A Method for Modeling and Quantifying the Security Attributes of Intrusion Tolerant Systems*

Bharat B. Madan[1]†, Katerina Goševa-Popstojanova[2], Kalyanaraman Vaidyanathan[3] and Kishor S. Trivedi[1]

[1] *Department of Electrical and Computer Engineering*
Duke University, Durham, NC 27708
{bbm,kst}@ee.duke.edu

[2] *Lane Department of Computer Science and Electrical Engineering*
West Virginia University, Morgantown, WV 26506
katerina@csee.wvu.edu

[3] *RAS Computer Analysis Laboratory, Sun Microsystems, Inc.*
9515 Towne Centre Drive, USAN10-103, San Diego, CA 92121
kalyan.vaidyanathan@sun.com

Abstract

Complex software and network based information server systems may exhibit failures. Quite often, such failures may not be accidental. Instead some failures may be caused by deliberate security intrusions with the intent ranging from simple mischief, theft of confidential information to loss of crucial and possibly life saving services. Not only is it important to prevent and/or tolerate security intrusions, it is equally important to treat security as a QoS attribute at par with other QoS attributes such as availability and performance. This paper deals with various issues related to quantifying the security attributes of an intrusion tolerant system, such as the SITAR *system. A security intrusion and the response of an intrusion tolerant system to an attack is modeled as a random process. This facilitates the use of stochastic modeling techniques to capture the attacker behavior as well as the system's response to a security intrusion. This model is used to analyze and quantify the security attributes of the system. The security quantification analysis is first carried out for steady-state behavior leading to measures like steady-state availability. By transforming this model to a model with absorbing states, we compute a security measure called the "mean time (or effort) to security failure" and also compute probabilities of security failure due to violations of different security attributes.*

*This work is sponsored by the U.S. Department of Defense Advanced Research Projects Agency (DARPA) under contract N66001-00-C-8057 from the Space and Naval Warfare Systems Center - San Diego (SPAWARSYSCEN). Katerina Goševa-Popstojanova is funded in part by a grant from the NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) managed through the NASA Independent Verification and Validation (IV&V) Facility, Fairmont, West Virginia. The views, opinions and findings contained in this paper are those of the authors and should not be construed as official DARPA or SPAWARSYSCEN's positions, policy or decision.

†Contact author

1 Introduction

It is imperative for well designed software systems to meet certain Quality-of-Service (QoS) requirements, such as reliability, availability and performance. Increasingly, such systems are being put to use in mission critical applications in military, aerospace, e-commerce, governmental, and health care applications. At the same time, most such software systems are network accessible through public networks, such as the Internet. As a result, these applications and systems have become prone to security intrusions. The range of security intrusions may vary from minor mischief for pleasure, denial of service, and criminal intent for stealing or destroying assets controlled by such systems. This has brought security attribute of a software to the forefront of QoS specifications. As is the case with other common QoS measures, (reliability, availability etc.), qualitative evaluation of security attributes may no longer be acceptable. Instead, we need to quantify security so that a software system may be able to meet contracted levels of security.

1.1 Related work

Most of the reported research in the literature on security characterization has dealt with the security of computing and information systems from a qualitative point of view. A system is assigned a given security level with respect to the presence or absence of certain functional characteristics and the use of certain development techniques. Very few studies have considered the quantitative assessment of security. From the point of view of the systems analysis, there are some similarities between the failures caused by accidental faults and the failures caused by the security intrusions. Consequently, we can leverage vast amount available research dealing with reliability and availability analysis of systems for carrying security quantification analysis. Littlewood *et. al* [15] discuss the commonalities between reliability and security from the perspective of evaluating measures of operational security of software systems. This paper identified several open questions that need to be answered before the quantitative approach can be taken further. In Section 1.2 below, we elaborate further on similarities and differences between failures caused by accidental faults and deliberate security intrusions. Ortalo *et. al* [19] describe a methodology for modeling known Unix security vulnerabilities as a *privilege graph*. Swiler *et. al* [22] and Jha *et. al* [11] use a slightly data structure which they refer to as the *attack graph* to model the security vulnerabilities of a system and their exploitation by an attacker. Ortalo $et.al$ [19] also describe a technique for transforming a privilege graph into a Markov chain. The states of the resulting Markov chain denote the enhanced privileges gained by an attacker (or from the system's point of view, progressive deterioration towards the security failed states) as a result of series of atomic attacks on a system. The arcs on the other hand, represent the effort *e* spent by an attacker to cause state transitions in this Markov chain. The attacker's effort in injecting an atomic attack is modeled as a random variable with exponential distribution function $P(e) = 1 - exp(-\lambda e)$, where $1/\lambda$ is the mean effort to succeed in a given elementary attack. This model allows the evaluation of the proposed measure of operational security *mean effort to security failure*, analogous to mean time to failure. A quantitative analysis of attacker behavior based on empirical data collected from intrusion experiments was presented in [12]. A typical attacker behavior comprises of three phases: the learning phase, the standard attack phase, and the innovative attack phase. The probability for successful attacks, although for different reasons, is small during the learning and the innovative phases. During the standard attack phase the probability of successful attacks is considerably higher; the collected data indicated that the time between breaches in this phase are exponentially distributed.

Software is an inherently complex system. Despite the best efforts of software architects and coders, there are always some residual and unintended faults and/or security vulnerabilities present in a software system. The OASIS program [2, 21] for Information Assurance (IA) has recognized this reality and as a result, it has come out three generations of secure software systems. These are,

1. 1^{st} generation: Emphasis is on protection or prevention of security intrusions.

2. 2nd generation: Accepting that a small number of intrusions are impossible to prevent, the emphasis is on detecting such intrusions and alerting administrators for initiating remedial measures.

3. 3rd generation: Tolerating security intrusion and reconfiguring, regenerating or rejuvenating a system after a security intrusion has occurred.

Our interest is in studying the so called 3rd generation intrusion tolerant systems (such as the SITAR) for quantifying their security attributes. In this paper we propose to utilize stochastic modeling techniques for quantitative assessment of security attributes for intrusion tolerant systems. The resulting model turns out to be a generic model that considers intrusions with different impacts (e.g., compromise of confidentiality, compromise of data integrity, and denial of service attacks) and captures the dynamic behavior of the system in response to these intrusions. Clearly, such an analysis needs to take into account the not only how a system responds to an attack, but also the behavior of the attackers attempting to cause security intrusions. Both these aspects are random phenomena and it is a challenging problem to suitably characterize these random phenomena in terms of probability distributions and their parameterization. In order to keep our focus on developing the security quantization model, for the present, we ignore the issues related to model parameterization and instead used reasonable guessed values for various model parameters.

1.2 Intrusion tolerance versus fault tolerance

A software system is a complex system. Despite adherence to best software engineering practices, it is nearly impossible to eliminate all bugs or faults that will eventually cause the software to fail during its operational phase. Alternative to creating totally bug free software is to think in terms of building fault tolerance that ensures effective recovery from failures. Similarly, it may not be either possible or it may not be cost effective to design and implement software systems, that are guaranteed to be entirely secure. In the context of security, even a best thought of software may have inadvertent security vulnerabilities, which over the operational life time of a system will be identified and exploited by the attackers to cause security failures. The progression of software system towards failure due to accidental faults or security vulnerabilities is illustrated below in Figure 1:

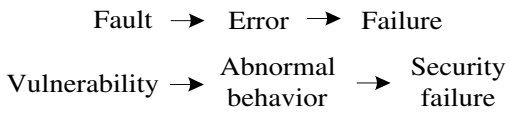


Figure 1. Failure progression

Intrusion tolerance is similar in philosophy to fault tolerance and may be a practical alternative to building secure software systems. An intrusion tolerant system allows for a finite probability that the system's security may be breached. However, an intrusion tolerant system will be able to detect either the insertion of a security attack into a system or the impending security failure due to such an attack and subsequently respond to such an attack that nullifies the adverse effects of an attack. Despite some similarities, there are also a few differences between fault tolerance and intrusion tolerance as enumerated below:

- Hardware or software failures experienced by a system are almost invariably accidental in nature. Such failures are caused either by the design faults, physical wear and tear, environmental conditions or by a peculiar set of inputs/excitations given to the system. In contrast, security intrusions are caused by deliberate user actions. It is however quite possible that a security intrusion may manifest itself as a failure. For example, stack overflow may either crash a system resulting in denial of service or it may be used to invoke a piece of hostile code.

- As mentioned in the previous point, there is an active attacker who causes a security intrusion unlike a failure that occurs accidentally. As a result, an attacker has to spend time and effort in order to be able to cause a security intrusion. In general, these attacks could arrive at a random point in time, just as a failure may occur randomly. In either case, this randomness can be described by suitable arrival processes (e.g., Poisson, Markov modulated Poisson process (MMPP), Non-homogeneous Poisson process (NHPP) etc.) [23]. Similarly, the amount of time or effort that an attacker has to spend in injecting an attack can be modeled as a random variable that can be described by chosen distribution functions.

- Before injecting an attack into a system, the attacker has to actively identify vulnerabilities present in the system that could be exploited to subsequently cause a security intrusion. This contrasts with the fault tolerance situation in which a system is always assumed to be vulnerable to failures. Intermittent and latent faults are exceptions to this.

- Once a system has been subjected to a security attack, an intrusion tolerant system responds to this security threat in a manner similar to the actions initiated by a fault tolerant system, though the exact details of such actions will vary. This similarity allows us to adopt some of the well established stochastic modeling and analysis techniques (e.g., Markov chains, semi-Markov processes etc.) [23] that have been extensively used in the field of fault tolerance for modeling and analyzing the security intrusion behavior of a system.

Based on the above discussion, in this paper, we use the state transition model of the SITAR intrusion tolerant system described in [10]. From the security quantification point of view, since some of the sojourn time distribution functions may be non-exponential, the underlying stochastic model needs to be formulated in terms of a semi-Markov process (SMP). Next, we solve the SMP model to compute the following quantities for the purposes of quantifying the security measures.

- After computing steady-state probabilities of all the states, we can compute the steady-state availabilities.

- By making system failure states as *absorbing states* [23], the effort or time required to reach such absorbing states is computed to yield the *MTTSF* in a manner similar to the notion of MTTF.

- Computing the eventual probabilities of reaching each of the absorbing states, we can separate out the causes of different types of security violations.

The rest of the paper is organized as follows. In Section 2, we develop a semi-Markov model for a intrusion tolerant system like the SITAR system [25] from the security quantification view point. Section 3 deals with the irreducible SMP model that may be used to find the steady-state probabilities leading to the computation of steady state security measures, such as, availability, confidentiality or integrity. In Section 4, the SMP model with absorbing states is presented for computation of the *MTTSF* measure and the eventual probabilities of absorption into different security failed states. Numerical results of the analysis performed on the models for a sample model are presented in Section 5. Final conclusions are presented in Section 6 along with some future directions pertaining to estimating the parameters of the models used in this paper.

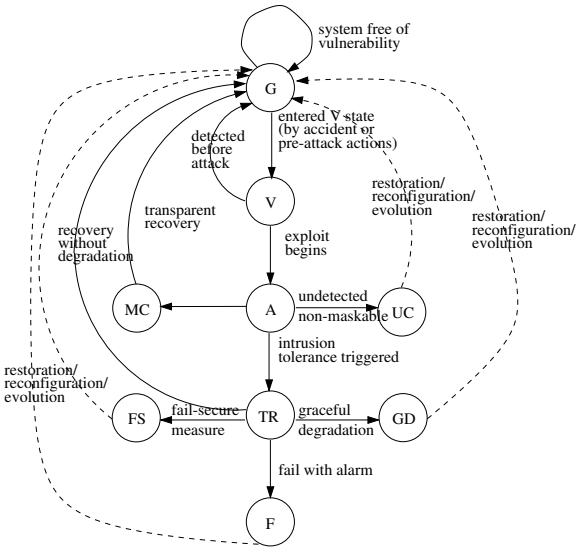
2. SMP Model for Security Quantification

A software system that is security intrusion tolerant has to be capable of reorganizing itself, preferably automatically, in order to mitigate the effects of a security intrusion. To analyze and quantify the security attributes of such a system, we have to take into account the actions of an attacker who is trying to cause security failures in conjunction with the corrective actions taken by the intrusion tolerant under attack. Therefore, we would require a composite security model that incorporates the behavior of both these elements. SITAR [1] [25] is a framework of

[1] SITAR is an acronym for "Scalable Intrusion Tolerant Architecture"

an intrusion tolerant architecture and its implementation using commercial off the shelf (COTS) components, e.g., Apache/Win-NT/Win2000 Web servers, standard firewalls, Java/Jini software technology etc., primarily targeted for the running intrusion tolerant Web service applications. The SITAR proprietary software sits in between standard firewall(s) and set of diverse and redundant Web servers. Various sub-systems communicate with each other using a shared space based on the the Jini/JavaSpaces [5]. Details of this architecture are beyond the scope of this paper and the interested readers may refer to [25]. In the context of the present paper, the SITAR system provides the conceptual components that allows us to define a generic abstract system capable of providing intrusion tolerance due to the presence of diversity and redundancy in the system.

2.1. Generic state transition model



G	good state	TR	triage state
V	vulnerable state	FS	fail-secure state
A	active attack state	GD	graceful degradation state
MC	masked compromised state	F	failed state
UC	undetected compromised state		

Figure 2. A state transition diagram for intrusion tolerant system

Figure 2 depicts the state transition model which we proposed in [10] as a framework for describing dynamic behavior of an intrusion tolerant system. This is a generic model that enables multiple intrusion tolerance strategies to exist and supports tolerance of intrusions with different impacts (e.g, compromise of confidentiality, compromise of data integrity, and denial of service attacks). Case studies that map several known vulnerabilities to this model are presented in [10]. Here, we briefly describe the basic concepts.

Traditional computer security leads to the design of systems that rely on resistance to attacks, that is, hardening for protection using strategies such as authentication, access control, encryption, firewalls, proxy servers, strong configuration management, and upgrades for known vulnerabilities. If the strategies for resistance fail, the system is brought from good state G into the vulnerable state V during the penetration and exploration phases of an attack. If the vulnerability is exploited successfully, the system enters the active attack state A and damage may follow. Intrusion tolerance picks up where intrusion resistance leaves off. The four phases that form the basis for all fault tolerance techniques are error detection, damage assessment, error recovery, and fault treatment [14]. These can and should be the basis for the design and implementation of an intrusion tolerant system.

Strategies for detecting attacks and assessment of damage include intrusion detection (i.e., anomaly based and signature based detection), logging, and auditing. If the probing that precedes the attack is detected, the system will stay in the good state. The other possibility is to detect the penetration and exploration phases of an attack and bring the system from the vulnerable state back to the good state. So far, the resistance and detection of attacks have received most of the attention, and once active attack state is entered damage may follow with little to stand in the way. Therefore, it is critical to use strategies for recovery which include the use of redundancy for critical information and services, incorporation of backup systems in isolation from network, isolation of damage, ability to operate with reduced services or reduced user community.

The best possible case is when there is enough redundancy to enable the delivery of error-free service and bring the system back to the good state by masking the attack's impact (MC). The worst possible case is when the intrusion tolerance strategies fail to recognize the active attack state and limit the damage, leading to an undetected compromised state UC, without any service assurance.

When an active attack in exploitation phase is detected, the system will enter the triage state TR. In the triage state, the system has to figure out different possible ways it can respond to an attack so as to recover or limit the damage that can be caused by the attack. Ideally, of course, the system should have in place some measures for eliminating the impacts produced by an attack, providing successful restoration to the good state. However, when restoration to the good state is not feasible, the system could attempt to limit the extent of damage while maintaining the essential services. Essential services are defined as the functions of the system that must be maintained to meet the system requirements even when the environment is hostile, or when failures or accidents occur that threaten the system [9]. Of course, there is no "one size fits all" solution. Moreover, it should be recognized that for an intrusion tolerant system, the impacts are more important than the causes. If the aim is to protect the system from denial of service (DoS) attack, the system should enter the graceful degradation state GD, maintaining only essential services. However, if the aim is to protect confidentiality or data integrity the system must be made to stop functioning. This is called the fail-secure state FS, analogous to the fail-safe state in fault tolerance. If all of the above strategies fail then the system enters the failed state, F, and signals an alarm.

Recovering the full services after an attack and returning to the good state by manual intervention is represented by transitions denoted with dashed lines. In order to reduce the effectiveness of future attacks it may be required to use techniques such as reconfiguration or evolution of the system. This phase can be considered analogous to fault treatment phase in fault tolerance.

Next, we develop the stochastic model of intrusion tolerant systems appreciating that the uncertainty in security arises from the incompleteness of our knowledge. To an attacker with an incomplete knowledge of the system, there is uncertainty as to the effects of the attack. To the system designer/owner/operator, there is uncertainty as to the type, frequency, intensity and the duration of the attack, and even as to whether a particular attack would result in a security breach. In developing such a theory we need to describe the events that trigger transitions among states in terms of probabilities and cumulative distribution functions (CDF).

2.2. Attacker's behavior and system's response

In order to analyze the security attributes of an intrusion tolerant system, we need to consider the actions undertaken by an attacker as well as the system's response to an attack. An attacker always tries to eventually send such a system into a security-failed state. Obviously, this requires the attacker to spend time and effort. In general, this time or effort [2] is best modeled as a random variable. Depending on the nature of an attack, this random variable may follow one of several distribution functions. In this paper, we borrow some of the common distribution functions used in the field of reliability theory. Deterministic, exponential, hyper-exponential, hypo-exponential, Weibull, gamma and log-logistic etc. are some of the distribution function that make sense in the context of security analysis [23]. The hypo-exponential distributions may be used to model transitions that may

[2] Henceforth, we will use time to represent both time and/or effort

involve multi-stage activities. For example, the *Code-Red* worm [20] first causes the parameter stack buffer to overflow by sending a long URL to the web server that is to be attacked. In the next stage, normal return address (already stored on this stack) is over-written with a bad return address placed in this URL. In the final stage, this bad return address points to a rogue piece of *Code-Red* code (also supplied as a part of the long URL) that gets invoked next time the return from a call is executed. The above discussion suggests that we need to consider non-exponential type of distributions. The hypo-exponential distribution may be used to model threat situations that can cause monotonically increasing failure rate (IFR) of security. Similarly, hyper-exponential distribution may be used to model threats that have can cause monotonically decreasing failure rate (DFR). Weibull distribution function may be used to model constant failure rate (CFR), DFR or IFR type of threats by suitably choosing its parameters. For more complex attack scenarios, that are characterized by having a decreasing rate of success initially, followed by an increasing rate of success (or vice-a-versa), we can use the log-logistic type of distribution function. It should also be noted that an attacker may not always be successful in causing a security failure, i.e., probability of success ≤ 1. In relation to the state transition diagram described in Figure 2, an attackers actions are modeled by the states $\{G, V, A\}$.

An intrusion tolerant system needs to constantly evaluate the presence of any security penetration. Once a security attack is detected, the system needs to initiate suitable remedial actions. After detecting an attack, a SITAR like system can respond in a variety of ways. However, the basic nature of this response would be to try to make the system move back to a secure state from a security-compromised state. Obviously this movement will require time or effort on the part of the system. As before, this time or effort is best modeled as a random variable that is described by a suitable probability distribution function. It should again be remarked here that it is not guaranteed that the system will be able to detect all attacks, i.e., probability of detection of an attack is ≤ 1 in general. Thus system's response may be parameterized by a set of distribution functions and a set of probabilities. For a SITAR like system, the system's response to a security intrusion may be described by the states $\{MC, UC, TR, FS, GD, F\}$ and the transitions between these states. A system's response to a security attack is fairly automated and could be quite similar to how it may respond to accidental faults. Let $\{X(t) : t \geq 0\}$ be the underlying stochastic process with a discrete state space $X_s = \{G, V, A, TR, MC, UC, FS, GD, F\}$. To analyze an SMP, we need to deal with two sets of parameters [23, 7]

1. mean sojourn time sojourn time h_i in state $i \in X_s$, and
2. the transition probabilities p_{ij} between different states $i \in X_s$ and $j \in X_s$.

We note that the analysis carried out in this paper depends only on the mean sojourn time and is independent of the actual sojourn time distributions for the SMP states. If we were to carry out a transient analysis of the SMP, this will no longer be true.

When analyzing security, we may also be interested in computing the calendar time it takes to cause such a transition. In such cases, we have to establish a relationship between effort and time. In general, effort is a random function of time since time required to cause a transition depends on several randomly behaving entities, e.g., attacker's level of expertise and background, robustness of the system, type of pre-existing system vulnerabilities etc. This will result in a doubly-stochastic model. However, this paper limits itself to dealing with a SMP model only. The doubly stochastic model is planned to be covered separately in a future paper. In the present paper, we will ignore the difference between time and effort and use these terms interchangeably.

2.3. Security Attributes

In this section, we discuss various security related quality attributes using the SMP model presented in the previous section. Based on the classification of the Markov chains, two different types of security attributes can be visualized, based on the two classes of SMPs, namely *irreducible SMP* and *SMP with absorbing states* [23].

An irreducible SMP can be analyzed in terms of the long term steady-state probabilities of various states. Many researchers consider the steady-state security related quality attributes to be a part of the dependability [8]. Dependability is defined as the property of computer system such that reliance can justifiably be placed on the service it delivers [13]. Dependability attributes include:

- *Reliability*, continuity of service
- *Safety*, non-occurrence of catastrophic consequences
- *Maintainability*, ability to undergo repairs and evolutions.
- *Availability*, readiness for usage
- *Integrity*, data and programs are modified or destroyed only in a specified and authorized manner
- *Confidentiality*, sensitive information is not disclosed to unauthorized recipients.

The present work is concerned primarily with evaluating the last three attributes. Associating integrity and availability with respect to authorized actions, together with confidentiality, leads to *security* [4]. The degree to which each of these properties is needed varies from application to application. For instance, the defense industry is primarily interested in confidentiality. In contrast, the banking industry is primarily interested in integrity, and the telephony industry may value availability most. The exact requirements that are needed for a particular system or application are expressed in the security policy for that system or application. Section 3, discusses the use of the irreducible SMP model and steady-state probabilities to arrive at the steady-state availability analysis of the SMP security model.

In contrast to the irreducible SMP model, the analysis of an absorbing state SMP model deals with identifying the absorbing states and finding (i) the time (or effort) required for the SMP to reach any of the absorbing states (ii) finding the probabilities of reaching different absorbing states. Typically, the absorbing states are the (security) failed states. Therefore, analysis of this SMP model will yield the *MTTSF* measure. Section 4 deals with the evaluation of the *MTTSF* security measure of a system.

3 Irreducible SMP- Availability analysis

While the methods for quantitative assessment of dependability attributes such as reliability, availability, and safety are well established, so far the security attributes have been mostly assessed from the qualitative point of view. In this section, we derive and evaluate dependability attributes that are relevant to security.

Instantaneous availability $\mathcal{A}(t)$ of a system is defined as the probability that the system is properly functioning at time t. We are interested in the *steady state availability* $\mathcal{A}$ as the limiting value of $\mathcal{A}(t)$ as t approaches infinity. For our model, the system is unavailable in states FS, F, and UC, that is, the availability $\mathcal{A}$ is given by

$$\mathcal{A} = 1 - (\pi_{FS} + \pi_F + \pi_{UC}) \tag{1}$$

where, $\pi_i, i \in \{FS, F, UC\}$ denotes the steady state probability that the SMP is in state i. Note that for some applications and types of attacks the system may be considered available in the state UC.

Availability is an appropriate measure for the compromise of data integrity and DoS attacks. It should be pointed out that in the case of DoS attacks which are aimed at disrupting normal services by consuming large amounts of service resources, states MC and FS do not make sense. Thus, it is not possible to mask DoS attack by using redundancy. Also, intentionally making the system to stop functioning, i.e., bringing it to the fail-secure state FS will accomplish the goal of DoS attack. Therefore, the states MC and FS will not be part of the state diagram describing DoS attacks. It follows that for the DoS attacks the system availability reduces to

$$\mathcal{A}_{DoS} = 1 - (\pi_F + \pi_{UC}) \tag{2}$$

In a similar manner, confidentiality and integrity measures can be computed in the context of specific security attacks. For example, Microsoft IIS 4.0 suffered from the so-called ASP vulnerability as documented in the Bugtraq ID 1002 [1]. Exploitation of this vulnerability allows an attacker to traverse the entire web server file system, thus compromising confidentiality. Therefore, in the context of this attack, states UC and F are identified with the loss of confidentiality. Similarly, if the well known $Codered$ worm is modified to inject a piece of code into a vulnerable IIS server to browse unauthorized files, states UC and F will imply loss of confidentiality. Therefore, the confidentiality measure can then be computed as:

$$\mathcal{C}_{ASP} = 1 - (\pi_F + \pi_{UC}) \tag{3}$$

The integrity measure in the presence of integrity compromising attacks can be computed in a similar manner. Take for example the CGI vulnerability present in the Sambar server as reported in the Bugtraq ID 1002 [1]. Exploitation of this vulnerability permits an attacker to execute any MS-DOS command including deletion and modification of files in an unauthorized manner, thus compromising the integrity of the system. Once again, states UC and F signal compromise of the integrity measure $\mathcal{I}_{CGI}$ which can be computed as in equations for $\mathcal{A}_{DoS}$ and $\mathcal{C}_{ASP}$.

3.1. DTMC steady-state probability computations

It was explained earlier that in order to carry out the security quantification analysis, we need to analyze the SMP model of the system that was described by its state transition diagram. The SMP corresponding to Figure 2 can be described in terms of its embedded DTMC shown in Figure 3.

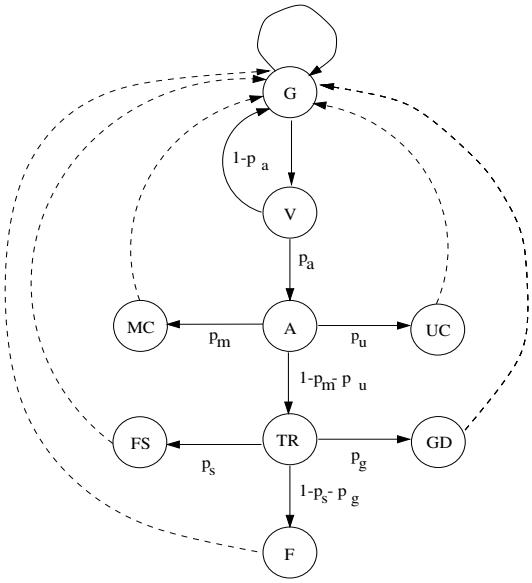


Figure 3. Embedded DTMC for the SMP model

As stated in Section 2 complete description of this SMP model requires the knowledge of various parameters, *viz.* mean sojourn time in each state and the transition probabilities. Some of the parameters of the SMP model are summarized here:
- h_G, Mean time for the system to resist becoming vulnerable to attacks
- h_V, Mean time for the system to resist attacks when vulnerable
- h_A, Mean time taken by the system to detect an attack and initiate triage actions

- h_{MC}, Mean time the system keeps the effects of an attack masked
- h_{UC}, Mean time that an attack remains undetected while doing the damage
- h_{TR}, Mean time the system takes to evaluate how best to handle an attack
- h_{FS}, Mean time the system operates in a fail secure mode in the presence of an attack
- h_{GD}, Mean time the system is in the degraded state in the presence of an attack
- h_F, Mean time the system is in the fail state despite detecting an attack
- p_a, Prob. of injecting a successful attack, given that the system was vulnerable
- p_u, Prob. that a successful attack has remained undetected
- p_m, Prob. that the system successfully masks an attack
- p_g, Prob. that the system resists an attack by gracefully degradation
- p_s. Prob. that the system responds to an attack in a fail secure manner.

Clearly for the model to be accurate, it is important to estimate accurately the model parameters (i.e., mean sojourn times and the DTMC transition probabilities) listed above. In this paper, however, our focus is primarily on developing a methodology for analyzing quantitatively the security attributes of an intrusion tolerant system rather than accurate model parameterization. In Section 6, we briefly discuss methods based on a-priori knowledge and intrusion injection experiments as suggested in [16, 17] that may be used to estimate these parameters. In the absence of exact values of model parameters, it will, however be meaningful to evaluate the sensitivity of security related attributes to variations in model parameters. In Section 5 we present some numerical results to evaluate the sensitivity of the $MTTSF$ and the steady-state availability A to various model parameters.

For computing the availability measure, we first need to compute the steady-state probabilities $\{\pi_i, i \in X_s\}$ of the SMP states. π_i's in turn can be computed in terms of the embedded DTMC steady-state probabilities ν_i's and the mean sojourn times h_i's [23]:

$$\pi_i = \frac{\nu_i h_i}{\sum_j \nu_j h_j}, \qquad i, j \in X_s. \tag{4}$$

The DTMC steady-state probabilities ν_i's can be computed as,

$$\overline{\nu} = \overline{\nu} \cdot P \tag{5}$$

where, $\overline{\nu} = [\nu_G \ \ \nu_V \ \ \nu_A \ \ \nu_{MC} \ \ \nu_{UC} \ \ \nu_{TR} \ \ \nu_{FS} \ \ \nu_{GD} \ \ \nu_F]$ and P is the DTMC transition probability matrix which can be written as:

$$P = \begin{array}{c} \\ G \\ V \\ A \\ MC \\ UC \\ TR \\ FS \\ GD \\ F \end{array} \begin{array}{c} \begin{array}{ccccccccc} G & V & A & MC & UC & TR & FS & GD & F \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \widetilde{p}_a & 0 & p_a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_m & p_u & \widetilde{p}_{mu} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_s & p_g & \widetilde{p}_{sg} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

where, $\widetilde{p}_a = 1 - p_a$, $\widetilde{p}_{mu} = 1 - p_m - p_u$ and $\widetilde{p}_{sg} = 1 - p_s - p_g$. In addition,

$$\sum_i \nu_i = 1, \ i \in \{G, V, A, MC, UC, TR, FS, GD, F\}. \tag{6}$$

The transition probability matrix P describes the DTMC state transition probabilities between the DTMC states as shown in Figure 3. The first step towards evaluating security attributes is to find the steady-state probabilities

ν's of the DTMC states by solving equations (5) and (6). Rewriting (5) into the elemental form yields relationship between DTMC steady-state probabilities as,

$$\begin{aligned}
\nu_G &= \nu_V(1-p_a) + \nu_{MC} + \nu_{UC} + \nu_{FS} + \nu_{GD} + \nu_F \\
\nu_V &= \nu_G; \quad \nu_A = \nu_V p_a = \nu_G p_a \\
\nu_{MC} &= \nu_A p_m; \quad \nu_{UC} = \nu_A p_u \\
\nu_{TR} &= \nu_A(1-p_m-p_u) = \nu_G p_a(1-p_m-p_u) \\
\nu_{FS} &= \nu_{TR} p_s \\
\nu_{GD} &= \nu_{TR} p_g; \quad \nu_F = \nu_{TR}(1-p_s-p_g)
\end{aligned} \tag{7}$$

Solving the above equations, in conjunction with the total probability relationship given by (6) gives,

$$\nu_G = \frac{1}{2 + p_a(3 - p_m - p_u)} \tag{8}$$

Substituting (8) into (7), will yield the expressions for the remaining ν's. In the next section, the DTMC steady-state probabilities are utilized to compute the SMP steady-state probabilities.

3.2. Semi-Markov model analysis

The mean sojourn time h_i in a particular state i is the other quantity that is needed to compute π_i's. h_i obviously is determined by the random time that a process spends in a particular state. In the computer security domain, there is a wide variety of attackers ranging from amateur hackers to cyber criminal syndicates to inimical intelligence agencies possessing a wide spectrum of expertise and resources. DARPA has recently initiated the Information Assurance (IA) program [16, 17] that aims to characterize a wide range of attacks. While many more studies need to be carried out to get a more complete understanding of the attacker behavior, DRAPA's IA studies point to the fact that in order to capture the attacker behavior, we need to consider a variety of attacks ranging from trivial to highly sophisticated. In the model being considered in this paper, the transitions $G \rightarrow V$ and $V \rightarrow A$ describe the attacker's behavior. Keeping in mind a wide range of attacks, we need to consider a variety of probability distribution functions describing attacker related transitions. A system's response on the other hand, is more algorithmic and automated that is not very different from how a system responds to conventional failures due to accidental faults. The important advantage of the approach developed in this paper for analyzing and quantifying security is its simplicity. Starting with the SMP model used to capture the security related behavior of a system, we can derive the embedded DTMC that involves only the transition probabilities. Given this DTMC model, the steady-state DTMC probabilities ν_i's can be easily computed as shown in the previous Subsection. Therefore, it suffices to know just the mean sojourn times h_i's, in order to compute SMP steady-state probabilities π_i's. As an example, if we assume the sojourn time distributions for two of the states, viz. G and V to be HypoEXP($\lambda_{g1}, \lambda_{g2}$) and Weibull($\lambda_v, \alpha_v$) respectively. Then, the corresponding mean sojourn times for state G and V are given by,

$$\begin{aligned}
h_G &= (1/\lambda_{g1} + 1/\lambda_{g2}), \\
h_V &= (1/\lambda_v)^{1/\alpha_v}\Gamma(1 + 1/\alpha_v)
\end{aligned} \tag{9}$$

The remaining states $\{A, MC, UC, TR, FS, GD, F\}$ are assumed to have mean sojourn times $\{h_A, h_{MC}, h_{GD}, h_{TR}, h_{FS}, h_{UC}, h_F\}$, respectively. The SMP steady-state probabilities π_i's can now be easily computed by using

equations (4), (5) and (6) as:

$$\begin{aligned}
\pi_G &= \frac{h_G}{h_G + h_V + p_a[h_A + p_m h_{MC} + p_u h_{GD} + (1-p_m-p_u)[h_{TR} + p_s h_{FS} + p_g h_{UC} + (1-p_g-p_s)h_F]]} \\
\pi_V &= h_V \frac{\pi_G}{h_G} \\
\pi_A &= h_A p_a \frac{\pi_G}{h_G} \\
\pi_{UC} &= h_{UC} p_a p_u \frac{\pi_G}{h_G} \\
\pi_{MC} &= h_{MC} p_m p_u \frac{\pi_G}{h_G} \\
\pi_{FS} &= h_{FS} p_a p_s (1-p_m-p_u) \frac{\pi_G}{h_G} \\
\pi_{TR} &= h_{TR} p_a (1-p_m-p_u) \frac{\pi_G}{h_G} \\
\pi_{GD} &= h_{GD} p_a p_g (1-p_m-p_u) \frac{\pi_G}{h_G} \\
\pi_F &= h_F p_a (1-p_s-p_g)(1-p_m-p_u) \frac{\pi_G}{h_G}. \qquad (10)
\end{aligned}$$

Given the steady-state probabilities, various measures, such as, availability, confidentiality and integrity may be computed via equations for $\mathcal{A}_{DoS}$ or $\mathcal{C}_{ASP}$.

3.2.1 Model of a SYN-flood DoS attack

A significant advantage of the SMP model described so far is its generic nature that can be easily specialized for specific security attacks. For example, when a system is being subjected to a SYN-flood DoS attack [3], the model reduces to states (G, V, A, UC, TR, GD, F). The resulting SMP with reduced number of states is as shown in Figure 4. Solution of this SMP based on the approach outlined earlier yields the following steady-state probabilities, π_i's, as a special case of (10).

$$\begin{aligned}
\pi_G &= \frac{h_G}{h_G + h_V + p_a[h_A + p_u h_{GD} + (1-p_u)h_{TR} + p_g(1-p_u)h_{UC} + (1-p_u)(1-p_g)h_F]} \\
\pi_V &= h_V \frac{\pi_G}{h_G} \\
\pi_A &= h_A p_a \frac{\pi_G}{h_G} \\
\pi_{UC} &= h_{UC} p_a p_u \frac{\pi_G}{h_G} \\
\pi_{TR} &= h_{TR} p_a p_s (1-p_u) \frac{\pi_G}{h_G} \\
\pi_{GD} &= h_{GD} p_a p_g (1-p_u) \frac{\pi_G}{h_G} \\
\pi_F &= h_F p_a (1-p_g)(1-p_u) \frac{\pi_G}{h_G} \qquad (11)
\end{aligned}$$

In the context of a DoS attack, availability is the only meaningful security attribute that can be computed using the equation for $\mathcal{A}_{DoS}$.

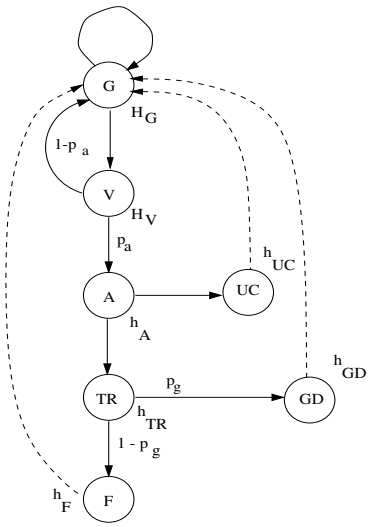


Figure 4. DoS attack - SMP model

4. SMP with absorbing states-MTTSF analysis

For quantifying the reliability of a software system, *Mean time to failure* (MTTF) is a commonly used reliability measure. MTTF provides the mean time it takes for the system to reach one of the designated failure states, given that the system starts in a good or working state. In reliability analysis, the failed states are made absorbing states. It is clear that once the system reaches one of the absorbing states, the probability of moving out of such a state is 0, i.e., outgoing arcs from such states are deleted. Using the MTTF analogy, we define *Mean time to security failure* (MTTSF) as the measure for quantifying the security of an intrusion tolerant system. MTTF or MTTSF can be evaluated by making those states of the embedded DTMC that are deemed to be failed or security-compromised states as the absorbing states. Classification of the SMP states into absorbing and transient categories depends on the actual nature of the security intrusion. For example, if the model is describing the *Sun web server bulletin board vulnerability (Bugtraq ID 1600)* [1], the states $X_a = \{UC, FS, GD, F\}$ will form the set of absorbing states, while $X_t = \{G, V, A, MC, TR\}$ will be the set of transient states. In contrast, for the SYN-flood DoS attack, $X_a = \{UC, GD, F\}$ and $X_t = \{G, V, A, TR\}$. The resulting transition probability matrix P then has the general form,

$$P = \left[\begin{array}{c|c} Q & C \\ \hline 0 & I \end{array}\right]$$

submatrices Q and C consists of the transition probabilities between transient states and from transient states to absorbing states respectively. Matrices Q and C are given by,

$$Q = \begin{array}{c} \\ G \\ V \\ A \\ MC \\ TR \end{array} \begin{array}{c} \begin{array}{ccccc} G & V & A & MC & TR \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1-p_a & 0 & p_a & 0 & 0 \\ 0 & 0 & 0 & p_m & 1-p_{mu} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

and,

$$C = \begin{array}{c} \\ G \\ V \\ A \\ MC \\ TR \end{array} \begin{array}{c} \begin{array}{cccc} UC & FS & GD & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ p_u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & p_s & p_g & 1-p_{sg} \end{bmatrix} \end{array}$$

where, $p_{mu} = p_m + p_u$ ans $p_{sg} = p_s + p_g$. To find the MTTSF using the approach outlined in [23, 6], $MTTSF$ may be written as,

$$\text{MTTSF} = \sum_{i \in X_t} V_i h_i \tag{12}$$

where V_i denotes the the average number of times the state $i \in X_t$ is visited before the DTMC reaches one of the absorbing states and h_i is the mean sojourn time in state i. The visit count elements V_i can be obtained by solving the system of equations,

$$V_i = q_i + \sum_j V_j q_{ji}, \qquad i, j \in X_t \tag{13}$$

where, q_i is the probability that the DTMC starts in state i. In our case, we assume that G is the initial state, that is,

$$\overline{q} = [q_i] = [1 \;\; 0 \;\; 0 \;\; 0 \;\; 0].$$

Solving (13) for the visit counts V_i's gives,

$$\begin{aligned} V_G &= \frac{1}{p_a(1-p_m)} \\ V_V &= V_G \\ V_A &= \frac{1}{1-p_m} \\ V_{MC} &= \frac{p_m}{1-p_m} \\ V_{TR} &= \frac{1-p_m-p_u}{1-p_m}. \end{aligned}$$

With the knowledge of the mean sojourn times h_i's in various states $\{i \in X_t\}$, we can use (12) to compute the MTTSF as,

$$\text{MTTSF} = \frac{h_G p_a^{-1} + h_V p_a^{-1} + h_A + p_m h_{MC} + (1-p_m-p_u) h_{TR}}{1-p_m} \tag{14}$$

When a system fails in the context of security (on an average, after MTTSF interval of time from the start time has elapsed) the system will find itself in one of the absorbing states. For example, for the Bugtraq 1600, this state will $\in \{UC, FS, GD, F\}$. Any security intrusion can have many security implication. Depending on the actual code inserted by intruder by exploiting the Bugtraq 1600 vulnerability, the intrusion may result in the compromise of user authentication and confidentiality in case the system finds itself in the UC state. Alternately, if the system reaches the FS or F, it may imply non-availability of some or all services. It is therefore important to be determine the final absorption state in probabilistic terms. In computational terms, this would require finding the SMP probabilities for the states $\in X_a$ after absorption. We now define a matrix $B = [b_{ij}]$, where, b_{ij} denotes the probability that the DTMC starting in a state $\{i \in X_t\}$ will eventually get absorbed in an absorbing state $\{j \in X_a\}$. It has been shown earlier in [18] that, $B = (I - Q)^{-1}C$. The first row elements of B can then be written as,

$$b_{1j} = \sum_{i \in X_t} V_i c_{ij} \quad j \in X_a$$

Therefore,

$$\begin{aligned} b_{1F} &= \frac{(1 - p_s - p_g)(1 - p_m - p_u)}{1 - p_m}, \\ b_{1FS} &= \frac{p_s(1 - p_m - p_u)}{1 - p_m}, \\ b_{1GD} &= \frac{p_g(1 - p_m - p_u)}{1 - p_m}, \\ b_{1UC} &= \frac{p_u}{1 - p_m}. \end{aligned} \tag{15}$$

In other words, once a security attack succeeds in causing a security failure, elements $\{b_{1j}, \quad j \in X_a\}$ give us the probability that after system has failed, it would reach the absorbing state j given that the system started in the G state.

5. Numerical Results

In this section we illustrate the evaluation of the security attributes for a numerical example. Since at this point accurate model parameter values for the SMP model are not known to us, we assume good 'guestimate' values for various model parameters. The use of 'gusetimate' model parameter values makes it relevant to also carry out sensitivity analysis of different measures to variations in the model parameter values.

Transition probabilities. It is assumed that a successful attack that brings the system from vulnerable state V to an active attack state A is less likely than detection of the attack and bringing the system back from vulnerable to good state G. In particular, the probability of a successful attack from the vulnerable state is assumed to be $p_a = 0.4$. Further, the probability that the system masks an attack successfully is $p_m = 0.3$ and the probability of an undetected attack is p_u = 0.2. Hence, the probability that an attack will be detected and the system will enter the triage state TR is $1 - p_m - p_u = 0.5$. From the triage state the system may transfer to a gracefully degradation state GD or fail secure FS state with probabilities $p_g = 0.6$ and $p_s = 0.3$ respectively. It follows that the probability that the system will fail with an alarm from the triage state is $1 - p_g - p_s = 0.1$.

Mean sojourn times. It is assumed that the system spends more time in the good G and vulnerable V states than in the active attack state A. Thus, the mean sojourn times for these states are $h_G = 1/2$, $h_V = 1/3$, and $h_A = 1/4$ time units. The mean time that the system spends in the state of masking an attack before it goes back to the good state is $h_{MC} = 1/4$ time units, while the mean time that an attack remains undetected before a manual

intervention brings the system back to the good state is $h_{UC} = 1/2$ time units. The system stays in the triage state on average $h_{TR} = 1/6$ time units. Further, we assume that the mean time that the system spends in fail secure state and graceful degradation state are $h_{FS} = 1$ and $h_{GD} = 4$ time units respectively. The mean time spent in the failure with an alarm state is assumed to be $h_F = 2$ time units.

Using the above values for the input parameters and equations (10) we obtain for following values for the steady – state probabilities of the semi Markov process:

$$\begin{array}{lll} \pi_G = 0.3092; & \pi_V = 0.2061; & \pi_A = 0.0618; \\ \pi_{MC} = 0.0185; & \pi_{UC} = 0.0.0247; & \pi_{TR} = 0.0206; \\ \pi_{FS} = 0.0371; & \pi_{GD} = 0.2691; & \pi_F = 0.0247. \end{array}$$

The steady - state probability π_i may be interpreted as the proportion of time that the SMP spends in the state i. For the assumed values of the input parameters, the proportion of time that the intrusion tolerant system spends in the good G and vulnerable state V is one order of magnitude higher that the time spend in any other state. Further, this particular intrusion tolerant system spends approximately 34 % of time in a good state. Assuming states FS, F and UC are the unavailable states the steady-state availability of this system is $\mathcal{A} = 0.9135$.

Using the same values for the input parameters and the equation (14) we estimate that the mean time to security failure is MTTSF = 3.5595 time units. Since a successful security attack may have different impacts on the system, we also estimate the probabilities of absorption in each of the states F, FS, GD, and UC. Using the equations (15) the following values for the probabilities of absorption are obtained:

$$\begin{array}{ll} b_{1F} = 0.0714; & b_{1FS} = 0.2143; \\ b_{1GD} = 0.4286; & b_{1UC} = 0.2857. \end{array}$$

For the assumed values of the input parameters, the system has the highest probability of absorption in the graceful degradation state (0.4286), followed by the probabilities of absorption in undetected compromise and fail secure states (0.2857 and 0.2143). The probability of system failing with an alarm is one order of magnitude lower than the probabilities of any other impact on the system. Sensitivity results for the sample model under consideration indiacte that both, the $MTTSF$ and the *steady-state availability* are more sensitive to p_a and h_G model parameters. Figure 5 and Figure 6 show $MTTSF$ and $Availability$ security measures, respectively, as functions of p_a and h_G. As is to be expected, $MTTSF$ monotonically decreases as p_a is increased. However, with respect to h_G, $MTTSF$ increases monotonically. The $Availability$ in contrast is a decreasing function of p_a and an increasing function of h_G.

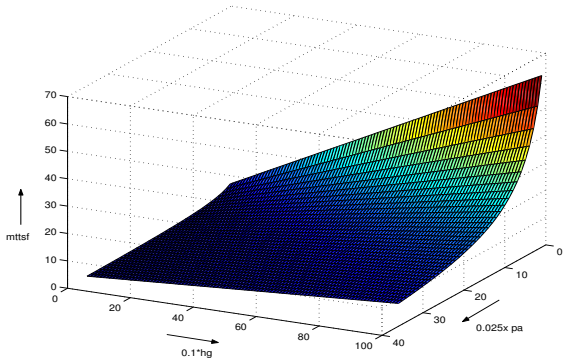


Figure 5. MTTSF as a function of p_a and h_G

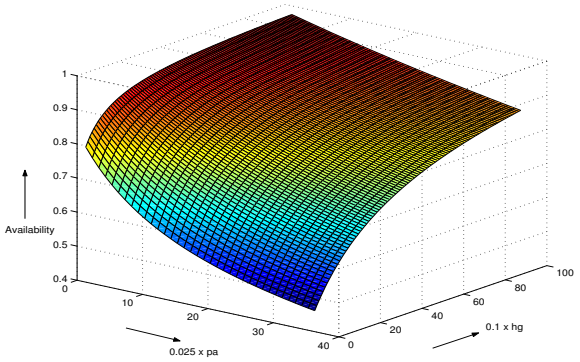


Figure 6. Availability as a function of p_a and h_G

6. Conclusions and Future Work

In this paper we have presented an approach for quantitative assessment of security attributes for an intrusion tolerant system. A state transition model that describes the dynamic behavior of such a system is used as a basis for developing a stochastic model. This is a generic model that enables the study of a variety of intrusion tolerance strategies as well as assess different impacts of a security attack. Since the memoryless property of exponential distribution implies the absence of aging and learning, it does not seem appropriate for modeling attacker behavior. In this paper, we have identified several general probability distribution functions that can be used to describe the attacker behavior and solved the semi-Markov process for several security related attributes. These include the steady-state availability and the mean time to security failure. Also, by differentiating between various absorbing states, we have computed the probability of security failure due to violations of different security attributes. The model analysis is illustrated in a numerical example.

One of the goals of our future work is to design and conduct experiments based on the recent experiences of the DARPA Information Assurance (IA) program [16]. These experiments should provide us with a better understanding of the behavior exhibited by attackers, help us to refine its stochastic description and lead to better estimates of the model parameters. As a part of the ongoing SITAR project, we plan to conduct semi-automated and automated experiments. Putting a human attacker team (Red Team) against a set of system's autonomic defenses is an example of semi-automated experiment. Fault injection [24], the well–known technique for testing and validating fault tolerant systems, is one of the techniques that provide a capability of automating the experimentation. Instead, we have sought to address the issue of the absence of exact values of model parameters by studying the sensitivity of different security attributes to small changes in the parameter values.

Another goal of our future research is to consider quality attributes such as performance, performability, and survivability in addition to the security attributes studied in this paper. The analysis of multiple quality attributes and their tradeoffs will yield insights into system's strengths and weaknesses and provide basis for carrying out cost/benefit analysis.

References

[1] Bugtraq archive. *http://www.securityfocus.com*.

[2] Organically assured and survivable information systems. *http://www.tolerantsystems.org. DARPA*.

[3] Sun microsystems secutity bulletin. *http://online.securityfocus.com/advisories/211*.

[4] In *Validation Framework Workshop discussions, DARPA OASIS PIs Mtg., Hilton Head, SC, March 11-15*, 2002.

[5] K. Arnold, B. Osullivan, R. W. Scheifler, J. Waldo, A. Wollrath, and B. O'Sullivan. *The Jini(TM) Specification (The Jini(TM) Technology Series*. Addison-Wesley, 1999.

[6] U. Bhat. *Elements of Stochastic Processes*. 2Ed., John Wiley, New York, 1984.
[7] D. Cox and H. Miller. *The Theory of Stochastic Processes*. Chapman and Hall, 1990.
[8] J. Dobson, J. Laprie, and B. Randell. Predictably dependable computing systems. *Bulletin of the European Association for Theoretical Computer Science*, 1990.
[9] R. J. Ellison et al. Survivability: Protecting your critical systems. *IEEE Internet Computing*, 1999.
[10] K. Goševa-Popstojanova, F. Wang, R. Wang, F. Gong, K. Vaidyanathan, K. Trivedi, and B. Muthusamy. Characterizing intrusion tolerant systems using a state transition model. In *DARPA Information Survivability Conference and Exposition (DISCEX II)*, volume 2, pages 211–221, 2001.
[11] S. Jha, O.Sheyner, and J. Wing. Minimization and reliability analysis of attack graphs. Technical report, CMU Tech. Report, CMU-CS-2-109, May, 2002.
[12] E. Jonsson and T. Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *IEEE Trans. Software Eng.*, 23(4):235, April 1997.
[13] J. C. Laprie. Dependability of computer systems: Concepts, limits, improvements. In *Proc. of the ISSRE-95*, pages 2–11, 1995.
[14] P. A. Lee and T. Anderson. *Fault Tolerance: Principles and Practice*. Springer Verlag, 1990.
[15] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, and D. Wright. Towards operational measures of computer security. *Journal of Computer Security*, 2:211–229, 1993.
[16] J. Lowry. An Initial Foray into Understanding Adversary Planning and Courses of Action. In *DARPA Information Survivability Conference and Exposition (DISCEX II)*, volume 1, pages 123–133, 2001.
[17] J. Lowry and K. Theriault. Experimentation in the IA Program. In *DARPA Information Survivability Conference and Exposition (DISCEX II)*, volume 1, pages 134–140, 2001.
[18] J. Medhi. *Stochastic Processes*. Wiley Eastern, New Delhi, 1994.
[19] R. Ortalo et al. Experiments with quantitative evaluation tools for monitoring operational security. *IEEE Trans. Software Eng.*, 25(5):633–650, Sept/Oct 1999.
[20] P.Mellor, S.Page, and D.Wright. Code red worm. *http://www.sarc.com/avcenter/venc/data /codered.worm.html*, 2001.
[21] R. Schantz, F. Webber, P. Pal, J. Loyall, and D. C. Schmidt. Protecting applications against malice with adaptive middleware. In *17th IFIP World Computer Congress, Montreal, Canada*, Aug. 2002.
[22] L. Swiler, C. Phillips, and T. Gaylor. A graph-based netwrok vulnerability analysis system. Technical report, SANDIA Report, SAND97-3010/1, Jan, 1998.
[23] K. S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications (2nd ed.),*. John Wiley & Sons, 2001.
[24] J. M. Voas and A. K. Ghosh. Software fault injection for survivability. In *DARPA Information Survivability Conference and Exposition (DISCEX'00)*, volume 2, pages 338–346, 2000.
[25] F. Wang, F. Gong, C. Sargor, K. Goseva-Popstojanova, K. Trivedi, and F. Jou. SITAR: A scalable intrusion-tolerant architecture for distributed services. *Proc. of 2nd Annual IEEE Systems, Man, and Cybernetics Informations Assurance Workshop, West Point, NY*, June 2001.