Time Lag-Based Modelling for Software Vulnerability Exploitation Process

Adarsh Anand[1], Navneet Bhatt[1,*], Jasmine Kaur[1] and Yoshinobu Tamura[2]

[1]*Department of Operational Research, University of Delhi, India*
[2]*Department of Industrial & Management Systems Engineering, Faculty of Knowledge Engineering, Tokyo City University, Japan*
E-mail: adarsh.anand86@gmail.com; navneetbhatt@live.com; jasminekaur.du.aor@gmail.com; tamuray@tcu.ac.jp
[*]*Corresponding Author*

Received 10 January 2021; Accepted 15 March 2021;
Publication 14 June 2021

Abstract

With the increase in the discovery of vulnerabilities, the expected exploits occurred in various software platform has shown an increased growth with respect to time. Only after being discovered, the potential vulnerabilities might be exploited. There exists a finite time lag in the exploitation process; from the moment the hackers get information about the discovery of a vulnerability and the time required in the final exploitation. By making use of the time lag approach, we have developed a framework for the vulnerability exploitation process that occurred in multiple stages. The time lag between the discovery and exploitation of a vulnerability has been bridged via the memory kernel function over a finite time interval. The applicability of the proposed model has been validated using various software exploit datasets.

Keywords: Exploits, patch, security, updates, vulnerability, vulnerability discovery models.

Journal of Cyber Security and Mobility, Vol. 10_4, 663–678.
doi: 10.13052/jcsm2245-1439.1042

1 Introduction

The rapid growth of new technology has impacted the concern of software firms towards the security profile of a software product. In spite of upgrading a software product with the addition of new features, the software program still contains various flaws. The addition of new code and features consequently increases the potential flaws present in the software and hence affects the security of the software system. In the operational phase of software, some bugs that are identified are more dangerous than the others and may weaken the security profile of the software system. Thus, these flaws are noted as software vulnerabilities. A number of definitions for 'vulnerability' have been proposed, but the definition given by Krsul highlights all the different areas in which a vulnerability can be originated as "an instance of a mistake in the specification, development or implementation of a software such that its execution may violate the security policy" [1]. These flaws are accidentally created during the software development life cycle (SDLC) and later came into the existence as a vulnerability and are required to be patched speedily. The impact of these vulnerabilities ranges from inconvenience to economic damage. A well-known vulnerability named Code-Red contained in Windows operating system affected more than three million computers worldwide. Another vulnerability caused by the 'Slammer worm' was discovered in 2003 and it exploited 75 thousand computers within 15 minutes. Likewise, in Oct. 2014, a flaw existed in Windows operating system that allowed the intruders to remotely install malware into the target computer.

Vulnerabilities are normally discovered during the operational phase of software, and, the discovery is majorly credited to the external detectors available in the field. If a vulnerability has been discovered by a black hat user then the probability of exploiting the vulnerability significantly rises. Instead of alerting the software vendor, the black hat user might exploit the vulnerability himself and likely sell the information to the hacker group. A vulnerability in software undergoes a life cycle, going from first identification to its eventual patching. As described by Ozment, the various stages accompanied by a vulnerability can be described as following events [2]:

- Injection date: "It is the date on which a vulnerability is first checked in the source code or the code is built or compiled."
- Discovery date: "It corresponds to the date on which a loophole is first detected."
- Disclosure date: "It is the date on which the vendor is notified by the detector."

- Public date: "It corresponds to the event on which a detailed description of the vulnerability is made publicly known."
- Patch date: "The patch date is the time instance on which a fix is supplied for the vulnerability."
- Exploit date: "It is the date on which an exploit for the vulnerability is released."

The collection of different vulnerabilities identified in software systems have been maintained by different databases such as National Vulnerability Database (NVD), CERT, Exploit Database (EDB), and security focus. The organizations like CERT inform the vendor when a vulnerability is detected and allocates a time interval for delivering a patch for the discovered vulnerabilities. After the time window, the vulnerability information is revealed to the public. Similarly, organizations like EDB provide vulnerability exploit information for the vulnerabilities that can be exploited and failure in providing a suitable patch for the vulnerabilities might be disastrous for both users and vendor(s).

This article attempts to provide a mathematical framework that can model the vulnerability exploit process by making use of vulnerability discovery models (VDMs). To outline the discovery of a vulnerability in a software system, various VDMs are available to predict the rate at which vulnerabilities are discovered. The VDMs are time-based models that help in assessing the security profile of a software system by determining the loopholes present in the software, and the respective rate at which vulnerabilities are identified. Further, the VDMs can be utilized to assess the information related to exploited vulnerabilities as those vulnerabilities which are not patched might get exploited based on their characteristics. Thus, the goal of this paper is to develop a model that can furnish a functional relationship that exists between the vulnerability discovery and vulnerability exploit phenomenon. Our model allows a software vendor to assess the exploit status of vulnerabilities in software and to allocate resources in the development of patches. To our knowledge, no previous work has used VDMs in studying the exploit phenomenon of vulnerabilities. The exploits occur after a time-lag in the vulnerability discovery process. In this paper, we have considered that after the discovery of vulnerabilities in a software system a significant time lag happens after that only the vulnerabilities are exploited. The proposal helps in knowing the exploit trend based on the number of vulnerabilities discovered.

The paper is divided into various sections. In Section 2, we review the relevant literature. Section 3 provides the proposed modeling framework describing the vulnerability exploit phenomenon based on the discovered

vulnerabilities. In Section 4, we present an empirical illustration of the developed proposal. Finally, Section 5 concludes the work.

2 Literature

The classification of vulnerability discovery models (VDMs) is grouped in time-based and effort-based modeling. The prior captures the vulnerabilities discovered with respect to the time, and the latter predicts the vulnerabilities based on the efforts applied. The criteria to predict the vulnerabilities considered time as the governing factor, which was also a major attribute of most of the VDM papers in the literature. It was Anderson who shows the vulnerability discovery phenomenon follows a similar software reliability growth modeling trend [3]. In particular, Rescorla estimates linear and exponential trends in the discovery of vulnerabilities [4]. The models performed well in the case of the Redhat 7.0 version, however unable to capture the trend in the case of WinNT4, Solaris 2.5.1, and FreeBSD datasets. Later, Alhazmi and Malaiya developed an S-shaped logistic growth model (AML) to anticipate the behavior of vulnerabilities discovered as per the learning phenomenon accompanied by the users [5]. The AML model fitted the data sets of different types of software effectively and closely. Of late, many researchers have deduced VDMs based on different discovery patterns [6–13].

The vulnerability exploit phenomenon has received increasing attention in the domain of cybersecurity. However, there exists little work in this line of research as related to the works proposed for predicting vulnerabilities in a software system. The majority of work has been done in predicting the cyber exploit based on the machine learning approach. Bozorgi et al. [14] considered a model that gathers features from a database namely Open Source Vulnerability Database (OSVDB) which is now discontinued to predict the exploits based on the Proof of Concepts (PoCs) availability. In their work, 73% of vulnerabilities were recorded as exploited as compared to ones recorded in the literature [15, 16]. Later, Sabottke et al. [17] developed an exploit prediction model using a dataset acquired from Twitter having links to CVE-IDs and from Symantec threat signatures for the positive labels. Of late, Almukaynizi et al. [18] deduced a model that considers data from various sources to predict the likelihood of exploitation, and Bhatt et al. [19] developed an exploit prediction framework and claimed it a highly effective approach for the exploit that could be seen in the wild.

3 Mathematical Modeling

The proposal is developed based on the following assumptions that have been considered in this research work:

- The potential number of vulnerabilities is fixed during the discovery process.
- The number of vulnerabilities that are discovered at any time point t is directly proportional to the remaining number of potential vulnerabilities which are undiscovered at that time t.
- Discovery and exploitation process are connected to each other.
- The exploitation takes place after the vulnerabilities are discovered in software.

This section is divided into two major parts. In the first section, we talk about the vulnerability discovery phenomenon and its modeling, and the second part discusses the vulnerability exploit process for the discovery modeling. These two categories provide a brief outline of the discovery phenomenon for the exploited vulnerabilities.

3.1 Vulnerability Discovery Process

During this stage, vulnerabilities are discovered in a software system. The vulnerability discovery phenomenon considered during this stage is considered as proposed by Rescorla [4]. The model assumed that the number of vulnerabilities discovered at any time point or the vulnerability intensity is proportional to the remaining number of undiscovered vulnerabilities.

The differential equation depicting the discovery scenario can be mathematically modeled as:

$$\frac{d\Omega_1(t)}{dt} \propto (N - \Omega_1(t)) \tag{1}$$

where N, the total number of the potential vulnerabilities; $\Omega_1(t)$, the cumulative number of discovered vulnerabilities by time t. The vulnerability discovery process will capture the left-over undiscovered vulnerabilities with a constant rate α. The above differential Equation (1) can be written as:

$$\frac{d\Omega_1(t)}{dt} = \alpha(N - \Omega_1(t)) \tag{2}$$

The solution of the above Equation (2) can be found using the initial condition, $\Omega_1(0) = 0$. A closed form of the above Equation (2) can be written as:

$$\Omega_1(t) = N(1 - e^{-\alpha t}) \tag{3}$$

Equation (3) represents the number of active vulnerabilities discovered by time *t* which might get exploited. The discovery model has been obtained by using the non-decreasing mean value function which follows the exponential growth pattern in discovery process. Further, other forms of discovery patterns can also be used such as logistic, hump-shaped, *etc.*

3.2 Vulnerability Exploitation Process

The objective of this research is to study the effect of the time lag between vulnerability discovery and its exploitation. The time lag approach has been utilized by making use of distributed time lag approach. The time delay during the exploitation process is established as a weighted response. The time delay is measured over a definite interval of time using the appropriate memory kernel. Since, after the discovery of vulnerabilities there exists a finite time lag before the software vulnerability is exploited. Hence, the functional relationship between the discovery and exploitation process is discussed below. The vulnerabilities which got discovered in the previous stage will become the potential vulnerabilities that might be exploited for this stage, as some of the vulnerabilities would be exploited after the discovery. Hence, the vulnerabilities discovered as given in Equation (4) can be considered as an upper limit for the next stage. Hence, the equation for exploitation process can be written as:

$$\frac{d\Omega_2(t)}{dt} = b(\Omega_1(t) - \Omega_2(t)) \tag{4}$$

where $\Omega_2(t)$, the cumulative number of vulnerabilities exploited by time *t*; *b*, constant rate of exploitation process.

To capture the time gap between the vulnerability discovery and its final exploitation, a distributed time lag approach has been utilized to understand the exploitation process. Further, using the ideology given by Diamond it has been assumed that the vulnerabilities discovered in past time would be exploited in the present time [20]. And, the time lag that has been considered is continuously distributed rather than discrete time lags as advocated by Cushing [21]. To describe this phenomenon, the influence of time delay in Equation (4) can be measured through an appropriate memory kernel

over a finite past time. Therefore, Equation (4) can be rewritten as an integro-differential equation in the presence of time lag as:

$$\frac{d\Omega_2(t)}{dt} = b\int_0^t K(t-\tau)(\Omega_1(t) - \Omega_2(t))d\tau \tag{5}$$

In this paper, we have limited ourselves by considering a weak memory kernel form for the analysis, i.e.,

$$K(t) = ve^{-vt} \tag{6}$$

where, *v* represents the parameter or rate at which past has a bearing on the present. In Equation (6), v^{-1} can be described as the time scale of the system. So, with the passage of time the weighted response of kernel gets influenced and falls exponentially. It implies that the kernel would be less reliable when the past is remoter [21]. The reason for considering the weak memory kernel function is because as soon as the vulnerabilities are discovered the chances of getting exploited is more as the patches for the vulnerabilities are normally provided by the vendors instantly and then exploitation would not be possible.

In order to solve Equation (5), the expressions for $\Omega_1(t)$ and the kernel function from Equations (3) and (6) are plugged into it. Then, the Equation (5) can be written as:

$$\frac{d\Omega_2(t)}{dt} = b\int_0^t ve^{-v(t-\tau)}(N(1-e^{-\alpha\tau}) - \Omega_2(\tau))d\tau \tag{7}$$

After solving the Equation (7) by using the Laplace transformation, with initial condition $\Omega_2(t = 0) = 0$, the cumulative number of exploited vulnerabilities by time *t* can be written as:

$$\Omega_2(t) = N\left(1 - \left(1 - \frac{v}{2A}\right)e^{-\frac{v}{2}t} - \left(e^{-\alpha t} - e^{-\frac{v}{2}t}\right.\right.$$
$$\left.\left.\left(cos\,(At) + \frac{1}{A}\left(\frac{v}{2} - \alpha\right)\sin(At)\right)\right)\frac{bv}{\alpha(\alpha - v) + bv}\right) \tag{8}$$

where, $A = \sqrt{bv - \frac{V^2}{4}}$

Hence, the Equation (8) represent the two-stage vulnerability exploitation process by considering the impact of time delay in the vulnerability discovery process and is comparable to the model proposed by Aggarwal et al. [22].

4 Parameter Estimation

To exemplify the accuracy and predictive ability of the proposed model based on the time delay approach, we have used two software vulnerability exploit data sets. The two data sets used in the analysis are of the well-known software platforms, namely Microsoft (DS-I) and Solaris (DS-II), obtained from Exploit Database (https://www.exploit-db.com/). It is important to note that the data sets considered in this paper are of mature software releases and all of these have similar ages (compiled for the period of around 14–16 years). The parameter estimation for the proposed model has been performed in SPSS software based on the non-linear least square method. Furthermore, we have evaluated various goodness-of-fit comparison criteria.

For DS-I, the data set corresponds to vulnerabilities exploited in applications for the Microsoft Windows platform, and the data has been collected from the year 1995 to 2018. For DS-II, the data set corresponds to vulnerabilities exploited in Solaris operating system from the year 1990 to 2018. The parameter estimation and comparison criteria of the proposed model have been calculated and can be viewed respectively through Tables 1 and 2.

From Table 1, it can be clearly noticed that the parameters α and b describing the rate of exploitation is higher than the rate of discovery of a

Table 1 Parameter estimates

	Dataset	
Parameter Estimates	DS-I	DS-II
N	8579.98	229.05
v	0.0984	0.1020
α	0.3656	0.8500
b	0.5818	0.8796

Table 2 Comparison criterion

Comparison Criteria	Dataset	
	DS-I	DS-II
MSE	354980	396
Bias	−119.96	−3.39
Variation	541.90	18.11
RMSPE	555.02	18.43
R^2	0.973	0.935

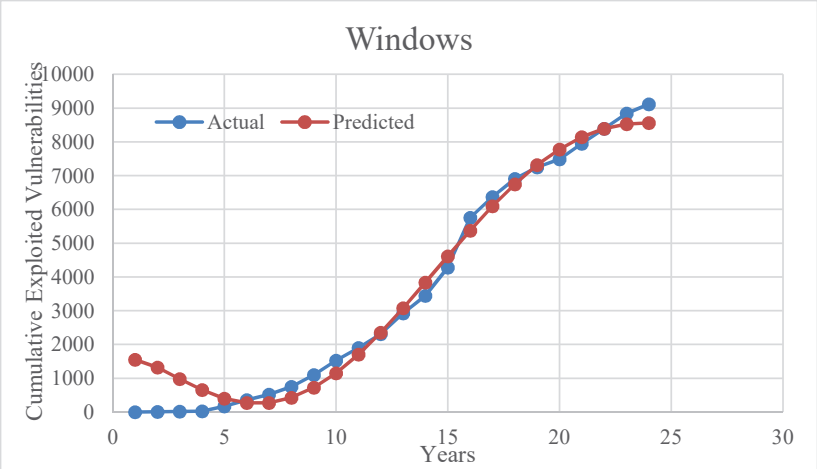

Figure 1 Goodness of fit curve for windows exploit data.

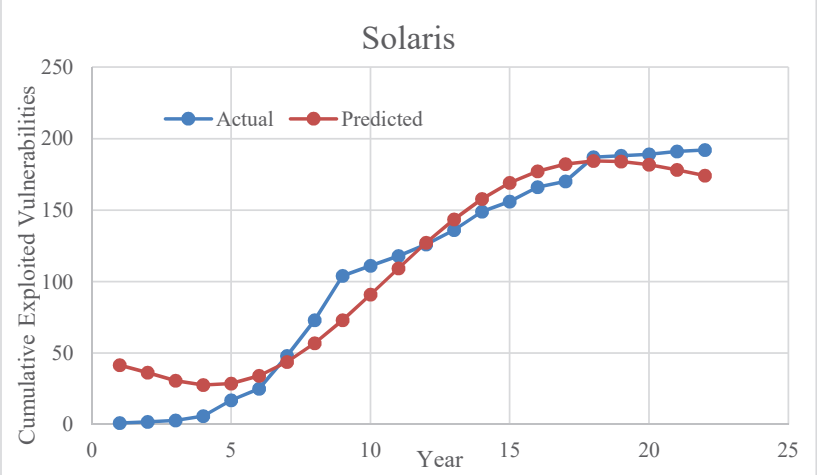

Figure 2 Goodness of fit curve for Solaris exploit data.

vulnerability. It is because discovering the vulnerability might take a significant time. But once the vulnerabilities are discovered the rate of exploitation would be larger as hackers already know about the vulnerabilities. The parameter N represents the total number of potential vulnerabilities to be identified in a software system.

Table 2 provides the comparison criterion calculated to provide the significance of the proposed model. The validation for the proposed model has been considered by computing various criteria, such as Mean Square Error (MSE), Coefficient of Determination (R2), Bias, Variation and Root Mean Square Prediction Error (RMSE) and their respective formulas can be found in [23].

Figures 1 and 2 deals with the cumulative number of vulnerabilities exploited with respect to years for datasets DS-I and DS-II. As can be seen in Figure 1, the predicted values obtained through the model are quite close to the actual dataset. Similarly, in Figure 2, the goodness of fit is quite good on the S-Shaped predicted data. This supports the predictive capability of the model.

5 Conclusion

As software becomes increasingly important in systems that perform complex and critical activities, e.g., stocks, banking, etc., the need for safe and secure software system also increases. In order to achieve better performability, software should avoid the breaches which arise due to the loopholes that are released as the part of the software. Hence, the software developer needs to continuously monitor the exploitation process for the vulnerabilities discovered in order to schedule patches required for their removal. In this paper, a mathematical model has been proposed that relates the exploitation phenomenon as a two-stage process. In the first stage the vulnerabilities are discovered and then after a finite time lag their exploitation might happen. We have considered memory kernel function to join the discovery and exploitation process for a vulnerability. A weak memory kernel has been utilized to denote the real-life aspect of the discovery and exploitation process based on the time of discovery, when the vulnerabilities are discovered the chances of getting exploited also increases if the vendor fails to provide a proper patch. Further, the validity and accuracy have been tested on two different vulnerability exposure data sets.

References

[1] Krsul, I. V. 1998. Software vulnerability analysis. Purdue University, West Lafayette, IN.

[2] Ozment, J. A. 2007. Vulnerability discovery & software security, Doctoral dissertation, University of Cambridge.
[3] Anderson, R. 2002. Security in open versus closed systems—the dance of Boltzmann, Coase and Moore. Technical report, Cambridge University, England.
[4] Rescorla, E. 2005. Is finding security holes a good idea?, *IEEE Security & Privacy*, 3(1), 14–19.
[5] Alhazmi, O.H., Malaiya, Y.K. and Ray, I., 2007. Measuring, analyzing and predicting security vulnerabilities in software systems. *Computers & Security*, *26*(3), 219–228.
[6] Anand, A. and Bhatt, N. 2016. Vulnerability discovery modeling and weighted criteria based ranking. *Journal of the Indian Society for Probability and Statistics*, *17*(1), 1–10.
[7] Anand, A., Das, S., Aggrawal, D. and Klochkov, Y. 2017. Vulnerability discovery modelling for software with multi-versions. In *Advances in reliability and system engineering*. Mangey Ram and J. Paulo Davim, eds. Springer, Cham. pp. 255–265.
[8] Bhatt, N., Anand, A., Yadavalli, V.S.S. and Kumar, V. 2017. Modeling and characterizing software vulnerabilities. *International Journal of Mathematical, Engineering and Management Sciences*, 2(4), 288–299.
[9] Bhatt, N., Anand, A., Aggrawal, D. and Alhazmi, O.H. 2018. Categorization of Vulnerabilities in a Software. *System Reliability Management: Solutions and Technologies,* Adarsh Anand and Mangey Ram, eds. *CRC Press, Boca Raton, FL*, pp. 121–135.
[10] Bhatt, N., Anand, A. and Aggrawal, D. 2019. Improving system reliability by optimal allocation of resources for discovering software vulnerabilities. *International Journal of Quality & Reliability Management*. 37(6/7), 1113–1124.
[11] Anand, A., Bhatt, N. and Alhazmi, O.H., 2020. Modeling Software Vulnerability Discovery Process Inculcating the Impact of Reporters. *Information Systems Frontiers*, doi: 10.1007/s10796-020-10004-9
[12] Liu, Q. and Xing, L., 2021. Survivability and Vulnerability Analysis of Cloud RAID Systems under Disk Faults and Attacks. *International Journal of Mathematical, Engineering and Management Sciences*, 6(1), 15–29.
[13] Anjum, M., Kapur, P.K., Agarwal, V. and Khatri, S.K., 2020. Assessment of software vulnerabilities using best-worst method and two-way analysis. *International Journal of Mathematical, Engineering and Management Sciences*, *5*(2), 328–342.

[14] Bozorgi, M., Saul, L.K., Savage, S. and Voelker, G.M., 2010, July. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. pp. 105–114.

[15] Edkrantz, M. and Said, A., 2015. Predicting cyber vulnerability exploits with machine learning. In *Thirteenth Scandinavian Conference on Artificial Intelligence*, pp. 48–57.

[16] Allodi, L. and Massacci, F., 2014. Comparing vulnerability severity and exploits using case-control studies. *ACM Transactions on Information and System Security (TISSEC)*, *17*(1), 1–20.

[17] Sabottke, C., Suciu, O. and Dumitraş, T., 2015. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*. pp. 1041–1056.

[18] Almukaynizi, M., Nunes, E., Dharaiya, K., Senguttuvan, M., Shakarian, J. and Shakarian, P., 2019. Patch before exploited: An approach to identify targeted software vulnerabilities. In *AI in Cybersecurity.* Leslie F. Sikos, ed. Springer, Cham. pp. 81–113.

[19] Bhatt, N., Anand, A. and Yadavalli, V.S.S., 2020. Exploitability prediction of software vulnerabilities. *Quality and Reliability Engineering International*, 37(2): 648–663. doi: 10.1002/qre.2754

[20] Diamond Jr, A.M., 2005. Measurement, incentives and constraintsin Stigler's economics of science. *The European Journal of the History of Economic Thought*, *12*(4): 635–661.

[21] Cushing, J.M., 1975. An operator equation and bounded solutions of integro-differential systems. *SIAM Journal on Mathematical Analysis*, *6*(3): 433–445.

[22] Aggarwal, R., Singh, O., Anand, A. and Kapur, P.K., 2019. Modeling innovation adoption incorporating time lag between awareness and adoption process. *International Journal of System Assurance Engineering and Management*, *10*(1): 83–90.

[23] Anand, A., Kapur, P. K., Agarwal, M., and Aggrawal, D., 2014. Generalized innovation diffusion modeling & weighted criteria based ranking. In *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization* (pp. 1–6). IEEE.

Biographies

Adarsh Anand did his doctorate in the area of Software Reliability Assessment and Innovation Diffusion Modeling in Marketing. Presently he is working as an Assistant Professor in the Department of Operational Research, University of Delhi (INDIA). He has been conferred with Young Promising Researcher in the field of Technology Management and Software Reliability by Society for Reliability Engineering, Quality and Operations Management (SREQOM) in 2012. He is a lifetime member of the SREQOM. He has publications in journals of national and international repute. His research interest includes software reliability growth modelling, modelling innovation adoption and successive generations in marketing, and social network analysis. He has worked with CRC Press for two editorial projects; "System Reliability Management: Solutions and Technologies" and "Recent Advancements in Software Reliability Assurance". He has also authored one text book with CRC group; "Market Assessment with OR Applications".

Navneet Bhatt received his B.Sc. in Computer Science, M.Sc. in Applied Operational Research and Ph.D. degrees from University of Delhi in 2011, 2013 and 2021, respectively. He is a lifetime member of the Society for Reliability Engineering, Quality and Operations Management (SREQOM).

His current research is focused on Software Vulnerability Discovery Modeling, Software Reliability, Machine Learning and Multi-criteria decision modeling.

Jasmine Kaur is presently pursuing Ph.D. from Department of Operational Research, University of Delhi, Delhi (INDIA). She obtained her B.Sc. (H) Mathematics, M.Sc. in Applied Operational Research, M.Phil. in Operational Research degree in 2013, 2015 and 2017 respectively from University of Delhi, Delhi (INDIA). She joined as a research scholar in the Department of Operational Research in 2015. Her research areas are Software Reliability and Software Security. She has publications in journals of national and international repute.

Yoshinobu Tamura received the BSE, MS, and Ph.D. degrees from Tottori University in 1998, 2000, and 2003, respectively. From 2003 to 2006, he was a Research Assistant at Tottori University of Environmental Studies. From 2006 to 2009, he was a Lecturer and Associate Professor at Faculty of Applied Information Science of Hiroshima Institute of Technology, Hiroshima, Japan. From 2009 to 2017, he was an Associate Professor at

the Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Ube, Japan. From 2017 to 2019, he has been working as a Doctor at the Faculty of Knowledge Engineering, Tokyo City University, Tokyo, Japan. Since 2020, he has been working as a Doctor at the Faculty of Information Technology, Tokyo City University, Tokyo, Japan. His research interests include reliability assessment for open-source software, big data, clouds, reliability. He is a regular member of the Institute of Electronics, the Information and Communication Engineers of Japan, the Operations Research Society of Japan, the Society of Project Management of Japan, the Reliability Engineering Association of Japan, and the IEEE. He has authored the book entitled as OSS Reliability Measurement and Assessment (Springer International Publishing, 2016). Dr. Tamura received the Presentation Award of the Seventh International Conference on Industrial Management in 2004 , the IEEE Reliability Society Japan Chapter Awards in 2007, the Research Leadership Award in Area of Reliability from the ICRITO in 2010, The Best Paper Award of the IEEE International Conference on Industrial Engineering and Engineering Management in 2012, Honorary Professor from Amity University of India in 2017, the Best Paper Award of the 24th ISSAT International Conference on Reliability and Quality in Design in 2018.