Journal of mathematics and computer science 15 (2015), 70-77

Contents list available at JMCS
Journal of Mathematics and Computer Science
Journal Homepage: www.tjmcs.com

Software Security Modeling Based On Petri Nets

A. Mohsenzadeh[1]

[1]Department of Information Technology, Mazandaran University of Science and Technology, Babol, Iran
a.mohsenzadeh@ustmb.ac.ir

Article history:
Received May 2014
Accepted January 2015
Available online February 2015

Abstract

Nowadays, mostly security solutions are mainly focused on how to defend against various threats, including insider threats and outsider threats, instead of trying to solve security issues from their sources. This paper proposes a security modeling process and an approach to modeling and quantifying component security based on Petri Nets (PN) in the software design phase. Security prediction in the design phase provides the possibility to investigate and compare different solutions to the target system before realization. The analysis results can be used to trace back to the critical part for security enhancing.

Keywords: Software security, Petri net, Security models

1. Introduction

Security has become an important topic for many software systems. Currently, related reports of security failures are becoming very common. According to the results of the Software Engineering Institute's CERT Coordination Center's survey, it shows that the number of reported application vulnerabilities rose from 171 in 1995 to 5,990 in 2005 [1]. When the system designers are constructing the blueprint of an organization, they often neglect that the system design must be considered from the overall view and security technologies cannot be incorporated at random. Bruce Schneier also states that "Security is a chain; it's only as secure as the weakest link. Security is process, not a product [2]."

Security has been identified as a major stumbling block in the realization of highly trustworthy software systems. The cost is much higher to repair the flaws found in the late phase of development than those found in the early phase. To reduce development cost and effort, attempts to improve software security should be done as early as possible. Software security modeling in the software design phase provides the possibility to investigate and compare different solutions to the target system before realization. Sensitivity analysis on parameters in the model enables the identification of security bottlenecks.

Petri nets [3] are powerful formal models. They are based on strict mathematical theories. Petri nets are appropriate for modeling and analyzing systems with parallelization, synchronization and confliction [4]. Many verification and analysis methods have been developed around them and many mature analysis tools are available [5,6]. They provide convenience for qualitative and quantitative analysis in the software design phase. A system modeled with Petri nets is easily extended. They also provide visual and hierarchical modeling methodologies. Therefore, in this paper, each software component is modeled by an Petri net (PN).

The rest of this paper is organized as follows. Section 2 introduces security modeling process. Issues related to Petri nets are presented in Section 3. Section 4 proposes a hierarchical software security modeling method based on PN. A case study is provided in Section 5. Section 6 concludes this paper.

2. Security Modeling Process

Current mostly security professionals would rather focus on how to defend against various threats than overcome the causes of security issues in the information system. There are several existing works on security requirements engineering [7,8,9,10]. Based on these researches we present a security model - in 12 steps - based on security requirements engineering process. One view of security modeling is given in Figure.1

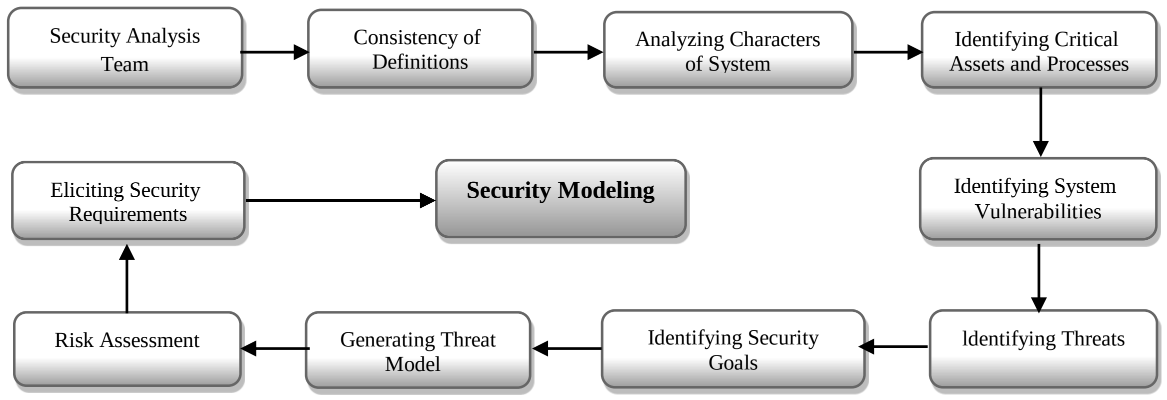

Security Analysis Team
Consistency of Definitions
Analyzing Characters of System
Identifying Critical Assets and Processes
Eliciting Security Requirements
Security Modeling
Identifying System Vulnerabilities
Risk Assessment
Generating Threat Model
Identifying Security Goals
ldentifying Threats

Figure1. Security modeling process

- Security Analysis Team: In general, security requirements team may be constitute of security personnel, technical personnel and non-technical personnel.
- Consistency of Definitions: First of all, before security requirements analysis of an organization will be done, all members must be agree on all related terms and definitions during the process of security analysis. The best approach is to develop a common dictionary which can help all members have a consistency of views.

- Analyzing Characters of System: security requirements team need to understand the system completely. This means they must understand every component and its interconnections, and consider how to define usage scenarios and identify assumptions and dependencies.
- Identifying Critical Assets and Processes: Identifying critical assets and processes is an important step in the security requirements engineering process. Assets and processes are abstract or concrete resources which a system must protect from insider threat. These identified critical assets and processes will help analysts develop the following security requirements steps. On the one hand, they will help the security team respectively prioritize the analysis of vulnerabilities, threats, and risks according to the critical degree of assets and processes. On the other hand, they will be helpful to further discuss critical issues against consuming human resource, time, and energy.
- Identifying System Vulnerabilities: lots of vulnerabilities can just be known after security accidents have happened. The analysis of system vulnerabilities mainly focuses on system hardware, software, artificial factor, policy, procedure, and so on. In the security requirements engineering process, iterative analysis of vulnerabilities will be an essential component for ensuring complete security of an enterprise system. The representation of vulnerability analysis will be different when a system lies in different phases. It can be divided into three phases: designing phase of system, realizing phase of system, running phase of system.
- Identifying Threats: The goal of this step is to use these information collected by previous steps and identify threats of a system. The best way of enumerating threats is to view each critical asset and critical process as a root node, then to analyze all potential vulnerabilities concerning the asset/process. Finally, system threats will be thoroughly identified by traversing all vulnerabilities according to confidentiality, confidentiality, integrity of information.
- Identifying Security Goals: Based on the list of threats identified by the previous step, security goals can use them to prevent or avoid the actions on the asset that realizes the threat. The aim of this step is to require administers of the organization and the security requirements team that must come to an agreement on a set of prioritized security goals. Moreover, the existing security goals will confine the scope of the rest of the security requirements engineering process.
- Generating Threat Model: In order to protect each critical asset and process in system from insider abuse or insider attack, it is very necessary to combine with security goals and implement some structure graphs and models during the process of security requirements analysis, such as attack tree model, and misuse case diagram.
- Risk Assessment: The purpose of this step is to determine the likelihood that the threats will materialize as real attacks and assess theirs impact and risk. Many methods of risk assessment can be used in this step, and the methods can be divided into three types: 1) qualitative assessment; 2) quantitative assessment; 3) mixed assessment, namely a combination of qualitative and quantitative assessment.
- Eliciting Security Requirements: This step is the core of the security modeling and will derives final security requirements. Correct security requirements or security requirements set can be attained by detailedly analyzing each security goal, seriously considering related threats, overall executing risk assessment, and reducing risks to an acceptable step.
- Security Modeling: The purpose of this step is providing security model Based on the previous steps. Hence, in Section 4, we present a security model based on these 12 steps.

3. Petri nets

A Petri net is a 5-tuple [3], PN = (P, T, F, W, M0). P is a finite set of places (drawn as circles). T is a finite set of transitions (drawn as rectangles). F is a set of arcs. An arc connects a transition to a place or a place to a transition. W: F → {1, 2, . . .} is a set of weight functions. M0: P → {1, 2, . . .} is the initial marking. $P \cap T = \varphi$ and $P \cup T \not\neq \varphi$. A transition is enabled if and only if each of its input places contains at least one token. The firing of a transition removes one token from each input place and places one token in each output place.

4. PN Based Security Modeling

Suppose that every software component contains vulnerabilities which can be compromised. The failed component can be repaired using some techniques. A software system consists of several such components hierarchically in sequence, parallel, loop styles.

4.1. Security Modeling

Figure 2 demonstrates a simple security evaluation model of a component represented by Petri net. For description convenience, c_i is used to represent the model. The transition t_s represents the normal behavior of the component. Each token appearing in the place p_s , called the safe place, indicates that the component has been executed successfully. An attack on the component by an intruder is represented by the transition t_u. Each token appearing in the place p_u, called the unsafe place, denotes that the component ci has been compromised. After being compromised, a recovery action should be taken, such as rebooting. The transition t_r represents the recovery action after being compromised. Each token appearing in the place p_r, called the recovery place. The places p_i and p_o , are input and output places of the component respectively. As well, place and transition descriptions are in Tables 1 and 2

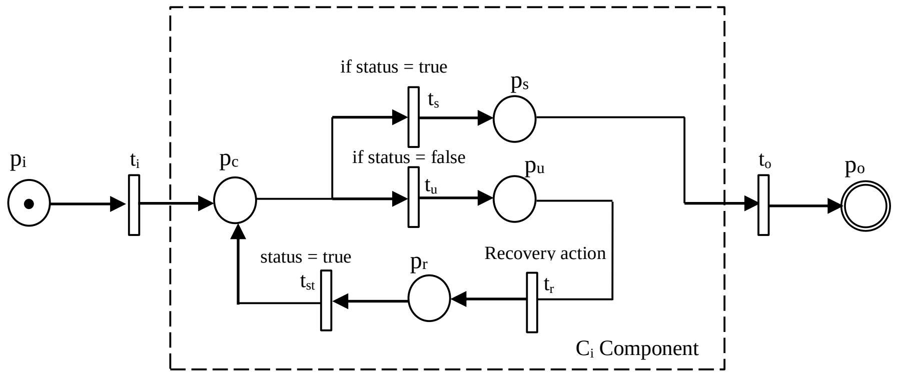


Figure2. Security evaluation model for component c_i

Table1. PNs places for security evaluation model c_i

Place	Place Name	Description
p_i	p_{input}	Initial state
p_o	p_{output}	Final state
p_c	p_{check}	Check the component status
p_s	p_{safe}	Success state
p_u	p_{unsafe}	Failure state
p_r	$p_{recovery}$	Repair state

Table2. PNs transitions for security evaluation model c_i

Transition	Transitions Name	Description
t_i	t_{input}	To evaluate component security , t_i will fire
t_o	t_{output}	After completing the security survey of component c_i, this transition will fire
t_s	t_{safe}	if the execution of the component is normal this transition will fire
t_u	t_{unsafe}	if the component has been attacked then t_u will fire
t_r	$t_{recovery}$	To recover the damaged component, this transition will fire
t_{st}	t_{status}	To survey component status, t_i will fire

4.2. Loop model

A loop model is used in an iterative execution environment, in which a component is executed iteratively for some times. An example of this model is depicted in Figure3. The number of iterations determined in transition t_i . The transition t_l in activates the iterated component. The transition t_o causes the loop to stop. Suppose that the probability of a successful intrusion by an intruder in the iterative component is β. The probability of successful execution in a loop model, with n times iteration, is $\prod_{i=1}^{n}(1 - \beta)$.

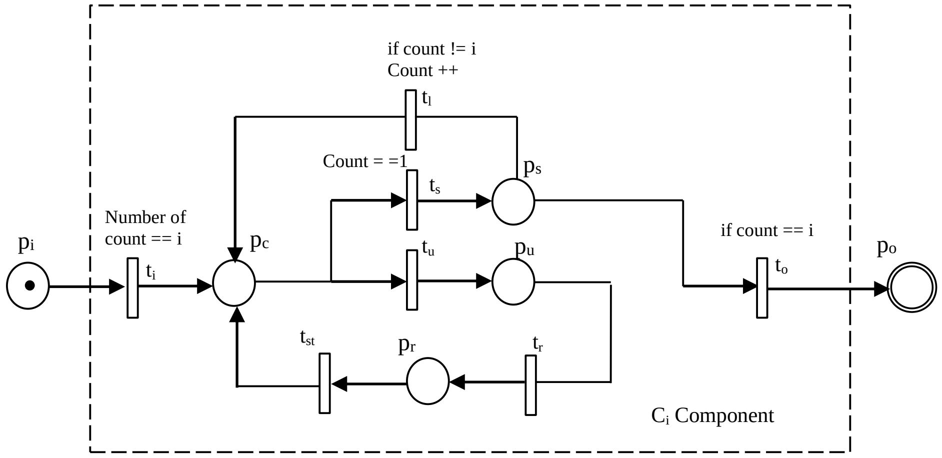


Figure3. A Loop security evaluation model

4.3. Sequence model

In a sequence model, components are executed in a sequential manner. Only a single component is executed at any instant of time. The control is transferred to its successor upon the completion of a component. Figure 4 shows an example of components composed in sequence style. Suppose that the probability of successful intrusion by an intruder in a component i is β_i. The probability of successful execution without compromise in a sequence model composed of n components is $\prod_{i=1}^{n}(1 - \beta_i)$.

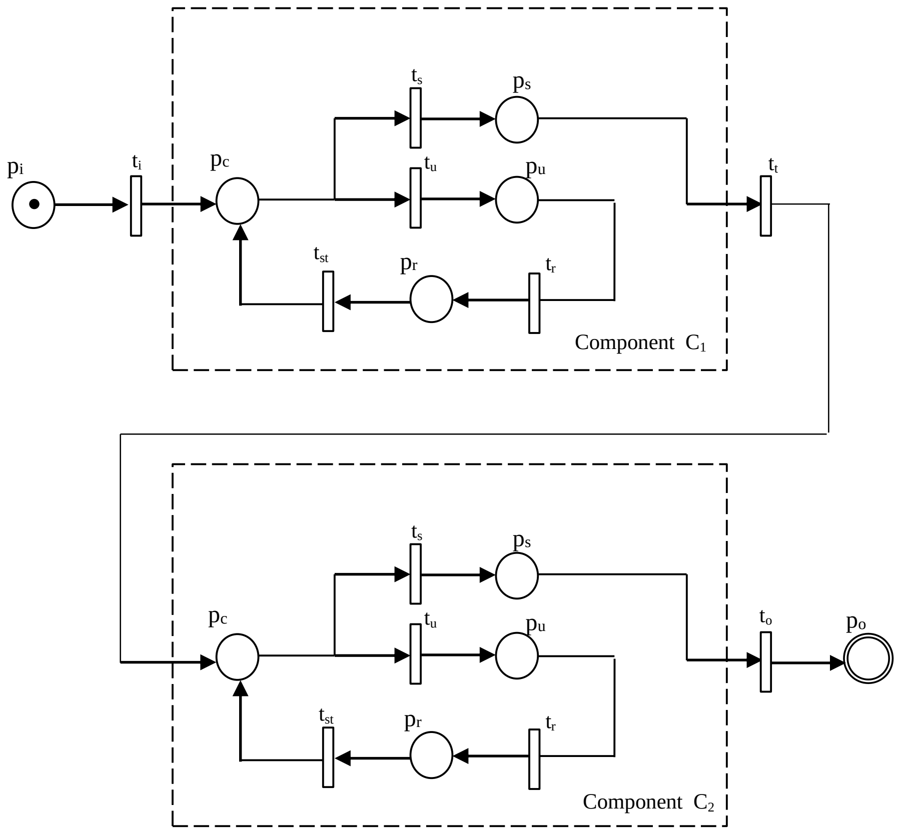


Figure4. A sequence security evaluation model

4.4. Parallel model

A parallel model is usually used in a concurrent execution environment, in which a set of components are executed concurrently to improve performance. An example of this model is depicted in Figure 5. Suppose that the probability of successful intrusion by an intruder in a component i is βi. So the probability of successful intrusion of a composed system, consisting of n components, is $Max_{i=1}^{n}$(βi). The probability of successful execution in a parallel model composed by n components is 1-$Max_{i=1}^{n}$(βi).

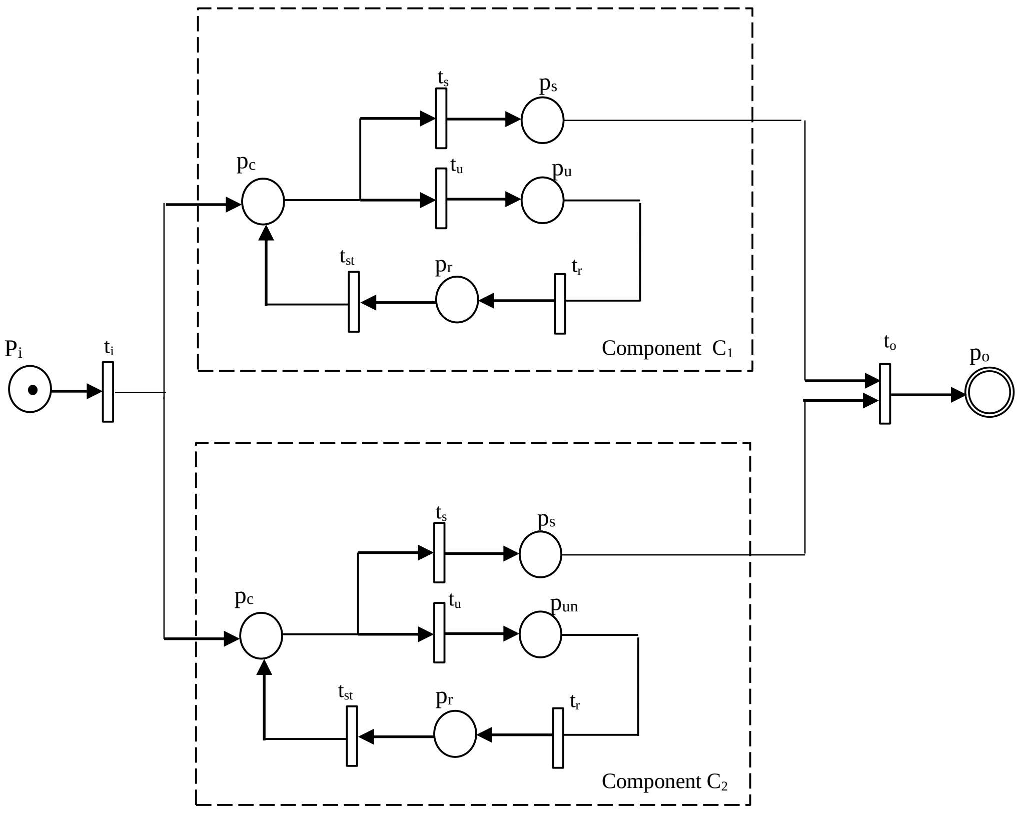
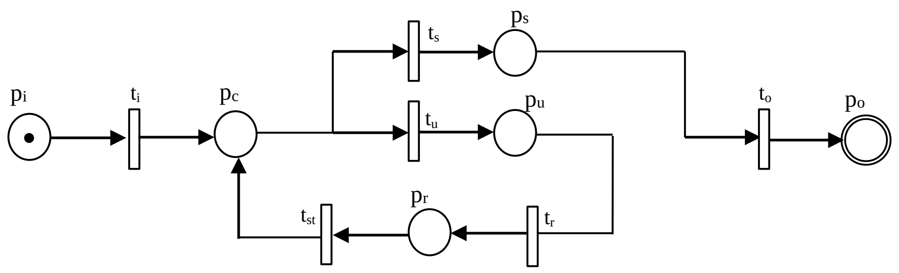


Figure5. A parallel security evaluation model

5. Case study

This section presents a case study to show the applicability and feasibility of our method. The study for security modeling evaluation of a single component is illustrated. Suppose that there is a software system including a critical component. Also, a recovery mechanism is used to resume it when it is compromised. Figure 6 shows a simple security critical software component modeled by PN. The transition t_u represents an intrusion action. The resume action is depicted by t_r. Occurrence of tokens in the place p_u represents a compromised state caused by an intrusion. The transition t_s represents a successful execution of the component. The reachable markings, shown in Table3, are obtained from Figure5.



Marking	p_i	p_o	p_c	p_u	p_r	p_s
M_1	1	0	0	0	0	0
M_2	0	1	0	0	0	0
M_3	0	0	1	0	0	0
M_4	0	0	0	1	0	0
M_5	0	0	0	0	1	0
M_6	0	0	0	0	0	1

6. Conclusion

We have proposed a method to model and evaluate a software component security based on Petri nets. In fact, a hierarchical modeling method for a software system with vulnerabilities and recovery mechanisms based on Petri nets is proposed. A complicated software system is modeled hierarchically according to the composing styles based on different components. Parallelization, synchronization and confliction characteristics can be easily modeled with Petri nets. We will work on the following open issues in the future. 1) A security Unified Modeling Language (UML) profile will be proposed for quantitatively representing security properties in design models and 2) The methods for translating UML models, annotated with quantitative security information, into PNs will be developed.

References

[1] "*CERT/CC Statistics 1988-2005*", Pittsburgh, CERT CC,http://www.cert.org/stats/cerCstats.html, Feb. (2006).

[2] B. Schneier. "*Secrets & Lies*". John Wiley & Sons, Inc.,(2000).

[3] T. Murata, "*Petri nets: properties, analysis and applications*", Proceedings of the IEEE 77 (4) (1989) 541–580.

[4] N. Yang, H. Yu, H. Sun, Z. Qian, "*Modeling UML sequence diagrams using extended Petri nets*", in: International Conference on Information Science and Applications, ICISA2010, IEEE Computer Society, (2010), pp. 596–603.

[5] A.V. Ratzer, L. Wells, H.M. Lassen, M. Laursen, J.F. Qvortrup, M.S. Stissing, M. Westergaard, S. Christensen, K. Jensen, "*CPN tools for editing, simulating,and analysing coloured Petri nets*", in:24th International Conference on Applications and Theory of Petri Nets, ICATPN 2003, in: Lecture Notes in Computer Science, vol.2679, Springer, Berlin, Heidelberg, (2003), pp. 450–462.

[6] S. Baarir, M. Beccuti, D. Cerotti, M.D. Pierro, S. Donatelli, G. Franceschinis, "*The great SPN tool: recent enhancements*", ACM SIGMETRICS Performance Evaluation Review 36 (4) (2009) 4–9.

[7] N. R. Mead, T. Stehney, "*Security Quality Requirements Engineering (SQUARE) Methodology*", Proc. of the 2005 workshop on software engineering for secure systems--building trustworthy applications, Missouri, USA, (2005), pp. 1-7.

[8] C. B. Haley, R. Laney, J. D. Moffett, et aI., "*Security Requirements Engineering: A Framework for Representation and Analysis*", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 34(1) (2008), pp. 133-153.

[9] D. Gordon, T. Stehney, N. Wattas, E. Yu, "*Quality Requirements Engineering (SQUARE): Case Study on Asset Management System*", Phase II (CMU/SEI-2005-SR-005). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, (2005).

[10] Hui Wang, Zongpu Jia, Zihao Shen,"*Research on Security Requirements Engineering Process*" 978-1-4244-3672-9/09/$25.00 © (2009) IEEE – pp 1285-1288.