J Indian Soc Probab Stat (2016) 17:1–10
DOI 10.1007/s41096-016-0006-4

RESEARCH ARTICLE

Vulnerability Discovery Modeling and Weighted Criteria Based Ranking

Adarsh Anand[1] · **Navneet Bhatt**[1]

Accepted: 2 May 2016 / Published online: 12 May 2016

Abstract Attacks on code based systems have been recent area of concern for the software developers. Of late, this side of the coin has received much attention as the loss happening due to this exploitation has been understood to a good extent. In today's neck to neck competitive marketplace, firms have to come with their software products as quickly as possible. In order to do so, they are releasing their offerings at a much higher pace as it used to be earlier and so, many bugs sustain in the software at the time of release. Such a code is prone to be easily attacked by any community working in the field. With the goal of predicting or investigating these potential number of loop holes (vulnerabilities); many vulnerability discovery models (VDMs) have been proposed in the literature. In this paper, we develop a model which follows a hump-shaped curve while discovering the security vulnerabilities. Furthermore, we have compared different set of VDMs with the proposed model using the five comparison criteria and each criterion has been assigned different weight in order to capture the ranking of proposed model. For checking the veracity and predictive capabilities of proposed model, validation is done on two different data sets and results shows that the weighted criteria methodology shows very promising results for model comparison.

Keywords Hump-shaped curve · Ranking · Vulnerability discovery modeling · Weighted criteria introduction

✉ Adarsh Anand
adarsh.anand86@gmail.com

1 Department of Operational Research, Faculty of Mathematical Sciences, University of Delhi, Room No. 207, 2nd Floor, Delhi 110007, India

1 Introduction

Software engineering process has changed noticeably from a decade or two ago. With limited or no network access, computers were like an island in and of themselves. Software applications were usually deployed as independent units after being tested for functionality on isolated systems. Software service marketplace did not exist at that time except in the concept stage. Security was not considered as a key aspect since the worst that could happen following a breach was that the exploit could attack only them in a bounded environment. But with the growth of the Internet and the advancement of technology led to a rapid paradigm shift in the way how computer systems are networked and development of software applications begins, resulting in a momentous effect on security. The high mark of computer and network connectivity available in today's society implies that the applications not intended to function securely are more susceptible to attack from both insiders and outsiders.

Security breaches are of great concern in today's virtually connected economy. Computer systems need to be protected and must be expected to function in potentially hostile environments. With the digitalization of every department, software systems play a key role while performing any operation. For the well-being of national security and economic growth; secure system are the need of the time. If any firm or government organization is directly involved in software/application development, either as an internal or to meet business needs, security measures must be incorporated from the starting (requirement/planning) phase of the software project and processed all the way through the operational phase. Unfortunately, security in most cases of software development is an afterthought. Limited resources, lack of time, and a pervasive lack of awareness about security, are a number of reasons that prevent implementing security measures from the beginning. Hot fixing a security issue is often bolted aftermath, as a response to some threat or exploit. But encrypting the code and maintaining security throughout all the phases of the Software Development Life Cycle (SDLC), has been proven to be more effective and less expensive rather accumulating security to an operational system.

At some points in the course of software development, coders unintentionally create loop-holes that are later noticed and mitigated. As the software gets deployed and goes into its operational phase, it immediately becomes a possible target for attacks. Ideally, from the moment of implementation it requires to shield itself from the intruders. In this regard, software developers have to work efficiently and proficiently in order to detect and remove—and later avoid—vulnerabilities before product has been released. With many advantages a malicious attacker might leave a defender in a dilemma. For an attacker it is easier to exploit only one weakness, whereas the defender has to secure itself against all potential threats. As well, exploiters have the comfort of attacking software systems whenever they like, while the defender needs to be on constant watch. Accumulating enough security measures would veer the advantage to the defender, putting the attacker in isolation.

With proper modeling of the security design and attack surface, the developer becomes aware of the potential vulnerabilities that can be exploited, and can mitigate the threats before they're exploited by the attacker. Schultz et al. (1990) defined vulnerability as a "defect which enables an attacker to bypass security measures". Suggesting a proper alerting and on demand automated system in the software that aids the developer, and allows the software developer to be on watch constantly. Rising concern over security aims to fill the void amongst developer and attacker. Researchers have been showing an increased attention by quantifying the vulnerabilities in the software. As a result, few authors have incorporated the concept of Software Reliability Modeling in order to provide a quantitative aspect of security. They have considered that rate at which vulnerabilities get discovered is analogous to the rate of fault detection as in reliability assessment. As a brief review of related research, security vulnerability discovery in software has attracted some attention recently. Anderson (2002), used the Brady et al. (1999) model developed for software reliability to find out the trend in vulnerability discovery. Rescorla (2005) has attempted to classify trends in the vulnerability data by applying the linear and exponential models. Alhazmi and Malaiya (2005) showed the relationship between the cumulative vulnerabilities with time and further developed a logistic and an effort based exponential model by segregating the efforts required to discover vulnerabilities. Later, Kapur et al. (2015) showed the use of logistic detection rate while discovering the vulnerabilities. Unlike these studies, in this paper, we further investigate the modeling of security vulnerability. Our model incorporates the unique pattern of vulnerability exposure, such as the hump-shaped vulnerability discovery rate function. In addition, because the VDMs cannot provide good results for a particular data set, in this study, we formulate a ranking based analysis making use of weighted criteria method (Anjum et al. 2013) for comparing different available models in the literature with our proposed model for a given software vulnerability data. The models are ranked based on the overall weight assigned to different models.

Rest of the paper is organized as follows. Section 2 reviews some existing VDMs. Section 3 comprises of notations and we have discussed about the mathematical formulation of our proposed model. The proposition has been validated on real data sets; therefore, the results are given in Sect. 4. Finally, conclusion is given in Sect. 5.

2 Literature Review

Categorically, the vulnerability discovery models (VDMs) proposed so far are classified as time based and effort based models. In the following section we will describe and illustrates some well-known time based VDMs.

2.1 Rescorla Exponential Model

This model attempts to fit the vulnerability finding rate exponential with time. Rescorla used the famous Goel–Okumoto SRGM (Goel and Okumoto 1979) to fit the security vulnerabilities. The exponential model can be given as:

$$\omega(t) = N\lambda e^{-\lambda t} \tag{1}$$

where N represents the total number of vulnerabilities in the software system and λ is the rate constant. After integrating the above Eq. (1), equation for the cumulative number of vulnerabilities can be obtained as:

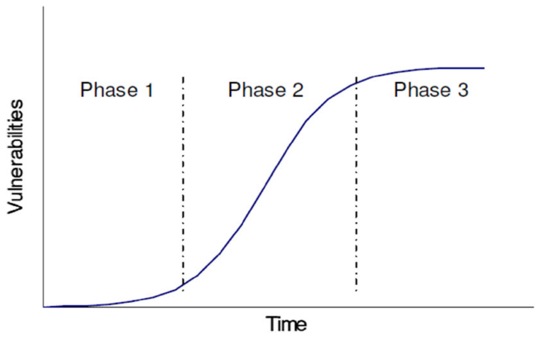

Fig. 1 The basic 3-phase S-shaped model proposed by Alhazmi and Malaiya (2005)

$$\Omega(t) = N(1 - e^{-\lambda t}) \tag{2}$$

2.2 Alhazmi–Malaiya Logistic Model (AML)

The AML is an S-shaped, time based logistic model proposed by Alhazmi and Malaiya (2005). According to this model vulnerability discovery occurs in three phases: learning, linear, and saturation. The starting phase is a learning phase, in which the detectors begin to understand the target system, so they do not report much vulnerability. After the learning phase, the detectors are familiar with the product and the new system will attract a significant number of users and the number of vulnerabilities reported grows linearly. After a certain time, the system starts getting replaced by newer versions. The users start to mitigate or upgrade to a more modern system. At the same time, detectors are less interested in the system, and the vulnerability reporting rate declines. This is termed the saturation phase. Figure 1 depicts these phases as identified by Alhazmi and Malaiya.

The vulnerability discovery rate of the AML model is given by the differential equation:

$$\frac{d\Omega(t)}{dt} = A\Omega(B - \Omega(t)) \tag{3}$$

where $\Omega(t)$ is the cumulative number of vulnerabilities, t is the calendar time. A and B are the empirical constants determined from recorded data. Solving the differential Eq. (3) we obtain

$$\Omega(t) = \frac{B}{BCe^{-ABt} + 1} \tag{4}$$

Where C is the constant introduced while solving the Eq. (3).

2.3 Kapur et al. (2015)

This model is known as flexible model and uses logistic detection rate. The differential equation depicting the vulnerability discovery is based on the removal phenomenon as given by Kapur and Garg (1992).

$$\frac{d\Omega(t)}{dt} = r(t)(a - \Omega(t)) \tag{5}$$

$$\frac{d\Omega(t)}{dt} = \frac{r}{1 + \beta e^{-rt}}(a - \Omega(t)) \tag{6}$$

where r is the rate of vulnerability detection. Solving the above differential equation, under initial condition $\Omega(0) = 0$, we get mean value function as:

$$\Omega(t) = a\left(\frac{1 - e^{-rt}}{1 + \beta e^{-rt}}\right) \tag{7}$$

where $\Omega(t) =$ cumulative number of vulnerability detected by time t.

3 Modeling Vulnerability Discovery Process of Software

Although SRGMs have been in use for almost four decades, the security community has only begun to apply these models to vulnerability data in the past few years. As a result, many researchers were able to fit their respective vulnerability discovery models corresponding to various data projects like RedHat 6.2, OpenBSD, WinNT4 and Win98 (Rescorla 2005; Ozment 2007; Alhazmi and Malaiya 2005).

In real life situation, while discovering vulnerabilities it involves much more testers in the testing process and most of the testers are attackers trying to exploit the loop holes. Further, due to the attractiveness of the software more users try to find the potential vulnerable points in the software. More specifically, each release of software can attract increasing number of testers (attacker and defender) in the early phase since more and more people know it and use it. After the number of testers reaches at the peak, it will decrease since the software is losing its attractiveness over time. Consequently, the number of vulnerability reaches its peak, and will decrease as the product loses its attractiveness. Accordingly, it is reasonable to assume that the vulnerability detection rate follows a hump-shaped curve (Li et al. 2011). In order to describe this special property, the first derivative of logistic function is selected and it is given by

$$r(t) = \left(\frac{r^2\beta e^{-rt}}{(1 + \beta e^{-rt})^2}\right) \tag{8}$$

Therefore, the mean value function for vulnerability detection for the proposed model can be written using Eq. (8) in Eq. (5) as follows:

$$\frac{d\Omega(t)}{dt} = \left(\frac{r^2\beta e^{-rt}}{(1 + re^{-rt})^2}\right)(a - \Omega(t)) \tag{9}$$

Solving the differential Eq. (9) under the boundary condition $\Omega(0) = 0$, we get:

$$\Omega(t) = a\left(1 - e^{-r\left\{\left(\frac{1}{1+\beta e^{-rt}}\right) - \left(\frac{1}{1+\beta}\right)\right\}}\right) \tag{10}$$

Table 1 Parameter estimates of the proposed model DS-I

Parameter estimates	Models under comparison			
	Rescorla	AML	Kapur et al.	Proposed
a	186541.4346	876.5189803	942.1483834	3565.538441
b	0.000233496	0.000421763	0.316680475	0.316100681
b	–	0.052122998	28.4109939	33.04984889

Table 2 Parameter estimates of the proposed model DS-II

Parameter estimates	Models under comparison			
	Rescorla	AML	Kapur et al.	Proposed
a	67942.12386	649.3660296	654.8518541	1602.933986
b	0.000724951	0.000884829	0.538932293	0.53586275
β	–	0.080632759	38.99594129	49.86224165

It is interesting to note the behaviour of our proposed model. Initially, when the vulnerability discovery process begins, i.e. at $t = 0$; we have $\Omega(t) = 0$ and at later stages when the vulnerability discovery process is carried on for an infinite time i.e. at $t = \infty$; the vulnerabilities are almost discovered by the testers giving

$$\Omega(t) = a\left(1 - e^{-\left(\frac{r\beta}{1+\beta}\right)}\right). \tag{11}$$

4 Data Analysis

The proposed model has been validated on the security vulnerability data of two software products namely Microsoft Windows XP (DS-I) and Apple Macintosh Server (DS-II) obtained CVE details (Mac Os X Server 2016; Windows Xp 2016). Further, a set of models have been used in order to evaluate which model is performing best by comparing their overall ranking using the weighted criteria method. We have estimated the parameters of the developed model using SPSS tool based on non-linear least square method. The parameter values of the proposed model along with the models proposed in the literature have been calculated and are given in Tables 1 and 2.

4.1 Comparison Criteria

A model can generally be analysed according to its retrodictive capability (i.e. its ability to reproduce the observed behaviour of the software), and predictive capability (i.e. its ability to predict future behaviour of the software from the observed failure data) (Huang and Lyu 2011). Since the data sets used for the validation are of vulnerability count, we employ the Goodness-of-fit criteria to evaluate the performance of the model

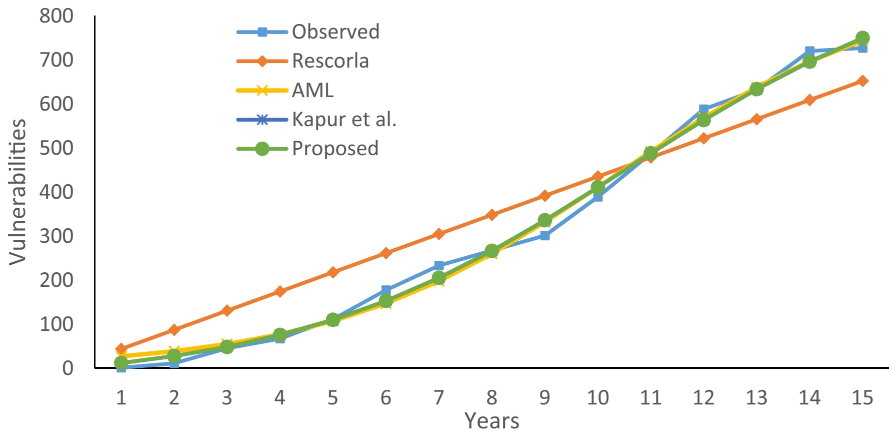

Fig. 2 Goodness of fit for DS-1

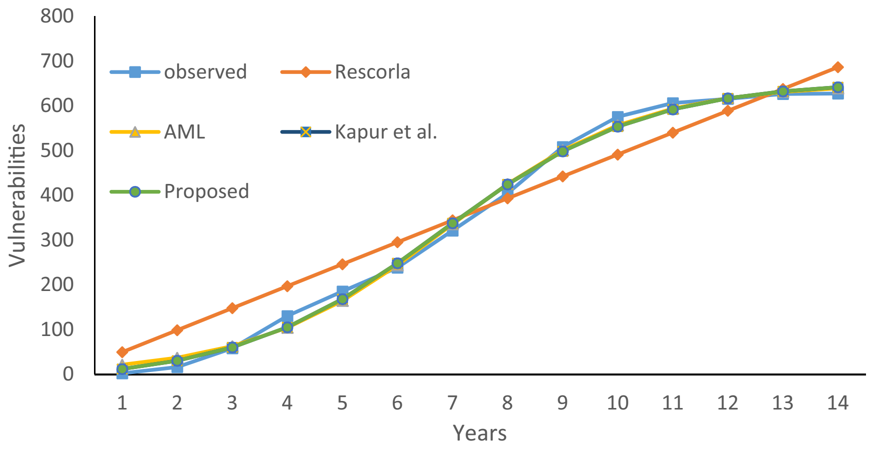

Fig. 3 Goodness of fit for DS-2

which comprises of the mean square error, bias, variation, RMSPE and the coefficient of multiple determinations (Kapur et al. 2011). From the Figs. 2 and 3 of the observed vulnerabilities and the predicted vulnerabilities of different models it was not clear that which of them is performing best. Thus it is required to apply a method to rank the models and get the best amongst them on the basis of certain comparison criterions.

4.2 Weighted Criteria Value Method

The steps involved in this approach follow (Anand et al. 2014; Anjum et al. 2013):

i. For each model compute the attribute value a_{ij} for the value of jth criteria of ith model. Let us consider n number of models with m criteria. Further, maximum value and minimum value of each criterion is determined.

ii. As these criterion ratings are different for each model, the criteria ratings are determined as under:
Case 1 When smaller value of the criterion represents fitting well to the actual data i.e. best value:

$$X_{ij} = \frac{Max.Value - Criteria\,Value}{Max.Value - Min.Value}$$

Case II When large value of the criterion represents fitting well to actual data i.e. best value:

$$X_{ij} = \frac{Criteria\,Value - Max.\,Value}{Max.Value - Min.Value}$$

iii. The weight matrix can be represented as:

$$W_{ij} = 1 - X_{ij}$$

iv. Weighted criteria value matrix is computed by the product of weight of each criterion with the criteria value i.e.

$$A_{ij} = W_{ij}^{*} a_{ij}$$

v. Permanent value is the weighted mean value of all criteria. This value is given by below:

$$Z_i = \frac{\sum_{i=1}^{m} A_{ij}}{\sum_{i=1}^{m} W_{ij}}$$

where $i = 1, 2, 3, \ldots\ldots\ldots., n$

The Model value is taken to be absolute of the permanent value. The lower parameter value indicates a better fit to the model. The results of weighted criteria approach are shown in Tables 3, 4, 5 and 6.

After applying weighted criteria value method in determining the ranking for different vulnerability discovery models, it has been found that our proposed model is at rank one followed by Kapur et al. (2015) model for the data set DS-I (Table 4). Further (Table 6) show that Kapur et al. (2015) is ranked one followed by our proposed model

Table 3 Weighted value of criteria (DS-I)

Model	MSE	Bias	Variation	RMSPE	R square
Rescorla	7211.407	−30.9662	75.29213	81.41138	0.9
AML	544.9552	−2.24451	21.48736	21.60427	0.993
Kapur et al.	436.3341	−1.14487	19.30275	19.33668	0.994
Proposed	436.1846	−1.14681	19.29931	19.33335	0.994

Table 4 Model permanent value and ranking (DS-I)

Model	Sum of weight	Sum of weighted value	Model value	Model rank
Rescorla	3	7368.11	2456.037	4
AML	2.044201	9.199488	4.500286	3
Kapur et al.	2.000137	−0.13903	−0.06951	2
Proposed	1.999935	−0.15273	−0.07637	1

Table 5 Weighted value of criteria (DS-II)

Model	MSE	Bias	Variation	RMSPE	R square
Rescorla	4107.359	−17.3335	58.88841	61.38645	0.936
AML	304.824	−1.24279	16.00827	16.05644	0.996
Kapur et al.	273.2647	−0.40874	15.20013	15.20563	0.996
Proposed	273.4	−0.41516	15.20371	15.20938	0.996

Table 6 Model permanent value and ranking (DS-II)

Model	Sum of weight	Sum of weighted value	Model value	Model rank
Rescorla	3	4227.63402	1409.21134	4
AML	1.995872236	2.915458957	1.460744282	3
Kapur et al.	2	0.587262607	0.293631304	1
Proposed	1.99981909	0.593126336	0.296589996	2

for DS-II, but as reflected from the Table 6; the values in the column, model value; of our proposed model and of Kapur et al. (2015) model are approximately equal to each other. Hence, suggesting that the proposed model and the model given by Kapur et al. (2015) are performing very well. Also, it is observed that the respective ranking of models AML and Rescorla is not changing with change in the data.

5 Conclusion

Concern arises due to the security issues has shifted the roots of information industry to the uprising events of virtual thefts through the means of exploiting sensitive information. The Vulnerability Discovery approach provides a new paradigm of software development, where uprooting the vulnerable points has shown a clear-cut shield over the intruders. With the release of software, the attractiveness amongst the malicious users is generally increases and over time it gets decreases. In order to describe this unique property, in this paper; a new vulnerability discovery model has been proposed which uses a hump-shaped detection curve. Furthermore, the validity and accuracy of the proposed model have been carried out on two different data set of operating system. The results are quite promising as can be viewed through the numerical illustration. A clear cut inference was not depicted by the goodness of fit measures. Hence, a technique called weighted criteria has been implemented in order to rank the various models for determining the optimal fit.

Acknowledgments The research work presented in this paper is supported by grants to the first author from University of Delhi, R&D Grant No-RC/2015/9677, and Delhi, India.

References

Alhazmi OH, Malaiya YK (2005) Modeling the vulnerability discovery process. In: Proceedings of 16th IEEE international symposium on software reliability engineering (ISSRE'05), pp 129–138

Anand A, Kapur PK, Agarwal M, Aggrawal D (2014) Generalized innovation diffusion modeling & weighted criteria based ranking. In: Reliability, infocom technologies and optimization (ICRITO) (Trends and Future Directions), pp 1–6

Anderson RJ (2002) Security in opens versus closed systems—the dance of boltzmann coase and moore. Open Source Software: Economics, Law and Policy, Toulouse, France

Anjum M, Harque MA, Ahmad N (2013) Analysis and ranking of software reliability models based on weighted criteria value. Int J Inform Technol Comput Sci 5:1–14

Brady RM, Anderson RJ, Ball RC (1999) Murphy's law, the fitness of evolving species, and the limits of software reliability. Cambridge University Computer Laboratory Technical, Report No 471

Goel AL, Okumoto K (1979) Time-dependent error detection rate model for software and other performance measures. IEEE Trans Reliab 28:206–211

Huang CY, Lyu MR (2011) Estimation and analysis of some generalized multiple change-point software reliability models. IEEE Trans Reliab 60(2):498–514

Kapur PK, Garg RB (1992) A software reliability growth model for an error removal phenomenon. Softw Eng J 7(4):291–294

Kapur PK, Pham H, Gupta A, Jha PC (2011) Software reliability assessment with OR application. Springer, Berlin

Kapur PK, Sachdeva N, Khatri SK (2015) Vulnerability discovery modeling. In: International conference on quality, reliability, infocom technology and industrial technology management, pp 34–54

Li X, Li YF, Xie M, Ng SH (2011) Reliability analysis and optimal version-updating for open source software. Inform Softw Technol 53:929–936

Mac Os X Server (2016) Vulnerability statistics. http://www.cvedetails.com/product/2274/Apple-Mac-Os-X-Server.html?vendor_id=49. Accessed 6 Feb 2016

Ozment A (2007) Vulnerability discovery & software security. Dissertation, University of Cambridge

Rescorla E (2005) Is finding security holes a good idea? Secur Priv 3:14–19

Schultz E Jr, Brown DS, Longstaff TA (1990) Responding to computer security incidents. Lawrence Livermore National Laboratory. ftp://ftp.cert.dfn.de/pub/docs/csir/ihg.ps.gz

Windows Xp (2016) Vulnerability statistics. http://www.cvedetails.com/product/739/Microsoft-Windows-Xp.html?vendor_id=26. Accessed 6 Feb 2016