Vulnerability Discovery Models for Database Management System Using CVE

DU SUN YOON, TAE-SUNG KIM

***Abstract*—**Software vulnerabilities that can be the target of malicious attack have a direct impact on software security and require quantitative analysis for effective management. The vulnerability standards that identify, classify, and evaluate the vulnerabilities can be a metric for quantitative analysis. The Vulnerability Discovery Model (VDM) helps to predict the vulnerability incidence rate and the number of vulnerabilities in the future. In this study, we describe the vulnerability management systems used by major countries and the vulnerability standards of the United States that are in general use. For empirical study, we select five DBMS vulnerabilities and examine quantitative analysis that applies the VDM.

***Keywords*—**Vulnerabilities, Vulnerability Discovery Model, CVE, CVSS, NVD

I. INTRODUCTION

THE software used for national defense, finance, industry, and the day-to-day life of the average person has increased in importance with the development of IT technology. Even the national infrastructure systems, such as water and electricity supply systems and, traffic and communication systems, are now controlled through software. The use of software improves the efficiency of the systems and makes possible the systematic control of resources.

The internet can now be easily accessed using smart phones and tablet PCs, with no regard to time and place. As a result, the social community is expanding due to information sharing.

However, the complexity of software has also been increasing in order to provide these services and systems. This has caused defects which can affect the accuracy and safety of the software. The software defects that give rise to security threats can be vulnerabilities, which are faults that can be viciously used to harm the security of a software system [1]. Through these vulnerabilities, threats and attacks can affect the security of critical infrastructure, industrial control systems, and Supervisory Control and Data Acquisition (SCADA) control systems [2].

Any system that uses vulnerable software can be attacked. Therefore, the security of software depends on how the vulnerabilities are managed. Although the qualities of existing vulnerability management strategies have been intensively studied quantitative analysis vulnerability is still needed for efficient management. Quantitative methods of achieving target levels of security make it possible to allocate resources. These methods also allow for the numerical evaluation of allocating resources for security testing and scheduling the development of security patches and released patches. These quantitative methods are also used by end- users to analyze risk and estimate potential vulnerabilities [3].

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0025512). This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2013S1A5A2A01017485). This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the "Employment Contract based Master's Degree Program for Information Security" supervised by the KISA (Korea Internet Security Agency) (H2101-13-1001).

Du Sun Yoon is with the Department of Management Information Systems, Chungbuk National University, 12 Gaeshin-dong, Heungduk-gu, Cheongju, Chungbuk 361-763, South Korea (corresponding author. phone: +82-43-283-3658; fax: +82-43-273-2355; e-mail: tongtong1211@naver.com).

Tae-Sung Kim is with the Department of Management Information Systems (Big Data Service Model Optimization Team, BK21 Plus), Chungbuk National University, 12 Gaeshin-dong, Heungduk-gu, Cheongju, Chungbuk 361-763, South Korea (corresponding author.. phone: +82-43-261-3343; fax: +82-43-273-2355; e-mail: kimts@cbnu.ac.kr).

II. LITERATURE REVIEW

A. Software vulnerability definitions

The definitions of vulnerability vary by researchers. Krsul (1998) defines vulnerability as "an instance of [a mistake] in the specification, development, or configuration of software such that its execution can violate the [explicit or implicit] security policy [4]." Pfleeger (1997) defines vulnerability as a "weakness in the security system that might be exploited to cause loss or harm [5]." Mehrez and Henda (2006) defines vulnerability as a "defect, or a bug, or a flaw [6]."

Judging by the above definitions, the vulnerability of software can be defined as "software defects can give security threats."

B. Software vulnerability lifecycle

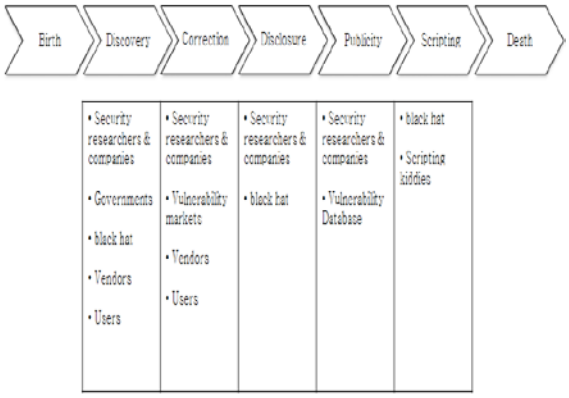

Fig. 1 Software vulnerability lifecycle

A vulnerability has a lifecycle that occurs through the seven stages shown figure 1 and that progresses by the actions of research institutions and stakeholders associated with each stage [7].

First, in the birth stage, the vulnerability is made during the software development process. It is not corrected and discovered at this stage because the software has not yet been distributed. The Second stage is discovery, which occurs between the stages of birth time it is discovered by external stakeholders.

Stakeholders corresponding to this stage include the vendors who distribute the software, attackers who try to threaten the system, and the relevant researchers and security agencies. The third stage is correction, which is the period of time between when the vendors that analyzed the software's vulnerability are able to develop a patch, and its publication. The fourth stage is disclosure, which is intended to expose the vulnerability found during the discovery phase. The type of disclosure varies depending upon how much is disclosed and how disclosure is made. The fifth stage is publicity. There is difference between disclosure and publicity, which differs from disclosure in that disclosure involves the exposure of vulnerability information among stakeholders for the patch, while publicity involves announcement of the presence of vulnerability to general users and corporations for the purpose of vulnerability warnings. The sixth stage is scripting, where hackers develop programs and scripts for automating that will allow extensive exploitation of the vulnerability. A worm is a typical example of a script.

The final stage is death and represents the time when the patch for the vulnerability has been completed or when interest in or attacks on the vulnerability are reduced.

However, not all vulnerabilities progress through all seven stages. Depending on the discoverer and the purpose of the countermeasures, the vulnerability lifecycle can be changed, especially during the discovery and disclosure stages.

The vulnerability lifecycle can be divided into two broad categories: discovery by hackers or discovery by others. If the vulnerability is discovered by hackers, it is exploited or disclosed to an inside community and a black market can be formed. In the black market, knowledge about the vulnerabilities is traded with other hackers or organizations. In contrast if the vulnerability is discovered by others (e.g., security researchers, security companies, and users), a white market formed, where vendors who have developed the vulnerable software provide rewards for reporting vulnerabilities in order to develop quick patchs.

C. Software vulnerability database

In the United States, the National Vulnerability Database (NVD) is the responsibility of the Department of Homeland Security (DHS), the National Institute of Standards and Technology (NIST), and the MITRE Corporation. Figure 2 shows the vulnerabilities collecting system of NVD. The vulnerability data constituting the NVD is stored using identification, classification, and evaluation standards. We will describe standards of NVD in detail in the following section.

In the case of Japan, the Information-technology Promotion Agency (IPA) and Japan Computer Emergency Response Team

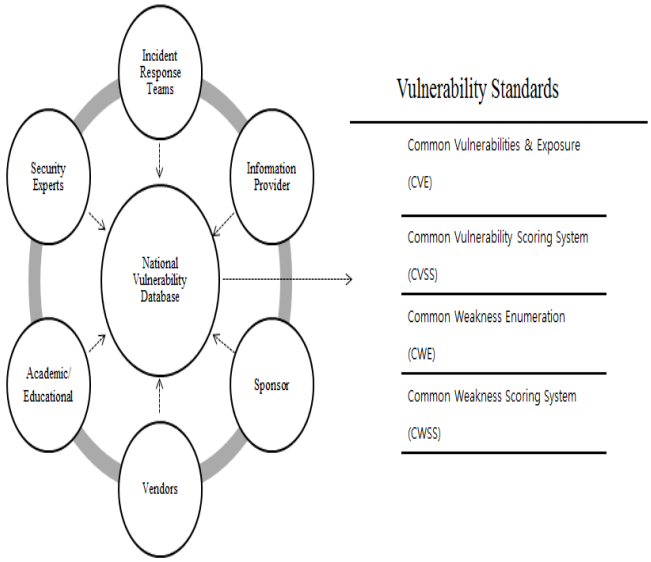

Fig. 2 Vulnerabilities collecting system of NVD

Coordination Center (JPCERT/CC) operates the vulnerability databases. These agencies have set up an Information Security Early Warning Partnership for efficient collection and patching of vulnerabilities. They collect the vulnerabilities through a cooperation system similar to that used in the United States and involving vendors, software developers, security researchers, security organizations, and other stakeholders as shown in figure 3 [8]. The agencies also collect vulnerabilities from international cooperating institutions such as NIST and the Centre for the Protection of National Infrastructure (CPNI) in the UK. The Collected vulnerabilities are disclosed on Japan Vulnerability Notes (JVN) and JVN iPedia. The difference in the two databases lies in their correspondence systems. JVN discloses the vulnerabilities immediately using its own vulnerability classification system and assessment system, whereas JVN iPedia uses the identification, classification, and evaluation standards of United States. JVN iPedia also discloses vulnerabilities with countermeasures by conducting a survey and vulnerability analysis every 2 to 4 weeks.

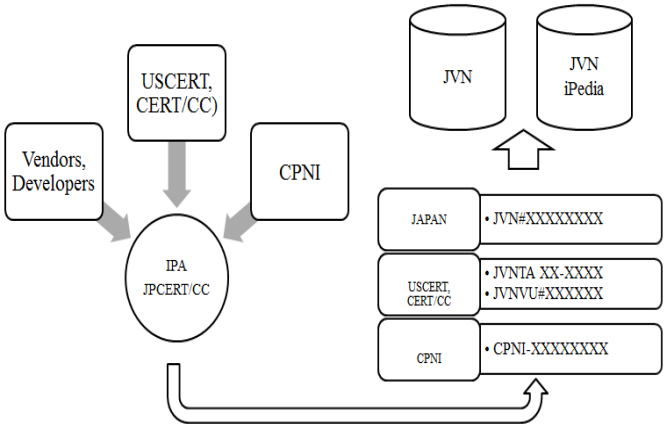

Fig. 3 Vulnerabilities collecting system of JVN and JVN iPedia

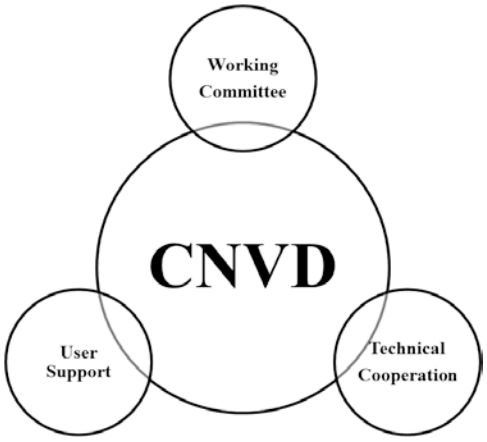

Fig. 4 CNVD cooperating system

China operates a vulnerability database under the name of the China National Vulnerability Database (CNVD). As shown in figure 4, CNVD is operated with a technical cooperation system composed of the China Computer Emergency Response Team Coordination Center (CNCERT/CC), software developers, more than 200 networks security companies, and white hackers. Similar to the United States and Japan, they collect vulnerability data through cooperation with external and internal organizations. Registration of vulnerabilities on CNVD is divided into informal registration and public registration. Informal registration uses CNVD-ID, which is its own identification system. CNVD-ID consists of the year and a 5 digit serial number [9].

D. Software vulnerability standards

1) Common Vulnerability and Exposures (CVE)

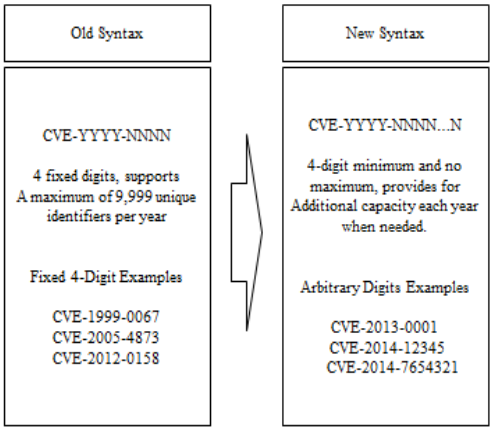

Fig. 5 Revised CVE-ID

Vulnerability data will be collected by the coordination system, which is made up of a variety of relevant institutions and professionals, including security professionals, software developers, and computer emergency response teams (CERTs). It is possible for confusion to occur due to the use of different names in each institution for each collected vulnerability; therefore, it will be arranged through the CVE classification system. Organized vulnerabilities are listed on the NVD with the year and serial number by a compilation committee [10].

CVE is a useful tool for the quantitative analysis of software-security vulnerabilities. It is possible to analyze the number of vulnerabilities according to calendar time. In addition, classification by type of vulnerability is also possible. More than 58,258 CVEs of 1,919 softwares have been listed on the NVD to date [11].

Figure 5 shows the information of the revised CVE-ID applied on January 1, 2014. The serial numbers of the original CVE-ID were restricted to 4 digits. Therefore, the CVE numbers that could be published were limited to 9,999. The revised CVE-ID expanded the serial numbers to 7 digits, so publication of 9,999,999 CVEs is now possible [12].

2) Common Vulnerability Scoring System (CVSS)

CVSS is a framework for scoring the severity of vulnerabilities. Thus, it helps to remove vulnerabilities according to their priority. The CVSS scoring system is made up of a base metric, a temporal metric, and an environmental metric. The base metric, which is evaluated considering the inherent characteristics of the vulnerability, is made up of sub-scales that evaluate the difficulty and influence of attack on the vulnerability. The temporal metric, which is a measuring element that can be affected by time, has sub-scales for evaluating the possibility of an attack and the difficulty of patching the vulnerability. The environmental metric measures the elements affected by the environment, such as additional impacts of an attack and the target distribution [13].

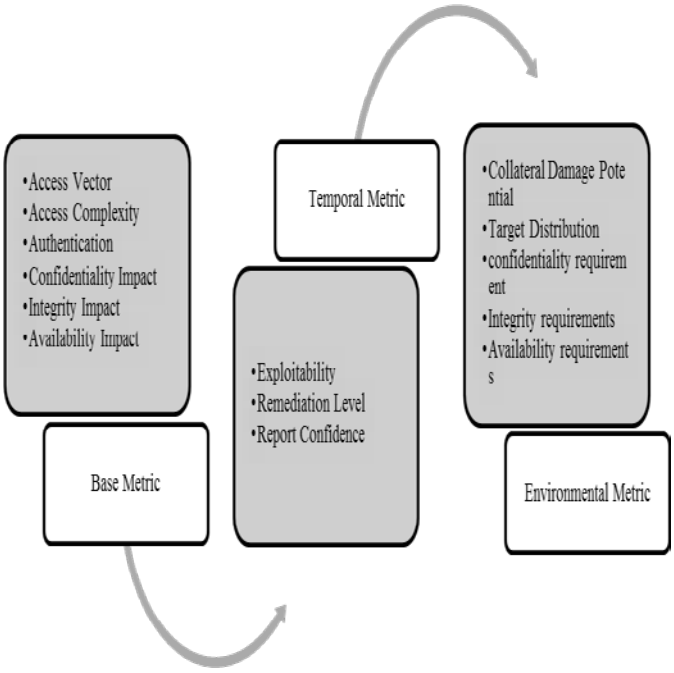

Fig. 6 CVSS scoring system

The values of the three metrics are obtained by separate equations, and the values to be substituted into the equations are obtained by each sub-scale. As Shown in figure 6, the final values of vulnerability severity are obtained by successive calculations of the three metrics. The basic metric value is included in the equation of the time metric, while the time metric value is included in the equation of the environmental metric. The final score of the vulnerability has a value between 0.0 and 10. The CVSS version has been updated to CVSS v2.9 from CVSS v1.0 to date, and CVSS v3.0 is currently under development [14]. A version is updated by revision of the equations and sub-scales.

3) Common Weakness Enumeration (CWE)

CWE is a classification standard of vulnerabilities. Whereas CVE describes the vulnerability of specific software, CWE provides information about weaknesses that might commonly occur in software. Accordingly, it is intended to improve the security and services program for diagnosing vulnerability by providing information to security experts and developers regarding the type of vulnerabilities. The CWE classification system consists of Views, Categories, and Compound Elements including Weaknesses. Views classify the weaknesses depending on the perspective and the concept, while categories classify the weaknesses with common characteristics and Compound Elements describes the weaknesses of complex elements rather than one weakness. As of November 2013, 940 CWE entries consisting of 31 Views, 187 Categories, 741 Weaknesses, and 8 Compound Elements were published [15].

4) Common Weakness Scoring System (CWSS)

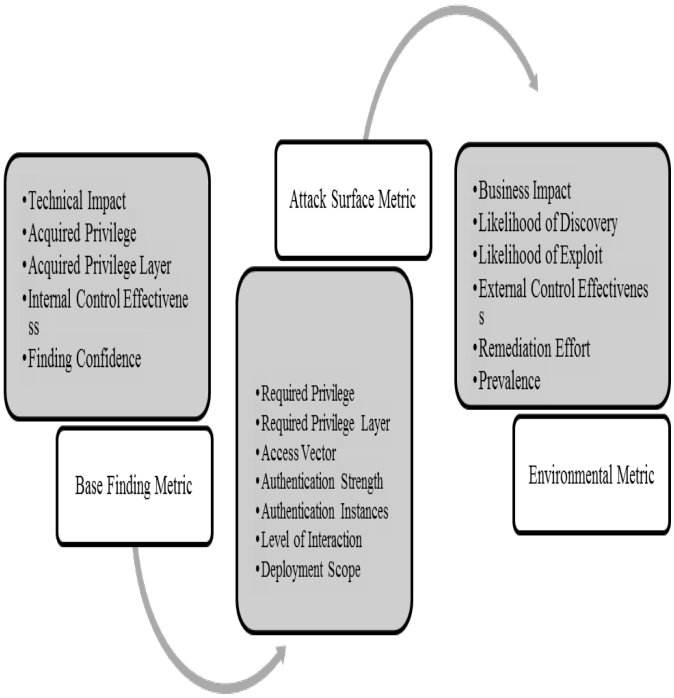

Fig. 7 CWSS scoring system

CWSS is a framework for scoring the severity of CWE. CVS S evaluates the CVE with three metrics, and CWSS also evalua tes the CWE with three metrics as shown in figure 7. The first metric is the Base Finding Metric Group, which consists of sub -scales such as the impact of weakness and authority level obtai ned by an attack on a weakness. The second metric is the Attac k Surface Metric Group, which is composed of sub-scales inclu ding scoring authority level and possible locations to attack we akness. The last metric is the environmental metric, which has sub-scales to score the difficulty of finding weaknesses, potenti al of attack, and the business impacts of an attack. Similar to th e CVSS scoring system, the CWSS severity score is obtained b y successive calculation of three metrics, but the final score has a value between 0.0 and 100, while the CVSS has a value bet ween 0.0 and 10. The version of CWSS has been updated CWS S v0.1 to CWSS v0.8 at present [16].

E. VDM

VDM is a useful tool for the quantitative analysis of software security vulnerabilities. It was started on the basis of the Software Reliability Model (SRM). The SRM assumes that the reliability of the program is based on the number of errors the program has. Depending on the detection and removal of errors, system's errors may be reduced enough to make the system more reliable.

The SRM is used to predict the number of errors remaining in the system and when they are to be generated. This prediction may be used to measure the amount of reliability tests required [17]. Thus, the SRM uses statistical methods to detect errors during testing and operation to predict the reliability of the products [18]. Applying the SRM to vulnerability data has not been done for years. Alhazmi and Malaiya (2005) proposed the term VDM applying the SRM to vulnerability data [19].

We can predict the cumulative number and the detection rate of vulnerability through VDM. This also makes it possible to measure the time and resources necessary to maintain the system, to estimate the required time for quality assurance, and to compare similar systems.

VDM can be classified into two models. The first, the Time-Based Model, is used to predict the cumulative number of vulnerabilities over time. Time-Based Models have been studied as follows:The Anderson Thermodynamic Model (AT) proposed by Anderson [20], the Rescorla Quadratic Model (RQ) and the Rescorla Exponential Model (RE) proposed by Rescorla [21], the Logarithmic Poisson Model (LP) proposed by Musa and Okumoto [22], and the Alhazmi and Malaiya Logistic model (AML) proposed by Alhazmi and Malaiya [23]. The second, the Effort-Based Model is used to predict the cumulative number of vulnerabilities based on the number of users and market share. Alhazmi and Malaiya proposed Alhazmi and Malaiya Effort-Based model (AME) [25].

In this paper, we did not examine the application of the Effort-Based Model because it would have been difficult to collect the objective data which would have included the number of product users, and the market share.

III. Empirical Study

A. VDM Models for DBMS

In this paper, the vulnerability data of the five DBMSs (ORACLE DATABASE SERVER, MYSQL, MS-SQL SERVER, POSTGRE-SQL, and DB2) were collected up to May 1, 2013 from the NVD. These DBMSs were ranked from first place to fifth places according http://www.db-engines.com/ on May 1, 2013 [24].

Alhazmi and Malaiya applied all the existing VDMs to targeting major Operating Systems. Then, in order to measure the difference between the observed value and the actual model, we performed the chi-square goodness of fit test. The results showed that AML is the most significant in many Operating Systems [25]. Therefore, in this paper, we applied the AML to the collected vulnerability data. In addition, for comparison, we applied the Linear Model (LM), estimated by linear regression analysis. Then, through the chi-square goodness of fit test, we tested the models in how close they were to the actual observations. The data used in this test was comprised of quarterly accumulated vulnerabilities.

AML is based on an S-shaped behavior that can be divided into three phases. The first is the learning phase. In this phase, hackers are interested in newly released software. They learn about the software and start reporting vulnerabilities. The second phase is the linear phase. In this phase, hackers understand the software and market acceptance of software gets increased. Thus, reporting of the software's vulnerabilities rises linearly. The third is the saturation phase. In this phase, simple vulnerabilities have been found. In addition new versions of software are released so that hacker's is drawn to them. Finally, the number of cumulative vulnerabilities decreases.

$$\frac{d\Omega}{dt} = A\Omega(B-\Omega) \tag{1}$$

$$\Omega(t) = \frac{B}{BCe^{-ABt}+1} \tag{2}$$

Equation (2) is AML. It is obtained by solving a differential equation of (1). Equation (1) is composed of two factors (A, B). A is the increasing rate of vulnerabilities and B is the total number of accumulated vulnerabilities that will eventually be found.

Ω is the cumulated number of vulnerabilities. Where C is a constant introduced while solving (1), t = 0 initially, and A, B are empirically determined from the recorded data [23].

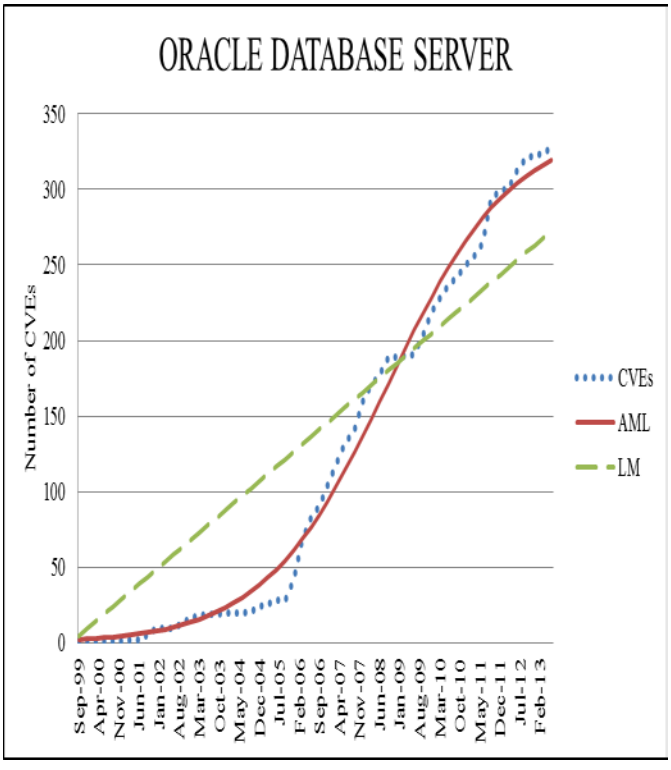

Fig. 8 ORACLE DATABASE SERVER fitted to the models

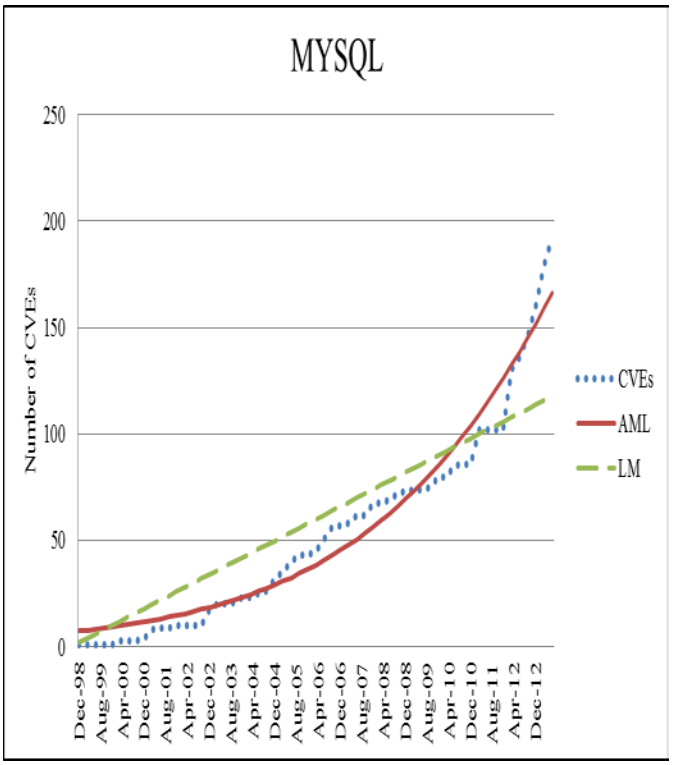

Fig. 9 MYSQL fitted to the models

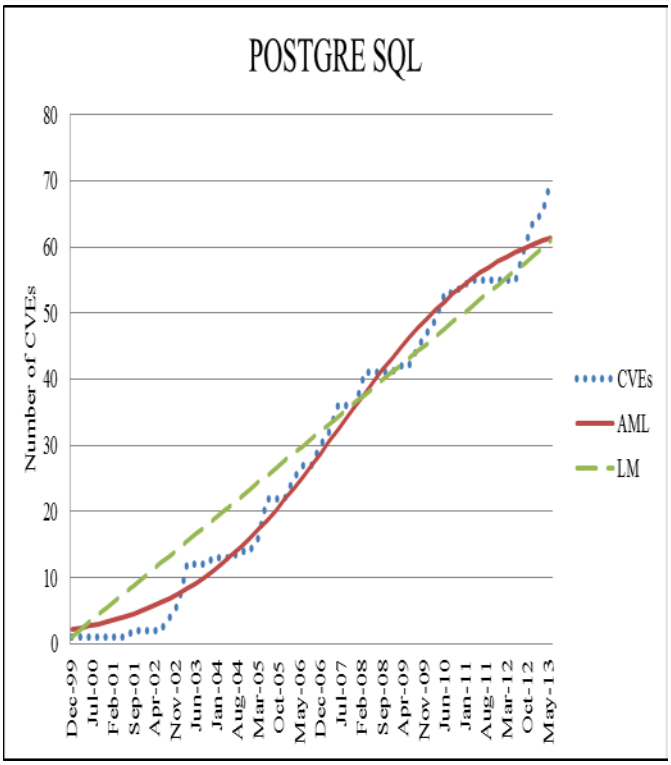

Fig. 11 MS-SQL SERVER fitted to the models

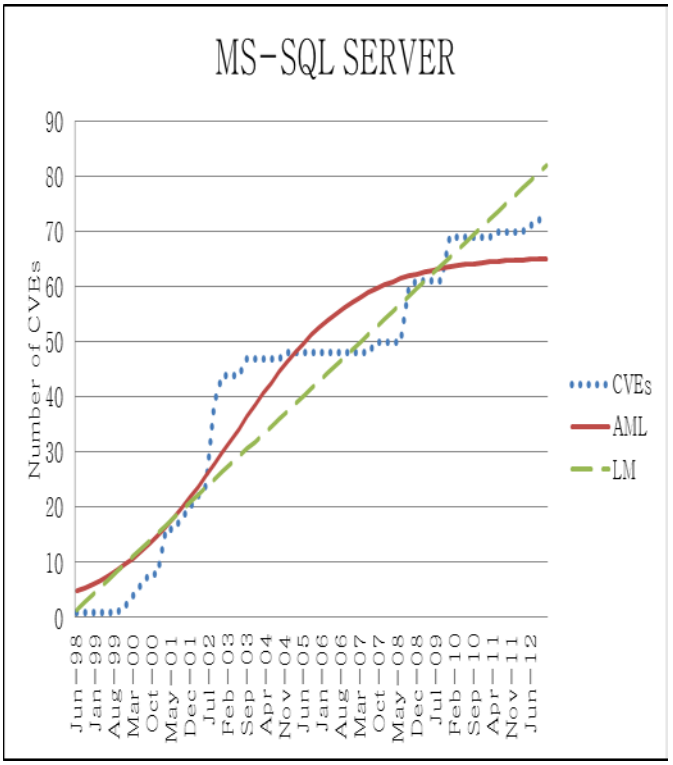

Fig. 10 MS-SQL SERVER fitted to the models

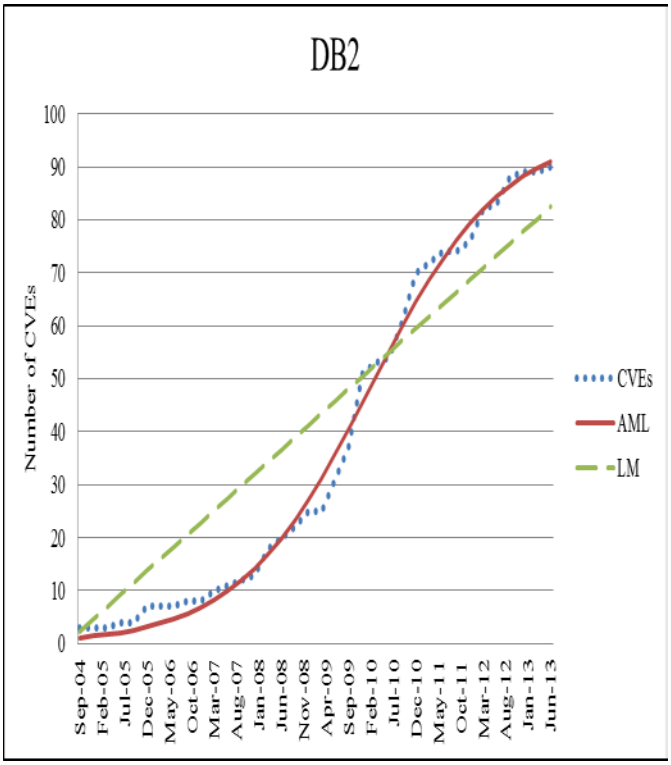

Fig. 12 DB2 fitted to the models

Figures 8, 9, 10, 11, and 12 are graphs that were obtained by applying the VDM to vulnerabilities of MS-SQL SERVER and DB2 respectively. The dotted line shows the actual quarterly

accumulated vulnerabilities. The solid and dashed lines are models estimated by applying the AML and LM to the data.
Figure 8 shows a graph of the ORACLE DATABASE SERVER's 328 vulnerabilities from the third quarter, 1999 to the second quarter, 2013. Figure 9 shows a graph of the MYSQL's 193 vulnerabilities from the firth quarter, 1998 to the second quarter, 2013. Figure 10 shows a graph of the MS-SQL SERVER's 73 vulnerabilities from the second quarter, 1998 to the fourth quarter, 2012. Figure 11 shows a graph of the POSTGRE-SQL's 70 vulnerabilities from the firth quarter, 1999 to the second quarter, 2013. Figure 12 shows the DB2's 90 vulnerabilities from the third quarter, 2004 to the second quarter, 2013.

Figure 9 shows the exponential behavior. Figure 10 shows the S-shaped behavior of AML, but the vulnerabilities increased rapidly in 2000, 2002, 2009, and 2010 instead of steadily. However, in Figure 8, Figure 11, and Figure 12, these can be seen that the models and the actual data are almost identical.

B. Chi-square Goodness of fit test

Table 1. AML Chi-square Goodness of fit test

	DF	Chi-square	Chi-square critical (5%)	P-value	Result
ORACLE DATABASE SERVER	55	96.58969177	85.749	0.00045217	N/S
MYSQL	58	120.6242366	89.477	0.00000272	N/S
MS-SQL SERVER	58	95.46324599	89.477	0.00141685	N/S
POSTGRE-SQL	54	28.66034354	84.502	0.99819694	S
DB2	35	22.76729623	60.275	0.94479405	S

Table 2. LM Chi-square Goodness of fit test

	DF	Chi-square	Chi-square critical (5%)	P-value	Result
ORACLE DATABASE SERVER	55	1288.424185	85.749	2.2931E-233	N/S
MYSQL	58	272.7755515	89.477	1.43185E-29	N/S
MS-SQL SERVER	58	118.5453287	89.477	4.7932E-06	N/S
POSTGRE-SQL	54	88.21977088	84.502	0.002264418	N/S
DB2	35	144.0919039	60.275	3.46329E-15	N/S

We used the chi-square goodness of fit test to determine whether or not the expected values obtained by AML fit the observed values. It was performed with a significance level at 5%. When the obtained chi-square value was below the chi-square critical value, the model fit the data [26]. However, if the obtained chi-square value was higher than the chi-square critical value, the model did not fit the data.

Table 1 shows the results of the ORACLE DATABASE SERVER, MYSQL, and MS-SQL SERVER were insignificant, but the result of the POSTGRE-SQL and DB2 were significant. On the other hand, Table 2 shows the results of performing the chi-square goodness of fit test by applying LM showed that not all of the models were significant.

IV. CONCLUSIONS

In this study, we examine the vulnerability management systems of major countries and vulnerability standards of the United States that are utilized in general. Also we used VDM analysis as a quantitative method of analyzing software security vulnerabilities. We collected five DBMS vulnerabilities from the NVD for empirical research. The collected data on vulnerabilities was accumulated quarterly vulnerabilities and shown in the graphs. Since then AML and LM were applied to the data and compared to each other through the chi-square goodness of fit test. The results indicated that AML is more significant than LM.
The models of DB2 and POSTGRE-SQL products were shown to be significant through the application of the AML. It was also shown that they have a variety of uses. By predicting the cumulative number over the model, we can make a purchase decision of DBMS. In addition, the development team of the vendors can use the models for manpower allocation and patch schedules.

REFERENCES

[1] Chunguan, K., Qing, M., Hua, C., Analysis of Software Vulnerability, in *Proc. Of the 5th WSEAS International Conference on Information Security and Privacy, 2006*, pp.218-223.
[2] Giovanni, C., Taihoon, K., Seoksoo, K., Improving SCADA Control Systems Security with Software Vulnerability Analysis, in *Proc. Of the 12th WSEAS International Conference on Automatic Control, Modelling & Simulation, 2010*, pp. 409-414.
[3] Alhazmi, O. H., Malaiya, Y. K., Ray, I., Measuring, Analyzing and Predicting Security Vulnerabilities in Software Systems, *Computers & Security*, Vol.26, No.3, 2007, pp. 219–228.
[4] Krsul, I. V., Software Vulnerability Analysis, Ph.D. Dissertation, Computer Sciences Department, Purdue University, 1998.
[5] Pfleeger, C. P., Security in Computing, Prentice-Hall, 1997.
[6] Mehrez, E., Henda, B. G., Addressing Software Application Security Issues, in *Proc. Of the 10th WSEAS International Conference on Computers, 2006*, pp. 362-367.
[7] John, T. C., Vulnerability Disclosure Framework : Final Report and Recommendations by the council, National Infrastructure Advisory Council, 2004.
[8] Dongjin, k., Sungje, C., An analysis of Domestic Foreign Security Vulnerability Management Systems based on a National Vulnerability Database, *Internet and Information Security*, Korea, 2010, pp.130-147.

[9] A study on Construction of a Vulnerability Management System for New Information Technologies, Industry Academic Cooperation Foundation of Dankook University, 2010.
[10] http://cve.mitre.org/about/index.html, accessed on Oct 1, 2013.
[11] http://www.cvedetails.com, accessed on Nov 2, 2013.
[12] http://cve.mitre.org/cve/identifiers/syntaxchange.html, accessed on Oct 1, 2013.
[13] http://www.first.org/cvss/cvss-guide, accessed on Oct 1, 2013.
[14] http://www.first.org/cvss/history, accessed on Oct 1, 2013.
[15] http://cwe.mitre.org/data/lists/2000.html, accessed on Nov 2, 2013.
[16] http://cwe.mitre.org/cwss/index.html, accessed on Oct 1, 2013.
[17] Ozment, A., Vulnerability Discovery and Software Security, unpublished Ph.D. Dissertation, Computer Laboratory Computer Security Group & Magdalene College, University of Cambridge, 2007.
[18] AIAA/ANSI, 1993. Recommended Practice: Software Reliability. ANSI.ISBN 1-56347-024-1. R-013-1992, pp. 26-27, 37 and 67.
[19] Alhazmi, O. H., Malaiya, Y. K., Modeling the vulnerability discovery process, in *Proc. Of the 16th IEEE International Symposium on Software Reliability Engineering*, 2005. pp. 129-138.
[20] Anderson R. J., Security in open versus closed systems - the dance of Boltzmann, Coase and Moore, int. *Conf. on Open Source Software: Economics, Law and Policy*, 2002, pp. 1-15.
[21] Rescola E., Is finding security holes a good idea? , *Security and Privacy*, Vol.3, No.1, 2005, pp. 1 -19.
[22] Musa J. D., Okumoto K., A logarithmic Poisson execution time model for software reliability measurement, in *Proc. 7th International Conference on Software Engineering*, 1984, pp. 230-238.
[23] Alhazmi, O. H., Malaiya, Y. K., Quantitative vulnerability assessment of systems software, In *Proc. of the IEEE Reliability and Maintainability Symposium (RAMS'05)*, 2005, pp. 615.
[24] http://db-engines.com/en/ranking, accessed on May 1, 2013.
[25] Alhazmi, O. H., Malaiya, Y. K., Application of vulnerability discovery models to Major Operating Systems, in *Proc. Of the IEEE transactions on reliability*, 2008, pp. 14-22.
[26] Shahabuddin,, F. A. A., Kamarulzaman, I., Abdull, A. J., On The Comparison of Several Goodness of Fit Tests:With Application to Wind Speed Data, in Proc. 3rd WSEAS Int. Conf. on Renewable Energy Sources, 2009, pp. 394-398.

Du Sun Yoon is a master course student at the Department of Management Information Systems at Chungbuk National University. He received his bachelor in International Business from Chungbuk National University. His research areas include vulnerability management and security in software.

Tae-Sung Kim is a professor at the Department of Management Information Systems at Chungbuk National University. He received his bachelor, master, and doctoral degrees in Management Science from Korea Advanced Institute of Science and Technology (KAIST). He worked for Electronics and Telecommunications Research Institute (ETRI) as a Senior Researcher for more than three years. Also, he worked as a visiting professor at the Department of Business Information System and Operation Management, the University of North Carolina at Charlotte and a visiting research scholar at the School of Computing, Informatics and Decision Systems Engineering, Arizona State University. His research areas include management and policy issues in telecommunications and information security. His recent research papers have appeared in international journals, such as European Journal of Operational Research, ETRI Journal, Journal of the Operations Research Society, Journal of Intelligent Manufacturing, Operations Research Letters, and Stochastic Analysis and Applications.