I Know You Are Watching Me: Stackelberg-based Adaptive Intrusion Detection Strategy for Insider Attacks in the Cloud

Omar Abdel Wahab*, Jamal Bentahar*, Hadi Otrok†*, Azzam Mourad‡

*Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada
†Department of ECE, Khalifa University, Abu Dhabi, UAE
‡Department of Computer science and Mathematics, Lebanese American University, Beirut, Lebanon
Email addresses: {o_abul, bentahar}@ciise.concordia.ca, Hadi.Otrok@kustar.ac.ae, azzam.mourad@lau.edu.lb

***Abstract*—Insider attacks in which misbehaving Virtual Machines (VMs) take part of the cloud system and learn about its internal vulnerabilities constitute a major threat against cloud resources and infrastructure. This demands setting up continuous and comprehensive security arrangements to restrict the effects of such attacks. However, limited security resources prohibit full detection coverage on all VMs at all times, which can be exploited by attackers to examine the selective detection strategies and adjust their own attack plans accordingly. Motivated by the absence of any approach that accounts for such a challenge in the domain of cloud computing, we propose in this work an adaptive detection strategy that formulates a Stackelberg security game to enable the cloud system to optimally exploit its available amount of security resources to maximize the detection of distributed attacks, knowing that attackers have the ability to monitor the cloud system's strategies and adjust their own attack plans. Experiments carried out on the CloudSim framework reveal that the proposed solution maximizes the detection of distributed attacks and minimizes false negatives and positives compared to a maximin-based detection strategy, while being scalable to the increase in both the number of co-hosted VMs and percentage of co-resident attackers.**

***Keywords*-Load distribution; intrusion detection; Stackelberg game theory; limited resources; virtualized cloud; security.**

I. Introduction

Security concerns have accompanied the notion of cloud computing since its inception until now. On the one hand, security is crucial to guarantee in cloud systems since cloud data centres are supposed to provide a safe and secure environment for users/providers to host their data/resources. On the other hand, cloud-based systems are exposed to a wide set of security threats; even more than those that target traditional computing systems because of the cloud's virtual and elastic properties [1], [2]. The sources of such threats might come either from outsider or insider parties. While outsider attacks are carried out by malicious individuals/entities having no direct access to the cloud system's components, insider attacks are executed alas by malicious entities having authorized access to the system's infrastructure and resources. This makes the latter type of attacks more painful in terms of entailed damage and harder to capture and detect. Therefore, we focus in this work on the problem of detecting insider attacks that occur at the cloud's virtualization layer and consider an attack model in which malicious (or compromised) co-hosted Virtual Machines (VMs) collaborate together to create distributed attacks with the aim of breaching the security of the host cloud system. For instance, such misbehaving VMs might take advantage of the VM exit operations/hypercalls to collectively inject malicious code fragments resulting in crashing or slowing down their host hypervisor [3].

Problem Statement. Although plenty of Intrusion Detection Systems (IDSs) targeting cloud-based applications can be found in the literature, most of these techniques are developed and upgraded from traditional detection techniques used in non-cloud environments, which limits their effectiveness when applied in a cloud setting [1]. In fact, the existing detection systems can be classified into three main branches: network-based, host-based, and hypervisor-based IDSs [4]. Network-based systems put the monitoring agents at the network's level to monitor the circulating traffic and recognize any malicious behavior. The fact that these systems operate at the network's layer only makes them unable to catch insider attacks that sneak into the internal virtualized system. To remedy this shortcoming, host-based IDSs deploy the monitoring agents at the VMs' layer to monitor their activities and report any abnormal behavior. The main limitation of this approach lies in the burdens it puts on the users who are required to spend their own resources and efforts to maintain the health of the monitoring agents. To alleviate these burdens, hypervisor-based systems place the monitoring agents at the cloud system's layer and assign the host hypervisors with the role of observing the VMs' system metrics and identifying malicious activities.

Unfortunately, all of the three branches of IDSs suffer from two essential problems that limit their performance in practical cloud systems. Firstly, they are based on the simplistic idea of monitoring and analyzing events without accounting for the malicious strategies of the attackers that take advantage of the cloud's virtual and elastic features to perplex the detection system and complicate the detection

process. This raises the need for elaborating an up-to-date intelligent detection technique that considers the strategies of the attackers in its design to increase the awareness of the cloud system and enable it to cope with complex attack scenarios. Secondly, the current IDSs do not explain how to deal with the cloud system's limited resources problem in the detection process; thus assuming (directly or indirectly) that the cloud system is able to provide permanent and full detection coverage on all its nodes. However, it is no secret that the magnitude of resources that can be devoted to intrusion detection is bounded by a certain budget that is determined in such a way that does not affect the portion of resources consecrated to serving clients. This necessitates thinking of a resource-aware selective detection strategy that distributes the cloud system's detection load among the different VMs in such a way that respects the limited security resources' budget and maintains at the same time an optimal detection effectiveness.

Recently, a selective detection load distribution strategy [1] that accounts for the attackers' strategies and resources constraints has been proposed by our research group to maximize the detection of distributed malicious attacks in the cloud. Although this technique has shown to be able to improve the detection compared to the state-of-the-art detection systems, the main shortcoming of this solution (as is the case for the rest of existing solutions) lies in its simplistic attack scenario which supposes that attackers launch their attacks without having prior knowledge of the intrusion detection arrangements adopted by the cloud system. Albeit such an assumption might hold for some limited time, attackers are becoming smart enough to observe the cloud system's detection strategies over the time and adjust their own attack strategies accordingly in order to complicate the detection process.

Contributions. Motivated by this challenge, we design in this work an adaptive intrusion detection strategy that allows the cloud system to optimally distribute the detection load among its guest VMs, while taking into account the fact that attackers are observing this detection strategy and adjusting their own attack strategies accordingly. The solution is modeled as Stackelberg game [5] in which the cloud system takes the role of the *leader* and attackers are the *followers*. The strategy of the attackers is to create distributed malicious attacks by choosing the optimal distribution of the attacks over the set of VMs co-hosted on a single cloud infrastructure after observing the detection plans adopted by the cloud system so as to minimize the chances of being detected. Knowing this fact, the strategy of the cloud system is to choose the optimal distribution of the detection load among VMs in such a way to maximize the likelihood of detection, while respecting the limited security resources constraints. The game is converted into a Mixed-Integer Linear Programming (MILP) and solved using the backward induction reasoning [6] by computing first the best responses of the attackers to the cloud system's detection strategy and integrating then this knowledge into the cloud system's optimization problem. Intuitively, this means that the cloud system is aware that attackers will play their best responses to its detection strategy and incorporates this knowledge into its detection load distribution optimization problem to maximize the detection efficacy. The outcome of the game is the optimal detection load distribution strategy over the set of co-hosted VMs that best randomizes the detection process in real-time and makes it unpredictable for attackers. In summary, the main contributions of this work are:

- Proposing an adaptive intrusion detection strategy that allows the cloud system to optimally distribute the detection load among its co-resident VMs. To the best of our knowledge, this strategy is the first in the domain of cloud computing that accounts for the fact that attackers have the ability to observe the cloud system's detection strategy and adapt their own attack strategies to complicate the detection process.
- Modeling the problem as a sequential Stackelberg game in which the cloud system acts first and then do the attackers after having examined the former's move. This allows us to represent a complex and realistic attack scenario wherein each party is aware of the other party's strategies and contributes hence in broadening the learning space of the cloud system in deciding about its detection strategies.

The performance of the proposed solution is tested using the CloudSim framework [7] and compared with a recent maximin-based intelligent intrusion detection strategy. Experimental results reveal that our solution is able to maximize the detection of distributed attacks and minimize the false positives and negatives, while being resilient to the increase in both the number of co-hosted VMs and percentage of attacking VMs.

Paper Outline. Section II presents an overview of the existing intrusion detection systems in the field of cloud computing. Section III formulates the problem and devises the utility functions of the cloud system and attackers. In Section IV, we characterize the Stackelberg game between the cloud system and attackers as a MILP problem and demonstrate how to solve it using the backward induction reasoning. In Section V, we conduct experiments to validate the performance of our solution. Finally, we recapitulate the main insights of the paper in Section VI.

II. Related Work

As explained in our previous work [4], the main intrusion detection techniques proposed for cloud-based systems can be classified into three major categories: network-based, host-based, and hypervisor-based systems.

A. Network-based IDSs

In [8], the authors discuss an intrusion detection framework that monitors network traffic using a cluster-based architecture to support multiple security domains. The basic idea is to export the intra-VM network traffic to be processed by a physical IDS. Moreover, a traffic deduplication technique is advanced to remove redundant network traffic and minimize the overhead.

In [9], a customer-controllable on-demand Intrusion Detection System as a Service (IDSaaS) framework is introduced. The network interactions among VMs within a pre-defined virtual network are monitored and suspicious activities are registered and analyzed. The performance of the proposed framework is adaptable based on the volume of traffic load in the network, where, for example, the number of IDS components can be adjusted on the basis of the amount of traffic circulating inside the target network.

In [10], a cooperative detection technique for Denial of Service (DoS) attacks is discussed. The main idea is to deploy a monitoring agent in each cloud region to gather and analyze packets. The type of the collected packets is compared against a block table which registers the malignant packets that should be blocked. If the type of a given packet is found in the table, then the packet is directly dropped. Otherwise, the gravity of the suspicious packets is verified, where seriously malignant packets are dropped, slightly malignant packets are ignored, and moderately malignant packets undergo outliers detection and data clustering processes to make the appropriate decision.

B. Host-based IDSs

In [11], a multi-tier detection model for large-scale cloud environments called *Varanus* is proposed. The basic idea is to split VMs into a set of groups using the k-nearest neighbor algorithm on the basis of the similarity among their configuration settings (e.g., database servers, Web servers). Thereafter, the VMs belonging to the same cluster exchange their monitoring information and the under-utilized VMs are selected to analyze the whole collected data.

A distributed detection technique is advanced in [12] to identify Distributed Denial of Service (DDoS) attacks in the cloud. To this end, an IDS is deployed within each VM to monitor activities and gather alerts. These alerts coming from different VMs are forwarded and stored then into a Mysql database and converted into basic probability assignments to be analyzed using Dempster-Shafer's rule.

In [13], the authors discuss a host-based intrusion detection technique that selectively monitors (only) the failed system call traces of the VMs. These traces are then analyzed and classified either normal or malicious using the k-nearest neighbor classification algorithm. Finally, users are alerted of any malicious activity in their system.

C. Hypervisor-based IDSs

In [14], the authors propose an online anomaly detection technique that operates at the hypervisor's layer. The system architecture consists of four main components, namely the Cloud Resilience Manager (CRM), System Resilience Engine (SRE), Network Analysis Engine (NAE), and System Analysis Engine (SAE). At the first stage, the CRM deployed on each cloud node collects features from the VMs and their local networks and sends this data to the NAE and SRE components. These two latter components employ the one-class Support Vector Machine classification technique to carry out a local anomaly detection. Based on the output generated by the classifier, the SRE takes the responsibility of recovering actions.

The authors in [15] advanced *Collabra*, a distributed IDS that is integrated into Xen hypevisors to preserve the security of the cloud system. Collabra scans each hyper-call made by every application of the VMs to guarantee the integrity of the cloud infrastructure and ensure fail-safe transaction processes. Collabra performs in a collaborative fashion in the sense that it embeds communication protocols that allow it to make contact with its instances deployed on other hypervisors to exchange information and enhance the results of the real-time detection.

Lombardi and Di Pietro propose in [16] a virtualization-supported detection technique called Advanced Cloud Protection System (ACPS). In ACPS, the system-call invocations of the VMs are constantly watched by an entity situated in the kernel space of the host called *Interceptor*. The suspected activities are then registered and prioritized into a *Warning Pool*. Finally, the *Evaluator* entity inspects these activities to make the appropriate decision on whether there exists a security threat or not.

In summary, the existing IDSs suffer from two main limitations that make them insufficient to deal with practical cloud-based systems. In the first place, they totally ignore the attackers' strategies in the design of the detection system, which minimizes their chances of capturing sophisticated attacks. In the second place, they do not explain how the proposed detection techniques can work under a limited budget of security resources, which restricts their effectiveness in realistic resource-constrained applications. In a recent work [1], a maximin-based detection load distribution strategy has been developed by our research group. Specifically, a maximin game is modeled between the attackers trying to minimize the cloud system's detection probability by distributing their attacks over a set of VMs and the hypervisor trying to maximize this minimization by optimally distributing the detection load among VMs. Similar to our current work, this work takes into consideration the strategies of the attackers in the design of the detection system and is able to work using a limited budget of security resources. Beyond this work, our solution is able to deal with a more complex

attack scenario wherein attackers are able to observe the cloud system's detection strategies and adapt their attack plans accordingly.

III. Problem Formulation

Our system model consists of a set of virtual machines $V = \{v_1, v_2, ..., v_k\}$ hosted on a shared hypervisor. Note that when $i \in k$ can be understood from the context, we simply use v instead of v_i. These VMs might be either well-behaving or attacking [17]. Well-behaving VMs are those that aim at doing their jobs smoothly without having the intention to harm neither the cloud system nor other VMs. On the other hand, attacking VMs seek to harm the cloud system and/or other co-hosted VMs by continuously and collaboratively sending malicious code fragments to form up distributed malicious attacks. Such VMs might be either (1) malicious in case their real owners create the attacks or (2) compromised in case the source of attacks is a third party who manipulates VMs and injects his/her malicious code through them[1]. Knowing this fact, the cloud system has to find the optimal detection strategy that maximizes the detection of such attacks. To do so, the hypervisor, acting on behalf of the cloud system, has a specific amount of resources R that comprises both the amount R_c of resources to be dedicated to serving clients and the amount R_d of resources to be dedicated for intrusion detection such that $R = R_c + R_d$. Thus, the objective of the hypervisor becomes finding the optimal detection load distribution strategy that maximizes the detection of distributed attacks, while respecting the budget R_d of resources. We model this situation as a Stackelberg security game of two players, i.e., hypervisor and attackers. The game is played sequentially in the sense that the hypervisor representing the *leader* of the game commits first to a certain detection load distribution strategy and then attackers (*followers*) choose their attack distribution strategy after having observed the hypervisor's move.

Formally, the hypervisor's set of pure strategies consists of a (sub)set $v_d \in V$ of VMs to put the detection load on (normally the whole set of VMs would be selected to maximize the chances of capturing attacks, i.e., $v_d = V$). To rip off and confuse attackers, the hypervisor would select a mixed strategy (i.e., probability distributions) at each time moment x belonging to the fixed interval of time $[t_1, t_2]$. The mixed strategy consists of the vector $H_x(V) = (h_x(v_1), \ldots, h_x(v_k))$ over the set V of VMs hosted on top of the hypervisor such that $\sum_{v_i \in V} h_x(v_i) = 1$. In this way, the hypervisor would assign a different detection load probability to each of its VMs. The attackers observe the hypervisor's detection load distribution strategy and then decide about their own strategies consisting also of a (sub)set $v_a \in V$ of VMs through which the attack is to be launched. In their turn, attackers might choose a mixed strategy at time moment $x \in [t_1, t_2]$ consisting of the vector $A_x(V) = (a_x(v_1), \ldots, a_x(v_k))$ over the set V of VMs to attack through such that $\sum_{v_i \in V} a_x(v_i) = 1$. In this way, each attacker would pick a VM $v \in V$ at time x with a probability of $a_x(v) \in A_x(V)$ to attack through so as to confuse the cloud system. Note that attackers may decide not to attack through any VM at a certain time moment.

[1]In the rest of the paper, the term *attacker* is used to refer to both cases.

Based on the strategies adopted by both the hypervisor and attackers, a payoff is assigned to each of these parties. Particularly, when the hypervisor selects the pure strategy i and the attacker selects the pure strategy j, the hypervisor receives a payoff of U_{ij} and the attacker receives a payoff of Q_{ij}. The values of these payoffs are influenced by two main factors: (1) the likelihood of the detection strategy in capturing the attack on a particular VM; and (2) the damage caused by the attacker assaulting through a particular VM.

Specifically, the hypervisor receives the following payoff at time $t_2 + 1$ (current system time based on the interval $[t_1, t_2]$):

$$U_{ij}(t_2 + 1) = \beta([t_1, t_2]) \times w(v) - mon(v) \tag{1}$$

This payoff represents the success of the hypervisor in protecting its virtual machine v of worth $w(v)$ (the worth of a VM depends on its price as well as the criticality of the applications running in it) minus the cost $mon(v)$ of monitoring v. Since the success of detection depends heavily on the IDS's detection probability as mentioned earlier, the payoff of the hypervisor is weighed based on its average detection rate $\beta([t_1, t_2])$ during the time window $[t_1, t_2]$, which is computed as per Eq. (2).

$$\beta([t_1, t_2]) = 1 - \sum_{x=t_1}^{t_2} \sum_{v \in V} \frac{(a_x(v) - h_x(v))}{t_2 - t_1} \text{ for each } a_x(v) > h_x(v), \tag{2}$$

Note that all the calculations in the rest of the paper are done at time $t_2 + 1$ (i.e., the current time for the hypervisor and attackers). Thus, we simplify the notation and use U_{ij} (respectively Q_{ij}) instead of $U_{ij}(t_2 + 1)$ ($Q_{ij}(t_2 + 1)$) when referring to hypervisor's (attacker) utility at time $t_2 + 1$.

On the other hand, the attacker's payoff would be:

$$Q_{ij} = w(v) - att(v) \tag{3}$$

This payoff quantifies the gain of the attacker from exploiting the VM of worth $w(v)$ minus the cost $att(v)$ incurred by attacking v.

IV. Stackelberg-beased Adaptive Detection Load Distribution Strategy

We formulate in this section the intrusion detection problem as a Stackelberg security game between the cloud system and attackers. Practically, the hypervisor (acting on behalf of the cloud system) plays the role of the game leader and makes the first move by choosing its detection load distribution strategy over VMs, whereas attackers are the followers that observe the leader's strategy and choose

their best responses to it in terms of attack distribution strategies. The game is modeled first as Mixed-Integer Quadratic Program (MIQP) and then converted into a Mixed-Integer Linear Program (MILP) to be solved using a linear programming solver tool. The backward induction reasoning is employed to determine the optimal strategies of both the cloud system and attackers. This is done by first deriving the best response of the attackers to a (fixed) observed strategy of the cloud system and then integrating this best response to the cloud system's optimization problem to help it select the optimal detection load distribution strategies. Intuitively, this means that the cloud system anticipates that attackers will play their best responses to its (observed) detection load distribution strategy and embeds this knowledge into its optimization problem to select the optimal detection load distribution strategy using this information.

Let L and F denote the index sets of the hypervisor (leader) and attacker's (follower) pure strategies, respectively. Let l represent a vector of the hypervisor's pure strategies (a.k.a hypevisor's policy) and f represent a vector of the attacker's pure strategies (a.k.a attacker's policy). Thus, the value l_i would represent the proportion of times in which the hypervisor plays the pure strategy i from its policy set. Similarly, the value f_j would represent the proportion of times in which the attacker plays the pure strategy j from its policy set. Based on the strategies chosen by both the hypervisor and attacker and as explained in Section III, U and Q represent the payoff matrices such that U_{ij} is the gain of the hypervisor and Q_{ij} is the gain of the attacker when the hypervisor selects the pure strategy i and the attacker selects the pure strategy j.

Let us fix first the hypervisor's policy to a certain policy l. After observing l, the attacker needs to solve the following linear programming optimization problem in order to determine its optimal response to l:

$$\begin{aligned} \text{maximize} \quad & \sum_{j\in F}\sum_{i\in L} Q_{ij}\times f_j \times l_i & \\ \text{subject to} \quad & \sum_{j\in F} f_j = 1, & \forall j \in F \\ & f_j \in [0\cdots 1], & \forall j \in F \\ & f_j \geq 0 & \forall j \in F. \end{aligned} \tag{4}$$

Knowing the fixed strategy l of the leader, the best response $f_j(l)$ of the attacker should yield a non-negative utility to the attacker, which means that Problem (4) has to satisfy the following constraint:

$$f_j \times \sum_{i\in L} Q_{ij}\times l_i \geq 0, \quad \forall j \in F \tag{5}$$

Moreover, given that $f_j(l)$ is the attacker's best response strategy, any deviation from this strategy (i.e., $1-f_j$) would lead the attacker to undergo a loss in terms of utility. Thus, Problem (4) has to satisfy as well the following constraint:

$$(1-f_j) \times \sum_{i\in L} Q_{ij}\times l_i \leq 0, \quad \forall j \in F \tag{6}$$

Let's move now to the cloud system's side. The hypervisor, knowing that the attacker will play its best response $f_j(l)$ to every hypervisor's strategy l, incorporates this knowledge into its optimization problem to determine the solution l that maximizes its own payoff. Thus, the hypervisor has to solve the following problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i\in L}\sum_{j\in F} U_{ij}\times f_j(l) \times l_i & \\ \text{subject to} \quad & \sum_{i\in L} l_i = 1, & \forall i \in L \\ & l_i \in [0\cdots 1], & \forall i \in L \\ & l_i \geq 0 & \forall i \in L. \end{aligned} \tag{7}$$

Problem (7) can be completed by incorporating the characterization of $f_j(l)$ depicted in Problem 4 and Eqs. (5) and (6). Taking into account the fact that given any optimal mixed strategy $f_j(l)$, then all the pure strategies in its support are also optimal [18], we can consider only the optimal pure strategies of the attacker (which exist always) and symbolize the optimal pure strategies using binary variables. Thus, the hypervisor's problem becomes:

$$\begin{aligned} \underset{l,f}{\text{maximize}} \quad & \sum_{i\in L}\sum_{j\in F} U_{ij}\times l_i \times f_j & \\ \text{subject to} \quad & \sum_{i\in L} l_i = 1, & \\ & \sum_{j\in F} f_j = 1, & \\ & (1-f_j) \times \sum_{i\in L} Q_{ij}\times l_i \leq 0, & \forall j \in F \\ & f_j \times \sum_{i\in L} Q_{ij}\times l_i \geq 0, & \forall j \in F \\ & l_i \in [0\cdots 1], & \forall i \in L \\ & f_j \in \{0,1\}, & \forall j \in F \end{aligned} \tag{8}$$

In Problem (8), the first and fifth constraints compel a feasible mixed policy for the hypervisor, whereas the second and sixth constraints compel a feasible pure strategy for the attacker. The sixth constraint restricts as well the actions' vector of the attacker to be a pure distribution over F. The third and fourth constraints force the best response $f_j(l)$ to be optimal for the attacker in terms of gained utility. Note that Problem (8) is an integer program with a non-convex quadratic objective [18]. Thus, the final step is to convert the Mixed-Integer Quadratic Programming (MIQP) at hand into a Mixed-Integer Linear Programming (MILP) by removing the non-linearity of the objective function. This can be achieved by assigning the value of $l_i \times f_j$ to a new variable z_{ij}. Thus, the problem becomes:

$$
\begin{aligned}
\underset{l,f}{\text{maximize}} \quad & \sum_{i\in L}\sum_{j\in F} U_{ij}\times z_{ij} \\
\text{subject to} \quad & \sum_{i\in L}\sum_{j\in F} z_{ij} = 1 \\
& f_j \le \sum_{i\in L} z_{ij} \le 1 \\
& \sum_{j\in F} f_j = 1 \\
& (1-f_j)\times\sum_{i\in L} Q_{ij}\times z_{ij} \le 0 \qquad \forall j\in F \\
& f_j \times \sum_{i\in L} Q_{ij}\times z_{ij} \ge 0 \qquad \forall j\in F \\
& z_{ij}\in[0\cdots 1], \qquad \forall i\in L, j\in F \\
& f_j \in \{0,1\}, \qquad \forall j\in F
\end{aligned}
\tag{9}
$$

Having linearized the problem, the MILP in Problem (9) can be now solved using a linear programming solver tool (e.g., simplex) to derive the optimal mixed strategies L_V and F_V of both the cloud system and attackers respectively [19]. For example, a possible optimal mixed strategy for the cloud system over the set $V = \{v_1, v_2, v_3\}$ of VMs could be to assign 35% of the detection load to v_1, 25% to v_2, and 40% to v_3, when attackers are attacking 30% of times over v_1, 30% of times over v_2, and 40% over v_3.

V. Experimental Results

In this section, we describe the experimental setup and present experimental results.

A. Experimental Setup

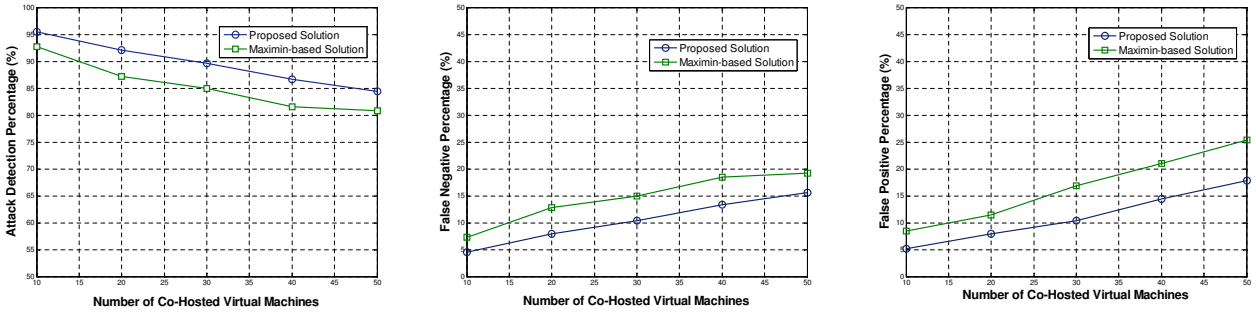
To perform experiments, we create our own cloud datacenter using the CloudSim framework [7] that allows us to simulate realistic cloud characteristics including large-scale co-hosted virtualized services, network connections among cloud constituents, and resources allocation policies. The datacenter consists of five physical machines; each hosting a number of VMs varying from 10 to 50. Each VM has an image size of 10000 MB. The physical machines are supplied with five CPU cores of 1000 Millions of Instructions Per Second (MIPS) each. The memory RAM capacity on each machine is of 16 GB and the hard drive storage is of 1000000 MB. The datacenter's network is equipped with a bandwidth portion of 50 Mbps. The physical resources are distributed among VMs initially in an equal manner so that each VM receives an amount of memory equal to 512 MB, a network bandwidth share of 1000 Kbit/s, and a processor speed of 1000 MIPS. Xen has been adopted as virtualization architecture in the created datacenter. The proposed solution is compared with a recent maximin-based solution that models a maximin game to help the cloud system find the optimal detection load distribution strategy; but does not account for the fact that attackers have the ability to monitor the cloud system's strategies and adjust their own strategies. The performance of these two solutions is studied under two distinct scenarios: (1) increase in the number of co-hosted VMs; and (2) increase in the percentage of co-resident attacking VMs. Note that we do not show and discuss the comparison results between our solution and the fair allocation strategy (the commonly used resources distribution technique in cloud computing) for space constraints. However, experiments conducted in our previous work [1] reveal that the maximin-based solution already outperforms the fair allocation approach. Thus, we believe that it is sufficient to cut across and present only the comparison results with the maximin-based approach.

B. Experimental Results

As mentioned earlier, the performance of the two studied solutions is tested in two different scenarios: increase in the number of VMs and increase in the percentage of attacking VMs. In the first scenario, we vary the number of VMs co-hosted on a single cloud system from 10 to 50 to examine how the performance of our model and the maximin-based model would be affected by this increase. We observe from Fig. 1 that the detection performance of the two models (in terms of attack detection, false negative, and false positive percentages) keeps decreasing as the number of deployed VMs increases. This result is expected since augmenting the number of co-hosted VMs on one cloud system increases the attack space of attackers by enabling them to have a larger number of VMs to distribute their attacks over. Moreover, increasing the number of VMs requires the cloud system to allocate additional resources to the intrusion detection duties and diminishes hence the efficacy of the available security budget in filling the detection needs. Though, the results in Fig. 1 demonstrate that our solution remains more scalable to an increased number of co-hosted VMs compared to the maximin-based model.

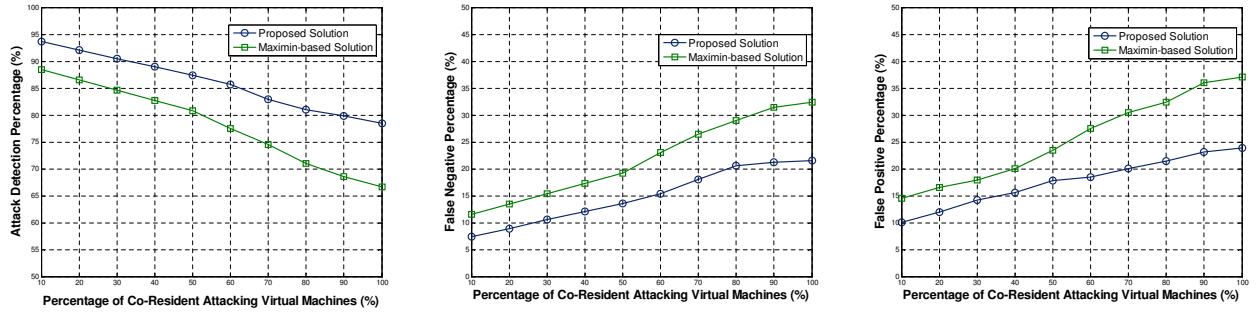
We measure in Fig. 1a the attack detection percentage which represents the percentage of attacks captured by the IDS and is computed as per Eq. (2). The results reveal that our solution is able to maximize the detection of distributed attacks up to $\approx 5\%$ compared to the maximin-based model. This is because our model accounts for the fact that attackers are able to observe the detection strategies of the cloud system and incorporates this information into the latter's optimization problem to help it determine the optimal detection load distribution strategy. Practically, although the maximin-based solution accounts for the strategies of the attackers when determining the optimal detection strategy, this model assumes that these attackers act without having prior knowledge about the cloud's system detection plan, which limits the learning space of the cloud system upon deciding about its detection load distribution strategy.

In Fig. 1b, we study the percentage of false negatives which represents the percentage of attacks that the IDS was not able to recognize as such. This percentage is computed



(a) Attack Detection (b) False Negatives (c) False Positives

Figure 1: Our solution improves the percentage of detected attacks and minimizes the percentages of false positives and negatives compared to the maximin-based solution, while being resilient to the increase in the number of co-hosted VMs.



(a) Attack Detection (b) False Negatives (c) False Positives

Figure 2: Our solution improves the percentage of detected attacks and minimizes the percentages of false positives and negatives compared to the maximin-based solution, while being resilient to the increase in the percentage of co-resident attackers.

by subtracting the detection load probability distributions of the cloud system from the attack probability distributions of the attacker when the values of the former are smaller. Eq. (10) shows how to compute the false negative rate $\alpha([t_1,t_2])$ during the time window $[t_1,t_2]$:

$$\alpha([t_1,t_2]) = \sum_{x=t_1}^{t_2} \sum_{v \in V} \frac{(a_x(v) - h_x(v))}{t_2 - t_1} \text{ for each } a_x(v) > h_x(v). \quad (10)$$

The results in Fig. 1b demonstrate that our solution is able to decrease the percentage of false negatives up to $\approx 5.5\%$ compared to the maximin-based model. The reason is that our solution warns the cloud system about the ability of attackers to observe its detection strategy and allows it hence to benefit from this knowledge to adjust and optimize its detection load distribution strategy. This helps enhance the detection accuracy and minimize the possibility of overlooking actual attacks.

Fig. 1c presents the percentage of false positives or false alarms, which represents the percentage of non-malicious activities identified as attacks by the IDS. As explained in Eq. (11), the false positive rate $\gamma([t_1,t_2])$ during the time window $[t_1,t_2]$ is computed by subtracting the detection load probability distributions of the cloud system from the attack probability distributions of the attacker when the values of the former are larger.

$$\gamma([t_1,t_2]) = \sum_{x=t_1}^{t_2} \sum_{v \in V} \frac{(h_x(v) - a_x(v))}{t_2 - t_1} \text{ for each } h_x(v) > a_x(v), \quad (11)$$

We notice from Fig. 1c that our solution is able to decrease the percentage of false alarms up to $\approx 6\%$ compared to the maximin-based solution due to the same arguments explained earlier in the contexts of attack detection and false negative percentages.

In the second scenario, we vary the percentage of attacking VMs co-residing on a single cloud system from 10% up to 100% to explore the effects of this variation on the performance of the studied solutions. Fig. 2 reveals that the performance of the two solutions decreases with the increase in the percentage of attacking VMs. This unsurprising result is due to the fact that the bigger is the number of VMs attacking the system, the less is the ability of the cloud system to capture attacks under the limited budget of security resources. Fortunately, our solution shows a better scalability to an increased percentage of attacking VMs compared to the maximin-based model even in the worst case in which all the VMs are attacking the cloud system.

Moreover, we notice from Fig. 2 that our solution maximizes the attack detection (Fig. 2a) and minimizes the false negative (Fig. 2b) and false positive (Fig. 2c) percentages under the second scenario. Again, the reason is that our solution enables the cloud system to anticipate the best responses of the attackers to its adopted intrusion detection arrangements and use this information to adapt and optimize its detection strategies. This allows the cloud system to select the detection probability distributions that best randomize the detection process and make it unpredictable for attackers.

VI. Conclusion

This paper proposes an adaptive intrusion detection strategy that allows the cloud system to optimally distribute the detection load among its guest virtual machines, while taking into account that attackers have the ability to monitor the adopted strategies and adjust their own attack plans. This situation is modeled as a Stackelberg game in which the hypervisor plays the role of the game leader and attackers are the followers. The outcome of the game guides the cloud system on the optimal detection load distribution strategy over VMs that best maximizes the detection of distributed attacks in real-time. Experiments conducted using the CloudSim framework reveal that our solution maximizes the attacks detection up to 5% and minimizes false positives and negatives up to 6% compared to a recent maximin-based detection strategy. The results show as well that our solution is resilient to the increase in both the number of co-hosted VMs and percentage of co-resident attacking VMs.

Acknowledgment

This work has been supported by the Fonds de Recherche du Québec - Nature et Technologie (FRQNT), Natural Sciences and Engineering Research Council of Canada (NSERC), Khalifa University, Associated Research Unit of the National Council for Scientific Research (CNRS-Lebanon), and Lebanese American University.

References

[1] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "How to distribute the detection load among virtual machines to maximize the detection of distributed attacks in the cloud?" in *IEEE SCC*, 2016.

[2] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game," *IEEE Transactions on Services Computing*, 2016.

[3] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the hypervisor attack surface for a more secure cloud," in *ACM CCS*, 2011, pp. 401–412.

[4] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Optimal load distribution for the detection of VM-based DDoS attacks in the cloud," *IEEE Transactions on Services Computing*, 2017.

[5] H. von Stackelberg, *Market Structure and Equilibrium*, translation ed. Springer Berlin Heidelberg, 2011.

[6] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A stackelberg game for distributed formation of business-driven services communities," *Expert Systems with Applications*, vol. 45, pp. 359–372, 2016.

[7] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[8] B. Li, P. Liu, and L. Lin, "A cluster-based intrusion detection framework for monitoring the traffic of cloud environments," in *CSCloud 2016*, pp. 42–45.

[9] T. Alharkan and P. Martin, "IDSaaS: Intrusion detection system as a service in public clouds," in *CCGrid 2012*, pp. 686–687.

[10] C.-C. Lo, C.-C. Huang, and J. Ku, "A cooperative intrusion detection system framework for cloud computing networks," in *ICPPW*. IEEE, 2010, pp. 280–284.

[11] J. S. Ward and A. Barker, "Varanus: In situ monitoring for large scale cloud systems," in *CloudCom*, vol. 2. IEEE, 2013, pp. 341–344.

[12] A. M. Lonea, D. E. Popescu, and H. Tianfield, "Detecting DDoS attacks in cloud computing environment," *International Journal of Computers Communications & Control*, vol. 8, no. 1, pp. 70–78, 2013.

[13] P. Deshpande, S. Sharma, S. Peddoju, and S. Junaid, "HIDS: A host based intrusion detection system for cloud computing environment," *International Journal of System Assurance Engineering and Management*, pp. 1–10, 2014.

[14] M. R. Watson, A. K. Marnerides, A. Mauthe, D. Hutchison *et al.*, "Malware detection in cloud computing infrastructures," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 192–205, 2016.

[15] S. Bharadwaja, W. Sun, M. Niamat, and F. Shen, "Collabra: a Xen hypervisor based collaborative intrusion detection system," in *ITNG 2011*, pp. 695–700.

[16] F. Lombardi and R. Di Pietro, "Secure virtualization for cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1113–1122, 2011.

[17] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A survey on trust and reputation models for Web services: Single, composite, and communities," *Decision Support Systems*, vol. 74, pp. 121–134, 2015.

[18] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus, "Playing games for security: an efficient exact algorithm for solving bayesian stackelberg games," in *AAMAS 2008*, pp. 895–902.

[19] T. S. Ferguson, "Game theory," *Mathematics Department, UCLA*, 2008.