IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 22, 2019, accepted September 11, 2019, date of publication September 23, 2019, date of current version October 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943273

A Model for Availability and Security Risk Evaluation for Systems With VMM Rejuvenation Enabled by VM Migration Scheduling

MATHEUS TORQUATO[1,2], (Student Member, IEEE), PAULO MACIEL[3], (Member, IEEE), AND MARCO VIEIRA[1], (Member, IEEE)

[1]CISUC, Department of Informatics Engineering, University of Coimbra, 3004-531 Coimbra, Portugal
[2]Federal Institute of Alagoas (IFAL), Arapiraca 57317-291, Brazil
[3]Center of Informatics (CIn), Federal University of Pernambuco (UFPE), Recife 50670-901, Brazil

Corresponding author: Matheus Torquato (matheustor4.professor@gmail.com)

This work has been partially supported by project MobiWise project: From mobile sensing to mobility advising (P2020 SAICTPAC/0011/2015), cofinanced by COMPETE 2020, Portugal 2020-Operational Program for Competitiveness and Internationalization (POCI), European Union's ERDF (European Regional Development Fund) and the Portuguese Foundation for Science and Technology (FCT).

ABSTRACT Most companies and organizations rely nowadays on virtualized environments to host and run their applications. Some of these applications have stringent availability and security requirements. An important challenge for high availability in virtualized systems is software aging, which can lead the system to hangs or other types of failures. Software rejuvenation is applied to cope with software aging problems, whereas previous research suggests the use of Virtual Machine (VM) migration to reduce the downtime related to Virtual Machine Monitor (VMM) software rejuvenation. However, there is still a gap regarding the security implications of applying VM migration scheduling as support for VMM software rejuvenation. In this paper, we propose a security evaluation approach based on an availability model for virtualized systems with VM migration for VMM rejuvenation. The goal is to find the proper rejuvenation scheduling to reach the desired levels (or at least to avoid the undesired levels) of security risk and availability. We present three case studies comprising major security threats, namely Man-in-the-middle and Denial of Service attacks. Results provide insightful information regarding the tradeoff between availability and security risk when applying VM migration scheduling for rejuvenation purposes.

INDEX TERMS Availability, cloud computing, security, software aging and rejuvenation, VM migration.

I. INTRODUCTION

Cloud computing security and dependability appear as essential challenges for industry and academia. Existing surveys show that Cloud Computing security is at the top of users' concerns [1]. Likewise, assuring high levels of dependability in Cloud computing remains a significant research challenge. Specifically, there is a need for developing holistic models for Cloud dependability evaluation [2].

Previous works highlighted indicators of software aging accumulation in Cloud components [3]. Software aging is a cumulative process that can lead software to hangs or other failures [4], whereas software rejuvenation is used to counteract software aging [5]. VM migration scheduling is an approach to reduce the downtime related to VMM software rejuvenation [6], [7]. However, potential security issues of applying VM migration as support for VMM rejuvenation are still not understood. Besides that, a security-aware software rejuvenation scheduling is hard to achieve due to the uncertainty related to security events [8]. Pietrantuono and Russo [9] also highlight the importance of studying the security impacts caused by software rejuvenation policies.

There are several studies on the availability evaluation of VM migration scheduling as support for VMM software rejuvenation [10], [11]. Mainly, these aim at finding the optimal rejuvenation schedule to maximize system availability, but none of them deal with security issues. From a security perspective, there are also some works on VM migration security, as presented in [12], for example. However, none of those covers software aging and rejuvenation aspects.

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

Our work aims at evaluating the security impact caused by VM migration scheduling as support for VMM software rejuvenation. This paper intends to address the following research question:

RQ_{main} - What is the security risk impact of applying different VM migration policies for VMM software rejuvenation purposes?

On top of this, we consider two sub-questions:

- *RQ_{s1} - What is the VM migration policy that reduces the system security risk?*
- *RQ_{s2} - What are the trade-offs between availability and security when using VM migration scheduling as support for VMM rejuvenation?*.

To answer these questions, we propose an availability model based on Stochastic Rewards Nets (SRN) for systems with VM migration scheduling for VMM software rejuvenation purposes. The system architecture considered has three main components: *VM* - a virtual machine running the desired application; *Main Node* - a physical machine that hosts the *VMs*; and *Standby Node* - a physical machine used to receive VM migrations. The selected architecture covers the main components of virtualized environments. Indeed, complex virtualized infrastructures in Cloud Computing also have similar architectural components [11], [13].

From the proposed availability model, we extract a security measure named RiskScore. Instead of assuming the attacker behavior (which is very difficult to characterize), this metric is based on the time that the system spends on risky states (from a security perspective). The assumption is that the attack success is related to the time spent in a risky (or vulnerable) state. So, RiskScore metric captures the elapsed time in a condition (or state) which raises or enables a successful security attack. Therefore, as we have different preconditions for different security attacks, RiskScore calculation depends on the considered security threat (e.g., Denial of Service, Man-in-the-middle).

We present three case studies to validate our proposal. The first considers the Man-in-the-middle security threat, the second considers Denial of Service attacks, and the last one is a combination of both threats. We provide a set of scenarios in each case study covering different VM migration scheduling alternatives to support the analysis of the tradeoff between availability and security risk.

Availability is a dependability and also a security attribute [14]. This way, both dependability events (e.g., failures, crashes or hangs) and security events (e.g., malicious activities or security attacks) can affect system availability. However, in this paper, the availability evaluation is considered only from the dependability perspective. From the security perspective, we propose and evaluate the RiskScore, a metric focused on measuring the security risk (more details in Section II-D).

The highlights of this paper are:

- We propose a simple and flexible approach to support security evaluation considering different threats and using the same availability model;
- Results allow studying the VM migration policy that globally maximizes the security or availability levels;
- For each case study, we provide an extensive set of scenarios to support the decision-making process.

Up to our knowledge, this is the first research effort on the analysis of the tradeoff between availability and security risk in virtualized systems with VM migration as VMM rejuvenation. We present a comprehensive set of results that provides some understanding of the availability and security risk tradeoffs raised by such a rejuvenation technique. Our security evaluation approach can be adapted for other threat models without requiring availability model modification.

Note that this paper is not intended to propose high-availability mechanisms (i.e., 99.999% of system availability [15]) for virtualized systems with VM migration as VMM software rejuvenation. However, we aim to maximize the system availability observing the considered assumptions used in this research (more details in Section II). The focus of this paper is on the security aspect of applying VM migration scheduling as a VMM software rejuvenation technique.

The rest of this paper is organized as follows. Section II presents our evaluation assumptions. Section III elaborates on the proposed model. Section IV presents and discusses the results. Section V Finally, Section VI puts forward conclusions and ideas for future work.

II. ASSUMPTIONS

This section states our assumptions for the proposed modeling and evaluation.

A. SYSTEM ARCHITECTURE

The system architecture considered includes three main components: *Main Node*, *VM*, and *Standby Node*. The *Main Node* represents the physical machine that runs the *VM*, the *VM* runs the target application, and the *Standby Node* is a standby machine used as VM Migration target. As mentioned earlier, this configuration is part of common virtualized environments. Cloud computing platforms, such as Openstack,[1] include the *compute nodes* that act as the *Nodes* of our system architecture.

We consider two major security threats in our evaluation: i) Man-in-the-middle attacks, and ii) Denial of Service (DoS) attacks. More details about the security threats are presented in the Section IV. Figure 1 depicts the system architecture.

Our previous research [7], [11] showed software aging effects on the VMM software component. Thus, in this first work, we decided to focus on the software aging effects in the VMM component. To overcome software aging problems, we propose a software rejuvenation strategy based on VM migration. The idea behind the rejuvenation technique (explained in details in the Section II-C) is to move the *VM* from the physical machine (i.e., *Main Node*) experiencing software aging problems to another physical machine without aging accumulation (i.e., *Standby Node*). We consider

[1] https://www.openstack.org/

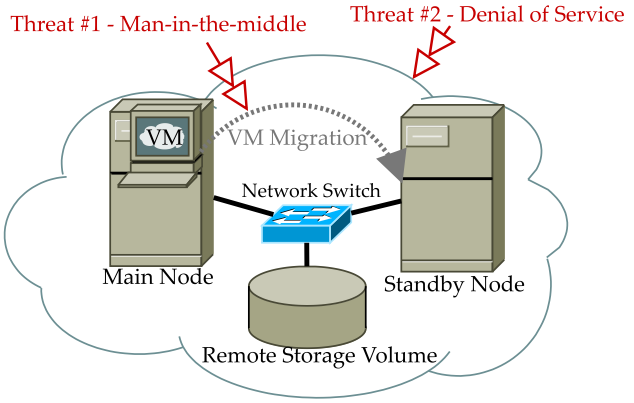

FIGURE 1. System architecture.

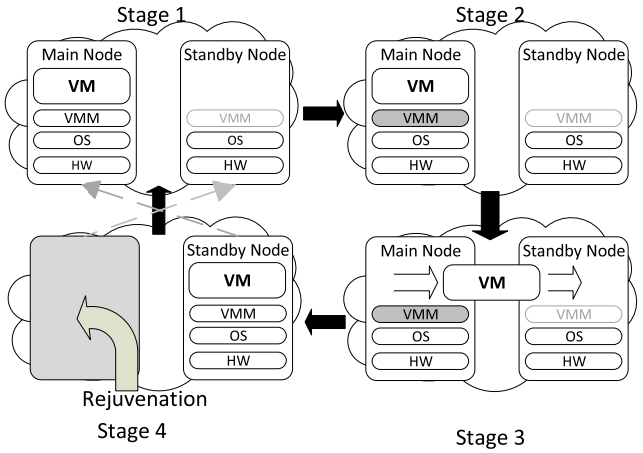

FIGURE 2. VMM rejuvenation technique workflow.

a remote storage volume to enable VM live migration [16]. Such remote volume persists the virtual disk of the VM. Moreover, in the VM live migration algorithm, the system only has to transfer the memory pages and processor state instead of all the VM components (processor state, memory, and virtual disk).

We also consider that the VM migration process uses a *pre-copy* algorithm [16]. We can loosely define the *pre-copy* algorithm in two phases: 1) *copy-phase* - allocating resources on the migration target and transferring memory pages, and 2) *downtime* - transferring the processor state and acknowledging the migration. During the *copy-phase* the VM is still running and delivering service. We also highlight that the VM migration cost depends on the amount of resources used in the VM, specifically the amount of dirty memory pages and the dirty page rate.

B. FAILURE AND OPERATIONAL MODES

The system becomes non-operational after a *VM* failure. The system returns to operation after a *VM* repair. As the *VM* depends on the *Main Node* to perform its operations, failures on the *Main Node* affect the VM availability. After a *Main Node* failure, the system turns operational again by performing a two-step recovery: repair the *Main Node* and restart the *VM*. Failures on the *Standby Node* do not represent a system failure. Nevertheless, *Standby Node* failures prevent the *VM* migration. The proposed model also considers failures that are not related to software aging (e.g., Hardware or Operational System failures).

The model does not consider the following failures: (i) network failures; (ii) system failure during VM migration; (iii) VM Live migration failures; and (iv) failures in the remote storage volume. The inclusion of these behaviors in the proposed model may cause a state-explosion, and then turning the model reward computation too long. Therefore, we intend to cover them in our future work by applying techniques such as interacting models [17].

C. VMM REJUVENATION TECHNIQUE

The VMM rejuvenation technique has four stages. In the first stage, the *Main Node* and the *VM* run without software aging accumulation. As the *Standby Node,* is not running any VMs, the Virtual Machine Monitor (VMM) does not suffer from software aging accumulation. As time passes, VMM software starts to experience aging accumulation. This event leads to the second stage, where the *Main Node* presents symptoms of software aging accumulation. When the rejuvenation interval is reached, the VM Live Migration is triggered. The next stage is the VM Live Migration. Assuming that the *Standby Node* is running, the VM is moved from the *Main Node* to the *Standby Node*. In the last stage, the *Standby Node* takes the role of *Main Node*. In this stage, we also perform software rejuvenation on the *Main Node* to clear the software aging accumulation. After software rejuvenation, that machine is ready to receive VM migrations. Thus it becomes the *Standby Node*. Figure 2 summarizes these steps.

D. SECURITY EVALUATION APPROACH

We propose a metric named RiskScore, which reflects the probability of the system to be in a condition that enables (or improves the likelihood of) a successful specific attack. Our approach allows the computation of the RiskScore metric for different threats using the same availability model.

To clarify the proposed approach, let us consider an illustrative example. Consider a Virtual Machine with migration capabilities. The Virtual Machine has three main states, as shown in Figure 3. The `UP` state means that the VM is running, the `DW` state represents that the VM is down, and `MG` represents that the VM is migrating. The state machine diagram is presented in Figure 3 that also includes the transitions between states. In this example, we neglect possible VM failures during migration.

Now, let us suppose that such a system is liable to suffer an attack related to data steal during migration. So, in this case, the only state that raises a security concern is `MG`. Therefore, the RiskScore is a measure based on the probability of the system being in the `MG` state. With this, the system manager may adjust the α parameter (which is related to the frequency of migrations) to achieve the desired levels of RiskScore.

This security evaluation approach has two main advantages: i) ***focus on the system state rather than on the attacker's behavior*** - we assume that the attacker's behavior

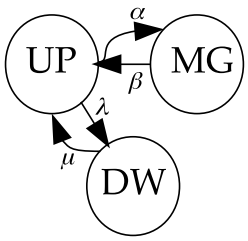

FIGURE 3. Illustrative example - State-machine diagram.

during the attack is unpredictable. Therefore, our security evaluation method computes the security levels only from the system state; ii) ***security evaluation using unaltered availability models*** - the security evaluation approach does not require the modification of the availability models. As we obtain security levels from the system state, the evaluation is made from the steady-state probability of the system being in a specific state(s). Therefore, we can perform the security evaluation using proper model's reward measures.

We highlight that the RiskScore metric is intended to measure the security risk associated with the time spent in risky states. RiskScore does not assumes attacks characteristics as attack probability or rate.

III. MODEL

Figure 4 presents the proposed model. We adopt a monolithic model, but for this explanation let us divide two main areas of this monolithic model: a) the Clock Model composed by the places `Clock` and `Schedule`, and transitions `Trigger` and `ResetClock`; and b) the Main Model composed by the other places and transitions.

The Clock Model represents the behavior of the rejuvenation scheduling on the environment. The Clock can be any type of component capable of counting time and communicating with the system. The token in the `Clock` place represents that the time counting to VM Migration submission is active. The deterministic transition `Trigger` represents the time interval between submitting VM Migration requests to the environment. When the deterministic time is reached, the `Trigger` transition fires putting a token in the `Schedule` place. The `Schedule` place with tokens represents that the VM Migration is about to start. However, the VM Migration start (`StartLM` transition firing) depends on two more conditions: 1) *Main Node* and *VM* running (a token in the `UP` place), and 2) *Standby Node* running (a token in `SN_UP` place). Once those conditions are satisfied, the VM Migration starts (`StartLM` firing). `StartLM` firing removes the tokens from the places `UP` and `SN_UP` and puts a token in place `LM`. `StartLM` also swaps[2] the token in the place `Schedule`. We use this strategy to avoid including guard functions in the model, therefore improving its readability. We highlight that the token swap does not affect token aging because the `StartLM` transition is an immediate transition. When the system starts the migration, the Clock component has to start the time counting for the next VM Migration (`ResetClock` firing). `ResetClock` firing swaps the token from the `LM` place, and moves the token from the `Schedule` place to the `Clock` place, thus restarting the VM migration interval time counting.

Regarding the VM Migration process, when the `LM` place has tokens, the transition `PC` becomes enabled. This transition is related to the time of the *copy-phase* of the VM migration algorithm. We parameterized this transition based on [16] with the mean time of 72 seconds. The transition `LM_dwt` represents the *downtime* of the VM migration. As the time of VM migration may vary depending on the workload, we used exponential transitions instead of deterministic transitions. When the VM migration finishes (`LM_dwt firing`) the system returns to operation (a token is deposited in the `UP` place), and the previous *Main Node* will pass through software rejuvenation (a token is deposited in the `SN_W` place). VM migration will be enabled again after the software rejuvenation action. This behavior is represented by `Rej` transition firing, moving the token from the `SN_W` place to the `SN_UP` place.

We assume that the *Main Node* and the *VM* are operational at the start of the model analysis. This assumption is represented by the token in `UP` place. At this point, if the *VM* fails (represented by `VM_f` transition) the system goes out-of-service (the token is removed from the `UP` place and deposited in the `VM_DW` place). If the *VM* is correctly repaired, then the system returns to activity. The delay time for *VM* repair is represented by the `VM_r` transition. However, the *VM* failure may be followed by a *Main Node* failure. This behavior is represented when the `MN_f2` transition fires and puts a token in the `DW` place. A *Main Node* failure can also occur before a *VM* failure (`MN_f` transition firing). The system recovery after a *Main Node* is made in two steps: first, the *Main Node* repair (represented by `MN_r` transition firing), after the *Main Node* repair the VM is stopped (a token is deposited in the place `VM_S`. The second step of the system recovery is the *VM* reboot (`VM_rb` transition firing).

A 4-phase Erlang sub-net represents the aging accumulation process. This type of subnet can represent the Increasing Failure Rate (IFR) of a software aging accumulation process [13]. We highlight that other distributions can be applied in the software aging modeling process [18]. However, the inclusion of such distributions may require complete model redesign and are out of the scope of this paper. We selected Erlang sub-nets in the software aging modeling because it is widely adopted for this purpose [19], [20]. The `AgingPhase` transition represents the phases of the Erlang sub-net. A failure caused by software aging occurs when the accumulation status reaches critical levels. In the model, this behavior is represented when the `AgingFailure` transition fires, removing the token in the `UP` place and putting it in the `DW2` place. The `Repair` transition represents the recovering process after a software aging failure. The `ClearAging` and `ClearAging2` transitions represent events that clear the software aging accumulation process. We consider that

[2]receives and gives back

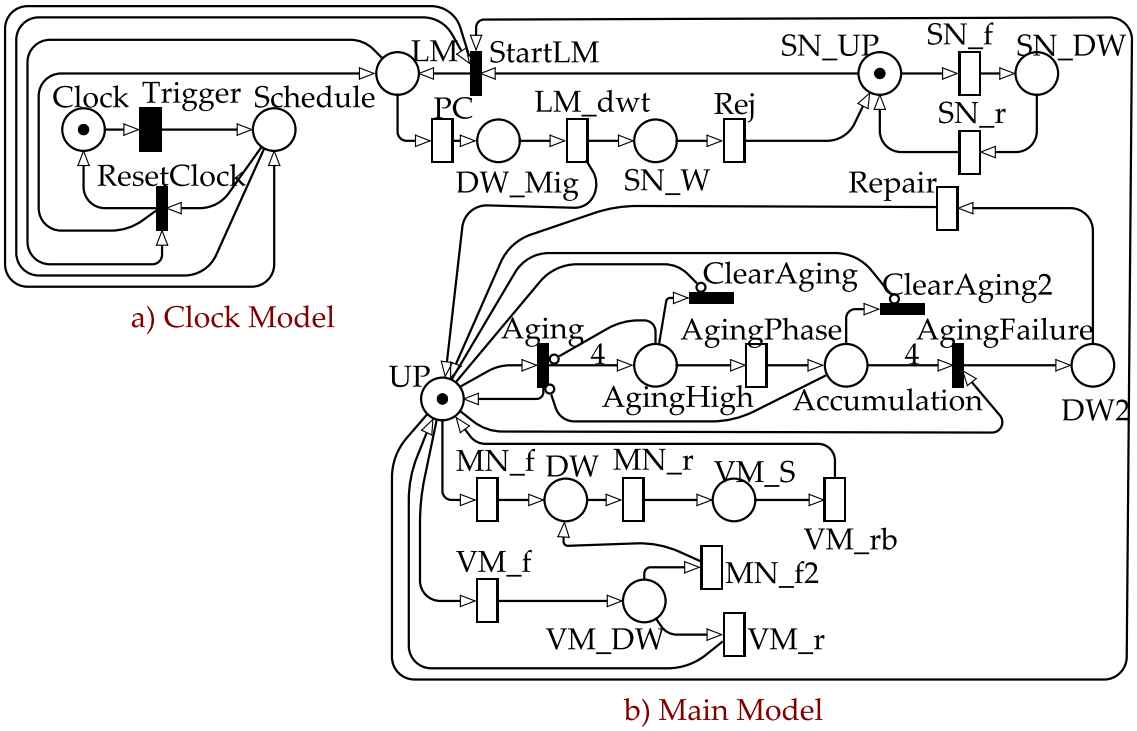

FIGURE 4. SRN model.

TABLE 1. Immediate transitions and their meaning.

Transition	Meaning	Input arcs	Output arcs
`ResetClock`	Restart time counting for VM migration	`Schedule` (1) `LM` (1)	`Clock` (1) `LM` (1)
`StartLM`	Start of VM migration process	`SN_UP` (1) `UP` (1) `Schedule` (1)	`Schedule` (1) `LM` (1)
`Aging`	Start of software aging accumulation	`UP` (1) `AgingHigh` (0) `Accumulation` (0)	`UP` (1) `AgingHigh` (4)
`AgingFailure`	System failure occurence due to software aging	`UP` (1) `Accumulation` (4)	`DW2` (1)
`ClearAging1`	Software aging accumulation clearance	`AgingHigh` (1) `UP` (0)	Transition without output arcs
`ClearAging2`	Software aging accumulation clearance	`Accumulation` (1) `UP` (0)	Transition without output arcs

the *Main Node* and the *VM* repairs comprise rejuvenation actions. Therefore, the software aging accumulation cleanup occurs when the *Main Node* or the *VM* fail. The other case of software rejuvenation occurs on VM Live Migration. After VM Live Migration, the *VM* arrives on a fresh state VMM. Therefore, the software aging status is removed. We used inhibitor arcs[3] from the `UP` place to the `ClearAging` and `ClearAging2` transitions to represent the behaviors related to software rejuvenation.

The model also covers *Standby Node* failures, which affect system availability indirectly as it prevents VM migration turning a software aging failure more likely to occur. The transitions `SN_f` and `SN_r` represent the *Standby Node* failure and repair events, respectively. The places `SN_UP` and `SN_DW` are related to the *Standby Node* availability and unavailability, respectively.

Table 1 summarizes the immediate transitions, their meaning, and their input and output arcs with respective weights.[4] For the sake of readability, we present the list of the places used in our model. Table 2 presents the information about the places. The column **meaning with tokens** represents the system state when the considered place has tokens.

Our availability model comprises software aging and rejuvenation behavior, non-aging failures and details of VM migration (e.g., pre-copy and downtime phases). However, we applied some simplifications to avoid the state explosion problem.

[3]arc terminating in a circle instead of an arrow

[4]Associated number of tokens. We use zero (0) for inhibitor arcs.

TABLE 2. Places description.

Place	Meaning with tokens
Clock	Timer counting for the next migration
Schedule	Timer reaches the VM migration interval time
LM	VM migration is started
DW_Mig	VM migration is on the downtime phase
SN_W	Standby Node is waiting for the rejuvenation action
SN_UP	Standby Node is up
SN_DW	Standby Node is down
UP	Main Node and VM are running
AgingHigh	VMM starts to accumulate software aging
Accumulation	The number of tokens in this place represents the VMM software aging accumulation status
DW2	The system is down due to a software aging failure
DW	The system is down due to a Main Node non-aging failure
VM_DW	The system is down due to a VM non-aging failure
VM_S	VM is waiting for a reboot after a Main Node repair

TABLE 3. Parameters used in the timed transitions.

Parameters		Values
Transition Name	*Description*	*Mean time*
MN_f, MN_f2	*Main Node* Failure Delay	1236.706 h
MN_r	*Main Node* Repair Delay	1.094 h
SN_f	*Standby Node* Failure Delay	1236.706 h
SN_r	*Standby Node* Repair Delay	1.094 h
VM_f	Virtual Machine Failure Delay	2880 h
VM_r	Virtual Machine Repair Delay	30 min
VM_rb	Virtual Machine Reboot Delay	5 min
PC	VM Live Migration pre-copy phase time	72 s
LM_dwt	VM Live Migration Downtime	4 s
Rej	Rejuvenation Node Delay	2 min
SARec	Software Aging Recovery Delay	1 h
AgingPhase	Time to Aging (Phases)	62.5 h
Trigger	Interval to VM Live Migration	$1 \rightarrow 168$ h

IV. RESULTS AND ANALYSIS

The numerical evaluations of the models are obtained using the TimeNET tool [21]. Table 3 contains the parameters used in our evaluation. These parameters are based on previous experimental studies and consolidated papers [11], [16], [22].

For the three case studies, we compute the system unavailability using the following: $Unavailability = 1 - Availability$. $Availability$ is obtained through $Availability = P(\#UP > 0) OR (\#LM > 0)$, where we are computing the probability of token presence in the places UP or LM. In other words, we obtain the system availability observing the probability of the system being running or in the *copy-phase* of VM migration. Unavailability is the probability that the system is out of such a state.

A. CASE STUDY 1 - MAN-IN-THE-MIDDLE ATTACK

In Man-in-the-middle (MITM) attacks, the attacker has access to the data link between two communication endpoints [23] With such access, the attacker can deploy attacks to change the network traffic or to eavesdrop the communication [24]. Even with proper VM migration data traffic encryption, an attacker may recognize the migrating VMs in the network.

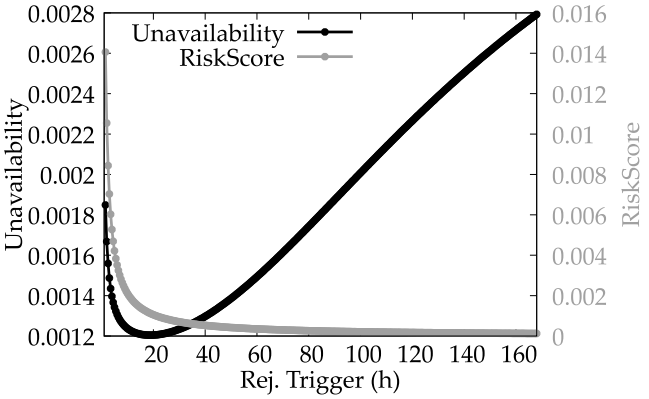

FIGURE 5. Case study 1 results - Man-in-the-middle.

1) THREAT MODEL

We consider an attacker that has the necessary skills to hijack the VM migration route and to perform a malicious action (e.g., secretly copying the data in traffic, changing or destroying the data in the VM migration packets).

As we are using VM migration scheduling for VMM rejuvenation purposes, frequent migrations may raise the concern regarding this type of attack. We compute the RiskScore of this case study with the following expression: $RiskScore = MigrationProbability$. The $MigrationProbability$ is the probability of the system on a VM migration. We obtain $MigrationProbability$ by observing the probability of token presence in places LM or DW_mig. We use the following metric: $MigrationProbability = P(\#LM > 0)\ OR\ (\#DW\_Mig > 0)$, where $P\{\}$ refers to probability and $\#LM$ and $\#DW\_Mig$ refer to the number of tokens in the LM and DW_Mig places, respectively.

2) RESULTS AND DISCUSSION

Figure 5 presents the results obtained. The Y-axis represents the system unavailability, and the X-axis represents the rejuvenation trigger considered (in hours). We add a secondary Y-axis in the right side with the values of the RiskScore. The black dotted line represents unavailability, and the gray dotted line is the RiskScore. We vary the rejuvenation trigger from one hour to 168 hours (a week) with a half-hour step. We highlight that the RiskScore in this case study only considers the probability that the system is performing a VM migration.

Figure 5 shows a valley for the system unavailability values. This point represents the rejuvenation trigger that maximizes system availability. It shows the balance between too frequent rejuvenation that may impair availability due to each migration downtime, and less frequent migrations that raise the probability of software aging failures. Besides that, it is

TABLE 4. Evaluation scenarios of Case study 1 - Man-in-the-middle.

Scn #	Criteria	Rej. Trigger	UNAVAILABILITY	Downtime (h/yr)	RISKSCORE	Risk classification
0	*Optimal policy*	20 h	0.00120484	10.5544	0.00105544	5% of MAXRS
1	100% UNAVAILABILITY	19 h	0.00120442	10.55072	0.00111098	5.27% of MAXRS
2	75% UNAVAILABILITY AND 25% RISKSCORE	22.5 h	0.00120813	10.58322	0.00093818	4.45% of MAXRS
3	50% UNAVAILABILITY AND 50% RISKSCORE	27 h	0.00122098	10.69578	0.00078183	3.71% of MAXRS
4	25% UNAVAILABILITY AND 75% RISKSCORE	35.5 h	0.00126506	11.08193	0.00059464	2.82% of MAXRS
5	100% RISKSCORE	168 h	0.00279284	24.46528	0.00012566	0.59% of MAXRS

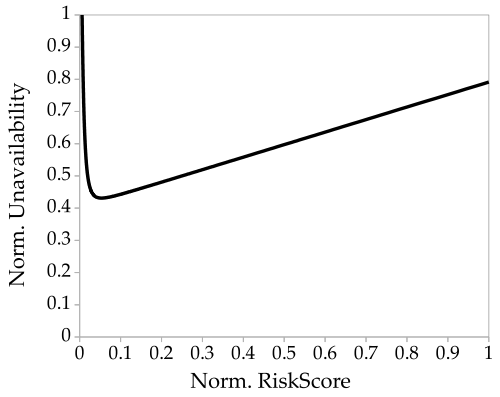

FIGURE 6. Case study 1 results - Man-in-the-middle - Normalized plot.

noticeable that, as expected, less frequent migrations reduces the security risk associated with MITM attacks.

Unavailability and RiskScore are non-benefitial metrics (i.e. the lower the better). The plot in the Figure 6 presents the normalized values for Unavailability versus the normalizes values for RiskScore. We decided to normalize the data to reduce possible bias due to the difference in magnitude of the metrics. We apply a simple normalization, as follows: $Norm(UA_{(n)}) = UA_{(n)}/Max(UA)$, where $Norm(UA_{(n)})$ is the normalized value of Unavailability when applying rejuvenation interval of n hours, $UA_{(n)}$ is the actual Unavailability value when using rejuvenation interval of n hours, and, $Max(UA)$ is the maximum observed value of Unavailability in the considered rejuvenation interval range of values (i.e. from one hour to 168 hours, using a half-hour step).

In Figure 6, the point in the curve with the minimum distance to the origin (0,0) represents the configuration that globally reduces the combination of Unavailability and RiskScore. We computed the Euclidean distance of all the points of the curve to the origin point. The point with the shortest distance corresponds to the rejuvenation policy with VM migration interval of 20 *hours*. The detailed results are presented in Table 4 (Scenario #0).

Our analysis also aims at finding the rejuvenation policies that minimize the values of Unavailability or RiskScore or even a composition of both. By composition, we mean the desired proportion of each metric. For example, a system manager may want to deploy a specific VM migration policy that considers both metrics (Unavailability and RiskScore) equally (50% Unavailability and 50% RiskScore), or some other values. We computed some intermediate scenarios using proportions for both metrics. To avoid decision bias, we used the normalized data mentioned earlier. The results are shown in Table 4, which includes the following: **Scn #** - Proposed scenario; **Criteria** - considered proportion for each metric in the composition; **Rej. Trigger** - specific VM migration policy observing the desired criteria; **UNAVAILABILITY** - steady-state unavailability; **Downtime (h/yr)** - estimated downtime in hours per year; **RISKSCORE** - steady-state risk score; and **Risk classification** - a comparison between the obtained RiskScore and the maximum RiskScore observed in the analysis[5] (MaxRS). The **Risk classification** result provides a proportion of the increase/reduction impact due to the selected VM migration policy when compared to the scenario with the worse security levels. Thus, **Risk classification** results are useful to understand the actual levels of security improvement when applying the VM migration policies.

Each case study results presented in this paper provide the necessary inputs for answering RQ_{s1} and RQ_{s2}. The answer for the RQ_{s1} is on the tables with the case studies results in the scenario which minimizes RiskScore metric (100% Risk row on the results table). The plots and tables in each case study provide the information for answering RQ_{s2}.

From the results, we can observe that the rejuvenation policy decision depends on the system manager criteria about Unavailability and RiskScore. We provide a comprehensive set of results to support such a decision. We highlight the following conclusions from this case study: 1) a VM migration interval of 20 hours provides the best overall rejuvenation policy considering both metrics equally; 2) when putting more weight on Unavailability in the rejuvenation policy decision, the VM migration interval tends to be shorter than in scenarios with more weight on RiskScore; and 3) longer VM migration intervals provide the best RiskScore result. We recall that the maximum limit of our analysis is 168 hours. As Man-in-the-middle attacks depend on VM migration, less frequent migrations will reduce the value of RiskScore.

B. CASE STUDY 2 - DENIAL OF SERVICE (DOS) ATTACK

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are the main threats for Cloud Computing availability [25]. Cloud Computing DoS attacks usually aim at flood the network resources or impair the application

[5] Variation of Rej. Trigger from one hour to 168 hours with half-hour step.

running on the VMs. DoS attacks are even more devastating on Cloud because of its flexibility. So, when receiving a high workload, the Cloud environment starts to allocate more resources (i.e., servers and VMs) to handle the incoming requests. Therefore, by flooding just one of the servers, the DoS attack may end up affecting the overall Cloud Computing availability.

1) THREAT MODEL

The considered DoS attack consists of a high workload demand for an application running upon a VM. So, the RiskScore metric for this case study has to cover both aspects: network and application state. The reasoning is that the VM migration process affects the network state, and software aging affects the application state. Besides that, software aging accumulation forces the system to use more memory. Thus, software aging accumulation also affects the VM migration network overhead as there are more dirty memory pages.

We compose the RiskScore for this case study as follows. $RiskScore = (w1 \times RiskMigration) + (w2 \times RiskAging)$. $w1$ and $w2$ are the assigned weights for VM migration security risk and software aging security risk, respectively. As VM migration produces high bursty network traffic, we assign more weight to *RiskMigration*. In practice, the weight configuration is $w1 = 0.6$ and $w2 = 0.4$. *RiskAging* is the expected software aging accumulation. We compute the *RiskAging* from the expected number of tokens in the place `Accumulation` (see model in Fig. 4). The *RiskMigration* is obtained using the equation: $RiskMigration = MigProbability \times RiskAging$, where *MigProbability* is the probability of the system being performing a VM migration (as in the previous case study). We decided to add the *RiskAging* in the *RiskMigration* to capture the following behavior: a migration of a VM with more accumulation of software aging will produce a higher impact in the network than a VM with less software aging accumulation.

2) RESULTS AND DISCUSSION

Figure 7 presents the obtained results. As in the previous case study, the black lines are related to the system unavailability, and the gray lines are related to the RiskScore metric. We also added a secondary y-axis with the scale for the RiskScore values. The plot also contains two continuous lines for the baseline (i.e., without software rejuvenation) architecture. The variation of the VM migration scheduling obeys to the one presented earlier, from one hour to a week (168 hours) with half-hour steps.

The gray lines interception represents the point where the VM migration scheduling is no longer beneficial for the RiskScore when compared to the baseline. The exact intercept is when the rejuvenation interval is equal to 57 hours, which means that larger rejuvenation intervals produce worse security results than the system without VM migration scheduling. Figure 8 presents the normalized results for

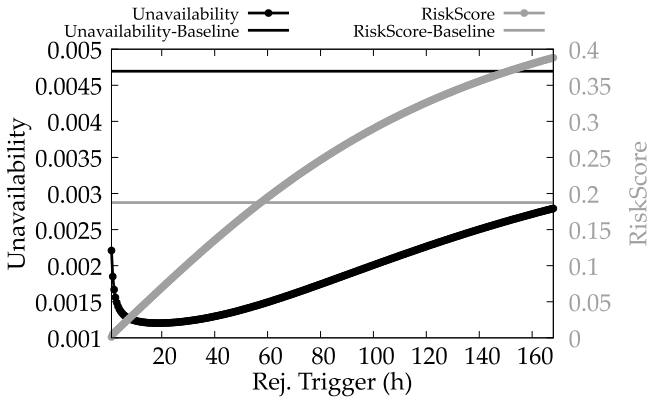

FIGURE 7. Case study 2 results - Denial of Service.

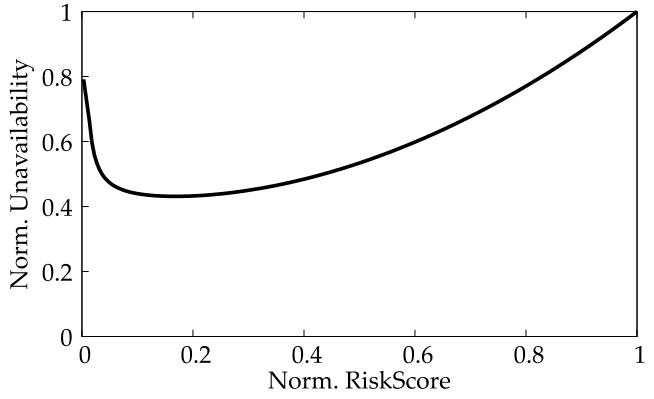

FIGURE 8. Case study 2 results - Denial of Service - Normalized plot.

both metrics, Unavailability and RiskScore. We computed the same set of scenarios of the previous case study. The results are presented in Table 5 (see the previous section for the meaning of the columns in the table).

We highlight three key conclusions from the results. First, the best overall rejuvenation policy, which globally minimizes both Unavailability and RiskScore, has a VM migration interval of 12 hours. Second, RiskScore raises with the increase of the VM migration interval. As the time that a VM spends migrating is substantially lower than the time that a VM spends running, the software aging accumulation is the most relevant aspect for the RiskScore metric in this scenario. More frequent migrations may produce network overhead, but, also avoid severe software aging accumulation. Therefore, more frequent migrations result in less software aging accumulation, and thus a lower RiskScore. The last conclusion is that the criteria focused on minimizing RiskScore produces more reduction when compared to the previous case study.

C. CASE STUDY 3 - COMPOSITION OF ATTACKS

In the previous case studies, we presented the analysis for the two threats separately (DoS and MITM). However, in some context, we may need to deal with both threats simultaneously. Therefore, we present a study case that combines the three metrics of interest: Unavailability, RiskScore of MITM (RS-MITM) and RiskScore of DoS (RS-DoS).

TABLE 5. Evaluation scenarios of Case study 2 - Denial of Service.

Scn #	Criteria	Rej. Trigger	UNAVAILABILITY	Downtime (h/yr)	RISKSCORE	Risk classification
0	*Optimal policy*	12 h	0.00122172	10.7023	0.041906	10.79% of MAXRS
1	100% UNAVAILABILITY	19 h	0.00120442	10.5507	0.06600	17% of MAXRS
2	75% UNAVAILABILITY AND 25% RISKSCORE	10.5 h	0.00123266	10.7981	0.03670	9.45% of MAXRS
3	50% UNAVAILABILITY AND 50% RISKSCORE	6.5 h	0.00129305	11.3271	0.02276	5.86% of MAXRS
4	25% UNAVAILABILITY AND 75% RISKSCORE	4 h	0.00139666	12.2347	0.01402	3.61% of MAXRS
5	100% RISKSCORE	1 hr	0.00221065	19.3652	0.00116	0.3% of MAXRS

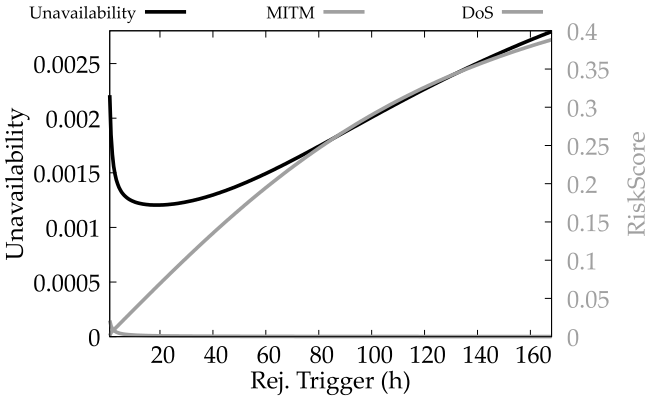

FIGURE 9. Case study 3 results - Composition.

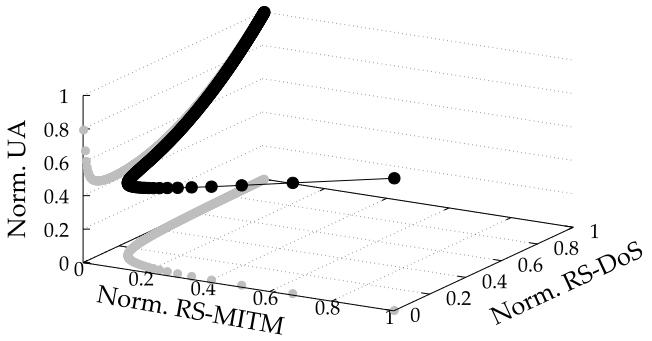

FIGURE 10. Case study 3 results - Composition - Normalized plot.

Figure 9 presents a plot with the results considering different rejuvenation policies. The plot basically merges the plots from Figures 5 and 7.

Figure 9 shows that the RS-MITM value is smaller than the RS-DoS value. RS-MITM considers only VM migration as a security risk. Therefore its absolute values are related to the steady-state probability that the system is performing a VM migration. In a stationary perspective, the time that the system spends on a migration state is lower than the time that it spends on a normal operation (i.e., running and accumulating software aging effects).

It is, however, hard to set up a direct comparison between both RiskScore metrics due to the different nature of the threats. Just comparing the absolute values of the two metrics may not be enough for a tradeoff analysis. For example, the consequences of a MITM attack may be more critical than the consequences of a DoS attack (depending on the business context; e.g., MITM attacks can try to affect system confidentiality, which can be devastating for some types of organizations). If we compare only the absolute values, we tend to neglect the influence of each threat in the overall system security. So, we normalized the data to provide a more fair tradeoff analysis. The normalization allows us to deal with a maximum and minimum range of our data. The normalization approach is the one used in the previous case studies. Figure 10 presents a 3d plot with the results for each normalized metric. We added shades (in gray) for each point of the 3d plot in the xy and yz planes.

Due to the different nature of the considered threats (MITM and DoS), we decided to neglect a general formula for RiskScore calculation for this particular case study. The general formula results may hide how much RiskScore is due to the MITM or DoS threats.

We can use the data from Figure 10 to find the optimal policy when considering Unavailability, RS-MITM and RS-DoS. We obtain the best overall policy by finding the point with the shortest distance for the origin (point (0,0,0)). We compute such a distance using the Euclidean Distance. We also present some alternative scenarios with different weight configurations for the three metrics. Table 6 presents the results. The columns **% UA** and **% MITM** and **% DoS** show the weight for each metric. The columns **MR-MITM** and **MR-DoS** represents the RiskScore reduction when compared to the maximum observed RiskScore of MITM and DoS, respectively. As in the previous case studies, we assume that the VM migration interval range between 1 and 168 hours. The remaining columns have the same meaning as presented in the previous cases.

We highlight the following from this study case: 1) the policy that globally reduces the three metrics has the VM migration interval of 13.5 hours; 2) results from scenarios 5 and 7 reveal that migration policies for RS-MITM and RS-DoS are incompatible, thus trying to reduce one metric will increase the other (the results presented offer relevant information regarding finding the best balance); and 3) an interesting approach for decision making in such complex scenarios is the definition of unacceptable service levels instead of desired service levels, which allows us to remove the worse solutions from the decision-making process.

V. RELATED WORK

Matos *et al.* [26] and Dantas *et al.* [27] present analytical models for cloud computing availability evaluation and

TABLE 6. Evaluation scenarios of Case study 3 - Composition.

Scn #	% UA	% MITM	% DoS	Rej. Trigger	UA	Downtime (h/yr)	RS-MITM	% MR-MITM	RS-DoS	% MR-DoS
0	*Optimal policy*			13.5 h	0.00121414	10.63587	0.00156355	7.41	0.04709	12.13
1	100%	0%	0%	19 h	0.00120442	10.55072	0.00111098	5.27	0.06600	17.00
2	50%	25%	25%	13 h	0.00121635	10.65523	0.00162367	7.70	0.04537	11.68
3	33.33%	33.33%	33.33%	12 h	0.00122172	10.70227	0.00175895	8.34	0.04191	10.79
4	25%	50%	25%	15.5 h	0.00120785	10.58077	0.00136182	6.46	0.05400	13.90
5	0%	100%	0%	168 h	0.00279284	24.46528	0.00012566	0.59	0.38828	100
6	25%	25%	50%	8.5 h	0.00125502	10.99398	0.00248305	11.77	0.02974	7.66
7	0%	0%	100%	1 h	0.00221065	19.36529	0.02108543	100	0.001165	0.30

sensitivity analysis. The papers deal with virtualized scenarios with and without redundancy. Their work provides a relevant background for our availability model design. However, none of these papers deal with software aging or security evaluation.

The papers from Machida *et al.* [13], [19] provide useful insights for our modeling framework design. Our previous papers [10], [20] were the first step of our research endeavor in availability evaluation of VM migration scheduling for VMM rejuvenation purposes. The model present in this research was based on our previous work published in [11], which was focused on power consumption evaluation. Different from all these papers, we are now focused on the security aspects of the virtualized system with VMM rejuvenation based on VM migration scheduling.

Nguyen *et al.* [28], [29] provide a comprehensive dependability modeling of virtualized data centers. Their work covers the reliability and availability of different scenarios and virtualized architectures. The presented analysis also covers secondary aspects as power consumption and cost. Besides that, the proposed models also include software aging and rejuvenation behaviors. Different from those authors, we present a security-oriented analysis to measure the RiskScore associated with rejuvenation policies based on VM migration.

Moving Target Defense (MTD) consists of dynamically changing the available attack surface to ''confuse'' or to nullify attackers' knowledge about the environment [30]. VM migration is among the MTD techniques used on Cloud Computing environments [31]. Alavizadeh *et al.* [32], [33] present a modeling framework for security evaluation of MTD on Cloud Computing environments. The authors use a Hierarchical Attack Representation Models (HARM) to evaluate metrics as System Risk, Attack Cost, Return on attack, and Availability. Their results comprise a combination of different MTD techniques. Although we are also considering VM migration in a security evaluation context, our goal is to evaluate the security impacts caused by VM migration scheduling for software rejuvenation purposes. Besides that, different from [32], [33], our models comprise software aging aspects.

Aung and Park [34] presents an approach to apply software rejuvenation as a security defense. The idea is to perform rejuvenation actions to kill attackers' processes and shutdown unauthorized connections. The paper presents experimental results based on attack free and attacks data sets. The paper neglects system availability evaluation.

TABLE 7. Related work comparison.

Papers	Availability Modeling	Software Aging and Rejuvenation	Security Evaluation
[26] [27]	Yes	No	No
[19] [13] [20] [10] [11]	Yes	Yes	No
[28] [29]	Yes, also with reliability	Yes	No
[32] [33]	Yes	No	Yes
[34]	No	Yes	Yes
This paper	Yes	Yes	Yes

Table 7 summarizes the scope of the related works above in contrast with the current paper.

VI. CONCLUSION AND FUTURE WORK

This paper presented a comprehensive evaluation of the availability and security of virtualized systems with VMM rejuvenation enabled by VM migration scheduling. Our results provide a tradeoff analysis in three case studies: 1) Man-in-the-middle (MITM), 2) Denial of Service (DoS), and 3) combination of MITM and DoS attacks. Each case study provides a set of scenarios comprising the optimal rejuvenation scheduling, and intermediary configurations of weights for security and availability, providing more information for the decision making process.

Our security evaluation approach is flexible and can consider other threat models and be used in other modeling frameworks (e.g., performance models and Markov Chains). The final results show that reducing the risk associated with the Man-in-the-middle attack is incompatible with reducing the risk related to Denial of Service, as policies reducing one tends to increase the other.

Regarding our main research question, RQ_{main} - *What is the security impact caused by applying different VM migration policies for VMM software rejuvenation purposes?*, We perceived that the security impact depends on the security threats considered. Specifically about our case studies, we highlight: i) when considering only MITM attacks, VM migration frequency determines the system RiskScore levels; ii) when considering only DoS attacks, the time spent on a state with software aging accumulation is the most critical factor for

system RiskScore; iii) when considering both threats, rejuvenation policies that reduce DoS RiskScore are preferred because the system tends to stay more time in a state with software aging accumulation than ''in migration''.

There are several future research directions. We aim to investigate other threat models as penetration attacks and specific domains of Denial of Service (e.g., buffer overflow or web server overload). We also aim at expanding the proposed model for more prominent system architectures (e.g., including multiple VMs). Besides that, we intend to gradually increase our model scope by adding more kinds of dependability and security events. A possible approach is to expand the models using the *interacting models* method. Future work may also consider including specific business models (e.g., a virtualized system of the financial market) in the modeling framework.

REFERENCES

[1] RightScale. (2018). *Rightscale 2018 State of The Cloud Report*. [Online]. Available: https://www.suse.com/media/report/rightscale_2018_state_of_the_cloud_report.pdf

[2] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto, and A. N. Toosi, ''A manifesto for future generation cloud computing: Research directions for the next decade,'' *ACM Comput. Surv.*, vol. 51, no. 5, p. 105, 2018.

[3] R. Matos, J. Araujo, V. Alves, and P. Maciel, ''Characterization of software aging effects in elastic storage mechanisms for private clouds,'' in *Proc. IEEE 23rd Int. Symp. Softw. Rel. Eng. Workshops*, Nov. 2012, pp. 293–298.

[4] D. L. Parnas, ''Software aging,'' in *Proc. 16th Int. Conf. Softw. Eng.*, May 1994, pp. 279–287.

[5] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, ''Software rejuvenation: Analysis, module and applications,'' in *25th Int. Symp. Fault-Tolerant Comput. Dig. Papers*, Jun. 1995, pp. 381–390.

[6] M. Torquato, J. Araujo, I. Umesh, and P. R. M. Maciel, ''Sware: A methodology for software aging and rejuvenation experiments,'' *J. Inf. Syst. Eng. Manage.*, vol. 3, no. 2, p. 15, 2018.

[7] M. Torquato, P. Maciel, J. Araujo, and I. M. Umesh, ''An approach to investigate aging symptoms and rejuvenation effectiveness on software systems,'' in *Proc. 12th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2017, pp. 1–6.

[8] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, ''A survey of software aging and rejuvenation studies,'' *ACM J. Emerg. Technol. Comput. Syst.*, vol. 10, no. 1, p. 8, 2014.

[9] R. Pietrantuono and S. Russo, ''A survey on software aging and rejuvenation in the cloud,'' *Softw. Qual. J.*, to be published.

[10] M. Melo, J. Araujo, R. Matos, J. Menezes, and P. Maciel, ''Comparative analysis of migration-based rejuvenation schedules on cloud availability,'' in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 4110–4115.

[11] M. Torquato, I. M. Umesh, and P. Maciel, ''Models for availability and power consumption evaluation of a private cloud with VMM rejuvenation enabled by VM live migration,'' *J. Supercomput.*, vol. 74, no. 9, pp. 4817–4841, Sep. 2018.

[12] J. Oberheide, ''Exploiting live virtual machine migration,'' in *Proc. Black Hat DC Briefings*, Washington, DC, USA, Feb. 2008.

[13] F. Machida, D. S. Kim, and K. S. Trivedi, ''Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration,'' *Perform. Eval.*, vol. 70, no. 3, pp. 212–230, 2013.

[14] A. Avizienis, J.-C. Laprie, and B. Randell, ''Fundamental concepts of dependability,'' Dept. Comput. Sci., Univ. Newcastle upon Tyne, Newcastle upon Tyne, U.K., Tech. Rep. CS-TR-739, 2001.

[15] J. Gray and D. P. Siewiorek, ''High-availability computer systems,'' *Computer*, vol. 24, no. 9, pp. 39–48, Sep. 1991.

[16] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, ''Live migration of virtual machines,'' in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implement.*, vol. 2, May 2005, pp. 273–286.

[17] M. Torquato and M. Vieira, ''Interacting srn models for availability evaluation of VM migration as rejuvenation on a system under varying workload,'' in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2018, pp. 300–307.

[18] G. Levitin, L. Xing, and H. Ben-Haim, ''Optimizing software rejuvenation policy for real time tasks,'' *Rel. Eng. Syst. Saf.*, vol. 176, pp. 202–208, Aug. 2018.

[19] F. Machida, D. S. Kim, and K. S. Trivedi, ''Modeling and analysis of software rejuvenation in a server virtualized system,'' in *Proc. IEEE 2nd Int. Workshop Softw. Aging Rejuvenation*, Nov. 2010, pp. 1–6.

[20] M. Melo, P. Maciel, J. Araujo, R. Matos, and C. Araújo, ''Availability study on cloud computing environments: Live migration as a rejuvenation mechanism,'' in *Proc. 43rd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2013, pp. 1–6.

[21] A. Zimmermann, ''Modelling and performance evaluation with TimeNET 4.4,'' in *Proc. Int. Conf. Quant. Eval. Syst.* Cham, Switzerland: Springer, 2017, pp. 300–303.

[22] D. S. Kim, F. Machida, and K. S. Trivedi, ''Availability modeling and analysis of a virtualized system,'' in *Proc. 15th IEEE Pacific Rim Int. Symp. Dependable Comput.*, Nov. 2009, pp. 365–371.

[23] CAPEC. *Capec-94: Man in the Middle Attack*. Accessed: Aug. 19, 2019. [Online]. Available: https://capec.mitre.org/data/definitions/94.html

[24] M. Conti, N. Dragoni, and V. Lesyk, ''A survey of man in the middle attacks,'' *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2027–2051, 3rd Quart., 2016.

[25] S. Subashini and V. Kavitha, ''A survey on security issues in service delivery models of cloud computing,'' *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, 2011.

[26] R. D. S. Matos, P. R. M. Maciel, F. Machida, D. S. Kim, and K. S. Trivedi, ''Sensitivity analysis of server virtualized system availability,'' *IEEE Trans. Rel.*, vol. 61, no. 4, pp. 994–1006, Dec. 2012.

[27] J. Dantas, R. Matos, J. Araujo, and P. Maciel, ''Eucalyptus-based private clouds: Availability modeling and comparison to the cost of a public cloud,'' *Computing*, vol. 97, no. 11, pp. 1121–1140, Nov. 2015.

[28] T. A. Nguyen, D. Min, E. Choi, and T. D. Tran, ''Reliability and availability evaluation for cloud data center networks using hierarchical models,'' *IEEE Access*, vol. 7, pp. 9273–9313, 2019.

[29] T. A. Nguyen, D. Min, and E. Choi, ''A comprehensive evaluation of availability and operational cost for a virtualized server system using stochastic reward nets,'' *J. Supercomput.*, vol. 74, no. 1, pp. 222–276, 2018.

[30] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, vol. 54. New York, NY, USA: Springer, 2011.

[31] S.-J. Moon, V. Sekar, and M. K. Reiter, ''Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration,'' in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1595–1606.

[32] H. Alavizadeh, J. B. Hong, J. Jang-Jaccard, and D. S. Kim, ''Comprehensive security assessment of combined MTD techniques for the cloud,'' in *Proc. 5th ACM Workshop Moving Target Defense*, Oct. 2018, pp. 11–20.

[33] H. Alavizadeh, J. Jang-Jaccard, and D. S. Kim, ''Evaluation for combination of shuffle and diversity on moving target defense strategy for cloud computing,'' in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 573–578.

[34] K. M. M. Aung and J. S. Park, ''Software rejuvenation approach to security engineering,'' in *Proc. Int. Conf. Comput. Sci. Appl.* Berlin, Germany: Springer, 2004, pp. 574–583.

MATHEUS TORQUATO received the bachelor's degree in computer science from the Federal University of Alagoas and the master's degree in computer science from the Federal University of Pernambuco. He also has a certificate in computer networks from the Federal Institute of Alagoas. He is currently on leave from his teaching activities with the Federal Institute of Alagoas, Arapiraca, to pursue the Ph.D. degree with the University of Coimbra. He already involved in building and managing of Cloud Computing Private Environments. His research interests comprise subjects like cloud computing, performance and dependability evaluation, computer networks, and security. His website is http://www.matheustorquato.com

PAULO MACIEL received the degree in electronic engineering, in 1987, and the M.Sc. and Ph.D. degrees in electronic engineering and computer science from the Federal University of Pernambuco, Recife, Brazil. He was a Faculty Member with the Department of Electrical Engineering, Pernambuco University, Recife, Brazil, from 1989 to 2003. Since 2001, he has been a member of the Informatics Center, Federal University of Pernambuco, where he is currently an Associate Professor. In 2011, during his sabbatical from the Federal University of Pernambuco, he stayed with the Department of Electrical and Computer Engineering, Edmund T. Pratt School of Engineering, Duke University, Durham, NC, USA, as a Visiting Professor. His current research interests include performance and dependability evaluation, Petri nets and formal models, encompassing manufacturing, embedded, computational, and communication systems as well as power consumption analysis. He is a Research Member of the Brazilian Research Council.

MARCO VIEIRA received the Ph.D. degree from the University of Coimbra (UC), Coimbra, Portugal, in 2005, where he is currently a Full Professor. His research interests include dependability and security assessment and benchmarking, fault injection, software processes, and software quality assurance, subjects in which he has authored or coauthored more than 200 articles in refereed conferences and journals. He has participated and coordinated several research projects, both at the national and European level. He has served on program committees of the major conferences of the dependability area and acted as a Referee for many international conferences and journals in the dependability and security areas.