A Framework for Software Security Risk Evaluation using the Vulnerability Lifecycle and CVSS Metrics

HyunChul Joh and Yashwant K. Malaiya
Computer Science Department
Colorado State University
Fort Collins, CO, USA
{dean2026, malaiya}@cs.colostate.edu

***Abstract*— A vulnerability that has been discovered but is unpatched represents a security risk to a system. During the lifetime of a software system, new vulnerabilities are discovered over time. There are two opposing actors, the patch developers and the potential exploiters. An exploit can happen immediately after a disclosure, perhaps even before the disclosure if the discovery is made by a black-hat finder. Here, a framework for software risk evaluation with respect to the vulnerability lifecycle is proposed. Risk can be evaluated using the likelihood of a security breach and the impact of that adverse event on the system. The proposed approach models the vulnerability lifecycle as a stochastic process. Some of the CVSS metrics can be used to evaluate the impact of the breach. The model uses the information about the transition rates with the related distributions and can lead to simplified as well as detailed modeling methods. It allows a comparison of software systems in terms of the risk and potential approaches for optimization of remediation.**

Keywords- Security vulnerabilities; Vulnerability Risk Index (VRI); CVSS; Vulnerability lifecycle

I. INTRODUCTION

Quantitative measures are now commonly used to measure some attributes of computing such as performance, availability 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 some criticism of the quantitative attempts [2] due to the lack of data for validating quantitative methods, but still such methods have been used for comparing alternative software systems including comparisons of the number of vulnerabilities found. However, it can be argued that vulnerabilities that have been found and remedied using patches do not represent a risk, while the vulnerabilities that are likely to be found or to remain unpatched for some period represent risk. Considering that today banking, stock market trading, travelling, dating, critically depends on the Internet based computing, the risk to the society due to exploitation of vulnerabilities is massive. Yet the society is willing to take the risk, since the Internet has made the markets and the transactions much more efficient [3].

In spite of recent advances in secure coding, it is unlikely that completely secure systems will become possible anytime soon [4]. Thus, it is necessary to take risk and take precautionary measures that are commensurate. To keep the overall risk within acceptable limits, people need to measure risks in their system. As Lord Calvin stated "If you cannot measure it, you cannot improve it. [3]"

Informally, risk is sometimes stated as the probability that an asset will suffer an event of a given negative impact [5] or the possibility of a harm to occur [6]. Formally, risk has to be a weighted measure depending on the consequence. A widely used expression for risk can be stated as [7]:

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

This presumes a specific time period for the evaluated likelihood such as a year for *annual loss expectancy*. 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-level matrix is often constructed that divides both likelihood and impact values into discrete ranges that can be used to classify applicable causes [8]. Sometimes both the likelihood and the impact are measured using scores that use a logarithmic scale. In case of a security vulnerability, a successful breach due to a vulnerability constitutes an adverse event [9] which can impact one or more of the security attributes. Recently, Cox [8] has pointed out the need for a careful interpretation of the terms and possible need for refining the computational approaches when traditional risk equations are used. However, there are no clear alternatives to the widely accepted fundamental formulations for risk such as Equations (1).

In this paper, risk is defined from the point of view of the software vulnerability lifecycle, considering both the probability that a software vulnerability in a system will be exploited and the impact of exploitation. A vulnerability is defined as software defect or weakness in the security system which might be exploited by a malicious user causing loss or harm [6]. With respect to the Equation (1), a stochastic model of the vulnerability lifecycle is used for calculating the *Likelihood of an adverse event* while impact related metrics from the Common Vulnerability Scoring System (CVSS) [10] are utilized for estimating *Impact of the adverse event*. When the risk analysis uses only qualitative measurement, it is likely that the analysis may turn out to be very subjective in the end. Here we propose a framework for risk analysis that can be used for either detailed modeling or for arriving at reasonable approximations.

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 reviews recent related work involving CVSS metrics. Section 3 introduces the software vulnerability lifecycle. Section 4 gives the mathematical formulations for risk including their illustration. Section 5 analyzes a simplified model. Finally, discussion is presented followed by conclusion identifying the future research needs.

II. CVSS METRICS AND RELATED WORK

A few researchers have started to use CVSS scores for analyzing their security risk models. The CVSS score system provides vendor independent framework for communicating the characteristics and impacts of known vulnerabilities [10].

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 index is associated with each level. CVSS is composed of three metric groups: Base, Temporal and Environmental. The base metrics represent the intrinsic characteristics of a vulnerability, and are the only mandatory metrics. 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-groups, exploitability and impact metrics. Formulas for CVSS scores, calculated using the metrics, are chosen and adjusted such that a score is a decimal number in the range [0.0, 10.0]. In the proposed approach, we use the impact metrics which measure how a vulnerability will directly affect an IT asset in terms of the degree of losses in confidentiality, integrity, and availability. The impact attributes are all assessed in terms of None, Partial, or Complete. The base metrics can be easily found from the publically available vulnerability databases.

Stango et al. [13] have proposed a general method for threat analysis, and have pointed out that it is hard to prioritize threats due to the lack of effective metrics and the complex and sensitive nature of security. They combine the Bruce Schneier's attack trees [14] and the CVSS scoring system. CVSS scores are assigned to the attack tree nodes to evaluate security. Mkpong-Ruffin et al. [15] use empirical data about the security attributes of each vulnerability 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.

Wang et al. [16] propose an ontology-based approach for analyzing the security for software products. They first create an ontology for vulnerability management which has all the information about vulnerabilities based on widely accepted standards such as CVSS, CVE, CWE, CPE, and CAPEC[1]. Then they calculate overall environmental score for given product according to the proposed algorithm. Houmb et al. [17][18] have discussed a model for the quantitative estimation of the security risk level of a system called CVSS risk level estimation model by utilizing Bayesian Belief Network topology. They estimate frequency and impact of vulnerabilities using reorganized original CVSS metrics. And, finally, the two estimated measures are combined to calculate risk levels.

This paper explicitly considers a vulnerability lifecycle for evaluating risk levels which has not been done before.

III. SOFTWARE VULNERABILITY LIFECYCLE

A vulnerability is created accidently from the careless coding mistake. After the birth, the first event could be the discovery. Sometimes a vulnerability can be patched unnoticeably when other vulnerabilities are patched. The discovery rate can be described by vulnerability discovery models (VDM) [19]. It has been shown that VDMs are also applicable when the vulnerabilities are partitioned according to underlying cause or severity levels [20]. In some cases it has been found that a linear VDM holds [21] for a sufficiently long period suggesting a stable discovery rate for that duration. Some of the CVSS base and temporal metrics assess the ease of exploiting a vulnerability [10].

When a white hat researcher discovers a vulnerability, the next transition is likely to be the internal disclosure leading to patch development. On the other hand, if a black hat hacker discovers a vulnerability, the next transition could be an exploit or internal disclosure to his underground community. Some active black hats might develop scripts that exploit the vulnerability.

After being noticed by white hat researcher, software vendors are given a few days to release the patch. Typically, vendors are given 30 or 45 days for to develop patches [22]. On the other hand, if the disclosure event occurred within a black hat community, the next possible transition may be the exploit or script. Informally, practitioners use the term *zero day vulnerability* to refer to unpublished vulnerability that are actively 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[2] found that from the public disclosure to the exploit event takes a median of 5 days (data is highly asymmetric).

When a script is available, it enhances the probability of an exploitation. 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. Sometimes a patch can inject a new vulnerability. Beattie et al. [25] have examined 136 CVE entries, and they found that 92 patches were good patches, 20 were revised or pulled patches, and 24 had no patches.

Frei has [11] found that 78% of the examined exploitations occur within a day, and 94% by 30 days from the public disclosure day. He also found that a vulnerability could be exploited before a script is available. Sometimes

[1] http://measurablesecurity.mitre.org/

[2] http://www.honeynor.no/research/time2exploit/

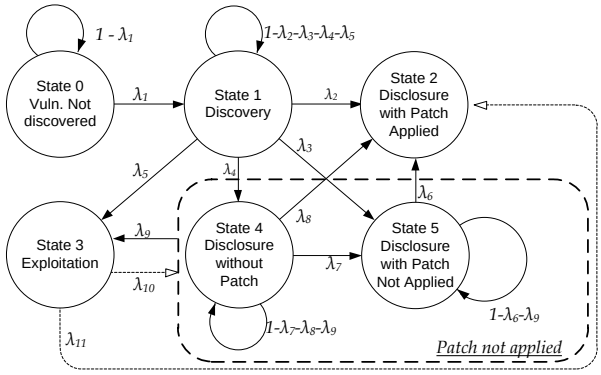

Figure 1. Stochastic model for a single vulnerability

exploitation occurs before releasing the patch. In addition, he analyzed the distribution of discovery, exploit, and patch time of vulnerabilities with respect to the public disclosure date based on very large datasets from the major vulnerability databases and security information advisories. He reports that the time from public disclosure to the exploitation fits a Pareto distribution. The time from public disclosure to patch availability follows Pareto distribution for negative values and Weibull distribution for positive values.

IV. EVALUATING THE RISK LEVEL

We first consider evaluation of the risk on account of a single vulnerability. Later the result is generalized to include all potential vulnerabilities in a software system. Fig. 1 models the lifecycle of a single vulnerability defined by the 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 black hats, there is a chance it will be soon exploited. State 2 represents the situation when the vulnerability is disclosed along with the patch release and the patch is applied to software right away. Hence, the state is safe state and is an absorbing state. In State 5, the vulnerability is disclosed with 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 an exploitation which leads to State 3. The two dashed arrows 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 Fig. 1, for a single vulnerability, risk in a specific system at time t can be expressed as probability of the vulnerability being in State 3 at time t multiplied by the consequence of the vulnerability exploitation.

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

If the system behavior can be approximated using a Markov process, the probability that a system is in State 3 at t could be obtained by using Markov modeling. Computational methods for semi-Markov [26] and non-Markov [27] processes exist. However, since they are more complex, we illustrate the approach using the Markov assumption. Since the process in Fig. 1 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 Fig. 1 where t is elapsed discrete time point. Hence, at the 1st time step, the transition matrix is $\mathbb{P}(1)$, at the 2nd time step, the transition matrix is $\mathbb{P}(1) \times \mathbb{P}(2), \cdots$, and the n^{th} time step transition matrix is $\mathbb{P}(1) \times \mathbb{P}(2) \times \cdots \times \mathbb{P}(n) = \prod_{t=1}^{n} \mathbb{P}(t)$. Let the x^{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 at time t is given as:

$$Risk_i(t) = (\alpha \prod_{k=1}^{t} \mathbb{P}_i(k))_3 \times exploitation\_impact_i \quad (2)$$

The exploitation 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:

$$exploitation\_impact_i = f_c(I_C R_C, I_I R_I, I_A R_A) \quad (3)$$

where f_c is a suitably chosen function. CVSS defines environmental metrics termed *Confidentiality Requirement*, *Integrity Requirement* and *Availability Requirement* that can used for R_C, R_I and R_A. The function f_c may be chosen to be additive or multiplicative (if it is felt that the scale used for scores is effectively logarithmic). CVSS also defines a somewhat complex measure termed *AdjustedImpact*, although no justification is explicitly provided. It will result in an additive effect when the impact scores are small. Further, CVSS specifies a measure *Impact_Score* that does not use environmental metrics. Houmb and Franqueira [18] define a Misuse Impact score as a three element vector based on base and environmental metrics. A suitable choice of the impact function needs further research.

In some cases, the risk is measured as the weighted impact of possible exploitations within a specific time window Δ, for example, a year. We can compute this period risk for duration $(t, t + \Delta)$ as:

$$Risk_i(t, \Delta) = \int_0^{\Delta} (\Pr\{A\} \times \Pr\{B\}) \times exploitation\_impact_i \quad (4)$$

where $\{A\}$ and $\{B\}$ are {vulnerability$_i$ is in an exposed state at $t + \tau$} and {exploitation from the exposed state occurs in $d\tau$} respectively. Note that this allows taking multiple exploitations into account within the same time window. The risk to specific organizations with many systems or to the community of users of that software would depend on the number of systems affected.

We now generalize the above discussion to the realistic case when there should be 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 another vulnerability), the total risk in a software system can be obtained by the risk due to each single vulnerability given by

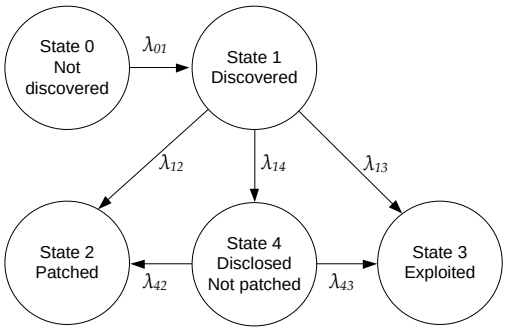

Figure 2. A simplified model of the vulnerability lifecycle

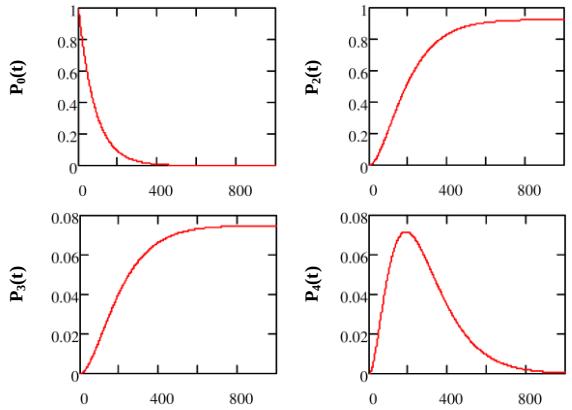

Figure 3. Computed probabilities; P_0, P_2, P_3 and P_4 for Fig. 3. [x-axis represents time *t*]

Equation (2). We can define a measure *Vulnerability Risk Index* (*VRI*) as given below to represent the risk level at time *t* for a single software system.

$$VRI(t) = \sum_i (\alpha \prod_{k=1}^{t} \mathbb{P}_i(k))_3 \times exploitation\_impact_i \quad (5)$$

In some cases, an exploitation requires presence of two or more vulnerabilities. That can be taken into account by using the probability of the two specific vulnerabilities being in the exposed state at the same time in Equation (4).

The developed framework could be utilized to measure risks in various of scales from a single software vulnerability risk to an organization wide software related risk. Estimating the organization wide risk could be achieved based on measuring the vulnerability risk indices for software systems installed in the organizations. Finally, the projected risk values could be speculated for optimization of remediation.

V. Analyzing a Simplified Model

Fig. 2 shows a simplified single vulnerability lifecycle which is used to quantitatively illustrate the software risk analysis using continuous time Markov modeling. In State 0 the vulnerability is not yet discovered. Discovery of the vulnerability causes the process to go to State 1. State 2 represents the situation when the vulnerability has been patched. In State 4 the vulnerability has been disclosed, but a patch is not available or has not been applied. State 3 is entered when a successful exploitation occurs.

The process of Fig. 2 can be represented by a system of differential equations for the system probabilities. The system of differential equations can be solved by assuming that initial state is State 0 or $P_0(0) = 1$. While in general the solutions may have to be obtained using numerical computation, the system of equations for Fig. 2 is actually simple enough for a closed-form solution. Fig. 3 gives plots of the state probabilities using some assumed transition values for illustration. In general, we can expect that $\lambda_{42} \gg \lambda_{43}$ and $\lambda_{12} \gg \lambda_{14} \gg \lambda_{13}$ when we consider that majority of the vulnerabilities is not suffering from possible major exploitations. Fig. 3 shows the computed probabilities with some reasonable values of λ along with the time line. The probability $P_0(t)$ of staying in State 0 drops as time goes by. $P_4(t)$, the probability of being in the exposed state rises sharply and then peaks, as a transition to either State 2 or State 3 occurs, which are both absorbing states. Eventually about 93% of the time, the process ends up in the safe State 2, and about 7% of the time exploitation occurs when State 3 is entered. The long term risk is given by the product of $P_3(t)$ and the corresponding impact. The period risk for a given duration can be calculated by using equation (5).

VI. Discussion

Risk evaluation considering the software vulnerability lifecycle has been established very little attention while a preliminary examination of some of the software lifecycle transitions has recently been done by some researchers [11][12]. In this paper, an approach for software risk evaluation is presented which uses a stochastic model for the vulnerability lifecycle, along with the CVSS impact metrics.

The stochastic model is used to calculate the probabilities of the process being in a specific state while CVSS metrics are used to estimate the impact of an exploitation. If we can assume that the transitions at Fig. 1 and Fig. 2 are governed by an exponential distribution, Markov modeling can be used. If not, the stochastic process can be modeled as a semi-Markov process [26] for computing the state probabilities.

VII. Conclusion

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

While some data has started to become available, research is needed to develop methods for estimating the applicable transition rates [11][18][28]. 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.

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] D. Verdon and G. McGraw. Risk analysis i*n software de*sign. *IEEE Security and Privacy*, 2(4), 2004, pp. 79–84.

[6] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, 3rd ed. Prentice Hall PTR, 2003.

[7] National Institute of Standards and Technology (NIST), Risk management guide for information technology systems, 2001. Special Publication 800-30.

[8] L. A. (Tony) Cox, Jr, Some Limitations of Risk = Threat × Vulnerability × nsequence for Risk Analysis of Terrorist Attacks, *Risk Analysis*, 28(6), 2008, pp. 1749-1761.

[9] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. Mcdermid, and D. Gollmann, Towards Operational Measures of Computer Security, *Journal of Computer Security*, Vol. 2, 1993, pp. 211-229.

[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] A. Stango, N. R. Prasad, and D. M. Kyriazanos, A threat analysis methodology for security evaluation and enhancement planning, Third International Conference on Emerging Security Information, Systems and Technologies, 2009, pp. 262–267.

[14] B. Schneier, Attack trees, *Dr. Dobb's Journal of Software Tools*, 24(12), 1999, pp. 21–29.

[15] I. Mkpong-Ruffin, D. Umphress, J. Hamilton, and J. Gilbert, Quantitative software security risk assessment model, ACM workshop on Quality of protection, 2007, pp. 31–33.

[16] J. A. Wang, M. Guo, H. Wang, M. Xia, and L. Zhou, Environmental metrics for software security based on a vulnerability ontology, in Secure Software Integration and Reliability Improvement, 2009, pp. 159 –168.

[17] S. H. Houmb, V. N. Franqueira, and E. A. Engum, Quantifying security risk level from cvss estimates of frequency and impact, *Journal of Systems and Software*, 83(9), 2010, pp. 1622-1634.

[18] 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.

[19] 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.

[20] S-W. Woo, O. H. Alhazmi and Y. K. Malaiya, An Analysis of the Vulnerability Discovery Process in Web Browsers, Proc. 10th IASTED Int. Conf. on Software Engineering and Applications, Nov. 2006, pp. 172-177.

[21] O. H. Alhazmi, Y. K. Malaiya and I. Ray, Measuring, Analyzing and Predicting Security Vulnerabilities in Software Systems, *Computers & Security Journal*, 26(3), May 2007, pp. 219-228.

[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] 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.

[26] 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, NY, 2008.

[27] 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.

[28] 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.