An automatic method for CVSS score prediction using vulnerabilities description

Article *in* Journal of Intelligent and Fuzzy Systems · August 2015

DOI: 10.3233/IFS-151733

3 authors, including:

Mohammad Ghasemzadeh
Hasso Plattner Institute
29 PUBLICATIONS **64** CITATIONS
SEE PROFILE

Vali Derhami
Yazd University
45 PUBLICATIONS **97** CITATIONS
SEE PROFILE

Journal of Intelligent & Fuzzy Systems 30 (2016) 89–96
DOI:10.3233/IFS-151733
IOS Press

An automatic method for CVSS score prediction using vulnerabilities description

Atefeh Khazaei*, Mohammad Ghasemzadeh and Vali Derhami
Department of Electrical and Computer Engineering, Yazd University, Yazd, Iran

Abstract. In this paper we introduce an objective method for CVSS score calculation. CVSS is a well known and mostly used method for giving priority to software vulnerabilities. Currently it is being calculated by some slightly subjective methods which require enough skill and knowledge. This research shows how we can benefit from natural language description of vulnerabilities for CVSS calculation. The data that were used for implementation and evaluation of the proposed models consists of the available CVE vulnerability descriptions and their corresponding CVSS scores from the OSVDB database. First, feature vectors were extracted using text mining tools and techniques, and then the SVM and Random-Forest algorithms as well as fuzzy systems were examined to predict the concerned CVSS scores. In spite of the fact that SVM and Random-Forest are mostly used and trusted methods in prediction, results of this research bear a witness that using fuzzy systems can give comparable and even better results. In addition, implementation of the fuzzy based system is much easier and faster. Although so far, there have been so little efforts in using the information embedded in textual materials regarding vulnerabilities, this research shows that it will be valuable to utilize them in systems security establishment.

Keywords: Description of software vulnerability, Common Vulnerability Scoring System (CVSS), Support Vector Machine (SVM), Random-Forest, fuzzy systems

1. Introduction

A vulnerability can be a bug, flaw, or event within an application, system, device, or service that could cause an implicit or explicit failure of confidentiality, integrity or availability [12]. Each year thousands of vulnerabilities are discovered and reported. Since few organizations have the resources to address every vulnerability that might affect their systems, administrators must prioritize their efforts to address the most critical vulnerabilities first.

Several practical organizations and systems that rank and/or assess risk of vulnerabilities exist. For example, CERT/CC (Computer Emergency Readiness Team/ Coordination Center) calculates a numeric score ranging (from 0 to 180) based on factors such as what sort of preconditions are required to exploit the vulnerability and whether the internet infrastructure is at risk [18]. The SANS (deriving from System admin, Audit, Networking, and Security) vulnerability analysis scale considers the weakness whether it is found in default configurations or server or client systems [16]. Microsoft's proprietary scoring system reflects the difficulty of exploitation and the overall impact of the vulnerability [10]. FIRST's (Forum of Incident Response and Security Teams) Common Vulnerability Scoring System (CVSS) is a quantitative metric and uses simple formulas that return a value as severity of vulnerability [3].

But in order to compare vulnerabilities with each other a standard metric must exist. The CVSS appears to be the emerging standard in community. The CVSS is composed of three metric groups: Base, Temporal,

*Corresponding author. Atefeh Khazaei, Department of Electrical and Computer Engineering, Yazd University, Yazd, Iran. Tel.: +98 35 31232359; Fax: +98 35 31232357; E-mail: atefeh.khazaei@stu.yazd.ac.ir.

```
BaseScore = round_to_1_decimal(((0.6*Impact)+(0.4*Exploitability)-1.5)*f(Impact))

Impact = 10.41*(1-(1-ConfImpact)*(1-IntegImpact)*(1-AvailImpact))

Exploitability = 20* AccessVector*AccessComplexity*Authentication

f(impact)= 0 if Impact=0, 1.176 otherwise

AccessVector       = case AccessVector of
                          requires local access: 0.395
                          adjacent network accessible: 0.646
                          network accessible: 1.0

AccessComplexity   = case AccessComplexity of
                          high: 0.35
                          medium: 0.61
                          low: 0.71

Authentication     = case Authentication of
                          requires multiple instances of authentication: 0.45
                          requires single instance of authentication: 0.56
                          requires no authentication: 0.704

ConfImpact         = case ConfidentialityImpact of
                          none: 0.0
                          partial: 0.275
                          complete: 0.660

IntegImpact        = case IntegrityImpact of
                          none: 0.0
                          partial: 0.275
                          complete: 0.660

AvailImpact        = case AvailabilityImpact of
                          none: 0.0
                          partial: 0.275
                          complete: 0.660
```

Fig. 1. The CVSS base score formula [12].

and Environmental [12]. But often "Base Score" is used in databases and researches. Base Score represents the innate characteristics of each vulnerability that is not affected by time, place or system. Figure 1 shows the CVSS base score formula.

Although CVSS is a public standard, but as with above scoring systems, it has few restrictions. The CVSS score calculation (as can be seen in Fig. 1) can be somewhat subjective and two users may be having different CVSS scores. On the other hands, the users should be familiar with both, the vulnerability characteristics and CVSS scoring systems.

The goal of this study is predicting the CVSS base scores based on vulnerability description using text mining. Describing the vulnerabilities to natural language is easier than to find and compute CVSS base score parameters.

Until now, few attentions have been paid to textual information hidden in vulnerability databases. Text mining can be an appropriate tool to provide knowledge for administrative decisions.

The rest of this paper is organized as follows: related works in this area are dealt with in Section 2. Section 3 discusses the method. The experiments and their results will be discussed in Section 4. Finally, Section 5 discusses and concludes the paper.

2. Related works

Until now, little attention has been paid to textual information hidden in vulnerabilities databases. Based on the Bozorgi et al. claim, they were the first researchers that focused on this issue. Their research explores how to build predictive models based on the information hidden in these texts. Their results showed that text mining can be an appropriate tool to provide knowledge for administrative decisions. In their

work, most features were binary and extracted from text fields. These binary features were derived using a bag-of-words representation for each text field. After feature extraction and creating feature vector for each vulnerability, classifiers were trained and they predicted exploitation [8].

After the Bozorgi et al. paper, researchers have paid more attention to vulnerability textual information. For example, Mokhov et al. employed natural language processing (NLP) and artificial intelligence (AI) methods for reporting vulnerabilities, using static code analysis in artificial constructed languages (e.g. programming languages) [15]. In another research, Scandariato et al. focused on predicting the vulnerable components of software. Their approach is also based on text mining the source code of the components. It was tested with 20 Android applications and had acceptable results [13].

The previous results of researches showed that text mining can be an appropriate tool for the security research field. The goal of our study is to predict the CVSS score using vulnerability descriptions. Based on the best of our knowledge, no research has focused on CVSS score prediction base on vulnerability textual descriptions, until now. Our paper results confirmed the potential of text mining methods in predicting the severity of software vulnerabilities.

3. Method

In the first step for the CVSS score prediction, the suitable data should be aggregated. Then, the appropriate features should be extracted from it. Data selection and feature extraction will be discussed in Section 3-1. The extracted features from vulnerability descriptions are high-dimensional. To increase training speed and the accuracy of classifiers, dimension reduction methods were used. In Section 3-2 the used dimension reduction methods were explained. To use the Support Vector Machine (SVM) and Random-Forest algorithms, the samples should be labeled. The labeling method was described in section 3-3.

3.1. Data and feature extraction

Overall vulnerability databases can be divided to two categories: public databases such as CVE (Common Vulnerabilities and Exposures) and OSVDB (Open Source Vulnerability Database), and vendor's databases such as MFSA (Mozilla Foundation Security Advisories). In public databases, there are many vulnerability reports from various applications and they are generally related to products with large number users. Vulnerabilities in vendor's databases are related to certain products and generally are not available to the public. OSVDB and CVE are famous public vulnerability databases which have been used in this research.

OSVDB is a large database containing reports on over 100,000 vulnerabilities until end of 2013 [11]. CVE provides the standard for information security vulnerability names and it contains over 60,000 vulnerabilities until end of 2013 [1]. In this study, the CVE vulnerability descriptions and the CVSS base scores from OSVDB were used. The OSVDB and CVE records were integrated and number of samples was 19320.

To create feature vectors, in the first step, words were extracted from vulnerability descriptions. Then stop-words were removed. Stop-words are words that almost never have any capability to distinguish documents, such as articles *a* and *the* and pronouns such as *it* and *them.* These common words can be discarded before completing the feature generation process [9]. In the next step, remaining words were stemmed. Stemming is the process of reducing words to their stem and it reduces different forms of words [9]. After these steps remaining words were considered as features. Feature vector length was 8965. Finally TF-IDF (Term Frequency – Inverse Document Frequency) values were calculated and were used as value of features. It should be noted that TF-IDF is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate the importance of a word in a document, in a collection of documents. TF-IDF is

$$f_{ij} \log \frac{\text{number of documents}}{\text{number of documents that include word } i}$$

In this formula f_{ij}, is frequencies for word i in document j. In TF-IDF, the term frequency is modulated by a factor that depends on how the word is used in other documents [9]. If the word is in the document, the value of TF-IDF is not equal to zero. Otherwise, when the word does not exist in the document its value for the vector is zero. WVT (word vector tool) was used to extract these features [17]. The WVT is a flexible Java library for statistical language modeling. In particular it is used to create word vector representations of text documents. Figure 2 shows the feature extraction steps.

3.2. Dimension reduction

Extracted features from vulnerability descriptions are high-dimensional (vector length is 8965). For

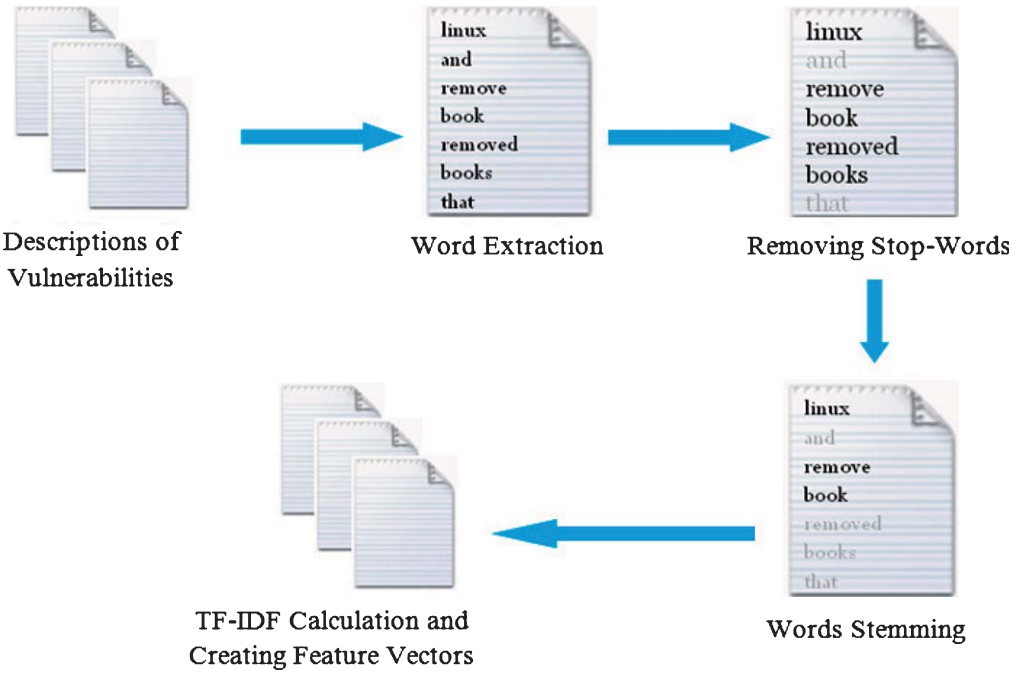

Fig. 2. Feature extraction steps.

high-dimensional data, training speed is slow. Therefore, to increase training speed and possibly the accuracy of classifiers, the LDA (Linear Discriminant Analysis) and PCA (Principal Component Analysis) dimension reduction methods were used.

The PCA is a mathematical procedure to convert a set of possibly correlated variables into a set of uncorrelated variable values. The number of principal components is less than or equal to the number of original variables [4]. The PCA returned the 6923 features for the extracted feature vector from description of vulnerabilities. If a certain number of features are considered, the same number of the best PCA features can be selected.

The LDA and the PCA are closely related to each other. Both of them look for linear combinations of features which best explain the data. The LDA explicitly attempts to model the difference between the classes of data. But, the PCA does not take into account any difference in class [4]. In this study, the LDA returned the nine features for ten classes. Deng Cai's MATLAB implications for LDA and PCA were used in this study [2].

3.3. Label of samples

The CVSS score has the continuous value between zero and ten as severity of vulnerability. Whatever the CVSS score is closer to ten, the vulnerability is more critical; and whatever the score is closer to zero, the vulnerability is more trivial.

To use the most of classifiers, for example SVM and Random-Forest, the number of classes should be limited. So the continuous CVSS scores could not be used for these classifiers.

To determine the label of samples, CVSS interval (that is [0 10]) was discretized to ten equal subintervals, and each of them is a class. For example, the CVSS score between zero and 0.9 is a member of the Zero-Class, the score between 1 and 1.9 is a member of the One-Class, and so on the score between 9 and 10 is a member of Nine-Class.

4. Evaluation method and results

In this study, the experiments are divided into two categories. The offline and online CVSS prediction experiments will be described in Section 4-1 and 4-2 respectively.

4.1. Offline CVSS prediction

4.1.1. SVM and Random-Forest classifiers

The SVM is a classifier that works effectively with large datasets. It finds the optimal separating

Table 1
The SVM and Random-Forest results

		SVM	Random-Forest
Without data dimension reduction	Training	57.8%	86.92%
	Testing	55.89%	66.09%
The LDA dimension reduction	Training	**86.46%**	99.1%
	Testing	**86.23%**	85.78%
The LDA dimension reduction on the PCA features	Training	86.45%	99.14%
	Testing	86.11%	85.3%

hyper-plane. The hyper-plane is a decision boundary that separates the samples of one class from each other [4]. Many software packages are available for the SVM algorithm; in this study, the LIBSVM implementation [7] was used.

The Random-Forest is a classification method that consists of many decision trees. This method builds a randomized decision tree, in each iteration of the bagging method [4]. Many software packages are available for this algorithm, too. The Random-Jungle [14] which is an implementation of Random-Forest for high dimensional data was deployed.

In offline experiments, the classifiers were trained with two thirds of the samples (12880 samples) and the remaining (6440 samples) were used for testing. Members of these two sample sets were selected randomly from 19320 vulnerability samples. It should be noted that the averages of several rounds of experiments were presented in this paper.

Three cases were considered for the SVM and Random-Forest experiments:

- Using the 8965 features (without data dimension reduction)
- Using the LDA dimension reduction
- Using the LDA dimension reduction on the PCA features

Table 1 shows a summary of these experiments. As it is shown in Table 1, best case obtained from the SVM classifier with the LDA features.

The quality of SVM classifiers is better than the Random-Forest classifiers, because there are small differences between training and testing accuracies. The results of Table 1 showed that the Random-Forest classifiers somewhat fitted the training set better than the test set.

4.1.2. Fuzzy system

The design methods of fuzzy systems are divided to four categories: 1) Modeling using expert knowledge, 2) Modeling using numerical data (rule extraction), 3) Mathematical modeling, and 4) Hybrid modeling [5, 6]. In this study, a combination of expert knowledge and numerical data modeling was used. The LDA features were used in our fuzzy system. This system has the following characteristics:

- The Mamdani fuzzy system
- Nine inputs (LDA features) with 10 Gaussian membership functions for each input (Fig. 3)
- One output (that is CVSS score) with 10 Gaussian membership functions (Fig. 4)
- Ten rules (one rule for each class) (Fig. 5)
- Inference mechanism:
  - And method: Product
  - Implication method: Product
  - Aggregation method: Sum
- Centroid defuzzifier

Type of membership functions is Gaussian. Gaussian membership function formula is

$$e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2}$$

The c is center and the σ is standard deviation.

When creating membership functions of inputs, for each class a membership function for each input was defined. The center of each Gaussian membership function was the average of feature values in its class. In these experiments some different σ were examined. The membership functions of inputs that obtained from training data was plotted in Fig. 3.

Same as the previous section, fuzzy systems were created with two thirds of the samples and the remaining of them was used for testing. The results of these systems were presented in Table 2. Table 2 shows that different values for σ have no significant impact on the results. This Mamdani fuzzy system has 88% accuracy and this is better than the SVM and Random-Forest classifiers. The small differences between training and testing accuracies show that overfitting to training data did not occur.

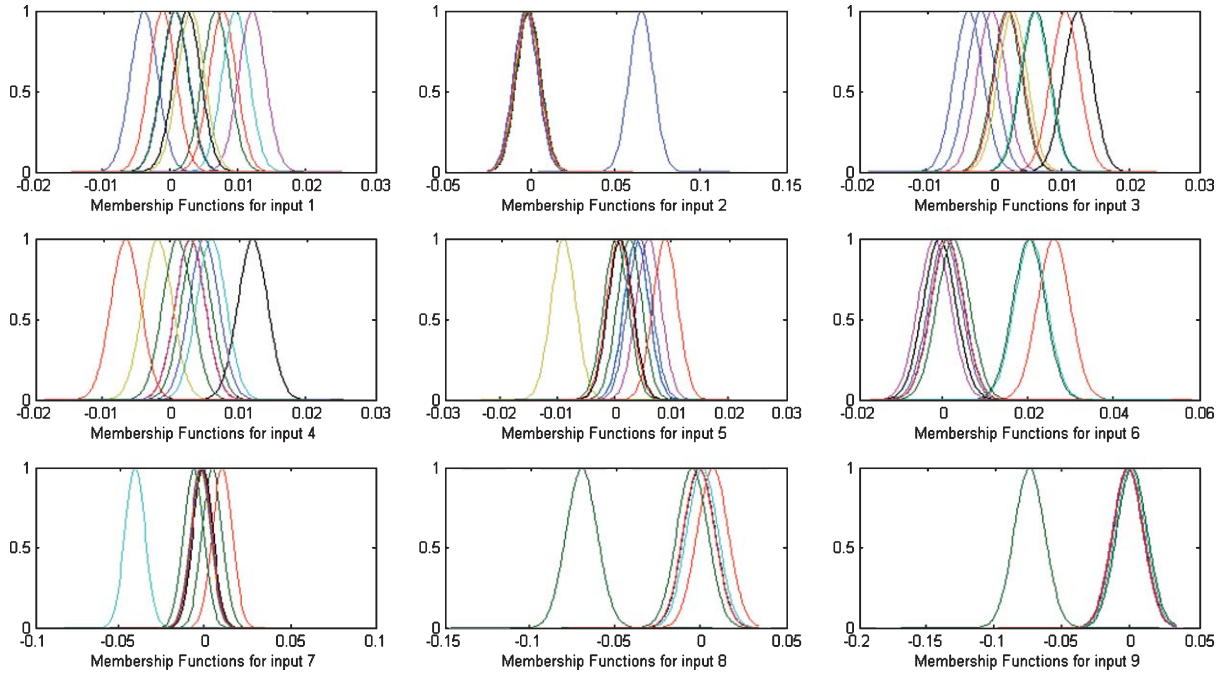

Fig. 3. Inputs membership functions for fuzzy system.

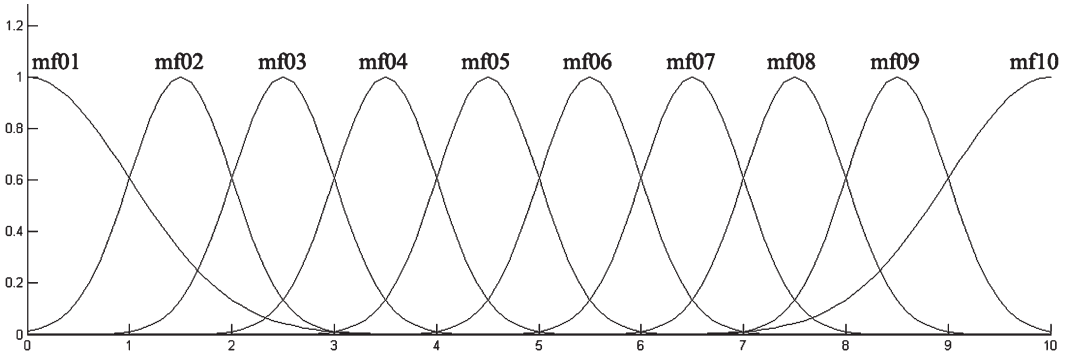

Fig. 4. Output membership functions for fuzzy system.

1) **If** (Input1 is mf01) and (Input2 is mf01) and (Input3 is mf01) and (Input4 is mf01) and (Input5 is mf01) and (Input6 is mf01) and (Input7 is mf01) and (Input8 is mf01) and (Input9 is mf01) **then** (Output is mf01) (1)

2) **If** (Input1 is mf02) and (Input2 is mf02) and (Input3 is mf02) and (Input4 is mf02) and (Input5 is mf02) and (Input6 is mf02) and (Input7 is mf02) and (Input8 is mf02) and (Input9 is mf02) **then** (Output is mf02) (1)

3) **If** (Input1 is mf03) and (Input2 is mf03) and (Input3 is mf03) and (Input4 is mf03) and (Input5 is mf03) and (Input6 is mf03) and (Input7 is mf03) and (Input8 is mf03) and (Input9 is mf03) **then** (Output is mf03) (1)

Fig. 5. The examples of fuzzy system rules.

Table 2
The fuzzy system results

Gaussian MFs σ	Training accuracy	Testing accuracy
(Range of Inputs)/10	87.07%	86.71%
(Range of Inputs)/20	**88.01%**	**88.37%**
(Range of Inputs)/30	88.38%	87.84%
(Range of Inputs)/40	88.32%	88.09%

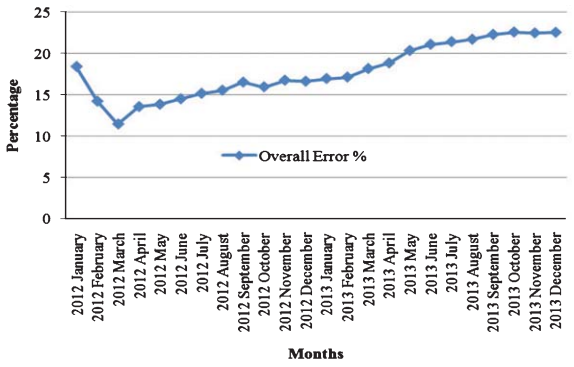

Fig. 6. Cumulative error for predicting CVSS scores in an online deployed setting.

4.2. Online CVSS prediction

The offline experiment in the previous section showed the potential of text mining for CVSS score prediction. But in a real-world vulnerability discovery come over time and gradually; the training set should extend with new vulnerabilities and train new classifiers based on both the most recent information and the whole bulk of data.

In this part of experiment, a real world was emulated with the best proposed fuzzy system in the offline settings. A date was considered as the beginning and classifier was trained with the samples whose disclosed dates were before the beginning. The next month samples were shown to this classifier as test samples. In the next step, classifier was updated with the previous and new month samples. This process was repeated every month.

To evaluate this experiment performance, cumulative error was used. Cumulative error is calculated by number of prediction errors for one-month and summed up with total number of previous months errors, divided by the total number of vulnerabilities observed up to the day.

Beginning of 2012 up until the end of 2013 period was considered. Figure 6 shows the total classification error over the time period. These results show that after the initial fluctuations, the classifier was stabilized and at the end, the overall error rate of 22.5% was obtained. This stability in Fig. 6 demonstrates the feasibility of deploying a classifier in an online setting to the CVSS score prediction.

5. Discussion and conclusion

The tens of thousands of vulnerabilities were discovered until now. But almost all organizations have the limited resources to address all vulnerabilities that might affect their systems. The CVSS is the standard prioritization system for vulnerabilities. The calculation of CVSS score is somewhat subjective and the users should be familiar with both, the vulnerability characteristics and the CVSS scoring systems.

The goal of this study was predicting the CVSS base scores. This prediction was based on vulnerability descriptions and text mining techniques. The results of this study showed the potential of text mining for CVSS score prediction.

To create the CVSS predictors the SVM and Random-Forest algorithms and fuzzy systems were examined. Using of these automatic predictors reduces the human errors and increases the CVSS calculation speed.

The best predictor was obtained with a fuzzy system and it can be predicted the CVSS score with 88% accuracy. In comparison to the SVM and Random-Forest algorithms, the fuzzy systems were also much easier and faster.

One area of future work will be to merge different predictors. The bagging and boosting methods can be used and may increase the accuracy of CVSS prediction.

References

[1] CVE Editorial Board. Common Vulnerabilities and Exposures: The Standard for Information Security Vulnerability Names, http://cve.mitre.org/, (last visited 2014-01-01).

[2] D. Cai, *Spectral Regression: A Regression Framework for Efficient Regularized Subspace Learning*, PhD Thesis, Department of Computer Science, UIUC 2009.

[3] Forum of Incident Response and Security Teams (FIRST), Common Vulnerabilities Scoring System (CVSS), http://www.first.org/cvss/, (last visited 2014-25-12).

[4] J. Han, M. Kamber and J. Pei, *Data Mining: Concepts and Techniques*. Third Edition, Morgan Kaufmann Publishers, 2011.

[5] J.S. Jang, C.T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice Hall, 1997.

[6] L.X. Wang, *A Course in Fuzzy Systems and Control*, Prentice-Hall International Inc, 1997.

[7] LIBLINEAR: A Library for Large Linear Classification. http://www.csie.ntu.edu.tw/∼cjlin/liblinear/, (last visited 2014-01-01).

[8] M. Bozorgi, L.K. Saul, S. Savage and G.M. Voelker, Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. KDD'10, 2010.

[9] M.W. Sholom, I. Nitin and Z. Tong, *Fundamentals of Predictive Text Mining*, Springer Publishing Incorporated Company, 2010.

[10] Microsoft Corporation. Microsoft Security Response Center Security Bulletin Severity Rating System, http://www.microsoft.com/technet/security/bulletin/rating.mspx, (last visited 2014-25-12).

[11] OSVDB: The Open Source Vulnerability Database, http://osvdb.org/, (last visited 2014-01-01).

[12] P. Mell, K. Scarfone and S. Romanosky, A Complete Guide to the Common Vulnerability Scoring System Version 2.0. Forum of Incident Response and Security Teams (FIRST), 2007.

[13] R. Scandariato, J. Walden, A. Hovsepyan and W. Joosen, Predicting vulnerable software components via text mining, *Software Engineering, IEEE Transactions on* **40**(10) (2014).

[14] Random Jungle, http://www.randomjungle.org, (last visited 2014-01-01).

[15] S.A. Mokhov, J. Paquet and M. Debbabi, The Use of NLP Techniques in Static Code Analysis to Detect Weaknesses and Vulnerabilities, *In Advances in Artificial Intelligence, 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014*, Montréal, QC, Canada, 2014.

[16] SANS Institute. SANS Critical Vulnerability Analysis Archive, http://www.sans.org/newsletters/cva/, (last visited 2014-25-12).

[17] The Word Vector Tool, http://wvtool.sf.net, (last visited 2014-01-01).

[18] United States Computer Emergency Readiness Team (US-CERT), US-CERT Vulnerability Note Field Descriptions. http://www.kb.cert.org/vuls/html/fieldhelp, (last visited 2014-25-12).