Journal of Uncertain Systems
Vol.9, No.2, pp.133-138, 2015
Online at: www.jus.org.uk

How to Predict the Number of Vulnerabilities in a Software System: A Theoretical Justification for an Empirical Formula

Beverly Rivera[*], Olga Kosheleva
University of Texas at El Paso, 500 W. University, El Paso, TX 79968, USA

Received 1 June 2014; Revised 16 June 2014

Abstract

Software systems are ubiquitous in modern life. Every time a vulnerability is discovered in one of the widely use software systems (e.g., in an operating system), a large amount of effort is spent on dealing with this vulnerability. It is therefore desirable to be able to predict the number of vulnerabilities that will be discovered at different future time intervals. There exist empirical formulas which allow us to use the past performance of the software system to provide a reasonably good prediction of this future number. In this paper, we provide a theoretical justification for these empirical formulas.

Keywords: software vulnerabilities, function approximation

1 Formulation of the Problem

Need to predict the number of software vulnerabilities. Nowadays, the whole world depends on computers, and thus, depends on large software systems like operating systems. While every effort is made to make these systems secure, vulnerabilities are discovered in all the systems all the time.

Each newly discovered vulnerability requires spending a considerable amount of effort – temporarily disrupting potentially vulnerable services, checking whether any of these services have actually been compromised, performing updates, etc. It is therefore desirable to be able to predict an approximate number of new vulnerabilities that would be discovered in a given system.

Predicting the number of vulnerabilities is possible. Predicting vulnerabilities themselves is a very difficult task – these systems are complex, a vulnerability can happen in each of the numerous systems, and it is difficult to check them all. In contrast, predicting the *number* of vulnerabilities is possible.

This possibility is easy to explain on the qualitative level:

- if an operating system has a history of continuously showing new vulnerabilities, it is safe to predict that many new vulnerabilities will be discovered in the future as well;
- on the other hand, if an operating system has been unusually safe, with very few vulnerabilities discovered so far, it is safe to predict that few vulnerabilities will be discovered in the future as well.

Such a prediction – based on the past history of the software system – is possible not only on the qualitative level, it is also possible on the quantitative level. In other words, it is possible to predict how the total number $y(t)$ of vulnerabilities discovered in the system by time t changes with time.

Specifically, in [1], it was shown that this dependence is well described by a differential equation $\frac{dy}{dt} = f(y)$, where the function $f(y)$ is:

- either a constant $f(y) = a$, in which case the number of discovered vulnerabilities grows linearly with time t,

[*]Corresponding author.
Emails: barivera@miners.utep.edu (B. Rivera), olgak@utep.edu (O. Kosheleva).

- or a quadratic expression $f(y) = b \cdot y + c \cdot y^2$, in which case the dependence of discovered vulnerabilities on time t is described by a logistic formula

$$y(t) = \frac{k_1}{1 + k_2 \cdot \exp(-At)}.$$

In both cases, the function $f(y)$ is a quadratic function: $f(y) = a + b \cdot y + c \cdot y^2$.

Is this empirical formula a crude approximation, or is it theoretically justified? How much can we trust the above empirical formula?

It could be that this formula is just a crude approximation to reality, with no theoretical justification, in which case we should not put much trust in this formula – after all, any function $f(y)$ can be expanded in Taylor series and thus, approximated by a quadratic expression.

It could also be that this formula is not only empirically valid, it also has a deeper theoretical meaning. In this case, we have reasons to believe that this formula is applicable to other software systems as well – and thus, we should place more trust in predictions based on this formula.

What we do in this paper. In this paper, we provide a possible theoretical explanation for the above empirical formula.

This theoretical explanation makes us more confident that this formula can be applied not only to describe how many vulnerabilities appeared in the past, this formula can also be confidently for predicting how many vulnerabilities will appear in the future.

2 Analysis of the Problem

How to count the number of vulnerabilities? To derive and test the formula for the dependence of the number of discovered vulnerabilities y on time t, the paper [1] tries to match the recorded numbers of vulnerabilities $y(t_i)$ at different moments of time t_i.

A natural question is: where do these numbers come from? How do we count the number of vulnerabilities in the first place?

At first glance, the answer sounds simple: just count. However, if one looks deeper, the answer becomes less clear. Let us explain what we mean.

When do we start counting? Shall we start counting vulnerabilities when the software system was released for beta-testing? when the system was commercially released?

Shall we also count vulnerabilities discovered when the system was still being designed? This may be helpful: if we know that a large number of vulnerabilities have been discovered at the design stage, and only a few after that, then should be more confident in the system.

Depending on when we start counting, we will get different numerical values of the quantity y: if we start counting earlier, then, instead of the original value y, we have a modified value $y' = y + y_0$, where y_0 is the number of vulnerabilities discovered when we started counting originally.

Do we count related vulnerabilities? Subsystems of software systems are usually closely inter-related. As a result, a discovery vulnerability in one subsystem often leads to a discovery of related vulnerabilities. Do we count all these related vulnerabilities in our count – or do we count the whole group of related vulnerabilities as a single vulnerability? Or shall we only group together very similar and closely related vulnerabilities.

Depending on how we count:

- we may get y vulnerabilities – if we count only *groups* of related vulnerabilities;
- or we may get $y' = c \cdot y$ vulnerabilities – if we count all *individual* vulnerabilities, related or nor (her c is the average number of related vulnerabilities in a single group).

Let us come up with a family of approximating functions. We want to approximate functions $f(y)$ corresponding to different software systems. A natural way to do it is to select a small number of *basic* functions $f_1(y)$, ..., $f_k(y)$, and to approximate the dependence $f(y)$ of $\dfrac{dy}{dt}$ on y by linear combinations of the basic functions: $f(y) \approx C_1 \cdot f_1(y) + \cdots + C_k \cdot f_k(y)$.

Reasonable requirements on the family of approximating functions. Since the value y is determined modulo transformations

- $y \to y' = y + y_0$ (*shift*) and
- $y \to y' = c \cdot y$ (*scaling*),

it is reasonable to require that for the selected basic functions $f_i(y)$, the resulting family of approximating functions, i.e., the family of all the functions of the type $C_1 \cdot f_1(y) + \cdots + C_k \cdot f_k(y)$, should not change under these transformations.

An additional requirement: that all approximating functions are differentiable. In practice, the values $y(t)$ are integers, so the dependence $y(t)$ is discontinuous: it jumps to the next value $y+1$ every time we discover a new vulnerability. However, following [1], we consider a smooth (differentiable) approximation to this dependence, in which the function $y(t)$ is differentiable.

In this approximation, it is reasonable to require that the approximating basic functions be differentiable as well.

Now, we are ready to formulate the problem in precise terms.

3 Definitions and the Main Result

Definition 1. *By an* approximating family*, we mean the family of all the functions of the type*

$$C_1 \cdot f_1(y) + \cdots + C_k \cdot f_k(y),$$

where $f_1(y)$, ..., $f_k(y)$ are fixed linearly independent differentiable functions, and $C_1, \ldots, C_k$ are arbitrary real numbers.

Example. The family of all quadratic functions $a + b \cdot y + c \cdot y^2$ is an approximating family, with $f_1(y) = 1$, $f_2(y) = y$, and $f_3(y) = y^2$.

Definition 2. *We say that an approximating family $\mathcal{F}$ is* shift-invariant *if for every function $f(y) \in \mathcal{F}$ and for every real number y_0, the function $f(y + y_0)$ also belongs to the family $\mathcal{F}$.*

Definition 3. *We say that an approximating family $\mathcal{F}$ is* scale-invariant *if for every function $f(y) \in \mathcal{F}$ and for every real number c, the function $f(c \cdot y)$ also belongs to the family $\mathcal{F}$.*

Example. One can easily check that the family of all quadratic functions is shift- and scale-invariant. It turns out that all shift- and scale-invariant families of functions are families of polynomials:

Proposition. *Every shift- and scale-invariant family of functions is the family of all polynomials of order $\le k - 1$.*

Discussion. Thus, for 3-parametric families of approximating functions, with $k = 3$, we get a theoretical explanation for why quadratic functions $f(y)$ are a good approximation – because such functions are the only family which satisfies the reasonable conditions of shift- and scale-invariance.

This result also shows what we need to do if we want to come up with even more accurate approximations: consider families of cubic, quartic, etc, functions $f(y)$.

4 Proof of the Proposition

0°. The main ideas of this proof are similar to the ideas of proof of similar results described in [5, 6].

1°. Each of the basic functions $f_i(y)$ belongs to the approximating family: it corresponds to $C_i = 1$ and $C_j = 0$ for all $j \ne i$. Thus, due to shift-invariance, for each y_0, the function $f_i(y + y_0)$ also belongs to the approximating family. By definition of the family, this means that for every y_0, there exist values $C_{ij}(y_0)$ for which

$$f_i(y + y_0) = C_{i1}(y_0) \cdot f_1(y) + \cdots + C_{ik}(y_0) \cdot f_k(y). \tag{1}$$

2°. We know that the functions $f_i(y)$ are differentiable. Let us prove that the functions $C_{ij}(y_0)$ are differentiable as well.

Indeed, if we take the equation (1) for k different values $y_1, \ldots, y_k$, then we get a system of k linear equations for k unknowns $C_{i1}(y_0)$, ..., $C_{ik}(y_0)$:

$$\begin{aligned} f_i(y_1 + y_0) &= C_{i1}(y_0) \cdot f_1(y_1) + \cdots + C_{ik}(y_0) \cdot f_k(y_1); \\ f_i(y_2 + y_0) &= C_{i1}(y_0) \cdot f_1(y_2) + \cdots + C_{ik}(y_0) \cdot f_k(y_2); \\ &\vdots \\ f_i(y_k + y_0) &= C_{i1}(y_0) \cdot f_1(y_k) + \cdots + C_{ik}(y_0) \cdot f_k(y_k). \end{aligned} \tag{2}$$

Due to the well-known Cramer's rule (see, e.g. [3]), each component of a solution to the system of linear equations can be described as a ratio of two determinants, i.e., as a differentiable function of coefficients and free terms. The coefficients $f_j(y_\ell)$ do not depend on y_0 at all; the free terms $f_i(y_j + y_0)$ are differentiable functions of y_0 (since $f_i(y)$ is a differentiable function). Thus, the functions $C_{ij}(y_0)$ are indeed differentiable.

3°. Since all the functions $f_i(y)$ and $C_{ij}(y_0)$ are differentiable, we can differentiate both sides of equation (1) with respect to y_0, and then take $y_0 = 0$. As a result, we get the following equation:

$$f_i'(y) = c_{i1} \cdot f_1(y) + \cdots + c_{ik} \cdot f_k(y), \tag{3}$$

where $f_i'(y)$ denotes the derivative and $c_{ij} \stackrel{\text{def}}{=} C_{ij}'(0)$.

By combining the equations (3) corresponding to $i = 1, \ldots, k$, we conclude that the functions $f_i(y)$ satisfy a system of ordinary differential equations with constant coefficients c_{ij}.

It is known (see, e.g., [2, 4, 7]) that each solution to such a system is a linear combination of functions of the type $y^n \cdot \exp(a \cdot y)$, where $n = 0, 1, 2, \ldots$ is a natural number and a can be any complex number.

Thus, each function $f_i(y)$ is a linear combination of such functions.

4°. Let us now use scale-invariance. Due to scale-invariance, for each real number c, the function $f_i(c \cdot y)$ also belongs to the approximating family. By definition of the family, this means that for every c, there exist values $A_{ij}(c)$ for which

$$f_i(c \cdot y) = A_{i1}(c) \cdot f_1(y) + \cdots + A_{ik}(c) \cdot f_k(y). \tag{4}$$

5°. We know that the functions $f_i(y)$ are differentiable. Let us prove that the functions $A_{ij}(c)$ are differentiable as well.

Indeed, if we take the equation (4) for k different values $y_1, \ldots, y_k$, then we get a system of k linear equations for k unknowns $A_{i1}(c)$, ..., $A_{ik}(c)$:

$$\begin{aligned} f_i(c \cdot y_1) &= A_{i1}(c) \cdot f_1(y_1) + \cdots + A_{ik}(c) \cdot f_k(y_1); \\ f_i(c \cdot y_2) &= A_{i1}(c) \cdot f_1(y_2) + \cdots + A_{ik}(c) \cdot f_k(y_2); \\ &\vdots \\ f_i(c \cdot y_k) &= A_{i1}(c) \cdot f_1(y_k) + \cdots + A_{ik}(c) \cdot f_k(y_k). \end{aligned} \tag{5}$$

Due to the Cramer's rule, each component of a solution to this system of linear equations is a differentiable function of coefficients $f_j(y_\ell)$ and free terms $f_i(c \cdot y_j)$. The coefficients $f_j(y_\ell)$ do not depend on c at all; the free terms $f_i(c \cdot y_j)$ are differentiable functions of c (since $f_i(y)$ is a differentiable function). Thus, the functions $A_{ij}(c)$ are indeed differentiable.

6°. Since all the functions $f_i(y)$ and $A_{ij}(c)$ are differentiable, we can differentiate both sides of equation (4) with respect to c, and then take $c = 1$. As a result, we get the following equation:

$$y \cdot f_i'(y) = a_{i1} \cdot f_1(y) + \cdots + a_{ik} \cdot f_k(y), \tag{6}$$

where $a_{ij} \stackrel{\text{def}}{=} A'_{ij}(1)$.

For each function $f(y)$, the product $y \cdot \dfrac{df}{dy}$ can be described as $\dfrac{df}{dY}$, where we denoted $dY \stackrel{\text{def}}{=} dy/y$. By integrating this relation, we get $Y = \ln(y)$.

Thus, if we introduce a new variable $Y = \ln(y)$ (for which $y = \exp(Y)$) and new functions $F_i(Y) \stackrel{\text{def}}{=} f_i(\exp(Y))$, then for these new functions, the equation (6) takes the form

$$F'_i(Y) = a_{i1} \cdot F_1(y) + \cdots + a_{ik} \cdot F_k(y). \tag{7}$$

Thus, the functions $F_i(Y)$ are solutions to a system of linear ordinary differential equations with constant coefficients. Therefore, each function $F_i(Y)$ is a linear combination of terms of the type $Y^m \cdot \exp(b \cdot Y)$, where m is a natural number and b is, in general, a complex number.

Once we know the functions $F_i(Y)$, we can determine the original functions $f_i(y)$ as $f_i(y) = F_i(\ln(Y))$. By substituting $Y = \ln(y)$ into the formula $Y^m \cdot \exp(b \cdot Y)$, we conclude that each original functions $f_i(y)$ is a linear combination of terms of the type $(\ln(y))^m \cdot \exp(b \cdot \ln(y))$. Here,

$$\exp(b \cdot \ln(y)) = e^{b \cdot \ln(y)} = \left(e^{\ln(y)}\right)^b = y^b.$$

Thus, we conclude that each function $f_i(y)$ is a linear combination of functions $(\ln(y))^m \cdot y^b$.

7°. From the shift-invariance, we know that every function $f_i(y)$ is a linear combination of terms of the type $y^n \cdot \exp(a \cdot y)$.

Both these functions and the functions described in Part 6 of this proof are analytical functions, so they can be extended to the complex values of y. If at least one of the terms $y^n \cdot \exp(a \cdot y)$ has $a \neq 0$, then the corresponding function grows exponentially in the complex domain and thus, we cannot represent this term as a linear combination of slower-growing terms of the type $(\ln(y))^m \cdot y^b$.

Therefore, $a = 0$ and thus, each function $f_i(y)$ is a linear combination of terms x^n for natural n – i.e., a polynomial.

8°. Let us prove that if the family $\mathcal{F}$ contains *a* polynomial of order m, then it contains *all* polynomials of order $\leq m$.

Indeed, let $f(y) \in \mathcal{F}$ be a polynomial of order m, i.e., $f(y) = a_m \cdot y^m + a_{m-1} \cdot y^{m-1} + \cdots$, where $a_m \neq 0$. Since the family $\mathcal{F}$ is a linear space, it also contains a polynomial $g_m(y) = a_m^{-1} \cdot f(y)$ which has the form $g_m(y) = y^m + b_{m,m-1} \cdot y^{m-1} + \cdots$, for some values $b_{m,i}$.

Since the family $\mathcal{F}$ is shift-invariant, it also contains a shifted polynomial $g_m(y+1)$. Since $\mathcal{F}$ is a linear space, it also contains the difference

$$g_m(y+1) - g_m(y) = ((y+1)^m - y^m) + b_{m,m-1} \cdot ((y+1)^{m-1} - y^{m-1}) + \cdots$$

Here, for every degree d, we have

$$(y+1)^d - y^d = d \cdot y^{d-1} + \frac{d \cdot (d-1)}{2} \cdot y^{d-2} + \cdots,$$

hence

$$g_m(y+1) - g_m(y) = m \cdot y^{m-1} + c \cdot y^{m-2} + \cdots$$

for some c.

Since $\mathcal{F}$ is a linear space, it also contains the polynomial

$$g_{m-1}(y) \stackrel{\text{def}}{=} \frac{1}{m} \cdot (g_m(y+1) - g_m(y))$$

which thus has the form

$$g_{m-1}(y) = y^{m-1} + b_{m-1,m-2} \cdot y^{m-2} + \cdots$$

By applying a similar construction to this polynomial $g_{m-1}(y)$, we conclude that the family $\mathcal{F}$ contains a polynomial $g_{m-2}(y)$ of the type

$$g_{m-2}(y) = y^{m-2} + b_{m-2,m-3} \cdot y^{m-3} + \cdots$$

and, in general, for each $k \leq m$, a polynomial

$$g_k(y) = y^k + b_{k,k-1} \cdot y^{k-1} + \cdots .$$

Let us now prove, by induction over k, that for each $k \leq m$, the family $\mathcal{F}$ contains all polynomials of order $\leq k$.

Indeed, for $k = 0$, the family $\mathcal{F}$ contains the constant function $g_0(y) = 1$, and since $\mathcal{F}$ is a linear space, it contains all constant functions $f(y) = \text{const}$.

Let us now assume that for some $k \leq m$, we have already proven that the family $\mathcal{F}$ contains all the polynomials of order $\leq k-1$, let us now prove that this family contains all the polynomials of order k as well. Indeed, the family $\mathcal{F}$ contains the function $g_k(y) = y^k + (b_{k,k-1} \cdot y^{k-1} + \cdots)$. Since the terms $b_{k,k-1} \cdot y^{k-1} + \cdots$ form a polynomial of order $\leq k-1$, they are also contained in the family $\mathcal{F}$. Since the family $\mathcal{F}$ is a linear space, this family contains the difference

$$g_k(y) - (b_{k,k-1} \cdot y^{k-1} + \cdots) = y^k.$$

Again, since the family $\mathcal{F}$ is a linear space, and it contains y^k and all the monomials y^p with $p \leq k-1$, it thus contains all linear combinations of these monomials – i.e., all polynomials of order $\leq k$. The statement is proven.

9°. Because of Parts 7 and 8 of this prove, the family $\mathcal{F}$ consists of all polynomials of order m, for some m.

The family $\mathcal{F}$ is a set of all linear combinations of k linearly independent functions, so its dimension is k.

The family of all polynomials of order m is the set of all linear combinations of $m+1$ functions $y^m, y^{m-1}, \ldots, y^0$. Thus, its dimension is $m+1$.

From $k = m+1$, we conclude that $m = k-1$. The proposition is thus proven.

Acknowledgments

This work was supported by the University of Texas at El Paso Regional Cyber and Energy Security Center (RCES), a center supported by the City of El Paso's Planning and Economic Development division.

References

[1] Alhazmy, O.H., Malaiya, Y.K., and I. Ray, Measuring, analyzing, and predicting security vulnerabilities in software systems, *Computers and Security*, vol.26, pp.219–228, 2007.

[2] Birkhoff, G., and G.-C. Rota, *Ordinary Differential Equations*, Wiley and Sons, New York, 1978.

[3] Dianant, S.A., and E.S. Saber, *Advanced Linear Algebra for Engineers with Matlab*, CRC Press, Boca Raton, Florida, 2009.

[4] Gershenfeld, N., *The Nature of Mathematical Modeling*, Cambridge University Press, Cambridge, UK, 1999.

[5] Kosheleva, O., and M. Ceberio, Why polynomial formulas in soft computing, decision making, etc.?, *Proceedings of the IEEE World Congress on Computational Intelligence*, pp.3310–3314, 2010.

[6] Nguyen, H.T., and V. Kreinovich, *Applications of Continuous Mathematics to Computer Science*, Kluwer, Dordrecht, 1997.

[7] Robinson, J.C., *An Introduction to Ordinary Differential Equations*, Cambridge University Press, Cambridge, UK, 2004.