Efficient Stopping Rules for Quantitative Security Testing of Cyber-Attacks

Susan J. Simmons
UNCW, Department of Mathematics and Statistics
Wilmington, NC 28411, USA
Email: simmonssj@uncw.edu

Mehmet Sahinoglu
Auburn University Montgomery, Department of Cybersystems and Information Security
Montgomery, AL 36124, USA
Email: msahinog@aum.edu

James H. Matis
Texas A & M University, Department of Statistics
College Station TX 77840, USA
Email: matis@stat.tamu.edu

ABSTRACT

A new challenge to software testing lies in the concept of a monitored security testing and most essentially in the determination of an epoch as to when to stop testing. Under minimum assumptions regarding growth of failures or breaches, due to chance (reliability) or malicious (security) reasons, we can objectively define an appropriate stopping rule to timely avoid further damage saving resources with a cost efficient plan. This research topic opens new avenues in a very critical area of cyber-security defined to be quantitative stopping rules in security testing as compared to existing conventionally qualitative rules which do not lend themselves to probabilistic and cost-effective reasoning. We employ two probabilistic models to determine appropriate stopping rules and compare the approaches using two well-known data sets known as DR 4 and DR 5 (Sahinoglu, 2007).

1. INTRODUCTION

The damage inflicted by security breaches and software failures in computer and communication networks as experienced by related businesses or government entities is measured by multiples of billions of dollars. The analysis of such malicious activities as to when to act at the right moment to assure cost efficiency and maximum security are of a paramount interest to computer scientists and risk analysts, in addition to the business owners and their customers. In most situations, testing continues until the time-to-release date or the budget is depleted. This conventional subjective stopping decision inhibits the testers from understanding the extent of potential security breaches or failures when the product is released and can be extremely costly and inefficient. Herein, we consider two methods defining appropriate stopping rules in security testing, the logistic growth model (LGM) and the compound Poisson process model (CPM). These two methods model failure times based on probabilistic models and develop criteria-based stopping rules to support each other in synergy.

There is another aspect of software security testing which deals with the functional testing of secure software (as in the metaphor of walking a high wire with a safety net), an entirely different domain and conceptually different than what this research paper addresses. The two common methods for testing whether software has met its security requirements are functional (Allen, Barnum, Ellison, McGraw, Mead, 2008) and risk-based security testing (McGraw, 2006). The methods proposed herein follow the latter risk-based testing derived from a risk analysis to encompass not only the high-level risks identified during the design process but also low-level risks derived from the software itself.

2. METHODS

The LGM was originally defined by Verhulst (1845) and used to model population growth of species for many years (Larralde-Corona et al., 1997; Matis *et al.*, 2009; Piegorsch and Bailer, 2005; Simmons, 2008; Simmons *et al.*, 2009). The LGM has also been used to model software failures (Yamada *et al.*, 1984). However, this approach has not been previously used to quantitatively define an appropriate stopping rule for testing or security. This approach uses a coverage-based criterion to determine an appropriate stopping rule. The second method, referred to as CPM, models failure times as a Poisson process with clusters following an LSD distribution. The model is referred to as a compound-Poisson model and utilizes three criteria in its stopping rule. The three criteria are coverage, expected cost of continued testing and amount saved if testing stops. Sections 2.1 and 2.2 describe each approach in greater detail.

2.1 Logistic Growth Model

The LGM can easily be fit by many statistical software packages. This approach models the cumulative failures or breaches of a system as a logistic growth curve, which is defined as

$$f(x) = \frac{\beta_0}{1+e^{-\beta_1-\beta_2 x_i}} \ .$$

In this parameterization of the LGM, the parameter β_0 is the maximum cumulative number of failures or breaches in the system. Looking at the rate in which failure or breaches occur, the maximum rate occurs at the inflection point of the curve, which is at $-\beta_1/\beta_2$. This method does not require researchers to reach the maximum cumulative number of failures or breaches in order to estimate the parameters in the model. Once an LGM is fit to the data, we can estimate the value where $100q\%$ of the failures or breaches occurs, for $0 < q < 1$. For example, Figure 1 illustrates a simulated LGM data set generated with $\beta_0 = 50$, β_1=-2 and $\beta_2 = 0.65$ using a simple standard normal error structure. The true maximum failure or breach rate occurs at $x = 3.077$ (- (-2/0.65)) and the maximum cumulative number of failures or breaches is 50 (β_0).

In the process of testing software, the full curve would not be known (unless testing continued until all failures or breaches occur, in this instance a total of 50 breaches or failures was found). As a simple example, say the researchers stop at $x = 3.00$ (before the maximum growth rate occurs), and it is of interest to estimate where the maximum growth rate occurs and the maximum cumulative number of failures or breaches. Using this truncated data and the SAS PROC NLIN procedure with initial estimates obtained by using Piegorsch and Bailer (2005), the following estimates were obtained for β_0, β_1 and β_2

$$b_0 = 49.6963,\ b_1 = -1.9794,\ b_2 = 0.6510.$$

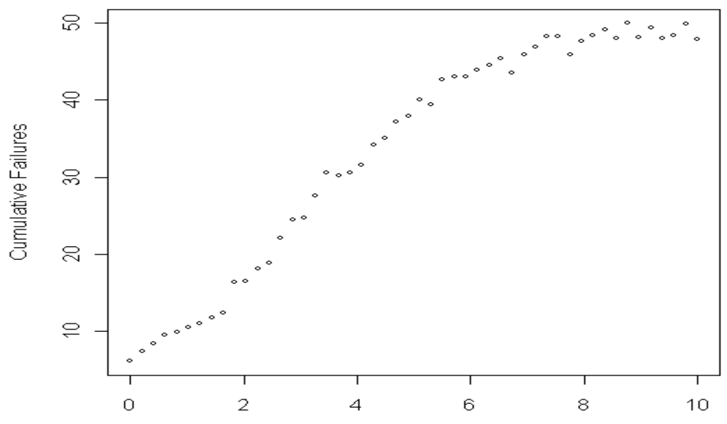
These estimates give an approximate maximum growth rate at 3.04 (compared to the truth at 3.077) and an approximate maximum cumulative number of failures of 50 (rounding the β_0 estimate). By using the estimated growth curve, we can utilize a stopping rule based on the desired percent of coverage. For example, if a coverage of 0.8 is desired, we need to find the value of x associated with at least 80% coverage or $0.8b_0 = 39.75704$. Solving the following equations for x,

$$39.75704 = \frac{49.6963}{1 + e^{1.9795 - 0.6519x}}$$

gives a stopping rule of $x = 5.126$. In general, a stopping rule with $100q$ coverage ($0 < q < 1$) is

$$\frac{log\left(\frac{1-q}{q}\right)+b_1}{-b_2}.$$

Figure 1. Simulated software failure (or security breach) cumulative count data (y) versus time (x)



2.2 Compound Poisson Model

An innovative method incorporating the cost of testing and the discrete nature of the data is the compound Poisson method (Sahinoglu, 1992; Sahinoglu and Can, 1997; Sahinoglu, 2003; Sahinoglu, 2007). In this method, the total number of failures or breaches is assumed to follow a Poisson process with a compounding distribution that describes the failure size. This method assumes that the total number of failures, $X(t)$ follow a Poisson process, where

$$\{X(t), t \geq 0) = \sum_{i=1}^{N(t)} w_i$$

with $N(t) > 1$. The failure size, denoted as $w_1, w_2, \ldots$, are independent, identically distributed logarithmic series distribution (LSD) with

$$f(w) = a\frac{\theta^w}{w}; \;\; 0<\theta<1, \;\; a>0, \;\; w=1,2,...$$

It can be shown that the compound Poisson distribution with LSD as the compounding distribution is equivalent to the negative binomial distribution with $E(X) = kp$ and

$$\lambda = -k\ln(1-\theta) = k\ln q \tag{1}$$

$$q = \frac{1}{1-\theta}$$

where $p=q-1$ and $\lambda = E(N(t))$ where $N(t) \sim$ Poisson(λ). The parameter θ represents the autocorrelation within each clump or failure size (correlation within each clump size is assumed to be positive). Since the autocorrelation within each clump size is not constant, the parameter θ can be treated as a random variable. An empirical Bayes approach is used where the prior for θ is Beta(α,β). Then, $\theta|X\sim$ Beta($\alpha + X, \beta + k$) and

$$E(X) = k\,\frac{\alpha + X}{\beta - 1 + k} \tag{2}$$

In Sahinoglu (2007), under the assumption of a compound Poisson distribution, three criteria describe when to stop testing. First is the one-step-ahead approach which stops testing when the i^{th} incremental step at time t has a larger (or equal to) expected cost of stopping than expected cost of continuing, or in other words (Randolph and Sahinoglu, 1995)

$$aE(X_{i+1}) \le bE(X_i) + c,$$

where a = cost of fixing a failure once the software has been released, b = cost of fixing software during testing and c = a fixed value for testing. When $E(X_{i+1}) = E(X_i)$, the above formula simplifies to $E(X_{i+1}) - E(X_i) \le \frac{c}{a-b} = d$, and we can rewrite the above criteria as

$$e(X) = E(X_{i+1}) - E(X_i) \le d,$$

Substituting from equation (2) above leads to a stopping criterion of

$$e(X) = k_{i+1}\,\frac{\alpha + X_{i+1}}{\beta - 1 + k_{i+1}} - k_i\,\frac{\alpha + X_i}{\beta - 1 + k_i} \le \text{d}. \tag{3}$$

The second criterion is to ensure that a certain level of branch coverage is obtained. In order to ensure that a certain amount of coverage is obtained, the total number of failures or breaches must be known or estimated. When the total cumulative number of breaches are known, the 100q% coverage is obtained when q*(total cumulative number of failure or breaches) for $0 < q < 1$ has been reached. The third and final criterion used to define when to stop testing is that a savings is positive when the testing is stopped. The program **M**ath-Statistical Cost-**E**fficient **S**topping-Rule **A**lgorithm for **T**esting, or MESAT-1 (Sahinoglu, 2007) is an easy-to-use software developed to implement the compound Poisson distribution with the above three criteria for stopping. The MESAT software calculates a rule for stopping the testing activity of failure/attack in software/security testing, as contrary to exhaustive testing, given the input parameters including the cost factors, both for effort-based (time-independent or discrete) and time-continuous. The inputs required to use the MESAT-1 software are

1. Values for α and β in the Beta distribution.
2. Range of θ.
3. Initial value of k(0).
4. Coverage criterion.
5. Number of coverage.
6. Minimum number of test cases.

The values for α and β relay the information about the prior distribution for θ, the autocorrelation of failure or breach size. We recommend using a left-skewed distribution indicating larger autocorrelation ($\alpha = 8$ and $\beta = 2$). The difference of the θ input (indicating the total range of the autocorrelation values) should remain 1, unless there is good evidence that an improper beta should be used as the prior distribution for θ, which would change the range to a value less than one. The value of d should be derived from the previously stated values of a, b and c. The value of $k(0)$ is an initial estimate for k as defined in equation (1). The k parameter in the model is not very sensitive to the initial starting value of k(0). We recommend using the value 0.12 (Sahinoglu, 2007) for $k(0)$. The number of coverage is the maximum cumulative number of failures or breaches and is automatically inputted from the data set. The minimum number of test cases defaults to 0, unless there are a pre-specified number of minimum test cases required. In addition to the previous input, the information for the cost analysis must also be entered. This includes the values of a, b and c, and if there is a budget to the testing, this amount should also be factored into the analysis.

If we were to stop at a discrete interval i, we would assume that the failures or branch coverage discovered will have to accrue in the field a cost of „a' per failure or branch after the fact or following release of the software. Thus, there is an expected cost over the interval $\{i, i + 1\}$ of $aE\{X_{i+1}\}$ for stopping at time $t = t_i$ or test case i. If we continue testing over the interval, we assume that there is a fixed cost of c for testing and a variable cost of b of fixing each failure found during the testing before the fact or preceding release of the software. Note that „a' is almost always larger than b since it should be considerably more expensive to fix a failure (or recover an undiscovered branch) in the field than to observe and fix it while testing in house. Thus, the expected cost for the continuation of testing for the next time interval or test case is $bE(X_i) + c$. Opportunity or shadow cost is not considered here since such an additional or implied cost may be included within a more expensive and remedial after-release cost coefficient denoted by a. Some researchers are not content with these fixed costs and argue that these values change as testing continues. However, the MESAT-1 tool employed here can accommodate this problem through a variable costing data-driven approach as needed by the testing analyst. That is, a separate value is entered at will in the MESAT-1 Java program for a or b or c at each test case, if these cost factors are defined to vary from test case to test case.

In order to ensure a stopping rule is cost efficient (Sahinoglu, 2003 and 2007), we recommend considering the following inequalities. Let RF= number of failures remaining after the stopping rule, and RT = number of test cases after the stopping rule. Then a cost-efficient stopping rule is one in which **(**RF)$a \leq$ (RF)b + (RT)c. From this equation, the following inequalities are derived:

$$a \leq \frac{(RF)b + (RT)c}{RF}$$

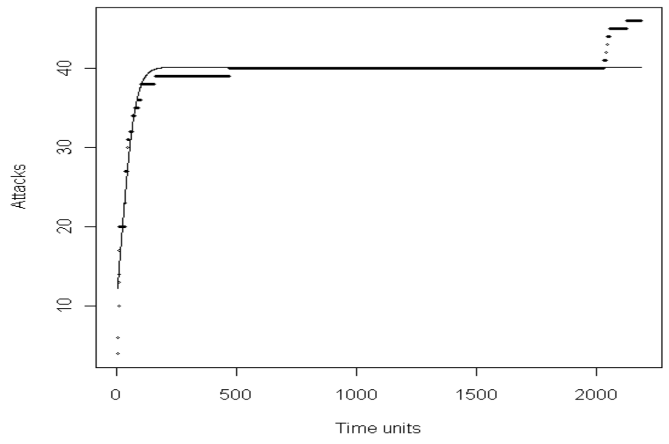
$$b \geq \frac{(RF)a - (RT)c}{RF}$$
$$c \geq \frac{(RF)a-(RF)b}{RT}.$$

3. EXAMPLES USING BOTH STOPPING RULES

The DR5 Data Set

We illustrate the two methods on a well-studied data set, DR5 (Sahinoglu, 2007). This data set has 2187 time units and 46 attacks. Figure 2 illustrates the data with the logistic growth curve overlaid on top of it. The logistic growth model estimated the parameters as β_0=40.1980, β_1= -0.8643, and β_2 = 0.0366, giving the approximate maximum growth rate occurring at x = - (-0.8643)/0.0366 ≈ 23, with an estimated number of cumulative attacks at this value of 40.1980/(1 + exp(0.8643 – 0.0366*24)) ≈ 20. Due to the symmetry of LGM, this corresponds to a stopping rule with 50% coverage. In this example, the maximum number of attacks estimated by the model is approximately 40 (instead of the actual 46). This is due to the late increase in attacks around time unit 2000.

Figure 2. The DR5 data set (points) with the Logistic growth curve overlaid



Running this data set through MESAT-1 software, with c = \$20, b = \$10 and a = \$50 (Figure 5) and 0.5 coverage gives a stopping rule at x=11with 20 attacks. One saves \$42,480.00 by halting at x=11. Notice that both MESAT-1 and the logistic growth curve stop testing at 20 attacks. However, the MSAT software recognizes that it is more cost efficient to stop at x = 11 as opposed to x = 23.

The cost analysis from the MESAT-1 software is shown in Table 1.

Table 1. Cost Analysis for DR5 data set in MESAT-1 software.

Item	Dollar amount
Cost of correcting all 46 errors by exhaustive - testing	\$460.00
Cost of correcting 20 pre - release errors using MESAT - 1	\$200.00

Cost of executing all 2187 test cases by exhaustive - testing	\$43,740.00
Cost of executing 11 test cases by using MESAT - 1	\$220.00
Savings for not correcting the remaining 26 by using MESAT - 1	\$260.00
Savings for not executing the remaining (2187 - 11) = 2176 test cases	\$43,520.00

Table 1 illustrates that when using a stopping rule of $x = 11$ with 20 identified errors or failures, the savings for not correcting the remaining 26 errors by using MESAT - 1 is \$260.00, plus \$43,520.00 saved for not executing the remaining 2,176 test cases equals a total savings of \$43,780.00. However, we will still incur \$1,300.00 post-release cost for correcting the 26 errors not covered during testing giving a total savings of \$42,480.00.

Notice that both methods, with a 50% coverage criterion stopped after 20 attacks were found. However, the CPM realized that stopping at x =11 would be the most optimum place to stop (the LGM stopped the testing after x= 23, which still only had 20 attacks).

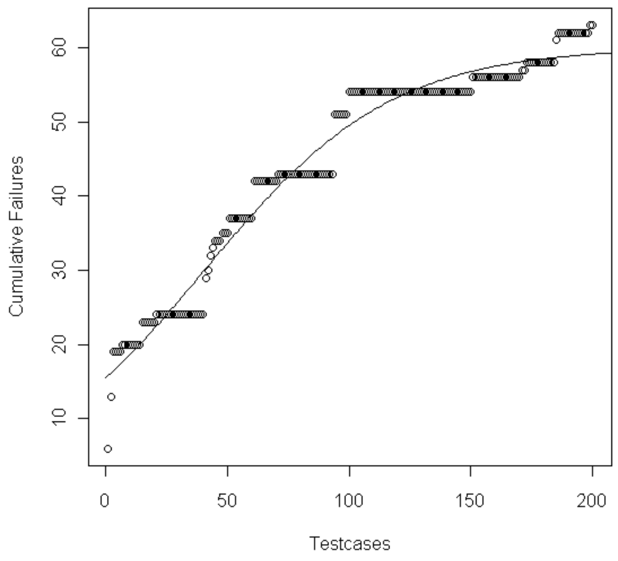
The DR4 Data Set

We use the DR4 data set from Sahinoglu (Sahinoglu, 2007) as another example of the two stopping rules. Figure 3 illustrates the data with the fitted logistic growth curve overlaid on the graph. The LGM calculated the following estimates:

$$b_0 = 60.0513,\ b_1 = -1.0545,\ b_2 = 0.0262.$$

This model estimates the maximum cumulative number of failures to be 60.0513 (actual maximum number of failures is 63). Using a stopping rule of 80% coverage indicates stopping at 0.8*60.0513 = 48.041 failures. Using the LGM, this method calculates that 80% coverage occurs at approximately 93 test cases.

Figure 3. The DR4 data set (points) with the Logistic growth curve overlaid



We ran this data set through the MESAT-1 software, which can be found online at www.areslimited.com, with c= \$100, b=\$200 a=\$1000 with d= {c/(a-b)}=0.125 and a coverage criterion of 0.80. The MESAT-1 stopping rule is at test case 95 with 51 failures and a total savings of \$900.00. Table 2 provides the cost analysis from the software.

Table 2. Cost Analysis for DR4 data set in MESAT-1 software.

Item	Dollar amount
Cost of correcting all 63 errors by exhaustive - testing	\$12600.00
Cost of correcting 51 pre - release errors using MESAT - 1	\$10200.00
Cost of executing all 200 test cases by exhaustive - testing	\$20000.00
Cost of executing 95 test cases by using MESAT - 1	\$950.00
Savings for not correcting the remaining 12 by using MESAT - 1	\$2400.00
Savings for not executing the remaining (200 - 95) = 105 test cases	\$10500.00

4. CONCLUSION

The magnitude of emerging technologies requires the use of an efficient stopping rule for software development based upon given standards (Leung, 1997) and recent security testing protocols (McGraw and Potter, 2004). Whether attacks are random or malicious in nature, an objective decision as to when to halt testing or move onto a new branch is needed. The logistic growth curve provides a quick, easy to use decision rule that can be found using most statistical software. In this method, the maximum number of attacks does not need to be known, hence decisions about when to stop testing can be made before the maximum number of cumulative errors are found. However, this method does not incorporate any costs associated with testing. Another, more comprehensive method is the compound Poisson (LSD) method which incorporates the discrete cost associated with testing, as well as the discrete nature of the data. When positive savings and $e(X) < d$ are satisfied before the coverage criterion, then the stopping rules of the two methods are similar.

MESAT-1 (which can be found at www.areslimited.com) is a cost-efficient stopping-rule algorithm used to save substantial numbers of test vectors in achieving a given degree of coverage reliability (Sahinoglu 2003, 2007). Through cost-benefit analysis, it is shown how cost-efficient this proposed stopping-rule algorithm performs, compared to those employing conventionally exhaustive “shotgun” or “testing-to-death” approaches. This cost-effective technique is valued for its industrial potential by keeping a tight rein on budgetary constraints as well as using a scientific one-step-ahead formula to optimize resource utilization. This quantitative evaluation is in sharp contrast to conventional techniques that require the usage of billions of test vectors to guarantee a certain degree of reliability.

Even though there are many extensive sources in the literature on testing software for reliability and/or security, there has been no in-depth analysis, specifically on the intricacies and complexities, and more fundamentally, on the science of when to stop most efficiently and economically. Usually, the stopping rule is either a time-to-release date, which is nothing more than a commercial benchmark or a time constraint, or it is a rough percentage of how many bugs detected out of a given total prescribed with no statistical data or trend modeling merged with cost-effective economic stopping rules. The focus in software testing for example is on determining when given the results of a testing process, whether white box (coverage), or black box (functional) testing, it is most economical to halt testing and release software under the conditions prevailing for a prescribed period of time. We are dealing with one way of conducting a quality control analysis of software testing activity with the goal of achieving a quality product most economically and accurately. Moreover, new data sets need to be analyzed from the authentic security world to see the pros and cons of each method proposed. However, these proposed methods are better than none where only attributes and adjectives are used conventionally when to stop without any actual feeling of cost efficiency.

Both LGM and CPM resulted with comparable solutions when using similar coverage criterion, first in the example (DR5) set at 20 breaches/attacks. Another data set (DR4) analysis indicated that when using an 80% coverage criterion stopping rule LGM corresponds a stop at x=93 test cases, and similarly x =95 in the CPM. This shows that that the two methods produce comparable stopping rules. However, in situations where test cases are extremely costly to correct, there may be much larger differences in the stopping rules of the two methods. In other words, the most optimal route to follow is to use first the LGM method and determine the stopping rule (such as stop at 20 breaches in the DR5 example) and then employ the CPM method to determine the savings, such as $42480.00 that correspond to the said stopping rule for a given set of cost factors, such as c=$20, b=$10 and a=$50, as in the DR5 data set. Finally, varying experimental data sets are needed for a more rigorous testing in estimating a cost efficient stopping rule using the both methods, LGM and CPM, concurrently.

5. REFERENCES

Allen, J.H., Barnum S., Ellison R.J., McGraw G., Mead N.R. (2008). Software Security Engineering: A Guide for Project Managers. SEI-CMU Software Security Series in Software Engineering. Addison Wesley, p.167.

Larralde-Corona, C. P., F. López-Isunza, et al. (1997). "Morphomotric Evaluation of the Specific Growth of *Aspergillus niger* Grown in Agar Plates at High Glucose Levels." Biotechnol. Bioeng. **56**(3): 287-284.

Leung, H.K.N. (1997). "Improving the Testing Process Based upon Standards", *Software Testing, Verification and Reliability***7**, No. 1, p. 3-18.

McGraw G. (2006). Software Security: Building Security In. Boston, MA: Addison Wesley.

Matis, J.H., Kiffe, T.R., van der Werf, W., Costamagna, A.C., Matis, T.I., Grant, W.E. (2009). "Population dynamics models based on cumulative density dependent feedback: A link to the logistic growth curve and a test for symmetry using aphid data". *Ecological Modeling***220** (15), p. 1745-1751.

Piegorsch, W. and Bailer, A.J. (2005). *Analyzing Environmental Data.*John Wiley & Sons. Ltd, The Atrium, Southern Gate, Chichester, West Sussex, England.

Randolph, P. and Sahinoglu, M. (1995). “A Stopping-Rule for a Compound Poisson Random Variable”. *Applied Stochastic Models and Data Analysis* ***11***, p. 135-143.

Sahinoglu, M. (1992). “Compound-Poisson Software Reliability Model”. *IEEE Transactions on Software Engineering* **18** (7), p. 624-630.

Sahinoglu, M. and Can, U. (1997). “Alternative Parameter Estimation Methods for the Compound Poisson Software Reliability Model with Clustered Failure Data”. *Software Testing, Verification and Reliability***7**, No. 1, p. 35-57.

Sahinoglu, M. (2003). “An Empirical Bayesian Stopping Rule in Testing and Verification of Behavioral Models”. *IEEE Transactions on Instrumentation and Measurement***52**, p. 1428-1443.

Sahinoglu, M. (2007). *Trustworthy Computing: Analytical and Quantitative Engineering Evaluation*. John Wiley & Sons, Inc., Hoboken, NJ.

Sahinoglu M., “Security Meter- A Practical Decision Tree Model to Quantify Risk,” *IEEE Security and Privacy Magazine*, **Vol. 3**, No. 3, April/May 2005, pp.18-24.

Sahinoglu M, “An Input-Output Measurable Design for the Security Meter Model to Quantify and Manage Software Security Risk”, IEEE Transactions on Instrumentation and Measurement, Vol. 57, **No. 6**, June 2008, pp. 1251-1260.

Simmons, S.J. (2008). “Estimating Daily Growth Estimates for Red Tide Algae”, Topic-Contributed Risk Session #301562: Environmental Risk: Risk Assessment and Dose-Response Models, Joint Statistics Meetings, 8/4/2008 Denver, CO.

Simmons S.J., Sahinoglu M., Matis J. (2009). “Determining an Efficient Stopping Rule for the Security testing of Cyber Attacks in Computer and Communication Networks”, Invited Risk Session #204120: Quantitative Security and Cybersytems, Joint Statistics Meetings, 8/5/2009 Washington, DC.

Verhulst, P.-F. (1845). "Recherches Mathématiques sur la Loi d'accroissement de la Population". *Nouv. mém. de l'Academie Royale des Sci. et Belles-Lettres de Bruxelles***18**, p. 1-41.

Yamada, S., Ohba, M., Osaki, S. (1984). “S-shaped Software Reliability Growth Models and Their Applications”. *IEEE Transactions on Reliability* R-**33**, p. 289-292.