IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 23, 2019, accepted January 4, 2020, date of publication January 9, 2020, date of current version January 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965090

Numerical Evaluation of Job Finish Time Under MTD Environment

ZHI CHEN, XIAOLIN CHANG, (Member, IEEE), ZHEN HAN, AND YANG YANG
Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China

Corresponding author: Xiaolin Chang (xlchang@bjtu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1836105, and in part by the Fundamental Research Funds for the Central Universities of China under Grant 2018JBZ103.

ABSTRACT Moving target defense (MTD) has recently emerged as a game-changer in the confrontation between cyberattack and defense. MTD mechanism constantly and randomly changes the system configurations to create uncertainty of the attack surface against cyber-adversaries. To date, researches on the evaluation of MTD techniques either focused on analyzing the effectiveness of MTD or studying the system performance loss due to the use of MTD. The impact on job/service running on the protected system is always ignored. In this paper, we propose an SRN (Stochastic reward net) based analytical modeling approach to investigate how MTD techniques influence the job running on protected system from the perspective of job finish time. The SRN model developed in this paper captures the behaviors of both the adversary and the job execution process. Furthermore, we carry out numerical analysis to study the impact of different system parameters on job finish time and other evaluation metrics. The results in this paper can help defenders choose a better MTD configuration to complete the job execution as soon as possible.

INDEX TERMS Moving target defense, job migration, stochastic reward net, job finish time.

I. INTRODUCTION

In cyber space, counterwork and game between adversaries and defenders never stops. Traditional defense techniques like IDS, Firewalls and Anti-virus are increasingly difficult to resist new types of attacks due to the static properties of the protected targets. Adversaries always have the time advantage and can gather enough vulnerability information of the target system before launching an attack. In order to eliminate this disadvantage, Moving Target Defense (MTD) [1] has emerged as a game changer as it provides a proactive defense strategy by creating asymmetric uncertainty in favor of defenders. To achieve this goal, MTD mechanism constantly changes the system configurations and then makes it hard for adversaries to find an vulnerable and available attack surface of the target system. In the past few years, various MTD techniques have been proposed and each of them focused on one or more aspects of system parameters. Meanwhile, there were also various researches studying how effective a MTD mechanism is from the perspective of protecting the targeted system. Notice that, while introducing a variable and unpredictable attack surface to the system, MTD technique also brings additional computational overhead and reduces the system performance. The job running on the MTD protected system will also be affected. Therefore, it is valuable to study the impact of MTD techniques on the job execution process. However, existing researches on the evaluation of MTD techniques either focused on analyzing the effectiveness of the MTD to bring security to the protected system or studying the system performance loss caused by MTD mechanism. None of them investigated the impact of MTD technique on the job executed on the protected system.

In this paper, we present a quantitative analytical modeling approach to analyze how the MTD mechanism affects the job execution process. Due to the existence of attack and MTD, the actual whole job execution time will be longer than needed. We use this time as a metric to investigate the job performance and its security while using MTD technique. We use Migration-based Dynamic Platform (MDP) as an example in this paper to illustrate our modeling approach. Specifically, MDP is a kind of MTD which dynamically changes the properties of a computing platform in order to complicate the attacks [2]. The computing platform here can be hardware or different OS attributes. Jobs or services can be constantly migrated among these platforms to reduce the probability of being destroyed by attacks. In our paper,

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

we consider a job which needs to run for a certain period of time. To protect the job execution process and finish the execution as soon as possible, multiple virtual machines (VMs) are prepared for running the job and the whole execution process is divided into multiple stages. At each stage, the system will randomly choose one VM to process the execution progress and then migrate to next VM to continue. The adversary's goal in our paper is to constantly compromise a VM and destroy the execution process if the job is found.

Our work aims at evaluating the performance and the security impact caused by MTD techniques. The analytical model proposed in this paper is based on Stochastic Rewards Nets (SRN) [3]. Our model captures the typical behaviors of each party in an MTD protected system.

The major contributions of our paper are summarized as follows:

(1) Our work in this paper, for the first time, evaluates the effectiveness of MTD mechanism from the perspective of the completion time of a job, which follows preemptive-resume (PRS) discipline [4]. Namely, the job we considered in this paper can continue its execution from the end point of last stage. More details are given in Section III.

(2) We proposed an SRN-based modeling approach to capture the typical behavior of both the adversary and the job execution process in an MTD protected system. Administrators or defenders can use our model to conduct a most appropriate system configuration of the MTD system.

(3) We give the numerical results of our model to quantify the impact of each parameter in the system, also our model can be easily modified by defenders according to the actual evaluation environment.

The remainder of this paper is organized as follows: Section II reviews related works of MTD. Section III describes the system considered in this paper and the corresponding SRN sub-models. We present the numerical analysis in Section IV. Finally, Section V puts forward conclusions and states our future works.

II. RELATED WORK

According to the research content, the researches related to MTD are mainly classified into three categories, which are theory research [5]–[7] including the design principles of MTD, strategy formulation as well as game theory [8]–[11] and new strategy research to find new moving parameters, and evaluation research including the evaluation of effectiveness and performance of MTD. Here we only focus on the evaluation researches on MTD.

There are researchers evaluating the MTD with simulation experiments. Zhuang et al. firstly used NeSSi2 to build a simulation testbed of network moving target defense system in [12]. They analyzed the effectiveness of MTD by simulating continuous network attacks and observing the impact of different system parameters on the success rate of attacks. Then they presented an analytical model for the effectiveness evaluation of MTD in enterprise networks and validated the model with their testbed [13]. Similar to Zhuang's work, Zheng et al. analyzed the effectiveness of IP address mutation based MTD with OMNet++ in [14]. Their simulation results show the optimal IP address mutation rate. These simulation works are not universal as they can only be applied to their own mechanisms.

Despite those simulation experimental researches, there are many evaluation works based on mathematical models or theoretical analysis. Evans et al. proposed an effectiveness analysis approach of diversity-based MTD mechanisms in [15]. They used a network confrontation experiment method to test the defense capability of diversity-based MTD such as address space layout randomization (ASLR) or instruction set randomization. Their results show that dynamic randomization techniques are ineffective against circumvention attacks or deputy attacks but can be used to defense entropy reduction attacks in some cases. Han et al. in [16] proposed a cyber epidemic dynamics approach to quantitatively analyze the effectiveness of MTD. The algorithm they offered shows an optimal deployment of MTD mechanism with the minimum cost. Carter et al. in [17] applied game theory and statistical analysis approach to study the deployment strategy of dynamic platform based MTD. They demonstrated that uniform deployment strategies are near optimal under some threat environment they considered. In addition, they developed a quantitative evaluation method for Lightweight Portable Security (LPS) and IP Hopping based MTD in [18]. Authors in [19] introduced a continuous time modeling approach which can help to perform dynamic platform based MTD while maintaining high network availability. Leeuwen *et al.* [20] developed an experimental technique to assess the effectiveness of network-based MTD. They quantified system attributes like weakness, protection and threat to measure the defense effect in a real experimental environment.

Stochastic reward net (SRN) is a form of Petri net where each transition is associated with a probability distribution function for the firing time [21]. SRN has been adopted to quantitative analysis of reliability, availability and performance in many areas [22]–[24], including the evaluation of MTD mechanisms. Torquato and Vieira [25] proposed a set of SRN models to evaluate the MTD in cloud environment. They analyzed the availability of MTD under different system parameters such as software age rate and different workloads. Cai *et al.* [26] also used SRN models for evaluating the effectiveness of MTD. However, those works only focused on the effectiveness of MTD but ignored the job performance in the protected system. Our work in this paper considers the job performance from the perspective of job finish time.

Connell et al. proposed a Markov chain analytic model to quantitatively analyze the resource availability and performance of MTD in [27]. Furthermore, in [28] they expanded their work to allow their models supporting the limitation of resources being reconfigured simultaneously. Different from our work, their models can only capture the behavior of the job and the computing resources under the MTD environment, but do not consider the impact or the attack on the

TABLE 1. System parameter definition and default value.

Variable	Definition	Default
N	Number of PMs	3
M	Number of phases	10
$1/\alpha$	Mean delay between the attack and the job starting being processed	1 day
$1/\beta$	Mean time to attack a VM	1 days
$1/\gamma$	Mean time to migrate the Job to another VM	1/48 days
$1/\lambda$	Mean time to execute the job in a VM (at a stage)	2 days
$1/\mu$	Mean time the attacker waits at a VM	30 minutes
A	Maximum job execution time threshold in SLA	100 days
B	Defender's benefits each day before the threshold	10 dollars/day
C	Defender's cost for each VM	1.429 dollars/hour
t_ask	Job execution time without attack	20days
T	Job finish time under attack and MTD	

system. However, our SRN models not only capture the job execution behavior on multiple VMs but also capture the attack's behaviors in the system.

III. SYSTEM AND MODEL DESCRIPTION

In this section, we first describe the system scenarios including the job execution behavior and the adversary's behavior. Then the stochastic models developed for the system and the evaluation metrics used in this paper are exhibited. The system parameters and their default value used in this paper are defined in LABLE1. The default values of attack-related parameters are from [29] and the others are set to show the effectiveness of MTD. All parameters in our model can be easily modified by defenders according to the actual evaluation environment.

A. SYSTEM DESCRIPTION

In this paper, we consider a scenario that a critical job needs to be executed for a certain period of time, we use t_ask to indicate this time. However, given the existence of adversaries and attacks, the actual completion time T is longer than the required time. In order to eliminate security concerns and finish the job execution as soon as possible, the administrator adopts MTD mechanism to protect the execution process. Here we take dynamic platform techniques (DPTs) based MTD as example which is by altering the computing platform to complicate attacks [30].

Specific to our system, there are multiple (N) virtual machines (VMs) in distributed cloud data center prepared for the critical job execution. They can be the same system environment or different system environments. Each VM has one or several vulnerabilities and can be exploited by adversaries. In order to resist the attacks, the execution is divided into multiple (M) stages. At each stage, the MTD system randomly chooses one VM to perform the job execution progress. There are only two cases where a new VM needs to be chosen. One is that the job at the current stage has been successfully executed. The other is that the job is disrupted due to attacks during the execution at current stage. Note that in either case, the selection of the VM has a memory, which means that the job will not be executed on the same VM whether it is starting the next stage or re-executing the current stage.

Turning now to the adversary's behaviors. We consider a determined and strong single adversary whose goal is to disrupt the job execution progress. As we said previously, there are N VMs that can be used to execute the target job. At any time, only one VM runs the job. We assume that the adversary cannot distinguish whether the VM is currently running the job execute progress before he successfully attacking the VM and getting the privileges. Another assumption is that the adversary can exploit all vulnerability to gain access to all VMs, but he can only attack one VM at a time, i.e., there is no parallel attacks. Based on these assumptions, typical attacking behaviors can be summarized as following steps: First, the adversary randomly selects a VM to attack. After the attack is successful and getting the access to the VM, the adversary determines whether the target job is running on this VM. If so, the adversary will destroy the job progress and reselects the next target VM immediately. If not, the adversary waits for a while to see if a job arrives. Noted that this waiting period has an upper limit $Tmax = 1/\mu$. Once this limit is reached, the adversary will abandon the current VM and return to do the VM selection progress. Different from VM selection for the job, the adversary's choice of the target VM is memoryless. That is, the adversary may select the same VM at the subsequent attacking phase. Note that the purpose of choosing this VM selection strategy is just to show that our model described next can be easily adapted to other VM selection methods. In other words, the job execution VM selection can be memoryless and the adversary VM selection can also have memory. Administrators of defenders can easily customize the evaluation model according to their needs.

According to the system description above, we give the evaluation metrics of interest in the paper as follows:

Metric 1: Mean completion time of job execution

Metric 2: Job fail times (The number of successful attacks to the target job)

Metric 3: Defender's profit for finishing the job execution.

B. STOCHASTIC MODEL DESCRIPTION

In this section, we present the SRN model, including three SRN sub-models, which are developed for the system described in the previous section. In order to understand the logic of our model easily, we take $N = 3$ as an example, as shown in the solid line parts in the figures. In other words, there are three VMs available for each job. The dotted line

parts in the figures are only used to indicate that our model supports large-scale evaluation, i.e. large N.

1) ATTACK SUB-MODEL

Figure1 illustrates Attack SRN sub-model. As we said in section III.A, the adversary's attacking behavior is memoryless, that is, the adversary may repeatedly select the last VM to attack. The guard functions of attack SRN sub-model are summarized in Table 2. The halting function *myhalt*() is used to determine whether the sub-model has completed the evaluation progress.

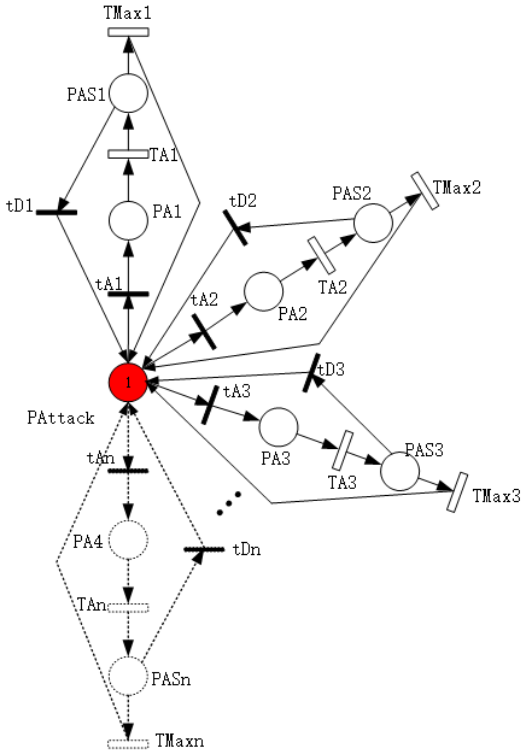

FIGURE 1. Attack SRN sub-model under MTD environment.

TABLE 2. Guard function definition in attack srn sub-model.

Function name	Function definition
guardtAi	if (myhalt()==1) return 1;else return 0;
guardtDi	if (#"PJi"==1 || #"PJi0"==1 || #"PJi1"==1 && myhalt()==1) return 1; else return 0;

In Attack SRN sub-model, place PAttack initially has a token, representing that the adversary started to select target VM. Place PAttack has three output arcs pointing to immediate transitions tA1, tA2 and tA3, respectively. The three immediate transitions have the same priority in our sub-model, indicating that they have the same probability of being fired. Once one of them is fired, the token in Pattack will be transferred to the corresponding output place. We use adversary to choose attack VM1 as an example, i.e. tA1 will be fired and the token from PAttack will be deposited to PA1. When transition TA1 fires, representing that the adversary has successfully attacked VM1 and the token from PA1 is deposited to PS1. There are two input arcs from PS1, one input to an immediate transition tD1, and another one input to a timed transition TMax1. Note that, as soon as the adversary successfully attacks VM1 and gets the required privileges, he will check to see if the job is executing here. In our sub-model, we use guard function guardtD1() to represent this checking operation. The return value of guardtD1() depends on the state of the Job Migration SRN sub-model which will be explained later. If the job was running on the VM1, function guardtD1() will return 1, and the transition tD1 fires. The token from PS1 to PAttack represents that the adversary has reached his target and destroyed the execution of the job under the current VM and proceed to the next round of attack. Otherwise, if the job is not running on VM1, function guardtD1() will return 0 and tD1 will not be fired, the token will stay at place PS1 indicating that the adversary stays at VM1 waiting for his target job. It is impossible for the adversary to wait for VM1 all the time. We use timed transition TMax1 to indicate the maximum waiting time. Once Tmax1 fires, the token in PS1 will also return to PAttack, representing that the adversary can not wait for the job to be migrated to VM1 and can only go back and carry out to start the next round of attack. Here we take VM1 as an example. The adversary attacks VM2 or VM3 in the same way.

2) JOB MIGRATION SUB-MODEL

Job Migration SRN sub-model captures the migration behaviors of job across heterogeneous VMs, illustrated in Figure2. The guard functions of job migration sub-model are summarized in Table3.

TABLE 3. Guard function definition in job migration srn sub-model.

Function name	Function definition
guardti	if (myhalt()==1) return 1; else return 0;
guardtFi	if (#"PASi"==1 && myhalt()==1) return 1; else return 0;

In Job Migration SRN sub-model, place PDelay initially has a token, and an input arc pointing to time transition TDelay. From a practical point, the adversary can not recognize the starting of the job execution, so the start time of the two sub-models should be different. We use this time transition to eliminate the time difference between the start of attack and the job execution.As soon as the transition TDelay fires, the token in PDelay will be deposited to place PJob. The immediate transitions t1, t2 and t3 have the same effect as tA1, tA2 and tA3 in Attack SRN sub-model, which are used to indicate that the job randomly selects a VM to start its current stage of execution. Similarly, we use the chosen of VM1 as an example, i.e. t1 fires and the token from PJob deposited into P1. Once P1 has a token, it indicates that the job starts its execution on VM1. We use a timed transition

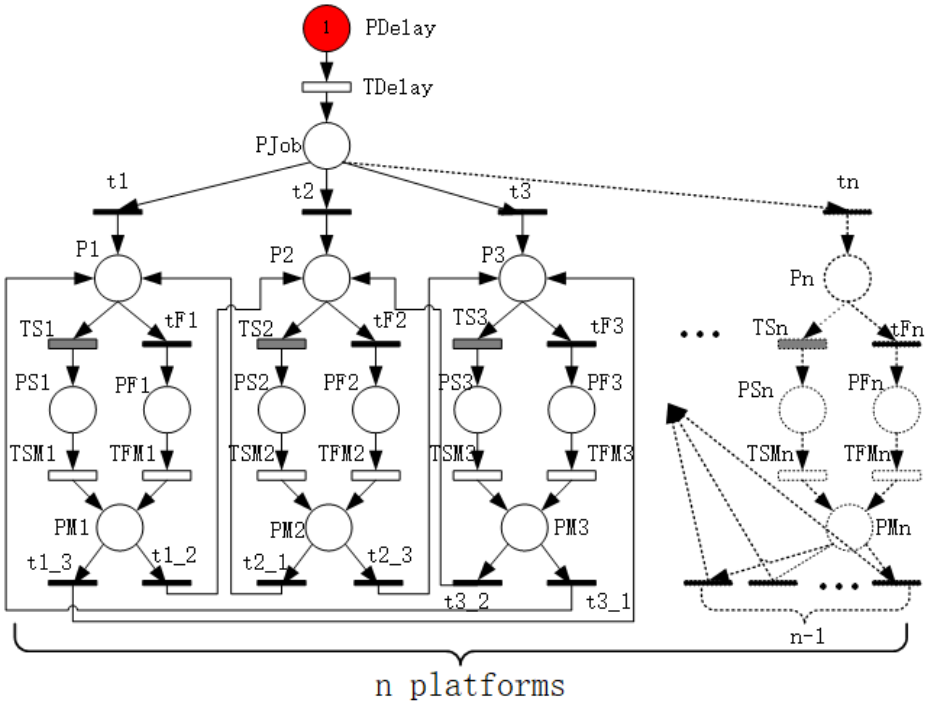

FIGURE 2. Job migration SRN sub-model under MTD environment.

TS1 to represent the complete execution time of the job on VM1. Unlike other transitions, we use gray solid rectangles to represent transition TSi, as we said before, the whole job execution was divided into multiple phases. At each phase, the job needs to be executed for a constant time, which is $1/\lambda$, see TABLE 1. Therefore, the firing time of transition TSi should be deterministic rather than exponential distributed like other timed transitions in our sub-model. After the job successfully finishes its current phase of execution on VM1, TS1 fires and the token in P1 will be deposited to PS1. It should be noted that P1 not only has an input arc to TJS1, but also an input arc to immediate transition tF1. Here, tF1 has a guard function guardtF1 to check whether if VM1 has been attacked. If the place PS1 in Adversary SRN sub-model has a token, guardtF1 returns 1, and tF1 fires, the token in PJ1 will move to place PF1. Notice that, when tF1 fires, the job may have already executing on VM1 for a while. After the VM1 is attacked the job need to be migrated to another VM to continue its execution. We use timed transition TFM1 to represent the time taken by the migration process. Also, even if the job has finished this phase of execution, i.e. PS1 has a token. The job still needs to be migrated to another VM to start its next phase of execution, so that there is a timed transition TSM1 between PS1 and PM1. Since TSM1 and TFM1 both represent the migration delay, they have the same transition rate. When TSM1 or TFM1 fires, the token in PS1 or PF1 will move to PM1, respectively. Place PM1 has two input art to transition t1_2 and t1_3 respectively. They get the same priority, indicating that the job will randomly select another VM to run next.

Before proceeding to the next sub-model description, it is important to solve a problem which will restrict the model's numeric analysis later. Since there are non-exponentially distributed transitions TSi existing in job migration sub-model which make our model a Non-Markovian SRN. This complicates the numeric analytic method to solve our model. Therefore, in this paper, we use a 3-phase Erlang sub-net to approximate the deterministic firing time of transition TSi and eliminate the restriction. We show this approximate replacement in Figure3. Transition TSi is replaced by 3 series

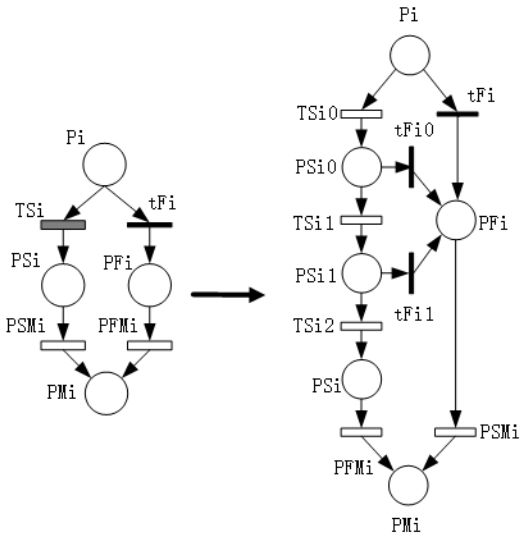

FIGURE 3. Sub-net of Job execution on each VM.

timed transition TSi0, TSi1 and TSi2. The firing time of them is $\lambda/3$, i.e. the rate of transition TSi0, TSi1 and TSi2 is $3/\lambda$. Meanwhile, the middle places PSi0 and PSi1 both indicate that the job is still executing on VMi for its current phase. Immediate transition tFi0 and tFi1 have the same guard function with tFi as the job execution process can be broken by adversaries at any time before it completes the current execution phase. i.e. the token from Pi arrives at PSi.

3) TIME CALCULATION SUB-MODEL

The time calculation SRN sub-model is used to calculate the total execution time of the job, illustrated in Figure 4. Guard functions and halting function to be used in this sub-model is defined in TABLE 4.

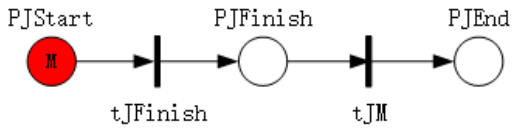

FIGURE 4. Sub-net of Job execution on each VM.

TABLE 4. Guard function definition in time calculation srn sub-model.

Function name	Function definition
guardDone	if (#"PS1"==1 || #"PS2"==1 || #"PS3"==1 … || #"PSi"==1) return 1; else return 0;
guardEnd	if (#"PM1"==1 || #"PM2"==1 || #"PM3"==1 … || #"PMi"==1) return 1; else return 0;
myhalt	if (#"PEnd "==M) return 0; else return 1;

In this sub-model. Place PJStart initially has M tokens. M indicates the total number of rounds required for the job execution. Immediate transition tJFinish has a guard function guardDone(). Function guardDone() checks the token in place PS1, PS2 and PS3, whenever one of them has a token, guardDone() returns 1 and tJFinish fires, indicating that the job has successfully finished the current phase of execution on a VM and a token will move from PJStart to PJFinish. We use immediate transition tJM to represent the delay of Job migration to the next VM after the job finishes its current phase of execution. tJM has a guard function guardEnd(). When PM1, PM2 or PM3 has a token, it means that the job has finished the migration process and started to choose its next running VM, guardEnd() returns 1 and tJM fires. The token in PJFinish will be deposited to PJEnd. Notice that when place PMi has a token, immediate transitions ti_j and tJM can all be fired at the same time, and this creates a conflict lock in our model. Once the transition ti_j fired firstly, the token in place PMi will be removed and the value of function guardEnd() will change from 1 to 0, which means the transition tJM can't be fired anymore and the token in PJFinish can only be stuck there. Therefore, we set the priority of transition tJM to 2 instead of its default value 1. This setting guarantees that all the tokens in PJStart can be moved to PJEnd one by one.

After all M tokens from PJStart arrive to PJEnd, this indicates that the job has finished all the execution and the sub-model will be halted in an absorbed state. We use a halting function myhalt() to determine this state. By calculating the MTTA, we can get the total job execution time.

C. RUNNING PROCESS OF THE SRN MODEL

To better describe our SRN model and the interactions between the sub-models, we provide pseudo code shown as Algorithm 1 which demonstrates how tokens are moved

Algorithm 1 Running Process of the Model

```
function Attack
    PAttack = 1
    while 1 do
        i = random(0, N)
        PAttack → PASi
        if Pi==1 then
            PASi → PAttack
        else
            Wait Tmax
            PASi → PAttack
        end if
    end while
end function
function Job
    PDelay = 1
    j = random(0, N)
    while 1 do
        PDelay → PJob
        PJob → Pj
        if PASj==1 then
            Pj → PFj
        else
            Wait TS
            Pj → PSj
        end if
        k = random(0, N)
        while k == j do
            k = random(0, N)
        end while
        j = k
    end while
end function
function JTime
    functionAttack()
    functionJob()
    PJStart = M, PJEnd = 0
    while PJEnd < M do
        if PSi==1 then
            PJStart → PJEnd
        end if
    end while
end function
```

among the places. → represents one token moves from the left place to the right place. Note that, we miss out some places and transitions due to space limitation.

D. AUTOMATIC MODEL GENERATION PROGRAM

So far, we use $N = 3$ as an example when explaining our model. Yet in real evaluation environment, defenders may have more VMs prepared for the critical job execution and may test different parameters of the model. We developed an automatic model generation program written by Python. As all the SRN sub-models in this paper are described in a language called CSPL (C-based SPN Language) which is an extension of C language [31]. It is easy for our program to generate the large-scale models with arbitrary parameters.

IV. MODEL RESULTS AND DISCUSSIONS

In this section we first obtain a number of numerical values for the system metrics given at section III.A based on the model and parameters described in the previous section. The numerical results are used for investigating the improvement of the job security as well as the performance loss while using the MTD mechanism. Then we present a verification of our model by comparing the numerical results with simulation results of the model. All the numerical evaluations of our model are obtained by using SPNP tool [32]. The SPNP software package also provides the simulation functions. We use the same parameters in the simulation as in the numerical analysis and compare their results to prove the accuracy of our model.

A. TOTAL JOB EXECUTION TIME UNDER DIFFERENT NUMBERS OF VMs

In this section, we give three sets of comparisons of job completion times for different numbers of VMs using for the job's running. The results are used to assess the effectiveness of MTD mechanism. We conduct statistic experiments by changing the amount of VM (N) from 2 to 9 while all other parameters are given the default values except the attack rate. We set the attack rate β to be 1/3, 1.0 and 5.0, indicating the strong, medium and weak attack environment, respectively. In addition, we give a case of how different number of VMs affects the profit of the defender.

Figure 5 presents the three comparisons results we obtained. The Y-axis represents the job finish time, and the X-axis represents the number of VMs. The three lines in the figure show the three different results under different attack rates. As we can see from the gray line with triangle marks, when the number of VMs increases from 2 to 9, job finish time decreases from 48.3 days to 24.3 days. The main reason is that as the number of VM increases, the VM compromised by the adversary is less likely happens to be the VM that running the job and the job is less likely to be destroyed by the adversary. As a result, the job execution process can be completed faster. This result also happens in the other two cases, as shown by the orange line and blue line in Figure 5. The difference is that when the attack is weak, the decrease trend of the job completion time is not as obvious as when the attack is strong. We can see that the orange line decreases from 26.8 to 21.4 and the blue line decreases from 22.4 to 20.6. The other thing we can observe from Figure 5 is that as the N becomes larger, the decreasing rate of the job finish time becomes slower. See that the 3 lines decline more gently with the growth of N. As in the real computing environment, more VMs mean higher costs. Therefore, it is important to choose the right number of VMs before processing the job execution. Here in this section, we show a case result of how defenders can use our model to choose the best number of VMs to execute the job when considering the job execution benefits. In our case, defenders choose a cloud service provider(CSP) and pays for the VMs for the job execution. According to the service level agreement(SLA), there is a slowest time threshold A, and the defender must complete the job execution progress before this time threshold. Moreover, the sooner the job is completed, the more benefits the defender will get. We use B to indicate the benefits the defender earns each day before the threshold. Except the benefits earn from the job execution, the defender also needs to pay for the cost of the virtual machine. We use C to indicate the cost of one VM. The default value of $C = 1.429\ dollars/hour$ is set according to Digital Ocean pricing list [33]. Given these parameters, we can calculate the defender's profit as follow:

$$P = (A - T) * B - T * N * C$$

Figure 6 shows the results when $A = 100\ days$ and $B = 10, 20, 30\ dollars/day$, respectively. We can see from all three

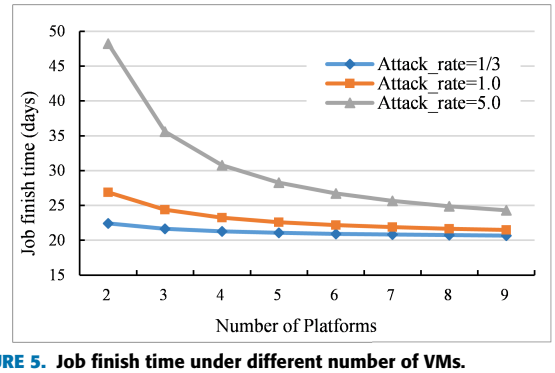

FIGURE 5. Job finish time under different number of VMs.

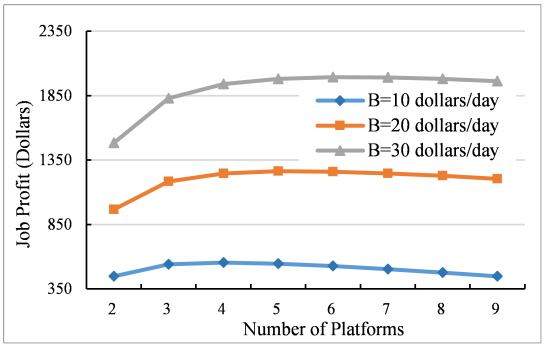

FIGURE 6. Job profit of cloud servive provider under different number of VMs.

lines that even though more VMs will shorten the job finish time, the total cost of the VMs will increase resulting in the lost of total profit. The highest points of the curves in Figure 6 represent the maximum benefit of the defenders.

In general, we can draw the conclusion that more VMs prepared for the job makes the execution more secure and more quicker. However, an appropriate number of VMs is necessary when considering the efficiency.

B. TOTAL JOB EXECUTION TIME UNDER DIFFERENT PHASES

Remember that the job execution is divided into M phases for using the MTD mechanism. In this section, we do numeric experiments by varying the number of M from 1 to 10 and investigates the impact of job execution phases on the whole processing time. Accordingly, the mean time of job execution on a VM $1/\lambda$ changes from 20days to 2days as the whole time needed for the job is 20days in our experiments. Again, we give three sets of results with different attack rates, represented by the three lines in Figure 7 and Figure 8.

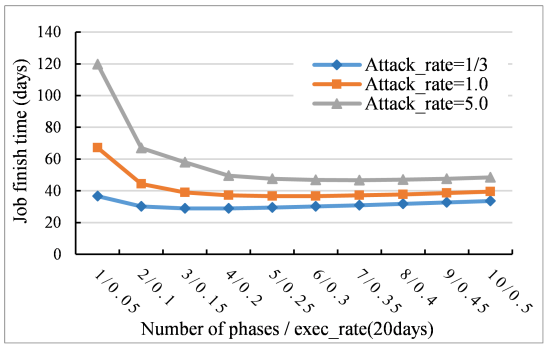

FIGURE 7. Job finish time under different number of phases.

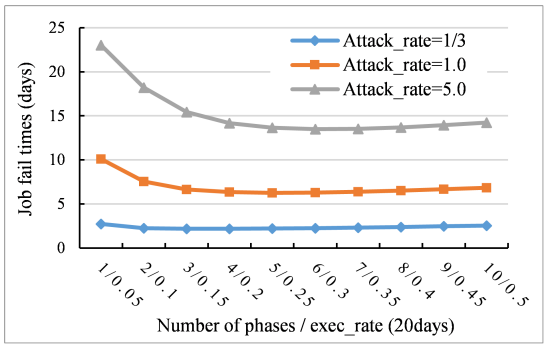

FIGURE 8. Job fail rate under different number of phases.

The purpose of dividing the job execution into multiple phases is to make it harder for the adversary to meet the target VM and reduce the total execution time since the job needed to be re-executed after being destroyed by the adversary. Therefore, intuitively, the more phases we split the execution process, the more secure the job is. Correspondingly, the job finish time will be shorter. However, the gray line with triangle marks in Figure 7 shows that, before a specific phase M, which is 7 in our model, the job finish time decreases as the number of phases M increases. After 7 phases, it keeps increasing. This counterintuitive result is more conspicuous in the other two cases. The orange line illustrates that when the attack rate is 1.0, the job finish time increases with the increasing number of phases M after 3 phases. For the weak attack case, this turning point is 2, illustrated by the blue line. The main reason for this counterintuitive result is that as the number of phases increases, the job migration times grows and the delay caused by the migration becomes longer. Therefore, when the attack is strong, more phases for the job execution would be necessary and effective. Otherwise, more stages may bring more delays. Such results also appear in Figure 8. In Figure 8, Y-axis represents the job fail times which is also the number of successful attacks. Similarly, the job fail times is decreasing before a certain M, then increasing. The main reason is that too many phases means too many migrations. Moreover, as the attacker can wait on PM for a period of time, which makes is more likely to meet the job.

C. IMPACT OF ADVERSARY'S ATTACKING CAPABILITY

So far, we have analyzed the impact of the defender's behavior on the job execution process. In this section, we investigate the impact of the adversary's behavior on the whole execution process. Remember the attack SRN sub-model in Figure 1. There are two parameters related to the adversary's behavior which are attack rate and wait rate. Therefore, by changing the values of these two parameters and calculating the completion time of the job, we can evaluate the adversary's impact on the job. Here we use three VMs as an example and other parameters are set as default.

Figure 9 illustrates the job finish time under different attack capabilities. Since the attack rate and the wait rate affect the job execution simultaneously. We present the results in one figure, as shown by the rainbow surface in Figure 9. We can

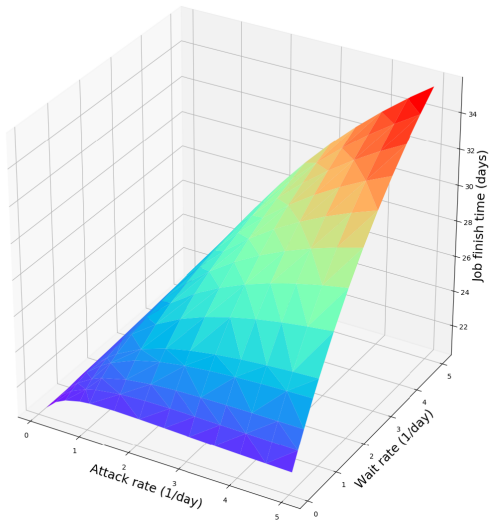

FIGURE 9. Job finish time under different attack capabilities.

observe that whether the attack rate gets faster or the waiting time in a VM gets shorter, it will both lead to an increase in job finish time. Therefore, form the perspective of the adversary, waiting for the job after completing the attack of a VM is a bad strategy, even though the attack rate is slow.

D. VERIFICATION OF THE SRN MODEL RESULTS

In this section, we use job finish time to verify our model and evaluations in the previous sections as the job finish time is the main metric defined in this paper. We do simulation experiments by varying the number of VMs N from 2 to 9 and all other values are set as default which are the same as in the numeric experiments in section A. Thus, our simulation results will be compared with the results shown in Figure 5. Figure 10 illustrates the comparison. We can see that the two results are close, verifying the accuracy of our evaluation results.

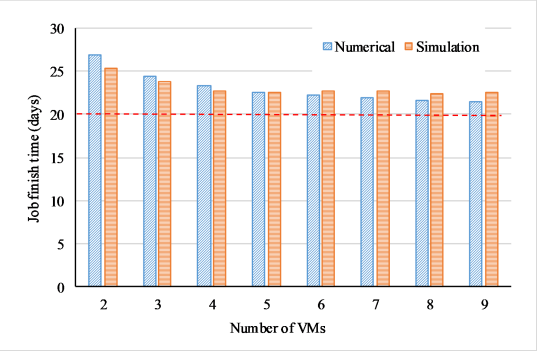

FIGURE 10. Comparison of simulation results and numerical results.

V. CONCLUSION AND FUTURE WORK

This paper presents an analytic modeling framework to evaluate the effectiveness of MTD technology from the perspective of job completion time. We use dynamic platform based MTD as an example and design several SRN sub-models to capture the behaviors of the adversary's and the job migration in the system. We carry out numerical solutions of the model and evaluate the impact of different parameters of both sides on the whole job execution time. Defenders can use our model to calculate an effective and cost-efficient defense strategy. Also, our evaluation model is flexible and can be used on other types of MTD.

Our work in this paper adopts a variety of assumptions on system environment and model parameters. Although, we believe that our assumptions are reasonable, we will try to improve the confidence of our model, such as using simulation verification or building a testbed.

REFERENCES

[1] C. Wang and S. Jajodia, Eds., *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, vol. 54. New York, NY, USA: Springer, 2011.

[2] H. Okhravi, J. Riordan, and K. Carter, "Quantitative evaluation of dynamic platform techniques as a defensive mechanism," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Cham, Switzerland: Springer, 2014, pp. 405–425.

[3] G. Ciardo, A. Blakemore, P. F. Chimento, Jr., J. K. Muppala, and K. S. Trivedi, "Automated generation and analysis of Markov reward models using stochastic reward nets," in *Linear Algebra, Markov Chains, and Queueing Models*, vol. 48. New York, NY, USA: Springer, 1993, pp. 145–191.

[4] V. G. Kulkarni, V. F. Nicola, and K. S. Trivedi, "The completion time of a job on multimode systems," *Adv. Appl. Probab.*, vol. 19, no. 4, pp. 932–954, Dec. 1987.

[5] R. Zhuang, S. A. Deloach, and X. Ou, "Towards a theory of moving target defense," in *Proc. 1st ACM Workshop Moving Target Defense-(MTD)*, 2014, pp. 31–40.

[6] R. Zhuang, "A theory for understanding and quantifying moving target defense," Comput. Inf. Sci., Kansas State Univ., Manhattan, Kansas, Tech. Rep. hdl.handle.net/2097/20525, 2015.

[7] R. Zhuang, A. G. Bardas, S. A. Deloach, and X. Ou, "A theory of cyber attacks: A step towards analyzing MTD systems," in *Proc. 2nd ACM Workshop Moving Target Defense-(MTD)*, 2015, pp. 11–20.

[8] C. Lei, H.-Q. Zhang, L.-M. Wan, L. Liu, and D.-H. Ma, "Incomplete information Markov game theoretic approach to strategy generation for moving target defense," *Comput. Commun.*, vol. 116, pp. 184–199, Jan. 2018.

[9] Q. Zhu and S. Rass, "Game theory meets network security: A tutorial," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.-(CCS)*, 2018, pp. 2163–2165.

[10] C. Qi, J. Wu, G. Cheng, J. Ai, and S. Zhao, "Security analysis of dynamic SDN architectures based on game theory," *Secur. Commun. Netw.*, vol. 2018, pp. 1–10, Jan. 2018.

[11] A. Chowdhary, S. Sengupta, S. Kambhampati, and D. Huang, "Markov game modeling of moving target defense for strategic detection of threats in cloud networks," 2018, *arXiv:1812.09660*, [Online]. Available: https://arxiv.org/abs/1812.09660

[12] R. Zhuang, S. Zhang, and S. A. DeLoach, "Simulation-based approaches to studying effectiveness of moving-target network defense," in *Proc. Nat. Symp. Moving Target Res.*, vol. 246, 2012.

[13] R. Zhuang, S. A. Deloach, and X. Ou, "A model for analyzing the effect of moving target defenses on enterprise networks," in *Proc. 9th Annu. Cyber Inf. Secur. Res. Conf.-(CISR)*, 2014, pp. 73–76.

[14] J. Zheng and A. S. Namin, "The impact of address changes and host diversity on the effectiveness of moving target defense strategy," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jun. 2016, pp. 553–558.

[15] D. Evans, A. Nguyen-Tuong, and J. Knight, "Effectiveness of moving target defenses," in *Moving Target Defense*. New York, NY, USA: Springer, 2011, pp. 29–48.

[16] Y. Han, W. Lu, and S. Xu, "Characterizing the power of moving target defense via cyber epidemic dynamics," in *Proc. Symp. Bootcamp Sci. Secur.-HotSoS*, 2014, p. 10.

[17] K. M. Carter, J. F. Riordan, and H. Okhravi, "A game theoretic approach to strategy determination for dynamic platform defenses," in *Proc. 1st ACM Workshop Moving Target Defense-(MTD)*, 2014, pp. 21–30.

[18] P. J. Donovan, J. W. Mclamb, H. Okhravi, J. Riordan, and C. V. Wright, "Quantitative evaluation of moving target technology," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Apr. 2015, pp. 1–7.

[19] M. I. El, A. Chowdhary, and D. Huang, "Software defined stochastic model for moving target defense," in *Proc. Int. Afro-Eur. Conf. Ind. Advancement*. Cham, Switzerland: Springer, 2016, pp. 188–197.

[20] B. Van Leeuwen, W. Stout, and V. Urias, "MTD assessment framework with cyber attack modeling," in *Proc. IEEE Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2016, pp. 1–8.

[21] G. Ciardo and K. S. Trivedi, "A decomposition approach for stochastic reward net models," *Perform. Eval.*, vol. 18, no. 1, pp. 37–39, Jul. 1993.

[22] T. A. Nguyen, K. Han, D. Min, E. Choi, T. D. Thang, and Y.-J. Choi, "A stochastic reward net-based assessment of reliability, availability and operational cost for a software-defined network infrastructure," *J. Supercomput.*, vol. 75, no. 8, pp. 4657–4683, Aug. 2019.

[23] Y. Liu, K. Wang, L. Ge, L. Ye, and J. Cheng, "Adaptive evaluation of virtual machine placement and migration scheduling algorithms using stochastic Petri nets," *IEEE Access*, vol. 7, pp. 79810–79824, 2019.

[24] A. Naghash Asadi, M. Abdollahi Azgomi, and R. Entezari-Maleki, "Unified power and performance analysis of cloud computing infrastructure using stochastic reward nets," *Comput. Commun.*, vol. 138, pp. 67–80, Apr. 2019.

[25] M. Torquato and M. Vieira, "Towards models for availability and security evaluation of cloud computing with moving target defense," 2019, *arXiv:1909.01392*. [Online]. Available: https://arxiv.org/abs/1909.01392

[26] G. Cai, B. Wang, and Y. Luo, "A model for evaluating and comparing moving target defense techniques based on generalized stochastic Petri net," in *Proc. Conf. Adv. Comput. Archit.* Singapore: Springer, 2016, pp. 184–197.
[27] W. Connell, D. A. Menascé, and M. Albanese, "Performance modeling of moving target defenses," in *Proc. Workshop Moving Target Defense-(MTD)*, 2017, pp. 53–63.
[28] W. Connell, D. A. Menasce, and M. Albanese, "Performance modeling of moving target defenses with reconfiguration limits," *IEEE Trans. Depend. Sec. Comput.*, early access, 2018, doi: 10.1109/TDSC.2018.2882825.
[29] X. Chang, S. Lv, R. J. Rodriguez, and K. Trivedi, "Survivability model for security and dependability analysis of a vulnerable critical system," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–6.
[30] G.-L. Cai, B.-S. Wang, W. Hu, and T.-Z. Wang, "Moving target defense: State of the art and characteristics," *Frontiers Inf. Technol. Electron. Eng.*, vol. 17, no. 11, pp. 1122–1153, 2016.
[31] S. P. Harbison, *C: A Reference Manual*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.
[32] G. Ciardo, J. Muppala, and K. Trivedi, "SPNP: Stochastic Petri net package," in *Proc. 3rd Int. Workshop Petri Nets Perform. Models*, Dec. 1989, pp. 142–151.
[33] DigitalOcean. *Pricing on DigitalOcean-Cloud Virtual Machine & Storage Pricing*. Accessed: Nov. 26, 2019. [Online]. Available: http://www.digitalocean.com/pricing/#Compute

ZHI CHEN is currently pursuing the Ph.D. degree with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University. His research interests include cloud computing, information security, and moving target defense.

XIAOLIN CHANG (Member, IEEE) received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology, in 2005. She is currently a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. Her current research interests include cloud data center and network security.

ZHEN HAN is currently a Professor with the School of Computer and Information Technology, Beijing Jiaotong University. He has published over 100 scientific articles in various journals and international conferences. His main research interests are trusted computing, cryptographic protocols, privacy preserving, and network security.

YANG YANG received the B.S. degree from East China Jiaotong University, China, in 2012, and the M.S. degree from Beijing Jiaotong University, where she is currently pursuing the Ph.D. degree with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation. Her research interests include cloud data center and machine learning.