Modeling the Vulnerability Discovery Process

O. H. Alhazmi and Y. K. Malaiya
Computer Science Department
Colorado State University
Fort Collins CO 80523
omar|malaiya @cs.colostate.edu

Abstract

Security vulnerabilities in servers and operating systems are software defects that represent great risks. Both software developers and users are struggling to contain the risk posed by these vulnerabilities. The vulnerabilities are discovered by both developers and external testers throughout the life-span of a software system. A few models for the vulnerability discovery process have just been published recently. Such models will allow effective resource allocation for patch development and are also needed for evaluating the risk of vulnerability exploitation. Here we examine these models for the vulnerability discovery process. The models are examined both analytically and using actual data on vulnerabilities discovered in three widely-used systems. The applicability of the proposed models and significance of the parameters involved are discussed. The limitations of the proposed models are examined and major research challenges are identified.

1. Introduction

The vulnerability discovery process is analogous to the process of finding defects in software during testing. Vulnerabilities are, after all, a class of software defects [20]. However, there are some remarkable differences.

Software testing conducted within an organization is a reasonably well-defined process. At the same time, the effort that goes into finding vulnerabilities can depend on the rising and falling share of the installed base. A significant fraction of the vulnerabilities are found externally. Some of the finders are experts in commercial security organizations. However, some of the finders may be potential "black-hat" individuals who may be tempted to use the vulnerabilities discovered for their own gain. It is considered acceptable for some known bugs to be present in the software; they are often not fixed until the next release. Nonetheless, the presence of a known and unremedied vulnerability is highly undesirable. The system developers need to release patches as soon as possible after a vulnerability is discovered. The presence of known vulnerabilities in operating systems and servers can represent an extremely high risk for some organizations such as banks, investment and brokerage houses and web-based merchants.

Just as it is not feasible for a software to be certified defect-free, it is not possible to identify and fix every vulnerability present in an operating system. The software developers and users need to be able to assess the risk posed by the vulnerabilities and must invest in effective counter-measures. It is now recognized that the risk can depend on the delay involved in developing and releasing a patch [6, 10]. A developer needs to allocate sufficient resources for continuous vulnerability testing and patch development to stay ahead of the hackers. The users need to invest in data safeguard mechanisms, intrusion detection and damage control. This investment must be proportional to level of risk involved.

Software reliability growth models [13, 17] have been used for characterizing the defect-finding process. Such models are used to assess the test resources needed to achieve the desired reliability level by the target date and are needed for evaluating the reliability level achieved. They can also be used to estimate the number of residual defects that are likely to be present.

There is a need to develop similar models for quantitative characterization of the security aspects of the software. There are two separate processes to be considered. The first is the vulnerability discovery process, while the second is exploitation of individual vulnerabilities discovered. In this paper we examine modeling the first process. While the two processes are distinct, evaluation of the overall risk should involve a

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 IEEE

joint consideration of both processes. Obviously a vulnerability needs to be discovered before it can be exploited. While those who attempt to exploit vulnerabilities may often be amateurs, those who discover vulnerabilities must have significant technical expertise.

In the security field, the *vulnerability exploitation models* (VEMs) were the first to be considered. Browne et al. [9] have examined the exploitation of some specific vulnerabilities and have presented a modeling scheme. Investigations on the modeling of the vulnerability finding process have recently been examined by a few researchers. We examine and evaluate these models, termed *vulnerability discovery models* (VDMs), in this paper. An evaluation of the risk would involve both characterization of vulnerability discovery process was well as their potential exploitation. This evaluation of the risk involves both VEMs and VDMs.

The first VDM model proposed by Anderson [5] is here termed the Anderson Thermodynamic (AT) model. The second termed the AML model, is a logistic model proposed by Alhazmi and Malaiya in [2] and investigated in [3]. Two trend models were examined by Rescola in [19]; a linear model and an exponential model; these are termed RL and RE respectively. Finally, we try to fit and test a new model LP, which is an application of a traditional Logarithmic Poisson reliability growth model [18] proposed by Musa and Okumoto. All of these can be termed time-based models since they consider calendar time as the main factor. An effort-based model has also been proposed by Alhazmi and Malaiya in [2]. It will not be considered in this paper since it uses a different approach that attempts to use test-effort as the main factor instead of calendar time.

Alhazmi and Malaiya [2] have examined applicability of the AML model to the Windows 98 and NT 4.0 cumulative vulnerability data and found the fit to be acceptable. On the other hand, Rescorla [19], using the non-cumulative vulnerability rate data, found the fit for the two models considered to be insignificant for three of the four systems under consideration and significant for Red hat Linux 6.2. The applicability of the model proposed by Anderson [5] has not yet been considered. No comparison of the VDMs has yet been done. This paper examines and compares all the proposed models in a uniform manner, using the same data sets.

Ideally a time-based model such as an SRGM or a VDM should fit the available data perfectly. However, a model is an approximation of actual behavior that may be subject to some minor factors which are not modeled, in addition to some statistical fluctuations. A model should be useful for making future projections and for identifying the current trends. Thus, it should attempt to model the longer term trend. It is desirable that a model have a simple interpretation of the parameters; however, such an interpretation is not always easily found. If a parameter has an interpretation, it may be possible to estimate the general range in which its value should lie. When curve fitting is iterative, it can be quite useful to start with a preliminary estimate of the parameters as an initial value. In this paper we briefly examine the basis of proposed models and evaluate the applicability of the models using actual vulnerability data.

Just as static models for defect density and fault exposure ratio can assist in use of the SRGMs, the metrics vulnerability density and vulnerability/defect ratio can be applied to complement and support VDMs. Alhazmi and Malaiya [2] have shown that for similar systems, the values of these attributes tend to fall within a range. Static metrics can be used to constrain parameter estimation during fitting [4].

This paper examines the proposed models and presents a comparison using actual data for vulnerabilities in three major operating systems. Statistical goodness of fit tests are used to examine how well models track the actual discovery process. In the next section we discuss the proposed models. In section 3 the data-sets used for evaluation and the methodology is described. In Section 4, the models' adequacy is examined; the section evaluates measures for goodness of fit and presents the findings. In Section 5, the results are further discussed. Finally the conclusions are presented and the future work is identified in the last section.

2. Vulnerability Discovery Models

Here we present a summary of the main features of the vulnerability discovery models (VDMs) proposed. Although some of them were not formally termed a "model," they are appropriate candidates for further examination. For uniformity, the models are presented in such a way as to give the cumulative number of vulnerabilities with time as the independent variable. A *vulnerability* is defined as "a defect which enables an attacker to bypass security measures" [20]. VDMs can describe the rate of vulnerability discovery, denoted by $\omega(t)$, or the cumulative number of vulnerabilities discovered, denoted by $\Omega(t)$. Note that $\Omega(t)$ *is* obtained by integrating $\omega(t)$ with respect to time.

Anderson Thermodynamic Model (AT): This model was originally proposed for software reliability in [8]. Later, Anderson applied it to vulnerabilities in [5]; Figure 1 shows a hypothetical plot of AT model for different values of k/γ and C. Let us suppose that

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 IEEE

there are $N(t)$ vulnerabilities left after t tests, and let the probability that a test fails be $\omega(t)$. The model assumes that encountering a vulnerability causes it to be removed and also that no bugs are reintroduced. Using an analogy from thermodynamics, Anderson argues that $\omega(t) \leq k / t$, where k is a constant. Arguing that equality should be a reasonable approximation, he finally arrives at the model

$$\omega(t) = \frac{k}{\gamma t}, \tag{1}$$

where γ is a factor that takes into account the lower failure rate during beta testing by the users compared with alpha testing. Since we want to compare cumulative models we integrate equation 3 to get the model in terms of the cumulative number of vulnerabilities given by the function $\Omega(t)$ as follows:

$$\Omega(t) = \int\left(\frac{k}{\gamma t}\right)dt$$

$$= \frac{k}{\gamma}\ln(t) + \left(\frac{k}{\gamma}\ln(C)\right)$$

where $\left(\frac{k}{\gamma}\ln(C)\right)$ represents the integration constant. Simplifying we get,

$$\Omega(t) = \frac{k}{\gamma}\ln(Ct), \tag{2}$$

where C is a constant introduced by the integration. Note that this $\Omega(t)$ in this model is not defined when t=0; hence we will only consider its applicability when $t \geq 1$. As t grows, $\Omega(t)$ grows with logarithmic increase. It should be noted that this model has a relationship to the well-known Logarithmic Poisson SRGM and the failure rate bound proposed Bishop and Bloomfield [7].

Alhazmi-Malaiya Logistic Model (AML): This model was proposed by Alhazmi and Malaiya in [2]. They have also proposed an effort-based model (AMEB); however, it is not comparable to the other models examined here. The AML model is based on the observation that the attention given to an operating system increases after its introduction, peaks at some time and then drops because of the introduction of a newer competing version; Figure 2 shows a hypothetical plot of AML model for different values of A, B and C. Thus the vulnerability discovery rate increases at the beginning, reaches a steady rate and then starts declining. The cumulative number of vulnerabilities thus shows an increasing rate at the beginning as the system starts attracting an increasing share of the installed base. After some time, a steady rate of vulnerability finding yields a linear curve. Eventually, as the vulnerability discovery rate starts dropping, there is saturation due both to reduced attention and a smaller pool of remaining vulnerabilities.

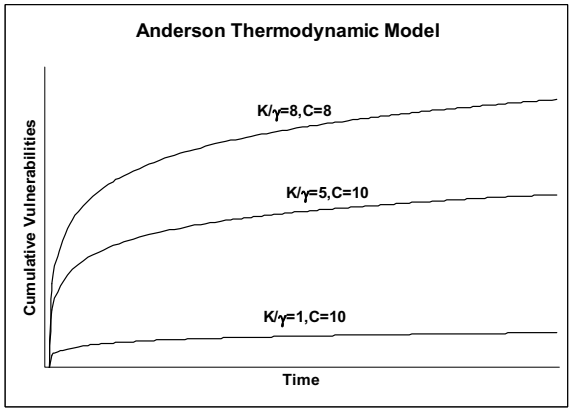

Figure 1 –Anderson Thermodynamic (AT) model

The model assumes that the rate of change of the cumulative number of vulnerabilities Ω is governed by two factors, as given in Equation 3 below. One of these factors declines as the number of remaining undetected vulnerabilities declines. The other factor increases with the time needed to take into account the rising share of the installed base. The saturation effect is modeled by the first factor. While it is possible to obtain a more complex model, this model provides a good fit to the data, as shown below.

Let us assume that the vulnerability discovery rate is given by the differential equation:-

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

where Ω is the cumulative number of vulnerabilities, t is the calendar time, and initially t=0. A and B are empirical constants determined from the recorded data.

By solving the differential equation we obtain

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

where C is a constant introduced while solving Equation 1. It is thus a three-parameter model given by the logistic function. In Equation 4, as t approaches infinity, y approaches B. Thus, the parameter B

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 **IEEE**

represents the total number of accumulated vulnerabilities that will eventually be found. The model given by Equation 4 will be referred to as the Alhazmi-Malaiya Logistic (AML) model [2].

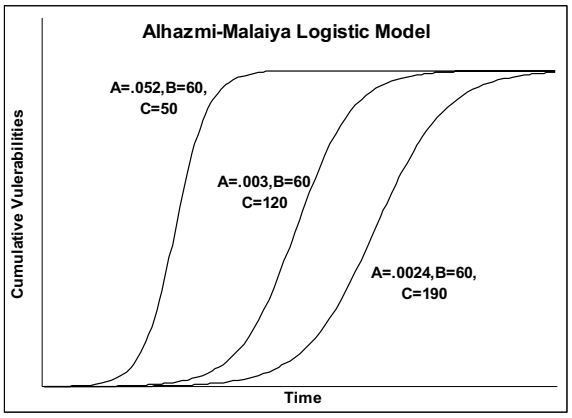

Figure 2 – Alhazmi-Malaiya Logistic (AML) model

This model addresses the fact that the vulnerabilities found in an operating system depend on its own usage environment. It should be noted that the saturation phase may not be seen in an OS which has not been present for a sufficiently long time. Also, if the initial adaptation is quick due to better prior publicity, in some cases early learning phase (when the slope gradually rises) may not be significant.

A non-linear regression may require an initial estimate of the parameter values. An initial estimate of B may be obtained by noting the size of the software and using the typical vulnerability density values of similar software. Using Equation 2, we can show that the maximum slope is given by $AB^2/4$, which occurs at y = B/2. It can also be mathematically shown that the two inflexion points in the derivative of Ω are 2.63/AB time period apart. This fact can be used guide parameter estimation during fitting [4].

Rescola Linear Model (RL): Rescola has attempted to identify trends in the vulnerability discovery data by applying some statistical tests. We here refer to these tests as the linear model and exponential model [19]; Figure 3 shows a hypothetical plot of RL model for different values of *B* and *K*.

First, we examine Rescorla linear model that attempts to fit the vulnerability finding rate linearly with time, rather than using the cumulative data. Vulnerabilities were grouped in 3-months-periods of time. The linear fit for the failure rate ω(t) implies the model:

$$\omega(t) = Bt + K \ , \qquad (5)$$

where B is the slope and K is a constant, while both are regression coefficients. The cumulative vulnerability discovery model can be derived by integrating (5):

$$\Omega(t) = \int (Bt + K)dt$$

$$\Omega(t) = \frac{Bt^2}{2} + Kt \ , \qquad (6)$$

here the integration constant is taken to be zero to allow Ω(t) to be zero at t =0. In this model as t grows, Ω grows polynomially, as given by the squared term.

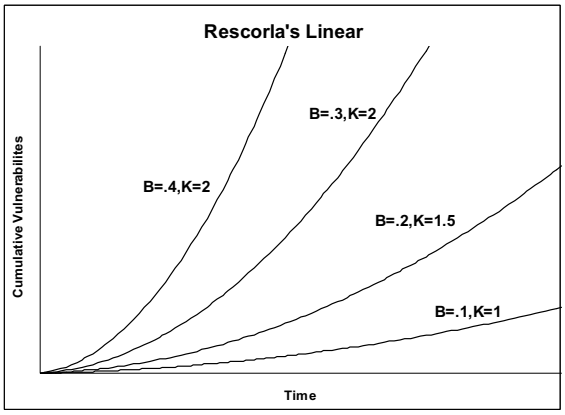

Figure 3 –Rescorla linear (RL) model

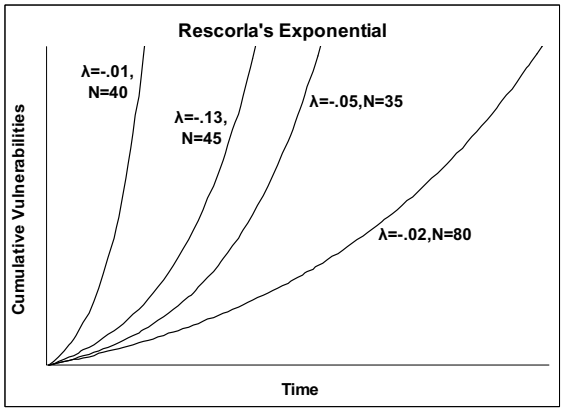

Figure 4 –Rescorla exponential (RE) model

Rescorla Exponential Model (RE): Rescorla [19] has also used Goel-Okumoto SRGM [11] to fit the data. This exponential model can be given as:

$$\omega(t) = N\lambda e^{-\lambda t}, \qquad (7)$$

where *N* is the total number of vulnerabilities in the system and λ is a rate constant. Again, we here integrate (7) to get the cumulative number of vulnerabilities. Figure 4 shows some plots of RE model for different values of λ, and *N*.

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 **IEEE**

$$\Omega(t) = \int N\lambda e^{-\lambda t}\, dt = N - Ne^{-\lambda t}$$
$$\Omega(t) = N(1 - e^{-\lambda t}), \quad (8)$$

where the integration constant has been equated to N to allow the initial value of Ω to be zero. Note that as time grows, Ω approaches N.

Logarithmic Poisson Model (LP): This model is also known as the Musa-Okomoto model [18]. A physical interpretation of the model and its parameters is complex. An interpretation in terms of the variability of the fault exposure ratio is given in [14]. Figure 5 shows various plots of LP model for different values of β_0 and β_1.

$$\Omega(t) = \beta_0 \ln(1 + \beta_1 t), \quad (9)$$

where β_0 and β_1 are regression coefficients. At t=0, Ω(t) = 0; Ω (t) grows indefinitely as the system ages with a logarithmic growth. In spite of the fact that the parameters have a complex interpretation, the model has been found to be among the better fitting SRGMs in many cases.

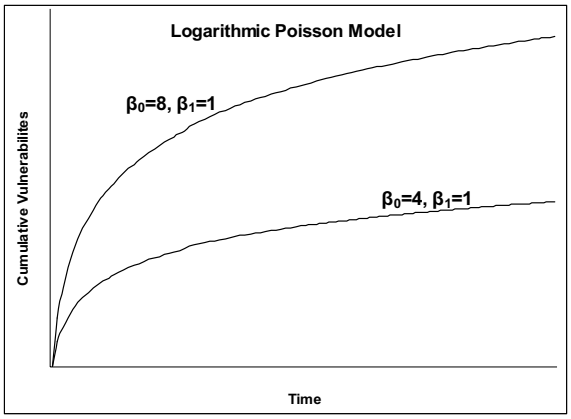

Figure 5 –Logarithmic Poisson (LP) model

Some of the models are somewhat related. The AT and LP models are given by rather similar expressions, with the significant difference that AT is undefined at t = 0. It can be shown that RE and LP may yield similar short term projections, but they differ significantly for very large values of t.

3. Methodology for Model Evaluation

Here we discuss how the data was collected and prepared for fitting, and then we describe how the goodness of fit was evaluated.

The data sources: Compared to data that has been used for SRGMS in the past, vulnerability data demonstrates some different characteristics. One of the main differences is that generally no information is available concerning the faults in SRGM data, whereas for vulnerabilities the data bases identify the specific vulnerability. The vulnerability data comes from well-known products, since the data for every operating system and server, both commercial and open-source, are available. The SRGM data comes only from some selected projects where the management has permitted disclosure of the data. On the other hand, vulnerability data has some limitations—namely, that it comes from a limited number of sources, and the number of vulnerabilities typically represents only a small fraction of the total number of defects.

Table 1: The data sets used

Systems	Lines of code (millions)	OS Type	Known Vulnerabilities	Vulnerability Density (per Ksloc)	Release Date
Windows 95	15	Commer-cial client	50	0.0033	Aug 1995
Windows XP	40	Commer-cial client	88	0.0022	Oct 2001
R H Linux 6.2	17	Open-source server	118	0.00694	Mar 2000

Vulnerability data needs to be manually extracted from data bases. Our major sources of data are the Mitre Corporation [16] website and the ICAT [12] metabase. ICAT is an easily searchable data base with the option of downloading an ACCESS database.

Evaluation of goodness of fit: We will apply two goodness-of-fit tests. The first is the chi-square goodness of fit test. The chi-square (χ^2) statistic is calculated as follows:

$$\chi^2 = \sum_{i=1}^{n} \frac{(o_i - e_i)^2}{e_i}, \quad (10)$$

where o_i is the observed value and e_i is the model's expected value.

For fit to be acceptable, the chi-square statistic should be less than the critical value for a given alpha level and the degrees of freedom. The P-value is the probability that a value of the χ^2 statistic at least as high as the value calculated by the above formula could have happened by chance. We use an alpha level

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 **IEEE**

of 5%; i.e., if the P-value of the chi-square test is below 0.05 then the fit will be rejected. A P-value closer to 1 indicates a better fit. P-value is calculated by using the number of degrees of freedom of the data set and chi-square distribution.

For model adequacy testing *Akaike Information Criteria (AIC)* [1] is also frequently used. AIC is used to make a fair comparison between the models. AIC is formally defined as

$$\text{AIC} = (-2 \times \log \text{likelihood}) + 2M$$

An equivalent way to compute AIC is

$$\text{AIC} = T\, ln(\text{RSS}) + 2M, \quad (11)$$

where M is the number of free parameters of the examined model, T the number of observations, and RSS the residual sum of squares. We use the formulation of AIC given by Equation 11. The Akaike

In our analysis, we have used the data sets for three significantly different operating systems. The vulnerability data of Windows 95 is for a client operating system that has existed for several years. It has gone through nearly a complete life-cycle and its remaining installed base is now very small. The data set for Windows XP represents a relatively new operating system which may be near the peak of its popularity. We have also included data for Linux Red Hat 6.2 which represents an open-source operating system. Some of the key attributes of the three systems are given in Table 1, [3, 12, 15, 16]. Windows XP is much bigger than Windows 95; however, its vulnerability density is comparable. It is likely that significant number of yet undiscovered vulnerabilities is present in Windows XP. The higher number of vulnerabilities in RH Linux 6.2 may be due to the fact that a larger fraction of the software's functionality may be devoted to access control.

4. Fitting Data to Proposed Models

The results of fitting the models to the data is presented graphically in the plots given in Figures 6,7 and 8, which show the fitted plots along with actual cumulative data. The parameter values obtained during the fit, and the corresponding measures of goodness of fit for the three operating systems' sets, are given in Tables 2, 3 and 4. First we discuss the results for each model individually and afterwards each software system's datasets.

The Windows 95 data (Figure 6) has a distinct s-shape due to the fact that vulnerability detection reached saturation some time ago. As we would expect, the AML model fits quite well. The fitted AT model gives negative values at the beginning and significantly diverges from the actual data, except near the end, that is why AT's chi-square value is incomparable with the other models. The fitted RE, RL and the LP models generate linear plots and thus show considerable divergence at the end.

The Linux Red Hat 6.2 data (Figure 7) shows a milder s-shape, permitting most models except AT to fit reasonably well. The data for Windows XP (Figure 8) show a very linear trend, allowing LP, RE and RL to fit quite well. This is likely to change when the vulnerability discovery rate for Windows XP eventually saturates. The AT model again has a problem with fitting during the initial stage and again in the last stage.

For the Linux Red Hat 6.2 data, LP, RL, RE and AML models successfully fitted with P-values ranging from 0.915 to 0.998 (Table 3). However, AML gave a better AIC test score of 429.89, while the others yielded values close to 469.

Table 4 shows that for the Windows XP dataset LP, RE, RL and AML successfully fit the data with P-values ranging between 0.918 to 0.99997. However, the AT model did not fit the dataset. The RL model has the lowest AIC of 249.4, while for the other models the value ranged from 273.4 and 373.19. The reason that the AML model was not better for Windows XP is that it is relatively new operating system and the saturation phase has not yet been reached. The RL model fits the Windows XP data very well.

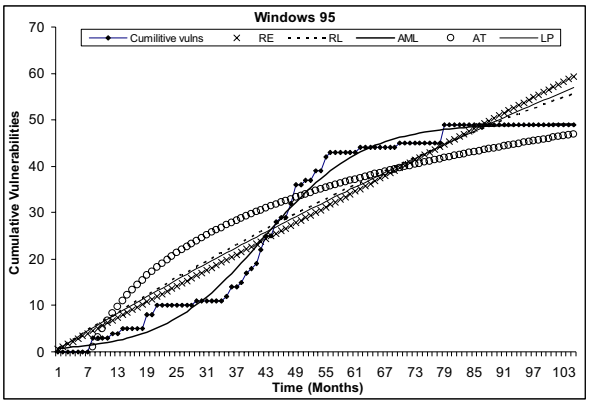

Figure 6 –Fitted VDMs with actual Windows 95 data

Comparative performance of the models: Here we examine the performance of each individual model.

The Anderson Thermodynamic (AT) model was unable to fit any of the data, exhibiting the highest AIC scores and lowest P-values. For the Windows 95 data, with an AIC of (947) it is 100 points away from the

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 **IEEE**

nearest model. For Linux 6.2 it has demonstrated an AIC of (572.79), which is 103 points. For Windows XP it was about 85 points higher than the closest model.

Rescorla linear (RL) model was able to fit two out of three of the datasets under consideration. RL was not able to fit the Windows 95 data (see Figure 6) because of the strong S-shaped trend of the data; at best, the model was stretched as linear as possible. Thus, it scored poorly in the chi-square test with a P-value of only 0.0035, whereas it should have been at least 0.05 to be acceptable. For Linux 6.2 it achieved a P-value of 0.915, which is a very good fit. It fits the Windows XP data very well with a P-value of 0.99997.

Table 2- Windows 95 goodness of fit results

Model	Parameters			χ^2 –test		RSS	AIC
				χ^2	P-Value		
*AT	K/ γ	C		222.3	2.14×10^{-12}	7961.2	947.14
	17.795	2.36					
LP	β_0	β_1		144.72	1×10^{-5}	3095	847.94
	157.63	4.14×10^{-3}					
RE	λ	N		135.82	0.00198	3300.3	854.69
	0.0001	5629.7					
RL	S	K		147.07	0.0035	3006.2	844.88
	-0.00137	0.674					
AML	A	B	C	48.4	.999999	416.27	639.3
	0.002	49.15	1.41				

*Chi-square test was applied to the positive values for the AT model.

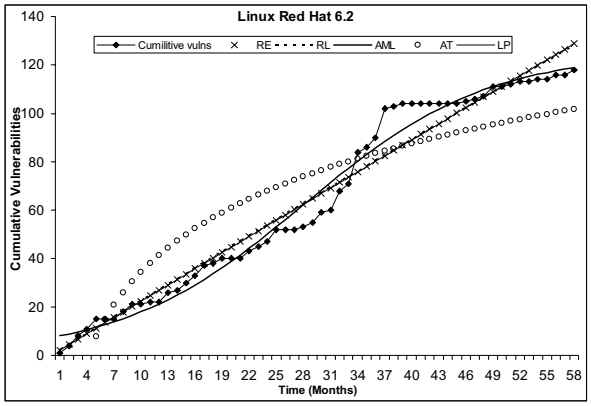

Figure 7 – Fitted VDMs with actual Red Hat Linux 6.2 data

The Rescorla Exponential (RE) model was able to fit two out of three of the data sets considered. RE was not able to fit the Windows 95 data (see Figure 6) because of the prominent S-shape of the data set curve, the fit was judged to be poor in the chi-square test with a P-value of only 0.0198, which is significantly less than the 0.05 that is considered acceptable. For Linux 6.2 and Windows XP the P-values were 0.925 and 0.97 respectively, a very good fit in both cases. However, RE scored 854.69 in the AIC test for Windows 95, which is higher than other models except AT. For Linux Red Hat 6.2, the AIC was 469.82 which is close to the values for LP and RL, although larger than the value for AML. For Windows XP, again it was with a high AIC of 288.58.

The Alhazmi-Malaiya Logistic Model (AML) is the only model that fits all three data sets well. The fit was especially superior for the Windows 95 and Linux Red Hat 6.2 data sets, where it gave the best AIC results. For Windows 95, AML has an AIC value of 639.3, which is much less than the other models which have scored between 844 and 947.14. The AML model fits the data with an excellent P-value very close to 1. The other four models failed the goodness of fit test for Windows 95, since they generated unacceptably high chi-square values and consequently low P-values below 0.02. AML was the best model in RedHat Linux 6.2 data set, with a low 429.89 on AIC with a 40 points better than the nearest model. Although AML fits well with Windows XP data, it is able to score only a modest 273.34 on the AIC. The RL model fit the Windows XP data better due to the fact that the Windows XP data has is very linear, which gives it an advantage over the AML model. We would, however, expect the Windows XP data to eventually saturate. The results show that the AML is the most consistent model for the three data sets.

Table 3- Fitting Results for Red Hat Linux 6.2

Model	Parameters			χ^2 –test		RSS	AIC
				χ^2	P-Value		
*AT	K/ γ	C		173.6	1.1×10^{-20}	18156.6	572.79
	38.3	9.39					
LP	β_0	β_1		38.9	0.998	3077.6	469.85
	6999.9	3.2×10^{-4}					
RE	λ	N		38.98	0.925	3076.18	469.82
	0.0004	5629.7					
RL	S	K		40.02	0.915	3069.8	469.7
	-0.0013	2.2816					
AML	A	B	C	32.43	0.996	1492.96	429.89
	0.0008	125.2	0.128				

*Chi-square test was applied to the positive values for the AT model

The Logarithmic Poisson (LP) model was able to fit the Linux 6.2 data with a P-value of 0.998 and Windows XP with a P-value equal to 0.9654. However, it failed to fit the Windows 95 data set with a low P-value of 1×10^{-5}. On the AIC scale for Linux 6.2, it scored 469.85, which is close to the score of the RL and RE models and better than the AT model. For

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 **IEEE**

Windows XP, the AIC was 284.49, better than the RE model and worse than the RL and AML models.

Both RE and RL demonstrate a good fit for a short term. Both the models are flexible and can adjust to a linear trend with an adjustment of their regression parameters. However, they may not work very well for the long term. The RE model is somewhat better than RL with the ability to capture the saturation in the later phase. On the other hand, the S-shaped AML model has shown excellent fit, especially for the long term. Furthermore, the AML model is flexible enough to also fit short term data.

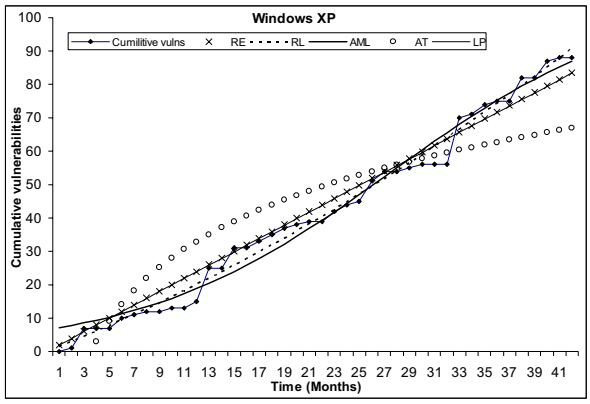

Figure 8 – Fitted VDMs with actual Windows XP data

Table 4- Fitting results for Windows XP

Model	Parameters			χ^2 –test		RSS	AIC
				χ^2	P-Value		
*AT	K/ γ	C		115	1.1×10^{-9}	6568.3	373.19
	27.1	7.61					
LP	β_0	β_1		26.15	0.9654	757.9	284.49
	7211.85	0.000277					
RE	λ	N		26.44	0.955	761.8	284.7
	0.00035	5628.8					
RL	S	K		13.82	0.99997	328.7	249.4
	0.0162	1.482					
AML	A	B	C	29.12	0.918	581.69	273.37
	0.001	107.5	0.144				

*Chi-square test was applied to the positive values for the AT model

5. Results and Observations

Of the five models examined, three of them—RL, RE and LP, appear to do a good job of following the shorter-term trends, especially when the cumulative vulnerabilities show a linear trend. The AT model generally does not appear to fit the actual data well. The AML model often provides the best fit since it can follow the S-shaped trend that is observed in some cases.

The AML model uses three parameters, while the others are two-parameter models. However, both the measures, the P-value and AIC, take the number of parameters into account and thus provide a fair basis for comparison.

Among the five models, the parameters of the RE and AML models have some simple interpretations. One of the parameters in both is related to the total number of vulnerabilities present in the software. If the expected range of vulnerability density values can be estimated based on past experience, a preliminary estimate of the total number of vulnerabilities may be empirically obtained [3]. However, empirical estimation of other parameters requires further investigation.

The evaluation presented here is based on three operating systems that represent two commercial systems in different life-cycle phases and one open-source software system. These observations need to be further validated using data from other systems.

The vulnerability data is generally of higher quality than that used for evaluating SRGMs in the past. However, there are some significant issues to keep in mind. One is that vulnerabilities are sometimes reported in batches, which affects the accuracy of the time a vulnerability is recorded. If cumulative data is being used for analysis, this will cause some fluctuations but should not change the longer-term trend present in the data.

ICAT metabase can have some ambiguity in identifying a specific version of a software system. Consequently, researchers need to check details of vulnerabilities and use some of the ICAT references for additional verification. Sometimes duplicate entries are encountered. Rescorla has mentioned some of these irregularities in [19].

Since data bases only include the publicly known vulnerabilities, for proprietary software systems we may not know about unpublished vulnerabilities. However, with external experts competing to discover vulnerabilities, it is likely that disclosure of vulnerabilities will be carried out as soon as a patch is available. For open-source systems, the discovered vulnerabilities are usually available in the public domain.

Vulnerabilities are shared among successive versions due to code reuse. This can raise the question of where the vulnerability was discovered—in the new version or in the older version. Sometimes vulnerabilities in the ICAT metabase are discovered prior to the specified version's release date, a problem that can be overcome by filtering omitted pre-release date vulnerabilities.

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 **IEEE**

The AT, RL, RE and LP models assume a stable level of testing effort over time. However, it would be reasonable to assume that the extent of the testing effort is driven by the cost/reward factors. Finding vulnerabilities in a widely-installed system should be more rewarding to internal testers, who would put in extra effort into minimizing the probability of exploitation in the popular systems. The external experts and hackers would find it more rewarding to find vulnerabilities in systems that are currently popular. Thus the testing effort varies with changes in the market share of software systems. This variability of effort was addressed in [2] by developing an equivalent effort model. The model addresses changes in the usage environment, which affects the discovery process. The model considered the time spent by the number of workstations using the software systems. The equivalent effort model fit very well but the model requires data that can be very hard to collect. The AML model attempts to implicitly model the effort variation.

6. Conclusions and Discussion

This paper presents an examination of the applicability of recently proposed quantitative vulnerability discovery models to actual data. Systematic software testing in a developing organization occurs prior to release and the bugs are found internally. On the other hand vulnerability discovery occurs throughout the product lifetime and the vulnerabilities are found both internally and externally. Moreover, compared with ordinary bugs, the number of vulnerabilities is very small. This raises a question about applicability of quantitative models, such as the SRGMs that have been successfully used. The results presented here show that some of the proposed models fit the actual vulnerability discovery process very well.

Four vulnerability discovery models were examined using Akaike Information Criteria (AIC) and chi-square (χ^2) tests. Results show that the AML model is generally the best for the longer term, performing better for Windows 95 and Linux 6.2. Because it captures the S-shape pattern in the data, it has better fit as determined by using AIC and the chi-square test. RL, RE, LP and AML all show very good fit for Windows XP, a system that has not yet shown signs of saturation. This can be attributed to the fact that they can fit trends that are largely linear. The AT model considered did not perform well in general.

These vulnerability discovery models ignore the architecture of the software system. They assume that vulnerabilities are found in random places within software systems. In some specific modules, exploiting information concerning the architecture of the system can focus security testing to the code with higher chance of finding vulnerability, which will make security testers more productive.

We have analyzed the models statistically and analytically. However since models with a good fit will not necessarily have good estimation capability, there is a need to examine the ability of the models to estimate the future vulnerability discovery rate. In a concurrent work [4], the prediction capabilities of two of the VDM models are examined. The results show that the prediction capability gets better as more data becomes available. The investigations show that putting some constraints on the models' parameters based on previous observations significantly improves the prediction capability. Examination of the prediction capabilities of the other models is still needed to show broader comparison of the other vulnerability discovery models.

VDMs can be termed dynamic models. It is possible to define and use static metrics that impact the parameter values in such models. Metrics such as vulnerability density and vulnerability/defect ratios [3] can be used to check the projections made using VDMs during early phases when the available data is insufficient, or when a VDM is known to have some specific limitations.

Vulnerability discovery models can be used by both the developers and the user community. Developers can assess the product readiness by projecting future vulnerability discovery trends. Developers need to allocate security maintenance resources to detect vulnerabilities, preferably before others do; and to release security patches as soon as possible. The users also need to assess the risk due to vulnerabilities before patches are applied. A patch may need to be tested for stability before it is applied as discussed by Brykczynski et al. [10] and Beattie et al. [6]. Effective security policy at an organization would require time and resources. Vulnerability discovery models can be used to quantitatively guide such policies.

The models considered have some limitations, namely, that they treat all vulnerabilities equally, even though some vulnerabilities may represent higher severity levels. Moreover, the models do not address code re-use and overlap among consecutive versions of software systems. Vulnerability discovery models assume that each specific release of an operating system is independent, and can be separately modeled. In practice, a significant sharing of the code occurs between successive releases. Thus a vulnerability detected in a version may also exist in previous versions. Further research is needed to model the impact of such shared code.

Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)
1071-9458/05 $20.00 © 2005 IEEE

7. References

[1] H. Akaike, Prediction and Entropy, MRC Technical Summary Report #2397. NTIS, Springfield, VA., 1982.

[2] O. H. Alhazmi and Y.K. Malaiya,, "Quantitative vulnerability assessment of systems software," Proc. Annual Reliability and Maintainability Symposium, 2005. pp. 615 – 620.

[3] O. H. Alhazmi, Y. K. Malaiya and I. Ray, "Security Vulnerabilities in Software Systems: A Quantitative Perspective," Proc. Ann. IFIP WG11.3 Working Conference on Data and Information Security, Aug. 2005.pp. 281-294.

[4] O. H. Alhazmi, Y. K. Malaiya, "Prediction Capability of Vulnerability Discovery Models", to appear in Proc. Reliability and Maintainability Symposium, January 2006.

[5] R. J. Anderson, "Security in Opens versus Closed Systems—The Dance of Boltzmann, Coase and Moore," Open Source Software: Economics, Law and Policy, Toulouse, France, June 20-21, 2002.

[6] S. Beattie, S. Arnold, C. Cowan, P. Wagle and C. Wright, "Timing the Application of Security Patches for Optimal Uptime," Proc. LISA XVI, November 2002. pp 233-242.

[7] P. G. Bishop and R. E. Bloomfield, "A Conservative Theory for Long-Term Reliability Growth Prediction," IEEE Trans. Reliability, Vol. 45, No. 4, Dec. 1996. pp

[8] R. M. Brady, R. J. Anderson, R. C. Ball, "Murphy's law, the fitness of evolving species, and the limits of software reliability", Cambridge University Computer Laboratory Technical Report No. 471 (September 1999), available at http://www.cl.cam.ac.uk/ftp/users/rja14/babtr.pdf.

[9] H. K. Browne, W. A. Arbaugh, J. McHugh, W. L. Fithen, "A Trend Analysis of Exploitation," Proc. IEEE Symposium on Security and Privacy, 2001, May 2001, pp. 214-229.

[10] B. Brykczynski and R. A. Small, "Reducing Internet-Based Intrusions: Effective Security Patch Management,", IEEE Software, Vol. 20, No. 1, Jan-Feb 2003, pp. 50-57

[11] A. L. Goel and K. Okumoto, "Time-Dependent Error Detection Rate Model for Software and Other Performance Measures," IEEE Trans. on Reliability, R-28, 3, August 1979, pp. 206-211.

[12] ICAT Metabase, http://icat.nist.gov/icat.cfm, February 2004.

[13] M. R. Lyu, Ed., Handbook of Software Reliability Engineering, McGraw-Hill, 1995.

[14] Y.K. Malaiya, A. von Mayrhauser, P.K. Srimani, "An Examination of Fault Exposure Ratio," IEEE Trans. Software Engineering, Nov. 1993, pp. 1087-1094.

[15] G. McGraw. "From the Ground Up: The DIMACS Software Security Workshop," IEEE Security & Privacy. Volume 1, Number 2, March/April 2003, pp. 59-66.

[16] The MITRE Corporation, www.mitre.org, February 2005.

[17] J. D. Musa, Software Reliability Engineering, McGraw-Hill, 1999.

[18] J.D. Musa and K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement," Proc. 7th International Conference on Software Engineering, Orlando, FL, 1984, pp. 230-238.

[19] E. Rescola, "Is finding security holes a good idea?" Security and Privacy, Jan-Feb 2005. pp. 14-19.

[20] E. E. Schultz, Jr., D. S. Brown and T. A. Longstaff, "Responding to Computer Security Incidents," Lawrence Livermore National Laboratory, ftp://ftp.cert.dfn.de/pub/docs/csir/ihg.ps.gz, July 23, 1990.