The Eleventh International Conference on Advanced Computational Intelligence,
June 7-9, 2019, Guilin, China

Intelligent Prediction of Vulnerability Severity level Based on Text Mining and XGBoost

Peichao Wang
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha, China
PeichaoW@163.com

Yun Zhou
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha, China
zhouyun@nudt.edu.cn

Baodan Sun
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha, China
13022438280@163.com

Weiming Zhang
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha, China
wmzhang@nudt.edu.cn

***Abstract*—Vulnerabilities have always been important factors threatening the security of information systems. The endless vulnerabilities pose a huge threat to the social economy and public privacy. The vulnerability database provides abundant materials for researchers to study the threat of vulnerabilities, while mining the text information of the database and obtaining valuable information can help to grasp the severity level of the vulnerability. Based on the textual description of vulnerabilities in the database, we first use text mining to extract main features. Then we utilize principal component analysis to gather sparse features which take sparse characteristic into consideration. Finally we use XGBoost to intelligently predict the severity level of vulnerabilities and compare them with the results of other machine learning methods based on same extracted features. The experiment on real-world vulnerability text description show the effectiveness of our method.**

Keywords*—*severity level of vulnerabilities, text mining, PCA, XGBoost, threat intelligence

I. INTRODUCTION

The information system plays a vital role in enterprises and organizations. With the implementation of office automation, the security and stability of information systems are of great importance in the business continuity. The vulnerability is a flaw in hardware, software, protocol implementation or system security policy, which allows an attacker to access or destroy an information system without authorization. With the huge size of source codes and the complexity of the logic, the exposure frequency of the vulnerability is getting higher and higher. The impacts of different vulnerabilities on information systems are different, which means that developers will ignore some of them after vulnerabilities being exposed, whereas some will be paid much attention to and patched in time.

With the improvement of national laws and regulations, in order to protect the interests of developers and stakeholders, the way to exploit vulnerabilities after exposure is no longer exposed, but it is difficult for developers to know the actual severity of the vulnerability. In open-source databases such as NVD (National Vulnerability Database), CVSS[1] (Common Vulnerability Scoring System) is often used to assess the severity level of vulnerabilities.

CVSS score is a commonly used approach to assess vulnerability severity. The scoring process usually requires the participation of experts. The current version includes 2.0 and 3.0, which will grade a vulnerability from three aspects: base, temporal and environment. One overall severity (high, medium, and low) of a vulnerability is obtained based on these three aspects. However, when a vulnerability has just been exposed, the severity level has not been evaluated yet, and the exploitation of the vulnerability often takes only a short time; developers usually need to arrange the patch of the vulnerability in a reasonable way while finding out the vulnerability severity level. Therefore, how to intelligently predict the severity level of a vulnerability based on its short description is a valuable research issue.

At present, researches on vulnerabilities are mostly concentrated on using the combination of source code and expert knowledge to extract features, and apply machine learning classifiers to intelligently detect the existence of vulnerability, or to automatically classify vulnerability types according to the extracted features to promote vulnerability management. However, there are few researches on the severity level of vulnerabilities. Hence, according to the description of vulnerabilities, we first use text mining to extract main features. Then we utilize principal component analysis to gather sparse features which take sparse characteristic into consideration. Finally we use XGBoost to intelligently predict the severity level of vulnerabilities and compare them with the results of other machine learning methods based on same extracted features. The experiment on real-world vulnerability text description show the effectiveness of our method.

II. RELATED WORKS

Traditional methods of security vulnerability analysis mainly include three types: static analysis, dynamic analysis and hybrid analysis [1]. Static analysis is a commonly used

[1] https://www.first.org/cvss/

manual analysis method, by which security personnel directly mine possible vulnerabilities from source codes. Dynamic analysis analyzes potential vulnerabilities in a program when it is running, which simulates real attackers to test and relies on the integrity of attack vector. Hybrid analysis is a combination of above two methods.

The above methods solve specific location of vulnerabilities in an information system. However, vulnerability analysis relying on security personnel within the organization alone is incapable in the current situation of endless vulnerabilities. The analysis and patch of vulnerabilities should be determined according to the severity level and the resources governed by the manager [2].

Open source vulnerability database (for instance, NVD, CVE, and CNNVD) provides good threat intelligence for security personnel, and real-time updated vulnerability database allows them to keep knowing newly discovered vulnerabilities. However, newly recorded vulnerabilities usually do not have an assessment of the corresponding severity level. Patching a vulnerability often requires huge manual labor and will have a great impact on the business continuity. The reasonable patch order of the vulnerabilities should be arranged according to severity level. The prediction of severity level is an effective way to provide a priority.

Machine learning [3] provide important approaches in the field of vulnerability research. Text mining [4] was used by Hovsepyan et al. [5] to analyze source codes of programs written in JAVA to identify the vulnerabilities contained in the source code. Yamamoto et al. [6] applied LDA, SLDA and other models to the crawled NVD data and extracted the text topic directly to describe the characteristics of the vulnerability. Toloudis et al. [7] used the method of text mining and Spearman correlation coefficient to find the correlation between the information description and the severity level of the vulnerability. Chen et al. [8] used the method of text mining to predict the severity level of phishing sites based on the warning of phishing sites and the corresponding financial loss. Chee-wooi et al. [9] proposed a vulnerability assessment framework to systematically evaluate the vulnerabilities of SCADA systems at three levels: system, scenarios, and access points. S. Nazir et al. [10] provided a comprehensive survey of simulation, modelling and related techniques which are helpful for assessing the cyber-attack vulnerabilities of SCADA systems. In general, there have been many studies that combine machine learning and text mining to discover vulnerabilities or to classify vulnerability types automatically, while there are few studies on the severity of vulnerabilities. Vulnerabilities' descriptions are good resources to predict their severity levels. Spanos' work [2] provided a good reference to make connections between descriptions and severity levels. They used text mining to process the data crawled from the vulnerability database, used TF-IDF to obtain keywords as features, and evaluated the vulnerability severity level with SVM, decision tree and other classifiers. However, they did not consider the information contained in the low-frequency vocabulary, and the final prediction accuracy was only about 80%. This paper draws on their work and considers the information of low-frequency vocabulary. It uses the XGBoost algorithm to evaluate the vulnerability severity level and achieves an accuracy rate over 90%.

III. Vulnerability Severity Prediction Model

A. The Description of the Methodological Framework

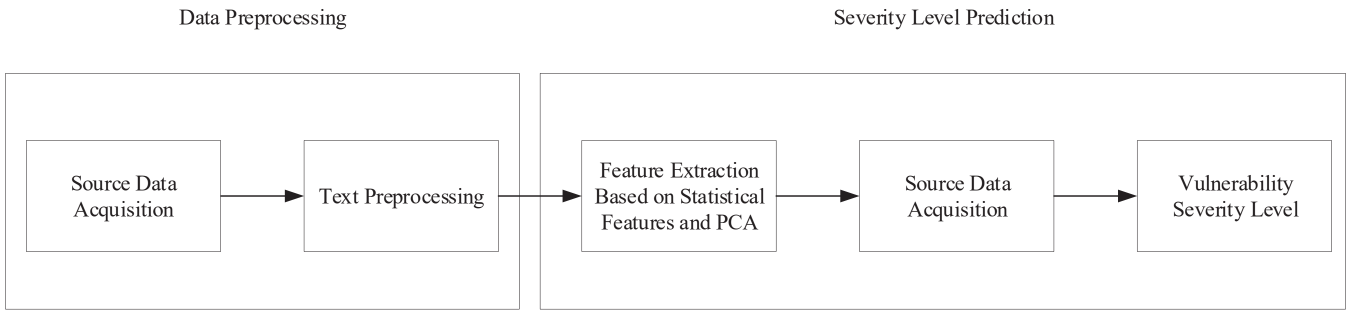
Data Preprocessing
Severity Level Prediction
Source Data Acquisition
Text Preprocessing
Feature Extraction Based on Statistical Features and PCA
Source Data Acquisition
Vulnerability Severity Level

Fig. 1. The framework of vulnerability severity prediction model

Figure 1 is the workflow framework of the proposed vulnerability severity prediction model of this paper, which contains two parts: data preprocessing and severity level classification. Data processing mainly solves the problem of data source and normalization, which can use crawler to crawl the required data from the cyber and use text mining to preprocess the acquired data. Severity level classification extracts features from processed data and uses XGBoost to intelligently evaluate the vulnerability severity level.

B. Data Preprocessing

Data preprocessing can be divided into two steps: source data acquisition and text preprocessing.

Open source vulnerability database usually contains the latest description of vulnerabilities, which is a comprehensive knowledge database including the release time, index, severity level and description information. For common users, the vulnerability description is the only source that can be used in its severity level's prediction. In this paper, we need to gather the text description and the severity level of the vulnerability. For example, a SQL injection vulnerability may have the following records in the NVD vulnerability database:

TABLE I. The sample of SQL injection vulnerability

Description	Severity level
The 'reportID' parameter received by the '/common/run report.php' script in the Quest KACE System Management Appliance 8.0.318 is not sanitized, leading to SQL injection (in particular, an error-based type)	High

Crawler is used to crawl the required data, and the crawled data contain description and severity level. These data can be represented as $\boldsymbol{D}_{vul} = \{D_{vul,1}, \dots, D_{vul,i}, \dots, D_{vul,m}\}$ and $\boldsymbol{I}_{vul} = \{I_{vul,1}, \dots, I_{vul,i}, \dots I_{vul,m}\}$, where vul represents the type of the crawled vulnerabilities, $D_{vul,i}$ represents the crawled i-th description of the vulnerability, and $I_{vul,i}$ represents the corresponding severity level. To be more general, we use CVSS v2 score to represent the severity level, in which the severity level can roughly be divided into three grades: High (score 7.0-10.0), Medium (score 4.0-6.9), Low (score 0.0-3.9) representing the possible impact of a vulnerability on the information system.

After obtaining sufficient text data, we apply text preprocessing to process the descriptions of the crawled data to prepare for the extraction of subsequent features. Here we use four steps: removing punctuation, removing stop words, removing words that are meaningless for analysis, and lemmatization. In our work we only consider English vulnerability database, which means that the words in one sentence can be directly divided according to spaces and inter-sentence punctuation. In this paper, the preprocessed record is represented as $\boldsymbol{D}'_{vul} = \{D'_{vul,1}, \dots, D'_{vul,i}, \dots, D'_{vul,m}\}$.

C. Vulnerability Severity level Prediction

Reasonable feature extraction paves the path for effective prediction of severity level. Firstly, the feature vector is constructed to quantitatively represent the pre-processed data. Then, we use XGBoost to predict the severity level of a vulnerability by using the constructed feature vector.

After preprocessing each record in the crawled document $\boldsymbol{D}_{vul}$, we count the occurrence time of each word in the document and establish a word bag model $\boldsymbol{B}_{vul}$ based on the document. After that, the vocabulary in the word bag model is arranged in descending order according to the appearance frequency and the sorted word bag model is obtained, $\boldsymbol{B}'_{vul} = \{(word_1, count_1), \dots (word_i, count_i) \dots, (word_n \ count_n)\}$, where $word_i$ represents the i-th word, and $count_i$ denotes the occurrence number of the corresponding word. Then the feature vector is established to quantitatively represent a record $D'_{vul,i}$. For one record, we select the top l words in $\boldsymbol{B}'_{vul}$ and count the appearance times of the corresponding words in the record $D'_{vul,i}$. The created vector is shown below:

$$v'_i = (N_{i,1}, \dots, N_{i,j}, \dots, N_{i,l})$$

In the vector, $N_{i,j}$ denotes the occurrence number of the word ranked in the j-th in the word bag model of record i. One word's frequency of occurrence in a document is varied. Thus, μ is manually defined as the frequency threshold. The appearance frequency of the words below μ is discarded in the word bag model $\boldsymbol{B}'_{vul}$ so that we can obtain the new word bag model $\boldsymbol{B}'_{vul,\mu}$ where $N_{B'_{vul,\mu}}$ represents the vocabulary number of the new model. In general, the manually defined μ will help to remove some words, but there are still many words left, and the statistical result of these words in a record as part of the corresponding vector which will have too many features. At the same time, these features are commonly sparse and will have an impact on the final classification results. If just considering high-frequency vocabulary, a lot of information will be lost. Therefore, we use PCA to reduce dimensions of the features other than high-frequency vocabulary.

PCA [11] is trying to achieve the aim of dimensionality reduction by reorganizing many original indicators with certain correlations into a new set of mutually independent comprehensive indicators instead of the original ones. A threshold ρ should be picked up for $\boldsymbol{B}'_{vul,\mu}$, then for the words of top ρ, the appearance time in $D'_{vul,i}$ is selected as feature directly. After that, PCA is utilized on remaining words and create v_i to represent record i:

$$v_i = (N_{i,1}, \dots, N_{i,\rho}, P_{i,1}, \dots, P_{i,k})$$

In the new vector, we add dimensionality-reduced data to the former vector v'_i. In this way, we use the obtained vector set $\boldsymbol{V} = \{v_1, \dots v_i, \dots, v_m\}$ to represent the quantitative representation of the crawled records.

After obtaining the features, here we use the XGBoost classifier to intelligently evaluate the vulnerability level based on them. XGBoost (eXtreme Gradient Boosting) is a boosting method using the CART regression tree, which can be formally represented by the model of K trees:

$$\widehat{I_{T,\iota}} = \sum_{k=1}^{K} f_k(v_i), f_k \in \boldsymbol{F}$$

In the formula, $\widehat{I_{T,\iota}}$ represents the prediction result, and $f_k(v_i)$ represents the corresponding results predicted by the CART tree using v_i, and $\boldsymbol{F}$ represents the set of all possible results of CART tree. In this paper, for the three severity levels of *high*, *medium* and *low*, the values of 2, 1, and 0 are respectively assigned.

IV. XSS Vulnerability Severity Prediction Experiment

To illustrate the applicability of our method, we carry out experiments with XSS vulnerability data crawled from the NVD vulnerability database. XSS is an abbreviation for Cross-Site Scripting; in order to distinguish it from Cascade Style Sheets (CSS), it is called XSS. When there exists one XSS vulnerability, the attacker may injects malicious code into the web page or the requesting data. When the victim browses the page or is tempted to click on a link containing malicious code, the XSS attack will be triggered.

Our experiment first uses the Requests[2] package which is written by Python 3.6 to program the crawler, and uses the crawler to crawl the XSS vulnerability description in the NVD[3] vulnerability database. Here we remove the data that has just appeared and been scored as no severity level. We use CVSS v2 score in our experiment, and obtain $\boldsymbol{D}_{XSS}$ which contains 8793 records. Some of them are shown in Table II below:

TABLE II. THREE SAMPLES OF XSS VULNERABILITY DATA

Description	Severity level
IBM Business Process Manager 8.6 is vulnerable to cross-site scripting. This vulnerability allows users to embed arbitrary JavaScript code in the Web UI thus altering the intended functionality potentially leading to credentials disclosure within a trusted session. IBM X-Force ID: 138135.	Low
In Zoho ManageEngine ServiceDesk Plus before 9403, an XSS issue allows an attacker to	Medium

[2] http://www.python-requests.org/en/master/

[3] https://nvd.nist.gov/

run arbitrary JavaScript via a /api/request/?OPERATION NAME=URI, aka SD-69139	
Cross-domain vulnerability in Apple Safari for Windows 3.0.1 allows remote attackers to bypass the \same origin policy" and access restricted information from other domains via JavaScript that overwrites the document variable and statically sets the document.domain attribute.	High

Then, each obtained data is text preprocessed. Firstly, we remove punctuation and collect common stop words from the Internet. Stop words appear frequently in sentences but have no impact on text analysis, such as *the*, *me*, *my* and so on. The amount of different stop words used in our experiment is 559, and we get rid of them in our data. Next, we remove the words which are meaningless to the analysis. The description of XSS vulnerability must mention its name, so we remove the following words: *xss*, *cross*, *site*, *scripting*, *cross-site*, *web* and *vulnerability*. Finally, lemmatization is performed with famous Python package NLTK[4]. After above procedures we create word bag $\boldsymbol{B}_{XSS}$ to represent words' situation of our data, and construct the sorted word bag model $\boldsymbol{B}'_{XSS}$. The word bag contains 24817 words, in which the highest frequency is 7499 while the lowest frequency is 1. The word frequency is drawn below, where we obtain the graph by adding up the frequencies of every 5 words in $\boldsymbol{B}'_{XSS}$:

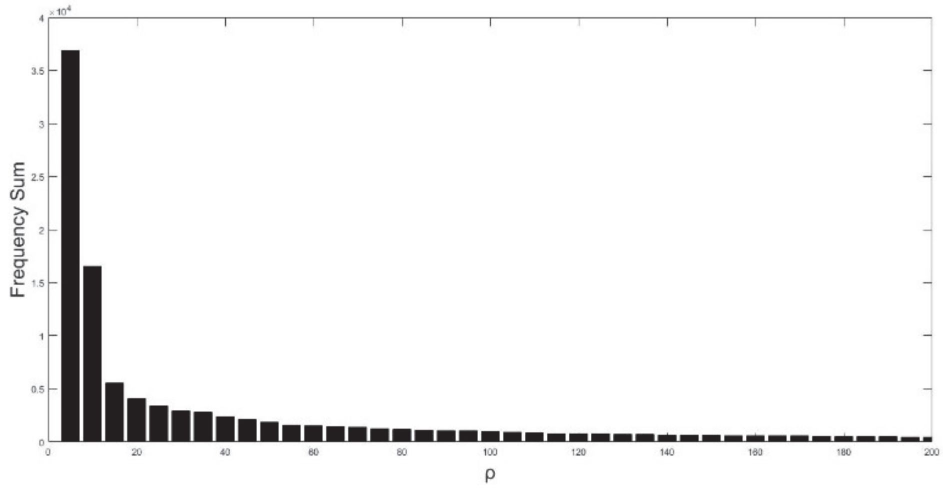

Fig. 2. The bar graph of sums of words frequency

As we can see from Figure 2, most of the words appear in the word bag very few times. Inspect the data we see up to 17737 words appear only once, and such low-frequency words will interfere with the subsequent assessment of the severity level. In our experiment, the threshold μ is set to 30, and we obtain $\boldsymbol{B}'_{XSS,30}$ containing the remaining 625 words, and the description of a vulnerability is quantitatively characterized by the number of these words in the corresponding vulnerability description. Here, we select the top 20 vocabulary words to create word cloud through Python package Wordcloud[5], and the result is shown in Figure 3:

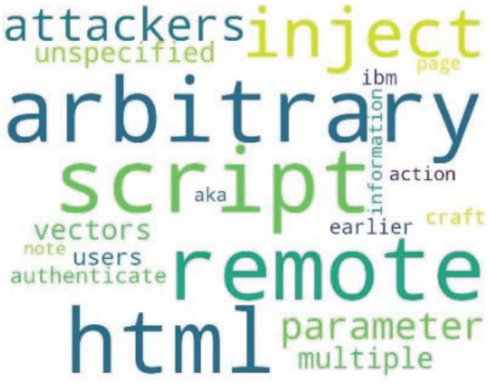

Fig. 3. The word cloud of high-frequency vocabulary

We can learn from the result that the high-frequency vocabulary contains many possible influences or implementation methods of an XSS attack. Taking different numbers of words directly as feature vectors will have different impacts on the final prediction of the severity level. In our experiment, let ρ increase by 5 from 5 to 200, and the top ρ words in $\boldsymbol{B}'_{XSS}$ are directly taken as features while we use PCA for the remaining words to reduce the dimension, so there are 80 sets of different feature vectors. Next, we use the XGBoost algorithm and the 10-fold cross validation method to evaluate the vulnerability severity level. At the same time, comparing to the methods only take the statistics of high-frequency words as features, the proposed method takes the frequencies less than ρ as features and others are directly discarded. In order to compare the effects of XGBoost to other algorithms, we carry out the experiments with Support Vector Machine (SVM), Logistic Regression (LR) and Random Forest (RF). The results are showing in the figure below:

[4] http://www.nltk.org/

[5] https://github.com/amueller/word_cloud

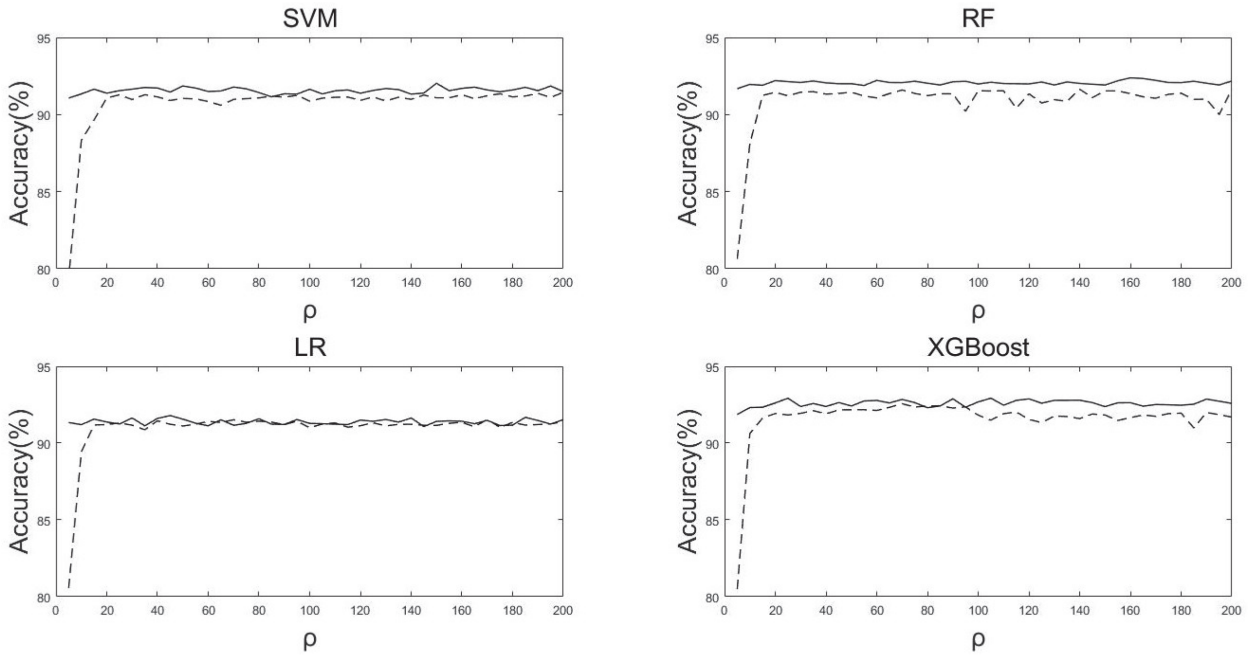

Fig. 4. The comparison experiment of different classification algorithms

Further observation of the results reveals that when ρ exceeds 40, the results tend to be stable. Here, the experimental results of ρ not exceeding 40 are as follows:

TABLE III. THE RESULTS OF PREDICTION EXPERIMENTS

ρ		**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**
The results of PCA extracting sparse vocabulary	SVM	91.069%	91.336%	91.649%	91.552%	91.552%	91.649%	91.757%	91.723%
	RF	91.679%	91.956%	91.899%	92.203%	92.138%	92.081%	92.177%	92.055%
	LR	91.342%	91.200%	91.563%	91.245%	91.245%	91.637%	91.126%	91.609%
	XGBoost	**91.870%**	**92.314%**	**92.337%**	**92.928%**	**92.928%**	**92.382%**	**92.581%**	**92.382%**
The results of original data	SVM	79.429%	88.340%	89.645%	91.282%	91.282%	90.975%	91.288%	91.146%
	RF	80.640%	88.130%	91.242%	91.441%	91.217%	91.450%	91.484%	91.325%
	LR	80.569%	89.434%	91.182%	91.308%	91.308%	91.168%	90.873%	91.453%
	XGBoost	80.480%	90.634%	91.634%	91.833%	91.833%	91.941%	92.114%	91.907%

As can be seen in Figure 4, the solid line denotes the result of extracting sparse words with PCA and combining it with high-frequency words, and the dashed line denotes the result of directly taking high-frequency words as features. It can be seen from the experimental results that the accuracy of the proposed method in this paper is generally higher than that of the direct use of high frequency vocabulary. Meanwhile, comparing these four algorithms, XGBoost gets the best result and the proposed method is more practical than others.

V. CONCLUSION

According to the descriptions of vulnerabilities, this paper uses text mining to extract keywords and take their statistical properties as features. Meanwhile, we use PCA to extract the principle components of the low-frequency words and establish feature vectors for quantitatively representing the descriptions of the vulnerabilities. The XSS vulnerability data obtained from NVD is processed by XGBoost algorithm and the experiment result shows that the accuracy is more than 90%. At the same time, compared with other commonly used algorithms, the proposed method achieves higher accuracy, which proves that it will help intelligently evaluate the vulnerability severity level effectively in time. In the future work, we would like to develop a practically available platform and integrate the proposed algorithm.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (Grant No. 61703416) and Natural Science Foundation of Hunan Province, China (Grant No. 2018JJ3614).

References

[1] X. Guo, S. Jin, and Y. Zhang, "XSS vulnerability detection using optimized attack vector repertory," 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). Xi'an, pp. 29-36, September 2015.

[2] G. Spanos, L. Angelis, and D. Toloudis, "Assessment of vulnerability severity using text mining," 21st Pan-Hellenic Conference. Larissa, pp. 1-6, September 2017.

[3] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine-learning and data-mining techniques: a survey," ACM Computing Surveys (CSUR), vol. 50, 2017.

[4] R. Feldman and J. Sanger, The Text Mining Handbook: Advanced Approaches In Analyzing Unstructured Data, 2006 p. 806.

[5] A. Hovsepyan, R. Scandariato, W. Joosen, and J. Walden, "Software vulnerability prediction using text analysis techniques," Proceedings of the 4th international workshop on Security measurements and metrics. pp. 7-10, 2012.

[6] Y. Yamamoto, D. Miyamoto, and M. Nakayama, "Text-mining approach for estimating vulnerability score," Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS). Kyoto, pp. 67-73, November 2015.

[7] D. Toloudis, G. Spanos, and L. Angelis, "Associating the Severity of Vulnerabilities with their Description," International Conference on Advanced Information Systems Engineering. Ljubljana, pp. 231-242, June 2016.

[8] X. Chen, I. Bose, A. C. M. Leung, and C. Guo, "Assessing the severity of phishing attacks: a hybrid data mining approach," Decision Support Systems, vol. 50, pp. 662-672, 2011.

[9] C. W. Ten, C. C. Liu, and G. Manimaran, "Vulnerability assessment of cybersecurity for SCADA systems," IEEE Transactions on Power Systems, vol. 23, pp. 1836-1846, 2008.

[10] S. Nazir, S. Patel, and D. Patel, "Assessing and augmenting SCADA cyber security: A survey of techniques," Computers & Security, vol. 70, pp. 436-454, 2017.

[11] H. Abdi and L. J. Williams, "Principal component analysis," Wiley Interdisciplinary Reviews Computational Statistics, vol. 2, pp. 433-459, 2010.