A Comprehensive Software Vulnerability Prediction and Risk Evaluation model based on Loss speed index

Gul Jabeen, Luo Ping, Junaid Akram, and Majid Mumtaz

School of Software Engineering, Tsinghua University, Beijing China

{jgl14,luop,znd15,maji16}@mails.tsinghua.edu.cn
http://www.tsinghua.edu.cn

Abstract. Software vulnerabilities present in different software systems represent significant security risks. Many papers have been published on evaluation and prediction of software vulnerabilities, but few established the link between the multiple variables of software security. In this paper, we consider a unified indicator called loss speed which integrates most important variables of software security, such as vulnerability occurrence time and severity which are used to evaluate the overall software quality measurement. This paper aims at filling the gap by addressing the severity of vulnerabilities and the effect of vulnerability severity that can cause a different amount of loss. Our work has several features such as: (1) It is used to predict the vulnerability severity/type and time in future. (2) Unlike traditional evaluation methods like expert scoring, our model uses historical data to predict the future loss speed of software. (3) The loss metric value is used to evaluate the risk associated with different software, which has a direct impact on software trustworthiness. Experiments performed on real software vulnerability datasets and its results are analyzed to check the correctness and effectiveness of the proposed model.

Keywords: Vulnerability loss, Vulnerability severity/type, Vulnerability time, Vulnerability Loss speed index, Risk evaluation

1 Introduction

With the speedy development of computer and Internet technology, software vulnerabilities pose real threats to computer users. In software, vulnerability is defined as a weakness in the security system that might be exploited to cause loss or harm [1].A highly severe vulnerability can allow an invader to gain full control of the system and also can potentially cause tremendous loss, not only to the software user but also to the entire society. Software systems such as an operating system, a management information system and other Internet applications have become an attractive target for malicious attacks that manipulate vulnerabilities and compromise systems security and trustworthiness. Vulnerability prediction models can help us to identify the vulnerability occurrences

and then risk evaluation can provide a quantitative indicator for software trustworthiness management. We believe this necessitates the need for a complete vulnerability prediction model and a risk evaluation metric to develop a secure and trustworthy software product.
It is well-known that trustworthiness of software is gradually developed from attributes such as reliability and security to the complex highly-composite concept. Traditional trustworthiness measurement models usually consider vulnerability occurrence time as a basic index/variable to predict the total number of vulnerabilities and their future occurrence time in the software [2][3].These predicted results only contribute to the identification of vulnerabilities in a specific time. However, only vulnerability time and number prediction are not enough to evaluate the risk associated with the software. Moreover, severity prediction is also an essential factor because a same number of vulnerabilities can cause a different amount of losses or damage based on their access scope and attack potential. Due to the different types of severities, different levels of vulnerability loss occur, including direct and indirect economic, repairing expense and so on. Therefore, we also have defined the total number of vulnerability severities in a system as vulnerability loss.
Software risk is always associated with the vulnerability severity. In this paper we further evaluated the software security risk based on the Loss speed. The loss speed defines the actual loss occurred in a specific time interval. It can combine the impact of time and the vulnerability loss, which gives the information about the overall loss speed for software when vulnerability has been exploited. As the vulnerability loss speed increases, software tends to get riskier and vice versa.
In this paper, we have considered the three primary variables to evaluate the trustworthiness of software in detail. As software risk is associated with vulnerability severity; it is the first variable we have considered. Vulnerability severity is an important variable because a same number of vulnerabilities might cause a different amount of losses based on their different access scopes and attack methods. The second variable of interest is Loss Speed. The loss speed defines the actual loss occurred in a specific time interval. It can combine the impact of time and the vulnerability loss, which gives the information about the overall loss speed for software when a vulnerability has been exploited. As the vulnerability loss speed increases, software tends to get more riskier and vice versa. The third variable of interest is the software vulnerability occurrence time, which is same as in the traditional software vulnerability models.
Based on the above three variables, this article proposes a comprehensive software vulnerability prediction model, which covers the essential information regarding vulnerabilities. The prediction results of this work can help to construct a risk evaluation model. The predicted loss speed value can be used to label the risk levels in software at different time intervals. It should make the risk management for companies and individuals more advantageous. This work will also be helpful in software patch management. According to the information of three specified predicted values (vulnerability loss speed, vulnerability severity vulnerability time), the developer can select a patch with high-risk values, which

ensures that the vulnerabilities with the highest priority are repaired first.
Our proposed model consists of the following parts. First, part is used to predict the vulnerability severity because our primary loss index is based on vulnerability severity. The predicted vulnerability severity value is used in the second phase of the model to predict the loss speed index. The second phase uses the historical data of vulnerabilities such as time and severity. A traditional Jelinski-Moranda model is used to predict the future vulnerability loss speed index. The third part gives the vulnerability occurrences time form the predicted values of first and second phases.

2 Related work

Currently, researchers focused on finding the complete, trustworthy and security metric methods in software development models. However, the increasing number of attacks via software vulnerabilities has also stimulated more attention on software vulnerability prediction and it risks evaluation. Vulnerability prediction models serve as a quantitative tool to describe the vulnerability discovery process and predict the future trend [2][3][4], which assist developers in assign resources to develop patches optimally and allow users to have a more clear understanding of the potential risks of software. Software vulnerability prediction models are divided into two groups according to the different sources of data such as attribute data and released data.

The attribute-based models focused on discovering the relationship between the code attributes and the vulnerabilities. Rahimi et al. [5] extracted the code complexity and the code quality from the source code to predict the vulnerabilities. Riccardo et al. [6] proposed a vulnerability prediction algorithm based on source code text mining. Shin et al. [7][8], used the software metrics, code churn and developer activity metrics as predictors by using logistic regression to detect vulnerabilities. They also investigated that fault prediction models provide a capability similar to the vulnerability prediction models for predicting vulnerabilities based on traditional metrics.

Similarly, vulnerability released-data-based models describe the relationship between the time and the accumulated number of vulnerabilities to predict the total number of vulnerabilities in a specified time. Some various kinds of models are proposed to achieve this goal. Alhazmi et al. [9][3] proposed a time-based model for a cumulative number of vulnerabilities. Later they have proposed an alternative model based on the effort, which is analogous to the software reliability growth models. Joh et al.[10] proposed alternative S-shaped models based on the Weibull, beta, gamma and normal distributions. Rescorla et al. [6], found that the number of vulnerabilities follows an exponential decay curve. Scandariato et al.[11] used a data mining techniques based on machine learning to predict which component of software application contain security vulnerabilities.

The models mentioned above focused on the recognition of vulnerability components and the prediction of the number of vulnerabilities. In a real application scenario, exportability and damage level triggered by vulnerabilities are different.

Nevertheless, a single attribute measurement not meets the requirement evaluation of single component[11][12]. herefore, in recent years, researchers focused on the metrics methods of overall trustworthiness in software and developed software trustworthiness metrics [13][14][15] models from different aspect. Therefore, it is essential to develop a standard and measurable trustworthy model, which comprehensively describe every attribute(security and reliability) associated with the trustworthiness of software and its variables (time, number, severity, Loss) in detail.

In the study of software risk evaluation, the basic idea of these methods is to determine evaluation factors to conduct the comprehensive assessment[16][17]. Although the improved algorithms are defined, but the evaluation results are influenced by expert scoring process. It is observed that the vulnerability prediction and risk evaluation methods are different. However, they are relate to each other which are based on effective prediction and construct a relevant evaluation model. We have developed as a complete trustworthy system based on vulnerability prediction and risk evaluation.

3 Description of basic attributes of software trustworthiness

To develop a complete trustworthy system, we have introduced a vulnerability prediction model which consider the three basic variables, which are used to examine not only the vulnerability occurrence rate but also the severity and Loss caused by these vulnerabilities in a specific time interval. They are explained as follows

Vulnerability time (t)*:* Vulnerability occurrence time is an integral part of security and lot of work has been done on time prediction. Traditional reliability and vulnerability prediction models used time as a basic measuring index [18][19][20] for the overall trustworthiness measurement of software. Because of the commonalities between faults and vulnerabilities allow development teams to used traditional fault prediction models and metrics for vulnerability prediction[8][21].

Vulnerability severity or Loss index (l)*:* Loss cause by software vulnerability, includes direct and indirect economic, potential and repairing expense and so on. For the sake of clarity, we have defined the amount of loss and its calculation method, and then discuss its prediction method in section 4.
Definition: The users of software must pay an extra cost (including money, maintenance workload, effort, lines of code and so on) caused by software vulnerability, which is called the amount of loss of software (or loss). The vulnerabilities information are used from the China National Vulnerability Database of Information Security, which are divided into four severity types. We have used these severities, and consider that lower value of severity cause low loss, and higher severity causes critical losses. Table 1 depict the detailed information:

Table 1. Categorization of based on Vulnerability Severities.

Severity types	Loss based on severities
1	Low loss
2	Moderate loss
3	High loss
4	Critical loss

According to the severity of losses classification, the greater the amount of loss, the more system became untrustworthy. Therefore, the above-classified losses in the metric model provides a basis for further risk evaluation and prediction to the degree of damage of the software system. Such as

$$l = \sum_{i=1}^{n} v_i (i = 1, 2, \cdots, n);$$

where l denotes loss, n denotes the number of vulnerability and v_i denotes the severity type of each vulnerability.

Vulnerability Loss speed Index (s)*:* Hence the above two variables are the most important attributes of software trustworthiness. We cannot ignore any of them to evaluate any risk associated with the software. Therefore, we have considered a new index is known as loss speed which measures loss per unit time. As vulnerability loss speed is an inclusive index, which constitutes the amount of loss occurred by vulnerability in a unit time. It is denoted as follow:

$$s = \frac{l}{T - t} \tag{1}$$

As Loss speed s is the rate of loss, which combines the impact of time and vulnerability loss based on different severities. l is the actual vulnerability loss and T is the actual vulnerability occurrence time, t is the current time. It is the comprehensive index which constitute all the information required to evaluate the trustworthiness of any software. However, the increase in loss speed value represents the untrustworthiness of software and lower value represents trustworthiness of software.

Example: If two vulnerabilities have same occurrence time, the risk associated with those vulnerabilities are compared by their actual loss speed. Such as, if vulnerability one has loss 2 in two days and other has loss 3 in the two days. The Loss speed of first and second vulnerabilities are 1 and 1.5 respectively. So the second vulnerability is considered as more untrustworthy with larger loss speed and it speeds up more quickly.

4 Model development

In this study, a complete software trustworthy model has been divided into three phases. The first phase estimates the severities associated with vulnerabilities. For prediction of severity of the vulnerability, data are expected to contain severity information about the vulnerabilities. The second phase constitutes the estimation of loss speed index prediction, which will denote the risk values associated with the software in future. As we have claimed that our proposed method is a complete trustworthiness model which consider all the basic variables (time, Loss, Loss speed index) associated with the specific trustworthy attributes (reliability, security). Therefore the mean time between vulnerabilities (MTBV) can also be estimated in the third phase.

4.1 Vulnerability Severity prediction model based on Markov Model

To predict the vulnerability loss l , data are expected to comprise of vulnerability severity $v_1, v_2, v_3 ... v_m : m > 0$ information. China Information Technology Security Evaluation Center has been maintaining the China National Vulnerability Database of Information Security (CNNVD) [22], each vulnerability has been classified based on its severity and seriousness. CNNVD has four severity levels such as critical-risk level (4), high-risk level (3), moderate-risk level (2) and low-risk level for each software. As the whole model is depends on the vulnerability severities, so to develop efficient and correct prediction is very important. Therefore, we have proposed a method for predicting severities based on Markov Chain. It is a method which offers more comprehensive descriptive model and has the potential to predict efficiently when very important events occur more than once. The software vulnerability severity prediction model is divided into two parts; testing phase and prediction phase. The testing phase is used to accumulate the data and get the initial probability vector and transition probability matrix and the prediction phase uses testing phase data for prediction. By this, we develop our vulnerability type prediction theorem.
Testing Phase
Step 1: In the test phase, each severity has been recorded to obtain the vulnerability sequences. We consider that the occurrence of vulnerabilities is random process $v_1, v_2, v_3 ... v_m : m > 0$, while the total number of vulnerabilities recorded in this phase are m, to obtain the predicted number of stages n.
Step 2: After the first step of the testing phase, the number of transitions of each vulnerability severity and its next vulnerability occurrence type shall be counted, namely the vulnerability severity transition matrix K and the one-step transition probability matrix P can also be calculated. At the same time, the initial probability vector of the test phase shall be obtained by recording the last occurrence of the vulnerability type v_m. The vulnerability transition matrix K

is represented as :

$$K = \begin{bmatrix} k_{11}\ k_{12}\ ...\ k_{1n}\ ... \\ k_{21}\ k_{22}\ ...\ k_{1n}\ ... \\ ...\ \ ...\ \ ...\ \ ...\ \ ... \end{bmatrix} \tag{2}$$

The k_{ij} denotes that the number of times the *i-th* type of vulnerabilities corresponds to the *j-th* type of vulnerability. One-step transition probability matrix P denoted as

$$P = \begin{bmatrix} p_{11}\ p_{12}\ ...\ k_{1n}\ ... \\ p_{21}\ p_{22}\ ...\ p_{2n}\ ... \\ ...\ \ ...\ \ ...\ \ ...\ \ ... \end{bmatrix} \tag{3}$$

The p_{ij} represents the probability of *i-th* vulnerability occurrence corresponds to the *j-th* type of vulnerabilities.As according to definition of conditional probability:

$$p_{ij} = P\{v_{s+1} = j | v_s = i\} = \frac{P\{v_{s+1} = j, v_s - i\}}{P\{v_s = i\}} \tag{4}$$

Because the occurrence of next vulnerability is mutually exclusive to each other, additionally the formula can be represented as:

$$P(v_s = i) = \sum_{j \epsilon l} P(v_s = i, v_{s+1} = j) \tag{5}$$

By combining 4 and 5 we get:

$$p_{ij} = \frac{P(v_s = i | v_{s+1} = j)}{\sum_{j \epsilon I} P(v_s = i, v_{s+1} = j)} \tag{6}$$

By considering the actual meanings of K

$$p_{ij} = \frac{k_{ij}}{\sum_{i,j \epsilon I} k_{ij}} \tag{7}$$

Therefore

$$p_{ij} = \frac{\frac{k_{ij}}{\sum_{i,j \epsilon I} k_{ij}}}{\frac{\sum_{j \epsilon I} k_{ij}}{\sum_{i,j \epsilon I} k_{ij}}} = \frac{k_{ij}}{\sum_{j \epsilon I} k_{ij}} \tag{8}$$

At this step transition probability matrix P has been determined completely. After determining transition probability matrix for vulnerability severities, the last step of testing phase is to determine the initial probability vector matrix. The initial probability vector $P^T(m)$ is the absolute probability vector , corresponding to the last vulnerability in the test phase.

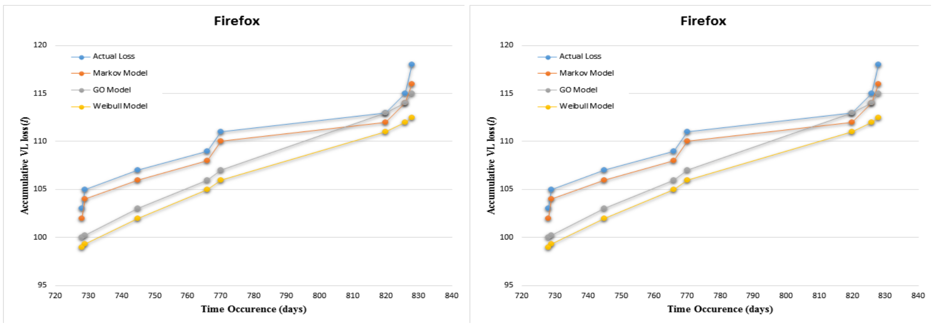
Prediction Phase

Step 3: According to the third step, the initial probability vector and transition probability matrix is obtained to calculate the absolute probability vector $P^T(m + r)$ for predicted phase

$$P^T(m + r) = P^T(m)P^{(r)} \tag{9}$$

Such the absolute probability vector for the $r(r\epsilon[1,n]) - th$ vulnerability in the predicted stage is: $P^T(m+r) = p_1(m+r), p_2(m+r)...p_k(m+r)$ and $max_{i\epsilon[1,k]}p_i(m+r)$ has the corresponding sequence number, $j\epsilon[1,k]$, the *j-th* term means that the *r-th* vulnerability most likely belongs to the *j-th* class. Then the probability of the *r-th* vulnerability is the *i-th* class in the software prediction stage $p_i(m+r)$, and most likely the type of vulnerability is *j-th* class.
Step 4: In the prediction stage, actual vulnerabilities are recorded for the purpose of obtaining the vulnerability type sequences $v_{m+1}, v_{m+2}, v_{m+3}...v_{m+n} : m > 0$, and to compare them with the step 3 predicted data, at the same time the actual data has been added to historical data to step 1 through 3 steps shall be repeated after which the probability occurrence can be again calculated for the next stage.

Evaluating predicting results We have used Adobe Flash Player and Firefox to evaluate resultsand compare the predicted results and risk levels. We used the same size of the dataset for both software. The dataset contains 58 vulnerabilities of both software which is collected from CNNVD [22], and the first 50 vulnerabilities were used to predict the next eight vulnerabilities. For comparison we choose the classical non-homogenous Poisson process (GO) model [19] and one of the most commonly lifetime distribution model in terms of reliability is Weibull model [10]. These models are generally used to predict the number of software vulnerabilities and failure and we used them to predict vulnerability severities.



(a) Predicted results for Firefox dataset (b) Predicted results of Adobe Flash Player

Fig. 1. Comparison of predicted results Markov model with different models

Figure 1 three models mentioned above. Our proposed model was applied to the dataset of Firefox, which shows better results than other models. Similarly, same model is applied to the dataset for Adobe Flash Player in Figure 1(b). These results show our developed Markov model provides more appropriate way of modeling, for vulnerability severity prediction problem because the occurrence of vulnerability severity is random and discrete and each vulnerability state has different transitions from one state to another

4.2 Vulnerability Loss Speed Prediction Model

TThe above-mentioned experiment showed that the predicted results of our proposed model have better results than other models. Therefore, the predicted values of vulnerability severity can be used further evaluate in Vulnerability Loss speed prediction. The predicted values of loss speed index are used to evaluate the overall trustworthiness T of software in future because it depicts the rate of loss in future at the specific unit time. The software risk is evaluated based on the loss speed predicted results.

Basic Mathematical relationship of loss speed: According to traditional vulnerability prediction models, the vulnerability interval time is considered as basic measuring criteria to evaluate the risk associated with software. We integrated the sub-attributes of software security such as the time and loss index and time in to one unified attribute such as Loss Speed s.

Assume that the stochastic variable ξ specify the loss speed in a specific time interval, while $V_{\xi}(s)$, represents the distribution function of ξ and s.

$$V_{\xi}(s) = P_r(\xi > s) \tag{10}$$

Trustworthiness $T_{\xi}(s)$ in terms of loss speed can be represented as follows:

$$T_{\xi}(s) = P_r(\xi > s) = 1 - V_{\xi}(s) \tag{11}$$

Equation 11 shows that with the increase of Loss speed s, the trustworthiness of software decreases and the vulnerability probability increases consequently. It also indicates that the $T_{\xi}(s)$ is the probability that software will not be vulnerable in the condition that when loss speed is s.
Loss speed rate $\lambda(s)$ is defined as the case where no software vulnerability occurred with loss speed s and under the condition that software can works normally $(s, \Delta s)$, however the loss speed is s and should be very small. The rate of loss speed is indicated as follows

$$\lambda(s) = \lim_{\Delta s \to 0} \frac{P_r(s + \Delta s > \xi > |\xi > s)}{\Delta s} \tag{12}$$

By expending above equation then we get

$$\lambda(s) = \lim_{\Delta s \to 0} \frac{P_r(s + \Delta s > \xi > |\xi > s)}{\Delta s.P_r(\xi > s)}$$

$$\lambda(s) = \lim_{\Delta s \to 0} \frac{V(s + \Delta s) - V(s)}{\Delta s.T(s)}$$

$$\lambda(s) = \frac{v(s)}{T(s)}$$

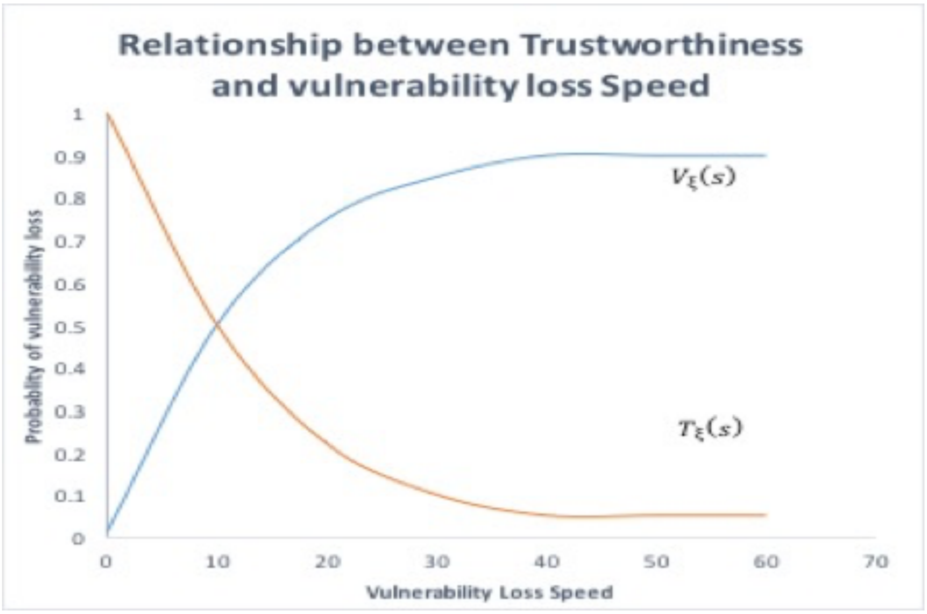


Fig. 2. Relationship between Trustworthiness and Vulnerability Loss Speed

However, $v(s)$ is the density function of stochastic variable ξ. Therefore, the above equation can be rewritten as

$$\lambda(s) = \frac{-T'(s)}{T(s)} \tag{13}$$

and assuming that the initial conditions that, $T(0)$ we get

$$T(s) = \exp - \int_0^s \lambda(s)ds \tag{14}$$

If loss rate is constant over a very short interval of time, it can be concluded as:

$$T(s) = \exp^{-\lambda s} \tag{15}$$

We have changed the traditional definition of software trustworthiness and used loss speed as a basic measurement unit so that the basic index will be changed in to new one such as MLSBV (Mean Loss Speed Between Vulnerabilities).

$$MLSBV = \int_0^\infty sv(s)ds = \int_0^\infty \lambda s e^{-\lambda s} ds = \frac{1}{\lambda} \tag{16}$$

Parameter Estimation: As Markov process model is an important class of stochastic model, which is widely used and plays a vital role in software vulnerability/ reliability predictions. Therefore, we have used it for vulnerability speed prediction because vulnerability loss also has a property of randomness and it is treated as probability problem. To prove the loss speed prediction accuracy, we used a traditional Markov model called Jelinski-Moranda model [21]. Basis assumptions are very important for traditional vulnerabilities and failure prediction models, because they help to simplify the computation and build the foundation of any model. Therefore, our proposed assumptions are as follow: Assume that the software will be vulnerable and its total loss is unknown constant L0; Each vulnerability in software is independent of one another; Loss rate between vulnerabilities is always constant, and its value is proportional to the remaining vulnerability loss in software. For the *i-th* vulnerability interval vulnerability loss rate function is a follows

$$\lambda(s_i) = \phi(L_0 - \sum_{i-1}^{n} l_{j-1}) \tag{17}$$

Where ϕ is proportionality constant, $L_0 - \sum_{i-1}^{n} l_{j-1}$ represents the total loss minus the total loss of j-1^{th} vulnerability.
Assume that $s_i$represents the vulnerability loss speed of the *i-th* interval, so its probability density function $v(s)$ is determined as:

$$v(s) = \lambda e^{-\lambda l} \tag{18}$$

Putting the value of λ in to equation 18:

$$= \phi(L_0 - \sum_{i-1}^{n} l_{i-1}) \exp\{\phi(L_0 - \sum_{i-1}^{n} l_{i-1})\} \tag{19}$$

Where L_0 and ϕ parameters are unknown constant. For the purpose of vulnerability prediction model, we choose maximum likelihood method for parameter estimation. It estimates the method by solving a set of equations. Assume that the loss peed interval measured in the test be $s_1, s_2, s_3 ... s_n$. The likelihood function of parameters L_0 and ϕ is

$$L(L_0, \phi) = \prod_{i=1}^{n} v(s_i) \tag{20}$$

$$L(L_0, \phi) = \prod_{i=1}^{n} \phi(L_0 - \sum_{i-1}^{n} l_{i-1}) \exp\{\phi(L_0 - \sum_{i-1}^{n} l_{i-1}) \tag{21}$$

By taking the logarithm on both side of above equation we get:

$$L(L_0, \phi) = \sum_{i=1}^{n} (ln\phi(L_0 - \sum_{i-1}^{n} l_{i-1}) - \phi(ln\phi(L_0 - \sum_{i-1}^{n} l_{i-1})s_i \tag{22}$$

By taking the partial derivatives of the log-likelihood function with respect L_0 to and ϕ respectively, and equating them to zero, we get the following likelihood equations as:

$$\frac{\partial ln(L_0,\phi)}{\partial L_0} = \sum_{i=1}^{n} \frac{1}{(L_0 - \sum_{j=1}^{i-1} l_j)} - \sum_{i=1}^{n} \phi s_i = 0 \tag{23}$$

By using numerical procedures, we can solve the above equation. Hence, it can be simplified as follows:

$$\sum_{i=1}^{n} \frac{1}{(L_0 - \sum_{j=1}^{i-1} l_j)} = \frac{n\sum_{i=1}^{n} s_i}{L_0 - \sum_{i=1}^{n} s_i - \sum_{i=1}^{n}(\sum_{j=1}^{i-1}) s_i}. \tag{24}$$

$$\phi = \frac{n}{L_0\sum_{i=1}^{n} s_i - \sum_{i=1}^{n}(\sum_{j=1}^{i-1}) s_i} \tag{25}$$

By using numerical procedures, the above two equations can be solved. The estimates of L_0 can be obtained by solving the Equation 24 and then inserting the estimated value of L_0 into the expression of ϕ, we may get the maximum likelihood estimate of ϕ.
Using the estimated values of L_0 and ϕ we can compute the unit vulnerability loss speed of the *k-th* vulnerability.

$$MLSBV = \int_0^{\infty} tf(l)dl = \frac{1}{\lambda} = \frac{1}{\phi(L_0 - \sum_{k=1}^{i-1} l_k)} \tag{26}$$

Above formula is used to estimation amount of loss speed of the next vulnerability, and is used to predict the vulnerability loss speed of each unit time, which is measured by a day. As $MLSBV$ increases software tends to be more untrustworthy, and lower value shows the trustworthiness of software.

4.3 Vulnerability Time prediction

Vulnerability occurrence time is the primary index of traditional software vulnerability prediction models. Loss speed has combined the impact of both time and vulnerability loss which is defined in section 3. Loss speed $l(t)$ is dependent on time T such as

$$s(t) = \frac{l}{T-t}$$

Hence it is clear from the above equations that vulnerability loss speed is the ratio of vulnerability loss type and time. Likewise $v_1, v_2, v_3...v_k$ is the severity type of vulnerability at different times. The total vulnerability loss l can be obtained based on these vulnerabilities in section 4.1. $MLSBV$ is the *k-th* unit loss speed $s(t)$ for the next vulnerability. It can be predicted using 4.2 . Since loss speed $s(t)$ is the ratio between the vulnerability severity and time. Therefore, we can also found the specific time intervals by using loss speed index and venerability loss. An equation 1 can be rearranged to get time as follows.

$$T = \frac{l}{s(t)} \tag{27}$$

The *i-th* time interval T can be found by using the above equation. It is understood that loss speed $s(t)$ is the vulnerability loss amount per unit time and we got this from MLSBV for the last vulnerability, l is the loss type caused by the vulnerability. Therefore, the time interval can be estimated by using the above equation.

The detailed diagram is illustrated in Figure 4.3

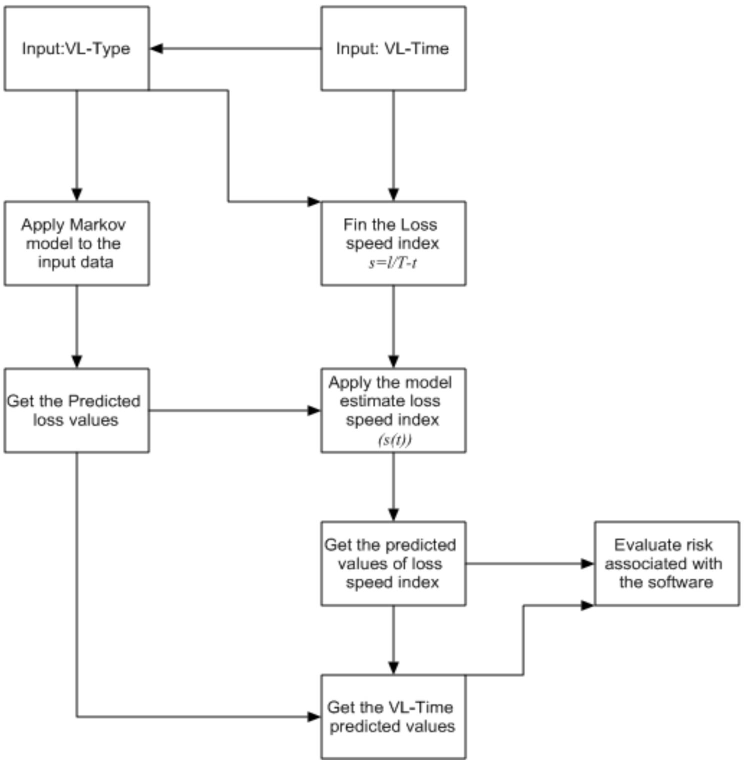

Input:VL-Type
Input: VL-Time
Apply Markov model to the input data
Fin the Loss speed index s=l/T-t
Get the Predicted loss values
Apply the model estimate loss speed index (s(t))
Get the predicted values of loss speed index
Evaluate risk associated with the software
Get the VL-Time predicted values

Fig. 3. Detailed diagram of proposed comprehensive software vulnerability prediction and risk evaluation model

5 Experimentation

To verify our model, we used the vulnerability data of Adobe Flash Player and Firefox (the data will be provided based on request). These two software systems were selected because they are most widely used systems. The data is obtained from CNNVD. China Information Technology Security Evaluation Center is responsible for the maintenance of CNNVD vulnerability data.

5.1 Experimental results illustration

To evaluate and compare the predicted results of Firefox and Adobe Flash Player, we used the same data size of both software. To avoid the chance occurrence of experimental results based on the Firefox and Adobe flash player datasets of 58 vulnerabilities. We choose two different number of vulnerability dataset such as 54 and 56. Let n be the size of dataset, the first $n-2$ vulnerabilities are used to predict the remaining two vulnerabilities based on our model.

Firefox: The prediction accuracy of our model is based on the loss index (l) prediction, therefore in section 4.1 we show the predicted accuracy of our model more clearly as compared to other classical models.

Table 2. Actual CNVVD data of Firefox

VL-No (n)	VL-Time (T)	VL-Type ($v(i)$)	VL-Loss l	VL-Loss-Speed $s(t)$
55	4	2	111	0.5
56	50	2	113	0.04
57	6	2	115	0.33
58	2	2	118	1.5

Table 3. Firefox Predicted values for dataset 1

VL-No (n)	Pred-VL-Loss (l)	Pred-VL-Loss-Speed $s(t)$	Pred-VL-Time (T)
55	111	0.6298	4.88
56	113	0.6184	4.89

Table 4. Firefox Predicted value for dataset 2

VL-No (n)	Pred-VL-Loss (l)	Pred-VL-Loss-Speed $s(t)$	Pred-VL-Time (T)
57	115	0.459	4.92
58	118	0.462	4.91

The estimated results of Firefox based on our proposed comprehensive trustworthy model are shown in Table 3 and 4. Table 3 shows the estimated results based on the first 54 datasets. The model shows that the total remaining vulnerabilities (n) are detected as 54, and the total loss occurrence (L_0) is estimated as 227. Based on these parameters the loss speed index is estimated for every future vulnerability. The VL-pred-Loss-Speed ($s(t)$) in Table 3 shows the estimated values for 55th and 56th vulnerabilities as 0.6298 and 0.6184 respectively. Similarly, Table 4 shows the estimated results based on the second Firefox dataset. Both of the tables also shows the estimated results of Pred-VL-Time(T) as defined in section 4.3.

Adobe Flash player: Similarly, for Adobe Flash Player, the two datasets are used based on 54 and 56 vulnerabilities. The next two vulnerabilities are estimated based on the every dataset. The actual dataset for predicted values are shown in Table 5.

Table 5. Actual CNVVD data of Adobe Flash Player

VL-No	VL-Time	VL-Type	VL-Loss	VL-Loss-Speed
(n)	(T)	$v(i)$	l	$s(t)$
55	6	4	157	0.666
56	13	2	159	0.153
57	21	3	162	0.142
58	17	4	166	0.235

Table 6. Adobe player Predicted value of DS-1

VL-No	Pred-VL-Loss	Pred-VL-Loss-Speed	Pred-VL-Time
(n)	(l)	$s(t)$	(T)
55	156	0.388	12.810
56	160	0.398	12.722

Table 7. Adobe flash player predicted values of DS-2

VL-No	Pred-VL-Loss	Pred-VL-Loss-Speed	Pred-VL-Time
(n)	(l)	$s(t)$	(T)
57	162	0.364	12.51
58	165	0.371	13.39

Similarly, the estimated results of Adobe flash player based on our proposed comprehensive trustworthy model are shown in Table 6 and 7. The predicted accuracy of our model is based on the Pred-VL-Loss (l) values for Adobe flash player. These values are further evaluated in second phase for loss speed estimation which will be the basic measuring index to evaluate the risk associated with the software in future. Table 6 shows the estimated results based on the first 54 datasets.Total remaining loss (L_0) is estimated as 309.6 and 322 for DS-1 and DS-2 respectively. However, the estimated parameters are used to predict the loss speed index for future vulnerability. The Pred-VL-Loss, Pred-VL-Loss-Speed and Pred-VL-Time are shown in Table 6 and Table 7, which demonstrate the detailed description about of the software vulnerabilities.

Application: Based on the above information one can get the detailed information about software vulnerability occurrence in future. As traditional evaluations methods used the historical data to evaluate the system with expert scoring. We also used the prediction results of in section 4.1 and Loss speed index estimation in section 4.2 , which are estimated based on historical data. We consider a case to allow a risk evaluation of software. Our model gives the detailed and complete information about the vulnerabilities, which helps to automate the risk evaluation method. The estimated loss speed index is applied to the security rating and patch management of software. It is known that, the patch management involves a significant scheming plan to assign limited resources to the patches. Different factors affect the patch priorities such as the number of users, the user roles, and the software dependencies. However, we have added a new factor to evaluate the risk associated with the software in future. Based on our proposed method if the loss speed index is increased in future, the patch is considered as more risky and untrustworthy. Therefore its priority should be high. If the loss speed index value for next vulnerabilities becomes less than it is not necessary to install the patch and the software is considered as trustworthy.

6 Conclusion

In this study, a comprehensive vulnerability prediction model is developed, which evaluates the software trustworthiness entirely. Our study has changed the basic concept of traditional software evaluation method, based on new loss speed index. It is used to integrate the main variables of software trustworthiness. It also combined the prediction with evaluation and gives a new approach to risk evaluation. Vulnerability loss has been estimated based on Markov model. The predicted loss values are used to estimate the loss speed index by using the traditional software reliability model. These values are significantly used to evaluate the overall trustworthiness of software system.

To verify the effectiveness of our prediction model, we have performed experiments using the vulnerability datasets. The results illustrated that the predicted values are more closed to the actual data. Therefore, it is viable to use the model results to conduct the risk assessment and patch management. We believe that future researchers focused on the measurement of loss speed index as a trustworthy software attribute like reliability, usability and so on. This unified indicator will be a basis for measuring overall credibility and trustworthiness of the software. It will be our best effort to get good results in our future work and show a research study to compare different software based on our proposed approach.

References

1. C. P. Pfleeger, S. Lawrence, Security in Computing, Prentice-Hall, 1997.
2. D. Last, Using Historical Software Vulnerability Data to Forecast Future Vulnerabilities.
3. O. Alhazmi, Y. Malaiya, Prediction capabilities of vulnerability discovery models, RAMS '06. Annual Reliability and Maintainability Symposium, 2006. 00 (C) (2006) 86–91. doi:10.1109/RAMS.2006.1677355.
   URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1677355`
4. Y. Liu, L. Zhang, P. Luo, Y. Yao, Research of trustworthy software system in the network, in: Proceedings - International Symposium on Parallel Architectures, Algorithms and Programming, PAAP, 2012, pp. 287–294. doi:10.1109/PAAP.2012.47.
5. S. Rahimi, M. Zargham, Vulnerability scrying method for software vulnerability discovery prediction without a vulnerability database, IEEE Transactions on Reliability 62 (2) (2013) 395–407. doi:10.1109/TR.2013.2257052.
6. E. Rescorla, Is finding security holes a good idea? (2005). doi:10.1109/MSP.2005.17.
7. An empirical model to predict security vulnerabilities using code complexity metrics, Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '08 (2008) 315doi:10.1145/1414004.1414065.
   URL `http://portal.acm.org/citation.cfm?doid=1414004.1414065`
8. Y. Shin, L. Williams, Can traditional fault prediction models be used for vulnerability prediction?, Empirical Software Engineering 18 (1) (2013) 25–59. doi:10.1007/s10664-011-9190-8.
9. Quantitative vulnerability assessment of systems software, Reliability and Maintainability Symposium, 2005. Proceedings. Annual (2005) 615–620doi:10.1109/RAMS.2005.1408432.
   URL `https://www.dropbox.com/s/pjc8a97q5vjomgp/Quantitativevulnerabilityassessmentofsystemssoftware.pdf?dl=0`
10. H. Joh, Y. K. Malaiya, Modeling Skewness in Vulnerability Discovery (September 2013). doi:10.1002/qre.1567.
11. R. Scandariato, J. Walden, A. Hovsepyan, W. Joosen, Predicting Vulnerable Software Components via Text Mining, IEEE Transactions on Software Engineering 40 (10) (2014) 993–1006.
12. H. G. Gurbuz, B. Tekinerdogan, Model-based testing for software safety : a systematic mapping studydoi:10.1007/s11219-017-9386-2.
13. Y. Huang, P. Luo, Metric model for trustworthiness of computer supported cooperative design platform considering effect of multiple short boards. (2014).
14. Y. Xi, G. Jabeen, L. Ping, A Unified Measurement Solution of Software Trustworthiness Based on S2S Framework, Journal of Computer Science and Technology.
15. R. Jiang, A Trustworthiness Evaluation Method for Software Architectures Based on the Principle of Maximum Entropy (POME) and the Grey Decision-Making Method (GDMM) (2014) 4818–4838doi:10.3390/e16094818.
16. N. GAO, G. Ling, Y. HE, Y. LEI, Q. GAO, Dynamic risk assesment based on bayesian attack graphs, Journal of sichuan university 48 (111-118).
17. A. Kumar, O. Williams, X. Li, Towards an efficient risk assessment in software projects Fuzzy reinforcement paradigm R 0 (2017) 1–14. doi:10.1016/j.compeleceng.2017.07.022.

18. J. D. Musa, A theory of software reliability and its application (1975). doi:10.1109/TSE.1975.6312856.
URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6312856`
19. A. L. Goel, K. Okumoto, Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures, IEEE Transactions on Reliability R-28 (3) (1979) 206–211. doi:10.1109/TR.1979.5220566.
URL `http://ieeexplore.ieee.org/document/5220566/`
20. M. Xie, G. Y. Hong, C. Wohlin, Software reliability prediction incorporating information from a similar project (1999). doi:10.1016/S0164-1212(99)00065-5.
21. P. K. Kapur, V. S. S. Yadavali, A. K. Shrivastava, A Comparative Study of Vulnerability Discovery Modeling and Software Reliability Growth Modeling (Ablaze) (2015) 246–251.
22. C. I. T. S. E. Center. china national vulnerability database of information security, `http://www.cnnvd.org.cn/`, accessed: 2017-09-10.