IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 1, 2019, accepted March 23, 2019, date of publication March 28, 2019, date of current version April 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907977

E-WBM: An Effort-Based Vulnerability Discovery Model

XIAJING WANG, RUI MA, BINBIN LI, DONGHAI TIAN, AND XUEFEI WANG
School of Computer Science and Technology, Beijing Institute of Technology, Beijing 10081, China

Corresponding author: Rui Ma (mary@bit.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016QY07X1404 and in part by the National Natural Science Foundation of China under Grant 61602035.

ABSTRACT Vulnerability discovery models (VDMs) have recently been proposed to estimate the cumulative number of vulnerabilities that will be disclosed after software is released. A precise VDM would offer an available quantitative insight to assess software security. Even though VDM has demonstrated its effectiveness in multiple software, it remains limited in accuracy, especially with weak versatility. We propose a novel effort-based VDMs, named E-WBM, to improve critical vulnerability discovery rate algorithm using Weibull probability distribution function towards efficient vulnerability discovery models. E-WBM accurately portrays the trend of software security vulnerabilities disclosure. We evaluate E-WBM on eight popular real-world operating systems and show the feasibility of the proposed model. We further compare E-WBM with a state-of-the-art effort-based model AME and time-based model JW on the above eight operating systems. Our comparison also demonstrates that E-WBM consistently outperforms AME and JW both at reducing the deviations and fitting curve trends. In addition to the model fitting, predictive capabilities of two effort-based models E-WBM and AME are also examined. The results show that the E-WBM model yields a more stable prediction with a significantly less error than AME.

INDEX TERMS Vulnerability discovery model, JW, E-WBM, AME, testing effort.

I. INTRODUCTION

Software vulnerability is a major issue that deserves attention throughout the software lifecycle, which can be broadly divided into two categories: non-security-related and security-related. The latter will lead to more serious harm than the former. The Organization for Internet Safety defines security-related vulnerabilities as [1]. A special type of software defects that may cause the software to be inconsistent with its design goals and may violate the security policies defined in software documentation.

For general non-security-related defects, several quantitative methods that use static metrics or software reliability growth models (SRGMs) are available to ensure the safety of software. Vulnerability discovery models (VDMs) can be regarded as a specialization of SRGMs since vulnerabilities are a special class of defects or bugs that can permit circumvention of the security measures [2]. Such models are dedicated to predicting its trends of security-related vulnerabilities disclosure by analyzing the number of eliminated security vulnerabilities in the software. The purpose of a VDM is not to identify vulnerabilities but to predict the number of security vulnerabilities that are likely to be discovered in the future. Apparently, accurate VDMs should not only depict real security vulnerability discovery trends, but also adapt to the dynamic changes of software security vulnerabilities.

Most popular VDMs, such as Joh Weibull (JW) model [3], use the calendar time as an independent variable to predict software vulnerabilities. These time-based models are relatively mature, but lack consideration of external environmental changes. Recently, several models, such as Alhazmi & Malaiya Effort (AME) model [1], have tried to use effort as the main factor. These effort-based models manage to overcome the impact of external environmental changes on vulnerability discovery capabilities, and have higher accuracy for predicting vulnerability counts and vulnerability discovery trends than time-based models. Unfortunately, the current effort-based models still fail to accurately predict the security vulnerabilities for every software.

In this paper, we introduce E-WBM, a novel effort-based vulnerability discovery model [4]. Unlike the prior

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Imran Tariq.

effort-based model AME, we combine Weibull probability distribution function to improve the efficiency of AME. Specifically, E-WBM is inspired by the vulnerability discovery rate in JW model obeying Weibull probability distribution function. Particularly, Weibull distribution can describe varying patterns such as the ''*S*-shaped'' [3], which is consistent with the characteristics of the software lifecycle. To this end, we consider improve the vulnerability discovery rate algorithm of original AME with the Weibull probability distribution function to establish a new effort-based model. Our extensive evaluation of 8 real-world programs covering 3 different operating systems, (e.g., Windows, Linux, and Mac OS) shows that E-WBM generally outperforms AME and JW.

Our primary contributions in this paper are as follows.

- Presentation of model: we present E-WBM, a novel effort-based VDM model that uses installed base as reference factor to forecast vulnerability discovery trends.
- Enhancement of vulnerability discovery rate algorithm: we enhance vulnerability discovery rate algorithm of AME by employing Weibull probability distribution function.
- Evaluation of model: we evaluate E-WBM by applying it to 8 operating system and comparing it with AME and JW on the above 8 operating systems. The experimental results demonstrate that the accuracy of our model exceeds AME and JW as a whole.
- Prediction of model: we examine the predictive capability of E-WBM by comparing it with AME. Results show that the prediction error of proposed model less than AME in most situations.

The rest of the paper is organized as follows. Section II and Section III introduce related work and background, respectively. Section IV depicts an overview of our methodology. We present our experimental results in Section V and analyze E-WBM's limitations in Section VI. Section VII concludes this paper.

II. RELATED WORK

Existing VDMs could be classified into two categories: time-based and effort-based. The former measures the cumulative counts of vulnerabilities found in the course of time, while the latter counts vulnerabilities regard to the testing effort.

A. TIME-BASED MODELS

Most VDMs proposed employ the time as an independent variable due to its intuitiveness and simplicity. Anderson [5] proposes Anderson Thermodynamic (AT) model, which is the first real VDM, but it exhibits poor data fit for most software datasets. Alhazmi and Malaiya [6] present Alhazmi & Malaiya Logistic (AML) model, which is an *S*-shaped model with three stages of ''learning phase – linear phase – saturation phase''. The AML model uses a logistic vulnerability discovery process which assumes an approximate symmetric shape around the peak discovery rate value. Although the AML model generally performs well, it has potential limitation for software that has not yet reached saturation stage. Rescorla Quadratic (RQ) model and Rescorla Exponential (RE) model [7], respectively, assume a linear relationship and an exponential relationship between vulnerability discovery rate and time using statistical tests. However, RQ and RE models merely appear to do a good job of following the shorter-term trends. Joh *et al*. [3] introduce Weibull model (JW), a new Weibull distribution [8]–[13] based VDM, which can be used to model datasets whose vulnerability discovery rate appears asymmetric. The results show that the JW model performs well in many cases, and may be considered as an alternative to the AML model. Recently, Chen *et al*. [14] present a multi-cycle vulnerability discovery model. Anand and Bhatt [15] propose a hump-shaped model and judge its performance using a weighted criteria based ranking approach. In addition, a few authors, such as Kim *et al*. [16] and Anand *et al*. [17], [18], have considered the discovery pattern in a multi-version software.

All of these time-based models use time intervals as the main factor to report vulnerabilities. The reason behind this is that it is easy to record vulnerabilities and link them to the time of discovery. This however does not consider the changes occurring in the environment during the lifetime of the software. A major environmental factor is the number of installations which depends on the share of installed base for specific software [1]. To alleviate the above limitations, it is much more rewarding to research effort-based vulnerability discovery models.

B. EFFORT-BASED MODELS

Effort can be any variables that capture environmental changes during the lifetime of software, referring to the number of installations, the number of test cases, or the CPU time occupied by software. However, mature effort-based models that have been proposed are still rare. Using effort as a factor was first discussed by Brocklehurst and Littlewood [19], [20], but they have not suggested a unit or a way to measure effort. Alhazmi *et al*. [21] present a reference formula for testing effort. Subsequently, Alhazmi and Malaiya propose AME model [1], which is the decent goodness of fit for the vulnerability discovery rate of major operating systems. This model takes into account the impact of external environment on vulnerability discovery, such as the number of installations. The experimental results confirm that effort-based models outperform time-based in accuracy. Nevertheless, a potential limitation of AME model is that it does not fully fit several operating systems, such as Windows 95 operating system. Moreover, Li HF, Wan YX *et al*. [22]–[26] further study effort-based models, and make good suggestions on how to better evaluate and apply effort-based models. Lately, a few authors introduce multivariate or multiattribute methods to model VDMs. For example, Johnston *et al*. [27]–[29] introduce multivariate methods to modeling vulnerability discoveries in web-browser software and present new techniques that apply structured expert-judgment data-gathering

methods. Kansal *et al*. [30] present a coverage-based VDM using multiattribute approach, which performs better and has the good fitness for browser data. However, these methods focus primarily on optimizing the disclosure time or discovery process of web-browser vulnerabilities.

C. VULNERABILITY DATASET

Note that the above VDMs could get different trends by considering different vulnerability dataset (such as NVD, MITRE, Bugzilla, BugTraq, *etc*.). The National Vulnerability Database (NVD) [31] is a comprehensive and detailed platform for obtaining vulnerability data. Additionally, since the NVD project is sponsored by the Department of Homeland Security, like the CVE project, it can be regarded to be a standard source [32]. To this end, this paper employs NVD as a unified dataset to evaluate different VDMs.

III. BACKGROUND

We briefly introduce several typical time-based VDMs and effort-based VDMs in Section III (A) and Section III (B), respectively. Section III (C) discusses the merits and limitations of these models.

A. TIME-BASED MODELS

Researchers [33], [34] have summarized existing ten fundamental time-based VDMs: Linear (LN) model [35], Logarithmic Poisson (LP) model [36], Anderson Thermodynamic (AT) model [5], Alhazmi & Malaiya Logistic (AML) model [6], Rescorla Quadratic (RQ) model [7], Rescorla Exponential (RE) model [7], Joh Weibull (JW) model [3], Multi-cycle vulnerability discovery model [14], Multi-version (MVD) model [16], and Folded model (YF) [33].

Among the available time-based VDMs, AML model has a better goodness of fit for most software, which has been used extensively to the improvement of subsequent research works. JW model with better performance is improved on the basis of AML model. Hence, this paper primarily introduces AML model and JW model.

1) AML MODEL

Alhazmi-Malaiya Logistic (AML) model was proposed by Alhazmi and Malaiya [6]. This model is based on the observation that the attention given to an operating system increases after its release, peaks at some time and then drops because of the release of a newer competing version. Thus the vulnerability discovery rate increases at the beginning, reaches a steady rate and then starts declining. The cumulative number of vulnerabilities thus shows an increasing rate at the beginning as the system starts attracting an increasing share of the installed base. After some time, a steady rate of vulnerability discovery yields a linear curve. Eventually, as the vulnerability discovery rate starts dropping, there is saturation due both to reduced attention and a smaller pool of remaining vulnerabilities.

The model assumes that the rate of change of cumulative number of vulnerabilities is governed by two factors. One of these factors declines as the number of remaining undetected vulnerabilities declines. The other factor increases with the time needed to take into account the rising share of the installed base.

The AML model has shown the descent goodness of fit for the vulnerability discovery rate of major operating systems. However, a potential limitation of this model is that it assumes a logistic vulnerability discovery rate that is symmetric around the peak value. Conceivably, in some situations, the behavior may not be symmetric, thus AML model does not fit these situations well.

2) JW MODEL

Joh *et al*. present JW model [3], a vulnerability discovery model using Weibull distribution that is suited to describe the vulnerability discovery rate. Further, since Weibull distribution can model an asymmetric probability distribution function, the JW model addresses the fact that AML model cannot detect vulnerability discovery rate which exhibits asymmetric on both sides of the peak value.

This model assumes that the vulnerability discovery rate $\omega(t)$ varied according to the Weibull probability distribution function as given in Eq. (1). The cumulative number of vulnerability $\Omega(t)$ is the integral of the vulnerability discovery rate with respect to the time, as shown in Eq. (2).

$$\omega(t) = \gamma \left\{ \frac{k}{\lambda} \left(\frac{t}{\lambda} \right)^{k-1} e^{-\left(\frac{t}{\lambda}\right)^k} \right\} \tag{1}$$

$$\Omega(t) = \gamma \left\{ 1 - e^{-\left(\frac{t}{\lambda}\right)^k} \right\} \tag{2}$$

where k is the shape parameter and λ is the scale parameter, which can decide the duration of VDM. The γ value refers to the maximum number of vulnerabilities.

The Weibull-based model JW is flexible enough to capture vulnerability behavior and fit the data well in most cases [3], [37]. In particular, this model tends to perform better than other time-based models which are symmetrical (*e.g*. AML model) for the same dataset. Unfortunately, like other time-based models, it also has a general limitation that does not consider the effect of environmental factors.

B. EFFORT-BASED MODELS

Alhazmi and Malaiya also propose AME model [1], which is the most representative effort-based model. They present a new measure termed *Equivalent Effort* (E) to represent testing effort, which considers the possible environmental changes during the lifecycle of the software system. The *Equivalent Effort* (E) is calculated by using Eq. (3).

$$E = \sum_{i=0}^{n} (U_i \times P_i) \tag{3}$$

where the parameter U_i is the total number of installed base of all software at the period of time i, n represents the last usage period, and P_i is the percentage of the installed base using this measuring software. The E refers to the cumulative counts of installed base using this software at the entire time period.

AME model would assume that the vulnerability discovery rate with respect to effort is proportional to the fraction of

TABLE 1. Summary of three VDMS.

Models	Merits	Demerits
AML	(I) Follow the *S*-shaped trend. (II) Good fitting performance for mature software.	(I) Only consider the case that vulnerability discovery rate is symmetric around the peak value. (II) Does not consider the impact of external environment.
JW	(I) Follow the *S*-shaped trend. (II) Good fitting performance in most cases. (III) Consider the case that vulnerability discovery rate asymmetric around the peak value.	Does not consider the impact of external environment.
AME	(I) Consider the impact of external environment. (II) Use testing effort as main factor.	(I) The fitting effect is modest for some software. (II) It is difficult to collect the effort data.

remaining vulnerability. Then an effort-based vulnerability discovery model, just like the exponential SRGMs, can be generated as Eq. (4).

$$\Omega(E) = B\left\{1 - e^{-\lambda_{uv}E}\right\} \tag{4}$$

where λ_{uv} is a parameter analogous to failure intensity in SRGMs. The parameter B represents the number of vulnerabilities that would eventually be found and E is testing effort. The $\Omega(E)$ refers to the cumulative number of vulnerabilities found when testing effort E regards as reference factors.

The AME model that employs effort as the main factor tends to be more accurate than time alone. Also, effort-based model becomes popular especially since AME model has shown its effectiveness. However, the fitting effect of AME is unsatisfactory for some software. Besides, this model requires the number of installed base for target products in market share, which may be difficult to obtain.

C. AN ANALYSIS OF THE VDMS

Table 1 summarizes their merits and demerits of the above three typical VDMs.

We analyze the above VDMs from 2 dimensions: model performance and reference factors.

a) Model performance. The fitting performance of JW model outperforms the other two VDMs.
b) Reference factors. Only the AME model employs testing effort as main factor, which considers the changes occurring in the environment during the lifecycle of the software. Hence, this model is more reasonable than time-based models such as AML and JW model.

Since time-based models merely use time as main reference factor, researchers employ testing effort to consider more environmental factors. However, existing effort-based models are not perfect including AME with decent performance. The AME model has a bit more deviations for several datasets, mainly because it directly adopts the same vulnerability discovery rate as the SRGMs, thus does not manage to clearly distinguish between non-security-related vulnerabilities and security-related vulnerabilities. The potential difference between the two types vulnerability discovery processes can lead to instability of the AME model. Once large gaps remain in the two types of vulnerability discovery rate curves, AME model may not fit well.

To address the limitations of effort-based AME model, this paper present E-WBM, a new effort-based vulnerability discovery model that combines Weibull probability distribution function. In particular, we are inspired by the vulnerability discovery rate of JW model obeying Weibull probability distribution function. To this end, this paper considers to employ Weibull probability distribution function to improve crucial vulnerability discovery rate algorithm of AME model.

IV. METHODOLOGY

In this section, we first clarify the related terminology and Weibull probability distribution function. Then, we describe our methodology in detail and formalize the modeling process of proposed model.

A. TERMINOLOGY

- *A testing effort* is the reference factor of VDMs for specific software throughout the entire lifecycle, usually using installed base as factor. It can be expressed as symbol E.
- *The maximum number of vulnerabilities* is the upper limit of security vulnerabilities that can be found by specific software, which can be denoted as symbol γ_{max}.
- *Vulnerability discovery rate* refers to the rate at which a certain software discovers vulnerabilities in terms of the testing effort. It can be denoted as symbol $\omega(E)$.
- *The cumulative number of vulnerabilities* is the total number of security vulnerabilities discovered by a specific software as the effort increases, which can be expressed as symbol $\Omega(E)$.

B. WEIBULL PROBABILITY DISTRIBUTION FUNCTION

Weibull distribution is often used for reliability evaluation for mechanical systems. It was originally proposed by Dr. Waloddi Weibull in 1937 and used for the study of machine life and structural strength analysis. In recent years, however, Weibull distribution has been widely used in lifetime distributions in reliability engineering [32], [38]. Considering that the Weibull model can account for different increasing and decreasing trends, we believe these trends can reflect the initial increases and eventually decreases during vulnerability discovery. Particularly, Weibull distribution includes the following 3 merits.

- Weibull is a continuous distribution with the probability density function, and its cumulative distribution curve follows the *S*-shaped trend, which is consistent with the software lifecycle trends.

- Weibull model has better goodness of fit in comparison to other reliability models for vulnerability occurrence across a wide range of software systems [37].
- The collection of vulnerability data is much easier than AME model.

The probability density function of Weibull distribution is shown in Eq. (5). F, the cumulative distribution function for the Weibull distribution, is as Eq. (6).

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k} & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (5)$$

$$F(x;\lambda,k) = 1 - e^{-\left(\frac{x}{\lambda}\right)^k} \quad (6)$$

where x is the random variables, k is the shape parameter and $k > 0$, and λ is the scale parameter of the distribution and should meet the criterion $\lambda > 0$. The Weibull distribution is related to several other probability distributions; in particular, it covers the exponential distribution ($k = 1$) and the Rayleigh distribution ($k = 2$).

In addition, when $k < 3$, the Eq. (5) shows a primarily increasing trend, which can be applied to the early stage of vulnerability discovery, while this equation can be used for the late stage when $k > 3$. When $k = 3$, the Eq. (5) is close to the maximum value, which is suitable for the middle and late stage of vulnerability discovery. Thus, we believe that the Weibull distribution should be able to depict the vulnerability discovery trends well.

C. E-WBM MODEL

Based on the above analysis of the Weibull distribution, we hypothesize that improve random variable x in the Weibull distribution with the effort reference could achieve a more precise fitting effect. Thus, we combine Weibull distribution with effort-based model AME to enhance the vulnerability discovery capability. In this section, we elaborate the modeling process of our model, including testing effort algorithm, vulnerability discovery rate algorithm, and cumulative number of vulnerability algorithm.

1) TESTING EFFORT

The testing effort algorithm introduced in the E-WBM model quotes *Equivalent Effort* (E) proposed by Alhazmi and Malaiya, as shown in Eq. (3).

Here we use a month as a unit of time and thus E would be in *users-months*. Using available data, testing effort E can be calculated for specific software.

2) VULNERABILITY DISCOVERY RATE

Here we introduce the vulnerability discovery rate of proposed model based Weibull probability density function, which can be denoted as symbol $\omega(E)$. It uses α_0, β_0 to substitute the k, λ of Weibull function, respectively, and selects E as the main factor to establish the vulnerability discovery rate algorithm of E-WBM as given in Eq. (7).

$$\omega(E) = \gamma_{max}\left\{\frac{\alpha_0}{\beta_0}\left(\frac{E}{\beta_0}\right)^{\alpha_0-1} e^{-\left(\frac{E}{\beta_0}\right)^{\alpha_0}}\right\} \quad (7)$$

where α_0 is the shape parameter which can determine the shape of the vulnerability discovery rate. When $0 < \alpha_0 < 3$, the curve has positive skewness, meaning that the vulnerability discovery rate increases, which corresponds to the early stage of VDM. When α_0 is approximately 3, the shape is symmetrical, which corresponds to the middle and later stages of VDM. The number of vulnerability found increases slowly and gradually approaches the maximum value. The curve has negative skewness when $\alpha_0 > 3$, representing a decrease in the vulnerability discovery rate, which corresponds to the late stage of VDM. The parameter β_0 is the scale parameter and $\beta_0 > 0$, which can stretch the working duration of software vulnerability discovery modeling. The γ_{max} refers to the maximum cumulative number of vulnerabilities that would eventually be encountered. The E is the cumulative testing effort over the entire period.

3) CUMULATIVE NUMBER OF VULNERABILITIES

The vulnerability discovery rate $\omega(E)$ is integrated with respect to testing effort E, and the result is as given in Eq. (8), which is the mean cumulative number of vulnerabilities $\Omega(E)$. The parameter α_0, β_0, γ_{max}, and E have the same meaning as in Eq. (7).

$$\Omega(E) = \gamma_{max}\left\{1 - e^{-\left(\frac{E}{\beta_0}\right)^{\alpha_0}}\right\} \quad (8)$$

As the testing effort E continues to accumulate, the cumulative number of vulnerabilities found gradually increases, then gradually stabilize, and infinitely approaches the maximum number of vulnerabilities γ_{max}. This process perfectly presents the classic S-shaped curve, thus the proposed model could be feasible.

$$\begin{cases} E = \sum_{i=0}^{n}(U_i \times P_i) \\ \omega(E) = \gamma_{max}\left\{\frac{\alpha_0}{\beta_0}\left(\frac{E}{\beta_0}\right)^{\alpha_0-1} e^{-\left(\frac{E}{\beta_0}\right)^{\alpha_0}}\right\} \\ \Omega(E) = \gamma_{max}\left\{1 - e^{-\left(\frac{E}{\beta_0}\right)^{\alpha_0}}\right\} \end{cases} \quad (9)$$

The E-WBM model can be established by combining the above three algorithms. The simultaneous equations of (3), (7), and (8) is as shown in Eq. (9), which is the E-WBM model.

V. EVALUATION

In this section, we discuss our experimental settings and how to assess proposed model. We then evaluate the effectiveness of E-WBM compared to AME and JW models regarding goodness of fit and the number of real vulnerabilities found. Next, we examine the predictive capability of two effort-based E-WBM and AME model.

A. EXPERIMENTAL SETUP

This paper primarily selects 8 typical operating system software to evaluate E-WBM due to their comprehensive and detailed datasets. All our fitting and simulation were performed on a data analysis tool STATA.

1) PARAMETERS

To perform the fitting and prediction experiment, we first need to obtain the parameters required by E-WBM, AME, and JW model, which include two categories: effort parameters and core parameters.

a: EFFORT PARAMETERS

As shown in Eq. (3), the testing effort parameters for E-WBM and AME model include i, E, U_i, P_i, which can be directly acquired by referring to historical statistics or simply calculating historical data. Where i uses the month as the unit of time period and E represents the cumulative number of installed base used by specific software during a certain period of time, which can be calculated by accumulating $(U_i * P_i)$ at each time point i. The parameter U_i can be obtained by referring to the research data of Alhazmi *et al.* [21] and the Internet Live Stats website [39], while the acquisition of P_i borrows the research data of Alhazmi *et al.* [21], the StatCounter Global Stats website [40], and the W3School website [41].

b: CORE PARAMETERS OF E-WBM

The core parameters of E-WBM model described in Eq. (7) include α_0, β_0, γ_{max}. Where γ_{max} can be obtained through historical data. The model parameters α_0 and β_0 need to be estimated using regression analysis to determine the best values. Here, to simplify the regression process, we utilize the statistical tool STATA to carry on linear regression, and the values of two parameters are ultimately determined by the least squares method. Specifically, the process of obtaining the core parameters of E-WBM model is as follows.

a) Linearization: Since the least squares method is primarily used to estimate the unknown parameters in the linear function, the Eq. (8) should first be linearized by taking logarithms twice to translate into a linear function.

$$\frac{1}{1-\frac{\Omega(E)}{\gamma_{max}}} = e^{\left(\frac{E}{\beta_0}\right)^{\alpha_0}} \tag{10}$$

We first perform pre-transformation on Eq. (8), and result is as shown in Eq. (10). The result of the first logarithm is shown in Eq. (11).

$$ln\left(\frac{1}{1-\frac{\Omega(E)}{\gamma_{max}}}\right) = \left(\frac{E}{\beta_0}\right)^{\alpha_0} \tag{11}$$

The second logarithm is taken for Eq. (11), and the result is shown in Eq. (12).

$$ln\left[ln\left(\frac{1}{1-\frac{\Omega(E)}{\gamma_{max}}}\right)\right] = \alpha_0\, ln\, E - \alpha_0\, ln\, \beta_0 \tag{12}$$

Compared to the linear regression equation $y = a + bx$, we can get the following Eq. (13).

$$\begin{cases} Y = ln\left[ln\left(\frac{1}{1-\frac{\Omega(E)}{\gamma_{max}}}\right)\right] \\ X = ln\, E \end{cases} \tag{13}$$

Finally, the Eq. (8) is transformed into a linear equation, as given in Eq. (14).

$$Y = \alpha_0 X - \alpha_0\, ln\, \beta_0 \tag{14}$$

b) The historical data obtained of a certain software, including effort E, the cumulative number of vulnerabilities $\Omega(E)$, and the maximum number of vulnerabilities γ_{max}, are respectively brought into the Eq. (13) to obtain a plurality of two-tuples (Y, X).

c) The linear regression is performed using STATA tool, and the values of parameters α_0 and β_0 are ultimately determined. Where regression coefficient α_0 can be acquired by the STATA tool, and the parameter β_0 is calculated according to the Eq. (15) which is derived from Eq. (14) using the least squares method.

$$\widehat{\beta_0} = e^{-\frac{\bar{Y}-\widehat{\alpha_0}\bar{X}}{\widehat{\alpha_0}}} \tag{15}$$

Here $\bar{X}$, $\bar{Y}$ respectively represent the average values of X, Y and $\widehat{\alpha_0}$, $\widehat{\beta_0}$ represent estimated values of α_0, β_0, respectively.

c: CORE PARAMETERS OF AME

The core parameters of AME model include λ_{uv}, B, which is described in Eq. (4). Where the value of B is the same as the parameter γ_{max} of E-WBM model, and the acquisition of λ_{uv} is analogous to the calculation method of core parameters α_0 and β_0 in E-WBM. By linearizing the AME model, the following Eq. (16) can be obtained.

$$Y = \lambda_{uv} X \tag{16}$$

where X and Y are calculated as shown in Eq. (17).

$$\begin{cases} Y = ln\left(\frac{1}{1-\frac{\Omega(E)}{B}}\right) \\ X = E \end{cases} \tag{17}$$

d: CORE PARAMETERS OF JW

The core parameters of JW model described in Eq. (2) include k, λ, γ. Their acquisitions are similar to the parameters α_0, β_0, γ_{max} of E-WBM model respectively. Eq. (18) can be obtained by linearizing the Eq. (2).

$$ln\left[ln\left(\frac{1}{1-\frac{\Omega(t)}{\gamma}}\right)\right] = k\, ln\, t - k\, ln\, \lambda \tag{18}$$

Then we can further obtain the following Eq. (19) to estimate the values of parameters k, λ.

$$\begin{cases} Y = ln\left[ln\left(\frac{1}{1-\frac{\Omega(t)}{\gamma}}\right)\right] \\ X = ln\, t \end{cases} \tag{19}$$

2) DATASETS

We choose 8 operating systems as target software to evaluate our model, so before starting fitting, we need to collect testing effort datasets and vulnerability datasets for these 8 operating systems.

a: EFFORT DATASETS

The effort datasets of 8 operating systems selected in this evaluation are all calculated according to Eq. (3), which is $E = \{E\_Windows\ 95, E\_Windows\ 98, E\_Windows\ XP, E\_Windows\ Vista, E\_Windows\ 7, E\_Windows\ 8, E\_Linux, E\_Mac\ OS\ X\}$. Among them, the Windows 95 and Windows 98 operating system have been released for a long time and stopped for maintenance in 2001 and 2006, respectively. These datasets thus are primarily derived from the statistical results of Alhazmi *et al*. [21]. Since the entire data of Windows XP obtained difficultly from one spot, its partly borrows the research data of Alhazmi *et al*. [21] (before 2004) and partly derived from the W3School website [41] (after 2004). Whereas other datasets are obtained from the StatCounter Global Stats website [40].

b: VULNERABILITY DATASETS

The vulnerability datasets investigated here are 8 typical operating systems, which can be expressed as $V = \{V\_Windows\ 95, V\_Windows\ 98, V\_Windows\ XP, V\_Windows\ Vista, V\_Windows\ 7, V\_Windows\ 8, V\_Linux, V\_Mac\ OS\ X\}$. These datasets are manually extracted from the publicly available NVD database managed by National Institute of Standard and Technology (NIST).

B. ASSESSMENT INDICATORS

To test the validity of the estimated model parameters, we employ Significance testing and Confidence interval. Besides, we further choose 2 statistical test methods, Goodness of fit and Chi-square test, respectively, to verify the fitting effect of proposed model. We also use 2 normalized predictability measures, Average Error (AE) and Average Bias (AB), to examine the predictability of E-WBM model [32].

1) INDICATORS FOR PARAMETERS

a: SIGNIFICANCE TESTING

The significance testing is used to judge whether the model parameters are significant under a certain significance level. We first assume an original null hypothesis H_0. Then we use F-value as test statistic, and employ P-value as test index to decide statistical significance. The P-value can be determined by the F-value. The *alpha* level is the upper limit of P-value, which means the probability of rejecting the null hypothesis H_0 when it is true. Here we take an *alpha* level of 5%; i.e., if the P-value is below 0.05 then we will reject the null hypothesis H_0, it can be further considered that the parameter is statistically significant.

b: CONFIDENCE INTERVAL

A confidence interval (CI) is a type of interval estimate in which the observed value of parameter falls under a certain probability. The confidence level is used to quantify the level of confidence that the parameter lies in the interval, which is equal to 1-*alpha*, where *alpha* is significant level. The confidence level is designated prior to examining the model. Most commonly, the 95% confidence level is used. When the estimated value of parameter lies in the confidence interval, the estimated value is close to the observed value, indicating that the estimated value can be accepted.

2) INDICATORS FOR FITTING

a: GOODNESS OF FIT

The goodness of fit describes how well a model fits a set of actual values. Measures of goodness of fit typically summarize the discrepancy between observed values and the expected values under the proposed model. That can be measured by the coefficient of determination R^2, which ranges from 0 to 1. When the value of R^2 approaches 1, it indicates that the proposed model has an outstanding goodness of fit.

b: CHI-SQUARE TEST

The chi-square test is used to determine whether there is a significant difference between the expected values and the observed values. The formula for calculating the chi-square coefficient χ^2 is as shown in Eq. (20).

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} \tag{20}$$

where o_i is an observed value of type i, e_i is an expected value of type i, and k represents the number of cells. In this experiment, o_i represents the actual cumulative number of vulnerabilities discovered, and e_i represents the cumulative number of vulnerabilities predicted by the proposed model.

The value of chi-square coefficient χ^2 approaches zero, indicating that there is almost no difference between the expected values and the observed values, so the proposed model has a high precision. Conversely, the prediction of model is inaccurate.

3) INDICATORS FOR PREDICTION

a: AVERAGE ERROR (AE)

AE, as given in Eq. (21), is a measure of how well a model predicts throughout the time period. The value of AE approaches zero, indicating that the proposed model has an outstanding predictability.

$$\mathrm{AE} = \frac{1}{n} \sum\nolimits_{t=1}^{n} \left| \frac{\Omega_t - \Omega}{\Omega} \right| \tag{21}$$

where n is the total number of data point during the prediction period, Ω is the actual number of cumulative vulnerabilities, and Ω_t is the estimated number of cumulative vulnerabilities at time t. Here we replace the time t with the Equivalent Effort E (in users in this case).

b: AVERAGE BIAS (AB)

AB indicates the general bias of the model which assesses its tendency to overestimate or underestimate, as given in

TABLE 2. Core parameters of E-WBM, AME, and JW model.

Operating Systems	E-WBM			AME		JW		
	α_0	β_0	γ_{max}	λ_{uv}	B	k	λ	γ
Windows 95	0.92592442	1500.2427	31	0.00069111	31	19.010143	14980.201	30
Windows 98	0.78075773	6042.8047	45	0.00016666	45	29.52536	15277.915	38
Windows XP	0.91203779	43149.172	800	0.00002764	800	14.17863	18937.875	800
Windows 7	0.75633526	37858.531	500	0.00002445	500	53.775345	19327.133	380
Windows Vista	1.2814624	35406.344	1000	0.00002397	1000	67.582413	18678.189	270
Windows 8	0.80715442	2485.5474	400	0.00031847	400	37.549168	20175.504	340
Linux	1.0243349	81.533646	80	0.00983672	80	19.686785	18958.566	80
Mac OS X	1.304238	4229.2451	930	0.00029889	930	30.902937	19333.301	920

TABLE 3. Parameters test results for three VDMS.

Operating System	E-WBM (α_0)			AME (λ_{uv})			JW(α_1)		
	F-value	P-value	[95% Conf. Interval]	F-value	P-value	[95% Conf. Interval]	F-value	P-value	[95% Conf. Interval]
Windows 95	1070.9004	1.766e-34	(0.8690346, 0.9828142)	375.61383	2.435e-24	(0.0006194, 0.0007628)	183.51584	5.095e-18	(16.18863, 21.83166)
Windows 98	1238.9994	6.122e-36	(0.7361599, 0.8253556)	1040.5852	3.416e-34	(0.0001563, 0.0001771)	747.30988	6.441e-31	(27.35377, 31.69695)
Windows XP	4497.0605	0	(0.8851961, 0.9388795)	3794.5508	0	(0.0000267, 0.0000285)	1127.6542	0	(13.34532, 15.01194)
Windows 7	5604.355	0	(0.7361802, 0.7764902)	8528.3652	0	(0.0000239, 0.000025)	556.20105	9.620e-35	(49.22653, 58.32416)
Windows Vista	2446.9624	4.142e-36	(1.229019, 1.333905)	960.85034	1.389e-28	(0.0000224, 0.0000255)	378.93082	2.310e-21	(60.55414, 74.61069)
Windows 8	10346.78	0	(0.7913599, 0.8229489)	2905.4863	0	(0.0003067, 0.0003302)	152.18527	4.165e-20	(31.49067, 43.60767)
Linux	286.73541	1.225e-11	(0.8960966, 1.152573)	445.45206	4.167e-13	(0.0088487, 0.0108247)	59.127865	9.233e-7	(14.25934, 25.11423)
Mac OS X	4848.001	0	(1.267066, 1.34141)	1611.0551	0	(0.0002841, 0.0003137)	374.19049	3.111e-35	(27.73266, 34.07321)

Eq. (22). The parameter n, Ω, Ω_t, and t have the same meaning as in Eq. (21).

$$\mathrm{AB} = \frac{1}{n}\sum_{t=1}^{n}\frac{\Omega_t - \Omega}{\Omega} \tag{22}$$

When the value of AB is positive, it means that the model generally trends to overestimate; while the model commonly trends to underestimate if its value of AB is negative.

C. RESULTS

To show the effect of the proposed model, we evaluated this model with 8 widely-used operating systems, namely, Windows 95, Windows 98, Windows XP, Windows Vista, Windows 7, Windows 8, Linux, and Mac OS X. We further compared E-WBM with an effort-based model AME and a time-based model JW on these 8 operating systems. Next, we examine the predictive capability of two effort-based model E-WBM and AME.

1) STEP 1: ACQUIRE PARAMETERS

The acquisition methods of effort parameters and core parameters see Section V (A).

a: EFFORT PARAMETERS

We first need to determine the appropriate timeline i, collect the installed base in each time period U_i, and calculate the percentage of installed base that uses specific operating system P_i. Then we use STATA tool to count the collected data, and further acquire effort datasets with time-referenced. We select month and millions users as the units of abscissa and ordinate, respectively.

b: CORE PARAMETERS

We put the time data, effort data and vulnerability data of each operating system into the E-WBM AME, and JW model, respectively, to determine their core parameters. The core parameters of each operating system are shown in Table 2.

2) STEP 2: PERFORM PARAMETERS CHECK ANALYSIS

To check the validity of above core parameters, we perform 2 assessment indicators described in Section V (B), Significance testing and Confidence interval, respectively. Since γ_{max}, B, γ are known parameters, and β_0, λ can be calculated by α_0, k, respectively, it is only necessary to check the estimated values of α_0, λ_{uv}, and k. The test results of three

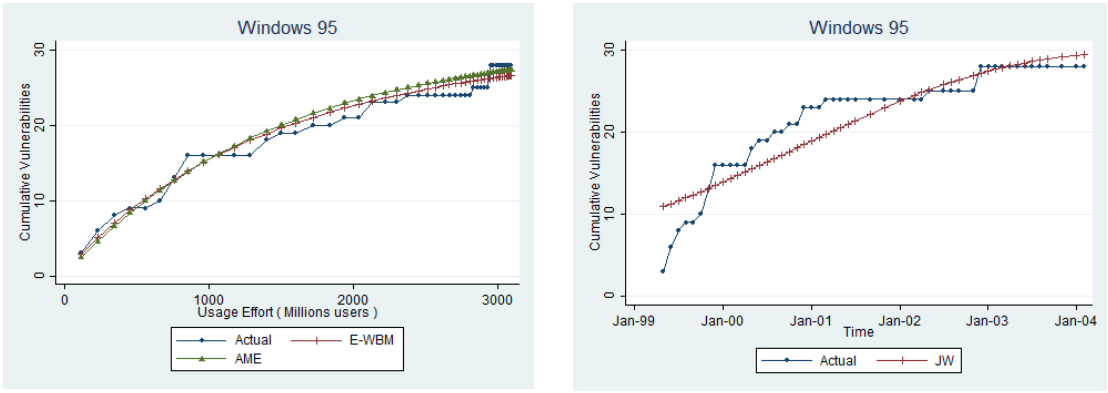


FIGURE 1. Model fitting for windows 95. (a) Effort-based model, (b) Time-based model.

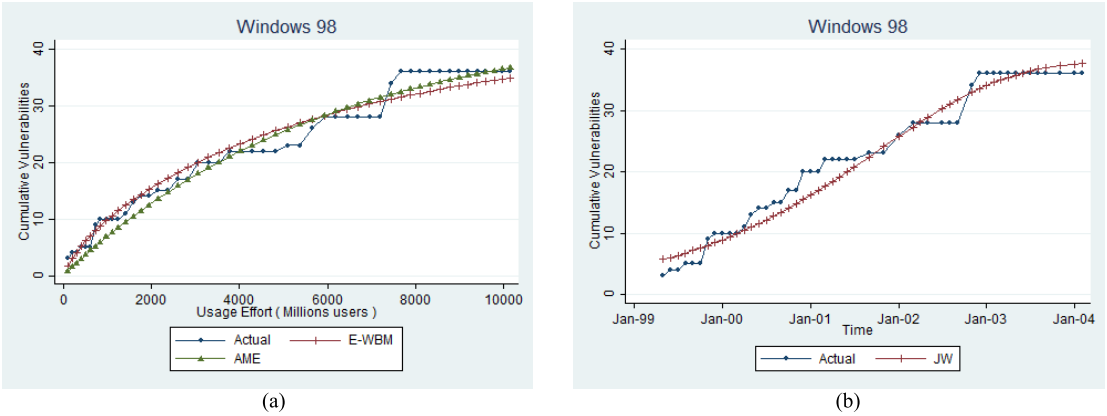


FIGURE 2. Model fitting for Windows 98. (a) Effort-based model, (b) Time-based model.

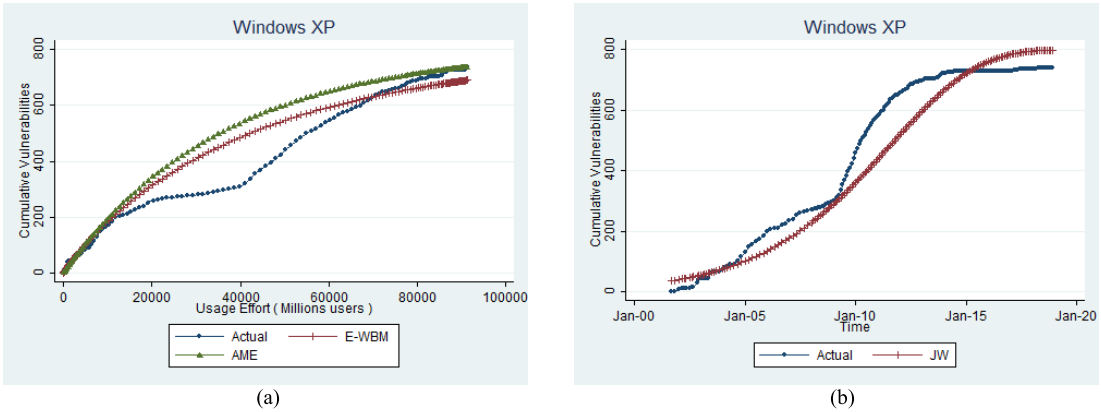


FIGURE 3. Model fitting for Windows XP. (a) Effort-based model, (b) Time-based model.

core parameters for several operating systems are shown in Table 3.

As shown in Table 3, the estimated values of all model parameters are valid. Next we analyze the results in detail from the perspective of 2 assessment indicators.

a) The P-value of these three model parameters are less than 5% and close to 0, stating that the estimated values of three model parameters are relatively significant.

b) The parameter α_0 of E-WBM model, the parameter λ_{uv} of AME model, and the parameter k of JW model all fall

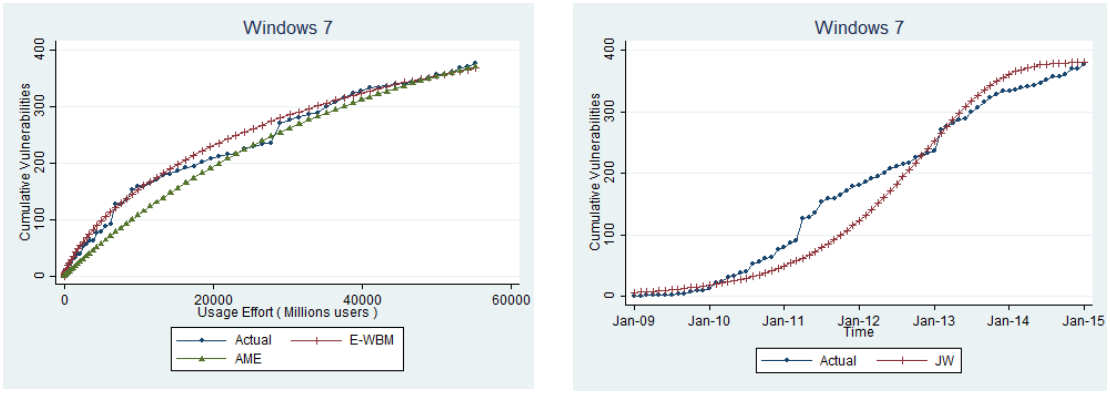


FIGURE 4. Model fitting for Windows 7. (a) Effort-based model, (b) Time-based model.

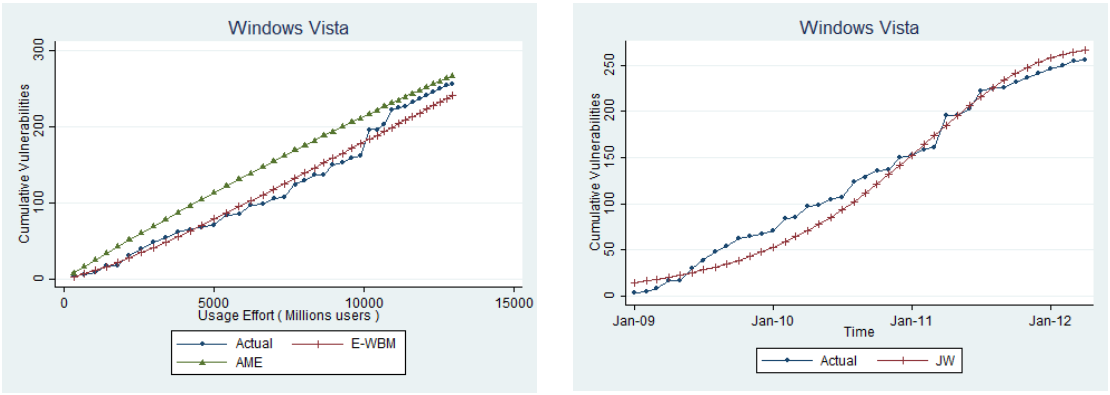


FIGURE 5. Model fitting for Windows Vista. (a) Effort-based model, (b) Time-based model.

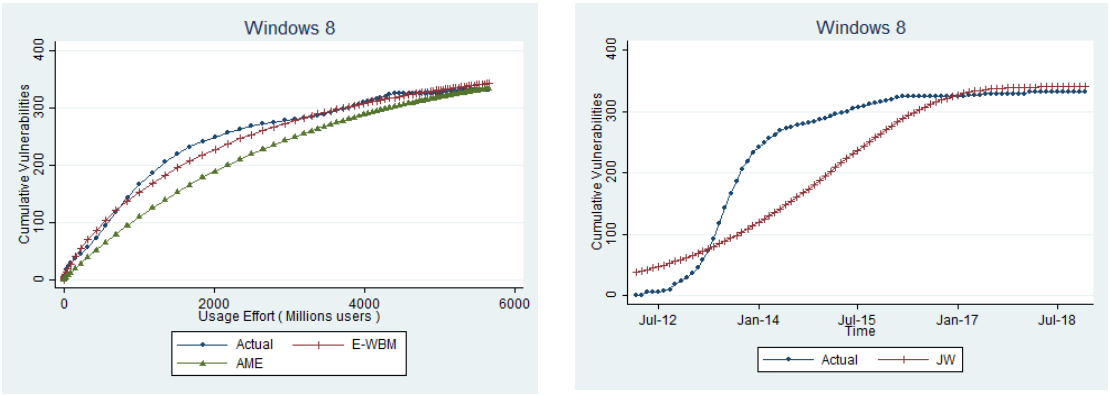


FIGURE 6. Model fitting for Windows 8. (a) Effort-based model, (b) Time-based model.

within the 95% confidence interval for all operating systems, which means the strong validity of these model parameters.

3) STEP 3: FIT THE VDMS TO OBSERVED SAMPLES

The values of parameters in Table 2 are taken into Eq. (9), Eq. (4), and Eq. (2), respectively, and the fitting graph of three models for each operating system can be obtained, as shown in Fig.1-Fig.8. In Fig.1-Fig.8, the Actual curve indicates the reality number of vulnerabilities found, which is obtained according to the vulnerability datasets V, while the E-WBM curve, AME curve, and JW curve are the vulnerabilities discovery trends which are fitted by the E-WBM model, AME model, and JW model, respectively. Since the reference of time-based model JW is different from the

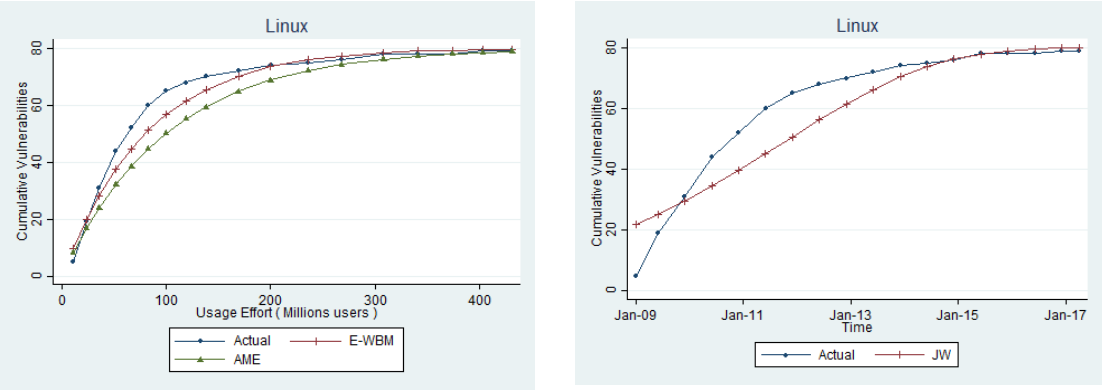


FIGURE 7. Model fitting for Linux. (a) Effort-based model, (b) Time-based model.

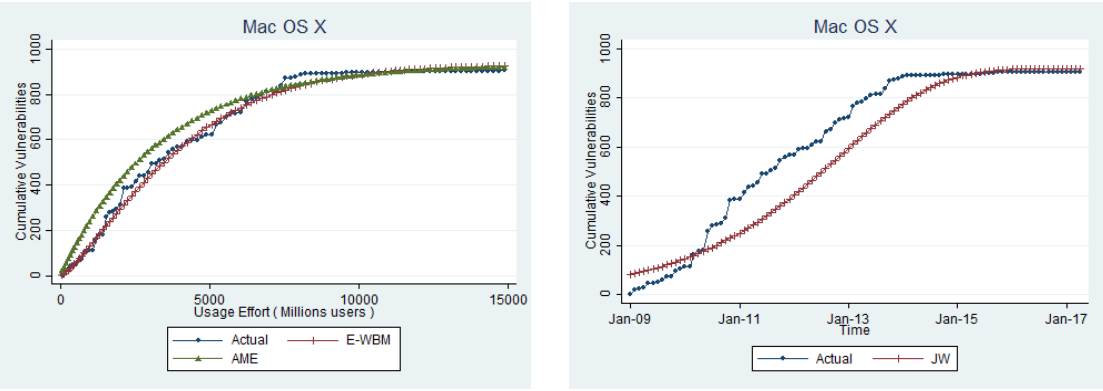


FIGURE 8. Model fitting for Mac OS X. (a) Effort-based model, (b) Time-based model.

other two models, we set abscissa reference for them separately through referring to the comparison of Woo et al. [2]. Fig.1(a)-Fig.8(a) show cumulative vulnerabilities by the number of each dataset installations in terms of million users-months and the fitted effort-based model E-WBM and AME. Fig.1(b)-Fig.8(b) show the cumulative number of vulnerabilities by month and the fitted time-based model JW for 8 operating systems.

Here we discuss the Fig.1-Fig.8 from the following two perspectives.

a: CURVE TRENDS

As can be seen from Fig.1-Fig.8, the fitting curves of E-WBM model are generally consistent with their Actual curves. In particular, the fitting curve of Mac OS X is almost consistent with the overall trend of Actual curve, which is the most accurate overall forecast. The E-WBM curve trend and Actual curve of Windows 7 and Windows 8 are generally the same in the early and late stages, but there are tiny deviations in the middle stage. The E-WBM curve of Linux fits well with the Actual curve in the middle and late stages. In addition, the E-WBM curve of Windows 95 is also excellent in the middle and late stages; while Windows 98 and Windows Vista are just opposite. Their E-WBM curve trends fit well in the early stage. Unfortunately, the E-WBM curve of Windows XP is not similar to the trend of Actual curve in the middle and late stages, which may be due to the different sources of datasets.

Furthermore, we also show the AME curves and the JW curves of these operating systems to compare the fitting effects of those three models. It can also be seen from Fig.1-Fig.8 that the fitting effects of E-WBM curves are significantly better than the AME curves and JW curves. In particular, the E-WBM curves of Mac OS X, Windows 7, and Windows 8 are analogous to the trend of Actual curves, but the curve rate of AME differs from the Actual curve in the early phase; while the curves of JW have large deviation in the middle stage. Three fitting curves and the Actual curves of Linux are alike; nevertheless, the growth rate of AME curve and JW curve are slower than the Actual curve over the entire period compared to E-WBM curve. Similarly, the E-WBM curve and AME curve of Windows 95, Windows 98, and Windows XP generally overlap, which are similar to the trend of Actual curve, indicating that they have analogical fitting effects on these three operating systems. The trend of JW curve is generally similar with the Actual curve for Windows 98 and Windows XP, but it deviates from the Actual curve for

Windows 95. For Windows Vista, the disparity between AME curve and Actual curve is obvious, while the E-WBM curve and JW curve only have a smaller deviation than AME.

Even though the Mac OS X has the most prominent fitting in these operating systems, here we take Windows 95 and Window 7 with modest performance as typical examples to analyze the fitting effect of proposed model since majority Windows operating system were adopted in the experiment. For Windows 95, the E-WBM curve and AME curve are basically coincident, and these two curves are roughly the same as the trend of Actual curve, while its JW curve has a large deviation. As the testing effort increases, the cumulative number of vulnerabilities shows an increasing rate at the beginning. After some time, a steady rate of vulnerability finding generates a linear curve. Finally, the number of vulnerabilities remains unchanged, which could be considered to be consistent with the characteristics of *S*-shaped curve. However, in the early and middle stages, the E-WBM and AME curve grow more slowly than the Actual curve. Overall, it can be considered that both models satisfy the distribution of actual vulnerabilities in Windows 95.

Similar to Windows 95, the cumulative number of vulnerabilities for Windows 7 also could be considered to basically conform to the characteristic of *S*-shaped curve. The fitting curves of three models are generally in line with the trend of actual curve, but the fitting effect of E-WBM is close to the actual curve than the other two models. Specifically, the AME curve and JW curve grow more slowly than the Actual curve in the early stage and middle stage, respectively. In general, these three models are considered to satisfy the distribution of actual vulnerabilities for Windows 7.

b: NUMBER OF VULNERABILITIES FOUND

At most effort nodes, the cumulative number of vulnerabilities predicted by E-WBM model is generally not much different from the actual number of vulnerabilities. Specifically, Mac OS X is still the most accurate for fitting the cumulative number of vulnerabilities. The number of vulnerabilities fitted by E-WBM for Linux is almost the same as the actual number in the middle and late stages, while Windows 7 and Window 8 are relatively accurate in the early and late stages. Although Windows 95 has a tiny deviation in the middle and late stages, its cumulative number of vulnerabilities is almost close to the actual number in the early stage. In addition, Windows 98, Windows Vista, Windows XP also hardly have deviation from the actual number in the early stages.

Additionally, we further compare the cumulative number of vulnerabilities in the E-WBM model with the AME model and JW model. The AME model and JW model for most operating systems are roughly similar to the actual values, but the overall deviation of E-WBM model from the actual number is less than AME model and JW model such as Mac OS X, Linux, Windows 7, Windows 8, Windows 95, and Windows 98. Besides, the fitting of E-WBM and JW are almost similar for Windows XP, and their overall deviation are less than AME model. Similarity, for Windows Vista, the number of vulnerabilities fitted by E-WBM and JW are also close to the actual number, while the fitted values of AME are extremely higher than the actual number.

Specifically, in Windows 95, the cumulative number of vulnerabilities fitted by the E-WBM model and AME model are nearly identical and are comparable to actual cumulative number of vulnerabilities in most cases, especially in the early stage and middle stage; nevertheless, there is a wide gap between the fitted value of JW model and the actual value. For example, when the effort is 1072 (i.e., Feb-00 in Time), the fitted values of both effort-based models are identical to the actual values, meaning that there is no deviation at this point, while the fitted values of JW model are around 12.5% lower than the actual value. However, in the middle and late stages, the cumulative number of vulnerabilities fitted by the AME model exceeds the E-WBM model, and the deviation of AME is larger than that of E-WBM model compared with the actual number of vulnerabilities, while the JW is lower than the actual value. For example, the fitted value of E-WBM is the same as the actual value when the effort is 2132 (i.e., Dec-00 in Time), while the fitted values of AME and JW are around 4.3% higher and 21.7% lower than the actual value, respectively. When the effort is 2618 (i.e., Jul-01 in Time), the fitted values of E-WBM and AME are approximately 4.1% and 8.3% higher than the actual values, respectively; nevertheless, the fitted value of JW is around 12.5% lower than the actual value. In general, the deviation of E-WBM model is relatively smaller than that of AME model and JW model. Hence, the accuracy of E-WBM model for fitting Windows 95 is slightly better than AME model and JW model.

In Windows 7, the fitted values of E-WBM are nearly identical to the actual values in most cases, especially in the early and late stages; nevertheless, the fitted values of AME and JW are only close to the actual values in the late stage and early stage, respectively. The fitted values of AME are generally smaller than the actual value in the early and middle stages; while the fitted values of JW are smaller than the actual values in the middle stage, and are larger than the actual values in the late stage. Overall, the deviation of E-WBM is generally smaller than the other two models for Windows 7.

Although the E-WBM model has tiny deviations in the partial life cycle of the software, the overall deviation is quite small, indicating that E-WBM model still works as expected. In other words, proposed model can commendably describe vulnerabilities discovery process for several operating systems such as Windows 98, Windows 7, Linux, and Mac OS X. In addition, the overall deviation of E-WBM model fitting on these operating systems is smaller than that of AME model and JW model.

4) STEP 4: PERFORM GOODNESS OF FIT QUALITY ANALYSIS

To verify the accuracy of proposed model, we perform a fitness test on these three models by employing 2 assessment indicators described in Section V (B). The goodness of fit

TABLE 4. Goodness of fit test results for three VDMS.

Operating System	E-WBM		AME		JW	
	R^2	χ^2	R^2	χ^2	R^2	χ^2
Windows 95	0.95710075	4.0079393	0.88668925	5.5820341	0.79267079	22.750101
Windows 98	0.96270394	9.8551292	0.95590609	28.812593	0.93964618	15.089759
Windows XP	0.96254331	1452.6803	0.95567364	2265.282	0.86565888	2381.5596
Windows 7	0.98783791	77.143715	0.99174356	526.33252	0.8896355	814.99268
Windows Vista	0.98470801	38.89164	0.96195626	316.0076	0.90885776	139.15123
Windows 8	0.99242264	38.099346	0.97320372	555.5321	0.6582827	2337.7976
Linux	0.94714856	8.7635155	0.86532685	28.767954	0.78702974	34.694008
Mac OS X	0.98018599	142.79208	0.84265836	1156.9932	0.79245663	2846.178

test results of three models for several operating systems are summarized in Table 4.

As is depicted in Table 4, for most operating system, the goodness of fit of E-WBM model observably exceeds the JW model for all operating systems, and also significantly outperforms the AME model such as Windows 95, Windows 98, Window XP, Windows Vista, Windows 8, Linux, and Mac OS X, but slightly weaker for Windows 7. Next, we compare in detail from the view of 2 assessment indicators.

a) The R^2 values of two effort-based model E-WBM and AME are close 1, and are better than the R^2 values of time-based model JW, indicating that the fitting effects of two effort-based models are commendable. The R^2 values of the E-WBM model, however, are generally greater than AME and closer to 1. More importantly, the R^2 values of the E-WBM model significantly exceed about 7.9%, 9.4%, and 16.3% of the AME on Windows 95, Linux, and Mac OS X, respectively. Besides, the R^2 values of E-WBM on Window 98, Windows XP, Windows Vista, and Windows 8 are slightly greater than AME, about 0.7%, 0.7%, 2.4%, and 1.9% of AME, respectively. For Windows 7, although the R^2 value of E-WBM is slightly lower than AME, they are both close to 1. Overall, the goodness of fit of proposed model is still acceptable.

b) For all the operating systems, the chi-square coefficient χ^2 of E-WBM model is extremely smaller than that of the AME model and JW model, which demonstrates the accuracy of proposed model. Specifically, compared to the χ^2 values of AME for Windows 95, Windows 98, Windows XP, Windows 7, Windows Vista, Windows 8, Linux, and Mac OS X, the χ^2 values of E-WBM are even declined by 28.2%, 65.8%, 35.8%, 85.3%, 87.6%, 93.1%, 69.5%, and 87.6% of AME, respectively. In addition, the χ^2 values of E-WBM are reduced by 82.3%, 34.7%, 39.0%, 90.5%, 72.0%, 98.3%, 74.7%, and 94.9% of JW, respectively. It shows that E-WBM model fits more accurate than AME model and JW model on all operating systems. In particular, for Windows 8, the χ^2 values of JW are extremely higher than the other two effort-based models, which indicates that the JW model does not fit well on Windows 8. The possible reason may be that the time intervals of original datasets we adopted are not exactly equal.

5) STEP 5: PERFORM PREDICTABILITY ANALYSIS

Even proposed model shows a nice goodness of fit in the previous step, it may not mean that the model can predict well in the future. The core purpose for a vulnerability discovery model is to estimate the number of security vulnerabilities that are likely to be discovered in the future using the available data. Thus, prediction capability should be considered more significant than model fitting. Models with good predictive capabilities can be used to evaluate the resources needed for maintenance and risk estimation.

Here, we use 2 normalized predictability indicators described in Section V (B) to examine the predictability of two effort-based model E-WBM and AME [32], that is Average Error (AE) and Average Bias (AB).

To compare E-WBM and AME models, we refer to the experimental setup of Woo *et al*. [2] to specify different starting points for these operating systems, for example, the starting points for Windows 95, Windows 98, and Windows 7 are chosen to be when cumulative installed base exceeds 300 million. Since only after some significant past data, the effort-based model can predict the future trend. The normalized prediction error values $((\Omega_i - \Omega)/\Omega)$ for these operating systems at each data point are plotted in Figure 9, and the values of AE and AB are given in Table 5. Note that while AE is always positive, AB may be positive or negative.

The error plots in Figure 9 clearly show that the E-WBM model yields a more stable prediction with a significantly less error in most situations than AME model. For Windows Vista, Windows 8, and Linux, the prediction error of E-WBM model smoothly fluctuates around zero and approaches zero, while the AME model fluctuates greatly. For Windows 7 and Mac OS X, the error of E-WBM model has obviously fluctuation in the early period, but gradually approaches zero in the late stage. In addition, for Windows 95, Windows 98, and Windows XP, in spite of the fitting curves of E-WBM and AME are rather similar, the two models produce significantly different predictions.

In Table 5, the AB and AE values illustrate that the E-WBM model always performs better than AME model. Specifically, the AB and AE values of E-WBM are substantially less than the values of AME, especially for Windows Vista, Windows 7, Windows 8, Linux, and Mac OS X, which

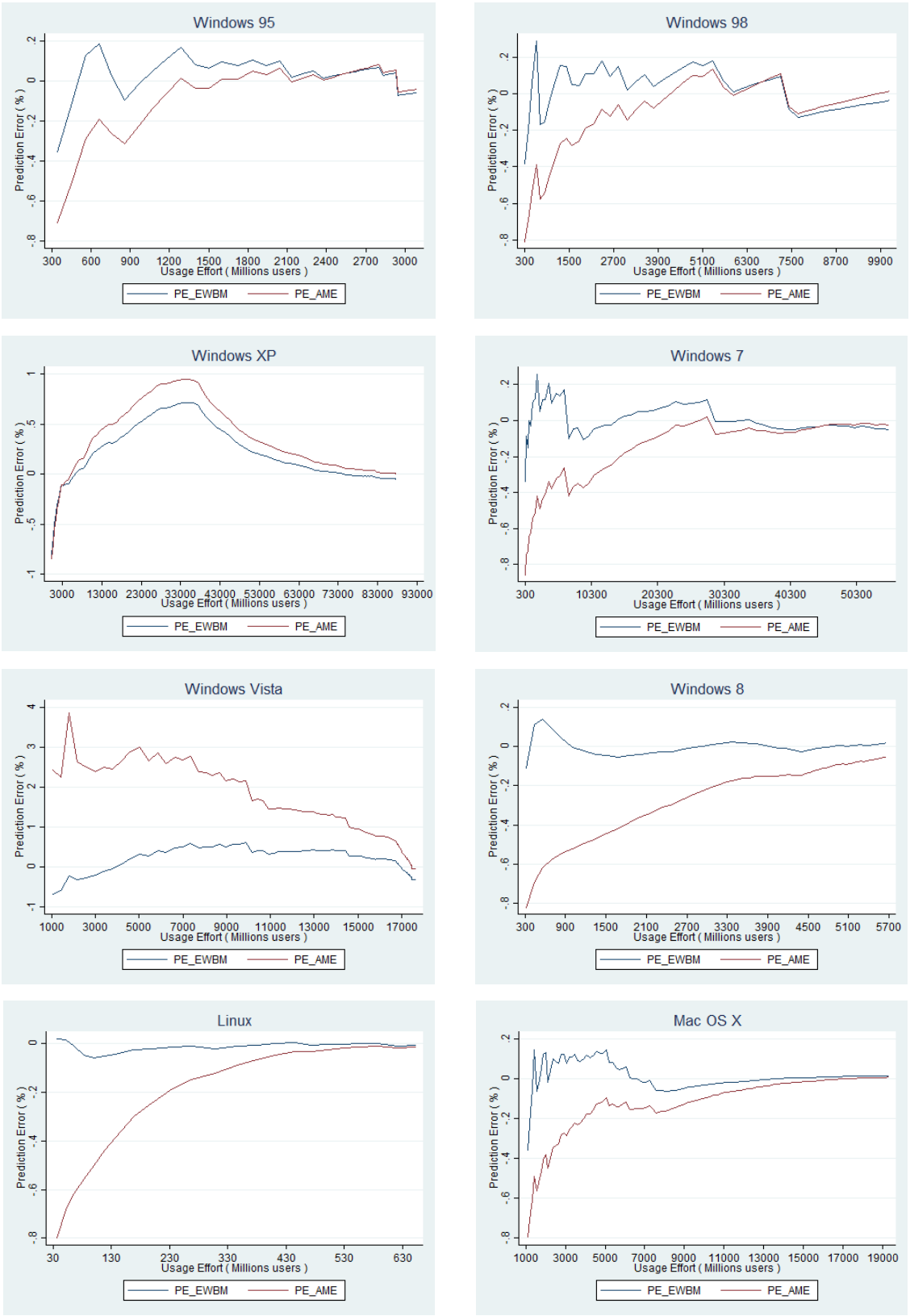


FIGURE 9. Prediction errors for E-WBM and AME models.

TABLE 5. Average bias and average error (unit: %).

Operating System	E-WBM		AME	
	AB	AE	AB	AE
Windows 95	0.01255229	0.07091355	-0.0506373	0.09049314
Windows 98	0.01802514	0.1030656	-0.13107689	0.16330409
Windows XP	0.08260208	0.16592178	0.17287928	0.21941578
Windows 7	0.00923179	0.0682444	-0.21149062	0.21227586
Windows Vista	0.0861505	0.29126519	1.023167	1.0366092
Windows 8	0.0005456	0.01954102	-0.19932722	0.19932722
Linux	-0.01383351	0.01725643	-0.23245387	0.23245387
Mac OS X	0.01162409	0.04730872	-0.14405665	0.14468642

demonstrates superior prediction capabilities for these operating systems. In addition, the Table 5 also suggests that the E-WBM generally tends to overestimate the actual values and AME mainly trends to underestimate the actual values.

VI. DISCUSSION

We discuss the experimental results of E-WBM, the limitations of E-WBM, and future direction of research.

a: VERIFICATION OF E-WBM

We first evaluate the efficiency of E-WBM on 8 operating systems by 2 assessment indicators. The result is summarized in Table 4. For all the operating systems tested, E-WBM has good data fit as expected, especially for Mac OS X, Linux, Windows 7, and Windows 8 operating systems. However, the fitting effect of E-WBM is not accurate enough for these operating systems, such as the early stage for Windows 95 and the middle and late stage for Windows 98. There are two reasons for this: 1) the lack of complete and detailed datasets, 2) other factors such as limited software information and software code sharing. Thus, it can be considered that the overall fitting effect of E-WBM for each operating system reaches the expected experiment results. In other words, E-WBM is quite suitable vulnerability discovery model for operating system software.

b: COMPARED WITH AME AND JW

We then compare E-WBM with other state-of-the-art effort-based model AME and time-based model JW; E-WBM outperforms them (Table 4). The comparison experiment concludes that E-WBM, AME, and JW generally meet the expected results for those 8 operating systems, but E-WBM achieves significantly more accuracy than AME and JW, especially for Linux, Windows Vista, Windows 7, Windows 8, and Mac OS X operating systems; nevertheless, the time-based model JW has a large deviation for Windows 8, especially in the middle stage. One of the possible reasons is that the time intervals of original datasets we adopted are not exactly equal. The JW model is heavily influenced by time, thus our partial datasets are not good exhaustive for time-based model.

c: PREDICTABILITY OF E-WBM

We further examine the predictive capability of two effort-based model E-WBM and AME using two normalized predictability measures AE and AB (Table 5). The prediction experiment shows that the prediction error of E-WBM is always less than AME for these operating systems, which means that our model has better predictive capability than AME model.

d: LIMITATIONS

Although accurate, E-WBM is a VDM for predicting future vulnerabilities, so the proposed model is uncertain, and still has a lot to be improved. Since it is difficult to obtain a complete dataset, the model that describes the vulnerability discovery process of some software is not accurate enough. For example, the absence of datasets after 2004 may be the main reason for the partial difference between the E-WBM curve and Actual curve trend for Windows 95 and Windows 98. The lack of datasets at the beginning of software release may also be one of the reasons for the unsatisfactory fitting effect of Windows Vista. Different sources of datasets are one of the possible reasons why the fitting of E-WBM for Windows XP fail to attain the anticipated results. Additionally, other factors, such as limited software information and software code sharing, also may affect the experiment results. We plan to overcome this limitation by extending other environment factors to enhance E-WBM's predictive capability.

VII. CONCLUSION

This paper presented E-WBM, an efficient effort-based VDM that uses a Weibull probability distribution function to improve the vulnerability discovery rate algorithm of AME model. We further demonstrated how testing effort parameters and core parameters can be efficiently obtained by performing least squares method on E-WBM model. E-WBM significantly outperformed the state-of-the-art vulnerability discovery model AME and JW in 2 assessment indicators: 1) its curve trend was closer to actual curve than the AME curve and JW curve, and 2) its overall deviation was either equal or less than that of the AME model and JW model. More importantly, E-WBM yields a more stable prediction with a significantly less prediction error than AME. Our work showcases a successful application of Weibull probability distribution function to effort-based VDMs. We believe our results demonstrate the vast potential of leveraging Weibull probability distribution function to perform software vulnerability analysis for vulnerability predicting, vulnerability evaluation, etc. We hope our work will encourage more researchers to explore such directions.

There are several interesting directions for future work. While the focus of our paper was on applying testing effort, it would be worth exploring how to apply, as comprehensively

as possible, the multivariate that can characterize multiple factors. Perhaps a combination of multiple external factors could be powerful enough to achieve preeminent prediction accuracy. Also, our datasets are currently agnostic to other application software. We are considering using some form of proxy as the alternatives of relevant variables, which could potentially obtain datasets more expediently.

REFERENCES

[1] O. H. Alhazmi and Y. K. Malaiya, "Quantitative vulnerability assessment of systems software," in *Proc. Annu. Rel. Maintainability Symp.*, Alexandria, VA, USA, Jan. 2005, pp. 615–620.

[2] S.-W. Woo, H. Joh, O. H. Alhazmi, and Y. K. Malaiya, "Modeling vulnerability discovery process in apache and IIS HTTP servers," *Comput. Secur.*, vol. 30, no. 1, pp. 50–62, Jan. 2011.

[3] H. Joh, J. Kim, and Y. K. Malaiya, "vulnerability discovery modeling using weibull distribution," in *Proc. 19th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2008, pp. 299–300.

[4] C. Z. Hu, R. Ma, Y. F. Zhang, B. Li, and Y. Liu, "A modeling method of software security vulnerability discovery model based on testing effort," C.N. Patent 2015 10 711 744.9, Sep. 29, 2017.

[5] R. Anderson, "Security in open versus closed systems—The dance of Boltzmann, coase and Moore," *Open Source Softw. Econ.*, pp. 127–142, Jul. 2002.

[6] O. H. Alhazmi and Y. K. Malaiya, "Modeling the vulnerability discovery process," in *Proc. 16th IEEE Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2005, pp. 129–138.

[7] E. Rescorla, "Is finding security holes a good idea?" *IEEE Security Privacy*, vol. 3, no. 1, pp. 14–19, Jan. 2005.

[8] F. T. Bonner and T. R. Dell, "The Weibull function: A new method of comparing seed vigor," *J. Seed Tech.*, vol. 1, no. 1, pp. 96–103, 1976.

[9] S. Yamada, J. Hishitani, and S. Osaki, "Software-reliability growth with a Weibull test-effort: A model and application," *IEEE Trans. Rel.*, vol. 42, no. 1, pp. 100–106, Mar. 1993.

[10] S. R. Cain, "Distinguishing between lognormal and Weibull distributions [time-to-failure data]," *IEEE Trans. Rel.*, vol. 51, no. 1, pp. 32–38, Mar. 2002.

[11] B. B. Sagar, R. K. Saket, and S. C. Gurmit, "Exponentiated Weibull distribution approach based inflection S-shaped software reliability growth model," *Ain Shams Eng. J.*, vol. 7, no. 3, pp. 973–991, Sep. 2015.

[12] A. Ahmad, A. A. Soliman, and M. M. Yousef, "Bayesian estimation of exponentiated weibull distribution under partially acceleration life tests," *Bull. Malaysian Math. Sci. Soc.*, vol. 39, no. 1, pp. 227–244, Jan. 2016.

[13] G. S. Rao, M. Aslam, and O. H. Arif, "Estimation of reliability in multicomponent stress–strength based on two parameter exponentiated Weibull Distribution," *Commun. Statist.-Theory Methods*, vol. 46, no. 15, pp. 7495–7502, Feb. 2017.

[14] K. Chen, D. G. Feng, P. R. Su, C. J. Nie, and X. F. Zhang, "Multi-Cycle vulnerability discovery model for prediction," *J. Softw.*, vol. 21, no. 9, pp. 2367–2375, 2010.

[15] A. Anand and N. Bhatt, "Vulnerability discovery modeling and weighted criteria based ranking," *J. Indian Soc. Probab. Statist.*, vol. 17, no. 1, pp. 1–10, Jun. 2016.

[16] J. Kim, Y. K. Malaiya, and I. Ray, "Vulnerability discovery in multi-version software systems," in *Proc. 10th IEEE High Assurance Syst. Eng. Symp. (HASE)*, Nov. 2007, pp. 141–148.

[17] A. Anand, S. Das, D. Aggrawal, and Y. Klochkov, "Vulnerability discovery modelling for software with multi-versions," *Adv. Reliab. Syst. Eng.*, pp. 255–265, Dec. 2017. doi: 10.1007/978-3-319-48875-2_11.

[18] N. Bhatt, A. Anand, V. S. S. Yadavalli, and V. Kumar, "Modeling and characterizing software vulnerabilities," *Int. J. Math., Eng. Manage. Sci.*, vol. 2, no. 4, pp. 288–299, 2017.

[19] S. Brocklehurst, B. Littlewood, T. Olovsson, and E. Jonsson, "On measurement of operational security," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 9, no. 10, pp. 7–16, Oct. 1994.

[20] T. Olovsson, E. Jonsson, S. Brocklehurst, and B. Littlewood, "Towards operational measures of computer security: Experimentation and modelling," *Predictably Dependable Comput. Syst.*, pp. 555–569, 1995. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-79789-7_31

[21] O. Alhazmi, Y. Malaiya, and I. Ray. (Jul. 2004). *Vulnerabilities in Major Operating Systems*. [Online]. Available: https://www.researchgate.net/publication/2906333

[22] Y. K. Malaiya and J. Denton, "What do the software reliability growth model parameters represent?" in *Proc. 8th Int. Symp. Softw. Rel. Eng.*, Nov. 1997, pp. 124–135.

[23] H. Li, Q. Li, and M. Lu, "A software reliability growth model considering an S-shaped testing effort function under imperfect debugging," *J. Harbin Eng. Univ.*, vol. 37, no. 2, pp. 149–154, Nov. 2011.

[24] H.-F. Li, S. Q. Wang, C. Liu, J. Zhen, and Z. Li, "Software reliability model considering both testing effort and testing coverage," *J. Softw.*, vol. 24, no. 4, pp. 749–760, Apr. 2013.

[25] Y. X. Wan and F. D. Shu, "Estimation model for software testing effort based on value," *Comput. Eng. Des.*, vol. 29, no. 3, pp. 544–546, 2008.

[26] Y. Kansal, P. K. Kapur, U. Kumar, and D. Kumar, "User-dependent vulnerability discovery model and its interdisciplinary nature," *Life Cycle Rel. Saf. Eng.*, vol. 6, no. 1, pp. 23–29, Mar. 2017.

[27] R. Johnston, "A multivariate Bayesian approach to modeling vulnerability discovery in the software security lifecycle," Ph.D. dissertation, Dept. Eng. Appl. Sci., George Washington Univ., Washington, DC, USA, 2018.

[28] R. Johnston, S. Sarkani, T. Mazzuchi, T. Holzer, and T. Eveleigh, "Bayesian-model averaging using MCMCBayes for Web-browser vulnerability discovery," *Rel. Eng. Syst. Saf.*, vol. 183, pp. 341–359, Mar. 2019.

[29] R. Johnston, S. Sarkani, T. Mazzuchi, T. Holzer, and T. Eveleigh, "Multivariate models using MCMCBayes for web-browser vulnerability discovery," *Elsevier Rel. Eng. Syst. Saf.*, vol. 176, pp. 52–61, Aug. 2018.

[30] Y. Kansal, P. K. Kapur, and U. Kumar, "Coverage-based vulnerability discovery modeling to optimize disclosure time using multiattribute approach," *Qual. Rel. Eng. Int.*, vol. 35, no. 1, pp. 62–73, Feb. 2018.

[31] *NVD Metabase*. Accessed: Apr. 21, 2017. [Online]. Available: http://nvd.nist.gov

[32] H. Joh and Y. K. Malaiya, "Modeling skewness in vulnerability discovery," *Qual. Rel. Eng. Int.*, vol. 30, no. 8, pp. 1445–1459, Dec. 2014.

[33] A. Younis, H. Joh, and Y. Malaiya, "Modeling learningless vulnerability discovery using a folded distribution," in *Proc. Int'l Conf. Secur. Manage.*, vol. 11, Jul. 2011, pp. 617–623.

[34] F. Massacci and V. H. Nguyen, "An empirical methodology to evaluate vulnerability discovery models," *IEEE Trans. Softw. Eng.*, vol. 40, no. 12, pp. 1147–1162, Dec. 2014.

[35] O. H. Alhazmi and Y. K. Malaiya, "Prediction capabilities of vulnerability discovery models," in *Proc. Annu. Rel. Maintainability Symp.*, vol. 2, Jan. 2006, pp. 86–91.

[36] J. D. Musa and K. Okumoto, "A logarithmic poisson execution time model for software reliability measurement," in *Proc. 7th Int. Conf. Softw. Eng.*, Mar. 1984, pp. 230–238.

[37] P. L. Li, M. Shaw, J. Herbsleb, B. Ray, and P. Santhanam, "Empirical evaluation of defect projection models for widely-deployed production software systems," *ACM SIGSOFT Softw. Eng. Notes*, vol. 29, no. 6, pp. 263–272, Nov. 2004.

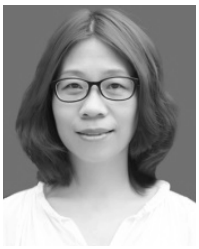
[38] Y. Zhou and J. Davis, "Open source software reliability model: An empirical approach," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–6, Jul. 2005.

[39] *Internet Live Stats*. Accessed: Apr. 21, 2017. [Online]. Available: http://www.internetlivestats.com

[40] *StatCounter Global Stats*. Accessed: Apr. 21, 2017. [Online]. Available: http://gs.statcounter.com

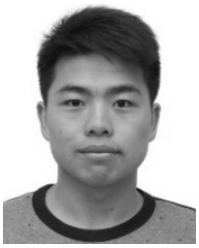
[41] *W3School*. Accessed: Jan. 8, 2019. [Online]. Available: https://www.w3schools.com/browsers

XIAJING WANG received the B.E. degree in software engineering from the Taiyuan University of Technology, China, in 2016. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Beijing Institute of Technology, China. Her research interests include information security and vulnerability discovery.



RUI MA received the Ph.D. degree from the Beijing Institute of Technology, China, in 2004, where she is currently an Associate Professor with the School of Computer Science and Technology. Her current research interests include software security, the Internet of Things, neural networks, and data mining.

DONGHAI TIAN received the Ph.D. degree from the Beijing Institute of Technology, China, in 2012, where he is currently a Teacher with the School of Computer Science and Technology. His current research interests include software security, malware analysis and detection, Android security, and cloud security.



BINBIN LI received the B.E. degree in software engineering from the Beijing Institute of Technology, China, in 2018, where he is currently pursuing the M.S. degree with the School of Computer Science and Technology. His research interests include software security and vulnerability analysis.

XUEFEI WANG received the B.E. degree in computer science and technology from Hunan Normal University, China, in 2018. She is currently pursuing the M.S. degree with the School of Computer Science and Technology, Beijing Institute of Technology, China. Her research interests include software security and vulnerability discovery.