Categorization of Microsoft Security Bulletins

Maliheh Zargaran
University of Texas at El Paso
mzargaran@miners.utep.edu

Christopher Mendoza
University of Texas at El Paso
camendoza7@miners.utep.edu

Gerson Santos
University of Texas at El Paso
gsantosmed@miners.utep.edu

Sumi Dey
University of Texas at El Paso
sdey2@miners.utep.edu

Aldo Pillado
University of Texas at El Paso
arpilladoro@miners.utep.edu

Rohan Baingolkar
University of Texas at El Paso
rubaingolkar@miners.utep.edu

Ishtjot Kamboj
University of Texas at El Paso
iskamboj@miners.utep.edu

ABSTRACT

This paper uses the Microsoft Security bulletins that are available online to train a Naive Bayes classifier to categorize the class of vulnerability. A new set of vulnerability categories and sub-categories was made to facilitate the classifier. This paper also explores the most predominant words for each of the new categories.

CCS CONCEPTS

• **Security and privacy** → *Database and storage security*;

KEYWORDS

Cyber Security, Data Minig, Text Classification, Software Vulnaribility

ACM Reference Format:
Maliheh Zargaran, Christopher Mendoza, Gerson Santos, Sumi Dey, Aldo Pillado, Rohan Baingolkar, and Ishtjot Kamboj. 2018. Categorization of Microsoft Security Bulletins. In *Proceedings of ACM Report (Data Mining Final Report).* ACM, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.475/123_4

	Report	Python Coding	Website	Data Processing
Maliheh Zargaran	X	X		X
Christopher Mendoza	X	X		X
Sumi Dey	X			X
Aldo Pillado			X	X
Rohan Baingolkar			X	X
Ishtjot Kamboj			X	X
Gerson Santos	X	X		X

Figure 1: Work Distribution

Data Mining Final Report, May 2018, El Paso, Tx, USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

1 INTRODUCTION

Software vulnerabilities enable an attacker to exploit the security flaw and pose a significant risk to the host computing system and its user. When software vendors detect vulnerabilities they will develop patches and mitigations to address this issue.

To resolve this issue work has been done to forecast future software security risks [10], [1], [3], [11]. To assess the severity of vulnerabilities, various types of scoring frameworks have been developed [6], [8], [7].

No previous work was found that has categorized the vulnerabilities to find the prominent words for the corresponding category. There are approximately 1530 security bulletins released by Microsoft from 1998 to 2017. This project utilizes these bulletins to make the dataset manually, Table Z. Then we kept the information we collected from the bulletins in tabular format. Then we made the category for each document based on the information. The labeling follows the format of category_subcategory.

The rest of this report is organized as: second section explains some related works presented in different papers. We have described about text classification, Naive Bayes and the application of Naive Bayes in the third section. The fourth section is about our approach that is how we made the dataset and which algorithm we implemented. After that we discussed our experiments and results. Conclusion and future work is discussed at the end of this report.

CVSS v2.0		**CVSS v3.0**	
Severity	Score Range	Severity	Score Range
None		None	0.0
Low	0.0 - 3.9	Low	0.1 - 3.9
Medium	4.0 - 6.9	Medium	4.0 - 6.9
High	7.0 - 10.0	High	7.0 - 8.9
Critical		Critical	9.0 - 10

Table 1: CVSS Comparison: Represent the severity level of bulletins

2 RELATED WORK

Related papers use CVE as the main feature in their experiments. The big data sets like NVD (National Vulnerability Data Base), MIRTE system (CVE.mitre.org) and CVSS (Common Vulnerability Scoring System SIG), even Microsoft bulletins are linked to those data sets. There are other common data bases such as EDB (Exploit Data Base), EDB records exploits and corresponding vulnerabilities. Our method does not use these data bases to classify or categorize data. It is found that in some years, especially in the past (1998-2005), the information relate to vulnerabilities are stored in CAN instead of CVE. CAN and CVE record security related bulletins but CVE records the most common ones. There are also cases where no CAN or CVE are associated with certain bulletins.

From the National Vulnerability Database (NVD) [4] looks for leverage trends in historical vulnerability data to forecast vulnerability discovery rates for individual software packages. With several time series distance measurements k-NN classification is used to select the appropriate regression models to forecast. To provide a margin of error 68 percent and 95 percent confidence bounds are generated around the actual forecast.

A deep learning based text classification method is discussed in [2] to predict severity level of software vulnerability based on only vulnerability description. For the prediction, this method uses word embeddings and a one-layer shallow Convolutional Neural Network (CNN) to automatically capture discriminative word and sentence features of vulnerability descriptions.

Several versions of defect predictor is discussed in [10] based on Naive Bayes theory and their difference estimation method and algorithm complexity was analyzed. It is found that in all kinds of Naive Bayes Multi-variants Gauss Naive Bayes (MvGNB) performs best based on the experimental results on the benchmarking data sets. This method is compared with decision tree learner J48 to prove the effectiveness of MvGNB.

NVD currently has 19 CWEs in its vulnerability classification scheme [9]. Each vulnerability also includes a set of scores of CVSS Base Metric Group that shows the intrinsic and fundamental characteristics of a vulnerability that are constant over the time and user environments. NVD uses CWE as a classification mechanism that differentiates CVEs by the type of vulnerability. Every CVE is related to a specific product but CWE is related to type of vulnerability.

3 BACKGROUND

3.1 Text classification

Text classification assigns documents to one or more classes according to their content. Classes are selected from previously established set. There are various kinds of text classification, for example, binary classification like spam filtering or simple sentiment analysis (POSITIVE, NEGATIVE), multiple class classification like selecting one category among several alternatives - movie genre classification (thriller, terror, romantic, etc), multi-label categorization like assigning all categories that apply to a single document.

3.1.1 Text classification: definition. Given document d from a document space $\mathcal{X}$; and a fixed set of classes $\mathcal{C} = c_1, c_2, ..., c_n$. Categories or labels are another name for classes. Usually, $\mathcal{X}$ is a high-dimensional space and classes are user defined according to the need of the application.

This is a supervised learning algorithm. We denote the supervised learning method by f and write $f(\mathcal{D}) = l$. The training set $\mathcal{D}$ is used as input in this learning method f and returns the learned classification function l. Once we have learned l, we can apply it to the test set (or test data), whose class is unknown.

3.2 Naive Bayes

The probabilistic model of Naive Bayes classifiers is based on Bayes theorem, and the assumption here is that the features in a dataset are mutually independent. Naive Bayes classifier tends to perform very well under the assumption of independence, even the assumption is violated. Naive Bayes classifier can outperform the more powerful alternatives especially for small sample sizes. Naive Bayes classifiers are used in many different fields due to its robustness and accuracy. For example, this is used in disease diagnosis, spam filtering etc.

3.2.1 Posterior probabilities. In the context of a classification problem, the *posterior probability* is the probability that a particular object belongs to class i given its observed feature values. The general notation of the posterior probability can be written as

$$p(c_j|\mho_i) = \frac{p(\mho_i|c_j).p(c_j)}{p(\mho_i)},$$

where c_j denotes the class j, $j \in 1, 2, ..., m$, $\mho_i$ denotes the feature vector of sample i, $i \in 1, 2, ..., n$ and $p(\mho_i|c_j)$ denotes the probability of observing sample $\mho_i$ given that is belongs to class c_j.

3.2.2 Class-conditional probabilities. In Naive Bayes classifier, with the assumption of independence, an additional assumption is the conditional independence of features. The class-conditional probabilities can be directly estimated from the training data instead of evaluating all possibilities of feature vector. For a $d-$dimensional feature vector $\mho$, the class conditional probability can be calculated as follows:

$$p(\mho|c_j) = p(\mho_1|c_j).p(\mho_2|c_j)....p(\mho_d|c_j) = \prod_{1 \le k \le d} p(\mho_k|c_j).$$

where $p(\mho|c_j)$ can be interpreted as "How likely is it to observe this particular pattern x given that it belongs to class c_j?"

3.2.3 Prior probabilities. The *prior probabilities* are also called class priors. This can be interpreted as the general probability of encountering a particular class. This can be

calculated as:

$$p(c_j) = \frac{N_{c_j}}{N_c},$$

where N_{c_j} counts the samples from class c_j and N_c counts all samples.

3.2.4 Evidence. The *evidence* can be interpreted as the probability of encountering a particular pattern that is independent from the class label. This can be calculated as follows:

$$p(\mathbf{f}_i) = p(\mathbf{f}_i|c_j).p(c_j) + p(\mathbf{f}_i|c_j^C).p(c_j^C)$$

3.3 Multi-variate Bernoulli Naive Bayes

In case of Multi-variate Bernoulli model, every token consists of two values - 0 or 1 in the feature vector of a document. The dimension of the feature vector is the number of words in the vocabulary. If the value of the feature vector is 1, that means that word appears in the document and the value 0 means, that word does not appear in the document. The Bernoulli trials can be expressed as

$$p(\mho|c_j) = \prod_{1 \le i \le m} p(\mho_i|c_j)^n.(1 - p(\mho_i|c_j))^{(1-n)}, \quad n \in (0, 1)$$

Parameter estimation The maximum-likelihood estimate that a particular word (or token) $\mho_I$ occurs in class c_j can be expressed as

$$p(\mho_i|c_j) = \frac{d_{\mho_i, c_j}}{d_{c_j}},$$

where $d_{\mho_i, c_j}$ is the number of documents in the training dataset consisting of feature $\mho_i$ that belong to class c_j and d_{c_j} is the number of documents in the training set belonging in class c_j.
The only issue with this estimation is it will give zero to the termclass probability if a feature does not appear in the document. To overcome this issue, Laplace smoothing is used. And the above equation is calculated by the following

$$p(\mho_i|c_j) = \frac{d_{\mho_i, c_j} + 1}{d_{c_j} + 2}.$$

3.4 Applications of Naive Bayes Algorithms

Some of Naive Bayes applications are:

- **Real Time Prediction:**
  Naive Bayes is an eager learning classifier and is efficient, making this technique ideal for real time predictions.
- **Multi class prediction:**
  This algorithm is also well known for its multi-class prediction feature. It can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:**
  Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:**
  Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not.

4 OUR APPROACH

We have come up with a new set of labels and are trying, Appendix C, to find the most predominant words or high probable words leaded to those labels, it means characterizing every type of vulnerability and come up with new labels or categories, those make digging into this new dataset interesting.

4.1 Predominant Words

Calculating the probability of all terms being in all categories, appendix A, is calculated by the following formula:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T'_{ct'} + 1)} = \frac{T_{ct} + 1}{\sum_{t' \in V}(T'_{ct'} + B')}, \quad (1)$$

By sorting the highest probability it is obtain as outcome the most predominant words to be in a category. For instance, the category Microsoft Product_Office generates word, excel, and office because the bulletins description contains these set of words to correlate them with these category.

4.2 Pre-processing

Initially, we wanted to extract features from the bulletins using .xml files. But unfortunately, all the bulletins are not in the same format. Then we have made features manually from all the bulletins available. Features include Year, Bulletin ID, Heading, Description, Severity. The information in description part is based on Executive Summary, Vulnerability Information and CVE. Severity is the severity level for the corresponding bulletin.

Year	Bulletin ID	Heading	Description	Severity	Category
2017	MS17-023	Security Update for Adobe Flash Player	Vulnerabilities in Adobe Flash Player when installed on all supported editions of Windows.	Critical	Third Party Software_Adobe
2017	MS17-022	Security Update for Microsoft XML Core Services	This security update is rated Important for Microsoft XML Core Services 3.0 on all supported releases of Microsoft Windows. The update addresses the vulnerability by changing how MSXML handles objects in memory.	Important	Services_Microsoft Services
2017	MS17-021	Security Update for Windows DirectShow	The vulnerability could allow an information disclosure if Windows DirectShow opens specially crafted media	Important	API_DirectShow
2017	MS17-020	Security Update for Windows DVD Maker	Information disclosure vulnerability in Windows DVD Maker.	Important	Microsoft Product_Media Player

Figure 2: Microsoft Dataset Sample

In the second stage of pre-processing, we added one more feature named "Category". Category represents the category as well as subcategory of the bulletin based on its description. There are eight main categories that are proposed, Networking, Memory, Operating System(OS), Microsoft Product, User Interface(UI), Services, API and Third-Party Software. The idea behind creating a unique label or labels (some bulletins include more than one label) is to find out the topic related to the vulnerability. Then subcategory is used to describe the vulnerability in greater detail as the main categories are too broad on their own. The format is consisted of Category-Subcategory. For example, Browser and Server can be two subcategories for Networking, Kernel and Driver as subcategories for Operating System(OS). Using this paradigm, we have classified all of the existing security bulletins. Then split the dataset into training and testing set.

4.3 StopWords

Common words that appear to often in a language and have low impact on word preprocessing, are called StopWords [4]. Removing stop words from the dataset improves speed and performance for information retrieval. The stop words list used during experimentation consisted of a combination between the most common English words and our dataset, like :

- an
- applet
- exploit
- vulnerability
- the
- ...

4.4 Methodology

After pre-processing the data, there are 56 categories. These categories is the target function. Naive Bayes performs really good not only for binary class, but also for hundreds of class. For this reason, we used Multinomial Naive Bayes algorithm to categorize the bulletin based on the description.

4.4.1 Naive Bayes text classification. The multinomial Naive Bayes or multinomial NB model is a probabilistic learning method. The probability of a document d being in class c is computed as

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{2}$$

where $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c. $P(t_k|c)$ is interpreted as a measure of how much evidence t_k contributes that c is the correct class. $P(c)$ is the prior probability of a document occurring in class c. If a document's terms are not significant enough for one class versus another, we choose the one that has a higher prior probability. $< t_1, t_2, ..., t_{n_d} >$ are the tokens in d that are part of the vocabulary we use for classification and n_d is the number of such tokens in d.

Our goal is to find the best class for the document in text classification. The best class in NB classification is the most likely or maximum a posteriori (MAP) class c_{map} :

$$c_{map} = \arg\max_{c \in \mathcal{C}} \hat{P}(c|d) = \arg\max_{c \in \mathcal{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c), \tag{3}$$

where $\hat{P}$ is the estimated value of the parameter P.

Multiplication of many conditional probabilities in (3) results in a floating point underflow. To avoid this, in most implementation of NB the maximization is calculated by the following:

$$c_{map} = \arg\max_{c \in \mathcal{C}} \ [log\ \hat{P}(c) + \sum_{1 \leq k \leq n_d} log\ \hat{P}(t_k|c)], \tag{4}$$

where $log\ \hat{P}(c)$ is a weight that denotes the relative frequency of c and $log\ \hat{P}(t_k|c)$ is a weight that denotes how good an indicator t_k is for c and their sum is a measure of how much evidence there is for the document being in the class and (4) selects the class for the most evidence.

Parameter estimation: Maximum likelihood estimation is the relative frequency and corresponds to the most likely value of each parameter given the training data. Hence, the estimated priors is calculated by

$$\hat{P}(c) = \frac{N_c}{N}, \tag{5}$$

where N_c is the number of documents in class c and N is the total number of documents. The estimated conditional probability is the relative frequency of term t in documents belonging to class c, that is,

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T'_{ct'}}, \tag{6}$$

where T_{ct} is the number of occurrences of t in training documents from class c, including multiple occurrences of a term in a document.

The problem with this estimate is that it is zero for a term–class combination that did not occur in the training data. To eliminate zeros, we use add-one or Laplace smoothing, which simply adds one to each count, that is,

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T'_{ct'} + 1)} = \frac{T_{ct} + 1}{\sum_{t' \in V}(T'_{ct'} + B')}, \tag{7}$$

where $B = |V|$ is the number of terms in the vocabulary.

TRAINMULTINOMIALNB($\mathcal{C}, \mathcal{D}$)
1 $V \leftarrow$ EXTRACTVOCABULARY($\mathcal{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathcal{D}$)
3 **for each** $c \in C$
4 **do** $N + c \leftarrow$COUNTDOCSINCLASS($\mathcal{D}, c$)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ CONCATENATETEXFALLDOCSINCLASS($\mathcal{D}, c$)
7 **for each** $t \in V$
8 **do** $T_ct \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

APPLYMULTINOMIALNB($\mathcal{C}, V, prior, condprob, d$)
1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
2 **for each** $c \in \mathcal{C}$
3 **do** $score[c] \leftarrow \log\ prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] + = \log\ condprob[t][c]$
6 **return** $\arg\max_{c \in \mathcal{C}}\ score[c]$

Pseudo Code 1. Naive Bayes algorithm (multinomial model): Training and testing.

5 RESULTS

In figure 3, you can see the trend of NB classifier evaluation using Accuracy, Precision, Recall and F1-Score. In most of test that have done the trend in all is increasing but in latest one Precision and F1-Score has decreased a little bit. It shows that by adding the last two years of bulletins (2016 and 2017), the number of False Positive increased and as F1-Score is called weighted harmonic mean of Precision and Recall, it changes as well. As figure confirms the close relation between those three evaluation measures. Removing Misc (Miscellaneous), as subcategory in some categories will solve this decreasing in trend of those two years and hopefully increases Accuracy more.

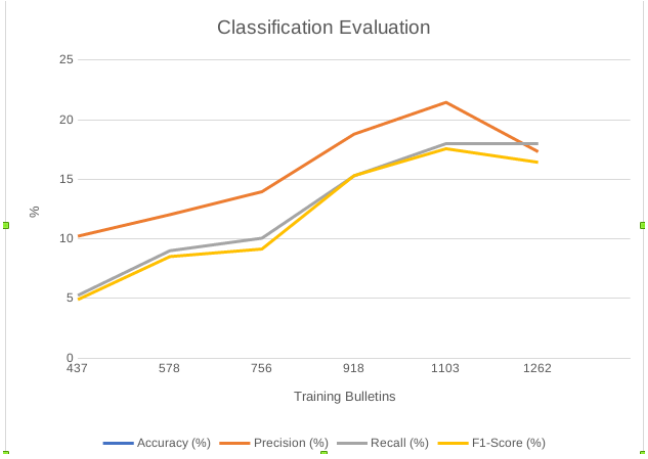

Figure 3: Ratio of results and number of bulletins

Figure 4 represents the correlation between the predicted outcome against a calculated value

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

Figure 4: Confusion Matrix Concept

The following formulas use the previous table result to indicate the ratio between each other.

$$Accuracy = \frac{t_p + t_n}{t_p + t_n + f_n + f_p} \tag{8}$$

$$Precision = \frac{t_p}{t_p + f_p} \tag{9}$$

$$Recall = \frac{t_p}{t_p + f_n} \tag{10}$$

$$F1 - Score = \frac{Precision * Recall}{Precision + Recall} \tag{11}$$

Figure 5 represents the confusion matrix of first three experiments out of six. As picture shows the most bulletins are categorized in Networking_Server, the brightest square and also there are some squares that are appearing. In the second training set, label 10 is presenting Microsoft Product_Office which has more bulletins than Networking_Browser but less than Networking_Server. Third experiment shows that with adding more data into training set some squares disappeared that means the classifier has classified them into other Categories_Subcategories. The table inside Figure 5 shows that we have the fix number of bulletins and also the same number of unique labels in all of the experiments but the number of unique labels is increasing in each experiment.

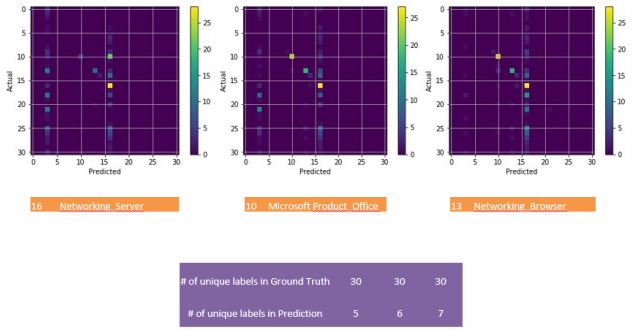

Figure 5: Confusion Matrix: 1st three experiments

Categories	Subcategories
Networking	Browser
	Protocol
	Server Services
	Services
	Socket
	Misc
Memory	Buffer
	Handling
Operating System	Kernel
	File System
	Virtualization
	Shell
	Core Functions
	Security
	Driver
	Misc
Microsoft Product	Office
	Multimedia
	Developing Product
	CrystalReport
	Visio
	Kodak
	Windows
	Color management
	MSN Messenger
	RichEdit
	Agent
	Address Book
	Lync
	WordPerfect
	Project
	Misc
User Interface	GUI
	Terminal
	Misc
Services	Microsoft Services
	Server Services
	Remote Services
	Security
	XML
API	.NET
	ActiveX
	OCX
	DirectX
	DirectShow
	DirectPlay
	DirectWrite Framework
	Developing Product
	Direct2D
	Misc
Third-Party Software	Adobe
	Third Party Developing Product

Table 2. Manual Categorizing

Figure 6 shows the other three experiments that have done after the three test that already has discussed. The result of these experiments supports the previous tests Empirically. In the fourth experiment the category OS_Kernel has appeared. It proofs that in the year 2011 and 2012 the vulnerabilities related to this category started to be showing up. The vulnerabilities in category OS_Kernel has increased in the following experiments as the color of square which represent this category has changed to brighter. Finally, in last test, categories Networking_Server, Microsoft Product_Office and Networking_Browser have the most vulnerabilities. The table[2], is in the following of table[1] that proofs with same test set, if we increase training data, the model will be able to predict more unique labels.

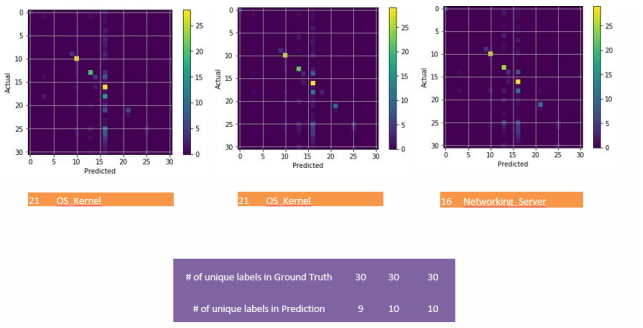

Figure 6: Confusion Matrix: 2nd three experiments

Training Years	Total Bulletins	Total Words	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
1998 - 2005	437	16295	21.84	10.23	5.26	4.91
1998 - 2007	578	17679	35.44	12.04	9.01	8.52
1998 - 2007, 2009-2010	756	21303	37.86	13.96	10.06	9.15
1998 - 2007, 2009-2012	918	23323	45.63	18.79	15.29	15.31
1998 - 2007, 2009-2014	1103	25164	50.97	21.46	17.98	17.57
1998 - 2007, 2009-2014, 2016-2017	1262	27614	51.46	17.32	18.02	16.43

Test Years	Total Bulletins	Total Words
2008, 2015	206	2070

Figure 7: Classifier Evaluation

Predicted	TRUE	Result
Microsoft Product_Office	Microsoft Product_Multimedia	FALSE
Microsoft Product_Multimedia	Microsoft Product_Multimedia	TRUE
Microsoft Product_Multimedia	Microsoft Product_Multimedia	TRUE
Microsoft Product_Multimedia	Microsoft Product_Multimedia	TRUE
Microsoft Product_Multimedia	Microsoft Product_Multimedia	TRUE
Networking_Server	Microsoft Product_Office	FALSE
Networking_Server	Microsoft Product_Office	FALSE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE
Microsoft Product_Office	Microsoft Product_Office	TRUE

Table 3. Sample of Naive Bayes Classifier

6 CONCLUSION

This project describes a new type of vulnerability analysis, *i.e.*, categorizing the vulnerability and finding prominent words for a particular category based on the description of the bulletin. And this analysis will help the non security experts to manage and prioritize the vulnerability in a simple way as it just needs the description of the bulletin. We have presented multinomial Naive Bayes text classification to support this vulnerability analysis. The reason of choosing this method is this is easy to implement, relatively robust, fast and accurate. We have splitted the dataset into train and test set. And we have found around 51 percent accuracy in the test set using this approach. We have also found the most predominant words in each category.

7 FUTURE WORK

We plan to extend our work to predict the category for different descriptions, for example, CVE description, attacker information etc. Also work with a new target function by combining category and severity and use the approach to predict the target function from various description. Also, it is necessary to keep track of this problem for the general public. By developing a website, users can consult and obtain information about common vulnerabilities on their computers. Finally, the code used on this work is under a Github repository [5].

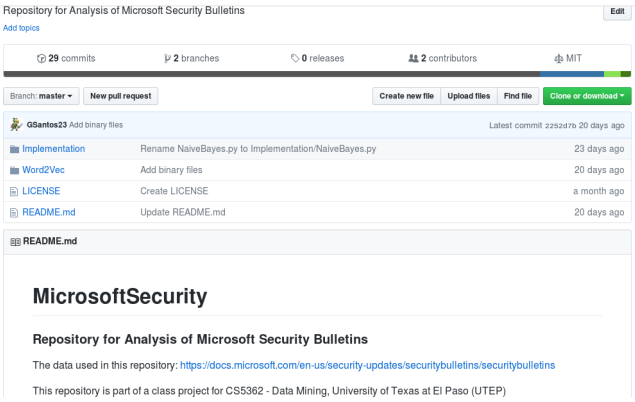

Figure 8: Github Page

REFERENCES

[1] O. H. Alhazmi and Y. K. Malaiya. 2005. Modeling the vulnerability discovery process. In *16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*. 10 pp.–138. https://doi.org/10.1109/ISSRE.2005.30

[2] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng. 2017. Learning to Predict Severity of Software Vulnerability Using Only Vulnerability Description. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 125–136. https://doi.org/10.1109/ICSME.2017.52

[3] Jinyoo Kim, Yashwant K. Malaiya, and Indrakshi Ray. 2007. Vulnerability Discovery in Multi-Version Software Systems. In *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE '07)*. IEEE Computer Society, Washington, DC, USA, 141–148. https://doi.org/10.1109/HASE.2007.78

[4] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[5] Gerson Santos Medina. 2018. MicrosoftSecurity. Retrieved May 11, 2017 from https://github.com/GSantos23/MicrosoftSecurity

[6] Microsoft. 2002. Microsoft security response center security bulletin severity rating system,. Retrieved March 30, 2017 from https://technet.microsoft.com/zhcn/security/gg309177.aspx

[7] I. SANS. [n. d.]. Sans critical vulnerability analysis archive,. Retrieved March 30, 2017 from http://www.sans.org/newsletters/cva/

[8] US-CERT. 2006. Uscert vulnerability note field descriptions,. Retrieved March 30, 2017 from http://www.kb.cert.org/vuls/html/fieldhelp

[9] Ju An Wang and Minzhe Guo. 2010. Vulnerability Categorization Using Bayesian Networks. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW '10)*. ACM, New York, NY, USA, Article 29, 4 pages. https://doi.org/10.1145/1852666.1852699

[10] Tao Wang and Wei hua Li. 2010. Naive Bayes Software Defect Prediction Model. *2010 International Conference on Computational Intelligence and Software Engineering* (2010), 1–4.

[11] Su Zhang, Doina Caragea, and Xinming Ou. 2011. An Empirical Study on Using the National Vulnerability Database to Predict Software Vulnerabilities. In *Proceedings of the 22Nd International Conference on Database and Expert Systems Applications - Volume Part I (DEXA'11)*. Springer-Verlag, Berlin, Heidelberg, 217–231. http://dl.acm.org/citation.cfm?id=2035368.2035388

A MOST PREDOMINANT WORDS

Category	Word1	Word2	Word3
API_.NET	net	framework	server
API_ActiveX	control	activex	explorer
API_Developing Product	net	server	sp2
API_Direct2D	direct2d	handle	figure
API_DirectPlay	directplay	content	office
API_DirectShow	directshow	media	rights
API_DirectWrite	directwrite	unicode	services
API_DirectX	directx	sami	types
API_Framework	ole	input	whenmicrosoft
API_Misc	mac	validation	uniscribe
API_OCX	ocx	services	rtf
Memory_Buffer	buffer	server	execute
Memory_Handling	memory	kernel-mode	drivers
Microsoft Product_Address Book	location	book	address
Microsoft Product_Agent	agent	url	memory
Microsoft Product_Color Management	management	color	module
Microsoft Product_Crystal Report	crystal	reports	business
Microsoft Product_Developing Product	visual	rights	server
Microsoft Product_Kodak	viewer	kodak	user
Microsoft Product_Lync	server	skype	lync
Microsoft Product_MSN Messenger	messenger	msn	request
Microsoft Product_Misc	journal	control	object
Microsoft Product_Multimedia	media	player	center
Microsoft Product_Office	office	excel	word
Microsoft Product_Outlook	outlook	mail	control
Microsoft Product_Project	project	resource	opening
Microsoft Product_RichEdit	rtf	richedit	ole
Microsoft Product_Visio	visio	files	external
Microsoft Product_Windows	win32	vulnerability	viewing
Microsoft Product_WordPerfect	wordperfect	vulnerability	software
Networking_Browser	explorer	memory	edge
Networking_Misc	site	message	services
Networking_Protocol	server	services	iis
Networking_Security	routing	nt	source
Networking_Server	server	services	exchange
Networking_Services	services	server	iis
Networking_Socket	address	winsock	driver
OS_Core Functions	memory	graphics	component
OS_Driver	driver	kernel-mode	font
OS_File System	installation	resolves	manager
OS_Kernel	kernel	kernel-mode	driver
OS_Misc	vulnerability	boot	registry
OS_Security	nt	password	service
OS_Shell	shell	servers	vulnerability
OS_Virtualization	virtual	vm	java
Services_Microsoft Services	services	server	service
Services_Misc	services	domain	local
Services_Remote Services	server	service	rpc
Services_Security	services	server	policy
Services_Server Services	services	server	memory
Services_XML	xml	services	core
Third Party Product_Misc	vm	java	earlier
Third Party Software_Adobe	adobe	websites	player
Third Party Software_Third Party Developing Product	xml	services	core
UI_GUI	component	graphics	document
UI_Misc	computer	users	screen

B CATEGORY FREQUENCY BY YEAR

Category_Subcategory	Frequency
Networking_Browser	5
Networking_Services	4
Networking_Server	2
Services_Remote Services	2
Microsoft Product_Office	2

Table 2: 1998

Category_Subcategory	Frequency
Networking_Server	11
Networking_Protocol	10
OS_Security	8
Microsoft Product_Office	5
Networking_Security	5

Table 3: 1999

Category_Subcategory	frequency
Networking_Services	15
Networking_Server	12
Memory_Buffer	11
Networking_Protocol	9
OS_Misc	8

Table 4: 2000

Category_Subcategory	frequency
Networking_Server	16
Networking_Protocol	12
Networking_Browser	5
Microsoft Product_Multimedia	4
OS_Shell	4

Table 5: 2001

Category_Subcategory	frequency
Memory_Buffer	24
Memory_Buffer	10
Networking_Server	9
OS_Core Functions	4
Networking_Browser	4

Table 6: 2002

Category_Subcategory	frequency
Memory_Buffer	11
Memory_Buffer	10
Networking_Server	5
Networking_Server	4
Networking_Browser	3

Table 7: 2003

Category_Subcategory	frequency
Networking_Server	7
Services_Microsoft Services	6
Services_Remote Services	6
Networking_Browser	5
Services_Security	4

Table 8: 2004

Category_Subcategory	frequency
Networking_Browser	8
Services_Microsoft Services	7
Networking_Services	5
Networking_Server	4
OS_Shell	3

Table 9: 2005

Category_Subcategory	frequency
Microsoft Product_Office	12
OS_Core Functions	9
Networking_Server	8
Networking_Browser	8
Memory_Buffer	7

Table 10: 2006

Category_Subcategory	frequency
Microsoft Product_Office	11
Networking_Browser	6
OS_Core Functions	6
Services_Microsoft Services	5
Networking_Server	5

Table 11: 2007

Category_Subcategory	frequency
Microsoft Product_Office	18
Networking_Server	11
Services_Remote Services	10
Networking_Protocol	8
Networking_Browser	8

Table 12: 2008

Category_Subcategory	frequency
Networking_Protocol	12
Microsoft Product_Office	11
Services_Server Services	8
Services_Remote Services	7
Networking_Browser	7

Table 13: 2009

Category_Subcategory	frequency
Networking_Server	21
Microsoft Product_Office	15
Services_Microsoft Services	10
Microsoft Product_Multimedia	8
OS_Kernel	6

Table 14: 2010

Category_Subcategory	frequency
OS_Driver	12
Microsoft Product_Office	11
OS_Core Functions	10
Networking_Services	7
API_.NET	6

Table 15: 2011

Category_Subcategory	frequency
Networking_Server	11
Memory_Handling	11
Microsoft Product_Misc	8
Microsoft Product_Office	8
Services_Microsoft Services	7

Table 16: 2012

Category_Subcategory	frequency
Services_Microsoft Services	15
Networking_Browser	13
Microsoft Product_Office	12
Networking_Server	10
Services_Remote Services	8

Table 17: 2013

Category_Subcategory	frequency
Networking_Browser	12
Microsoft Product_Office	10
Networking_Server	7
API_.NET	7
OS_Core Functions	6

Table 18: 2014

Category_Subcategory	frequency
Networking_Server	20
Networking_Browser	17
Services_Microsoft Services	15
OS_Core Functions	13
Microsoft Product_Office	11

Table 19: 2015

Category_Subcategory	frequency
Networking_Browser	29
OS_Kernel	16
Networking_Server	15
OS_Misc	12
Microsoft Product_Office	12

Table 20: 2016

Category_Subcategory	frequency
Third Party Software_Adobe	3
Networking_Server	3
Networking_Browser	2
Services_Microsoft Services	2
OS_Kernel	2

Table 21: 2017

C BULLETINS FREQUENCY

Category_Subcategory	of Bulletins	Category_Subcategory	of Bulletins
Networking_Server	197	Networking_Security	7
Networking_Browser	163	Networking_Misc	6
Microsoft Product_Office	160	Services_Misc	6
Services_Microsoft Services	102	Services_XML	5
Networking_Protocol	96	Microsoft Product_Outlook	5
Memory_Buffer	82	Third Party Software_Third Party Developing Product	5
OS_Core Functions	73	Third Party Product_Misc	4
OS_Kernel	67	Microsoft Product_Lync	4
Services_Server Services	54	Microsoft Product_MSN Messenger	4
Microsoft Product_Multimedia	51	Microsoft Product_Agent	3
API_.NET	43	API_Framework	3
Networking_Services	41	API_Misc	3
Services_Remote Services	41	Microsoft Product_Windows	3
Microsoft Product_Misc	36	Microsoft Product_Color Management	2
Memory_Handling	31	Microsoft Product_Crystal Report	2
Services_Security	28	API_DirectX	2
OS_Misc	28	Networking_Socket	2
OS_Driver	27	UI_Misc	2
API_ActiveX	26	Microsoft Product_Project	2
OS_File System	25	API_Developing Product	2
OS_Shell	19	Microsoft Product_Kodak	1
Third Party Software_Adobe	18	Microsoft Product_Address Book	1
Microsoft Product_Developing Product	16	API_OCX	1
UI_GUI	13	Microsoft Product_WordPerfect	1
OS_Virtualization	11	API_DirectWrite	1
OS_Security	9	Microsoft Product_RichEdit	1
Microsoft Product_Visio	8	API_DirectPlay	1
API_DirectShow	8	API_Direct2D	1