Online analysis of security risks in elastic cloud applications using probabilistic model checking

Athanasios Naskos*, Anastasios Gounaris*, Haralambos Mouratidis† Panagiotis Katsaros*
*Aristotle University of Thessaloniki, Greece, Email: {anaskos,gounaria,katsaros}@csd.auth.gr
†University of Brighton, UK, Email: H.Mouratidis@brighton.ac.uk

***Abstract*—Security-related concerns in elastic cloud applications call for a risk-based approach due to the inherent trade-offs between security and other non-functional requirements, such as performance. To this end, we advocate a solution that can be efficiently realized through modeling the application behavior as a Markov Decision Process, on top of which probabilistic model checking is applied. We explain the main steps and we illustrate how we can perform online analysis and decision making regarding elasticity decisions. Our runtime analysis is capable of providing evidence for key security-related aspects of the running applications, like the probability of data leakage in the next hour.**

Index terms: risk-based cloud elasticity, probabilistic model checking, horizontal scaling

I. Introduction

Cloud computing is established as the main option for the deployment of web applications. Such applications come from a wide range of stakeholders including big companies, SMEs, organizations and research institutes. Cloud computing has become attractive for two reasons: i) Computational resources are released on demand and application providers have the options to be charged in a pay-as-you-go manner. This means that up-front investments in equipment and human resources are minimized, and can be fully allocated to application deployment and management; (ii) resources are released in response to workload changes. This property, known as autoscaling, has an impact on the monetary cost of running a cloud application, since resources are used only when required.

Elastic applications are those that can benefit from the aforementioned inherent property of cloud infrastructures to provide computational resources on the fly according to their current needs. Computation resources are typically provided in the form of Virtual Machines (VMs). Elasticity is manifested in three main forms: (a) Horizontal scaling, where either new VMs are added or existing ones are removed. This type of elasticity provides the biggest potential for scalability and performance improvements, due to the perceived unlimited number of VMs that can be provided. (b) Vertical scaling, where certain properties (e.g., the number of cores) of the existing VMs are modified. (c) Live migration, where a VM is moved to a different physical host, while staying operational.

In the following sections, we narrow our focus on horizontal scaling and we argue that the goals of meeting performance requirements with the help of horizontal scaling contradict to those of security, thus calling for a risk-based security solution.

A. Security concerns and horizontal scaling in public clouds.

Public and hybrid clouds, as opposed to private ones, offer resources to arbitrary customers, who are also called tenants. Tenants do not have control on either the security policy for running the underlying infrastructure or the type of other tenants, whose VMs are collocated on the same physical machines. While this does not necessarily imply that public clouds are insecure, it has been repeatedly reported to prohibit migration of applications to the cloud [1].

The Cloud Security Alliance published a report in 2013, which identified the main cloud-related security threats [2]. Data breaches due to malicious co-tenants was on the top. This can lead to both data leakage and data loss. Simply speaking, data leakage is the unauthorized disclosure of data from one user to another, whereas data loss refers to a condition where data is destroyed and becomes unavailable. In addition, in a multi-tenant environment, issues such as lack of authorization mechanisms for sharing physical resources increase the risk of threats, such as service traffic hijacking, which occurs when attackers hijack cloud accounts by stealing security credentials and eavesdropping on activities and transactions; and side-channel attacks, which are based on information obtained from bandwidth-monitoring or other similar techniques. Moreover, when multiple tenants share an underlying infrastructure the risk of threats related to misconfiguration and uncoordinated change control is increased allowing a malicious tenant to gain access to another tenant's resources.

Keeping the number of VMs as low as possible could be regarded as an indirect way to mitigate security concerns related to data leakage and loss. However, this may entail an unacceptable compromise on performance. Performance is in the top three most studied Service Level Agreements (SLA) parameters, due to the fact that critical applications require responses in a fixed short period of time [3]. So, addressing security concerns and honoring Service Level Agreements (SLAs) should be examined in a combined manner to render a cloud application reliable.

An example is shown in Figure 1, which refers to a setting where an elastic NoSQL database serves end-user requests according to the The Yahoo! Cloud Serving Benchmark (YCSB). The exact deployment details are in [4]. The figure refers to a fixed rate of user requests and shows how the average and standard deviation values of the latency of responses vary with the number of VMs used. The observation is that a strict threshold on latency would force the system to acquire

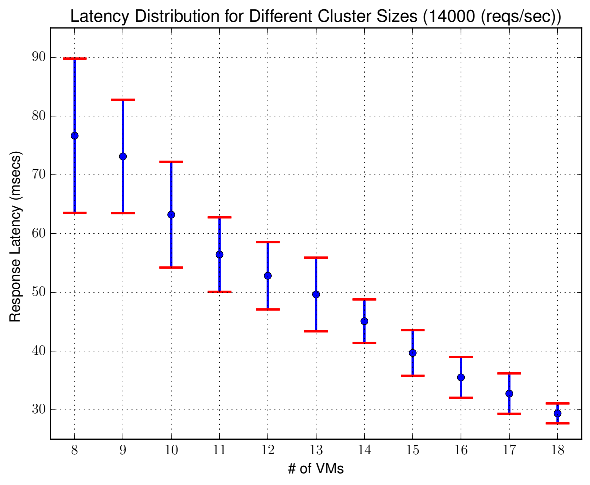

Fig. 1: Execution plans based on different cost metrics.

additional VMs, the exact quantity of which needs to be computed at runtime according to the current workload and the fact that the behavior of the system is volatile. However, increasing the number of VMs, and assuming that each VM runs on a different physical machine in the generic case, also increases the probability of a malicious tenant to be collocated. This probability may vary according to the type of the cloud provider [5], but the important issue is that it is non-negligible. Choosing a single trustworthy provider is not sufficient either, given that providers may offer VMs from other providers as well in periods of very high demands [6]. Overall, as explained above, the addition of new VMs poses a threat of data leakage and data loss. The magnitude of this threat is application-dependent. In the example, the threat of data loss is lower than in normal applications, because NoSQL databases are replicated at least two or three times, which makes it harder for a malicious user to destroy all copies.

Due to the inherent trade-offs between security and performance requirements, any solution to analyze and enforce security-aware horizontal scaling for cloud applications needs to be risk-based. It should account for the following aspects: (a) probability of data leakage and data loss; (b) dynamic evolution of the external environment and volatility of the system behavior; and (c) potential heterogeneity of the cloud infrastructure. The former essentially expresses a security risk, which stems from the inherent vulnerability due to co-tenancy and needs to be balanced with performance goals.

B. Our solution for risk-based security in a nutshell.

We advocate the usage of a formal verification approach, as a means to apply mathematical reasoning for providing security-oriented probabilistic guarantees for elastic cloud applications. The exact technique we employ is probabilistic model checking [7] on top of system models in the form of Markov Decision Processes (MDPs), which are instantiated on the fly. Our technique is capable of analyzing and providing evidence for key security-related aspects of the running applications, e.g., to answer questions like *"What is the probability that there will be a data leakage in the next hour?"*. Moreover, it is capable of driving elasticity decisions taking into account security constraints, e.g., given the current query load and the prediction for this load in the next half an hour, to decide on *how many VMs to add or remove in order to maximize a bi-objective utility functions that takes into account both performance and security objectives*.

Using probabilistic model checking for analyzing and driving elasticity decisions is a novel technique with particularly promising initial results, as reported in [4], whereas in [8], its potential in considering security requirements has been demonstrated. Most of the work in cloud security focuses on the identification of risks and vulnerabilities, security mechanisms, such as encryption, digital signatures and access control, and manners to attain security assurance, such as monitoring, certificates, auditability and so on [9]. However, the detailed investigation of security assurance during horizontal scaling, and even more, the security-aware elasticity decision making that we hereby enable is something that is novel in both the field of cloud security [9], [10] and the field of dynamic resource allocation in clouds [11], [12].

II. Modeling elastic applications

We advocate a model-based approach. Figure 2 illustrates a conceptual model of an elastic application that considers both security and performance SLA requirements and which is deployed on a public cloud. Each curved rectangle represents a conceptual state at a specific time instant. The evolution of the elastic application is modeled as a transition to a state at a future time point $t+\Delta t$ through elastic actions, such as adding or removing VMs.

For each state, we capture the features of interest for the analysis and decision making: (i) The mixture of VM types employed. In the figure, we assume that VMs are of two different types and/or providers, v and w. (ii) The total deployment cost, labeled as m. (iii) The probability of data leakage labelled as x. (iv) The probability of data loss labeled as y. (v) The probability of performance-related SLA condition violations labeled as z. Finally, (vi) the probability of no security threats or performance violations labeled as k. A reasonable assumption is to consider all these probabilities statistically independent. In that case,

$$k = (1-x)(1-y)(1-z)$$

Further, the probability of data leakage on a single machine of type A, dl_A, can be safely regarded as independent of the number of VMs of type A employed. So,

$$x = 1-(1-dl_v)^v(1-dl_w)^w$$

Similarly, y can be defined as a function of the number and type of VMs employed.

The evolution of the system due to elasticity actions refers to discrete time intervals of period Δt. We consider three actions: *add*, *remove* or *no_change*. Note that in the generic case, the effects of actions at time t may be delayed and not manifested at the next period but after multiple time points. For example, adding a new VM to serve a NoSQL database implies that a

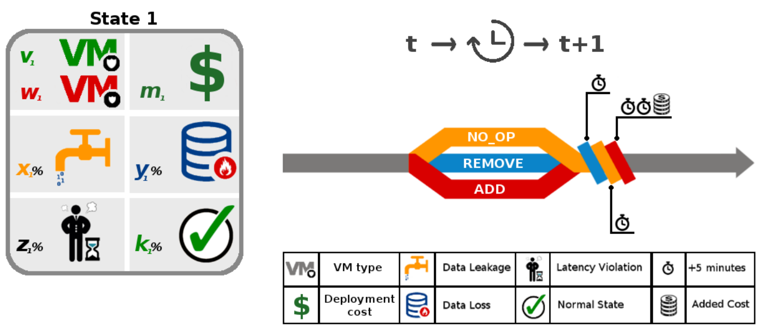

Fig. 2: A conceptual model of an elastic application that considers both security and performance SLA requirements.

new VM needs to be created, booted, configured and receive data, which might take more than Δt time. During this process, the system is supposed to be in a transient state.

A. MDP implementation of the conceptual model.

We implement our technique using the Markov Decision Processes (MDP) modeling approach. The choice of MDP is made because a) MDP enables both analysis and decision making, and b) it can capture both non-determinism and uncertainty in a given system [13]. Both properties are essential in an elastic cloud application. Because of horizontal scaling, at each time point, the number of VMs can increase, remain the same or decrease. This gives rise to non-determinism. Also, at any given point, there might or might not be a performance or security-related requirement violation. This necessitates the modeling of uncertainty.

MDPs are specified using: a) states, b) actions, c) probabilities, and d) rewards. The states represent system snapshots at specific time points, which are characterized by a set of system properties. The actions are transitions between the states, which express some change to the state properties. The probabilities refer to each triple *(state s)-(action a)-(state s')* and represent the probabilities of transition from one state to another due to a specific action thus quantifying uncertainty. Finally, the rewards are used to perform quantitative analysis (or solution) of MDP models.

The conceptual model needs to be implemented as a MDP according to the analysis requirements. Since, during analysis and elasticity decision making, we need to explicitly consider different types of the application behavior, we map each conceptual system state to multiple MDP model states, each for one behavior type. The behavior type is defined according to the application non-functional requirements. Let an application set three requirements: to avoid data leakage, data loss and latency above a user-specified threshold. Then, each combination of a binary variable that indicates the satisfaction of each requirement defines a behavior type (see Figure 3). Furthermore, each MDP state is annotated as to whether it refers to a transient state or not (not shown in the figure).

The actions are the same as in the conceptual model. The only difference is that, if the MDP state is a transient one, only *no_change* is allowed because taking further resizing decisions during instable periods is very prone to suboptimal decision making. The next step is to define the transition probabilities. In Figure 4, we provide a complementary view of the previous figure, where each path from s to s' corresponds to a MDP model state. The probabilities p_0 to p_7 in Figure 3 are the product of the probabilities in Figure 4 along the corresponding path. For example, $p_7 = (1-x)(1-y)(1-z)$. In general, i) for a given initial state s and action a, $\sum_s' p(s,a,s') = 1$; and ii) multiple actions can be plausible for each state.

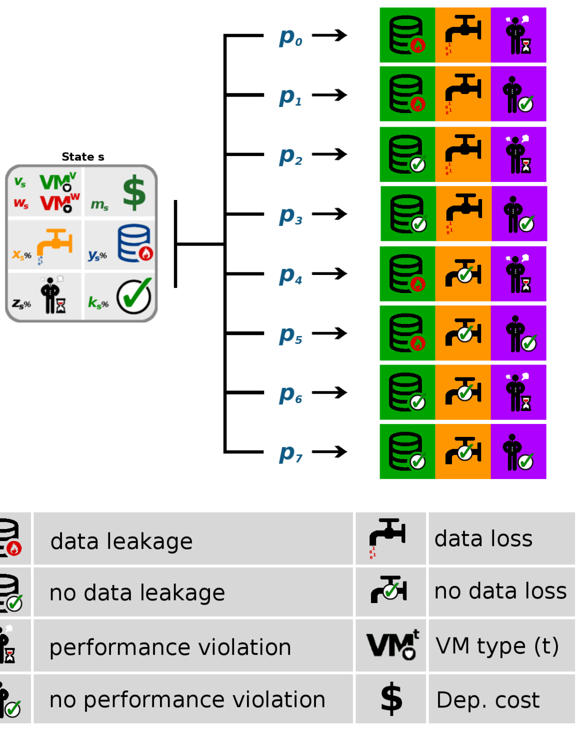

Fig. 3: Mapping of a conceptual state to a set of MDP states.

B. MDP instantiation.

In order to serve online analysis (discussed later in detail), the model is instantiated on the fly. To this end, the *decision depth* and the probabilities need to take actual values. The decision depth refers to how many periods the model can account for. If the depth is too small, then the system becomes too short-sighted; if it is too big, the prediction uncertainty

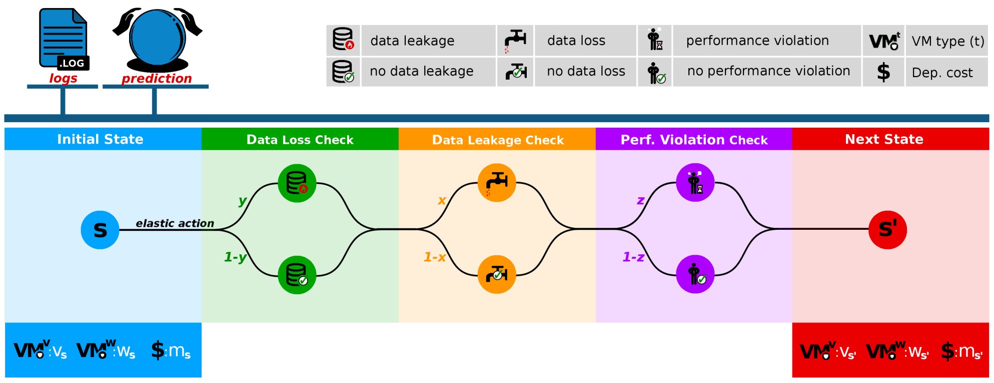

Fig. 4: Setting the MDP model probabilities.

is increased. Both situations lead to suboptimal decisions. Clearly, the number of MDP states grows exponentially in the number of Δt periods. If a period represents 5 minutes of real time, a model with depth set to 4 refers to a scenario where elasticity is re-assessed every 5 minutes, and the model looks ahead for 20 minutes. If the system evolves less rapidly, the application manager could map each period to a longer time. In general, setting Δt appropriately heavily relies on the volatility of the application environment, e.g., if during night hours the workload remains roughly stable, then Δt can be increased.

In our approach, the probabilities x, y and z are derived through logs. Past log entries referring to the same mixture of VM types as the state of interest are analyzed to estimate the probabilities. For performance-related metrics, such as the latency, not only the number of VMs needs to be considered, but also the external load of incoming requests. This implies the need to add a load prediction component, as shown in Figure 4. In general, for our approach to be applicable, we assume that a security analysis and profiling mechanism is in place, which is capable of deriving attack probabilities as a function of the cluster configuration. Such a mechanism is orthogonal to our approach and can be even more sophisticated. For example, upon instantiation, it may take into account whether any VM additions would involve usage of a new physical machine rather than deploying VMs on an already used one. Finally, since the models are instantiated on the fly, the attack probabilities can be dynamically refined.

III. Online Analysis and Decision Making

The analysis is based on verification of models instantiated on demand. To this end, we couple the probabilistic model with a probabilistic property specification language, namely Probabilistic Computation Tree Logic (PCTL), which is fed to the PRISM model checker [14]. We also show, how the analysis can directly support decision making with regards to elasticity.

PRISM can efficiently analyze complex models. We have been able to solve MDP models with 9958 states corresponding to 4 periods in 0.073 seconds using a machine with a quad core CPU and 8GB of RAM, while the program is reported to have processed models up to 10^{11} states on a single machine [14].

A. *Examples of Verified Analysis using PRISM*

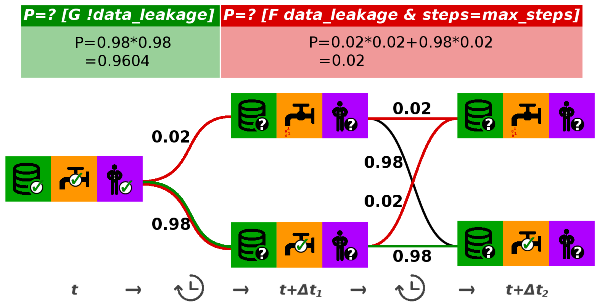

Fig. 5: An example of probabilistic model checking.

Figure 5 presents a concise view of the analysis of two PCTL properties. For simplicity, we have grouped the eight states of Figure 3 in two groups according to the data leakage property. Further, we assume that the decision depth is set to 2 and the probability of data leakage in a time interval x is 0.02. Also, in this toy example, only one action is allowed, e.g., *no_change*, i.e., there is no non-determinism. Obviously, in a more complete example with non-determinism including all elasticity actions, multiple such paths are eligible.

The first PCTL property (in green) answers the question *“What is the probability of having no data leakage incident?”*. The corresponding PCTL is expressed as $P = ?\ [G\ !data\_leakage]$, and is satisfied by the green path in the model, while the returned probability is 0.9604.

The second PCTL (in red) answers the question *“What is the probability of eventually (i.e., in the final state in the verification) having data leakage?”*. It is expressed as $P = ?\ [F\ data\_leakage\ \&\ steps = max\_steps]$ and returns the cumulative probability of the red transitions, which is 0.02.

In Table I, we present more complex examples of risk-based security analysis, where any of the three actions is allowed. Thus, during analysis, different sequences of actions are investigated. Formally, these sequences are called *adversaries*, or *policies*, or *strategies*. The results of each analysis may include multiple adversaries. The PCTL statements are mostly self-explanatory. The symbol U defines which state needs to follow the state on the left, while F stands for eventually reaching a state, G stands for a condition that needs to hold throughout the policy and X for the next state. The decision depth is termed as *max_steps*.

The examples in Table I fall into three categories. The first two rows refer to analyses, where the outcome is a probability of reaching certain model states. The next two examples return a boolean value indicating whether a specific property holds. The last two PCTL properties are *numerical* multi-objective ones, as they ask for the maximum possible probability of reaching a state under the condition that the probability of reaching another state is bounded by a given threshold. The objectives may either refer to a single property, like data loss in the fifth example, or multiple ones, like data leakage, data loss and monetary cost in the last example.

B. Decision Making

MDPs are inherently suited for decision making as well. To this end, each model state needs to be associated with a reward value. State rewards are computed using functions that quantify various aspects of the system like *performance* and *security* concerns or more concrete assets like the *number of active VMs* or the actual *deployment cost*. As an example, consider the following utility function that uses weights to balance three aspects: the normalized probability of *data leakage* ($\tilde{p}_{dleak}$), *data loss* ($\tilde{p}_{dloss}$) and the latency exceeding a threshold ($\tilde{p}_{perf}$):

$$u(vms) = a \cdot \tilde{p}_{dleak} + b \cdot \tilde{p}_{dloss} + c \cdot \tilde{p}_{perf}, \; a+b+c = 1 \quad (1)$$

Note that, in general, threats and objectives can be prioritized. This is reflected on the utility function by assigning different values to the weights. Also, our approach is orthogonal to any user-defined utility function.

Our decision making proposal is based on the computation of the cumulative reward of every adversary. The model solver examines the possible alternatives, i.e., all combinations of state transitions, and computes the optimal cumulative state reward along with the corresponding sequence of actions. For example, using the utility function above, the optimal reward is the minimum one. In PRISM, this can be done with the help of a different type of PCTL specification, which asks for reward minimization rather than probabilities: $R\{\text{“}cumulative\_reward\text{”}\}min =? \, [F \; steps = max\_steps]$.

Interestingly, several decision policies can be built on top of the aforementioned verification. For example, if multiple adversaries are returned with equal reward, then a second PCTL on other aspects, such as the probability of security violation as presented in Table I can be used to choose the final strategy.

Moreover, it is not necessary to perform all actions in that strategy. In [4], an elasticity decision making technique is presented tailored to NoSQL databases. That technique follows the steps mentioned earlier. After deciding on the adversary at each time point the decision mechanism is activated, only the first elasticity action is enacted, and then the whole adversary is re-evaluated from scratch. Such an approach is in line with a wide range of adaptive solutions, such as Model Predictive Control (MPC) [15], where a sequence of adaptations is computed, of which only the first step actually is applied. According to the evaluation results in [4], the quality of elasticity decisions outperform other proposals for scaling NoSQL databases in avoiding both violations of latency thresholds and over-provisioning of VMs.

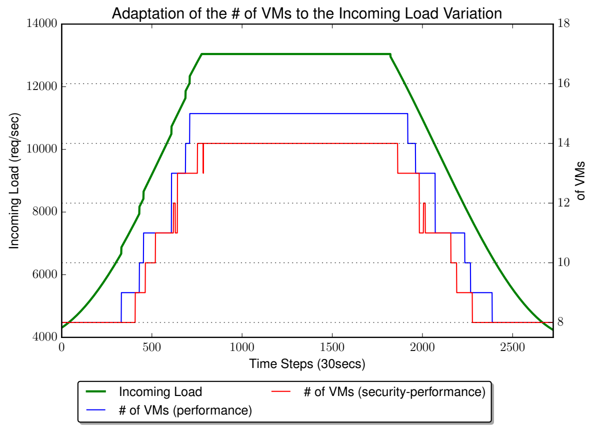

Fig. 6: Example of security-aware elasticity decisions.

In Figure 6, we show how a security-aware elasticity decision maker behaves in a setting similar to the one in [4]. While the load varies (green plot), the number of VMs is constantly re-evaluated. The blue plot shows that number when only performance requirements are considered, whereas the red plot shows the behavior when the state rewards are computed based on Eq. (1). In the latter case, the usage of VMs is more limited in order to mitigate the threat of data leakage and loss.

IV. Final Remarks

Our approach is of interest to both owners of elastic applications and cloud service providers. The outcomes of our proposal can be used either to analyse the (elastic) behaviour or to take elasticity decisions. Additionally, the analysis results can be used to fine-tune the utility function, acting as a feedback mechanism, so that the decisions are good in practice.

Analysis and decision making in elastic applications is, by its nature, an instance of autonomic computing problems. A key point to autonomic solutions is to render them dependable and endow them with a solid formal basis. To this end, employing probabilistic model checking is a promising direction, as explained in this article. Not only it allows for the continuous verification of system properties, but it is an effective tool for meeting both security- and performance oriented goals. The work in [4], [8] is a good starting point for exploring this direction further and deeper.

In the text above, we focused on horizontal scaling as the most drastic and less intrusive elasticity action. Regarding the security threats, data leakage and loss were selected as the

	Analysis Goal	PCTL property
1	*What is the maximum probability (among all possible adversaries) of moving from a state with data leakage to a state with no data leakage?*	$Pmax =? \ [data\_leakage \ U \ !data\_leakage]$
2	*What is the maximum probability (among all possible adversaries) of experiencing a data loss incident until eventually move to a state with no data loss?*	$Pmax =? \ [data\_loss \ U \ F \ !data\_loss \ \& \ steps = max\_steps]$
3	*Starting from any reachable state, is it always possible (i.e., is there at least one adversary) to eventually reach a state with no data leakage?*	$filter(exists, \ P >= 1 \ [F \ !data\_leakage \ \& \ steps = max\_steps])$
4	*Starting from a state with no data loss, do all adversaries eventually reach a state with no data loss?*	$filter(forall, \ P >= 1 \ [F \ data\_loss \ \& \ steps = max\_steps], \ !data\_loss)$
5	*What is the maximum probability of experiencing data loss in a state that immediately follows the initial state, while the probability to end up at a state with no data loss is greater than or equal to 0.9?*	$multi(Pmax =? \ [X \ data\_loss], \ P >= 0.9 \ [F \ !data\_loss \ \& \ steps = max\_steps])$
6	*What is the maximum probability of having total cost of deployment less than a specified budget, while the probability of experiencing any security incident does not exceed 0.05?*	$multi(Pmax =? \ [F \ total\_cost <= Budget \ \& \ steps = max\_steps], \ P <= 0.05 \ [G \ data\_leakage \ \& \ data\_loss])$

TABLE I: Additional examples of analyses enabled.

most prominent ones that depend on the number of occupied VMs, i.e., their severeness can be handled through horizontal elasticity. Our approach can cover additional security threats that are affected by scaling. Also, it is applicable to the two complementary types of elasticity, namely vertical elasticity and live migration. To this end, the envisaged models need to be more fine-grained, considering VM configuration types and physical machines rather than only the number of VMs. The analysis of all elasticity types in combination with additional security threats is a challenging avenue for further research.

References

[1] C. Kalloniatis, V. Manousakis, H. Mouratidis, and S. Gritzalis, "Migrating into the cloud: Identifying the major security and privacy concerns," in *Collaborative, Trusted and Privacy-Aware e/m-Services - 12th IFIP WG 6.11 Conference on e-Business, e-Services, and e-Society, I3E 2013, Athens, Greece, April 25-26, 2013. Proceedings*, 2013, pp. 73–87.

[2] Cloud Security Alliance, "The notorious nine - cloud computing top threats in 2013." [Online]. Available: https://cloudsecurityalliance.org/download/the-notorious-nine-cloud-computing-top-threats-in-2013

[3] F. Faniyi and R. Bahsoon, "A systematic review of service level management in the cloud," *ACM Comput. Surv.*, vol. 48, no. 3, 2015.

[4] A. Naskos, E. Stachtiari, A. Gounaris, P. Katsaros, D. Tsoumakos, I. Konstantinou, and S. Sioutas, "Dependable horizontal scaling based on probabilistic model checking," in *CCGrid*. IEEE, 2015.

[5] S. Ramgovind, M. M. Eloff, and E. Smith, "The management of security in cloud computing," in *Information Security for South Africa (ISSA), 2010*. IEEE, 2010, pp. 1–7.

[6] E. Casalicchio, D. A. Menascé, and A. Aldhalaan, "Autonomic resource provisioning in cloud systems with availability goals," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, 2013, pp. 1:1–1:10.

[7] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker, "Automated verification techniques for probabilistic systems," in *Formal Methods for Eternal Networked Software Systems (SFM'11)*, 2011, pp. 53–113.

[8] A. Naskos, A. Gounaris, H. Mouratidis, and P. Katsaros, "Security-aware elasticity for nosql databases," in *Model and Data Engineering - 5th International Conference, MEDI 2015, Rhodes, Greece, September 26-28, 2015, Proceedings*, 2015, pp. 181–197.

[9] C. A. Ardagna, R. Asal, E. Damiani, and Q. H. Vu, "From security to assurance in the cloud: A survey," *ACM Comput. Surv.*, vol. 48, no. 1, 2015.

[10] H. Mouratidis, S. Islam, C. Kalloniatis, and S. Gritzalis, "A framework to support selection of cloud providers based on security and privacy requirements," *Journal of Systems and Software*, vol. 86, no. 9, pp. 2276–2293, 2013.

[11] Z. A. Mann, "Allocation of virtual machines in cloud data centers&mdash;a survey of problem models and optimization algorithms," *ACM Comput. Surv.*, vol. 48, no. 1, 2015.

[12] S. Singh and I. Chana, "Qos-aware autonomic resource management in cloud computing: A systematic review," *ACM Comput. Surv.*, vol. 48, no. 3, 2015.

[13] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[14] M. Kwiatkowska, G. Norman, and D. Parker, "Prism: probabilistic model checking for performance and reliability analysis," *SIGMETRICS*, vol. 36, no. 4, pp. 40–45, 2009.

[15] J. Maciejowski, *Predictive control with constraints*. Prentice Hall, 2001.