Modeling system integrity of a security critical system using Colored Petri Nets

S. H. Houmb[1] & K. Sallhammar[2]
[1] *Department of Computer and Information Science, NTNU, Norway*
[2] *Centre for Quantifiable Quality of Service in Communication Systems, NTNU, Norway*

Abstract

Recently, the need for techniques for quantification of security attributes of IKT systems has been raised. This relates both to security requirements in QoS architectures, as well as input to trade-off analysis regarding the design and choice of security mechanisms to comply with an established security policy. Early research in this area has focused on state transition models, such as Markov or semi-Markov models. In the dependability domain these techniques are used to measure values such as mean time between failures (MTBF), and to quantify frequency and consequences of risks. The dynamics of security attacks makes it intractable to use, due to the problems with state explosions. To be able to express the complete state space of a security critical system, one needs to consider not only hardware, operating system, and application/services faults, but also the survivability of the system in terms of intentional and accidental security breaches. In this paper, we build a stochastic prediction system to estimate the system integrity of a security critical system. We make use of Colored Petri Nets (CPN), a higher-level formalism for stochastic modeling, analysis, and simulation. The prediction system is implemented as a generic and hierarchic CPN model.
Keywords: Colored Petri Nets, stochastic modeling, operational security, quantification of risk, risk management

1 Introduction

System integrity is the property that a system performs its intended function in an unimpaired manner, free from deliberate or accidental unauthorized manipulation of the system and its data. The ISO 15408 “Common Criteria” standard [1] pro-

vides criteria for qualitative evaluations of the security level of a system, while "ISO 13355 – Guidelines for the management of IT Security" [2] provides guidelines on risk management of IT security. However, none of these standards provides sufficient support to express the operational (run-time) security level of a system.

As for dependable systems, the behavior of a security critical system can be described in means of state transitions diagrams. However, to be able to express the complete state space of a security critical system, one need to consider not only intentional and accidental security breaches, but also hardware, operating system, and application/services faults. The latter three issues may be covered by the use of traditional techniques from the dependability domain, while the first two, which are characterized by being consciously performed with malicious intentions, pose more stringent requirements on the dynamics of the model.

In this paper we build a prediction system to quantify operational system integrity. The prediction system is based on the recommendations of ISO 13335 and Common Criteria, especially the class FPT - Protection of the ToE Security Functions, which focus on protection of system integrity. To cover all relevant aspects, we make use of Colored Petri Nets (CPNs). A CPN uses colored tokens, which gives the opportunity to distinguish between the different fault types. The remainder of this paper is organized as follows. In Section 2 we describe early work on quantifying operational security. In Section 3 we present the ontology for system integrity, while we in Section 4 describe the generic hierarchical CPN model. We conclude the paper by summing up the main contributions, as well as point to future work.

2 Related work

In Littlewood et al. [3] a first step towards operational measures of computer security is discussed. The authors point to the lack of quantitative measures for determining operational security, and presents a model based on traditional probability theory. They re-define the input space and usage environment for dependable systems, by including intentional attacks posed upon the system. In [4] Ortalo and Deswarte present a quantitative model to measure known Unix security vulnerabilities using a privilege graph, which is transformed into a Markov chain. The model allows for the characterization of operational security expressed as *mean effort to security failure*, as proposed in Littlewood et al.

In [5] Madan et al. consider security to be a Quality of Service (QoS) attribute and, based on ideas from [3], present an approach to quantify security attributes of intrusion tolerant software systems using stochastic modeling techniques. Wang et al. [6] extends the state transition approach in [5]. They argue for the difficulty of capturing details of real architectures in manually constructed Markov models, and advocate the use of Stochastic Petri Nets (SPN). A similar approach is used in Singh et al. [7], which describes an approach for probabilistic validation of an intrusion-tolerant replication system. They use a hierarchical stochastic activity

nets (SAN) model to validate intrusion tolerant systems, and to evaluate merits of various design choices.

In [8] Jonsson and Olovsson present a quantitative analysis of attacker behavior. The analysis is based on empirical data collected from intrusion experiments performed by undergraduate students at Chalmers University in Sweden. The result from the experiment showed that a typical attacker behavior comprises three phases; the learning phase, the standard attack phase, which is indicated to be exponentially distributed, and the innovative attack phase.

The work presented in this paper is based on the initial concepts discussed in [3] and adopted in [4–8]. We consider two aspects when quantifying risk; system failures caused by normal and accidental usage, and intentional attacks caused by the security environment. By not restricting our approach to intrusion tolerant systems, we generalize the idea of the hierarchical model in [7], and provide a more generic model than [6].

3 Concepts and ontology for operational system integrity

In [9] Jonsson and Olovsson present an integrated dependability and security framework. In the framework the input characteristics to a system are interpreted in protective terms, whereas the output characteristics are interpreted in behavioral terms, with respect to the user of the system. Furthermore, they distinguish authorized users from unauthorized users. The focus is on the functional requirements of the system, meaning that systems should be secure and dependable at the same time. Confidentiality and availability is defined as behavioral concepts, and integrity as a protective concept.

In our approach we extend the framework of Jonsson and Olovsson, by not only considering aspects related to the influence from the environment, but also system failures due to hardware faults, operating system faults, application/services faults, or other incorrect behavior from the system. Figure 1 illustrates the extended dependability and security framework. The system model consists of the object system, often called target of evaluation (ToE), operating in its security environment. The object system interacts with, and delivers services to its environment, which represents the system behavior. Furthermore, there is an influence on the system from its environment, which gives inputs to the system. Figure 2 depicts the ontology for the system integrity framework by describing the main concepts and their relations. There are two main sources that determines the system integrity of a security critical system; failure rate of object system describing the dependability of a system, and success rate of security breaches from the environment describing the security of a system. Dependability is defined as the combination of the reliability, safety, and maintainability of a system [10]. The security of a system is described the preservation of the security attributes confidentiality, integrity, and availability [1, 2].

A security breach is caused by a security attack, which represent any action that may compromise the security of the system. Safeguard is a practice, procedure or mechanism that reduces risk. A threat is a potential undesired event that

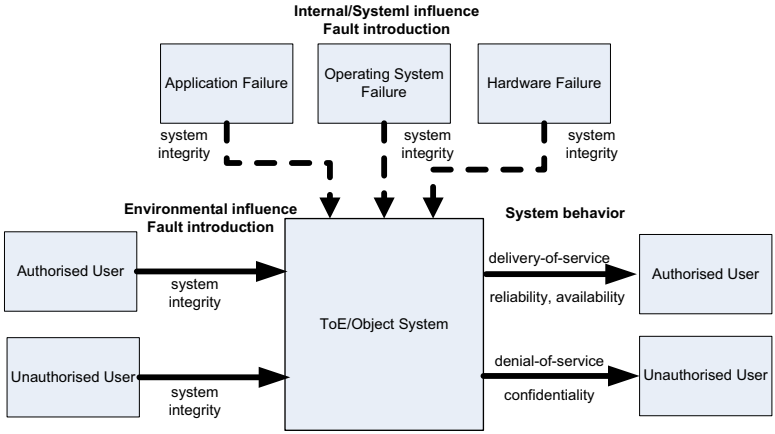


Figure 1: The extended dependability and security framework; describing system behavior as a result from environmental and internal influence on the object system.

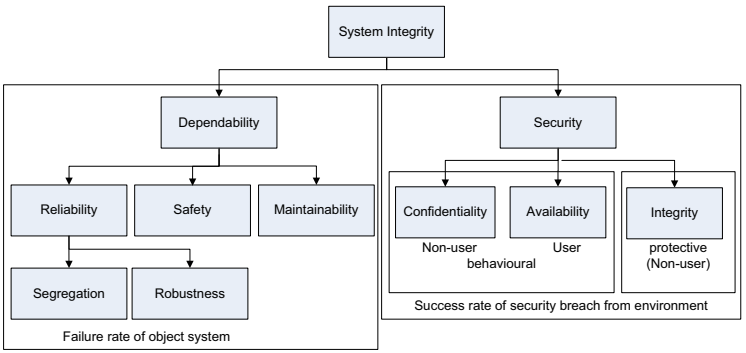


Figure 2: Concepts and their relations describing system integrity of security critical systems.

might exploit vulnerabilities in the object system, leading to an unwanted incident [6, 11]. Figure 3 describes this relationship as three different scenarios. (1) A threat is initiated, but no vulnerability exists. (2) A threat is initiated and exploits a vulnerability, which leads to an unwanted incident. (3) A threat is initiated, but a safeguard exists and prevents the threat from exploiting the vulnerability.

The failure rate of the object system, denoting system failures, may be due to hardware, operating system, and application/services faults or any combination of the three. We define the failure space as a set of type, attribute, and layer as depicted in Figure 4. *Type* denotes the type of failure, meaning whether or not the failure is caused by intentional or accidental events. *Attribute* denotes the

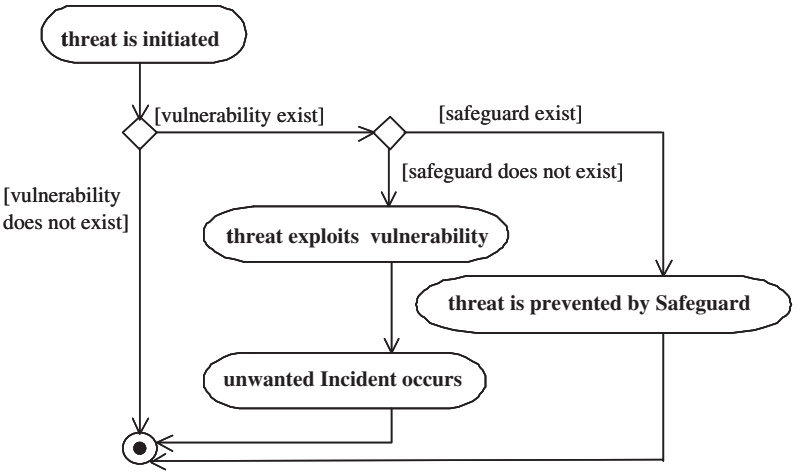


Figure 3: Overview of the relationship between threat, vulnerability, safeguard, and unwanted incident.

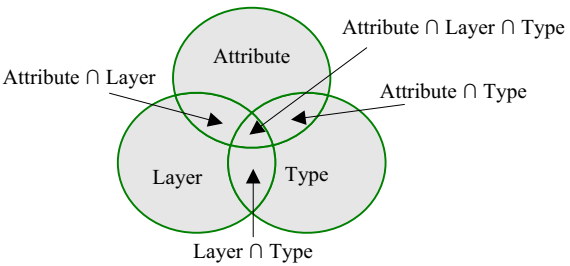


Figure 4: Possible combinations of failure type, security attributes, and the OSI layers presented as a Venn diagram.

security attributes involved; confidentiality, integrity, and availability. *Layer* refers to a certain layer in the OSI-model.

4 Modeling system integrity using CPN

The dynamics of possible combinations of fault sources, which may cause a system integrity breach, makes it intractable to use state transition models. Firstly, the state space will grow exponentially compared to the number of threats and vulnerabilities in the system being modeled, eventually leading to a state space explosion. Secondly, capturing details of real system architectures in a manually constructed Markov model is difficult [6]. We therefore advocate the use of Colored Petri Nets (CPN); a higher-level formalism for modeling, analysis, and simulation of state space models that is more concise and closer to a designer's intuition.

4.1 Colored Petri Nets

A CPN model of a system consists of *places*, *transitions*, and *tokens*. In contrast to state transition models, the possible states are not represented by the places, but rather by the *markings* of the CPN model, where the current marking is the number of tokens in each place. In CPNs the token are referred to as colored, and carries a data value that belongs to a given *type*. Arc expressions are used to describe changes in the state of the CPN when the transitions fires. To be able to *fire*, i.e. occur, a transition must have sufficient amount of tokens in its input places that match the arc expressions. The transition is then *enabled*. By using the time concept in CPN, it is possible to put deterministic or stochastic delays on transitions. An enabled transition may therefore not fire immediately, but rather after a random time delay. Hence, one is able to analyze the performance of not only Markov models, but also non-Markovian models by means of discrete event simulation. The formal definition of the syntax and semantics of the CPNs used in this paper can be found in [12].

4.2 A hierarchical CPN system model

We use the ontology for system integrity to design the generic and hierarchical CPN model. Figure 5 depict the top level of the generic hierarchical CPN model, while Figure 6 gives an overview of the data types used. The model includes intentional, accidental vulnerability insertion, as well as hardware, operating system, and application/services fault execution.

The model is defined to be in the state "system integrity breach", whenever the marking of place "System integrity breaches" has one or more tokens of any color. An ideal security critical system, operating in an ideal world, should initially be in a completely secure state; the marking where the place "Vulnerabilities outside system" contains all the tokens in the CPN model. Tokens of the color "Threat" represent the possible environmental vulnerabilities that may be inserted into the system, as well as non-environmental possible faults that may be enabled. However, in practice a system will always be vulnerable, at least with respect to physical deterioration of hardware components, and therefore another initial marking of the CPN model should be used.

When the transitions "Vulnerability insertion" and "Vulnerability removal" is enabled they will fire after timed delays, sampled from probability distributions, representing the variability of time to insertion of new vulnerabilities in the system, and the time to detect and remove vulnerabilities from the system. Note that there is a probability, (P_V), embedded in the transition "Vulnerability removal", which can be used to model the system's ability to detect different kind of vulnerabilities that may exist in the system. If the transition "Vulnerability insertion" fires, one or more tokens are transferred to the place "Vulnerabilities idle in system". This enables the transitions in the underlying sub-models, which describe how these vulnerabilities may cause systems integrity breaches.

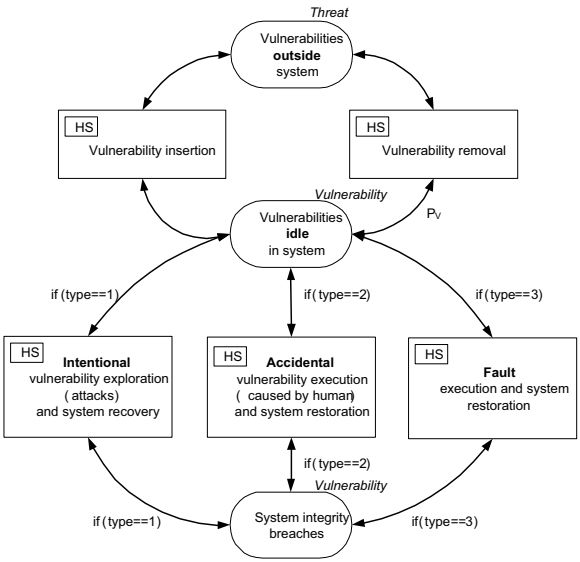
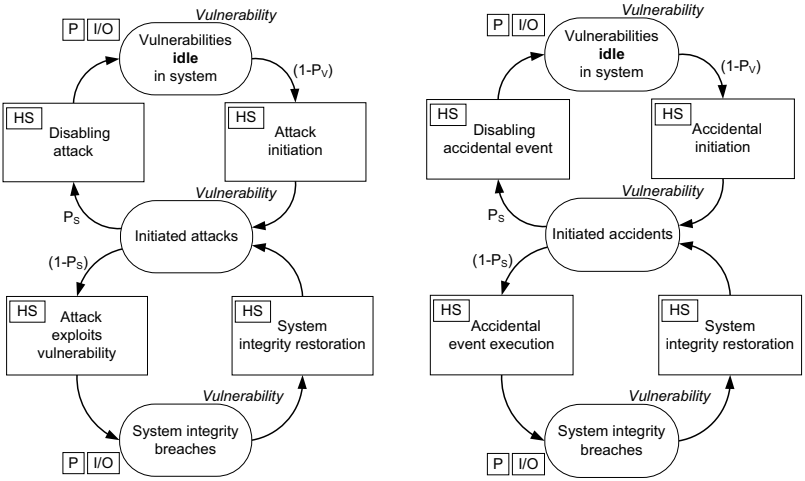


Figure 5: The top level of the hierarchical CPN model for system integrity breach analysis.

```
(* -- threats and vulnerabilities -- *)
color Threat = product AttributeList * LayerList * Type * Time timed ;
color Vulnerability = product AttributeList * LayerList * Type * Time timed ;
color Time = TIME ;

(* -- security attributes -- *)
color Attribute = with confidentiality | integrity | availability ;
color AttributeList = list Attribute ;

(* -- system layer -- *)
color Layer = with A/S | OS | HW ;
color LayerList = list Layer ;

(* -- type of breach/failure -- *)
color Type = with  intentional | accidental | failure ;
```

Figure 6: The data types of the hierarchical CPN model in Figure 5.

In the following we describe the CPN sub-models for the transitions representing intentional and accidental vulnerability execution. The sub-model "Fault execution and system restoration" models failure situations described by traditional dependability theory, and will not be explained in further detail in this paper. Readers are referred to textbooks on the subject, such as e.g. [13].



(a) Intentional Vulnerability Exploiting (b) Accidental Vulnerability Exploiting

Figure 7: The CPN sub-models.

4.3 Intentional vulnerability exploiting

Figure 7(a) describes the CPN sub-model for intentional vulnerability exploitation, existing defense mechanisms in the system, and system restoration. This sub-model is a generic model describing system integrity breaches caused by intentional vulnerability exploits. Tokens in the place "Vulnerabilities idle in system" represents the vulnerabilities that currently exists in the system, but that have not yet caused a system integrity breach. There is a probability, $(1 - P_V)$, that the system will fail to detect or remove these vulnerabilities, and thereby prevent the initiation of security attacks. The timed delay from a vulnerability is introduced into the system until a corresponding attack is initiated, i.e. the delay before the transition "Attack initiation" fires, represent the time elapsed before an attacker discovers the vulnerability, and has gained enough knowledge to start exploiting the vulnerability. This delay may be modeled as the "Learning Attack Phase" introduced in [8].

When an attack has been initiated, either the transition "Disabling attack", or the transition "Attack exploits vulnerability" will eventually fire. Which of the transitions that fires depends on whether a safeguard for that particular attack exists. The variable (P_S) models the probability that an existing safeguard manage to withstand the attack. If a safeguard for a particular vulnerability does not exist or cannot disable the attack, the attack may exploit the vulnerability. The time an attacker spends before succeeding with a particular attack, i.e. the stochastic delay before the transition "Attack exploits vulnerability" fires, may be modeled as the

"Standard Attack Phase" described in [8].

When the transition "Attack exploits vulnerability" fires, the attacker has succeeded with a particular attack and the token will be transferred from the place "Initiated attacks" to the place "System integrity breaches".

4.4 Accidental vulnerability execution

Figure 7(b) describes the sub-model for accidental vulnerability execution and system restoration. This model is an adapted version of the sub-model presented in the previous section. The model describes accidental events leading to vulnerability exploits. Such events may be prevented if an appropriate safeguard exists. An example of such an event is when user input is not checked for type–errors, which may lead to inconsistency or erroneous output. The sub-model covers all types of accidental events, such as stumbling over a network cable, accidentally turning off the power of a server, and user errors due to misunderstanding etc.

4.5 Simulation of CPN models

By running a large number of simulations of a CPN model one obtain a tight confidence intervals for steady state probabilities. In the hierarchical CPN model this means quantifying the system integrity of security critical systems. Measures used for quantification is the mean time to security integrity breach (MTSIB), or the corresponding measure; mean time to security failure (MTSF), as defined in [3] and utilized in [7] and [5]. However, in order to simulate the generic CPN model it must be refined into sub-models representing the level of the data available. In typical industrial applications, a CPN model usually consists of 10-100 different sub-models with varying complexity [12]. The next step is to assign probability distributions or deterministic values for all the timed transitions in the model. In some cases this task will be straight-forward, for example when there exists empirical data for certain kind of breaches on the system top level. However, in most cases the hierarchical sub-model must be refined into many subsequent places and transitions for which it is possible to approximate the timed delay, either by using empirical data, expert opinions, or a combination of both.

5 Conclusion and further work

The paper presents an approach for quantifying operational system integrity of security critical systems using CPN. The approach is based on the initial concepts of [3], and the suggestions provided in [4,5,8]. The extended security and dependability framework covers not only breaches caused by users and non-users, but also traditional dependability failures. CPN has the ability to distinguish between different sources of fault introduction, and makes it possible to model each combination explicitly. This is necessary in order to quantify system integrity, and to make use of the available data sources. Another important aspect is the complexity

of the model. Using CPN simplifies the structure of the model, even though the numbers of states derived from the model increases.

Further work will concentrate on constructing an extensive example, and demonstrate the feasibility of the modeling approach. The model will also be adapted to be used in connection with subjective expert judgment, as well as providing an approach for combining empirical and subjective data for predicting system integrity.

References

[1] ISO 15408: Common Criteria for Information Technology Security Evaluation, 1999. Http://www.commoncriteria.org/.

[2] ISO/IEC 13335: Information Technology - Guidelines for the management of IT Security. Http://www.iso.ch.

[3] Littlewood, B., Brocklehurst, S., Fenton, N., Mellor, P., Page, S., Wright, D., Dobson, J., McDermid J. & Gollmann, D., Towards operational measures of computer security. *Journal of Computer Security*, **2**, pp. 211–229, 1993.

[4] Ortalo, R. & Deswarte, Y., Experiments with quantitative evaluation tools for monitoring operational security. *IEEE Trans Software Eng*, **5(25)**, pp. 633–650, 1999.

[5] Madan, B., Vaidyanathan, K. & Trivedi, K., Modeling and quantification of security attributes of software systems. *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*, 2000.

[6] Wang, D., Madan, B. & Trivedi, K., Security analysis of sitar intrusion tolerance system. *ACM SSRS'03*, 2003.

[7] Singh, S., Cukier, M. & Sanders, W., Probabilistic validation of an intrusion-tolerant replication system. *International Conference on Dependable Systems and Networks (DSN'03)*, eds. de Bakker, J.W., de Roever, W.-P. & Rozenberg, G., 2001.

[8] Jonsson, E. & Olovsson, T., A quantitative model of the security intrusion process based on attacker behavior. *IEEE Trans Software Eng*, **4(25)**, p. 235, 1997.

[9] Jonsson, E. & Olovsson, T., On the integration of security and dependability in computer systems. *IASTED Int'l Conf. Reliability, Quality Control and Risk Assessment*, Washington, pp. 93–97, 1992.

[10] Avizienis, A., Laprie, J. & Randell, B., Fundamental concepts of dependability, 2001.

[11] Zimmermann, A., Dalkowski, K. & Hommel, G., A case study in modeling and performance evaluation of manufacturing systems using colored petri nets. *Proc. of the 8th European Simulation Symposium*, pp. 282–286, 1996.

[12] Jensen, K., An introduction to the theoretical aspects of coloured petri nets. *Lecture Notes in Computer Science; A Decade of Concurrency*, **803**, pp. 230–272, 1993.

[13] R. A. Sahner, A.P., K. S. Trivedi, *Performance and Reliability Analysis of Computer Systems*. Kluwer Academic Publishers, 1996.