Text-mining Approach for Estimating Vulnerability Score

Daisuke Miyamoto*, Yasuhiro Yamamoto†, Masaya Nakayama*
*Information Technology Center
The University of Tokyo
2-11-16 Yayoi, Bunkyo-ku, Tokyo 113-8658, JAPAN
{daisu-mi, nakayama}@nc.u-tokyo.ac.jp

†Graduate School of Engineering
The University of Tokyo
7-3-1 Hongou, Bunkyo-ku, Tokyo 113-8656, JAPAN

***Abstract*—This paper develops a method that can automatically estimate the security metrics of documents written in natural language. Currently, security metrics play an important role in assessing the impact and risks of cyberthreats. Security metrics also enable operators to recognize emerging cyberthreats and to prioritize operations in order to mitigate such threats. In this paper, we focus on estimating the ratings in the Common Vulnerability Scoring System by inspecting the threats described in the Common Vulnerability and Exposures dictionary. Our approach employs various techniques for processing natural language, and it uses the descriptions in the dictionary to estimate the base metrics. This paper also extends the algorithm to increase the accuracy of the estimate.**

***Keywords*—Security Information, Data Mining, Scoring**

I. Introduction

Most modern systems rely on software, and software bugs often increase the risk that remote attackers can gain unauthorized access to such systems. Therefore, it is important to find suitable methods for managing this vulnerability in order to protect society from these attacks. However, the number of vulnerabilities increases along with the amount of software, since bugs pervade every level of modern software [1]. Therefore, it is important to share information in order to form a knowledge base that can facilitate management of these vulnerabilities.

The National Vulnerability Database (NVD) is a popular knowledge base. It is composed of the Common Vulnerability and Exposures (CVE) dictionary of vulnerabilities [2] and the Common Vulnerability Scoring System (CVSS) [3], which estimates the impacts of those vulnerabilities.

However, the announcement of impacts can be delayed by a day or two following receipt of reports by the Computer Security Division of the National Institute of Standards and Technology (NIST), the organization that manages the NVD. This delay can potentially increase the risks, since it can delay the awareness of serious vulnerabilities.

In this paper, we present a way to rapidly estimate the impacts predicted by the CVSS, and we develop an automated method for estimating these impacts. We then use techniques for processing natural language to analyze the CVE descriptions, and we classify the documents and estimate the base metrics of the CVSS. We preformed a preliminary experiment to determine a suitable estimation method; we compared several machine learning algorithms: the naive Bayes classifier, Latent Dirichlet allocation (LDA) [4], Latent Semantic Indexing (LSI) [5], and Supervised LDA (SLDA) [6]. As the training dataset, we considered approximately 60,000 vulnerabilities reported during the period January 2002 to December 2013 and we used 1,300 definitions reported during the period January 2014 to May 2014.

We also propose a new learning algorithm that introduces an annual parameter. Within the algorithm, the training dataset is separated by year, and a model is generated for each year. For each model, the algorithm assigns a weight that reflects the annual effects of the CVE documents. Our results indicate that this often improves the estimate.

This paper makes the following contributions:

- We propose a method for estimating the CVSS base metrics for CVE documents.
- We perform experiments and evaluate the performance with f_1 measures.
- We observe that the SLDA gives the best performance.
- We find that the annual effect of the CVEs is a feasible parameter for providing better estimates.
- We design an algorithm to assign weights that reflect the annual effect.

II. Related Work

This section briefly explains our analysis targets, the CVE and the CVSS, which are presented in Sections II-A and II-B, respectively. Section II-C introduces earlier work that attempted to analyze the threats described by the CVE.

A. Common Vulnerability and Exposures

The CVE dictionary contains publicly known information about security vulnerabilities and exposures [2]. It is composed of identifiers (CVE-IDs) and the descriptions of the vulnerabilities. The CVE-IDs are assigned by the MITRE Corporation

TABLE I. AN EXAMPLE OF A CVE DESCRIPTION: CVE-2014-0001

```
<vuln:summary>
Buffer overflow in client/mysql.cc in Oracle
MySQL and MariaDB before 5.5.35 allows remote
database servers to cause a denial of service
(crash) and possibly execute arbitrary code
via a long server version string.
</vuln:summary>
```

TABLE II. AN EXAMPLE OF A CVSS DESCRIPTION: CVE-2014-0001

```
<cvss:score>7.5</cvss:score>
<cvss:access-vector>NETWORK</cvss:access-vector>
<cvss:access-complexity>LOW</cvss:access-complexity>
<cvss:authentication>NONE</cvss:authentication>
<cvss:confidentiality-impact>PARTIAL</cvss:confidentia
lity-impact>
<cvss:integrity-impact>PARTIAL</cvss:integrity-impact>
<cvss:availability-impact>PARTIAL</cvss:availability-i
mpact>
```

and the CVE Numbering Authorities [7]; there are currently more than 50,000 CVE-IDs.

In this paper, we regard the CVE descriptions as explanatory variables. Table I shows an example of a CVE that was reported in 2014.

B. Common Vulnerability Scoring System

The CVSS provides an open framework for communicating the characteristics and impacts of information security vulnerabilities [3]. It was originally defined and commissioned by the National Infrastructure Advisory Council [8], and it is currently managed by the Forum of Incident Response and Security Teams [9], which seeks to promote and improve the framework.

According to the CVSS, there are three assessment criteria: the base, temporal, and environmental metrics. The base metrics represent the fundamental characteristics of a given vulnerability, and they are neither time dependent nor affected by the user's environment. The temporal metrics reflect the current characteristics of a vulnerability that changes over time. The environmental metrics reflect the characteristics of vulnerabilities that are specific to a user's environment, and they also reflect the potential collateral damage of the vulnerability and the security requirement.

In this paper, for the period 1999 to 2014, we were able to obtain the vulnerability score from the base metrics; however, we were not able to do so for the other metrics. Therefore, in this paper, we employ the base metrics as the objective variable. Table II shows an example of the base metrics. The vulnerability score ranges from 0.0 to 10.0; we consider the scores in the range 7.0 to 10.0 to be high, those in the range 4.0 to 6.9 to be medium, and those in the range 0.0 to 3.9 to be low. The following factors were used to calculate the base metrics.

- Access Vector (AV) denotes the place where the vulnerability is accessed. This information is categorized as *LOCAL*, *ADJACENT NETWORK*, or *NETWORK*.
- Access Complexity (AC) denotes the difficulty of the conditions required to exploit the vulnerability. This is categorized as *HIGH*, *MIDDLE*, or *LOW*.

TABLE III. PRELIMINARY ANALYSIS OF THE PERFORMANCE OF OUR METHOD FOR ESTIMATING BASE METRICS

	Category	#	NBC	LSI	LDA	SLDA
AV	LOCAL	115	0.607	0.086	-	0.621
	ADJACENT NETWORK	41	-	0.067	-	-
	NETWORK	1158	0.962	0.609	0.981	0.962
AC	HIGH	50	-	0.084	0.069	-
	MEDIUM	624	0.650	0.456	0.644	0.676
	LOW	640	0.715	0.124	0.009	0.718
AU	MULTIPLE INSTANCES	5	-	-	-	-
	SINGLE INSTANCE	239	0.371	-	-	0.451
	NONE	1070	0.920	0.868	0.893	0.925
A	NONE	499	0.814	0.204	0.116	0.823
	PARTIAL	459	0.659	0.485	0.492	0.623
	COMPLETE	356	0.667	0.005	0.033	0.693
C	NONE	480	0.755	0.041	0.555	0.792
	PARTIAL	565	0.742	0.573	0.266	0.708
	COMPLETE	269	0.570	0.007	0.065	0.624
I	NONE	420	0.697	0.048	0.035	0.751
	PARTIAL	639	0.761	0.630	0.617	0.741
	COMPLETE	255	0.576	-	0.025	0.622

- Authentication (AU) denotes the number of times that the attacker must authenticate in order to exploit the vulnerability. This is categorized as *MULTIPLE INSTANCES*, *SINGLE INSTANCE*, or *NONE*.
- Availability (A) denotes the impact on the availability of a system when the system is attacked. This is categorized as *NONE*, *PARTIAL*, or *COMPLETE*.
- Confidentiality (C) denotes the impact on the confidentiality of the data in a system when the system is attacked. This is categorized as *NONE*, *PARTIAL*, or *COMPLETE*.
- Integrity (I) denotes the impact on the integrity of a system when the system is attacked. This is categorized as *NONE*, *PARTIAL*, or *COMPLETE*.

C. Analysis of CVE

Earlier analyses of CVEs focused on extracting topics from security vulnerability information. Neuhaus and Zimmerman [10] used CVEs published during the period 1999 to 2009 to analyze the trend of cyberthreats. They used the LDA [4] to classify the CVEs into 40 categories, such as cross-site scripting, SQL injection, and buffer overflows. Their results showed that by eliminating these vulnerabilities and by making PHP more secure, the majority of the CVEs were fixed.

To the best of our knowledge, no past study attempted to analyze the CVEs in order to assess the possible impact of a given vulnerability, and thus the CVSS score was not calculated.

III. METHODOLOGY

In this paper, we decided to use the NVD [11] provided by the NIST. The NVD consists of the CVE-IDs and descriptions provided by MITRE [12] and the CVSS base metrics calculated by the NIST team.

When extracting the CVE descriptions, we used the Porter stemming algorithm [13] to remove the inflectional endings from words. The list of the stopwords used in the algorithm was available online [14].

We considered various methods to determine a suitable algorithm for the analysis. An earlier analysis of CVEs [10]

showed that LDA [4] is a feasible method. LDA is a probabilistic model for extracting topics from a corpus of documents, and it uses dimensional reduction to determine the co-occurrence patterns of texts within the documents. The naive Bayes classifier (NBC), a common method for classifying texts, is also feasible. We attempted to use it with the multivariate Bernoulli model. For the dimension reduction, we also considered the LSI [5]. It should be noted that the NBC, LSI, and LDA each use unsupervised learning. We also considered using SLDA [6] to deal with the labeled documents. Since the NVD can be regarded as a set of labeled documents, we used SLDA to classify the vulnerability information along with the topics.

We then performed a preliminary evaluation with the aim of using text mining to estimate the impact score. For this evaluation, we used 10-fold cross-validation and determined the average performance. We used the f_1 measure, which is defined as follows:

$$f_1 measure = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}. \quad (1)$$

The results are summarized in Table III, where the first column denotes the factors defined above. The second column gives the category of each factor, and the third shows the number of the test datasets that were found to belong to each category. The remaining columns denote the f_1 measures for the NBC, LSI, LDA, and SLDA, respectively. The symbol $-$ means that the f_1 measure could not be calculated because both the recall and the precision were zero. The training data were a set of CVEs published during the period 2012 to 2013 (CVE-2012 to CVE-2013), and the test dataset was a set of CVEs published during the period January to May 2014 (CVE-2014).

In the preliminary evaluation, we observed that the SLDA usually performed better than the other algorithms. We note that the dimension reduction hindered both the LSI and the LDA. We tried to classify the CVEs as High, Medium, or Low without regard to genre, but we note that the topics estimated from an unsupervised model may correspond to genres, if that is the dominant structure in the corpus [6].

IV. Assignment of Annual Weights for the SLDA

In this section, we present our idea for a way to further improve the estimate of the vulnerability score. We consider a way to extent the SLDA algorithm by using the annual effect of the CVEs.

A. Annual analysis for the SLDA

Figure 1 summarizes the structure of the SLDA, where d denotes the documents, n denotes the identifier of the words, and k denotes the identifier of the topics. D, N, and K represent the number of the documents, words, and topics, respectively. Like the LDA, the SLDA assigns the label Y_d to a document; this is based on the topic $Z_{d,n}$, which is the topic for document d and word n, and the parameter n_{η,σ^2}. Note that α and β are the Dirichlet parameters; the former denotes the per-document topic distribution and the latter denotes the per-topic word distribution. θ_d is the topic distribution for the document d, and ϕ_k is the word distribution for the topic k.

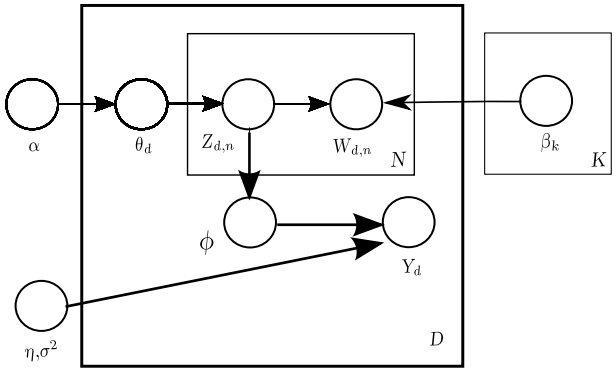

Fig. 1. Structure of the SLDA

TABLE IV. Conditions for identifying the relationship between the estimation performance and the temporal distance

#	Training Data	Testing Dataset	#	Training Data	Testing Dataset
1	CVE-2013	CVE-2014	13	CVE-2007	CVE-2008
2	CVE-2012	CVE-2014	14	CVE-2006	CVE-2008
3	CVE-2011	CVE-2014	15	CVE-2005	CVE-2008
4	CVE-2010	CVE-2014	16	CVE-2004	CVE-2008
5	CVE-2009	CVE-2014	17	CVE-2003	CVE-2008
6	CVE-2008	CVE-2014	18	CVE-2002	CVE-2008
7	CVE-2007	CVE-2014	19	CVE-2001	CVE-2008
8	CVE-2006	CVE-2014			
9	CVE-2005	CVE-2014			
10	CVE-2004	CVE-2014			
11	CVE-2003	CVE-2014			
12	CVE-2002	CVE-2014			

The SLDA requires two steps to predict the category. The first is the expectation step (E-step), which estimates the approximate posterior distribution for each document-response pair. The next is the maximization step (M-step), which uses the parameters calculated in the E-step to predict the label. The SLDA is estimated as follows:

$$E\left[Y_d \| \omega_{1:N}, \alpha, \beta_{1:K}, \eta_d, \sigma^2\right] = \eta_d^T E\left[Z_{d,i:N} \| \omega_{1:N}\right]. \quad (2)$$

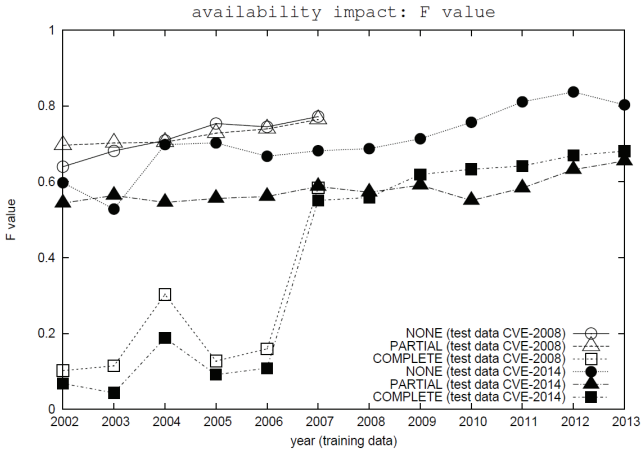

Fig. 2. Availability: Results for the relationship between the estimation performance and the temporal distance

TABLE V. PERFORMANCE OF THE BASIC SLDA AND THE ANNUAL WEIGHT-ASSIGNMENT ALGORITHMS

	Category	#	SLDA	(linear)	(sigmoid)
AV	LOCAL	115	0.649	0.663	0.637
	ADJACENT NETWORK	41	-	-	-
	NETWORK	1158	0.964	0.960	0.960
AC	HIGH	50	-	-	-
	MEDIUM	624	0.609	0.636	0.669
	LOW	640	0.731	0.769	0.765
AU	MULTIPLE INSTANCES	5	-	-	-
	SINGLE INSTANCE	239	0.622	0.303	0.506
	NONE	1070	0.940	0.914	0.931
A	NONE	499	0.783	0.860	0.855
	PARTIAL	459	0.597	0.726	0.707
	COMPLETE	356	0.632	0.694	0.695
C	NONE	480	0.781	0.794	0.798
	PARTIAL	565	0.701	0.761	0.742
	COMPLETE	269	0.562	0.653	0.634
I	NONE	420	0.708	0.753	0.750
	PARTIAL	639	0.741	0.780	0.770
	COMPLETE	255	0.571	0.634	0.622

We assumed that the estimate could be improved by assigning weights to the training dataset. A previous study [10] indicated that there might be a trend in the vulnerability information. Our key idea was to assign higher weights to more recently published CVEs.

To assess our hypothesis, we conducted a set of experiments for identifying the relationship between the accuracy of the estimate and the temporal distance. The experimental conditions are shown in Table IV, where each condition is assigned a number. For example, in Condition 1, we used CVE-2014 as the test data and CVE-2013 as the training data. We then used the SLDA to calculate the f_1 measure.

The results are shown in Figure 2, where the black shapes are used for the CVE-2014 test data (Conditions #1-12), and the white shapes are used for the CVE-2008 test data (Conditions #13-19). For clarification, we will focus our discussion on the performance of estimating the Availability. We classified the Availability of the CVEs as *NONE* (circles), *PARTIAL* (triangles), or *COMPLETE* (squares). We observed that the estimation performance was correlated with the temporal distance. The results for the other base metrics exhibited a similar trend, and hence, we considered the possibility that the estimation might be affected by the annual effect of the CVEs.

B. Annual weight assignment algorithms

This section describes our weight-assignment algorithms for the temporal distance of the CVEs, based on the experiments described in Section IV-A.

Figure 3 presents the key differences when the weights are added to the structure of the SLDA. In the SLDA, ϕ_d and η_d for document d are defined during the estimation. Our algorithms determines the parameters (ϕ_d, η_d) and the annual effect of the CVEs. In addition, our algorithm incorporates the annual parameter ω_d for (ϕ_d, η_d), and it then estimates the label.

The training data consisted of CVE-2002, $\cdots$, CVE-2013, and the test data were CVE-2014. We calculated twelve sets of the parameters (ϕ_d, η_d), and we used either a linear or a sigmoid function to assign weights for each parameter. The predicted value $Y_{predicted}$ can be calculated as

$$Y_{predicted} = \arg\max(evaluated\_value(Y_d)), \quad (3)$$

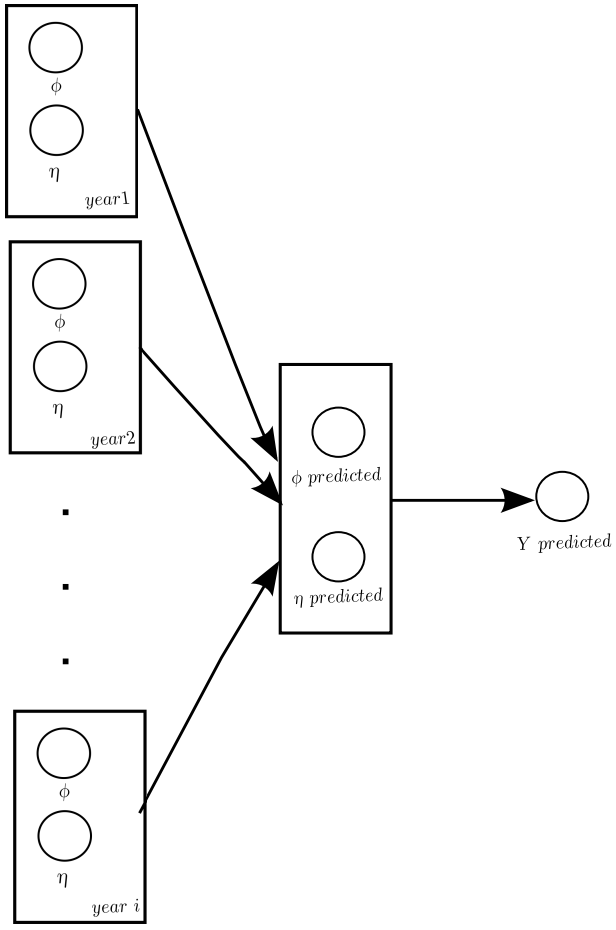

Fig. 3. Assignments of weights for the SLDA

where $evaluated\_value(Y_d)$, where document $d \in$ CVE-2014, can be calculated as

$$evaluated\_value(Y_d) = \sum_{k=2002}^{2013} \eta_{d,k} \cdot \phi_{d,k} \cdot \omega_k. \quad (4)$$

We calculated the annual parameter ω_d for a linear function (Equation 5) and for a sigmoid function (Equation 6):

$$\omega_k = \frac{(k - 2001)}{12} \quad (5)$$

$$\omega_k = \frac{1}{1 + e^{gain(-1+2(k-2001)/k)}} \quad (6)$$

where $k \in (2002, \cdots, 2013)$. The results are summarized in Table V, where the first column lists the base metrics, the second column lists the category, the third lists the number of test datasets classified into that category, the fourth denotes the f_1 measures for the SLDA, and the fifth and sixth columns list the performance obtained when an annual weight-assignment algorithm that is linear of sigmoid, respectively.

In many cases, our algorithms performed better than did the basic SLDA. Our algorithms gave better estimates of the Availability, Confidentiality, and Integrity. In the cases of Access Complexity and Access Vector, we observed that, in some categories, the SLDA performed better than our

TABLE VI. CWE AND CVSS BASE METRICS

	CWE id	16	20	22	59	78	79	89	94	119	134
AV	LOCAL	0.18	0.12	0.02	0.90	0.05	0.00	0.00	0.02	0.07	0.12
	ADJACENT NETWORK	0.03	0.02	0.00	0.00	0.05	0.00	0.00	0.00	0.01	0.00
	NETWORK	0.78	0.86	0.98	0.10	0.89	1.00	1.00	0.98	0.92	0.88
AC	HIGH	0.07	0.03	0.03	0.09	0.05	0.05	0.01	0.03	0.05	0.04
	MEDIUM	0.37	0.43	0.29	0.75	0.26	0.95	0.09	0.50	0.53	0.35
	LOW	0.56	0.54	0.67	0.16	0.69	0.00	0.90	0.46	0.43	0.60
AU	MULTIPLE INSTANCES	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SINGLE INSTANCE	0.07	0.10	0.10	0.02	0.30	0.10	0.07	0.07	0.04	0.06
	NONE	0.93	0.90	0.90	0.98	0.70	0.90	0.93	0.93	0.96	0.94
A	NONE	0.40	0.20	0.48	0.15	0.01	1.00	0.00	0.04	0.01	0.01
	PARTIAL	0.33	0.43	0.42	0.36	0.18	0.00	0.99	0.59	0.34	0.58
	COMPLETE	0.27	0.36	0.10	0.49	0.82	0.00	0.00	0.36	0.65	0.41
C	NONE	0.33	0.49	0.09	0.38	0.03	0.99	0.00	0.04	0.15	0.20
	PARTIAL	0.45	0.28	0.76	0.18	0.18	0.00	0.99	0.59	0.24	0.42
	COMPLETE	0.23	0.23	0.15	0.44	0.80	0.00	0.00	0.36	0.61	0.37
I	NONE	0.33	0.45	0.41	0.06	0.03	0.00	0.00	0.02	0.15	0.16
	PARTIAL	0.47	0.33	0.49	0.45	0.18	1.00	0.99	0.62	0.24	0.47
	COMPLETE	0.21	0.22	0.10	0.50	0.79	0.00	0.00	0.36	0.61	0.37
	CWE id	189	200	255	264	287	310	352	362	399	
AV	LOCAL	0.12	0.17	0.24	0.27	0.05	0.09	0.00	0.64	0.12	
	ADJACENT NETWORK	0.02	0.01	0.03	0.02	0.03	0.47	0.00	0.02	0.02	
	NETWORK	0.86	0.81	0.73	0.71	0.92	0.44	1.00	0.34	0.86	
AC	HIGH	0.03	0.03	0.03	0.04	0.04	0.04	0.03	0.26	0.02	
	MEDIUM	0.48	0.26	0.22	0.30	0.28	0.67	0.95	0.52	0.35	
	LOW	0.49	0.71	0.75	0.66	0.69	0.30	0.02	0.22	0.63	
AU	MULTIPLE INSTANCES	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	SINGLE INSTANCE	0.04	0.09	0.10	0.20	0.08	0.03	0.06	0.03	0.06	
	NONE	0.96	0.91	0.90	0.80	0.92	0.97	0.94	0.97	0.94	
A	NONE	0.04	0.94	0.56	0.43	0.31	0.36	0.10	0.22	0.01	
	PARTIAL	0.47	0.04	0.21	0.31	0.48	0.58	0.88	0.23	0.49	
	COMPLETE	0.49	0.02	0.23	0.26	0.20	0.06	0.02	0.55	0.50	
C	NONE	0.31	0.02	0.09	0.20	0.15	0.12	0.12	0.27	0.56	
	PARTIAL	0.30	0.89	0.63	0.55	0.65	0.81	0.86	0.22	0.16	
	COMPLETE	0.39	0.09	0.28	0.26	0.20	0.07	0.02	0.51	0.28	
I	NONE	0.32	0.93	0.47	0.27	0.19	0.27	0.00	0.39	0.58	
	PARTIAL	0.29	0.05	0.30	0.48	0.63	0.68	0.98	0.21	0.15	
	COMPLETE	0.39	0.01	0.23	0.25	0.19	0.05	0.02	0.40	0.27	

TABLE VII. PRELIMINARY PERFORMANCE ANALYSIS FOR THE TWO-STEP ESTIMATION METHOD

	Category	#	Two-Step
AV	LOCAL	115	0.336
	ADJACENT NETWORK	41	0.020
	NETWORK	1158	0.874
AC	HIGH	50	0.086
	MEDIUM	624	0.404
	LOW	640	0.103
AU	MULTIPLE INSTANCES	5	0.000
	SINGLE INSTANCE	239	0.449
	NONE	1070	0.924
A	NONE	499	0.618
	PARTIAL	459	0.459
	COMPLETE	356	0.495
C	NONE	480	0.523
	PARTIAL	565	0.601
	COMPLETE	269	0.523
I	NONE	420	0.591
	PARTIAL	639	0.602
	COMPLETE	255	0.474

algorithms, and it did so in all categories in the case of Authentication. However, of the 18 categories, in 13 cases, our algorithm with the linear function performed better than the SLDA, and in 12 cases, this was true of our algorithm with the sigmoid function. Hence, we believe that our algorithms can improve the estimation performance.

V. DISCUSSION

A. Towards an accurate estimation

In this study, we have searched for a method to more accurately predict the CVSS metrics. As shown in Section IV, we employed a weight assignment based on the year of the CVEs. This section explains some potential approaches to improving the accuracy.

Besides the CVSS and CVE, Common Weakness Enumeration (CWE) [15] hints at ways to assess the impact of the vulnerability. Table VI shows the relationship between the CWE identifier and the CVSS base metrics; the number denotes the percentile of the distribution. Consider a vulnerability categorized as CWE-ID = 89 (SQL Injection). In almost all cases, the base metrics could be guesstimated accurately: Attack Vector is *NETWORK*, Access Complexity is *LOW*, Authentication is *NONE*, and the Availability, Confidentiality, and Integrity impacts are *PARTIAL*, *PARTIAL*, and *PARTIAL*, respectively. The cases of CWE-ID = 79 (Cross-Site Scripting) and CWE-ID = 352 (Cross-Site Request Forgeries) are similar.

From the above observation, we considered that if a CWE-ID were assigned to the vulnerability information, it might be possible to improve the accuracy. We also speculated that it might be possible to create a two-step method that estimates the CWE-ID from the description and then use this to estimate the CVSS base metrics. The resulting f_1 measures are shown in Table VII. Compared to the SLDA, we observed

an improved performance for the CVE documents for which the Availability was *HIGH* and the Access Complexity was *ADJACENT NETWORK*. Unfortunately, the SLDA performed better than the two-step method in the other cases. As an area of future work, we want to find a way to predict a particular CWE.

B. Behind the annual effect of the CVE

This section considers the background of the assumption that there was a correlation between estimation performance and temporal distance.

According to a report by the SANS institute [16], the serious cyberthreats reported in the first decade of the 2000s targeted operating systems and their default installed services. For example, CVE-2001-0500 described the vulnerability in a Web service for Windows that allows remote attackers to penetrate, i.e., the CodeRed worms [17]. CVE-2002-0649 describes the Slammer worm [18], which exploits Microsoft's SQL services; CVE-2003-0352 describes the Blaster worm [19], which exploits Microsoft's RPC services; and CVE-2003-0533 describes the Sasser worm [20], which exploits the Active Directory service; all of these targets are part of Microsoft Windows operating systems. In addition, CVE-2002-0392 describes the Scalper worm, which exploits an Apache web service; CVE-2001-0011 describes the Lion worm, which exploits a BIND DNS service; and CVE-2002-1337 describes a vulnerability in the Sendmail email services. The modus operandi for all of these attacks was a buffer overflow, in which the data stored in the stack area of the computer are overwritten. These data are not only read/write-able, but also executable. If these attacks are successful, the attacker can run an arbitrary program on the host computer. In order to mitigate the risks of a buffer overflow, source code validation tools [21], [22], compilers [23], [24], and CPU supports, such as eXecute Disable/No eXecute, have been developed.

Instead of operating systems and their default installed services, web browsers and applications became the primary target during the middle of the first decade of the 21st century. In 2007, the most critical vulnerability in a client computer was ActiveX, the browser extension of Microsoft's Internet Explorer. According to a report by AV-test [25], attackers tended to target Java, Adobe Flash, and PDF, all of which were available as browser extensions. The modus operandi for these attacks was heap spraying, rather than buffer overflows. In addition to client computers, host computers also suffered attacks, primarily SQL injection, script injection (cross-site scripting), and OS command injection.

We concluded that the threats vary by decade. In order to better estimate the performance for CVE-2014, it can be assumed that it is better to train using CVE-2013 than to train using CVE-2002. Therefore, an annual weighting might be feasible in terms of both cybersecurity and statistics.

VI. Conclusion

This paper attempted to use CVE documents to estimate the impact of vulnerability information. Our motivation is that an automated assessment of the impacts would facilitate vulnerability management by helping notify security operators of severe threats and helping them prioritize their response. We postulated that an automated risk assessment based on vulnerability documents written in natural language could be of great assistance to these operators.

Our analysis was based on machine learning techniques, and we found that there was a correlation between the estimation performance and temporal distance of the CVE documents. That is, when estimating the impact of a CVE reported in 2014, knowledge of a CVE reported in 2013 is likely to be more meaningful than a CVE reported in 2002. In short, there was an annual effect both from statistical and cybersecurity points of view.

We therefore proposed an algorithm to assign an annual weight within the SLDA algorithm. In this paper, we introduced both linear and sigmoid functions for the weight assignment, and we showed that these could improve the estimation performance. In 13 of the 18 categories, our algorithm with the linear function performed better than the SLDA, and in 12 cases, our algorithm with the sigmoid function gave the better performance.

The remaining issue is to estimate the impact of a completely novel vulnerability. Since attackers also develop new methodologies to exploit software, this risk cannot be ignored. For such new attacks, our analysis, which is based on machine learning, would not work efficiently; it may be necessary to include the expert knowledge of experienced security operators. Our methodologies will mitigate the effects of cyberthreats by reinforcing operators' decisions and providing better estimations of the vulnerabilities.

Acknowledgments

This research was supported by the Strategic International Collaborative R&D Promotion Project of the Ministry of Internal Affairs and Communication, Japan, and by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 608533 (NECOMA). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the Ministry of Internal Affairs and Communications, Japan, or of the European Commission.

References

[1] D. A. Ramos and D. Engler, "Under-Constrained Symbolic Execution: Correctness Checking for Real Code," in *Proceedings of the 24th USENIX Security Symposium*, August 2015, pp. 49–63.

[2] MITRE Corporation, "Common Vulnerability and Exposure," Available at: https://cve.mitre.org.

[3] P. Mell, K. Scarfone, and S. Romanosky, "A Complete Guide to the Common Vulnerability Scoring System Version 2.0," Available at: https://www.first.org/cvss/cvss-guide, June 2007.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, no. 5, pp. 993–1022, 2003.

[5] S. Deerwester, S. T. Dumais, and R. Harshman, "Indexing by Latent Semantic Analysis," *Journal of The American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[6] D. M. Blei and J. D. McAuliffe, "Supervised topic models," in *Proceedings of the 21st Annual Conference on Neural Information Processing System*, December 2007, pp. 121–128.

[7] CNA, "CVE Numbering Authorities," Available at: https://cve.mitre.org/cve/cna.html.

[8] The Department of Homeland Security, "National Infrastructure Advisory Council," Available at: http://www.dhs.gov/national-infrastructure-advisory-counci.

[9] FIRST.org, “Forum of Incident Response and Security Teams,” Available at: https://www.first.org.

[10] S. Neuhaus and T. Zimmermann, “Security Trend Analysis with CVE Topic Models,” in *Proceedings of the 21st International Symposium on Software Reliability Engineering (ISSRE)*, November 2010, pp. 111–120.

[11] National Institute of Standards and Technology, “National Vulnerabilty Database,” Available at: https://nvd.nist.gov.

[12] MITRE, “The MITRE Corporation,” Available at: http://www.mitre.org.

[13] M. F. Porter, “An Algorithm for Suffix Stripping,” *Program*, vol. 14, pp. 130–137, 1980.

[14] M. Chaput, “stemming,” Available at: https://pypi.python.org/pypi/stemming.

[15] MITRE Corporation, “Common Weakness Enumeration,” Available at: https://cwe.mitre.org.

[16] SANS Institute, “SANS Information Security Training,” Available at: http://www.sans.org.

[17] Microsoft, “Unchecked Buffer in Index Server ISAPI Extension Could Enable Web Server Compromise (MS01-033),” Available at: https://technet.microsoft.com/library/security/ms01-033, June 2001.

[18] ——, “Elevation of Privilege in SQL Server Web Tasks (MS02-061),” Available at: https://technet.microsoft.com/library/security/ms02-061, October 2002.

[19] ——, “Buffer Overrun In RPC Interface Could Allow Code Execution (MS03-026),” Available at: http://www.microsoft.com/technet/security/bulletin/MS03-026.asp, July 2003.

[20] ——, “Security Update for Microsoft Windows (MS04-011),” Available at: https://technet.microsoft.com/library/security/ms04-011, April 2004.

[21] N. Nethercote and J. Seward, “Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation,” in *Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation*, June 2007, pp. 89–100.

[22] D. Evans and D. Larochelle, “Improving Security Using Extensible Lightweight Static Analysis,” *IEEE Software*, vol. 19, no. 1, pp. 42–51, 2002.

[23] P. Wagle and C. Cowan, “StackGuard: Simple Stack Smash Protection for GCC,” in *Proceedings of the GCC Developers Summit*, May 2003, pp. 243–255.

[24] H. Etoh and K. Yoda, “ProPolice: Improved Stack-smashing Attack Detection,” *IPSJ Journal*, vol. 43, no. 12, pp. 4034–4041, 2002, (in Japanese).

[25] AVTEST, “Adobe & Java Make Windows Insecure,” Available at: http://www.av-test.org/en/news/news-single-view/adobe-java-make-windows-insecure, December 2013.