Vulnerability Discovery Models for Database Management System

DU SUN YOON, TAE-SUNG KIM*, DUK SU KIM**
*Department of Information Security Management
Chungbuk National University
52 Naesudong-ro, Heungdeok-gu, Cheongju, Chungbuk
**Penta Security Systems Inc.
Hanjin Shipping Building, 20th Floor 25-11 Yoido-Dong, Youngdeungpo-Gu, Seoul
SOUTH KOREA
nicky1985@cbnu.ac.kr, kimts@chungbuk.ac.kr, dskim@pentasecurity.com

Abstract: The quantitative analysis of software security vulnerabilities has a direct effect on software quality management and development. Quantitative analysis method of vulnerabilities is various. In particular, Vulnerability Discovery Model (VDM) has been used successfully. VDM shows the current state of vulnerabilities. Also it helps to predict the vulnerability incidence rate and the number of vulnerability in the future. In this work, we select five DBMSs and try to examine quantitative analysis applying the VDM.

Key-Words: Vulnerabilities, Vulnerability Discovery Model (VDM)

1 Introduction
The quality of software depends on how vulnerabilities are managed. Existing vulnerability management has been intensively studied qualities. However, it is necessary to quantitatively analyze the vulnerability 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 [1].

2 Literature Review

2.1 Software vulnerability definitions and Common Vulnerability and Exposures (CVE)
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 [2].”

Pfleeger (1997) defines vulnerability as a “weakness in the security system that might be exploited to cause loss or harm [3].”

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

The National Vulnerability Database (NVD), which operates at the center of the National Institute of Standards and Technology (NIST) of the MITRE Corporation has standardized vulnerability as the above and integrated management. 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 [4].

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 57,500 CVEs of 1,907 softwares have been listed on the NVD to date [5].
2.2 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 [6]. Thus, the SRM uses statistical methods to detect errors during testing and operation to predict the reliability of the product [7]. 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 [8].

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 [9], the Rescorla Quadratic Model (RQ) and the Rescorla Exponential Model (RE) proposed by Rescorla [10], the Logarithmic Poisson Model (LP) proposed by Musa and Okumoto [11], and the Alhazmi and Malaiya Logistic model (AML) proposed by Alhazmi and Malaiya [12].
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) [12].

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.

3 Empirical Study

3.1 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 [13].

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 [14]. 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, 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.

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

(1)

$$\Omega (t) = \frac{B}{Bce^{-At} - \Omega} + 1$$

(2)

AML’s equation is (2). 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. \(\Omega\) is the cumulated number of vulnerabilities. Where \(C\) is a constant introduced while solving equation (1), \(t = 0\)
initially, and \( A, B \) are empirically determined from the recorded data [1].

\[
\begin{align*}
\text{Fig. 1 MS-SQL SERVER fitted to the models} \\
\end{align*}
\]

\[
\begin{align*}
\text{Fig. 2 DB2 fitted to the models} \\
\end{align*}
\]

Figures 1 and 2 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 1 shows a graph of the MS-SQL SERVER’s 73 vulnerabilities from the second quarter, 1998 to the fourth quarter, 2012. Figure 2 shows the DB2’s 90 vulnerabilities from the third quarter, 2004 to the second quarter, 2013.

Figure 1 shows the S-shaped behavior of AML, but the vulnerabilities increased rapidly in 2000, 2002, 2009, and 2010 instead of steadily. However, in Figure 2, it can be seen that the models and the actual data are almost identical.

3.2 Goodness of fit for two models
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. However, if the obtained chi-square value was higher than the chi-square critical value, the model did not fit the data.

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, the results of performing the chi-square goodness of fit test by applying LM showed that not all of the models were significant.

4 Conclusions
In this study, 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.

Acknowledgement
This research was supported by the MSIP (Ministry of Science, ICT and 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). 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 (NRF-2011-0025512).

References: