Security Quantification Method for Intrusion Tolerance Systems Based on Multi-recovery

Jian-Hua Huang, Liang-Jie Chen, Fan-Chao Li and Ze Fang

(Corresponding author: Liang-Jie Chen)

School of Information Science and Engineering, East China University of Science and Technology

No. 130 Meilong Road, Xuhui District, Shanghai, China

(Email: chenxifan1992@sina.cn)

(Received July 27, 2017; revised and accepted Nov. 22, 2017)

Abstract

Nowadays, virtualization has been widely used in the design of application systems. In this paper, we outline the characteristics of an intrusion tolerance system based on virtualization. The semi-Markov process of discrete time is used to model the system. The security of the system is analyzed and evaluated quantitatively from a number of perspectives such as time and space. Some new indicators are proposed to evaluate the security performance of the intrusion tolerance system more comprehensively and appropriately. We present the methods how to calculate the indicators. The simulation results showed that our methods are more accurate and comprehensive than the previous methods in describing the performance of such intrusion tolerance systems.

Keywords: Indicators; Intrusion Tolerance; Quantitative Analysis; Virtualization

1 Introduction

Intrusion tolerance [1, 5] is the popular third-generation network security technology. It assumes that the mission can be completed within a limited time even if there are intrusions in a system. The confidentiality, integrity, and availability of the system are still ensured in case attacks, failures and accidents have occurred. Classic intrusion tolerance models use the Byzantine fault tolerance (BFT) protocol to ensure system security. The BFT protocol requires 3f+1 replicas to ensure the continuality of a service. If attackers have enough time to attack the system, the redundant replicas can always be broken one by one to lead the failure of the intrusion tolerance mechanism. To address the problem, an approach called proactive recovery is used to recover replicas to their pristine states in order to block intrusions. Self-cleaning intrusion tolerance (SCIT) [9] is one of such approaches. In SCIT, the online replica providing a service is periodically forced offline to clean up malwares. A backup replica will go online to provide the service. The offline replica will be cleaned and recovered to a pristine state. Some solutions added intrusion detection mechanism to the system to trigger the reaction recovery. The online replica is immediately forced offline if a malicious intrusion is detected. The combination of the two recovery ways improves the security of self-cleaning intrusion tolerance. With the development of virtualization technology, a replica can be replaced by one instance of virtual machine images. The virtualization technology can quickly generate an instance for accelerating the recovery process and easily provide more replicas. Furthermore, providing several virtual machines in a physical server can reduce the implementation cost for intrusion tolerance and provides better scalability.

The main work of this paper is to analyze the security performance of an improved self-cleaning intrusion tolerance system based on virtualization. The system combines proactive recovery with reactive recovery. But the trigger conditions and recovery strategies of the two recovery methods are quite different. In order to refine the evaluation indicators, we propose MTTPR and MTTRR which correspond to the two recovery mechanisms respectively. Rotation recovery requires to provide backup replicas. We introduce the system virtual machine threshold to ensure the number of required replicas for system services. Instead of the single server level, we calculate the maximum tolerance time at the system level to evaluate the intrusion tolerance ability of the entire system. These security analysis parameters and their evaluation methods aim at evaluating the self-cleaning intrusion tolerance systems based on virtualization from new perspectives.

The rest of the paper is organized as follows. Section 2 briefly describes the relevant work. In Section 3, the semi-Markov model construction is described in detail. Section 4 introduces quantitative evaluation methods. Section 5 analyzes the experimental results. Finally, the summary is concluded in Section 6.

2 Related Work

Since Gong [21] proposed an extended intrusion tolerance distributed service model SITAR, intrusion tolerance gradually entered the people's vision and has been widely concerned. Various types of intrusion tolerance architecture have mushroomed. In 2003, Singh proposed the multi-replicas system of intrusion tolerance [19]. In 2006, Sord proposed the self-cleaning intrusion tolerance model (SCIT) [9]. In 2015, Zhang *et al.* designed an intrusion tolerance architecture for SCADA systems [22]. Nowadays, the intrusion tolerance architectures based on virtualization, such as SOA service architecture [13] and cloud-based SCIT model [14], have been widely used. Marco *et al.* proposed a multiple-replicas system which uses the Byzantine agreement to provide a theoretical basis for SCIT cluster security [18]. Iman *et al.* used the rotation model to serve cloud data centers citeMir2015Security. Georges [17] proposed an attack tolerance model serving for Web applications. Basing on the work, he further designed an intrusion detection and attack tolerance approach called CLARUS for cloud environments [16]. Syrine and Habib used intrusion tolerance in a cloud of databases environment [2, 3].

It is very important to analyze the security performance of intrusion tolerance systems before they are put into use. In order to describe the performance of an intrusion tolerance system better, F. Gong proposed a generic state transition model [6]. He divides the behavior of the system into nine different states. In 2004, Madan [11] *et al.* considered the relationship between the existence time of vulnerabilities and the probability of state transition, and proposed a semi-Markov process (SMP) model based on state transition matrix. They proposed the mean time to security failure as the security measure for evaluating the system. SMP based on state transition can more effectively describe the security attributes of intrusion tolerance systems than previous models. The presentation of MTTSF is of great significance in the security analysis of intrusion tolerance systems, which provides a basis for many researches. Inspired by the SITAR model and the SMP method, many scholars have also proposed their own methods. For example, Yang [8] proposed an evaluation approach based on attack behavior model and introduced the tolerability as a new quantitative indicator. Mir [12] proposed the combination of preventive recovery and the existing SCIT to improve the security of the system. He simulated the behavior of the system through the semi-Markov process, and quantified the cost of a single system recovery. Che [4] proposed a security quantitative analysis method for access control based on security entropy.

The shortcoming of state transition and SMP models is that the state transition probability and the mean sojourn time cannot be accurately determined. To address this problem, Sord used SMP to model SCIT [15] and introduced the concept of exposure window time for more accurate quantitative analysis. Based on this work, Gan [7] proposed a method for calculating evaluation parameters more accurately. This method considered the relationship among the exposure window time, attack success rate and the shielding rate. Luo [10] used SMP to analyse the intrusion tolerance capacity of systems. He proposed a kind of SMP model parameters algorithm. Tanha [20] designed a self-healing control center of critical infrastructures and analysed its security by using SMP.

Although the intrusion tolerance architectures based on virtualization have been well developed, the security evaluation on these architectures still uses several traditional methods. They only consider the states of a single virtual machine, but ignore the effect of recovery mechanism, multi-virtual machines, cluster and other characteristics. In this paper, some new security analysis parameters and evaluation methods are proposed to improve these shortcomings.

3 Intrusion Tolerance Architecture Based on Virtualization

3.1 Architecture

Virtualization technology can simplify software reconfiguration and allow multiple operating systems to run on the same physical platform at the same time. Applications can run in a separate space without affecting each other. The Low-cost and on-demand virtualization technology provides a good distributed environment for self-cleaning intrusion tolerance. The combination of proactive and reactive recovery ways improves the security of the system. Figure 1 shows a virtualization-based self-cleaning intrusion tolerance system combined with proactive and reactive recovery.

In Figure 1, each service implements its intrusion tolerance through self-cleaning mechanism. The Online VM is the primary replica which connects with the outside world via the network. It accepts requests from users and provides the appropriate services. In the architecture, there are two different kinds of recovery methods: proactive recovery and reactive recovery. The proactive recovery will make the online primary replica go offline at regular intervals. It will be replaced by a clean virtual machine to continue to provide services. When the intrusion detection system (IDS) detects a malicious behavior, the reactive recovery will be triggered to force the primary online replica into self-cleaning ahead of time. The interval time that the online replica provides services is called the online time To. If some requests have not yet been completed when the online VM is offline, the online VM has to go into the grace period to process the requests, but not accept any new requests. This period is called the grace time Tg. To and Tg together constitute the exposure time window. If a virtual machine is exposed to the network for a long time, it will be vulnerable to attacks. The virtual machine will go into self-cleaning after the grace period. Self-cleaning include deleting the malware and putting a patch. After the self-clean process,

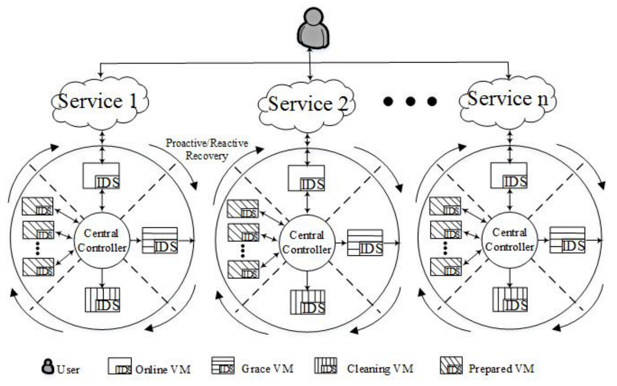

Figure 1: Intrusion tolerance architecture based on virtualization

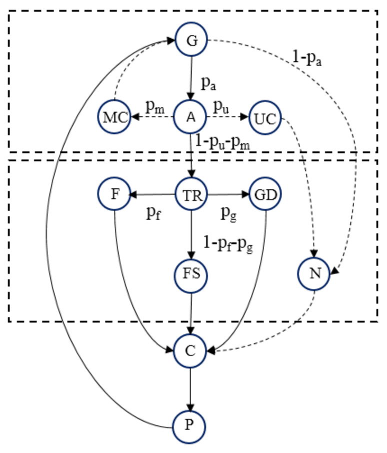

Figure 2: SMP model

the virtual machine is recovered to the initial clean and healthy state and the machine is ready for the next online service. There are several prepared VMs which are ready to go online. The combination of the two recovery methods can not only reduce the possibility of intrusions, but also will respond immediately after an attack. The mechanism improves the overall security of the system.

3.2 Modeling

The state transition model can clearly describe dynamic behaviors and state transition of a system, but it lacks the description of the state transition probability. The Markov process not only describes the states, but also describes the transition probability between adjacent states. Because the sojourn time in each state is associated with the malicious attack frequency, exposure window time, network environment and many other factors, the duration staying in each state is random and not exponential. It is appropriate to use the semi-Markov process of discrete time to describe this kind of self-clean intrusion tolerance system.

Based on the state transition model, this paper presents a semi-Markov state transition model, as shown in Figure 2. This model is used to describe the transition of the various states and the relationships between them. It is also used as a basic framework for quantitative analysis of security.

The states of the analyzed system include good state (G), attack state (A), masked compromised state (MC), normal recovery state (N), undetected compromised state (UC), triage state (TR), failed state (F), graceful degradation state (GD), fail-secure state (FS), self-cleaning (C), prepared state (P). The system is in good state (G) when it starts working. If no attack exists, it will enter the normal recovery state (N) which belongs to the grace period after the online time expires. When the system is attacked, it goes into the attack state A and then the intrusion detection mechanism will go into effect. If the detection mechanism cannot find the attack (minimum probability), the system will go into the undetected compromised state (UC). And it will go into the state N after the online time expires. When the attack is detected, if the system can shield damage through some preventive mechanisms (such as firewalls, repair vulnerabilities, etc.) to make the system unaffected, the system will enter the state MC. Then it will go into good state (G) after damage is completely shielded. Otherwise the system initiates the redundancy mechanism and enters the triage state (TR). Depending on the actual situation and the security policy, the system may choose to only keep the critical service and enter the graceful degradation state (GD). Or the system will stop all services into the fail-secure state (FS) to protect the data confidentiality and reliability. If the system has been significantly damaged and its confidentiality and reliability have failed, then it will enter the failed state (F). Finally, the offline virtual machine will start self-cleaning (state C). The system will enter the prepared state (P) after recovering to a secure and clean state.

States G, A, MC and UC together form the active period in the life cycle of a virtual machine. States TR, F, GD, FS and N form the grace period. States C and P correspond to the self-cleaning and prepared period respectively. The state transition diagram describes the complete life cycle of the virtual machines in detail. It lays a solid foundation for the numerical analysis and evaluation.

4 Quantitative Analysis

4.1 Availability Analysis

Based on the method in [11], system availability can be calculated.

Let $\{X(t) : t \geq 0\}$ be the underlying stochastic process with a discrete state space $X_s = \{G, A, MC, UC, TR, GD, F, FS, N, C, P\}$. To analyze a SMP, we need to determine a sets of parameters:

1) Mean sojourn time h_i in state $i \epsilon X_s$;

2) The transition probabilities p_i between different states $i\epsilon X_s$.

For computing the availability measure, we first need to compute the steady-state probabilities $\{\pi_i, i\epsilon X_s\}$ of the SMP states. π's in turn can be computed in terms of the embedded discrete time Markov chain (DTMC) steady-state probabilities v_i's and the mean sojourn times h_i's:

$$\pi_i = \frac{v_i h_i}{\sum\limits_j v_j h_j}, i, j\epsilon X_s \tag{1}$$

The DTMC steady-state probabilities v_i's can be computed as:

$$\begin{aligned} \bar{v} &= \bar{v} \cdot Q, v\epsilon\{v_G, v_A, v_{MC}, v_{UC}, \\ &\quad v_{TR}, v_{FS}, v_{GD}, v_F, v_N, v_C, v_P\} \end{aligned} \tag{2}$$

Q is the DTMC transition probability matrix which can be written as:

$$Q = \begin{array}{c} \\ G \\ A \\ MC \\ UC \\ TR \\ GD \\ F \\ FS \\ N \\ C \\ P \end{array} \left[\begin{array}{ccccccccccc} G & A & MC & UC & TR & GD & F & FS & N & C & P \\ 0 & p_a & 0 & 0 & 0 & 0 & 0 & 0 & \tilde{p_a} & 0 & 0 \\ 0 & 0 & p_m & p_u & \tilde{p_{mu}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p_g & p_f & \tilde{p_{gf}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \tag{3}$$

Where $\tilde{p_a} = 1 - p_a, \tilde{p_{mu}} = 1 - p_m - p_u$ and $\tilde{p_{fg}} = 1 - p_f - p_g$. In addition,

$$\sum_i v_i = 1, i\epsilon X_s \tag{4}$$

The transition probability matrix Q describes the DTMC state transition probabilities between the DTMC states as shown in Figure 2. Rewriting Equation (2) into the elemental form yields relationship between DTMC steady-state probabilities as:

$$\begin{aligned}
&v_G = v_{MP} + v_P, v_A = v_G p_a, \\
&v_{MC} = v_A p_m = v_G p_a p_m, \\
&v_{UC} = v_A p_u = v_G p_a p_u, \\
&v_{TR} = v_A(1 - p_m - p_u) = v_G p_a(1 - p_m - p_u) \\
&v_{GD} = v_{TR} p_g, v_F = v_{TR} p_f, \\
&v_{FS} = v_{TR}(1 - p_g - p_f), \\
&v_N = v_G(1 - p_a) + v_{UC} = v_G(1 + p_a(p_u - 1)), \\
&v_C = v_P = v_G D + v_F + v_{FS} + v_N = v_G(1 - p_a p_m).
\end{aligned} \tag{5}$$

Solving the above equations, in conjunction with the total probability relationship given by Equation (4) we obtain:

$$v_G = \frac{1}{2 + p_a(2 - 3p_m)} \tag{6}$$

Substituting Equation (5) into Equation (6) will yield the expressions for the remaining v's. The states $\{G, A, MC, UC, TR, GD, F, FS, N, C, P\}$ are assumed to have mean sojourn times $\{h_G, h_A, h_{MC}, h_{UC}, h_{TR}, h_{GD}, h_F, h_{FS}, h_N, h_C, h_P\}$, respectively. The SMP steady-state probabilities π's can now be easily computed by using Equation (1). In the case of calculating the steady-state probabilities, the availability of the system can be easily obtained:

$$Availability = 1 - \pi_F - \pi_{FS} - \pi_{GD} \tag{7}$$

4.2 Mean Time to Proactive Recovery and Mean Time to Reactive Recovery

The system presented in this paper has two recovery methods: proactive recovery and reactive recovery. [14] evaluated the recovery-based self-cleaning system by calculating MTTSF. In this paper, the security of the proposed system can be evaluated by calculating the mean time of two recovery methods. We refine the evaluation indicator by calculating mean time to proactive recovery (MTTPR) and mean time to reactive recovery (MTTRR).

Set the state space of proactive recovery method $X_p = \{G, A, MC, UC, N\}$ and the state space of reactive recovery method $Xr = \{G, A, TR, F, GD, FS\}$. The proactive recovery state transition matrix PRM can be obtained:

$$PRM = \begin{array}{c} \\ G \\ A \\ MC \\ UC \\ N \end{array} \left[\begin{array}{ccccc} G & A & MC & UC & N \\ 0 & p_a & 0 & 0 & 1 - p_a \\ 0 & 0 & p_m & p_u & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right] \tag{8}$$

MTTPR can be calculated as follows:

$$MTTPR = \sum_{i\epsilon X_p} V_i h_i \tag{9}$$

The reactive recovery state transition matrix RRM can be obtained:

$$RRM = \begin{array}{c} \\ G \\ A \\ TR \\ GD \\ F \\ FS \end{array} \left[\begin{array}{cccccc} G & A & TR & GD & F & FS \\ 0 & p_a & 0 & 0 & 0 & 0 \\ 0 & 0 & \tilde{p_{mu}} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_g & p_f & \tilde{p_{gf}} \\ 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & \end{array} \right] \tag{10}$$

MTTRR can be calculated as follows:

$$MTTRR = \sum_{i\epsilon X_r} V_i h_i \tag{11}$$

Where V_i denotes the average number of times which state $i\epsilon X$ is visited before the DTMC reaches one of the absorbing states and hi is the mean sojourn time in state i. The visit count elements V_i can be obtained by solving the following equation:

$$V_i = q_i + \sum_j V_j PRM_{ji}, i, j\epsilon X_p \tag{12}$$

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, $[q_i] = [1, 0, 0, 0, 0]$.

According to the above formulas, we can calculate MTTPR and MTTRR as:

$$MTTPR = \frac{h_G + p_a h_A + p_a p_m h_{MC} + p_a p_m h_{UC} + (1-p_a)h_N}{1 - p_a p_m} \tag{13}$$

$$\begin{aligned} MTTRR &= h_G + p_a h_A + p_a(1-p_m-p_u)* \\ &(h_{TR} + p_g h_{GD} + p_f h_F + h_{FS}(1-p_g-p_f)) \end{aligned} \tag{14}$$

If proactive recovery is triggered, it means that there is no malicious attack or malicious attacks are masked before the damage occurs. The system is still in a healthy state. If reactive recovery is triggered, it shows that the system is compromised by malicious attacks and has to force the virtual machine to go offline. We can calculate the ratio of MTTPR and MTTRR:

$$R = \frac{MTTPR}{MTTRR} \tag{15}$$

The higher the ratio R is, the longer the system stays in the proactive recovery cycle. It means that the system is less vulnerable to malicious attacks and more secure. This parameter is important while evaluating the security of the system.

4.3 Virtual Machine Threshold N

Ideally, assuming that the system has an infinite number of virtual machines. Whenever an intrusion occurs, the online virtual machine will be forced to go offline to recover. There will be a clean virtual machine to replace it immediately. Therefore, the entire system can endless rotate and always guarantee services. But the reality is not nearly satisfactory. The number of offline virtual machine replicas may be limited sometimes. This paper introduces the virtual machine threshold N that meets the endless rotation of the system in a given environment.

Assuming that the deal-failure rate of virtual machines is μ, then the duration of the self-cleaning state is $T_C = 1/\mu$. It represents the time from the beginning of the virtual machine offline to the end of self-cleaning. After that the virtual machine is recovered to state P and ready to go online again.

Assuming that the total attack frequency is λ. The attack frequency which causes the system to enter the self-cleaning state is:

$$\lambda_C = \lambda p_a(1-p_u-p_m) \tag{16}$$

Then the number of the attacks that cause the system to enter the self-cleaning state in time T_C is $\lambda_C T_C$.

Assuming that attacks are independent of each other. When attacks cause the system to enter the self-cleaning state, a new virtual machine must be enabled online. The number N of virtual machines must meet:

$$\begin{aligned} N &\geq \lambda_C T_C + 1 \\ &\geq \frac{\lambda p_a(1-p_u-p_m) + \mu}{\mu} \end{aligned} \tag{17}$$

According to literature [15], the online time T_o has a great impact on p_a. The probability distribution of these attack behaviors satisfies the Poisson distribution. Therefore, the attack behaviors and the response of the system are determined as a Poisson process N(t) with a rate of λ. The probability being attacked by k times per unit time is

$$P(X = k) = (\lambda^k/k)e^{-\lambda} \tag{18}$$

The random variable Y denotes the interval between two attacks. It obeys the exponential distribution: $f(t) = \lambda e^{-\lambda t}$. In time t, we get the probability $P(Y \leq t) = 1 - e^{-\lambda t}$. When time t is equal to the online timT_o, $P(Y \leq T_o) = 1 - e^{-\lambda T_o}$. $P(Y \leq T_o)$ indicates the probability that the time interval between two attacks is less than the online time. When the attack frequency is λ, the probability pa that the system is compromised can be replaced by $P(Y \leq T_o)$. Equation (17) can be written as:

$$N \geq \frac{\lambda(1-e^{-\lambda T_o})(1-p_u-p_m) + \mu}{\mu} \tag{19}$$

So the ability of systems resisting intrusion in a particular environment can be evaluated by calculating the threshold value N and comparing with the actual number of virtual machines. If the number of virtual machines which the system provides for each service is greater than or equal to N, the system can resist the intrusion for a long time through the rotation mechanism. On the other hand, the smaller the N is, the stronger the resistibility of the system is and the less the resource required is.

4.4 Maximum Tolerance Time T

Assuming that the system does not meet the threshold N. It means that the system cannot provide enough virtual machine resources for rotation. All the virtual machines will enter the self-cleaning state. The system will not be able to continue to guarantee the services and go into the crash state. At this moment the system has to stop the services until the failed virtual machines are recovered. So there exists a maximum tolerance time T. It represents the total time that a system can provide for regular rotation. We can evaluate the persistence of a system against intrusion by the maximum tolerance time T.

When the number of actual virtual machines $n \geq N$ and $T = \infty$, the system can rotate continuously. When $n < N$ and $\lambda_C T \leq n - 1$, the system will not crash. It means that the number of attacks that cause the system to enter the self-cleaning state is less than the total number of virtual machines. The maximum tolerance time T can be calculated:

$$\begin{aligned} T &\leq \frac{n-1}{\lambda_C} \\ &\leq \frac{n-1}{\lambda p_a(1-p_u-p_m)} \end{aligned} \quad (20)$$

When the number of actual virtual machines cannot reach the threshold, the constancy of the system to resist intrusion can be evaluated by calculating the maximum tolerance time T. The greater the T is, the stronger the ability for the system to resist the intrusion is.

5 Experimental Evaluation

Now we analyze a typical system with rotation recovery architecture. In this section we illustrate the evaluation of the security attributes through numerical examples. In order to reflect the relationships among the quantification parameters, we use the MATLAB software in the simulation experiment. The parameters involved include the state transition probabilities $p_i, i\epsilon\{A, MC, UC, GD, F\}$, the average sojourn time $h_j, j\epsilon\{G, A, MC, UC, TR, GD, F, FS, N, C, P\}$ and the online time T_o that can be used to express p_a.

5.1 Availability Analysis

Figure 3 shows the relationship between system availability and attack success rate p_a. It compares the model availability presented in this paper with the other two models. Curve 1 represents the rotation-based composite architecture proposed in this paper. Curve 2 represents the classical SCIT [15]. Curve 3 represents the improved SCIT in [12]. The results in the figure show that the availability of the three models decreases with the increase of the attack success rate. It indicates that p_a is the main factor affecting the availability of the system.When the system is in a more secure network environment ($p_a < 0.2$), the availability of our model is more than 0.95. Even in a more malicious network environment ($p_a > 0.8$), the availability can remain at 0.65 or more. This shows that our model has a high degree of tolerance. The availability of the proposed model is overall greater than the other two and the descending speed is slower. It indicates that the ability to deal with the intrusion has been improved. Compared with the other two models, the proposed model has a richer state space. It is the major reason for its performance improvement.

5.2 MTTPR/MTTRR

As shown in Figure 3, in a typical system based on rotation recovery mechanism, the greater the attack success rate p_a is, the more easily the system is compromised. Once the system is compromised and the detection mechanism detects the intrusion, the system will force the virtual machine to be offline for self-cleaning. Proactive recovery is converted to reactive recovery. Subsequently, the corresponding MTTPR will decrease and the MTTRR will increase. It results in a decrease of the ratio R. The higher the ratio R is, the longer the system stays in the proactive recovery cycle. It means that the system is less vulnerable to malicious attacks and more secure. The slope of the curve decreases with the increase of p_a, indicating that the change rates of MTTPR and MTTRR are also decreasing. The ability for the system to tolerate high-intensity intrusion is saturated and stabilized. When $p_a = 1$, the ratio R remains above 0.6. It indicates that the system can still maintain normal rotation under high-intensity intrusion. This parameter is important while evaluating the security of the system. If p_a=0, the system will only use proactive recovery and the ratio R will tend to infinity. Figure 4 only considers the case that the system is attacked. So we set $p_a\epsilon[0.1, 1]$.

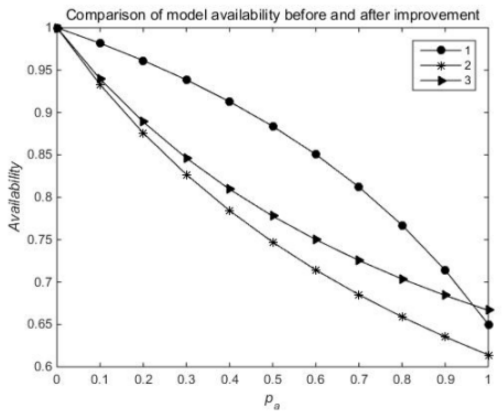

Figure 3: Comparison of model availability before and after improvement

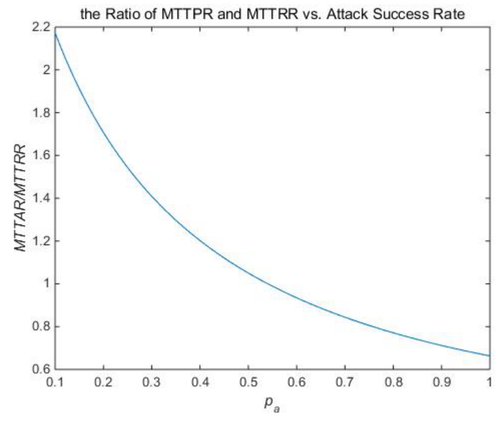

Figure 4: Ratio of MTTPR and MTTRR vs. attack success rate

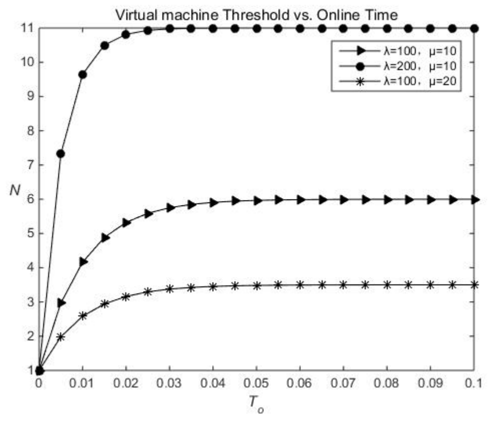

Figure 5: Virtual machine threshold vs. online time

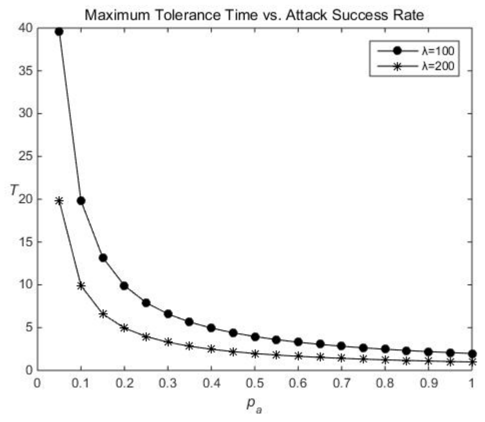

Figure 6: Maximum tolerance time vs. attack success rate

5.3 Threshold N

Figure 5 shows the relationship between the online time and the rotation virtual machines threshold. The following three cases are considered:

1) Attack frequency 100 times/unit time, recovery capacity 10 times/unit time;

2) Attack frequency 200 times/unit time, recovery capacity 10 times/unit time;

3) Attack frequency 100 times/unit time, recovery capacity 20 times/unit time.

With the increasing of online time T_o, the three curves are on the rise. The greater the threshold N is, the more virtual machines are required to ensure rotation. The requirement of virtual machine resources continues to increase. The λ of curve 2 is twice that of curve 1 and the N of curve 2 is always greater than the value of curve 1. It shows that the threshold is related to the attack frequency λ. The higher λ is, the higher the threshold is. It can be seen from the figure that the μ of curve 1 is twice that of curve 3 and the overall trend of curve 1 is greater than curve 3. It shows that the stronger the system recovery capability is, the less virtual machine resources are required. Therefore, in the application environment, the requirement of rotation virtual machines can be reduced by shortening the online time and improving the ability of system recovery.

5.4 Tolerance Time T

Figure 6 shows the relationship between maximum tolerance time and attack success frequency. The following two cases are considered:

1) Attack frequency 100 times/unit time;

2) Attack frequency 200 times/unit time.

Both curves show a downward trend, indicating that the maximum tolerance time T which the system can maintain the regular service will decrease with the attack success frequency pa increasing. The λ of curve 2 is twice that of the curve 1 and the maximum tolerance time T is one-half of curve 1, indicating that the increase of attack frequency will cause the system's tolerance time to drop. The simulation results show that the tolerance time decreases sharply with p_a when p_a is less than 0.5. Because the higher the attack frequency is, the faster the virtual machine rotates. The curve tends to be gentle when pa is greater than 0.5. Because the system's tolerance ability has reached saturation and the tolerance time has been reduced to a minimum. It means that the simulation system is less resistant to intrusion.

6 Conclusion

In view of the self-cleaning intrusion tolerance model based on virtualization, this paper builds a comprehensive SMP model which considers a variety of characteristics: the virtual machine rotation, the self-adaption, the combination of proactive and reactive recovery and etc. We add self-cleaning, prepared and normal recovery states to the model, so the architecture's unique life cycle is described in more detail. Most of the existing evaluation methods only evaluate single virtual machine behaviors and cannot be adapted to the comprehensive system. This paper focuses on modeling and evaluating the entire cluster architecture. We highlight the characteristics of virtual machine rotation from different perspectives such as time and space. The MTTPR, MTTRR, virtual machine threshold and maximum tolerance time are put forward. It enriches the methods of security evaluation of intrusion tolerance systems. The experimental results show that the improved model is more comprehensive to evaluate the security of the system than other models. The proposed indicators give us more way to evaluate the intrusion tolerance capability of the system, which provides a new way for the system security evaluation. But the evaluation process of these parameters needs to be analyzed and perfected by a lot of practice. We need to further consider the dynamic self-adaptability of virtual machines and the evaluation effect of these parameters in more complex cloud environment. These still need to be resolved in future research.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61472139 and a research Grant made to East China University of Science and Technology by Shanghai Education Commission. The authors are also grateful to the anonymous referees for their insightful and valuable comments and suggestions.

References

[1] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398–461, 2002.

[2] S. Chatti and H. Ounelli, "An intrusion tolerance scheme for a cloud of databases environment," in *International Conference on Network-Based Information Systems*, pp. 474–479, 2016.

[3] S. Chatti and H. Ounelli, "Fault tolerance in a cloud of databases environment," in *International Conference on Advanced Information Networking and Applications Workshops*, pp. 166–171, 2017.

[4] T. W. Che, J. F. Ma, N. Li, and C. Wang, "A security quantitative analysis method for access control based on security entropy," *International Journal of Network Security*, vol. 17, no. 5, pp. 517–521, 2015.

[5] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Os diversity for intrusion tolerance: Myth or reality?" in *EEE/IFIP 41st International Conference on Dependable Systems and Networks*, pp. 383–394, 2011.

[6] K. Gosevapopstojanova, K. Vaidyanathan, K. Trivedi, F. Wang, R. Wang, F. Gong, and B. Muthusamy, "Characterizing intrusion tolerant systems using a state transition model," in *DARPA Information Survivability Conference and Exposition II (DISCEX'01)*, vol. 2, pp. 211–221, 2001.

[7] J. H. Huang and H. S. Gan, "Quantitative approach to dynamic security of intrusion tolerant systems," *Journal of Computer Applications*, vol. 31, no. 1, pp. 123–126, 2011.

[8] J. H. Huang and T. Y. Yang, "A method for quantifying the security of intrusion tolerant system," in *International Symposium on Computer Network and Multimedia Technology (CNMT'09)*, pp. 1–4, 2009.

[9] Y. Huang, D. Arsenault, and A. Sood, "Incorruptible system self-cleansing for intrusion tolerance," in *IEEE International Conference on Performance, Computing, and Communications Conference (IPCCC'06)*, pp. 490–496, 2006.

[10] Z. Luo, B. You, P. Wang, J. Su, and Y. Liang, "Analysis and optimization of system intrusion tolerance capacity based on markov," *International Journal of Network Security*, vol. 19, no. 6, pp. 1036–1043, 2017.

[11] B. B. Madan, K. Vaidyanathan, and K. S. Trivedi, "A method for modeling and quantifying the security attributes of intrusion tolerant systems," *Performance Evaluation*, vol. 56, no. 1, pp. 167–186, 2004.

[12] I. E. Mir, S. K. Dong, and A. Haqiq, "Security modeling and analysis of a self-cleansing intrusion tolerance technique," in *International Conference on Information Assurance and Security*, pp. 111–117, 2016.

[13] Q. L. Nguyen and A. Sood, "Improving resilience of soa services along space-time dimensions," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, pp. 1–6, 2012.

[14] Q. L. Nguyen and A. Sood, "Designing scit architecture pattern in a cloud-based environment," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*, pp. 123–128, 2011.

[15] Q. L. Nguyen and A. Sood, "Quantitative approach to tuning of a time-based intrusion-tolerant system architecture," in *3rd Workshop Recent Advances on Intrusion-Tolerant Systems*, pp. 132–139, 2009.

[16] G. Ouffoue, A. M. Ortiz, A. R. Cavalli, W. Mallouli, J. Domingoferrer, D. Sanchez, and F. Zaidi, "Intrusion detection and attack tolerance for cloud environments: The clarus approach," in *IEEE International Conference on Distributed Computing Systems Workshops*, pp. 61–66, 2016.

[17] G. Ouffoue, F. Zaidi, A. R. Cavalli, and M. Lallali, "Model-based attack tolerance," in *31st International Conference on Advanced Information Networking and Applications Workshops (WAINA'17)*, pp. 68–73, 2017.

[18] M. Platania, D. Obenshain, T. Tantillo, R. Sharma, and Y. Amir, "Towards a practical survivable intrusion tolerant replication system," in *IEEE International Symposium on Reliable Distributed Systems*, pp. 242–252, 2014.

[19] S. Singh, M. Cukier, and W. H. Sanders, "Probabilistic validation of an intrusion-tolerant replication system," in *International Conference on Dependable Systems and Networks*, pp. 615–624, 2004.

[20] M. Tanha, F. Hashim, and S. Subramaniam, "Secure and self-healing control centers of critical infrastructures using intrusion tolerance," *International Journal of Network Security*, vol. 17, no. 4, pp. 365–382, 2015.

[21] F. Wang, F. Jou, F. Gong, C. Sargor, K. Gosevapopstojanova, and K. Trivedi, "Sitar: A scalable intrusion-tolerant architecture for distributed services," in *Proceedings of DARPA Information Survivability Conference and Exposition*, vol. 2, pp. 153–155, 2003.

[22] Y. Zhang, L. Wang, and Y. Xiang, "Power system reliability analysis with intrusion tolerance in scada

systems," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 669–683, 2016.

Biography

Jian-Hua Huang had received the B.S. and M.S. degrees from East China University of Science and Technology, Shanghai,China, and the Ph.D. degree in control theory and control engineering from East China University of Science and Technology, Shanghai,China. He has served as Associate Professor of Computer Science and Engineering at East China University of Science and Technology since 1998. His current research interests include computer networks, wireless sensor networks, information security, data mining, cloud computing, optimization and modeling. Dr. Huang has been a member of various network committees including Specialist Group of Shanghai Education and Research Network and Network Specialized Committee of Shanghai Higher Education Association. He is also the director of Development Center of Shanghai Education Network IPv6 Laboratory.

Liang-Jie Chen had received the B.Eng degree in software engineering from Fuzhou University, Fujian province, China. He is currently pursuing the M.Eng degree in computer science and technology from East China University of Science and Technology, Shanghai, China. His current research interests include intrusion tolerance systems and quantitative analysis.

Fan-Chao Li had received the B.Eng degree in computer science and technology from Nanjing Technology University, Jiangsu province, China. He is currently pursuing the M.Eng degree in computer science and technology from East China University of Science and Technology, Shanghai, China. His current research interests include intrusion tolerance systems and quantitative analysis.

Ze Fang had received the B.Eng degree from Hefei University, Anhui province, China. He is currently pursuing the M.Eng degree in computer science and technology from East China University of Science and Technology, Shanghai, China. His current research interests include intrusion tolerance systems.