Proceedings of the Pakistan Academy of Sciences:
A. Physical and Computational Sciences: 57 (3): 47-60 (2020)
Copyright © Pakistan Academy of Sciences
ISSN: 2518-4245 (print), 2518-4253 (online)

Pakistan Academy of Sciences

Research Article

An Optimization of Vulnerability Discovery Models using Multiple Errors Iterative Analysis Method

Gul Jabeen[1,2], Sabit Rahim[1*], Gul Sahar[1], Akber Aman Shah[2], and Tehmina Bibi[3]

[1]Karakoram International University Gilgit, Pakistan
[2]Tsinghua University, Beijing, China
[2]School of Economics and Management, University of Chinese Academy of Science, Beijing, China
[3]Institute of Geology, University of Azad Jammu & Kashmir Muzaffarabad, Pakistan

Abstract: A vulnerability discovery model (VDMs) play a central role to model the rate at which vulnerabilities are discovered for software. Though, these models have various shortcomings *viz*., multi VDMs, changes in VDMs, and development of new VDMS for different datasets due to diverse approaches and assumptions in their analytical formation. There is a clear need for intensive investigation and extensive use of these models to enhance the predictive accuracy of existing VDMs. In this paper, to enhance the predictive accuracy of existing VDMs, a multiple error iterative analysis method (MEIAM) along with artificial neural network sign estimators has been proposed based on the residual errors. Our findings reveal that the proposed method optimizes to fit historical vulnerability accurately and helps to predict future trends of vulnerabilities across different datasets and models. Repeated calculations of residual errors using these models are used to improve and adjust the forecast accuracy to the expected level. The experiment performed by using real vulnerability data of three type's popular software: Windows 10 (613), Android 7.0 (1018), Internet Explorer 11 (60), and Firefox 20 (502), starting from the first day of the issue or the earliest available in NVD database. The results demonstrate that the method is universally applicable to any of the VDMs to improve predictive accuracy.

Keywords: Optimization, Vulnerability, HPEIAM, Discovery models, Artificial neural network.

1. INTRODUCTION

With the development of internet technology, software systems have become larger and more complex. Software vulnerabilities have also increased rapidly, potentially causing an increasing number of serious security threats. A critical vulnerability provides an attacker with the ability to access full control of software[1]. However, much of the work on security has been qualitative, focused on the detection and prediction of vulnerabilities present in these systems. If the software developers can make accurate quantitative predictions of the vulnerability discovery, they can optimally assign the needed resources that are likely to be required for the patch management [2-4].

Several software vulnerability discovery models (VDM) have been proposed during the past few decades to model the vulnerabilities with code attributes: text analysis and mining of source code [5,6]. Code-attributes-based data models have some disadvantages: the source code is not available, especially for commercial software, and the software developers continuously change source code in the software. This continues until the release of the software (repair or update), thus predicting the future based on its static code attributes is often not possible. Much of the current research focuses on time-series-based vulnerability discovery models [7, 8].

The first VDM model is proposed by anderson, which is termed as Andeson Thermodynamic (AT)

Received: November 2019; Accepted : September 2020
*Corresponding Author: Sabit Rahim <sabit.rahim@kiu.edu.pk>

model [9]. Many other statistical models are used in literature, that either try to capture the underlying processes or apply principles used in other fields of science to discover vulnerabilities. Among them, the exponential model is designed to fit the real data [10]. In this model, two possible trends were examined such as the quadratic model and the exponential model. The logarithmic model shows the total number of vulnerabilities as logarithmic growth which was first proposed by poisson [11] and is used by Rahimi [12] by fitting the model to the vulnerabilities of a specific application. Alhazmi et al. proposed a logistic model called the AML model in [13] and examined it in [14]. The predictive capabilities were evaluated in [15] and [16] by using a different set of data. A multicycle vulnerability discovery model was proposed by Chan et al., which helps to extend the scope of existing models [17]. Other studies focused on increasing the accuracy of VDMs (weibull, normal, beta, and gamma distribution) by examining the skewness of the vulnerability data [18] or using the Bayesian theorem[19, 20].

A recent study in [21] compared the performance of the neural network and time-series models for vulnerability prediction and found that neural network outperforms in all the cases [22] proposed models for software vulnerability prediction and determine whether the software reliability growth models can be used to predict vulnerability discovery process and shows good prediction results [23]. Analyses the vulnerability data using the seasonal index and autocorrelation function approach, which can be used to improve the vulnerability discovery models. Sharma and Singh [24] proposed a new vulnerability discovery model based on the gamma distribution. Most of the research defined above gained great success in practice and attracted considerable attention. However, very little research has focused on the optimization of software vulnerability discovery models. Optimization indicates that the actual model property remains unchanged, but the performance of the model can be enhanced to fit better with the historical data and accurately predict the future vulnerability occurrences. These models use different physical approaches and make assumptions in their analytic formulation. Their parameters are defined explicitly and have physical interpretations. In order to determine the parameters, there are always a set of assumptions that have to be made. These assumptions generalize the model, and their applicability becomes a critical issue because there is no single model which can be universally applied in all situations. VDMs face challenges due to four assumptions: time, operational environment, independence, and static code [25].

Our purpose is to develop a technique that improves the performance of any generic model based on the expected accuracy that can be controlled by the user. We aim to find a universal technique, which is applicable to any data and model. Thus we propose an optimized method called multiple error iterative analysis method (MEIAM) that uses the residual error values between actual and estimated values iteratively to improve the fitting of historical vulnerability data, which is controlled externally by setting the expected accuracy for the specific failure dataset. The repeated combination of residual error modifications by using the proposed method facilitates a better process of the fitting model and provides a more accurate prediction of future accruing vulnerabilities in the software testing phase. An artificial neural network (ANN) is used to estimate signs, which are associated with residual errors. ANN facilitates the use of the proposed technique because VDMs do not deal with residual errors directly due to the randomly fluctuated positive and negative signs associated with them. The repeated computation of the same data makes the prediction accuracy of existing models significantly improved, and the expected precision can be achieved, without concern about the amount of data used.

The proposed technique applies to any of the vulnerability prediction models, so it is considered universal. Compared with VDMs, our technique can globally enhance the performance of models rather than locally change the functional attributes. It is the only study where we have introduced the expected accuracy, which can be controlled by the user by giving the threshold value to any of the specific data. We measure the performance of our proposed optimized technique in conjunction with the traditional statistical vulnerability discovery models by using different vulnerability datasets. The experimental results derived from the four datasets illustrate that the proposed technique can better fit the historical data and provides a more accurate prediction of future vulnerabilities. Our results demonstrate that our technique can work well with every set of data and improves the performance accuracy of every software vulnerability discovery model. The rest of this paper is organized as follows. In section 2, the proposed method is defined in detail. Numerical illustrations are given in section 3. In section 4, the experimental evaluation and performance analysis has been performed. Finally, we conclude our work in section 5.

2. PROPOSED WORK

Statistical vulnerability models are generally evaluated by using the fitting ability. However, the primary purpose of VDMs is to achieve the prediction accuracy precisely. We claim that the method which is proposed (MEIAM) can enhance the predicted accuracy of VDMs effectively. In the following section, our proposed method is described in detail.

2.1. Multiple Error Iterative Analysis Method (MEIAM)

To facilitate the description of our method, let us introduce some notations first. Assume that a set of known data $v_1(t), v_2(t), \ldots, v_n(t)$ is used to develop a mathematical model such as:

$$VDM = f(v_1(t), v_2(t), \ldots v_n(t), p_1, p_2, \ldots, p_r, t_1, t_2, \ldots, t_s) \quad (1)$$

Where $a_1, a_2, a_3, \ldots a_n$ are parameters and $t_1, t_2, \ldots t_s$ are specific variables. In general, these quantitative vulnerability discovery models are used to predict future trend data: $v_{n+1}(t), v_{n+2}(t), v_{n+3}(t), \ldots, v_{n+k}(t)$. The $p_1, p_2, p_3, \ldots p_n$ are unknown parameters that can be determined by using any input sequences $v_1(t), v_2(t), \ldots, v_n(t)$ with different methods such as the least-square method or the maximum likelihood method.

For convenience, this article discusses only one argument which is denoted as t. We can write the model Equation 1 as follows:

$$VDM = f(v_1(t), v_2(t), \ldots, v_n(t), p_1, p_2, \ldots, p_r, t) \quad (2)$$

After applying the input data sequences:

$$v_1(t), v_2(t), \ldots, v_n(t)$$

in any quantitative VDM, the parameters are estimated and also determine the approximation solution, which is also used to evaluate the fitting power of the model. The first approximation solution is written as follow:

$$\overline{v_1(t)}^{(1)}, \overline{v_2(t)}^{(1)}, \ldots, \overline{v_n(t)}^{(1)}$$

and its predicted values are defined as:

$$\overline{v_{n+1}(t)}^{(1)}, \overline{v_{n+2}(t)}^{(1)}, \ldots, \overline{v_{n+l}(t)}^{(1)}.$$

While the exact values corresponding to the predicted values are assumed to be:

$$v_{n+1}(t)^{(1)}, v_{n+2}(t)^{(1)}, \ldots, v_{n+l}(t)^{(1)}$$

Suppose the error values can be determined as:

$$\varepsilon_n{}^{(1)}(t) = v_n(t) - \overline{v_n(t)}^{(1)} \; (n = 1,2,\ldots),$$

Which is referred to as the first error sequence?

However, the errors obtained $\varepsilon_n^{(1)}(t)$ may be positive or negative depending on the predicted values. For more simplicity, we omit t, for instance $v_i(t), \overline{v_i(t)}^{(1)}, \varepsilon_i^{(1)}(t)$ and Equation (1) is abbreviated $v_i(t), \overline{v_i(t)}^{(1)}, \varepsilon_i^{(1)}(t)$ and $VDM=f(t)$ respectively. In our proposed technique, we used the error values $\varepsilon_n^{(1)}$ to get more accurate results than previously predicted solution $\overline{v_{n+i}}^{(1)}(i=1,2,\ldots,l)$ through multiple iterations. Therefore, we analyze the error data sequences $\varepsilon_1^{(1)}, \varepsilon_2^{(1)}, \ldots, \varepsilon_n^{(1)}$ same as $v_i^{(1)}(i= 1,2,\ldots,n)$. So, the mathematical VDM can be established as:

$$VDM = f\left(\varepsilon_1{}^{(1)}, \varepsilon_2{}^{(1)}, \ldots, \varepsilon_n{}^{(1)}, p_1{}^{(1)}, p_2{}^{(1)}, \ldots, p_r{}^{(1)}, t\right) \quad (3)$$

Where the parameters $p_1^{(1)}, p_2^{(1)}, \ldots, p_r^{(1)}$ are determined by the input error data sequences: $\varepsilon_i^{(1)}$, $i=1,2,\ldots,n$.

By using Equation 3, we obtained errors first approximate solution as: $\varepsilon_1^{(1)}, \varepsilon_2^{(1)}, \ldots, \varepsilon_n^{(1)}$ and errors predicted solution such as: $\overline{\varepsilon}_{n+1}^{(1)}, \overline{\varepsilon}_{n+2}^{(1)}, \ldots, \overline{\varepsilon}_{n+l}^{(1)}$. As we have used statistical vulnerability prediction models to estimate error approximation and predicted solutions, however, these models can use error inputs, but the signs associated with errors always change (positive or negative). Parametric VDMs only deals with positive input sequences, therefore we have changed the obtained errors sequences into positive values such as:

$$\overline{\varepsilon_n}^{(i)}(i = 1,2,\dots,l)$$

The positive and negative signs associated with errors can be represented as *s(i)*. We will further describe it in the next section. Hence it is obvious to get the second approximate solution:

$$\overline{v_i}^{(2)} = \overline{v_i}^{(1)} + \overline{\varepsilon_i}^{(1)}(i = 1,2,\dots,n)$$

By adding the first approximate solution and the first error approximate solution.
We observe that $v_i^{-(2)}$ is a more close approximate solution than that of first approximation solution $v_i^{-(1)}$ (i=1,2,…,l) to the exact solution v_i (i=1,2,…,n).

Similarly, the second error sequences obtained by subtracting actual input sequences to a second approximation solution: $\varepsilon_i^{(2)}=v_i^{(1)} - \overline{v}_i^{(2)}$ (i=1,2,…,n), which is known as the second error.

By using the same method as defined above, we can get the second error approximate solution $\varepsilon_i^{(2)}$ (i=1,2,…,n) of the second error $\varepsilon_i^{(2)}$ and its predicted values such as:

$$\overline{\varepsilon_{n+1}}^{(2)}, \overline{\varepsilon_{n+2}}^{(2)}, \dots, \overline{\varepsilon_{n+l}}^{(2)}.$$

The third approximation solution can be obtained as:

$$\overline{v_i}^{(3)} = \overline{vi}^{(2)} - \overline{\varepsilon i}^{(2)}(i = 1,2,\dots,n).$$

Therefore, by continuing the above multiple error iterative process, we get the predicted values closer to the exact values v_i (i = 1,2,…,l). Similarly, we get the kth approximate solution

$$\overline{v_i}^{(k)} = \overline{v_i}^{(k-1)} - \overline{\varepsilon_i}^{(k-1)}$$

by using the input sequences $v_i^{(k)}$ (i=1,2,…,l, k=1,2,…,m) which will be considered as a more exact approximate solution than $v_i^{(k-1)}$. Its predicted solution $\underline{v_{(n-1)}}^{(k)}$ (i = 1,2,…,n , k = 1,2,…,m) can also be determined. The kth input error sequence $\varepsilon_i^{(k)}$ (i=1,2,…,n , k=1,2,…,m) is used to determine the error approximate solution sequences $\varepsilon_i^{(k)}$ (i=1,2,… ,n , k=1,2,…,m).

2.2. Basic Theorem

The following theorem describes how multiple iterative analyses of residual errors obtained through mathematical modeling can significantly improve the predictive accuracy of software vulnerability prediction models and help to achieve expected results.

Theorem 1. Assume that a known data sequence v_i (i=1,2,…,l), which can be determined by function *VDM* = $f(v_1,v_2,\dots,v_n , p_1,p_2,\dots,p_r, t)$. By using our proposed technique (HPEIAM), *mth* predictive value $\overline{v_{n+i}}^{(m)}$ can be obtained which is more close to the exact solution v_{n+1} (i=1,2,…,l) than $\underline{v_{n+i}}^{(1)}$.

Proof. For any value n, the following common inequalities exist:

$0 \le | v_n - v_n^{(m)} | = | \varepsilon_n^{(m)} | \le | \underline{\varepsilon_n}^{(m-1)}) | \le \dots \le | \varepsilon_n^{(1)} | = | v_n - \underline{v_n}^{(1)} |$

Therefore, we consider $M=max\{ | v_i - v_i^{(1)} |, i=1,2,\dots,n\}$ and analyze the error sequence $\overline{\varepsilon_i}^{(k)}$ (i=1,2,...,n , k=1,2,...,m). From the structure of $\varepsilon_n^{(k)}$ and $v_n^{(m)}$, we can see that for any values of n, the following equations hold: $\varepsilon_n^{(1)} = v_n - \underline{v_n}^{(1)}$

$$\varepsilon_n^{(2)} = v_n - \overline{v_n}^{(2)} = v_n - \left(\overline{v_n}^{(1)} + \overline{\varepsilon_n}^{(1)}\right) = \varepsilon_n^{(1)} - \overline{\varepsilon_n}^{(1)}$$

$$\varepsilon_n^{(3)} = v_n - \overline{v_n}^{(3)} = v_n - \left(\overline{v_n}^{(2)} + \overline{\varepsilon_n}^{(2)}\right) = \varepsilon_n^{(2)} - \overline{\varepsilon_n}^{(2)}$$

By continuing the same process we get:

$$\varepsilon_n^{(m)} = \varepsilon_n^{(m-1)} - \overline{\varepsilon}_n^{(m-1)} .$$

Now we prove that Equation 4 is true when m→∞, and $\varepsilon_n^{(m)} \to 0$. Therefore from

$$\varepsilon_n^{(m)} = v_n - \overline{v}_n^{(m)}$$

we found that the solution $\overline{v_{(n+i)}}^{(m)}$ (i=1,2,...,l) is more closer to the exact solution $v_{(n+i)}$(i=1,2,…,l) than $\overline{v_{(n+i)}}^{(1)}$ (i=1,2,…,l). Since $0 \le | \varepsilon_n^{(1)} | = | v_n - \overline{v_n}^{(1)} | \le M$
then we obtain

$$0 \le \left|\varepsilon_n^{(2)}\right| = \left|v_n - \overline{v_n}^{(2)}\right| = \left|\varepsilon_n^{(1)} - \overline{\varepsilon_n}^{(1)}\right| \le \left|\varepsilon_n^{(1)}\right| \le M \tag{4}$$

So the following inequality can be obtained: $0 \le | \varepsilon_n^{(m)} | \le | \varepsilon_n^{(m-1)} | \le \dots \le | \varepsilon_n^{(1)} | \le M$ as defined in Equation (4), when m→∞ and $\varepsilon_n^{(m)} \to 0$.

2.3. Detailed Diagram

The overall HPEIAM prediction method is illustrated through a flow diagram in Fig. 1. The HPEIAM follows the following steps:

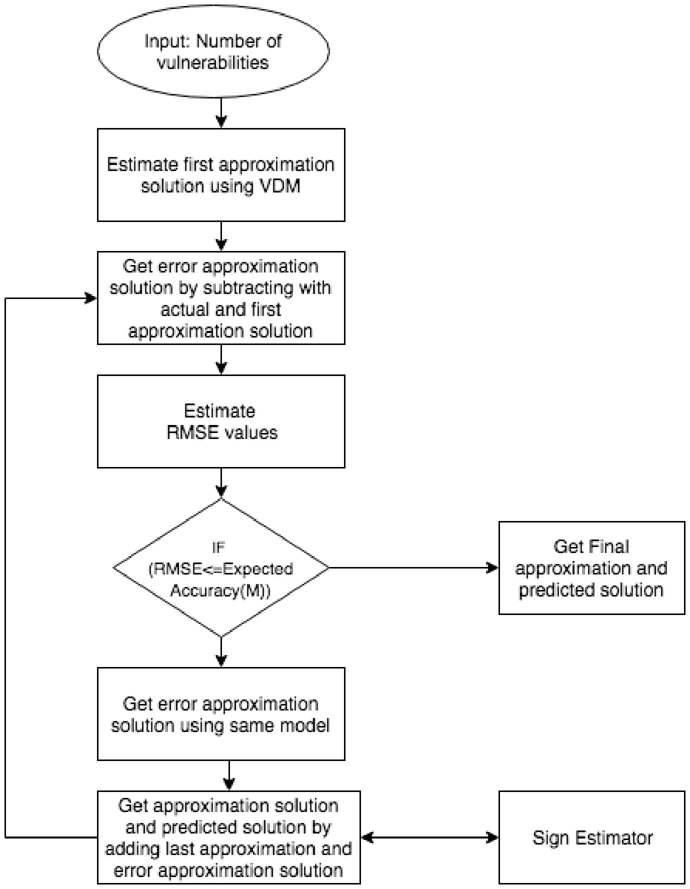
Step 1: Input the number of vulnerabilities in time.
Step 2: Estimate the first approximation solution using the selected VDM.
Step 3: By subtracting actual data with the previous approximation solution, the error of the approximation has been obtained.
Step 4: Estimate the root mean square values (RMSE).
Step 5: Check the Expected accuracy as selected by the user (RMSE<=M). If the condition is satisfied, get final results otherwise continue to the next step.
Step 6: Estimate the error approximation solution and error predicted solution using the same model.
Step 7: Get the next approximation solution and predicted solution by adding the last approximation and error approximation to the solution, which will be a more accurate estimated result than the previous estimation. After getting the next approximate solution, move to step 3 to repeat the same process.

2.4. Algorithm

The proposed method is presented through algorithm 1 which shows the iterative process to improve the accuracy of VDMs as follows:

Algorithm 1: High precision error iterative analysis algorithm

```
Input $v1, v2, \ldots, vn$, $\delta \geq 0$
Create a model: $VDM = f(v_1, v_2, \ldots, v_n, p_1, p_2, \ldots, p_r, t)$
Get the approximation solution: $v_i^{(1)}$ $(i=1,2,\ldots,l)$
Get error values: $\varepsilon_i^{(1)} = v_i - v_i^{(1)}$ $(i=1,2,\ldots,n)$
1. For $i = 1$ to m do
2.   $VDM = f(\varepsilon_1^{(i)}, \varepsilon_2^{(i)}, \ldots, \varepsilon_n^{(i)}, p_1^{(i)}, p_2^{(i)}, \ldots, p_r^{(i)}, t)$
3.   If $|\varepsilon^{(i)}| = \{\sum_{j-1} | vj - \bar{vj}^{(i)} |\} \leq \delta$ then
4.     Get the predicted solution
5.     $v_{n+i}^{(m)} = v_{n+i}^{(m-1)} + s(s)\varepsilon_{n+i}^{(m-1)}$
6.     Exit     n
7.   Else of for some other condition then
7.     Repeat for loop
8.   end if
9. end for
```



Input: Number of vulnerabilities
Estimate first approximation solution using VDM
Get error approximation solution by subtracting with actual and first approximation solution
Estimate RMSE values
IF (RMSE<=Expected Accuracy(M))
Get Final approximation and predicted solution
Get error approximation solution using same model
Get approximation solution and predicted solution by adding last approximation and error approximation solution
Sign Estimator

Fig. 1. Overview of HPEIAM prediction method

2.5. Residual error sign estimation (*s(i)*)

The above HPEIAM method improves the prediction accuracy of any model and gives an optimal solution. The residual errors are responsible for the prediction accuracy of error approximation and the signs *(s(i))* associated with these errors of the given model. VDMs cannot deal with directly residual errors because of their randomly fluctuated positive and negative signals. Therefore, we used an artificial neural network (ANN) to estimate the signs of residual errors. Different types of neural architectures are available. However, a multilayer backpropagation (BP) neural network is the most widely used. A BP network typically consists of three or more layers: an input layer, an output layer, and at least one hidden layer. The structure of the ANN sign estimator is shown in Fig 2.

To predict the signs *(s(i))* of predicted errors $\varepsilon_n^{(i)}$, we have used the two-state ANN model. For that, we introduced a dummy variable d(i) to indicate the sign of ith error. Assume that the sign of *ith* error is negative, then d(i)=0, otherwise it is 1. Then we set up an ANN model by using the values of d(n − 1) and d(n) to estimate the values of d(n + 1).

The characteristic equation for the sign of *ith* error, *s(i)*, is as follows:

$$s(i) = \begin{cases} +1 \text{ } if \text{ d(i)=1} \\ -1 \text{ } if \text{ d(i)=0} \end{cases} \quad i = 1,2,3, \ldots, n$$

According to the above-illustrated equation, the sign of estimated error can be predicted by using the ANN sign estimator. The actual estimated values can be shown as:

$$\overline{v_{n+i}}^{(m)} = \overline{v_{n+i}}^{(m-1)} + s(i)\overline{\varepsilon_{n+1}}^{(m-1)} (i = 1,2, \ldots, l)$$

Next, we will proceed to the software vulnerability prediction to examine the accuracy of our proposed method by using different parametric software vulnerability discovery models (VDMs).

3. NUMERICAL ILLUSTRATION

To demonstrate the effectiveness of the proposed method, we use four different statistical vulnerability discovery models as shown in Table 1. The HPEIAM technique is applied to every model to get a more accurate and optimal solution. To fit the vulnerability data to the models, the parameters are chosen in such a way that the sum of squared error is minimized such as Such as, if a model has parameters A, B, and C, they are optimized using the equation below:

$$A, B, C \,|\, min(\sum t = 1\, (v(t) - v(t)^2)$$

3.1. Data Description

To validate our model results, we collected the vulnerability datasets from the National

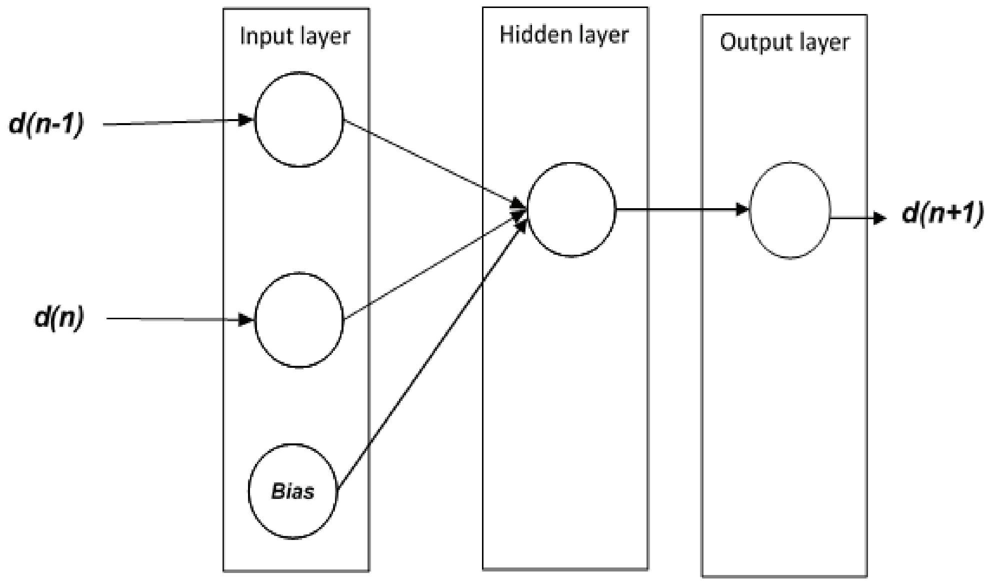


Fig. 2. Graphical representation of ANN sign estimator

Table 1. Parametric software reliability growth models

Models Name	Model Function	Description
Exponential Model Rescorla	$V(t) = N \times (1 - e^{-at})$	The number of vulnerabilities discovered at time t decays exponentially with the time
Logarithmic Model Poisson	$V(t) = a \times LN(1 + b \times t)$	It shows the total number of vulnerabilities as a logarithmic growth function
Alhazmi-Malaiya Logistic Model	$V(t) = \frac{B}{B \times C \times e^{ABt} + 1}$	It is based on capturing the underlying process of vulnerability discovery and the rate of vulnerability depends on two factors.
Weibull Model	$\mathrm{V}\,(t) = \gamma \left\{1 - e^{-(\frac{t}{\beta})^{\alpha}}\right\}$	It assumes that the vulnerability discovery rate varies according to the Weibull probability distribution function.

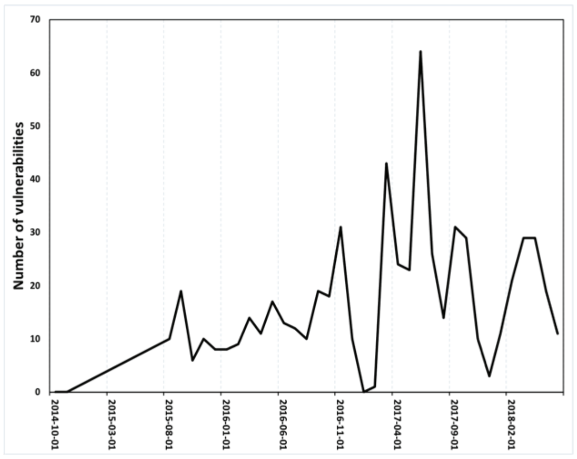
Vulnerability Database (NVD) (http://nvd.nist.gov) managed by the National Institute of Standard and Technology (NIST). Experts at NIST have analyzed the vulnerabilities reported to the NVD and assigned proper attributes to the defects before the data entries [26].

Therefore, it is considered a high-quality database, which has been used by several researchers. In total, we collected four recent versions of different vulnerability datasets: Windows 10, Android 7.0, Internet Explorer 11, and Firefox 20. The collected vulnerabilities for each dataset started from the first day of the release, or the earliest availability. These vulnerability datasets represent the major categories of software systems: operating systems, Web browsers, and Android applications. We collected the vulnerabilities of each application starting from the first day of the release of the earliest available data in NVD. We aggregated all vulnerabilities for each application over a monthly period. Table 2 shows the statistics of all vulnerability datasets. Fig 3. shows the number of vulnerabilities found in each month for all datasets.

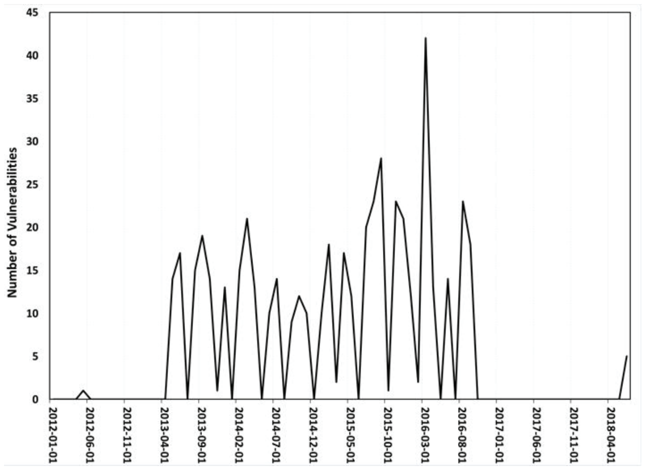
4. EXPERIMENTAL EVALUATION AND PERFORMANCE ANALYSIS

In our analysis, we have used datasets of four different software as shown in Fig 3. Based on our technique, we divide the data into two parts. The first part of the dataset is always used to get expected accuracy. The root means square error (RMSE) criterion is used to get the expected accuracy of the model based on our proposed technique for every iteration. We specify the expected accuracy value for every model as 10. The process will be stopped if the RMSE value got less value than the expected accuracy. Every model has been iterated repeatedly based on our proposed technique until expected. accuracy values get less than 10. The accuracy values of different models and their improved results for every iteration for different datasets are shown in Fig 4.

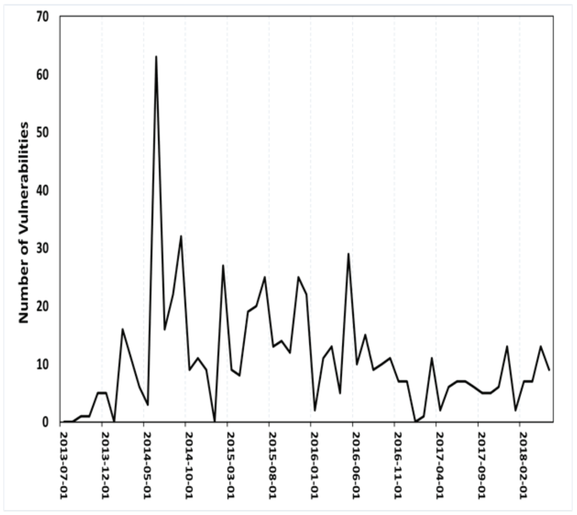
Fig 4. the bar values which show less than 10 RMSE values are considered to reach the expected accuracy and the iteration process is stopped. The second part of the dataset is used to evaluate the predictive power of the proposed technique. The sum of squared errors (SSE) criterion is selected to check the predictive capability of the proposed technique. Most of the VDMs have been evaluated using their fitting capability, while visually most of the models appear to fit well, but their predictive capability is considered non-satisfactory. The primary use of VDMs is to predict future trends based on the available data, rather than assessing the past data behavior. Therefore, we have used the last 10 data points of every dataset to evaluate the predictive capability of models and also checked



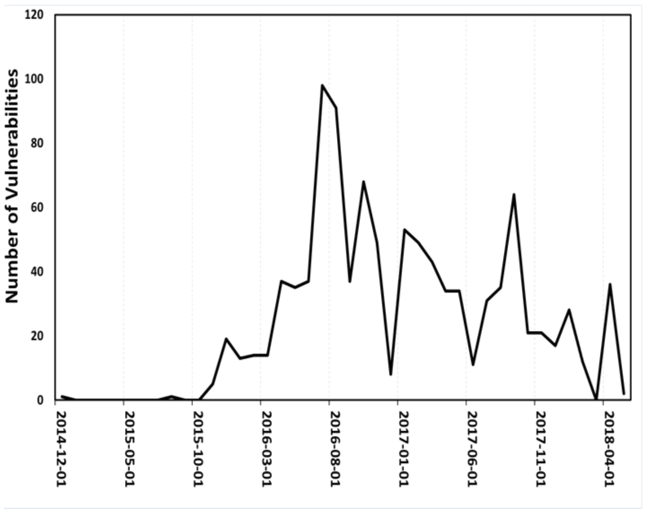
(a) Windows 10



(b) Firefox 7.0



(c) Internet explorer 11.0



(d) Android 20.0

Fig. 3. Number of vulnerabilities along with the calendar time

Table 2. Vulnerability datasets

Datasets	Data Collection period	Number of Vulnerabilities
Windows 10	Aug 2015-Jun-2018	613
Firefox 20.0	Dec 2014-Jun-2018	1018
Internet Explorer 11.0	Sep 2013-Jun-2018	640
Android 20.0	May 2012-Jun-2018	502

Table 3. Comparison of predictive results of all selected models using the HPEIAM technique for different datasets

	Fitting power (RMSE values)				Predicted power (SSE values)			
	Android 7.0	Firefox 20	Explorer 11	Windows 10	Android 7.0	Firefox 20	Explorer 11	Windows 10
Exponential model	26.57	10.17	10.91	13.21	3155.01	158.35	183.39	1107.60 3803
Exponential model iteration-1	14.76	6.18	7.53	9.92	2979.17	145.21	112.16	534.94
Exponential model iteration-2	9.97	-	-	-	1893.8	-	-	-
Logarithmic model	27.8	11.11	10.92	13.06	3688.37	1228.58	211.09	977.623 1669
Logarithmic model iteration-1	17.51	6	7.68	9.6	1701.18	680.92	86.81	463.35
Logarithmic model iteration-2	14.61	-	-	-	753.99	-	-	-
Logarithmic model iteration-3	10	-	-	-	282.22	-	-	-
AML model	16.29	10.822	11.01	12.45	4572.72	630.91	536.87	1049.02 935
AML model iteration-1	11.66	6.71	8.52	8.93	2928.82	250.81	488.52	773.64
AML model iteration-2	6.58	-	-	-	2185.46	-	-	-
Weibull model	16.52	11.22	15.36	11.83	4436.52	1608.73	271.79	1530.85
Weibull model iteration-1	11.9	6.288	7.24	8.41	2622	809.78	231.52	791.35
Weibull model iteration-2	6.78	-	-	-	2165	-	-	-

the predictive capability after applying the proposed technique on every VDM. Table 3 shows the fitted values obtained from the VDMs such as Exponential, Logarithmic, AML, and Weibull models and their improved results using the proposed technique by considering different datasets. The expected accuracy is selected based on the appropriate RMSE value and this experiment is taken as 10. From Table 3, it is shown that when HPEIAM is applied to an exponential model, it provides the best fitting and predictive results for different datasets. The exponential model shows different RMSE values for Android 7.0, Firefox 20, Internet Explorer 11, and Windows 10 such as 26.57, 10.17, 13.21, and 13.21 values respectively. As the expected accuracy value is selected as 10 so we iterate the process and apply the proposed technique. For Firefox 20, Internet Explorer 11, and Windows 10, the expected accuracy is achieved after the first iteration such as 6.18, 7.53, and 9.92 respectively. For Android 7.0, the expected accuracy is obtained after the second iteration as 9.97. The predictive accuracy also improved with every iteration such as for Android, it improves from 3155.01 to 2979.17 and then changed to 1893.8, which is the lowest SSE value. The predictive accuracy for Firefox 20, Internet Explorer 11, and Windows 10, also improve from 158.35, 183.39, and 1107.60 to 145.21, 112.16,

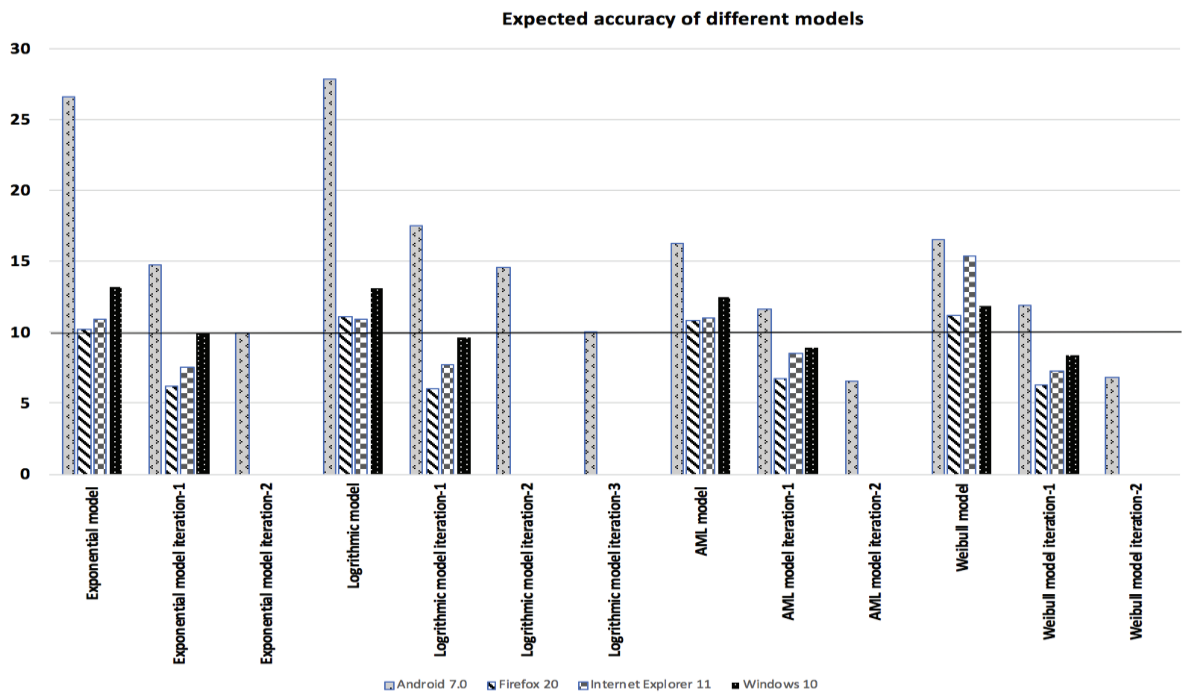


Fig. 4. Illustration of Predicted Errors for four VDMs using Android 7.0 dataset

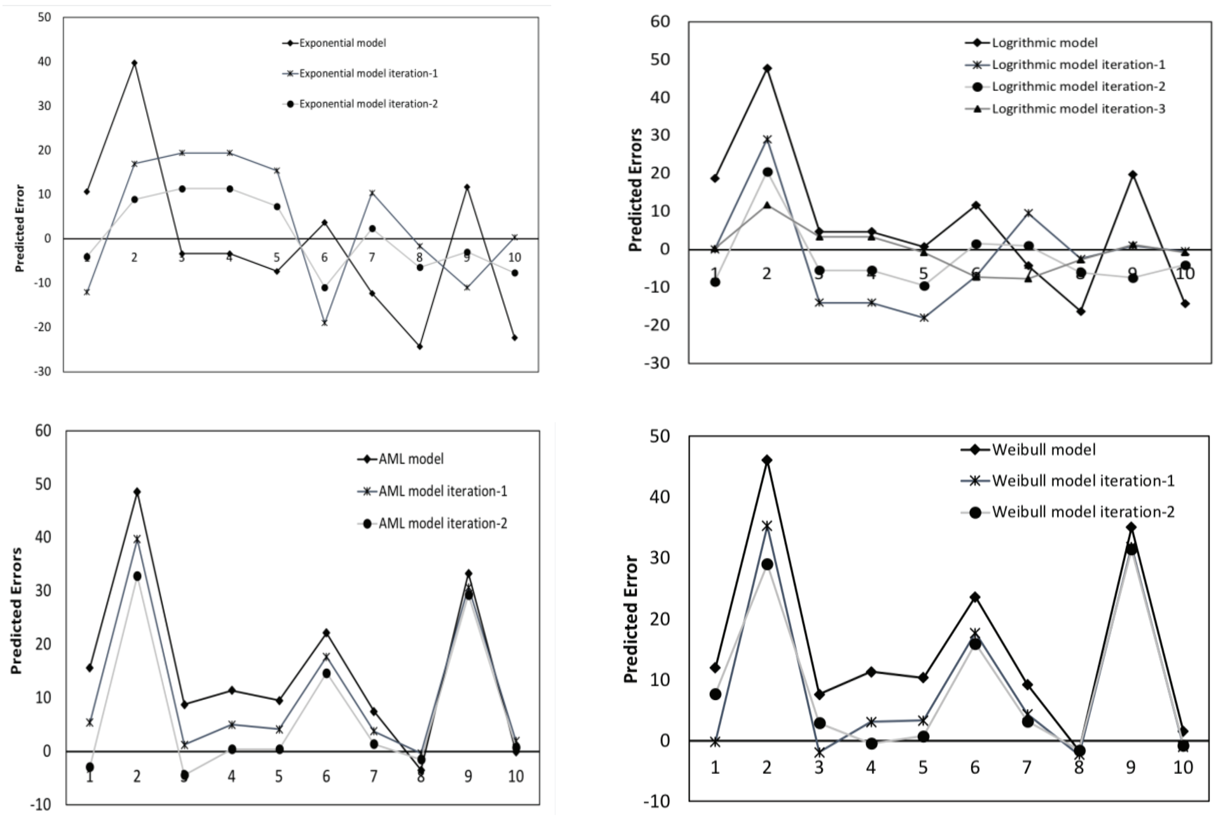
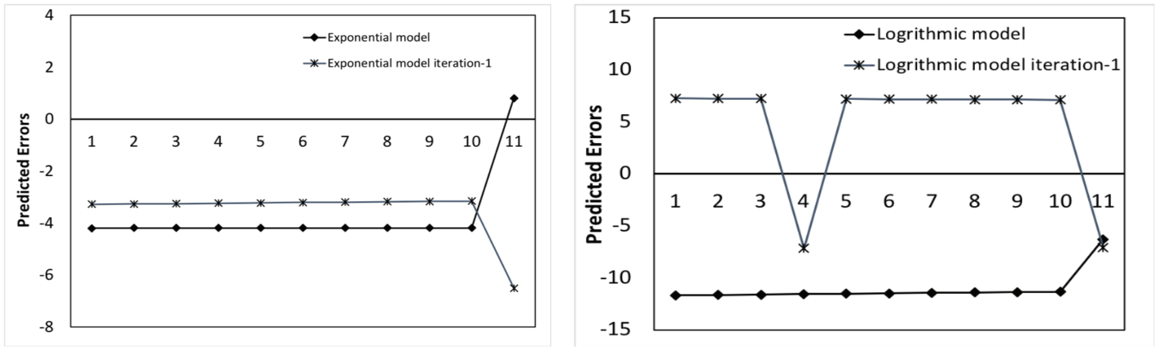
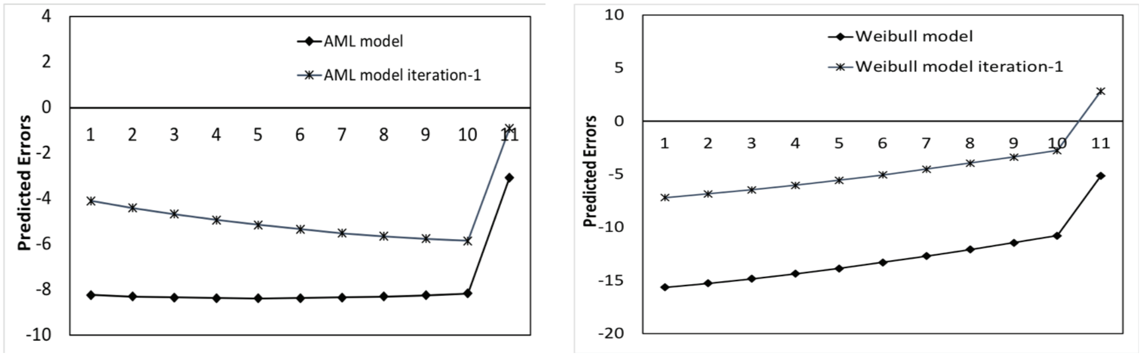


Fig. 5. Illustration of Predicted Errors for four VDMs using Android 7.0 dataset

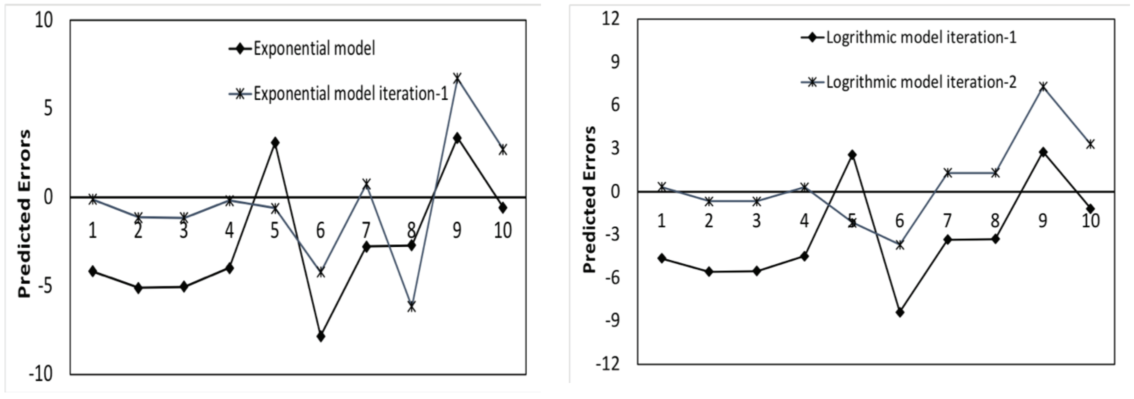


(a) (b)

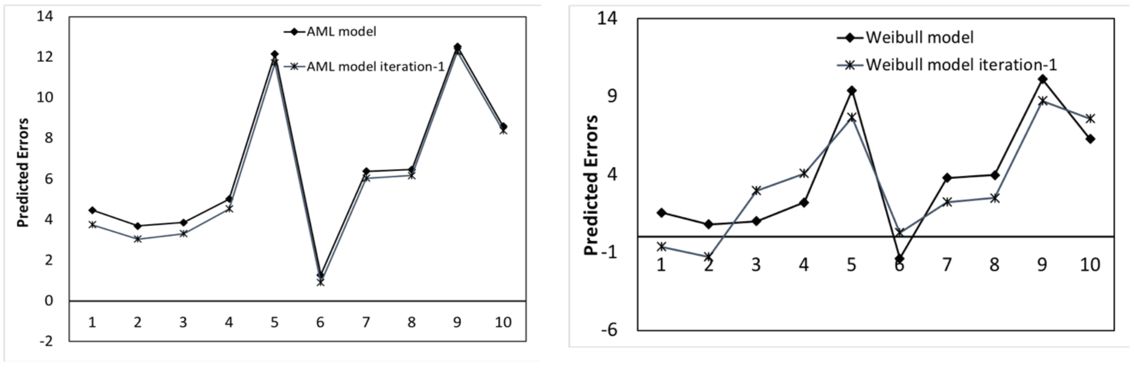


(c) (d)

Fig. 6. Illustration of Predicted Errors for four VDMs using Firefox 20 dataset



(a) (b)



(c) (d)

Fig. 7. Illustration of Predicted Errors for four VDMs using Firefox 20 dataset

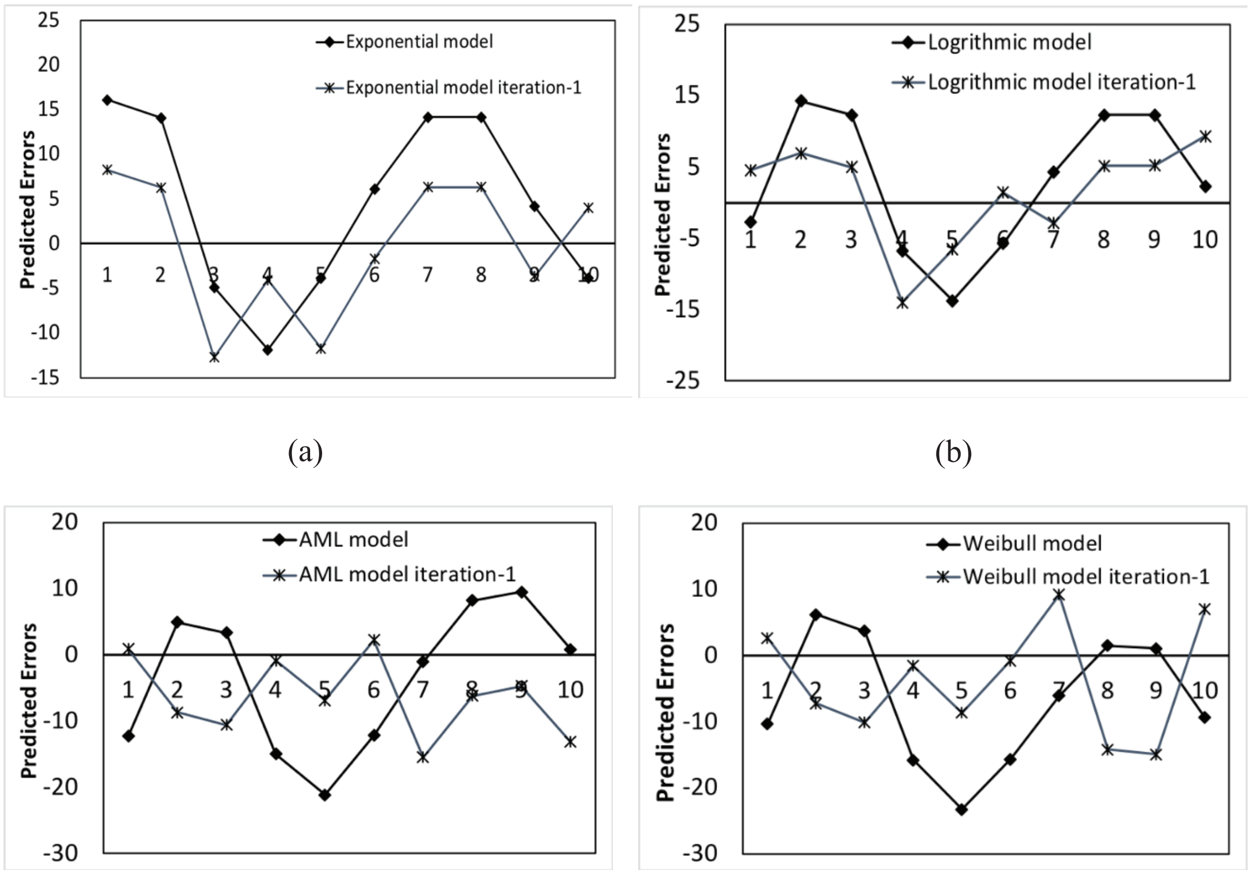


Fig. 8. Illustration of Predicted Errors for four VDMs using Windows 10 dataset

and 534.94 respectively. The predictive errors are illustrated for every dataset in Fig 5(a), 6(a), 7(a), and 8(a). When the proposed technique is applied to the Logarithmic model, the expected accuracy is obtained after the first iteration for Firefox 20, Internet Explorer 11, and Windows 10 such as 6.00, 7.68, and 9.60 respectively. For Android 7.0, the expected accuracy is obtained after the second iteration such as 10.00. The predictive accuracy also improves with every iteration such as for Android 7.0; it improves from 3688.37 to 1701.18 in the first iteration and then changed from 7753.99 to 282.22 in the third iteration, which is the lowest SSE value. The predictive errors of the improved Logarithmic model for every dataset are illustrated in Fig 5(b), 6(b), 7(b), and 8(b). For the AML model, the expected accuracy is obtained after the second iteration for Android 7.0 such as 6.58. For Firefox 20, Internet Explorer 11, and Windows 10, the expected accuracy is achieved after the first iteration such as 6.71, 8.52, and 8.93 respectively. The predictive accuracy shows improved results such as for Android 7.0; it improves from 4572.72 to 2185.46, which is the lowest predictive SSE value. The predictive accuracy for Firefox 20, Internet Explorer 11, and Windows 10 such as 250.81, 488.52, and 773.64 respectively. The predictive errors of the improved AML model for every dataset are illustrated in Fig 5(c), 6(c), 7(c), and 8(c).

Similarly, for the Weibull model, the expected accuracy is obtained after the second iteration for Android 7.0 such as 6.58. For Firefox 20, Internet Explorer 11, and Windows 10, the expected accuracy is achieved after the first iteration such as 6.288, 7.24, and 8.41 respectively. The predictive accuracy also shows improved results such as for Android 7.0, it improves from 4436.52 to 2165.00, which is the lowest predictive SSE value. The predictive accuracy for Firefox 20, Internet Explorer 11, and Windows 10 such as 809.78, 231.52, and 791.35

respectively. The predictive errors of the improved Weibull model for every dataset are illustrated in Fig 5(d), 6(d), 7(d), and 8(d).

The ANN sign estimator contains a tapped delay line from 1 to 4 and uses eight neurons in the hidden layer for every dataset. The network has tapped the delay line with a maximum delay of 4, begins by predicting the fifth value of the input series (sign). The last 10 data points of every dataset are used as a testing set and the remaining part is used for training the network. The output results are combined with the predicted error values of different VDMs to get the expected approximated solution. The fitting RMSE values of the proposed method improve with every iteration in each data interval. Therefore, it is found that the proposed technique improves the accuracy of every model using any data interval and the user can use it to get the model's accuracy up to an expected level.

5. CONCLUSIONS

In this paper, we have used multiple errors iterative accuracy methods and presented a detailed analysis of the optimization process by progressively acquiring optimal solutions using different VDMs. Experiments are used to demonstrate that the proposed method can overcome the shortcomings of current vulnerability discovery models such as the assumptions made by the models and also the dependency of models on the shapes or skewness of data. Moreover, the experimental results show that an expected accuracy which is specified as 10 has been achieved by optimizing the error data iteratively, and thus MEIAM provides more accurate predictions of the future trend of the number of vulnerabilities. The proposed model brought the lowest expected accuracy value for the exponential model such as android 9.97, Firefox 6.18, Explorer 7.53, and Windows 9.92. For the logarithmic model such as android 10, Firefox 6, Explorer 7.68, and windows got a 9.6 expected accuracy level. For the AML model such as android 6.58, Firefox 6.71, Explorer 8.52, and windows got 8.93. Similarly, the Weibull models also show improve results such as android 6.78, Firefox 6.28, Explorer 7.2, and 8.1.

Moreover, the method presented in this paper offers a universal optimization process rather than the limited applicability of existing vulnerability discovery models. Furthermore, a comparison of the applied technique results on different VDMs has also been provided regarding different criteria values (RMSE and SSE) on four different datasets. Experimental results show that MEIAM techniques can effectively enhance and improve the performance of each VDM by providing better accuracy and predictive power.

6. REFERENCES

1. C. P. Pfleeger, and S. L. Pfleeger, *Security in computing:* Prentice Hall Professional Technical Reference (2002).
2. V. H. Nguyen and L. M. S. Tran, Predicting vulnerable software components with dependency graphs, in *Proceedings of the 6th International Workshop on Security Measurements and Metrics,* (2010)
3. S. Rahimi, and M. Zargham, Vulnerability scrying method for software vulnerability discovery prediction without a vulnerability database, *IEEE Transactions on Reliability,* 62: 395-407 (2013).
4. R. Scandariato, J. Walden, A. Hovsepyan, and W. Joosen, Predicting vulnerable software components via text mining, *IEEE Transactions on Software Engineering,* 40: 993-1006 (2014).
5. J. A. Harer, L. Y. Kim, R. L. Russell, O. Ozdemir, L. R. Kosta, A. Rangamani, et al., Automated software vulnerability detection with machine learning, *arXiv preprint arXiv*:1803.04497 (2018).
6. Y. Shin, and L. Williams, Can traditional fault prediction models be used for vulnerability prediction?," *Empirical Software Engineering*, 18: 25-59 (2013).
7. G. Jabeen, L. Ping, J. Akram, and A. A. Shah, An Integrated Software Vulnerability Discovery Model based on Artificial Neural Network, in *SEKE*: 349-458 (2019).
8. G. Jabeen, and L. Ping, A Unified Measurable Software Trustworthy Model Based on Vulnerability Loss Speed Index, in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 18-25 (2019).
9. R. Anderson, Security in open versus closed systems—the dance of Boltzmann, Coase and Moore, Technical report, Cambridge University, England (2002).
10. E. Rescorla, Is finding security holes a good idea?,

IEEE Security & Privacy, 3: 14-19 (2005).

11. J. D. Musa and K. Okumoto, A logarithmic Poisson execution time model for software reliability measurement, in *Proceedings of the 7th international conference on Software engineering*, 230-238 (1984).
12. S. Rahimi, *Security vulnerabilities: Discovery, prediction, effect, and mitigation:* Southern Illinois University at Carbondale (2013).
13. O. H. Alhazmi and Y. K. Malaiya, Quantitative vulnerability assessment of systems software, in *Annual Reliability and Maintainability Symposium,* 2005. Proceedings, 615-620 (2005).
14. O. H. Alhazmi, Y. K. Malaiya, and I. Ray Measuring, analyzing, and predicting security vulnerabilities in software systems, *Computers & Security*, 26: 219-228 (2007).
15. O. H. Alhazmi and Y. K. Malaiya, Measuring and enhancing prediction capabilities of vulnerability discovery models for Apache and IIS HTTP servers," in *2006 17th International Symposium on Software Reliability Engineering* 343-352 (2006).
16. O. H. Alhazmi and Y. K. Malaiya, "Application of vulnerability discovery models to major operating systems, *IEEE Transactions on Reliability*, 57: 14-22 (2008).
17. K. Chen, D. Feng, P. Su, C. Nie, and X. Zhang, Multicycle vulnerability discovery model for prediction, *Journal of Software,* 21: 2367-2375 (2010).
18. H. Joh and Y. K. Malaiya, "Modeling skewness in vulnerability discovery, *Quality, and Reliability Engineering International*, 30: 1445-1459 (2014).
19. R. Johnston, S. Sarkani, T. Mazzuchi, T. Holzer, and T. Eveleigh, Multivariate models using MCMCBayes for web-browser vulnerability discovery, *Reliability Engineering & System Safety,* 176: 52-61 (2018).
20. R. Johnston, S. Sarkani, T. Mazzuchi, T. Holzer, and T. Eveleigh, Bayesian-model averaging using MCMCBayes for web-browser vulnerability discovery, *Reliability Engineering & System Safety,* 183: 341-359 (2019).
21. Y. Movahedi, M. Cukier, and I. Gashi, Vulnerability prediction capability: A comparison between vulnerability discovery models and neural network models, *Computers & Security*, 87: 101596 (2019).
22. P. Kapur, V. S. Yadavali, and A. Shrivastava, A comparative study of vulnerability discovery modeling and software reliability growth modeling, min *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management* (ABLAZE), 246-251 (2015).
23. H. Joh and Y. K. Malaiya, Periodicity in software vulnerability discovery, patching and exploitation, *International Journal of Information Security*, 16: 673-690 (2017).
24. R. Sharma and R. Singh, Vulnerability Discovery in Open-and Closed-Source Software: A New Paradigm," in *Software Engineering,* ed: Springer, 2019, pp. 533-539.
25. B. Liu, L. Shi, Z. Cai, and M. Li, Software vulnerability discovery techniques: A survey," in *2012 fourth international conference on multimedia information networking and security,* 152-156(2012).
26. S. H. Houmb, V. N. Franqueira, and E. A. Engum Quantifying security risk level from CVSS estimates of frequency and impact, *Journal of Systems and Software,* 83: 1622-1634 (2010).