Journal of AI and Data Mining
Vol 3, No 1, 2015, 77-83.
doi:10.5829/idosi.JAIDM.2015.03.01.08

Formal approach on modeling and predicting of software system security: Stochastic petri net

H. Motameni

Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

Received 23 November 2014; Accepted 13 January 2015
**Corresponding author: motameni@iausari.ac.ir (H.Motameni)*

Abstract
To evaluate and predict component-based software security, a two-dimensional model of software security is proposed by Stochastic Petri Net in this paper. In this approach, the software security is modeled by graphical presentation ability of Petri nets, and the quantitative prediction is provided by the evaluation capability of Stochastic Petri Net and the computing power of Markov chain. Each vulnerable component is modeled by Stochastic Petri net and two parameters, Successfully Attack Probability (SAP) and Vulnerability Volume of each component to another component. The second parameter, as a second dimension of security evaluation, is a metric that is added to modeling to improve the accuracy of the result of system security prediction. An isomorphic Markov chain is obtained from a corresponding SPN model. The security prediction is calculated based on the probability distribution of the MC in the steady state. To identify and trace back to the critical points of system security, a sensitive analysis method is applied by derivation of the security prediction equation. It provides the possibility to investigate and compare different solutions with the target system in the designing phase.

Keywords: *Software Security, Vulnerability, Stochastic Petri Net, Markov Chain, Sensitivity Analysis.*

1. Introduction

Security has been identified as a major stumbling block in the realization of highly trustworthy software systems [1]. Modeling and predicting software security in design phase provides the possibility of investigation and comparisons of different solutions of target systems. Petri Net is a formal method which is based on mathematical theories. Petri Net is useful for modeling and analysis of systems with parallelization, synchronization and conflict quality [2,3,4].
Stochastic Petri Net is extended from Petri Net where each is associated with a random variable. SPNs combine the powers of Petri Net and Markov chain processes.
In this paper, an advanced approach is suggested to develop the modeling and predicting software security with SPN in design phase. Vulnerability volume of each component to another component is a new parameter that is added to security modeling by SPN.As a result, we improve the accuracy of security in software system prediction. After modeling system security by SPN, The reachable graph is obtained from SPN; The Markov Chain corresponding reachable graph can be extracted and Markov chain calculation is performed. Finally, sensitivity analysis is launched on prediction equation of each component. Sensitivity analysis result can be used to identify the security bottlenecks and trace back to vulnerability points.
This paper is organized as follows: Section 2 discusses the related work. Issues related to stochastic Petri Nets are presented in section 3. Security modeling based on SPN is introduced in section 4. In section 5, an advanced approach is presented to modeling software security with SPN. Sensitivity analysis of software security model is proposed in section 6. A case study is provided in section 7. Section 8 concludes this paper.

2. Related works

Reliability and security analysis has received much attention over the past decades. There have

been some attempts to quantify the security of software system by means of Tiger Team Penetration practices, where a group of experts sit together and try to break in by exploiting any weakness it might possess. However this practice is subjective to the kind of people consisting of the Tiger Team and thus is non-reproducible [5].

There have been some approaches which focus on the process which is adopted while the software is being developed to access the security of final product [5]. One example of this is the SSECMM or Systems Security Engineering Capability Maturity Model. However, branching the software to be secured by evaluating its development process has not found much popularity. This is because even after following the best practices, there is scope of some weakness present in the final product, which would not be uncovered, until it is rigorously tested for its vulnerabilities.

To improve the trustworthiness of software design, formal Threat-Driven approach is represented and explores explicit behaviors of security threats as the mediator between security goals and applications of security features. Security crisis was modeled through Petri net-based aspects [6].

Architecture-based software reliability analysis has been especially investigated by researchers such as surveyed by Gokhale [7]. In that literature, the architecture-based techniques are classified into two path- and state-based categories. For the accuracy and other reasons, state-based approaches are usually adopted [7]. Markov model has been adapted in most previous state-based approaches [1,5,8,9,10].

Some disadvantages are inevitable in using Markov models as modeling tools. First, Markov models lack the abilities to represent parallelism, synchronization, confliction and preemption. Second, they support limited analysis capabilities. Last but not least, a system modeled by a Markov model is hard to extend. The Markov Chain structure changes greatly for even a small change to the system design [1].

In the recent approach, Stochastic Petri Nets have been used for system reliability modeling [11]. It eliminates the difficulty in construction of Markov Chain. Also, Petri nets retain much of the character of the system, such as parallelism, synchronization, confliction and preemption. Furthermore, Petri nets enable us to present system activities in hieratically graphical models so they are recommended to be appropriate state-based models for modeling and quantifying non-functional properties [12].

Sensitivity analysis is provided an approach to investigate influence of changes in different parameters. Gokhail et al. [13] developed an equation to analyze the sensitivity of the reliability. Yang et al. [14] introduced modeling, prediction and sensitivity analysis of a component and Nianhua et al. [1] proposed a combination of components in sequence, parallel, loop and selection style. This paper developed modeling and prediction of software system security with SPN and increased software security prediction.

3. Stochastic petri nets

Petri net is a 5-tuple [15], $PN = (P, T, I, O, M_0)$, where P is a finite set of places and T is a finite set of transitions. $P \cap T = \varnothing$ and $P \cup T = \varnothing$. I is input function where $I = (T * P) = \{0,1\}$, if there is an arc from p to t then $I(t, p) = 1$, so p is an input place for t. O is output function where $O = (T * P) = \{0,1\}$, if there is an arc from t to p then $O(t, p) = 1$, so p is an output place for t. $M_0: P \rightarrow \{1,2, \dots\}$ is initial marking. A transition is enable if each of its input places contains at least one token.

Stochastic Petri net or SPN [16] is a 6-tuple $(P, T, I, O, M_0, \lambda)$ where P, T, I, O and M_0 has the same meaning of a Petri net and λ is set of average firing rate of transitions.

4. Security modeling based on SPN

Suppose that in component based system, each software component contains vulnerability which can be compromised and failure can be repaired by some techniques. Vulnerability is a potential weakness which can be compromised. A component security modeling method based on SPN is proposed in [14]. A software system may contain combination of such component in series, parallel, loop or selection styles. Security modeling and prediction of a system with combination of these styles was proposed in [1].

5. Advanced approach to software security modeling based on SPN

The only parameter of software security modeling and prediction which is proposed in [1] is a successful attack probability of each component whereas there are some other parameters that can be effective in quantitatively prediction of software security. Two components with the same successfully attack probability may have different vulnerability level over whole system. This issue isn't considered in the proposed method by [1].

Vulnerability volume of a component over whole software system is such a parameter which was ignored. This parameter effect is obvious in series

and parallel styles of components. Vulnerability measure of one component depends on the type of software system. Software security prediction equations have to be rewritten by adding this variable. In this case, we will add Vulnerability volume of a component toward others components, namely, it must be investigated how much each component influences in the security of whole software system. To calculate system tolerance, successfully attack probability of a component must multiply by the ratio of its efficiency in the system security.

5.1. SPN model of a component

Probability density functions for normal execution, attack and repair action in a component are shown by λ_{i1} , λ_{i2} and λ_{i3} respectively. Figure 1, referenced from [14], demonstrates a model of a component represented by SPN where t_i^e represents the normal behavior of the component with execution rate of λ_{i1}. t_i^e Represents an attack on the component I. Its rate is λ_{i2}. P_i^b is start place. A token appearing in the place t_i^f denotes that the component i has been compromised so a recovery action should be taken, such as rebooting. The transition t_i^r represents the recovery action with the rate of λ_{i3}. t_i^s indicates successfully execution of component.

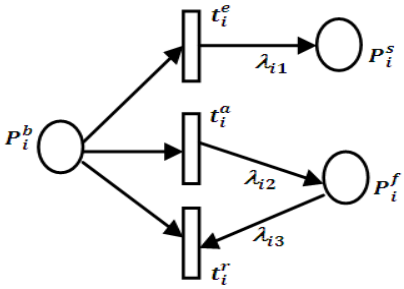

Figure 1. Security component model based on SPN.

5.2. Sequence components model based on SPN

In sequence model, components are executed in sequential manner. Only a single component is executed at instant of time. Figure 2 shows two components in sequence manner.

The probability of successful attack in a sequence model composed of n components is in (1):

$$\prod_{i=1}^{n}[(\mu_i) * (SAP_i)] \quad (1)$$

Where SAP_i is the successful attack probability of component i and μ_i is a new parameter, the vulnerability volume of component i over whole system, that is addition parameter to modeling.

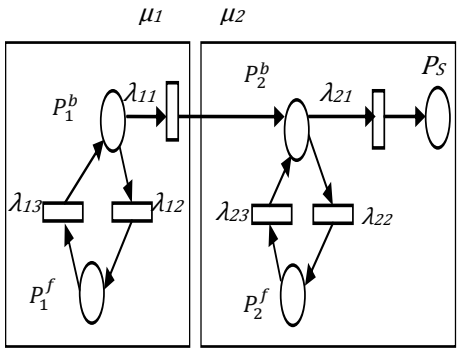

Figure 2. Sequence components model based on SPN.

The probability of successful execution without compromise in a sequence model composed of n components is in (2):

$$\prod_{i=1}^{n}[(1-\mu_i) * (1-SAP_i)] \quad (2)$$

5.3. Parallel components model based on SPN

A parallel model is usually used in a concurrent execution environment to improve performance. An example of this model is depicted in figure 3.

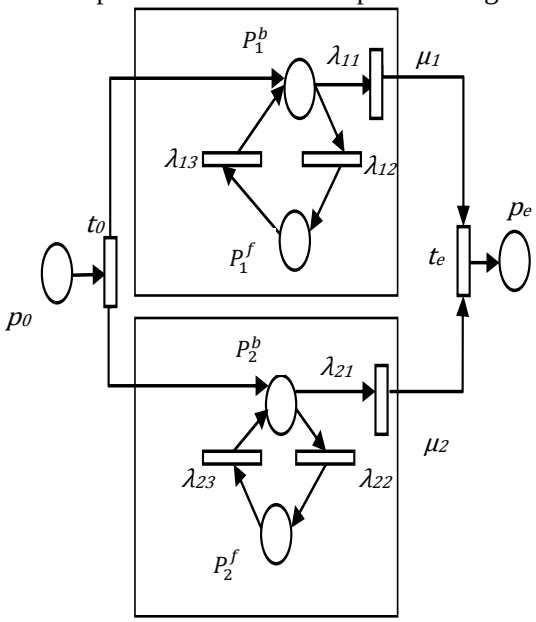

Figure 3. Parallel components model based on SPN.

The probability of successful attack in a parallel model composed of n components is in (3):

$$max_{i=1}^{n}[(\mu_i) * (SAP_i)] \quad (3)$$

The probability of successful execution without compromise in a parallel model composed of n components is in (4):

$$1 - max_{i=1}^{n}[(\mu_i) * (SAP_i)] \quad (4)$$

5.4. Loop component model based on SPN

A loop model is used in an iterative execution environment, in which a component is executed iteratively for some times. Figure 4 indicates an example of this model. The transition t_{loop} in figure 4 activates the iterated component.

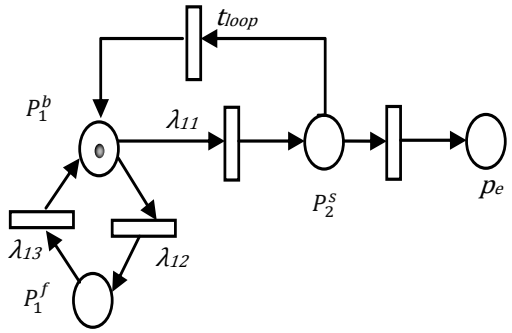

Figure 4. Loop component model based on SPN.

The probability of successful attack in a loop model is in (5):

$$\prod_{i=1}^{n}[(\mu) * (SAP)] \quad (5)$$

The probability of successful execution without compromise in a loop model is in (6):

$$\prod_{i=1}^{n}[(1-\mu) * (1-SAP)] \quad (6)$$

5.5. Selection component model based on SPN

In a selection model, components are executed with conflict. Only one component can be executed according to the selection condition. The probability of the system successfully compromised or executing in a selection model is equal to the selected component. If component i is selected, the probability of successful attack to system is calculated by (7):

$$[(\mu_i) * (SAP_i)] \quad (7)$$

The probability of successful execution without compromise in selection model is in (8):

$$(1-\mu_i) * (1-SAP_i) \quad (8)$$

5.6. Software security prediction evaluation

In [1, 14], an approach was presented for successfully attack probability by intruder to software system is security metric in steady state. SAP is computed by adding probability of system states that contain one token. The higher the SAP, the greater the probability the software system can be promised.

Quantifying the SAP on a component consists of following three steps.

- Construct an isomorphic MC from the SPN model;
- Evaluate the SPN steady state probability distribution based on the MC;
- Evaluate the SAP based on the steady state probability distribution of the SPN model.

Due toless memory regarding the exponential distribution of firing delays, SPN models are isomorphic to Continuous Time Markov Chains [1]. The method in [14] is used to evaluate the steady state probability distribution of reachable states. The method of evaluating compromised probability for a single component has appeared in [14]. A failure place in an SPN model is represented as p_{fr}, r = 1, 2, …, k. Thus, the SAP can be evaluated as (9):

$$SAP = \sum_{j=1}^{n}\sum_{r=1}^{k} P[M_j(p_{fr}) \geq 1] \quad (9)$$

$P[M_j(p_{fr}) \geq 1]$ indicates places of probability p_{fr} that contain at least one token in steady state. Thus, tolerance capacity of a component toward attack is represented in (10).

$$TP = 1 - SAP = 1 - \sum_{j=1}^{n}\sum_{r=1}^{k} P[M_j(p_{fr}) \geq 1] \quad (10)$$

So we can compute the security of hierarchal software system.

6. Sensitivity analysis

Sensitivity analysis is useful for software optimization in the early design phase [8]. It is difficult to study some model parameters in design phase. Sensitivity analysis can investigate change effects in parameters over quantitative analysis results. Successfully attack probability is computed by derivation over these variables [1] in (11).

$$\frac{d(SAP(\lambda_1, \dots, \lambda_{|T_t|})}{d\lambda_i} = \frac{d\sum_{j=1}^{n}\sum_{r=1}^{k} P[M_j(p_{fr}) \geq 1]}{d\lambda_i} \quad (11)$$

Equation (11) is a sensitivity analysis of security prediction for one component. According to the new parameter that is added to modeling, sensitivity analysis can be computed for new parameter, as follow:

$$\frac{d(SAP)}{d\mu} = \frac{d\sum_{j=1}^{n}\sum_{r=1}^{k} P[M_j(p_{fr}) \geq 1]}{d\mu} \quad (12)$$

7. Case study

To evaluate the new approach, first the security modeling and prediction evaluation of a single component is illustrated, and then the evaluation for a software system including different components in different styles and in different levels of hierarchical can be calculated based on the result of each single component.

7.1. A single component modeling

Figure 5 shows a single software security critical component based on SPN. The transition t_2 represents an intrusion to component. The resume action is shown by transitiont_3.

Existence of a token in place P_2 represents compromised state caused by an intrusion. Transition t_1 shows a successful execution of the component. To evaluate the prediction values using MC techniques, transition t_4 is added.

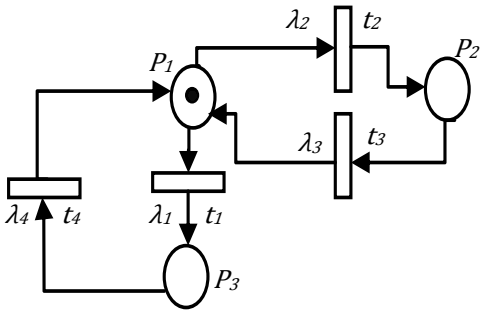

Figure 5. SPN model for evaluating security of a component.

7. 2. Extracting reachable graph

The reachable markings, shown in table 1, are obtained from figure 5.

Table 1. Reachable marking obtained figure 5.

Marking	P_1	P_2	P_3
M_1	1	0	0
M_2	0	1	0
M_3	0	0	1

Reachable graph is specified by reachable marking and isomorphism SPN model. Isomorphic Markov chain with SPN model in figure 7 is equivalent with reachable graph of figure 6.

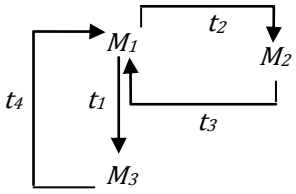

Figure 6. Reachable graph for SPN model.

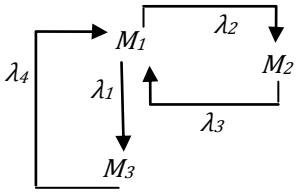

Figure 7. Markov chain isomorphic to SPN model.

7.3. Evaluating security prediction

Matrix Q regarding to Markov chain is as (13):

$$Q = \begin{matrix} M_1 \\ M_2 \\ M_3 \end{matrix} \begin{bmatrix} -(\lambda_1+\lambda_2) & \lambda_2 & \lambda_1 \\ \lambda_3 & -\lambda_3 & 0 \\ \lambda_4 & 0 & -\lambda_4 \end{bmatrix} \quad (13)$$

Suppose that $Y = (P(M_1), P(M_2), P(M_3))$. Thus we can get (14):

$$\begin{cases} YQ = 0 \\ P(M_1) + P(M_2) + P(M_3) = 1 \end{cases} \quad (14)$$

The calculated result for the probability distribution at steady state is shown (15):

$$\begin{cases} P(M_1) = \dfrac{\lambda_3\lambda_4}{\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3} \\ P(M_2) = \dfrac{\lambda_2\lambda_4}{\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3} \\ P(M_3) = \dfrac{\lambda_1\lambda_3}{\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3} \end{cases} \quad (15)$$

A token in M_2 indicates that the software component is compromised by an intrusion. According to (9), we can get (16):

$$SAP = P(M_2) = \frac{\lambda_2\lambda_4}{\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3} \quad (16)$$

By adding vulnerability volume of a component, over whole software system, namely μ , we can rewrite (16) as follow in (17):

$$\mu \times SAP = \frac{\mu\lambda_2\lambda_4}{\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3} \quad (17)$$

7.4. Sensitivity analysis

Sensitivity analysis of this component is calculated by derivation of (17)

$$\left|\frac{d(\mu \times SAP)}{\mu}\right| = \frac{\lambda_2\lambda_4}{\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3} \quad (18)$$

Because in a software system with a single component, vulnerability volume over whole system, μ, is equal to 1 so onlythe impact of changes of t_1, t_2 and $t_3$to SAP are considered. We have the followings:

$$\left|\frac{d(\mu \times SAP)}{d\lambda_1}\right| = \frac{\mu\lambda_2\lambda_3\lambda_4}{(\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3)^2} \quad (19)$$

$$\left|\frac{d(\mu \times SAP)}{d\lambda_2}\right| = \frac{\mu(\lambda_3\lambda_4{}^2 + \lambda_1\lambda_3\lambda_4)}{(\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3)^2} \quad (20)$$

$$\left|\frac{d(\mu \times SAP)}{d\lambda_3}\right| = \frac{\mu\lambda_2\lambda_4(\lambda_1 + \lambda_4)}{(\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3)^2} \quad (21)$$

$$\left|\frac{d(\mu \times SAP)}{d\mu}\right| = \frac{\lambda_2\lambda_4(\lambda_1 + \lambda_4)}{(\lambda_3\lambda_4 + \lambda_2\lambda_4 + \lambda_1\lambda_3)^2} \quad (22)$$

When the value of λ_i , $i = 1, 2, 3, 4$, is assigned, the sensitivity caused by them can be calculated by (19) – (22). The transition t_4 is used for the facility of the steady state computation. The execution time is very short. So the value for λ_4 is very large. Suppose that λ_4 equals to 1,000,000. Let $\lambda_3 = 6$, $10 \le \lambda_1 \le 30$ and $1 \le \lambda_2 \le 10$. Figure 8 shows the probability distribution of SAP for different normal execution and attack rates. It shows that the probability of the

component being in the compromised state increases with an increasing attack efficiency in the steady state.

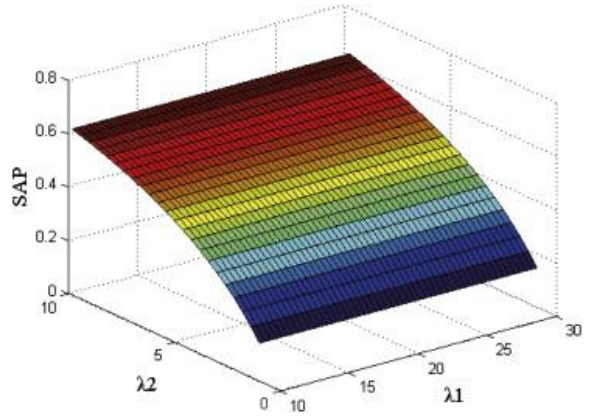

Figure 8. Relationship between normal execution rate λ_1, attack rateλ_2 and SAP.

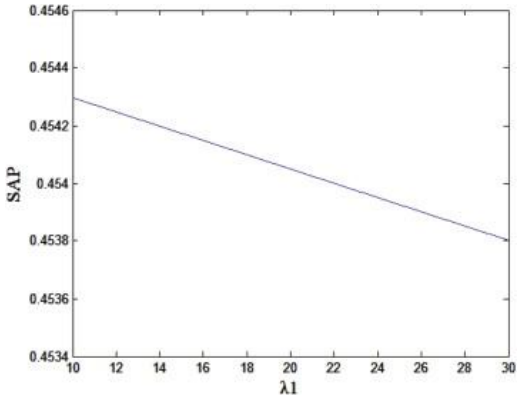

Figure 9. Relationship between normal execution rate $\lambda_1$and SAP.

Suppose that $\lambda_1 = 15$. Let $0 \leq \lambda_2 \leq 10$ and $0 \leq \lambda_3 \leq 15$. Figure 9 shows that the probability of the component being in the compromised state in steady state decreases with an increased resume in rate λ_3. It increases rapidly with increasing the attack rate λ_2.

Although accuracy improvement by advanced modeling and predicting software security is obvious with a new parameter; however, it is difficult to quantitatively express the improvement of a new method, but as it was mentioned in new approach, software system security is evaluated from new dimensioned that was ignored in recent approach. These two new approaches are compared in table 2.

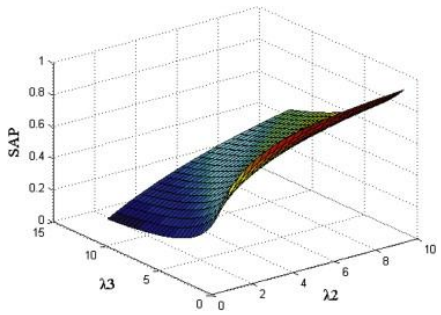

Figure 10. Relationship between the attack rate λ_2, resume rate λ_3 and SAP.

8. Conclusion

This paper proposes the two-dimensional method to model and predict software security based on stochastic Petri nets. The main contributions of the paper can be summarized as follows:

•An advanced method for security of software system based on Stochastic Petri net with added metric is proposed. A software system is modeled in view of the new metric, parallelization, synchronization and confliction characteristics of a component-based system can be easily modeled by stochastic Petri nets, while Markov Chains are absent of the abilities to represent these characteristics.

•Vulnerability volume of a component is added as a new parameter of system, and security prediction equations are rewritten. Thus, adding a new dimension of security in software system increases the accuracy of software security evaluation.

•A sensitivity analysis method is applied which provides a mean to identify and trace back to the critical components for security enhancement. It also provides the probability to investigate and compare different solutions to the target system before realization. We will work on the following open issues in the future:

- Modeling and predicting software system security based on stochastic Petri net by just vulnerability measure as a parameter.
- Advanced modeling and prediction of software system security with UML.
- Implementing the system by Petri net tools and Markov chain simulation to evaluate the security of software system.

Table 2. Advanced modeling and prediction with SPN vs. modeling and prediction suggested by [1].

Evaluating software system security approach	Stochastic Petri net	Advanced security modeling
Number of parameters for security modeling	1	2
Number of parameters for sensitivity analysis	1	2
Parameter/s	Successfully attack probability	Successfully attack probability, vulnerability measure
Accuracy measure	High	Higher

References

[1] Yang, N., Yu, H., Qian, Z., & Sun, H. (2012). Modeling and quantitatively predicting software security based on stochastic Petri nets. Mathematical and Computer Modelling, vol. 55, no. 1, pp. 102–112.

[2] Yang, N., Yu, H., Sun, H., & Qian, Z. (2010). Modeling UML Sequence Diagrams Using Extended Petri Nets. International Conference on Information Science and Applications. ICISA2010. IEEE Computer Society, pp. 596–603.

[3] Murata, T. (1989). Petri nets: Properties, analysis and applications. Proceedings of the IEEE, vol. 77, no. 4, pp. 541–580.

[4] Dempsey, J. R., Davis, G. W. A., Crossfield, A. S., & Williams, W. C. (1964). Program Management in Design and Development. SAE Technical Paper.

[5] De Miguel, M. A., Briones, J. F., Silva, J. P., & Alonso, A. (2008). Integration of safety analysis in model-driven software development. Software, IET, vol. 2, no. 3, pp. 260-280.

[6] Xu, D., & Nygard, K. E. (2006). Threat-driven modeling and verification of secure software using aspect-oriented Petri nets. Software Engineering, IEEE Transactions on, vol. 77, no. 4, pp. 265–278.

[7] Bode, E., Herbstritt, M., Hermanns, H., Johr, S., Peikenkamp, T., Pulungan, R., & Becker, B. (2009). Compositional dependability evaluation for STATEMATE. Software Engineering, IEEE Transactions on, vol. 35, no. 2, pp. 274–292.

[8] Sharma, V. S., & Trivedi, K. S. (2007). Quantifying software performance, reliability and security: An architecture-based approach. Journal of Systems and Software, vol. 80, no. 4, pp. 493–509.

[9] Wang, W.-L., Pan, D., & Chen, M.-H. (2006). Architecture-based software reliability modeling. Journal of Systems and Software, vol. 79, no. 1, pp. 132–146.

[10] Madan, B. B., Goševa-Popstojanova, K., Vaidyanathan, K., & Trivedi, K. S. (2004). A method for modeling and quantifying the security attributes of intrusion tolerant systems. Performance Evaluation, vol. 56, no. 1, pp. 167–186.

[11] Ammar, H. H., Huang, Y., & Liu, R. (1987). Hierarchical models for systems reliability, maintainability, and availability. Circuits and Systems, IEEE Transactions on, vol. 34, no. 6, pp. 629–638.

[12] Gokhale, S. S. (2007). Architecture-based software reliability analysis: Overview and limitations. IEEE Transactions on Dependable and Secure Computing, no.1, pp. 32–40.

[13] Gokhale, S. S., & Trivedi, K. S. (2002). Reliability prediction and sensitivity analysis based on software architecture. In Software Reliability Engineering, 2002. ISSRE 2003. Proceedings. 13th International Symposium on (pp. 64–75). IEEE.

[14] Yang, N., Yu, H., Sun, H., & Qian, Z. (2010). Quantifying software security based on stochastic Petri nets. Journal of Computational Information Systems, vol. 6, no. 9, pp. 3049–3056.

[15] Balbo, G. (2001). Introduction to stochastic Petri nets. In Lectures on Formal Methods and PerformanceAnalysis (pp. 84–155). Springer.

[16] Florin, G., Fraize, C., & Natkin, S. (1991). Stochastic Petri nets: Properties, applications and tools. Microelectronics Reliability, vol. 31, no. 4, pp. 669–697.

نشریه هوش مصنوعی و داده کاوی

دانشگاه شاهرود

روش رسمی مدل‌سازی و پیش بینی امنیت سیستم نرم افزاری: شبکه‌ی پتری تصادفی

همایون موتمنی

دانشگاه آزاد اسلامی واحد ساری، گروه مهندسی کامپیوتر، ساری، ایران.

ارسال ۲۰۱۴/۱۱/۲۳؛ پذیرش ۲۰۱۵/۰۱/۱۳

چکیده:

در این مقاله برای ارزیابی و پیش بینی امنیت سیستم نرم افزاری مولفه‌گرا، مدل دوبعدی امنیت نرم افزار با شبکه‌ی پتری تصادفی ارائه شده است. در این روش، امنیت نرم افزار با ویژگی نمایش گرافیکی شبکه‌ی پتری مدل می‌شود و پیش بینی کمی امنیت با قابلیت ارزیابی شبکه‌ی پتری تصادفی و قدرت محاسباتی زنجیر مارکوف انجام می‌شود. هر مولفه‌ی آسیب‌پذیر با شبکه‌ی پتری تصادفی و با دو پارامتر احتمال حمله‌ی موفق و میزان آسیب هر مولفه نسبت به مولفه‌های دیگر مدل می‌شود. برای افزایش دقت پیش بینی امنیت نرم افزار، پارامتر دوم به‌عنوان دومین بعد ارزیابی نرم افزار به فرآیند مدل‌سازی افزوده شده است. سپس زنجیر مارکوف معادل مدل SPN بدست می‌آید و پیش بینی میزان امنیت بر مبنای توزیع احتمال مدل مارکوف (MC) در حالت پایدار محاسبه می‌شود. برای شناسایی و بازگشت به نقاط بحرانی سیستم نرم افزاری، تحلیل حساسیت، با مشتق‌گیری از معادله‌ی پیش بینی امنیت، انجام می‌شود. پیش بینی امنیت و تحلیل حساسیت در فاز طراحی امکان بررسی و مقایسه راه‌حل های مختلف برای رسیدن به سیستم هدف قبل از تحقق آن را فراهم می‌کند.

کلمات کلیدی: امنیت نرم افزار، آسیب‌پذیری، شبکه‌ی پتری تصادفی، زنجیره مارکوف، تحلیل حساسیت.