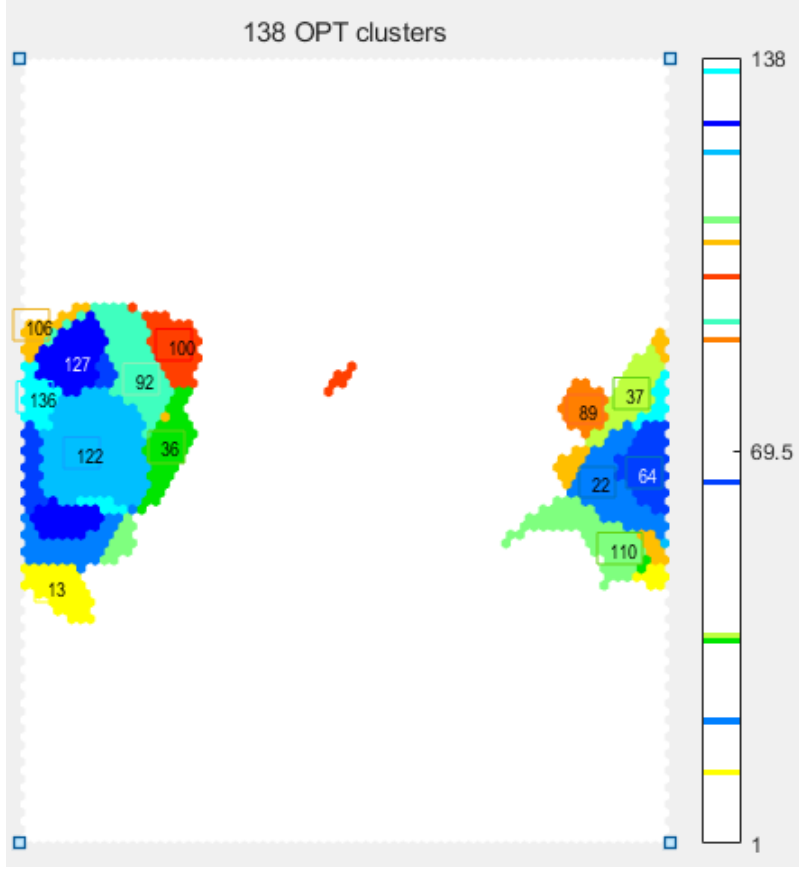


Figure 5.6 U-matrix of clusters only containing CWE-119 for prominent peak percentage more than 50%

that is being referred. Attackers give access to the user to a location that might have been associated with some other variables and they can modify the intended control flow or read critical information.

Figure 5.6 represents clusters 127, 64, 22, 122, 136, 92, 110, 37, 13, 106, 89, 100, 36 which has CWE-119 as prominent peak. In the figure, we can see that the super is cluster is divided into parts but because the u-matrix is in toroid shape it is wrapped around, one patch of clusters is on the left-hand side and the other one is on the right-hand side. Upon calculating the adjacent cluster distance between them it is visible that all the clusters are close to each other.

As described in table 5.9, the minimum distance is between 127 (highlighted with blue

Distance matrix for CWE-119	
Cluster number	Neighbouring Distance
136	1.7368
64	1.7966
22	1.8707
92	2.0454
122	2.3579

Table 5.9 Sample of distance matrix of adjacent clusters to cluster 127(in ascending order top five based on minimum distance)

color on map) and 136 (stressed with cyan color on map) which is 1.7368. The distance between the centroid of cluster 127 to the centroid of cluster 64 is 1.7966. The distance between the centroid of cluster 127 to the centroid of cluster 22 is 1.8707.

Chapter Six

FUTURE RESEARCH WORK

In conventional methods for capturing scattering of data in clusters, they are unable to measure the distribution in multidimensional data. Using the David-Bouldin index (DBI) for approximate estimation of the distance between clusters and their distribution to obtain a satisfactory quality partition of clusters is an effective approach to overcome the drawbacks of those traditional methods.

In this research, we focused on analyzing the cluster data and interpreting patterns represented so we can provide helpful information for mitigating vulnerabilities or weaknesses. Data space is divided using Vector quantization into subareas relevant to clusters, special cases, and outliers. By using qualitative information delivered by visualizations we can select subareas and subspaces from data and conduct quantitative analysis. Automatic post-processing can be applied in the future for providing relevant and easily comprehensible information about maps.

In the last chapter, the case studies, we noticed some of the clusters have similar CWE labels and we proposed the idea of creating superclusters with the clusters having similar CWE labels to create superclusters the proportion of each CWE in clusters should be more than 90%. Constructing superclusters from the clusters with similar CWE-ID will allow the interpreting of weakness effectively.

Another technique for discovering a solution for a similar problem is to define a function,

like the Unified Distance Matrix (UMAT), on the SOM output array and use graph-theory approaches to find local minima in the digital contour map of the function. These local minima can be used to define centroids of clusters in the tabular data. Tracing these centroids back to the annotation will allow us to compare them with the K-means representation. All these unsupervised machine-learning techniques involve parameters, like the length of the SOM output array, which will optimize to test their effect on clustering and, eventually, the significance for the cyber security of the patterns they reveal.

Chapter Seven

CONCLUSIONS

The primary focus of this paper is to obtain together many visualization methods for Self-Organizing Map and combine them with K-means for the visualization of multidimensional vector data set. In this thesis, we are using vector data set processed from the natural language processing model V2W-BERT. The original vectored data set was huge (close to 4 GB) in size so we emphasized compressing the data set close to 10 fold data compression. We focused on exploring a way to analyze the patterns in CVEs and CWEs mapping based on their attributes and used a Self-organizing Map for mapping multidimensional data to lower dimensions. The main reason for using SOM is to help us visualize data in an effective way and for better interpretation of characteristics for CVEs and CWEs. We utilized clustering using SOM to enable pattern recognition, image analysis, and process monitoring. Visualization of complicated multidimensional data is one of the main application areas of the SOM. SOM usually links specific objects located in the various visualization and the position of an object either can be similar or identical on a map.

Comparing data with a map helps in classifying the data and indicates the trained data distribution. Mapping CVEs to CWEs helps us understand the impact of code vulnerabilities and to develop mitigation techniques. V2W-BERT provides the necessary automation of this mapping but may result in erroneous classification due to the inherent challenges of machine learning applied to methods designed around human-language interpretations. V2W-BERT

is a refinement of a pre-trained BERT using results of human mapping of CVEs to CWEs as training examples. This specialization of BERT to cyber security is recognized to include errors evidenced by, for example, multiple paths to the same conclusion in the hierarchy of CWE classes. Furthermore, the trustworthiness of models based on deep learning is always questionable since the connection between input and class assignment is difficult or impossible to discern.

For these reasons, we undertook a mapping of CVEs to CWEs based on self-organizing maps of vectors derived from the natural language processing of CVEs, which were directed to clusters of CVEs without using the predictions of the V2W-BERT classification model. By analyzing the membership CVE clusters concerning the CWE labels predicted by V2W-BERT, we hope to gain more trust in automated methods, like V2W-BERT, to map CVEs onto CWEs.

Chapter Eight

Matlab Code

8.1 alldataClustering138.m file

```
1 %reads a (SOM_PAK format) ASCII data file and loads in sD
2 sD = som_read_data('V2W-LINK-distilbert-base-uncased-dp_rep.txt');
3 %assigning maximum cluster size
4 maxClusters=138;%based on DBI index select best value of k
5
6 % som_make to create, initialize and train a SOM
7 sM=som_make(sD, 'shape', 'toroid', 'mapsize', 'big', 'training', 'long',
8             'tracking', 0);
9
10 %som_show for basic visualization
11 som_show(sM, 'umat', 'all', 'empty', 'Labels', 'norm', 'd');
12
13 %automatically labels the SOM based on given data
14 sM = som_autolabel(sM, sD, 'vote');
15
16 %calculates BMUs for given data vectors and stores in prediction
17 prediction = som_bmus(sM, sD);%assigning values from 6417 to 99950 matrix
18
```

```

19 figure;
20 som_show(sM, 'umat', 'all');
21 %calculates the response of data on the map
22 h = som_hits(sM,sD);
23
24 %for adding hits and labels on the map
25 som_show_add('hit',h, 'MarkerColor', 'w', 'Subplot',1);
26
27 figure;
28
29 [c, p, err, ind] = kmeans_clusters(sM); % find k clusters and returns ...
    centroids, errors, and DBI index
30 [dummy,i] = min(ind); % select the one with smallest index
31 som_show(sM, 'color', {p{i}, sprintf('%d clusters',i)}); % visualize clusters
32
33 %k means algorithm with specified value of k
34 [codes,basesM,errors]=som_kmeans('batch', sM,maxClusters);
35 basesM=basesM';
36
37 %show's OPT clusters with k clusters
38 som_show(sM, 'color', {basesM, sprintf('%d OPT clusters using ...
    K-means', max(basesM))});
39
40
41 mapping=basesM(prediction); % for assigning cluster number to 99950 records
42 ResolvedData(:,3)=mapping; %Assign Cluster Number
43 ResolvedData(:,4)=prediction; %Assign BMU number
44
45 Unique_CWE = unique(ResolvedData(:,3));% for unique value of CWE in ...
    ResolvedData
46 for i = 1:length(Unique_CWE)

```



```

47         totalCVEs(i,2) = sum(ResolvedData(:,3)==totalCVEs(i)); % number ...
            of times each unique value is repeated
48     end
49
50 %for generating key value pair for cwe and count of them in each ...
    cluster using containers.Map
51
52     for i = 1:(maxCluster)
53         %CWECount key value pair
54         CWECount{i}=containers.Map('KeyType','char','ValueType','double');
55
56     end
57     for i = 1:length(ResolvedData(:,3))
58         index=ResolvedData(i,3);
59         CWECode=int2str(ResolvedData(i,2));
60         if (~isKey(CWECount{index},CWECode))
61             CWECount{index}(CWECode)=0;
62         end
63         CWECount{index}(CWECode)=CWECount{index}(CWECode)+1;
64     end

```

8.2 graph.m file

```

1 %code to show bar graph for all 138 clusters
2 %LM is the container with key value pair of CWE and it's count in each ...
    cluster
3 i=1;
4 for j = 0:19
5     figure();

```

```

6  for ii = 1:10
7      %pre-condition check for CWECount if it is empty
8      if isempty(CWECount{i}.keys)
9          i=i+1;
10     end
11     %to show 10 bar graphs in each figure
12     subplot(2,5,ii);
13     plot(i);
14     keys=CWECount{i}.keys;
15     val=values(CWECount{i},keys);
16     x=categorical(regexprep(keys, '\[(.*?)\]', '$1'));
17     y=cell2mat(val);
18     bar(x,y);
19     %to show title of each bar graph
20     title(['cluster ' int2str(i)]);
21     i=i+1;
22 end
23 end

```

8.3 percentageData.m file

```

1  %for calculating percentages based on 2 predominant peaks in displayed ...
   bar graph
2  for i = 1:maxClusters
3      if (~isempty(CWECount{i}.keys))
4          keys=CWECount{i}.keys;
5          val=values(CWECount{i},keys);
6          y=cell2mat(val);
7          %for getting first two most appearing CWE's based on count value

```

```

8         [B,I]=maxk(y,2);
9         %for calculating percentage based on total of both CWE count
10        percentCalc= sum(B)/sum(y)*100;
11        percentData(i,1)=i;
12        percentData(i,2)=totalCVEs(i,2);
13        percentData(i,3)=str2num(keys{I(1)});
14        percentData(i,4)=B(1);
15        percentData(i,5)=0;
16        percentData(i,6)=0;
17        if(length(I)>1)
18            percentData(i,5)=str2num(keys{I(2)});
19            percentData(i,6)=B(2);
20        end
21        percentData(i,7)=percentCalc;
22    end
23 end

```

REFERENCES

- [1] Omar Alhazmi, Yashwant Malaiya, and Indrajit Ray. “Security vulnerabilities in software systems: A quantitative perspective”. In: *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer. 2005, pp. 281–294.
- [2] Ehsan Aghaei and Ehab Al-Shaer. “Threatzoom: neural network for automated vulnerability mitigation”. In: *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*. 2019, pp. 1–3.
- [3] Siddhartha Shankar Das et al. “V2W-BERT: A Framework for Effective Hierarchical Multiclass Classification of Software Vulnerabilities”. In: *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. 2021, pp. 1–12.
- [4] Duc Truong Pham, Stefan S Dimov, and Chi D Nguyen. “Selection of K in K-means clustering”. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 219.1 (2005), pp. 103–119.
- [5] Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*. Vol. 38. M. Dekker New York, 1988.
- [6] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.
- [7] Madhu Yedla, Srinivasa Rao Pathakota, and TM Srinivasa. “Enhancing K-means clustering algorithm with improved initial center”. In: *International Journal of computer science and information technologies* 1.2 (2010), pp. 121–125.
- [8] Kadim Tasdemir and Erzsébet Merényi. “Exploiting data topology in visualization and clustering of self-organizing maps”. In: *IEEE Transactions on Neural Networks* 20.4 (2009), pp. 549–562.
- [9] Juha Vesanto and Esa Alhoniemi. “Clustering of the self-organizing map”. In: *IEEE Transactions on neural networks* 11.3 (2000), pp. 586–600.
- [10] Yudhistira Arie Wijaya et al. “Davies Bouldin Index Algorithm for Optimizing Clustering Case Studies Mapping School Facilities”. In: (2021).
- [11] Kristina P Sinaga and Miin-Shen Yang. “Unsupervised K-means clustering algorithm”. In: *IEEE access* 8 (2020), pp. 80716–80727.

- [12] Yunus Dogan, Derya Birant, and Alp Kut. “SOM++: integration of self-organizing map and k-means++ algorithms”. In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2013, pp. 246–259.
- [13] Anne Honkaranta, Tiina Leppänen, and Andrei Costin. “Towards practical cybersecurity mapping of stride and cwe—a multi-perspective approach”. In: *2021 29th Conference of Open Innovations Association (FRUCT)*. IEEE. 2021, pp. 150–159.

ProQuest Number: 29067731

INFORMATION TO ALL USERS

The quality and completeness of this reproduction is dependent on the quality and completeness of the copy made available to ProQuest.



Distributed by ProQuest LLC (2022).

Copyright of the Dissertation is held by the Author unless otherwise noted.

This work may be used in accordance with the terms of the Creative Commons license or other rights statement, as indicated in the copyright statement or in the metadata associated with this work. Unless otherwise specified in the copyright statement or the metadata, all rights are reserved by the copyright holder.

This work is protected against unauthorized copying under Title 17,
United States Code and other applicable copyright laws.

Microform Edition where available © ProQuest LLC. No reproduction or digitization of the Microform Edition is authorized without permission of ProQuest LLC.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA