An Integrated Software Vulnerability Discovery Model based on Artificial Neural Network

Gul Jabeen*, Junaid Akram*, Luo Ping*, Akber Aman Shah†
*State Key Laboratory of Information Security, School of Software Engineering, Tsinghua University China.
Email: [jgl14,znd15]@mails.tsinghua.edu.cn
Email: luop@mail.tsinghua.edu.cn
†School of Economics and Management, University of Chinese Academy of Science, Beijing, China.
Email: akberaman@hotmail.com

***Abstract*—Quantitative approaches for software security are needed for effective testing, maintenance and risk assessment of software systems. Vulnerabilities that are present in a software system after its release represent a great risk. Vulnerability discovery models (VDMs) have been proposed to model vulnerability discovery and have has been fined to vulnerability data against calendar time. Though, these models have various shortcomings include changes and development of VDMs for different dataset due to diverse approaches and assumptions in their analytical formulation. There is a clear need for an intensive investigation on these models to enhance predictive accuracy of existing VDMs and adopt the actual behavior of software vulnerabilities which were not modeled previously. This study proposed an integrated model to predict a number of software vulnerabilities by hybridizing the Multi-Layer Perceptron (MLP) artifical neural network and Vulnerability Discovery Models. The proposed model is also widely applicable across various vulnerability datasets and models due to its input diversity by providing improved fitting and predictive accuracy. Further, the experimental results show that this model not only retained the properties of traditional parametric VDM models as well as MLP's good nonlinear mapping ability and useful generalization.**

keyword Vulnerability discovery model, Artificial neural network, Integerated model, Security, Multi-Layer Perceptron neural network

I. Introduction

With the development of Internet technology, software vulnerabilities have increased rapidly and caused an increasing number of serious security issues. A critical vulnerability provides an attacker with the ability to access full control of a software [1] [2]. Therefore, a quantitative characterization of the vulnerability discovery rates is necessary to assess the risks associated with the product.

VDMs are the specialization of software reliability growth models (SRGM) that focus on security errors. Nonetheless, most previous studies reveal that these models are based on SRGMs, which are not empirically enough to deal with vulnerabilities of software [3] [4]. VDMs intensive evaluating the security profile of software as compared to the vulnerability predications models because they are focusing only on vulnerable components of software [5] [6] [7] [8] [9] [10]. Software vendors and customers are using accurate VDMs to understand security trends and patch management. However, current research studies are focusing to develop further improved VDMs to maximize their predictive accuracy. First VDM model (thermodynamic model) was proposed by Anderson, which was based on SRGMs. However, this model is considered as worst in terms of fitting empirical datasets.

Similarly, various statistical models are used either attempting to capture the underlying processes or applying the principles used in other fields of science to discover vulnerabilities. These models are classified into two categories: time-based and effort-based. Time-based models measure the total number of vulnerabilities over time while effort-based models count vulnerabilities based on testing efforts. The current study focuses on time-based models, which still need further investigation to enhance fitting and predictive accuracy of VDMs. Numerous time-based VDMs model are proposed in previous studies i.e., linear model [11], Rescorla's exponential (RE) models [12], Alhazmi and Malaiyas logistic (AML) [3] and multiversion models [13], Weibull model [14], Younis folded (YF) model, Kapur's logistic model [15], Anand and Bhatt's hump-shaped model [16], and Anand multi-version VDM [17], and Sharma's changing point model [18] to model the rate at which vulnerabilities discovered.

All of the above models are proposed to obtain better fitting and predicting models for different vulnerability datasets but most of them are against of certain vulnerability datasets. These models try to get a better model under a certain condition, but it cannot give good results with every vulnerability dataset. Nonetheless, these models have numerous shortcomings as discussed below:

1) VDMs uses different approaches with respect to the assumptions and parameters. In this regards, VDMs can predict different vulnerability discovery rates by using the same dataset.
2) A single software vulnerability discovery model premised on the constant assumptions and can predict different discovery rates using the same data.

As the neural network can be applied for a variety of areas because without an assumption similar to traditional models, the used model is more universal. In this study, we proposed an integrated approach to solve the aforementioned key challenges of traditional software vulnerability discovery models. The purpose of the study is to provide a flexible

DOI reference number: 10.18293/SEKE2019-168

TABLE I: Software Vulnerability Discovery models used in evaluation

Models Name	Model function	Description
Rescorla Exponential (RE) model [19]	$V(t) = N * (1 - e^{-at})$	Exponential model is proposed to fit real data. The number of vulnerabilities discovered at time t decays exponentially with the time.
Alhazmi-malaiya Logistic (AML) Model [3]	$V(t) = \frac{B}{B*C*e^{ABt}+1}$	It is based on the capturing the underlying process of vulnerability discovery and the rate of vulnerability depends on two factors.
Weibull model [14]	$V(t) = \gamma\{1 - e^{-(\frac{t}{\beta})^{\alpha}}\}$	It assumes that the vulnerability discovery rate varies according weibull probability distribution function.
Younis Folded (YF) Model [20]	$V(t) = \frac{\gamma}{2}[erf(\frac{t-\tau}{\sqrt{2\alpha}}) + erf(\frac{t+\tau}{\sqrt{2\alpha}})]$	It shows vulnerability discovery model based on the folded normal distribution.

method with an accurate representation of data frequencies regardless of their different vulnerabilities discovery rates. An integrated model takes the assistance of ANN (MLP), which serve as a non-linear hybrid system of traditional VDMs. The classical software vulnerability discovery models are used as the base models and the MLP technique is used to combine the results of base models, which helps to eliminate the influence of external parameters and other anomalies of VDMs, arise due to the assumptions made by these parametric models. The proposed model can take the advantage of classical VDMS in an application domain, as in the linear combination model, and the generalization ability of a neural network, and can improve the predictive ability of the software vulnerability assessment models.

The rest of this paper is organized as follows. In Section II, the proposed method is defined in detail. Experimentations and Results analysis are performed in Section III. In section IV, we discuss and highlights the threats to validity. In Section V, we have discussed the related work. Finally, we conclude our work in Section VI.

II. Proposed Model

The proposed method is elaborated in Fig 1. It is divided into two phases: Phase-I and Phase-II. Detailed elaboration of these phases has been presented below:

A. Phase-I

In the proposed integrated model the results of multiple basic vulnerability prediction models serve as input to the MLP neural network. Therefore, the appropriate base models needed to be selected from many of the VDMs. As vulnerabilities identified in software shows three stages of the S-shaped models [3]. The learning phase is started from the release of the system until the onset of sustained growth because of increasing popularity. It is followed by the linear phase when most of the vulnerabilities are to be discovered. The saturation phase is eventually considered as the last stage. Alhazmi and Malaiya defined mathematically the transition point between different phases for the AML model. Therefore S-shaped models are more accurate than non-S-shaped models for vulnerabiltiy discovery process. However, S-shaped models fitting and predictive capability also depends on the skewness in target vulnerabilities, therefore they never give good predictive results for every vulnerability dataset. In this regard, we have used the four most popular VDMs (i.e., models ranging from an exponential to S-shaped models), which are shown in Table 1, with detailed equations.

After selecting the based VDM models, combined results of these models, are used as input to the MLP neural network. We have used the cumulative number of vulnerabilities as a dependent variable, and time which is measured by months, as an independent variable. A set of known cummulative vulnerability data sequences $v_1(t), v_2(t), v_3(t) \cdots v_n(t)$ are used as a input to vulnerability discovery models such as:

$$VDM_i = f(v_1(t), v_2(t), \cdots, v_n(t), a_1, a_2, \cdots, a_r), (i = 1, 2, ..n) \quad (1)$$

where $a_1, a_2, \cdot \cdot, a_n$ are parameters, and t denots specific vulnerability occurence time. We have used it on a monthly basis. The input vulnerability data sequence is denoted as $v_1(t), v_2(t), \cdot \cdot, v_n(t)$ and the future trend vulnerability discovery rates can be written as: $v_{n+1}(t), v_{n+2}(t), v_{n+3}(t) \cdots v_{n+l}(t)$. After applying the input data sequences $v_1(t), v_2(t), \cdot \cdot, v_n(t)$ in each vulnerability discovery model (VDM_i). Here, i shows the specific number of VDMs. However, n denoted the total number of vulnerabilities used to estimate parameters and $n + j$ shows the future predicted values.

The output of first VDM_1 approximation solution/fitted data and its future trend or predicted values are determined as $VDM\_1(fit) = y_1^{(1)}, y_2^{(1)}, y_3^{(1)} .... y_n^{(1)}$ and $VDM\_1(pre) = y_{n+1}^{(1)}, y_{n+2}^{(1)}, y_{n+3}^{(1)} .... y_{n+l}^{(1)}$ respectively.

Same process is repeated for the i number of VDMs, which generate the specific outputs regarding the input vulnerability data sequences. After applying the input data sequences $v_1(t), v_2(t), \cdot \cdot, v_n(t)$ in i^{th} vulnerability discovery models, we determine the output of last VDM approximation solution/fitted data and its predicted values as

$VDM\_i(fit) = y_1^{(i)}, y_2^{(i)}, y_3^{(i)} .... y_n^{(i)}$
and
$VDM\_i(pre) = y_{n+1}^{(i)}, y_{n+2}^{(i)}, y_{n+3}^{(i)} .... y_{n+l}^{(i)}$ respectively.

In the first phase the input variable $v_1, v_2, ...v_n$ are used to get the estimated and predictive results of different VDMs. Assume that the i^{th} number of VDMs have been used to estimate the vulnerabilities and $y^{(i)}$ denotes estimated and predictive outputs of i^{th} models.

B. Phase-II

In the next stage, the results of vulnerability discovery models have been combined by using multi-layer perceptron.

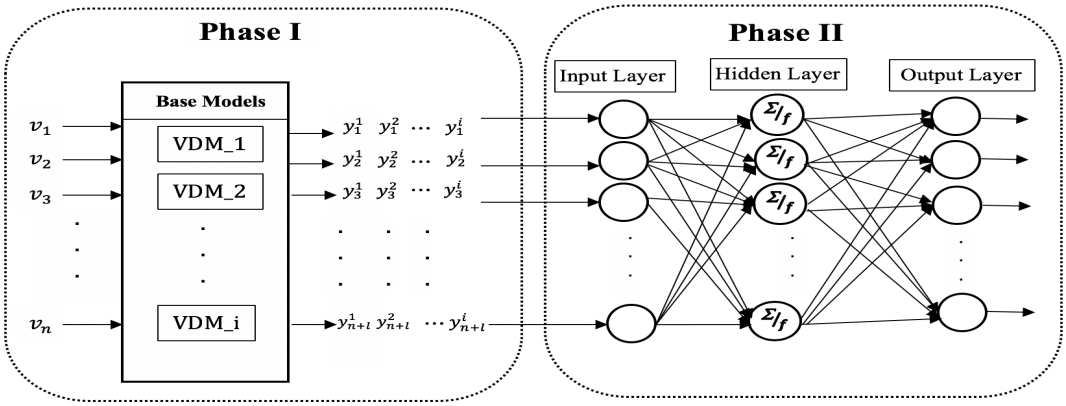

Fig. 1: Detailed diagram of an Integrated VDM model

The MLP is a deep artificial neural network. It is comprised of more than one perceptrons. It works the same way as feed-forward neural network where the back propagation algorithm is in the form of gradient descent function. Almost all algorithms that are applied for MLP training tried to reduce the amount of error by using appropriate functions. In this study, we have used three-layered multilayer perceptron which is composed of an input layer to receive input data a nd an output layer that predicts input, and in between those two layers, an arbitrary number of hidden layers are present that are the true computational engines of MLP. A nonlinear function is associated with each node like a sigmoid function, except for the input nodes. The MLP network learned a specific target function and adjust weights properly using a general method of linear optimization (gradient function). For this, the derivative of the errors function concerning the network weight is measured. The network weights are changed because of error decreases. The square Euclidean distance is used to compute the error between the actual output and the desired output of a network. The following steps are used, to perform experimentation:

1) **Step-1**: The input and output variables are presented first. The vulnerability discovery models outputs from phase 1 are used as input variable ($VDM\_1(fit)$, $VDM\_2(fit), ..., VDM\_i(fit)$) and the actual vulnerability data is used as target variable $v_1(t), v_2(t), v_3(t) \cdot \cdot \cdot v_n(t)$
2) **Step-2**: The entire set of vulnerability data is divided into two parts: training and the testing part. The training dataset is used to train the network for predicting software vulnerabilities. The fitted data of VDMs $y_1^{(i)}, y_2^{(i)}, y_3^{(i)} ....y_n^{(i)}$ are used to train the network. For testing the predicted values of different VDMs $y_{n+1}^{(i)}, y_{n+2}^{(i)}, y_{n+3}^{(i)} ....y_{n+l}^{(i)}$ are used.
3) **Step-3**: The next step in this study is to use 10-fold cross validation method, which divides the data into ten folds. The nine parts are used for training, and the tenth part is used for validation. It is the best validation process to maximize the utilization of vulnerability dataset through repeated resampling of the same dataset randomly.
4) **Step-4**: After dividing training data into 10-folds, the MLP model is applied for training the models.
5) **Step-5**: In this step, the tested data $y_{n+1}^{(i)}, y_{k+2}^{(i)}, y_{n+3}^{(i)} ....y_{n+l}^{(i)}$ of VDMs is applied to get the target predicted values $v_1(t), v_2(t), v_3(t) \cdot \cdot \cdot v_n(t)$. The target values are considered as more closer
6) **Step-6**: The statistical efficacy measures are estimated based on the predicted results for all the selected datasets.

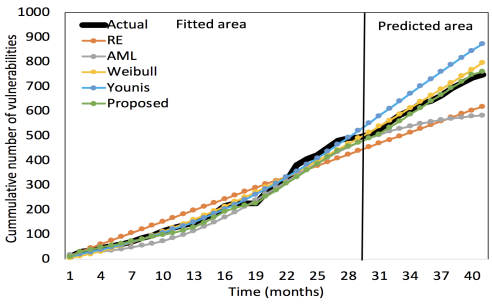

Fig. 2: Models fitting for Window 10

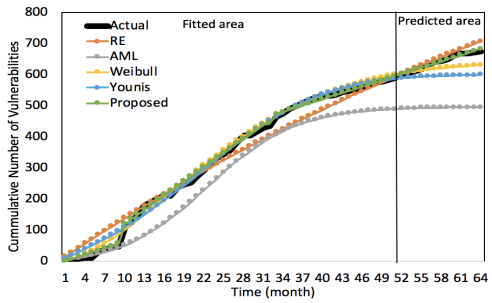

Fig. 3: Models fitting for Internet Explorer

III. Experimentation and Analysis

The proposed model has been validated on the security vulnerability data of two software products namely Microsoft

TABLE II: Comparison of VDM models used in evaluation

Models	Windows 10 ($\chi^2\_critical = 83.6753$)				Internet Explorer ($\chi^2\_critical = 56.9424$)			
	χ^2	$Pvalue$	DF	R^2	χ^2	$Pvalue$	DF	R^2
RE Model	409.5481	7.467E-52	64	0.9810	428.7739	4.57E-66	41	0.9823
AML Model	1686.6183	0.000E+00	64	0.9739	421.3551	1.33E-64	41	0.9815
Weibull Model	109.5576	3.422E-04	64	0.9938	91.3514	1.06E-05	41	0.9941
Younis Model	245.9183	3.906E-23	64	0.9880	138.7848	1.50E-12	41	0.9913
Proposed Model	34.9911	1.0000	64	0.9990	52.8216	0.102092	41	0.9959

Windows 10 and Internet Explorer obtained CVE details[1]. CVE details is a publicaly available CVE security vulnerability database/information source. Furthermore, a set of models have been used in order to evaluate which model is performing best by comparing their overall performance with the proposed model. We have estimated the parameters of the VDMs using R [21] tool.

A. Model fitting and goodness of fit analysis

To measure the goodness-of-fit, researchers always use Pearson's Chi-square (χ^2) and calculate the statistical value of the curve using the following function.

$$\chi^2 = \sum_{t=1}^{n} \frac{(O_t - E_t)^2}{E_t} \tag{2}$$

Where O_t and E_t are the observed and expected samples at time t (t^{th} value of the observed sample); E_t denotes the expected cumulative number of vulnerabilities. If the value of χ^2 of VDM for a specific dataset obtains a value less than the corresponding Chi-critical ($\chi^2\_critical$) value, with the given significant alpha level (0.05) and degree of freedom, it is considered that the model is acceptable. The P-value shows the probability that a statistical value as high as the values obtained by Equation (2) could have occurred by chance. The experimentation data is obtained from the analysis of four VDMs estimated and predicted results. The fitting results of the four VDMs (RE, AML, Weibull and Younis) for both of the software datasets, with there fitted entries based on the χ^2 test P-values.

For Windows 10, the proposed models fit has remarkable improvement than other models in Table II, and the fit is considered very good since the P-value of the χ^2 test is higher than 0.05. Also, the R^2 values are more close to 1 than other single VDMs. From Fig 2, of the fitted vulnerabilities and the predicted vulnerabilities for the selected models, shows clearly that the hybrid approach performing best results than others. Thus, it is required to combine the output results of different models than only depend on single model results. It not only retains the properties of traditional software discovery models but also combines neural network (MLP) good nonlinear mapping ability and useful generalization.

For Internet Explorer, the proposed model also shows good fit than other models, as shown in Table II. The P-value obtained is 0.102092, which is higher than χ^2 test P-value 0.05. Also, the R^2 values are more close to 1 than others. Fig 3, shows that the fitted and predicted curve of the proposed model has better predictive ability among all other models under consideration.

Thus, it is also concluded that the proposed integrated model can produce good results with different types of vulnerability datasets than the single traditional VDMs. It is also clearly visible in Fig 2 and 3 that the integrated model developed by combining four models have better fitting and end-point predictive capability than the models alone.

TABLE III: Average Bias and Average Errors comparison

	Windows 10		Internet Explorer	
	AB	AE	AB	AE
RE	0.2881	0.9990	-1.7984	1.7984
AML	-2.7989	0.0000	-1.5529	1.5529
Weibull	-0.4353	0.0000	0.3348	0.3348
Younis	-0.9040	0.0000	1.5041	1.5041
Proposed	0.0143	0.0000	-0.1472	0.2443

B. Improved predictive capabilities of VDMs

Goodness-of-fit tests often used to assess the applicability of VDMs. The main use of these models is to predict future trends based on the previous data, rather than reviewing the past behavior. Therefore, predictive capability should also be considered important than just model fitting. The estimated final values for each time point produced by the four existing and proposed integrated VDM are compared with the actual number of vulnerabilities to calculate the predicted errors. The prediction time span is selected as long term, which is the main concern of software users to decide whether the select the software for inclusion in a product with a longer lifetime [22]. Therefore, the last 12 months vulnerability data is predicted for both software. The two normalized prediction capability measures [23], average error (AE) and Average bais (AB), as shown in Equation 3 and 4 respectively, are evaluated.

$$AE = \frac{1}{n}\sum_{t=1}^{n} \left|\frac{\Omega_t - \Omega}{\Omega}\right| \tag{3}$$

$$AB = \frac{1}{n}\sum_{t=1}^{n} \frac{\Omega_t - \Omega}{\Omega} \tag{4}$$

In the above equations, n is a total number of time points (in months), and Ω_t is the estimated number of total vulnera-

[1] www.cvedetails.com

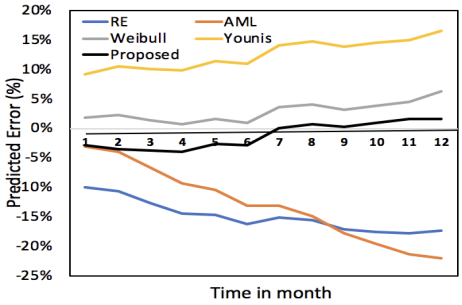

Fig. 4: Predicted Errors of different VDMs for Windows 10

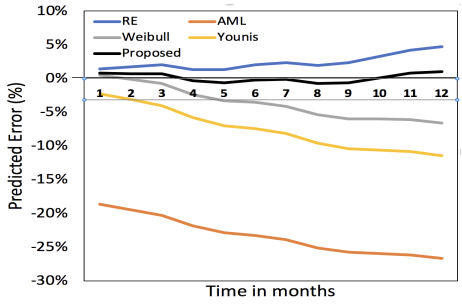

Fig. 5: Predicted Errors of different VDMs for Internet Explorer

bilities at time t, and Ω is the actual number of a total number of vulnerabilities.

The normalized error values $(\frac{\Omega_t - \Omega}{\Omega})$ for windows 10 and Internet Explorer 11 are plotted in Fig 4 and 5. The values of AB and AE are given in Table III. AE always give positive values and AB may give both positive and negative. Fig 4 and 5 shows that the improved models give better predictive results. As in Table 4, the AB and AE values for both yeilds good results after applying HPEIAM method on each of the models. The results yields after applying our technique give lower AB and AE values than the base models itself.

IV. Discussion and Threads to Validity

In this study, we put forward MLP based integrated vulnerability discovery model, and verify the model's superiority over other models with experimentation. This integrated model is comprised of two phases including the first phase, which reflects the linear combination VDMs, and a second neural network phase, which serve as non-parametric modeling of input data sequences. The proposed approach focuses on the relationship of the performance of VDMs with the specific vulnerability discovery datasets. The fitting capabilities of four VDMs along with the chi-square goodness of fit test as well as R-squared metric indicate that the integrated model fits well, for both of the datasets. Besides, the future trend predictive capability can also be estimated using the two main predictive measuring criterion: AB and AE. The results reveal that the integrated model's predictive results also show more accuracy than the other existing VDMs predictive capability. Therefore, we conclude that the proposed method can be applied as a solution for software vulnerability rates estimation problems, outperform competing VDMs investigated in this study.

As the proposed integrated model makes the results of multiple basic vulnerability models (base model) as input to the MLP neural network, the accuracy is in part dependent on the predictive accuracy of these models. Therefore the appropriate base models need to be selected from many software vulnerability models. In this study, only four VDMs have been selected as base models, however to analyzing the characteristics of software vulnerability data, and using the appropriate model selection based on specific criterion is required to get more accurate results. Another limitation is the sample data. Our study analyze publicly reported vulnerabilities without considering unreported data. While this is a common drawback among vulnerability research, overcomning this limitation requires direct contact with software vendors.

V. Related Work

Recently, many VDMs have been proposed by researchers, to model the software vulnerabilities accurately. These models either attempting to capture the underlying processes or applying the principles used in other fields of science to discover vulnerabilities. Each model uses a different approach and with different assumptions and parameters.

Among them, the exponential model is designed to fit the real data [12]. 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 that was first proposed by Poisson [24] and later it is used by Rahimi [25] adjusting fitting of the model to the vulnerabilities of a specific application. Alhazmi and Malaiya proposed a logistic model called AML model in [26] and analyzed AML in [27]. The predictive capabilities were evaluated in [28] and [4] by using a different set of data. Chan et al. proposed a multi-cycle vulnerability discovery model, which helps to extend the scope of existing models [29]. Younis et al. [20] inspected the applicability of Folded VDM and compared it to AML on Win7, OSX 5.0, Apache 2.0 and IE8, and stated that YF was somewhat better than AML. Joh and Malaiya proposed different S-shaped models based on the distribution of Weibull, normal, beta and gamma distribution to evaluate the applicability of the models using different approaches [30]. Kapur et al. proposed models for the prediction of software vulnerabilities and determine whether software reliability growth models can be used to predict the vulnerability discovery process and show good prediction results [31]. Recently, Anand and Bhatt [16] proposed a hump-shaped model to capture the vulnerability exposure pattern due to the attractiveness of a software product in the market using weighted criteria based ranking approach. Joh and Malaiya analyze vulnerability data using the seasonal index and auto-correlation function approaches, which can be used to improve the vulnerability discovery models [32]. Sharma and Singh proposed a new vulnerability discovery model based on the gamma distribution [33].

Each model defined above uses different approach with different assumptions and parameters. As a result, the VDMs can predict different vulnerability discovery rates using the same data and there is no guidance available about which model should be used in a given situation. This paper attempts to address these problems by integrating the classical VDMs and neural networks.

VI. Conclusion

In this study, based on the analysis of the neural network modeling and the linear combination model, we proposed a neural-network-based integrated model. The proposed model achieved better fitting and predictive capability and perform similar or superior as compared to classical and state of art VDM models. It not only retains the properties of vulnerability discovery models but also combines the MLP's good nonlinear mapping ability and generalization. To our knowledge, this is the first study which combines the VDMs and neural network to predict the number of vulnerabilities. Since, the proposed integrated method utilizes the results of classical software vulnerability discovery models serve as input to MLP neural network, so the further study is needed to select appropriate base models from number of software VDMs. In addition, a number of further investigations are possible such as replicate the experiment with advance machine learning techniques and more vulnerability datasets.

VII. Acknowledgment

This research was supported by Beijing National Research Center for Information Science and Technology (BNRist), and National Natural Science Foundation of China under Grant Nos. 90818021, 9071803.

References

[1] C. P. Pfleeger and S. Lawrence, *Security in Computing*. Prentice-Hall, 1997.

[2] X. Yang, G. Jabeen, P. Luo, X.-L. Zhu, and M.-H. Liu, "A unified measurement solution of software trustworthiness based on social-to-software framework," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 603–620, 2018.

[3] O. H. Alhazmi and Y. K. Malaiya, "Quantitative Vulnerability Assessment of Systems Software," *Reliability and Maintainability Symposium, 2005. Proceedings. Annual*, pp. 615–620, 2005.

[4] ——, "Application of vulnerability discovery models to major operating systems," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 14–22, 2008.

[5] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller, "Predicting vulnerable software components," in *ACM Conference on computer and communications security*. Citeseer, 2007, pp. 529–540.

[6] V. H. Nguyen and L. M. S. Tran, "Predicting Vulnerable Software Components with Dependency Graphs," *Proceedings of the 6th International Workshop on Security Measurements and Metrics - MetriSec '10*, p. 1, 2010.

[7] R. Scandariato, J. Walden, A. Hovsepyan, and W. Joosen, "Predicting vulnerable software components via text mining," *IEEE Transactions on Software Engineering*, vol. 40, no. 10, pp. 993–1006, 2014.

[8] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, "Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities," *IEEE Transactions on Software Engineering*, vol. 37, no. 6, pp. 772–787, 2011.

[9] H. K. Dam, T. Tran, T. Pham, S. W. Ng, J. Grundy, and A. Ghose, "Automatic feature learning for vulnerability prediction," *arXiv preprint arXiv:1708.02368*, 2017.

[10] Y. Shin and L. Williams, "Can traditional fault prediction models be used for vulnerability prediction?" *Empirical Software Engineering*, vol. 18, no. 1, pp. 25–59, 2013.

[11] O. H. Alhazmi and Y. K. Malaiya, "Prediction capabilities of vulnerability discovery models," in *RAMS'06. Annual Reliability and Maintainability Symposium, 2006.* IEEE, 2006, pp. 86–91.

[12] E. Rescorla, "Is finding security holes a good idea?" pp. 14–19, 2005.

[13] J. Kim, Y. K. Malaiya, and I. Ray, "Vulnerability discovery in multi-version software systems," in *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*. IEEE, 2007, pp. 141–148.

[14] H. Joh and Y. K. Malaiya, "Modeling Skewness in Vulnerability Discovery," *Quality and Reliability Engineering International*, no. September 2013, 2014.

[15] P. Kapur, N. Sachdeva, and S. Khatri, "Vulnerability discovery modeling," in *International conference on quality, reliability, infocom technology and industrial technology management*, 2015, pp. 34–54.

[16] A. Anand and N. Bhatt, "Vulnerability discovery modeling and weighted criteria based ranking," *Journal of the Indian Society for Probability and Statistics*, vol. 17, no. 1, pp. 1–10, 2016.

[17] A. Anand, S. Das, D. Aggrawal, and Y. Klochkov, "Vulnerability discovery modelling for software with multi-versions," in *Advances in Reliability and System Engineering*. Springer, 2017, pp. 255–265.

[18] R. Sharma, R. Sibal, and S. Sabharwal, "Change point modelling in the vulnerability discovery process," in *International Conference on Advanced Informatics for Computing Research*. Springer, 2018, pp. 559–568.

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

[20] A. Younis, H. Joh, and Y. Malaiya, "Modeling learningless vulnerability discovery using a folded distribution," in *Proc. of SAM*, vol. 11. Citeseer, 2011, pp. 617–623.

[21] R. C. Team, "R development core team. r: A language and environment for statistical computing. r foundation for statistical computing, vienna, austria; 2014," *Google Scholar*.

[22] F. Massacci and V. H. Nguyen, "An empirical methodology to evaluate vulnerability discovery models," *IEEE Transactions on Software Engineering*, vol. 40, no. 12, pp. 1147–1162, 2014.

[23] Y. K. Malaiya, N. Karunanithi, and P. Verma, "Predictability of software-reliability models," *IEEE Transactions on Reliability*, vol. 41, no. 4, pp. 539–546, 1992.

[24] 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*. IEEE Press, 1984, pp. 230–238.

[25] S. Rahimi, *Security vulnerabilities: Discovery, prediction, effect, and mitigation*. Southern Illinois University at Carbondale, 2013.

[26] O. H. Alhazmi and Y. K. Malaiya, "Quantitative vulnerability assessment of systems software," in *Reliability and Maintainability Symposium, 2005. Proceedings. Annual*. IEEE, 2005, pp. 615–620.

[27] O. H. Alhazmi, Y. K. Malaiya, and I. Ray, "Measuring, analyzing and predicting security vulnerabilities in software systems," *Computers & Security*, vol. 26, no. 3, pp. 219–228, 2007.

[28] O. H. Alhazmi and Y. K. Malaiya, "Measuring and enhancing prediction capabilities of vulnerability discovery models for apache and iis http servers," in *Software Reliability Engineering, 2006. ISSRE'06. 17th International Symposium on*. IEEE, 2006, pp. 343–352.

[29] K. Chan, D. Feng, P. Su, C. Nie, and X. Zhang, "Multi-cycle vulnerability discovery model for prediction," *Journal of Software*, vol. 21, no. 9, pp. 2367–2375, 2010.

[30] H. Joh and Y. K. Malaiya, "Modeling skewness in vulnerability discovery," *Quality and Reliability Engineering International*, vol. 30, no. 8, pp. 1445–1459, 2014.

[31] P. Kapur, V. S. Yadavali, and A. Shrivastava, "A comparative study of vulnerability discovery modeling and software reliability growth modeling," in *Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015 International Conference on*. IEEE, 2015, pp. 246–251.

[32] H. C. Joh and Y. K. Malaiya, "Periodicity in software vulnerability discovery, patching and exploitation," *International Journal of Information Security*, vol. 16, no. 6, pp. 673–690, 2017.

[33] R. Sharma and R. Singh, "Vulnerability discovery in open-and closed-source software: A new paradigm," in *Software Engineering*. Springer, 2019, pp. 533–539.