

# Defining and Assessing Quantitative Security Risk Measures Using Vulnerability Lifecycle and CVSS Metrics

HyunChul Joh<sup>1</sup>, and Yashwant K. Malaiya<sup>1</sup>

<sup>1</sup>Computer Science Department, Colorado State University, Fort Collins, CO 80523, USA

**Abstract** - Known vulnerabilities which have been discovered but not patched represents a security risk which can lead to considerable financial damage or loss of reputation. They include vulnerabilities that have either no patches available or for which patches are applied after some delay. Exploitation is even possible before public disclosure of a vulnerability. This paper formally defines risk measures and examines possible approaches for assessing risk using actual data. We explore the use of CVSS vulnerability metrics which are publically available and are being used for ranking vulnerabilities. Then, a general stochastic risk evaluation approach is proposed which considers the vulnerability lifecycle starting with discovery. A conditional risk measure and assessment approach is also presented when only known vulnerabilities are considered. The proposed approach bridges formal risk theory with industrial approaches currently being used, allowing IT risk assessment in an organization, and a comparison of potential alternatives for optimizing remediation. These actual data driven methods will assist managers with software selection and patch application decisions in quantitative manner.

**Keywords** - Security vulnerabilities; Software Risk Evaluation; CVSS; Vulnerability lifecycle

## 1 Introduction

To ensure that the overall security risk stays within acceptable limits, managers need to measure risks in their organization. As Lord Calvin stated "If you cannot measure it, you cannot improve it," quantitative methods are needed to ensure that the decisions are not based on subjective perceptions only.

Quantitative measures have been commonly used to measure some attributes of computing such as performance and reliability. While quantitative risk evaluation is common in some fields such as finance [1], attempts to quantitatively assess security are relatively new. There has been criticism of the quantitative attempts of risk evaluation [2] due to the lack of data for validating the methods. Related data has now begun to become available. Security vulnerabilities that have been discovered but remain unpatched for a while represent considerable risk for an organization. Today online banking, stock market trading, transportation, even military and governmental exchanges depend on the Internet based computing and communications. Thus the risk to the society due to the exploitation of vulnerabilities is massive. Yet peo-

ple are willing to take the risk since the Internet has made the markets and the transactions much more efficient [3]. In spite of the recent advances in secure coding, it is unlikely that completely secure systems will become possible anytime soon [4]. Thus, it is necessary to assess and contain risk using precautionary measures that are commensurate.

While sometimes risk is informally stated as the possibility of a harm to occur [5], formally, risk is defined to be a weighted measure depending on the consequence. For a potential adverse event, the risk is stated as [6]:

$$\text{Risk} = \text{Likelihood of an adverse event} \times \text{Impact of the adverse event} \quad (1)$$

This presumes a specific time period for the evaluated likelihood. For example, a year is the time period for which *annual loss expectancy* is evaluated. Equation (1) evaluates risk due to a single specific cause. When statistically independent multiple causes are considered, the individual risks need to be added to obtain the overall risk. A *risk matrix* is often constructed that divides both likelihood and impact values into discrete ranges that can be used to classify applicable causes [7] by the degree of risk they represent.

In the equation above, the likelihood of an adverse event is sometimes represented as the product of two probabilities: probability that an exploitable weakness is present, and the probability that such a weakness is exploited [7]. The first is an attribute of the targeted system itself whereas the second probability depends on external factors, such as the motivation of potential attackers. In some cases, the *Impact of the adverse event* can be split into two factors, the technical impact and the business impact [8]. Risk is often measured conditionally, by assuming that some of the factors are equal to unity and thus can be dropped from consideration. For example, sometimes the external factors or the business impact is not considered. If we would replace the impact factor in Equation (1) by unity, the conditional risk simply becomes equal to the probability of the adverse event, as considered in the traditional reliability theory. The conditional risk measures are popular because it can alleviate the formidable data collections and analysis requirements. As discussed in section 4 below, different conditional risk measures have been used by different researchers.

A *vulnerability* is a software defect or weakness in the security system which might be exploited by a malicious user causing loss or harm [5]. A stochastic model [9] of the vulnerability lifecycle could be used for calculating the *Likelihood of an adverse event* in Equation (1) whereas impact related metrics from the Common Vulnerability Scoring Sys-

tem (CVSS) [10] can be utilized for estimating *Impact of the adverse event*. While a preliminary examination of some of the vulnerability lifecycle transitions has recently been done by researchers [11][12], risk evaluation based on them have been received little attention. The proposed quantitative approach for evaluating the risk associated with software systems will allow comparison of alternative software systems and optimization of risk mitigation strategies.

The paper is organized as follows. Section 2 introduces the risk matrix. Section 3 discusses the CVSS metrics that are now being widely used. The interpretation of the CVSS metrics in terms of the formal risk theory is discussed in section 4. Section 5 introduces the software vulnerability lifecycle and the next section gives a stochastic method for risk level measurement. Section 7 presents a conditional risk assessing method utilizing CVSS base scores which is illustrated by simulated data. Finally, conclusions are presented and the future research needs are identified.

## 2 Risk matrix: scales & Discretization

In general, a system has multiple weaknesses. The risk of exploitation in each weakness  $i$  is given by Equation (1). Assuming that the potential exploitation of a weakness is statistically independent of others, the system risk is given by the summation of individual risk values:

$$\text{System Risk} = \sum_i L_i \times I_i \quad (2)$$

where  $L_i$  is the likelihood of exploitation of weakness  $i$  and  $I_i$  is the corresponding impact. A risk matrix provides a visual distribution of potential risks [13][14]. In many risk evaluation situations, a risk matrix is used, where both impact and likelihood are divided into a set of discrete intervals, and each risk is assigned to likelihood level and an impact level. Impact can be used for the x-axis and likelihood can be represented using then y-axis, allowing a visual representation of the risk distribution. For example, the ENISA Cloud Computing report [15] defines five impact levels from *Very Low* to *Very High*, and five likelihood levels from *Very Unlikely* to *Frequent*. Each level is associated with a *rating*. A risk matrix can bridge quantitative and qualitative analyses. Tables have been compiled that allow on to assign a likelihood and an impact level to a risk, often using qualitative judgment or a rough quantitative estimation.

The scales used for likelihood and impact can be linear, or more often non-linear. In Homeland Security's RAMCAP (Risk Analysis and Management for Critical Asset Protection) approach, a logarithmic scale is used for both. Thus, 0-25 fatalities is assigned a rating "0", while 25-50 is assigned a rating of "1", etc. For the likelihood scale, probabilities between 0.5-1.0 is assigned the highest rating of "5", between 0.25-0.5 is assigned rating "4", etc.

Using a logarithmic scale for both has a distinct advantage. Sometimes the overall rating for a specific risk is found by simply adding its likelihood and impact ratings. Thus, it would be easily explainable if the *rating* is proportional to the logarithm of the absolute value. Consequently, Equation (1) can be re-written as:

$$\begin{aligned} \log(\text{risk}_i) &= \log(\text{likelihood}_i) + \log(\text{impact}_i) \\ \text{rating\_Risk}_i &= \text{rating\_likelihood}_i + \text{rating\_impact}_i \end{aligned} \quad (3)$$

When a normalized value of the likelihood, impact or the risk is used, it will result in a positive or negative constant added to the right hand side. In some cases, higher resolution is desired in the very high as well as very low regions; in such cases a suitable non-linear scale such as using the *logit* or *log-odds* function [16] can be used.

The main use of risk matrices is to rank the risks so that higher risks can be identified and mitigated. For determining ranking, the rating can be used instead of the raw value. Cox [7] has pointed out that the discretization in a risk matrix can potentially result in incorrect ranking, but risk matrices are often used for convenient visualization. It should be noted that the risk ratings are not additive.

We will next examine the CVSS metrics that has emerged recently for software security vulnerabilities, and inspect the relationship (likelihood, impact) in risk and (exploitability, impact) in CVSS vulnerability metric system.

## 3 CVSS metrics and Related Works

Common Vulnerability Scoring System (CVSS) [10] has now become almost an industrial standard for assessing the security vulnerabilities although some alternatives are sometimes used. It attempts to evaluate the degree of risks posed by vulnerabilities, so mitigation efforts can be prioritized. The measures termed *scores* are computed using assessments (called *metrics*) of vulnerability attributes based on the opinions of experts in the field. Initiated in 2004, now it is in its second version released in 2007.

The CVSS scores for known vulnerabilities are readily available on the majority of public vulnerability databases on the Web. The CVSS score system provides vendor independent framework for communicating the characteristics and impacts of the known vulnerabilities [10]. A few researchers have started to use some of the CVSS metrics for their security risk models.

CVSS defines a number of metrics that can be used to characterize a vulnerability. For each metric, a few qualitative levels are defined and a numerical value is associated with each level. CVSS is composed of three major metric groups: Base, Temporal and Environmental. The Base metric represents the intrinsic characteristics of a vulnerability, and is the only mandatory metric. The optional Environmental and Temporal metrics are used to augment the Base metrics, and depend on the target system and changing circumstances. The Base metrics include two sub-scores termed *exploitability* and *impact*. The Base score formula [10], as shown in Equation (4), is chosen and adjusted such that a score is a decimal number in the range [0.0, 10.0]. The value for  $f(\text{Impact})$  is zero when *Impact* is zero otherwise it has the value of 1.176.

$$\begin{aligned} \text{Base score} &= \text{Round to 1 decimal} \{ \\ &[(0.6 \times \text{Impact}) + (0.4 \times \text{Exploitability}) - 1.5] \times f(\text{Impact}) \} \end{aligned} \quad (4)$$

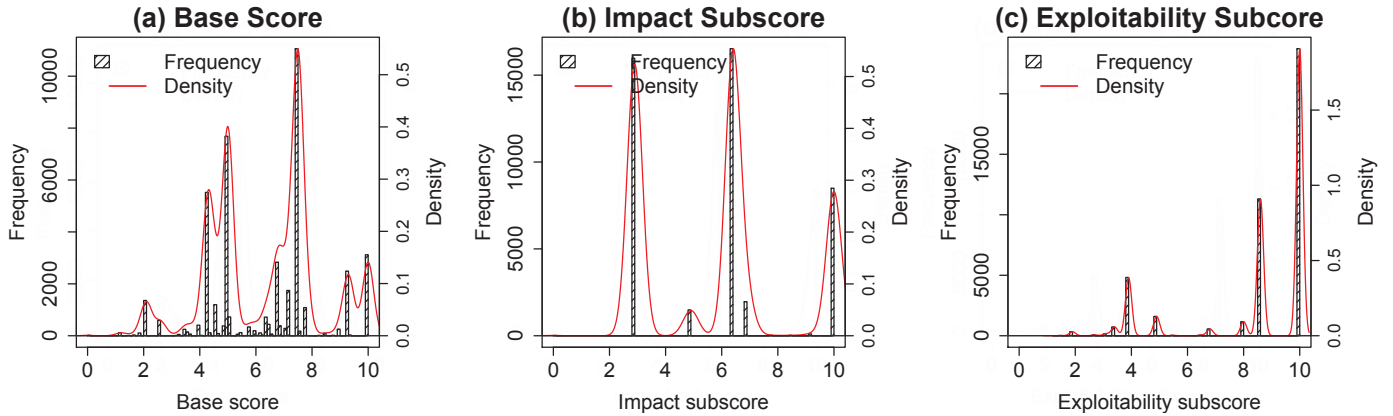


Figure 1. Distributions for CVSS base metric scores (100 bins); NVD [17] on JAN 2011 (44615 vuln.)

The formula for Base score in Equation (4) has not been formally derived but has emerged as a result of discussions in a committee of experts. It is primarily intended for ranking of vulnerabilities based on the risk posed by them. It is notable that the Exploitability and Impact sub-scores are added rather than multiplied. One possible interpretation can be that the two sub-scores effectively use a logarithmic scale, as given in Equation (3). Then possible interpretation is that since the Impact and Exploitability sub-scores have a fairly discrete distribution as shown in Fig. 1 (b) and (c), addition yields the distribution, Fig 1 (a), which would not be greatly different if we had used a multiplication. We have indeed verified that using  $Impact \times Exploitability$  yields a distribution extremely similar to that in Fig. 1 (a). We have also found that multiplication generates about twice as many combinations with wider distribution, and it is intuitive since it is based on the definition of risk given in Equation (1).

The Impact sub-score measures how a vulnerability will impact an IT asset in terms of the degree of losses in confidentiality, integrity, and availability which constitute three of the metrics. Below, in our proposed method, we also use these metrics. The Exploitability sub-score uses metrics that attempt to measure how easy it is to exploit the vulnerability. The Temporal metrics measure impact of developments such as release of patches or code for exploitation. The Environmental metrics allow assessment of impact by taking into account the potential loss based on the expectations for the target system. Temporal and Environmental metrics can add additional information to the two sub-scores used for the Base metric for estimating the overall software risk.

A few researchers have started to use the CVSS scores in their proposed methods. Mkpong-Ruffin et al. [17] use CVSS scores to calculate the loss expectancy. The average CVSS scores are calculated with the average growth rate for each month for the selected functional groups of vulnerabilities. Then, using the growth rate with the average CVSS score, the predicted impact value is calculated for each functional group. Houmb et al. [19] have discussed a model for the quantitative estimation of the security risk level of a system by combining the frequency and impact of a potential unwanted event and is modeled as a Markov process. They estimate frequency and impact of vulnerabilities using reorganized original CVSS metrics. And, finally, the two estimated measures are combined to calculate risk levels.

## 4 Defining conditional risk measures

Researchers have often investigated measures of risk that seem to be defined very differently. Here we show that they are conditional measures of risk and can be potentially combined into a single measure of total risk. The likelihood of the exploitation of a vulnerability depends not only on the nature of the vulnerability but also how easy it is to access the vulnerability, the motivation and the capabilities of a potential intruder.

The likelihood  $L_i$ , in Equation (2), can be expressed in more detail by considering factors such as probability of presence of a vulnerability  $v_i$  and how much exploitation is expected as shown below:

$$\begin{aligned} L_i &= \Pr\{v_i\} \times \Pr\{Exploitation \mid v_i\} \\ &= \Pr\{v_i\} \times \Pr\{V_i \text{ is exploitable} \mid v_i\} \times \\ &\quad \Pr\{v_i \text{ is accessible} \mid v_i \text{ exploitable}\} \times \\ &\quad \Pr\{v_i \text{ externally exploited} \mid v_i \text{ accessible \& exploitable}\} \\ &= L_{Ai} \times L_{Bi} \times L_{Ci} \times L_{Di} \end{aligned}$$

where  $L_{Bi}$  represents the inherent exploitability of the vulnerability,  $L_{Ci}$  is the probability of accessing the vulnerability, and  $L_{Di}$  represents the external factors. The impact factor,  $I_i$ , from Equation (1) can be given as:

$$\begin{aligned} I_i &= \sum_j \Pr\{Security \text{ attribute } j \text{ compromised for } v_i\} \times \\ &\quad \{Expected \text{ cost of } j \text{ compromised due to } v_i\} \\ &= \sum_j \Pr(attribute_j, v_i) \times C_{ji} \\ &= \sum_j I_{ji} \times C_{ji} \\ &= I_{iA} \times C_{ji} \end{aligned}$$

where the security attribute  $j=1,2,3$  represents confidentiality, integrity and availability.  $I_{iA}$  is the CVSS Base Impact sub-score whereas  $C_{ji}$  is the CVSS Environmental ConfReq, IntegReq or AvailReq metric.

The two detailed expressions for likelihood and impact above in terms of constituent factors, allow defining conditional risk measures. Often risk measures used by different authors differ because they are effectively conditional risks which consider only some of the risk components. The components ignored are then effectively equal to one.

As mentioned above, for a weakness  $i$ , risk is defined as  $L_i \times I_i$ . The conditional risk measures  $\{R_1, R_2, R_3, R_4\}$  can



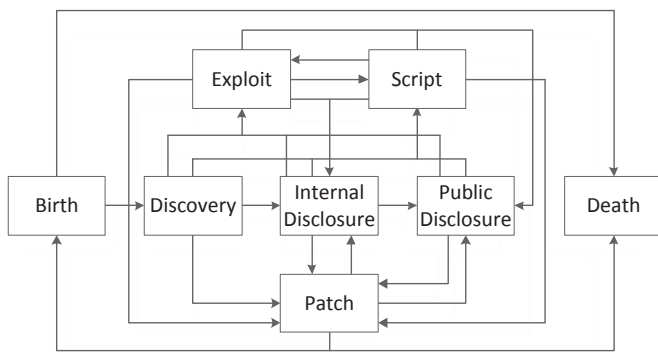


Figure 2. Possible vulnerability lifecycle journey

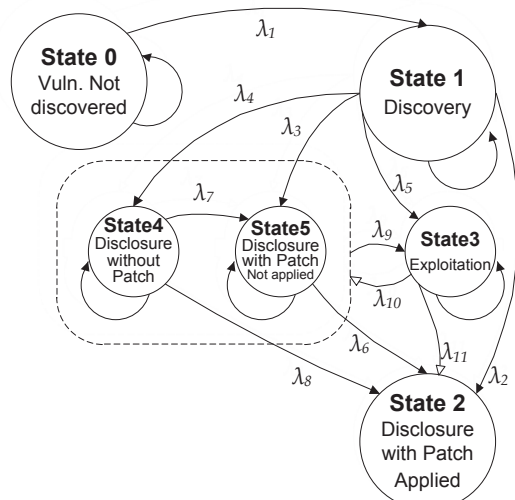


Figure 3. Stochastic model for a single vulnerability

be defined by setting some of the factors in the above equations to unity:

- $R_1$ : by setting  $\{L_{Ci}, L_{Di}, C_{ji}\}$  as unity. The CVSS Base score is a  $R_1$  type risk measure.
- $R_2$ : by setting  $\{L_{Di}, C_{ji}\}$  as unity. The CVSS temporal score is a  $R_2$  type risk measure.
- $R_3$ : by setting  $L_{Di}$  as unity. The CVSS temporal score is a  $R_3$  type risk measure.
- $R_4$ : is the total risk considering all the factors.

In the next two sections, we examine a risk measure that is more general compared with other perspectives in the sense that we consider the discovery of hitherto unknown vulnerabilities. This would permit us to consider 0-day attacks within our risk framework. In the following section a simplified perspective is presented which considers only the known vulnerabilities.

## 5 Software vulnerability Lifecycle

A vulnerability is created as a result of a coding or specification mistake. Fig. 2 shows possible vulnerability lifecycle journeys. After the birth, the first event is discovery. A discovery may be followed by any of these: internal disclosure, patch, exploit or public disclosure. The discovery rate can be described by vulnerability discovery models (VDM) [20]. It

has been shown that VDMs are also applicable when the vulnerabilities are partitioned according to severity levels [21]. It is expected that some of the CVSS base and temporal metrics impact the probability of a vulnerability exploitation [10], although no empirical studies have yet been conducted.

When a white hat researcher discovers a vulnerability, the next transition is likely to be the internal disclosure leading to patch development. After being notified of a discovery by a white hat researcher, software vendors are given a few days, typically 30 or 45 days, for developing patches [22]. On the other hand, if the disclosure event occurred within a black hat community, the next possible transition may be an exploitation or a script to automate exploitation. Informally, the term *zero day vulnerability* generally refers to an unpublished vulnerability that is exploited in the wild [23]. Studies show that the time gap between the public disclosure and the exploit is getting smaller [24]. Norwegian Honeynet Project [25] found that from the public disclosure to the exploit event takes a median of 5 days (the distribution is highly asymmetric).

When a script is available, it enhances the probability of exploitations. It could be disclosed to a small group of people or to the public. Alternatively, the vulnerability could be patched. Usually, public disclosure is the next transition right after the patch availability. When the patch is flawless, applying it causes the death of the vulnerability although sometimes a patch can inject a new fault [26].

Frei has [11] found that 78% of the examined exploitations occur within a day, and 94% by 30 days from the public disclosure day. In addition, he has analyzed the distribution of discovery, exploit, and patch time with respect to the public disclosure date, using a very large dataset.

## 6 Evaluating lifecycle risk

We first consider evaluation of the risk due to a single vulnerability using stochastic modeling [9]. Fig. 3 presents a simplified model of the lifecycle of a single vulnerability, described by six distinct states. Initially, the vulnerability starts in State 0 where it has not been found yet. When the discovery leading to State 1 is made by white hats, there is no immediate risk, whereas if it is found by a black hat, there is a chance it could be soon exploited. State 2 represents the situation when the vulnerability is disclosed along with the patch release and the patch is applied right away. Hence, State 2 is a safe state and is an absorbing state. In State 5, the vulnerability is disclosed with a patch but the patch has not been applied, whereas State 4 represents the situation when the vulnerability is disclosed without a patch. Both State 4 and State 5 expose the system to a potential exploitation which leads to State 3. The two white head arrows ( $\lambda_{10}$  and  $\lambda_{11}$ ) are backward transitions representing a recovery which might be considered when multiple exploitations within the period of interest need to be considered. In the discussion below we assume that State 3 is an absorbing state.

In the figure, for a single vulnerability, the cumulative risk in a specific system at time  $t$  can be expressed as proba-

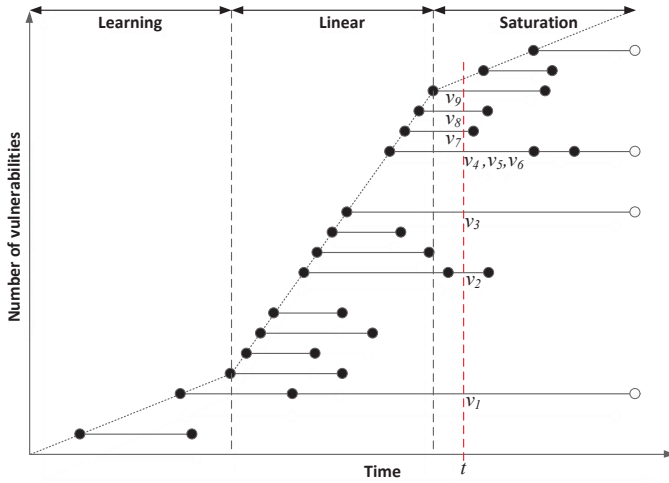


Figure 4. Example of the vulnerability discovery and patch in a system with simplified three phase vulnerability lifecycle

bility of the vulnerability being in State 3 at time  $t$  multiplied by the consequence of the vulnerability exploitation.

$$Risk_i(t) = \Pr\{Vulnerability_i \text{ in State 3 at time } t\} \times exploitation\_impact_i$$

If the system behavior can be approximated using a Markov process, the probability that a system is in a specific state at  $t$  could be obtained by using Markov modeling. Computational methods for semi-Markov [27] and non-Markov [28] processes exist, however, since they are complex, we illustrate the approach using the Markov assumption. Since the process starts at State 0, the vector giving the initial probabilities is  $\alpha = (P_0(0) P_1(0) P_2(0) P_3(0) P_4(0) P_5(0)) = (1 \ 0 \ 0 \ 0 \ 0 \ 0)$ , where  $P_i(t)$  represents the probability that a system is in State  $i$  at time  $t$ . Let  $\mathbb{P}(t)$  be as the state transition matrix for a single vulnerability where  $t$  is a discrete point in time. Let the  $x^{\text{th}}$  element in a row vector of  $v$  as  $v_x$ , then the probability that a system is in State 3 at time  $n$  is  $(\alpha \prod_{t=1}^n \mathbb{P}(t))_3$ . Therefore, according to the Equation (1), the risk for a vulnerability  $i$  for time window  $(0, t)$  is:

$$Risk_i(t) = (\alpha \prod_{k=1}^t \mathbb{P}_i(k))_3 \times impact_i \quad (5)$$

The impact may be estimated from the CVSS scores for Confidentiality Impact ( $I_C$ ), Integrity Impact ( $I_I$ ) and Availability Impact ( $I_A$ ) of the specific vulnerability, along with the weighting factors specific to the system being compromised. It can be expressed as:

$$impact_i = f_c(I_C R_C, I_I R_I, I_A R_A)$$

where  $f_c$  is a suitably chosen function. CVSS defines environmental metrics termed *Confidentiality Requirement*, *Integrity Requirement* and *Availability Requirement* that can be used for  $R_C$ ,  $R_I$  and  $R_A$ . The function  $f_c$  may be chosen to be additive or multiplicative. CVSS also defines a somewhat complex measure termed *Adjusted Impact*, although no justification is explicitly provided. A suitable choice of the impact function needs further research.

We now generalize the above discussion to the general case when there are multiple potential vulnerabilities in a software system. If we assume statistical independence of the vulnerabilities (occurrence of an event for one vulnerability is not influenced by the state of other vulnerabilities), the total risk in a software system can be obtained by the risk due to each single vulnerability given by Equation (5). We can measure risk level as given below for a specific software system.

$$Risk(t) = \sum_i (\alpha \prod_{k=1}^t \mathbb{P}_i(k))_3 \times impact_i$$

The method proposed here could be utilized to measure risks for various units, from single software on a machine to an organization-wide risk due to a specific software. Estimating the organizational risk would involve evaluating the vulnerability risk levels for systems installed in the organizations. The projected organizational risk values can be used for optimization of remediation within the organization.

## 7 Risk from known unpatches vulnerabilities

It can take considerable effort to estimate the transition rates among the states as described in the previous section. A conditional risk measure for a software system could be defined in terms of the intervals between the disclosure and patch availability dates that represent the gaps during which the vulnerabilities are exposed.

We can use CVSS metrics to assess the threat posed by a vulnerability. Let us make a preliminary assumption that the relationships between the Likelihood ( $L$ ) and the Exploitability sub-score ( $ES$ ), as well as the Impact ( $I$ ) and the Impact sub-score ( $IS$ ) for a vulnerability  $i$  are linear:

$$ES_i = a_0 + a_1 \times L_i \quad \text{and} \quad IS_i = b_0 + b_1 \times I_i$$

Because the minimum values of  $ES$  and  $IS$  are zero,  $a_0$  and  $b_0$  are zero. That permits us to define normalized risk values, as can be seen below.

Now, a conditional risk,  $Risk\_c_i$ , for a vulnerability  $i$  can be stated as:

$$Risk\_c_i = L_i \times I_i = \frac{ES_i IS_i}{a_1 b_1}$$

For the aggregated conditional risk is:

$$Risk\_c = \frac{1}{a_1 b_1} \sum_i ES_i IS_i$$

A normalized risk measure  $Risk'_c(t)$  can be defined by multiplying the constant  $a_1 b_1$ , expressed as:

$$Risk'_c(t) = \sum_i ES_i(t) IS_i(t) \quad (6)$$

This serves as an aggregated risk measure for known and exposed vulnerabilities. Its estimation is illustrated below using numerical data.

Fig. 4 is a conceptual diagram to illustrate the risk gap between vulnerability discoveries and patch releases on top of the simplified three phase vulnerability lifecycle in AML model [20]. In the initial learning phase, the software is gaining market share gradually. In the linear phase, the discovery rate reaches the maximum due to the peak popularity of the

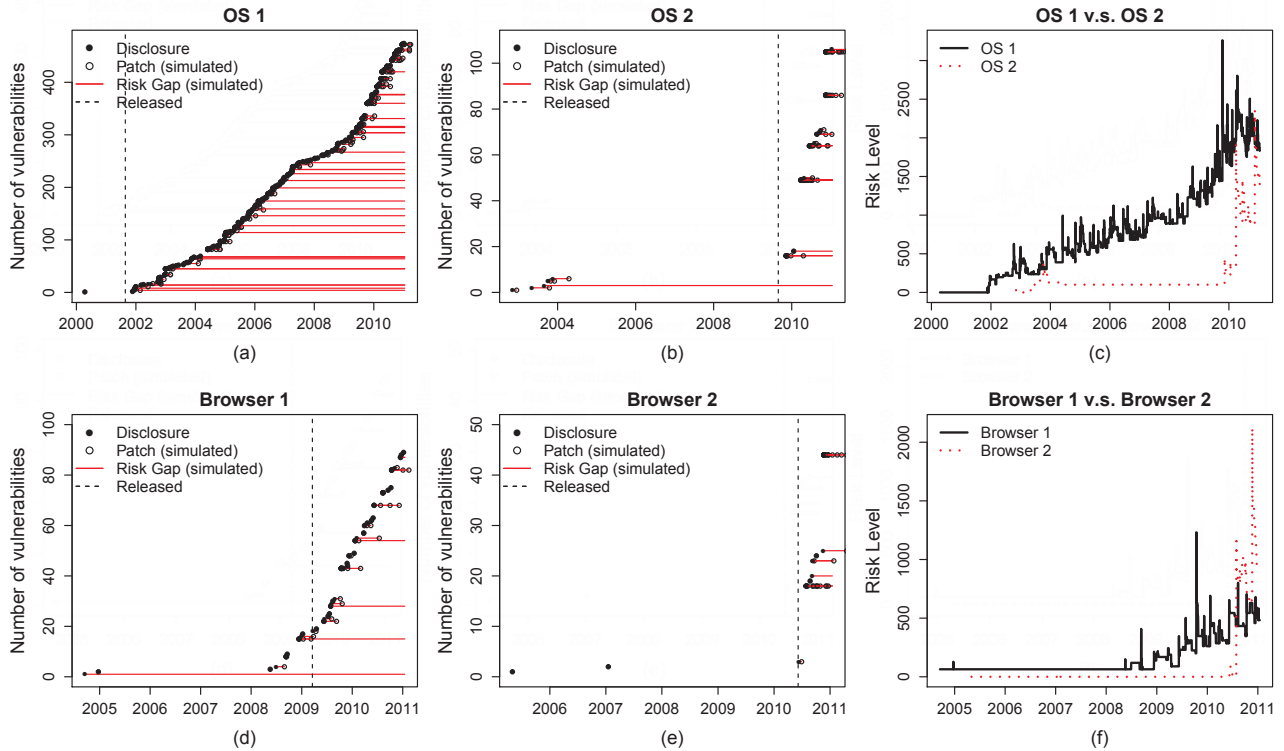


Figure 5. Evaluated risk gaps (a, b, d, e) and normalized risk level (c, f)

Table 1. Average patch time [11]

|           | 0-day | 30-day | 90-day | 180-day |
|-----------|-------|--------|--------|---------|
| Microsoft | 61%   | 75%    | 88%    | 94%     |
| Apple     | 32%   | 49%    | 71%    | 88%     |

Table 2. Simulated datasets for patch date

|                      | OS 1     | OS 2 | Browser 1 | Browser 2 |    |
|----------------------|----------|------|-----------|-----------|----|
| Simulated # of vuln. | 0 day    | 289  | 33        | 54        | 14 |
|                      | 1-30     | 66   | 18        | 12        | 7  |
|                      | 31-90    | 61   | 23        | 11        | 9  |
|                      | 91-180   | 28   | 18        | 5         | 7  |
|                      | No patch | 30   | 14        | 7         | 7  |
| Total [17]           | 474      | 106  | 89        | 44        |    |

software, and finally, in the saturation phase, vulnerability discovery rate slows down.

In the figure, each horizontal line represents the duration for an individual vulnerability from discovery date to patch availability date. When there are multiple dots at the right, the horizontal line represents multiple vulnerabilities discovered at the same time, but with different patch dates. A white dot is used when a patch is not hitherto available. For example, in Fig 4, at time  $t$  marked with the vertical red dashed line, there are nine known vulnerabilities with no patches. To calculate the conditional risk level at that time point, each single vulnerability risk level need to be calculated first and then added as shown in Equation (6).

We illustrate the approach using simulated data that has been synthesized using real data. Actual vulnerability disclosure dates [17] are used but the patch dates are simulated. XP is currently (Jan. 2011 [29]) the most popular OS with 55.26% share. Also, Snow Leopard is the most popular among non-Windows OSes. IE 8 and Safari 5 are the most adopted Web browsers for the two OSes. Considerable effort

and time would be needed for gathering the actual patch release dates [22], thus simulated patch dates are used here for the four systems. The patch dates are simulated using the aggregate data [11] representing the fraction of vulnerabilities patched, on average, within 0, 30, 90 and 180 days as shown in Table 1. Note that 6% and 12% of the vulnerabilities for Microsoft and Apple respectively are not patched by 180 days. Many of them are patched later, however because of lack of data, the simulated data treats them as unpatched vulnerabilities which would cause the data to differ from real data.

The simulated data sets are listed in Table 2; note that while OS 1, OS 2, Browser 1 and Browser 2 are based on XP, Snow Leopard, IE 8 and Safari 5 respectively, they are used here only to illustrate the procedure and not for evaluation the risk levels of the actual software.

Fig. 5 (a, b, d, e) give the risk gaps for the four datasets. The linear trend observed arises as special cases of the logistic process [30]. Fig. 5 (c, f) give the normalized risk levels calculated daily. As shown in the plots, OS 1 risk level has started to decline while OS 2 risk level is still rising. For the browsers, Browser 2 risk level rises sharply right after the release due to the two sets of vulnerability clusters with no available immediate patches. The long term rising trend observed might be caused by vulnerabilities we have presumed to be unpatched after 180 days. Since the data sets are simulated, the results only serve as an illustration of the approach and do not represent any actual products.

## 8 Conclusions

This paper presents formal measures of security risk that are amenable to evaluation using actual vulnerability data. It



also explores the relationship of CVSS metrics and scores with formal expressions of risk.

While a preliminary examination of some of the software lifecycle transitions has recently been done by some researchers [11][12], risk evaluation considering the vulnerability lifecycle has so far received very little attention. In this paper, a formal quantitative approach for software risk evaluation is presented which uses a stochastic model for the vulnerability lifecycle and the CVSS metrics. The model incorporates vulnerability discovery and potential 0-day attacks. The risk values for individual vulnerabilities can be combined to evaluate risk for an entire software system, which can in turn be used for evaluating the risk for an entire organization. A simplified approach for risks due to known but unpatched vulnerabilities is also given.

While some data has started to become available, further research is needed to develop methods for estimating the applicable transition rates [11][19][31]. In general, the computational approaches need to consider the governing probability distributions for the state sojourn times. Since the impact related scores may reflect a specific non-linear scale, formulation of the impact function also needs further research.

The proposed approach provides a systematic approach for software risk evaluation. It can be used for comparing the risk levels for alternative systems. The approach can be incorporated into a methodology for allocating resources optimally by both software developers and end users.

## 9 References

- [1] C. Alexander, *Market Risk Analysis: Quantitative Methods in Finance*, Wiley, 2008.
- [2] V. Verendel, Quantified security is a weak hypothesis: a critical survey of results and assumptions, Proc. 2009 workshop on New security paradigms workshop, Sept.08-11, 2009, Oxford, UK. pp. 37-49.
- [3] R. L. V. Scoy, Software development risk: Opportunity, not problem (cmu/sei-92-tr-030), Software Engineering Institute at Carnegie Mellon University, Pittsburgh, Pennsylvania, Tech. Rep., 1992.
- [4] S. Farrell, Why Didn't We Spot That?, *IEEE Internet Computing*, 14(1), 2010, pp. 84-87.
- [5] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, 3rd ed. Prentice Hall PTR, 2003.
- [6] National Institute of Standards and Technology (NIST), Risk management guide for information technology systems, 2001. Special Publication 800-30.
- [7] L. A. (Tony) Cox, Jr, Some Limitations of Risk = Threat  $\times$  Vulnerability  $\times$  Consequence for Risk Analysis of Terrorist Attacks, *Risk Analysis*, 28(6), 2008, pp. 1749-1761.
- [8] Open Web Application Security Project (OWASP) Top 10 2010 - The Ten Most Critical Web Application Security Risks, [http://www.owasp.org/index.php/Top\\_10\\_2010-Main](http://www.owasp.org/index.php/Top_10_2010-Main)
- [9] H. Joh and Y. K. Malaiya, A Framework for Software Security Risk Evaluation using the Vulnerability Lifecycle and CVSS Metrics, Proc. International Workshop on Risk and Trust in Extended Enterprises, November 2010, pp. 430-434.
- [10] P. Mell, K. Scarfone, and S. Romanosky, CVSS: A complete Guide to the Common Vulnerability Scoring System Version 2.0, Forum of Incident Response and Security Teams (FIRST), 2007.
- [11] S. Frei, *Security Econometrics: The Dynamics of (IN)Security*, Ph.D. dissertation at ETH Zurich, 2009.
- [12] W. A. Arbaugh, W. L. Fithen, and J. McHugh, Windows of vulnerability: A case study analysis, *Computer*, 33(12), 2000, pp. 52-59.
- [13] P. A. Engert, Z. F. Lansdowne, Risk Matrix 2.20 User's Guide, November 1999, <http://www.mitre.org/work/sepo/toolkits/risk/ToolsTechniques/files/UserGuide220.pdf>
- [14] J. P. Brashear, J. W. Jones, Risk Analysis and Management for Critical Asset Protection (RAMCAP Plus), Wiley Handbook of Science and Technology for Homeland Security, 2008.
- [15] European Network and Information Security Agency (ENISA), Cloud Computing - Benefits, risks and recommendations for information security, Ed. Daniele Catteddu and Giles Hogben, Nov 2009.
- [16] L. Cobb, A Scale for Measuring Very Rare Events, April, 1998, <http://www.aetheling.com/docs/Rarity.htm>
- [17] NIST, National Vulnerability Database (NVD), <http://nvd.nist.gov/>, Accessed on Feb. 2011
- [18] I. Mkpog-Ruffin, D. Umphress, J. Hamilton, and J. Gilbert, Quantitative software security risk assessment model, ACM workshop on Quality of protection, 2007, pp. 31-33.
- [19] S. H. Houmb and V. N. L. Franqueira, Estimating ToE Risk Level Using CVSS, International Conference on Availability, Reliability and Security, 2009, pp.718-725.
- [20] O. H. Alhazmi and Y. K. Malaiya, Application of vulnerability discovery models to major operating systems, Reliability, *IEEE Transactions on*, 57(1), 2008, pp. 14-22.
- [21] S.-W. Woo, H. Joh, O. H. Alhazmi and Y. K. Malaiya, Modeling Vulnerability Discovery Process in Apache and IIS HTTP Servers, *Computers & Security*, Vol 30(1), pp. 50-62, Jan. 2011
- [22] A. Arora, R. Krishnan, R. Telang, and Y. Yang, An Empirical Analysis of Software Vendors' Patch Release Behavior: Impact of Vulnerability Disclosure, *Information Systems Research*, 21(1), 2010, pp. 115-132.
- [23] E. Levy, Approaching Zero, *IEEE Security and Privacy*, 2(4), 2004, pp. 65-66.
- [24] R. Ayoub. An analysis of vulnerability discovery and disclosure: Keeping one step ahead of the enemy, Tech. Report, Frost & Sullivan, 2007.
- [25] Norwegian Honeynet Project, Time to Exploit, <http://www.honeynor.no/research/time2exploit/>, Accessed on Feb. 2011
- [26] S. Beattie, S. Arnold, C. Cowan, P. Wagle, and C. Wright, Timing the application of security patches for optimal uptime, Proceedings of the 16th USENIX conference on System administration, Berkeley, CA, 2002, pp. 233-242.
- [27] V. S. Barbu, and N. Limnios, *Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications: Their Use in Reliability and DNS Analysis*, Springer, New York, 2008.
- [28] Y. K. Malaiya and S. Y. H. Su, Analysis of an Important Class of Non-Markov Systems, *IEEE Transactions on Reliability*, R-31(1), April 1982, pp. 64 - 68.
- [29] NetMarketShare, Operating System Market Share, <http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=10>, Accessed on Feb. 2011
- [30] G. Schryen, Security of open source and closed source software: An empirical comparison of published vulnerabilities. Proceedings of the 15th Americas Conference on Information Systems. 2009.
- [31] M. D. Penta, L. Cerulo, and L. Aversano, The life and death of statically detected vulnerabilities: An empirical study, *Information and Software Technology*, 51(10), 2009, pp. 1469-1484.