A Moving-target Defense Strategy for Cloud-based Services with Heterogeneous and Dynamic Attack Surfaces

Wei Peng*, Feng Li†, Chin-Tser Huang‡, and Xukai Zou*
*Department of Computer and Information Science
†Department of Computer Information and Graphics Technology
Indiana University-Purdue University Indianapolis, Indianapolis, IN, U.S.A.
‡Department of Computer Science and Engineering
University of South Carolina, Columbia, SC, U.S.A.

***Abstract*—Due to deep automation, the configuration of many Cloud infrastructures is static and homogeneous, which, while easing administration, significantly decreases a potential attacker's uncertainty on a deployed Cloud-based service and hence increases the chance of the service being compromised. Moving-target defense (MTD) is a promising solution to the configuration staticity and homogeneity problem. This paper presents our findings on *whether* and *to what extent* MTD is effective in protecting a Cloud-based service with *heterogeneous* and *dynamic* attack surfaces—these attributes, which match the reality of current Cloud infrastructures, have not been investigated together in previous works on MTD in general network settings. We 1) formulate a Cloud-based service security model that incorporates Cloud-specific features such as VM migration/snapshotting and the diversity/compatibility of migration, 2) consider the accumulative effect of the attacker's intelligence on the target service's attack surface, 3) model the heterogeneity and dynamics of the service's attack surfaces, as defined by the (dynamic) probability of the service being compromised, as an S-shaped generalized logistic function, and 4) propose a probabilistic MTD service deployment strategy that exploits the dynamics and heterogeneity of attack surfaces for protecting the service against attackers. Through simulation, we identify the conditions and extent of the proposed MTD strategy's effectiveness in protecting Cloud-based services. Namely, 1) MTD is more effective when the service deployment is dense in the replacement pool and/or when the attack is strong, and 2) attack-surface heterogeneity-and-dynamics awareness helps in improving MTD's effectiveness.**

***Index terms*—moving-target defense, risk modeling, probabilistic algorithm, simulation**

I. Introduction

This paper presents our findings on *whether* and *to what extent moving-target defense (MTD)* is effective in protecting a *Cloud-based service* with *heterogeneous* and *dynamic* attack surfaces—the security implications of these two attributes will be explained shortly and be more fully evaluated by the end of this paper. The motivations behind our work are:

The proliferation of Cloud-based services. Cloud computing has been applied in recent years to various information services such web hosting (e.g., Amazon Web Services), storage (e.g., Dropbox and Copy.com), application delivery (e.g., Google App Engine and Microsoft Office 365), and scientific computing (e.g., FutureGrid). Cloud computing provides elasticity to clients through on-demand allocation and cost-effectiveness to service provider through efficient resource allocation. Therefore, we expect that the migration to the Cloud-based service model will sustain and expand in foreseeable future. The enabling technology of Cloud computing is the virtualization of commodity hardware/software.

The emergence of MTD as a promising defensive technique in the cyber-security research community. The motivating observation behind MTD is the asymmetry in the traditional attacker-defender relationship: Defenses are usually *reactive* to a preceding *proactive* attempt—the attacker always has an upper hand. MTD attempts to reverse this situation through *proactive defense* by moving the target in *anticipation*, rather than in response, to attacks. The purpose of moving the target is to decrease the utility (for attacking) of attackers' existing intelligence on the old target and increase attackers' uncertainty on the new target.

MTD is particularly applicable to Clouds for the following reason. The deep automation in Cloud management, although eliminates inconsistencies and mistakes that plague manual administration, creates Cloud infrastructures that are largely *static* and *homogeneous*. For example, a major Infrastructure-as-a-Service (IaaS) service provider, Amazon Elastic Compute Cloud (EC2), provides a relatively small number of options (i.e., instance type and operating system combinations) for users to bootstrap a new virtual machine (VM). Although this practice eases platform administration and support, it effectively makes the new VM a *static* target. Besides, the relative scarcity of options also make multiple VM instances to be largely *homegeneous*. Staticity and homogeneity in infrastructure give advantages to attackers by decreasing the attackers' uncertainty on potential targets. Therefore, MTD, the goal of which is to increase attackers' uncertainty, is quite applicable to Clouds.

Although previous works have examined MTD in general network settings (a few of which are surveyed in Section V), our work contributes new understandings on the effectiveness of MTD in the following aspects.

- We formulate a model Cloud-based service security model that incorporates features of modern Cloud infrastructures such as VM migration/snapshotting and the diversity/compatibility of migration (Section II-A).
- We consider the accumulative effect of the attacker's intelligence on the target service's attack surface (Section II-B) and model the heterogeneity and dynamics of the service's attack surfaces, as defined by the (dynamic) probability of the service being compromised, as an

978-1-4799-2003-7/14/$31.00 ©2014 IEEE

S-shaped generalized logistic function (Equation (1) in Section II-C).

- We propose a probabilistic MTD service deployment strategy that exploits the dynamics and heterogeneity of attack surfaces for protecting the service against attackers (Section III).
- Through simulation, we identify the conditions and the extension of the proposed MTD strategy's effectiveness for protecting Cloud-based services (Section IV-B). Namely, 1) MTD is more effective when the service deployment is dense in the replacement pool and/or when the attack is strong, and 2) attack-surface heterogeneity-and-dynamics awareness helps in improving MTD's effectiveness.

II. Model

We consider a model consisting a pair of rivalries: a Cloud-based *service* and an *attacker*.

- The service is deployed on one or multiple VM instances among a pool of such instances in the Cloud.
- The attacker, who targets at the service but does not know which VM instances the service is currently deployed on, repeatedly probes and attacks the pool of VM instances in order to disrupt the service by capturing the VM instances on which the service is deployed.

The goal of our design (Section III) is to devise a service deployment strategy to hold out the attack as long as possible. In our model, both the service and the attack are subject to the following conditions that are derived from realistic constraints in modern Cloud infrastructures.

A. Service model

The VM instances on which the service is currently deployed are *active*, whereas the rest instances in the pool are *inactive*. In our model (Section II-B), the attacker cannot differentiate between active and inactive VM instances in selecting and attacking targets. Therefore, the set of inactive VM instances protects the active ones from direct exposure to the attacker by increasing the attacker's uncertainty.

The service can be deployed on different sets of VM instances: an active VM instance can be *replaced* by another VM instance. This models the capability of modern VMMs in which VMs are allowed to migrate across physical machine boundaries. However, to ensure MTD effectiveness, migration should introduce enough diversity in configuration to thwart attacks, while, at the same time, migration is also subject to some technical constraints, e.g., current virtualization technology does not support migration between, say, Linux and Windows operating systems. In other words, the set of VM instances that serve as replacements for an active instance is the subset of the whole pool that are both diverse (in configuration) and similar (within technical feasibility) at the same time. For a VM instance j, let the set of such replacements of j be $R(j)$.

We consider a *snapshot-and-restore* service migration model instead of a *refreshing* model, in which the migration destination is always refreshed to a known good state (such as a freshly installed operating system). In the snapshot-and-restore model, when the service migrates away from an active VM instance, a snapshot is taken; if the service migrates back to this instance later, the snapshot is restored and service is resumed from there. This choice of service migration model is based on the following reasons.

- A service may require an execution context that gets destroyed under the refreshing model, but is preserved under the snapshot-and-restore model.
- While the refreshing model *fully* negates the attacker's accumulated advantage upon migration, the snapshot-and-restore model only *partially* negates the attacker's accumulated advantage—the snapshot preserves any backdoor left by the attacker along with the rest of the state of the VM instance prior to migration. We consider the more challenging (for the defender) snapshot-and-restore service migration model in order to isolate the effect of *moving*, rather than *refreshing*, in studying the effectiveness of MTD.

B. Attacker model

The attacker can probe any VM instance in the pool, but cannot differentiate between active and inactive VM instances (the justification is discussed in Section IV-A in evaluating the effectiveness of MTD). This could be realized by the defender through, for example, deploying fake services or honeypots on the inactive VM instances to confuse the attacker.

The attacker is constrained by an *attack budget*: There is an upper limit on the number of VM instances that the attacker can probe and attack within a given time constraint. In a discrete-event model in which the attacker a's attack budget within a unit of time is k, a can probe and attack at most k VM instances within the unit time.

The attacker's intelligence on the target is *accumulative*. When the attacker invests its budget on an *active* VM instance (i.e., a *hit*), the probability of the attacker successfully capturing the active VM instance *increases* (Section II-C). Conversely, if the attacker invests its budget on an *inactive* VM instance (i.e., a *miss*), although this inactive VM instance may eventually be compromised, the service is not affected, and the attacker's efforts will be negated for this VM instance by the snapshot restoration when the service migrates to it. Nevertheless, as discussed in Section II-A, we allow attacker's efforts to be preserved for active VM instances under the snapshot-and-restoration service migration model: The probability that an active VM instance will be captured by the attacker is a non-decreasing function against the number of hits.

C. Attack surface model

As hinted in Sections II-A and II-B, we consider a *heterogeneous* and *dynamic* attack surface model. We model the attack surface of an active VM instance as the probability of it being compromised by the attacker over the course of time that it is actively deployed.

Intuitively, attack surfaces can be modeled with an S-shaped function, characterized by a first phase of increasing growth rate starting from some value above 0, followed by a second phase of decreasing growth rate ending at some value below 1. The first phase corresponds to the reconnaissance stage that precedes almost all real attacks—the attacker has a relatively low success probability at this stage, but the attacker's intelligence on the targets grows fast (any new intelligence is good due to lack of intelligence at this stage). The latter phase

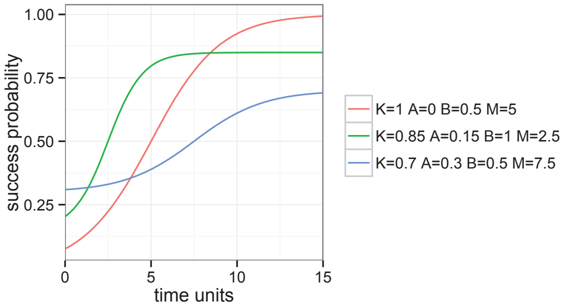

Fig. 1: We model the heterogeneity and dynamics of the service's attack surfaces by a non-decreasing probability density function over the period of time that the service is actively deployed, as shown here for the generalized logistic model defined by Equation (1), which is determined by the 4 parameters A_j (low asymptote), K_j (upper asymptote), B_j (growth rate), and M_j (time of maximal growth). An evaluation of the proposed design (Section III) against the examples shown here is presented in Section IV.

corresponds to a stage in which the attacker's intelligence on the target is saturated, but has a relatively large probability of success due to the very abundance of intelligence on the target. We model this effect with a generalized logistic function:

$$P_{a,j}(t) = A_j + \frac{K_j - A_j}{1 + e^{-B_j(t-M_j)}}. \tag{1}$$

$P_{a,j}(t)$ is the probability of, after t units of time, the attacker a successfully compromising the active VM instance j the next time a probes and attacks j. The parameters in Equation (1) have the following significance.

- A_j is the lower asymptote, which is the tight lower bound of $P_{a,j}(t)$. Hence, $A_j \geq 0$. a's first hit has a success probability of (very close to) A_j.
- K_j is the upper asymptote, which is the tight upper bound of $P_{a,j}(t)$. Hence, $A_j < K_j \leq 1$. a's success probability is never over K_j.
- B_j is the growth rate. B_j determines the growth in the attacker's success probability between subsequent hits.
- M_j is the time of maximal growth. The period before M_j has an increasing growth rate, whereas the period after M_j has a decreasing growth rate.

The effects of the parameters on the shape of the attacker's advantage model $P_{a,j}(t)$ (Equation (1)) are illustrated in Figure 1. The parameters in Equation (1) are determined by the following distinct but related aspects.

- The attacker's capability. In the real world, where the attacker's capability is unknown to the service defender, the parameters in Equation (1) quantify the strength of attacks that the service is designed to withstand. For example, a model with high B_j and low M_j quantifies a strong and determined attacker who has the capability of compromising a target fast.
- An active VM instance's attack surface, as defined by the totality of its externally accessible resources. Existing works (e.g., Manadhata and Wing [14]) that quantify attack surfaces can be adapted to determine the parameters in Equation (1).

III. Design

Based on the model described in Section II, especially the model on attacker's probability of successfully compromising an active VM instance (Equation (1)), we propose the following MTD service deployment strategy for securing the service against attacks.

During each unit of the discrete time, an active VM instance j makes a decision of whether to migrate to one of its replacements $R(j)$. Because it does not know the (faceless) attacker a's activities, j wishes for the best by preparing for the worst—j assumes that *a is capable to attack any VM instance anytime*. Under this assumption, for any VM instance $i \in R(j) \cup \{j\}$, if i has been active for a duration of $t_i - 1$ in the past, the probability that "i will be compromised if it is chosen as the migration destination" is $P_{a,i}(t_i)$ the next time it is targeted by the attacker.

Let $E_{a,j}(i) = P_{a,i}(t_i)$ be j's estimation of the risk of migrating to $i \in R(j) \cup \{j\}$. For $\delta > 0$, let

$$R'_\delta(j) = \{i | i \in R(j) \text{ and } E_{a,j}(i) \leq E_{a,j}(j) - \delta\}. \tag{2}$$

$R'_\delta(j)$ is the set of j's replacements that have an estimated risk level that is lower than the risk level of j by at least an amount of δ.

j makes a probabilistic decision to migrate to $i \in R'_\delta(j) \cup \{j\}$ as follows:

- If $R'_\delta(j) = \emptyset$, the decision is (trivially) "migrate to j", i.e., not migrating.
- Otherwise $R'_\delta(j) \neq \emptyset$, j migrates to $i \in R'_\delta(j) \cup \{j\}$ with a probability of

$$M^{\rightarrow}_{a,j}(i) = \frac{\sum_{k \in R'_\delta(j) \cup \{j\} \setminus \{i\}} E_{a,j}(k)}{|R'_\delta| \sum_{k \in R'_\delta(j) \cup \{j\}} E_{a,j}(k)}. \tag{3}$$

The intuition behind this proposed strategy (in particular, Equations (2) and (3)) is as follows.

- Random migration prevents the attacker from exploiting an otherwise fixed schedule to circumvent MTD.
- Confining migration targets to replacements that have lower estimated risk level minimizes bad migration decisions, which include, for example, migrating to a highly risky replacement or thrashing between several VM instances, which wastes valuable computation resources on unnecessary migration.
- By Equation (3):
  - $\sum_{i \in R'_\delta(j) \cup \{j\}} M^{\rightarrow}_{a,j}(i) = 1$. Thus, $M^{\rightarrow}_{a,j}(i)$ ($i \in R'_\delta(j) \cup \{j\}$) constitutes a probabilistic partition of 1.
  - A VM instance i with a higher estimated risk level $E_{a,j}(i)$ will have a lower probability $M^{\rightarrow}_{a,j}(i)$ of being selected as the migration target. Candidates that have the same estimated risk level have the same probability of being selected as the migration target.

Some illustrative numerical examples are as follows:

- Let the risk estimation of 3 migration candidates be 3 (by definition, this must be j), 2, and 1, the corresponding probabilities of them being selected are 1/4, 1/3, and 5/12, respectively. $1/4 + 1/3 + 5/12 = 1$ and $1/4 : 1/3 : 5/12 = 3 : 4 : 5$.
- Let the risk estimation of 3 migration candidates be 5 (by definition, this must be j), 0, and 0, the corresponding probabilities of them being selected are 0, 1/2, and 1/2, respectively.

IV. Evaluation

A. Setup

To evaluate the proposed MTD service deployment strategy, we implement it, along with two other service deployment strategies, in Common Lisp and compare their effectiveness in protecting service from attack through simulation[1]. The two other service deployment strategies are:

- *Static*. The set of active VM instances does not change after the service has been deployed.
- *Rotate*. Each of the active VM instance makes a probabilistic decision of either not migrating or migrating to one of its replacements that have a shorter deployment history, i.e., have been used as an active VM instance in the past for fewer units of time. More generally, an eligible replacement should have a deployment history that is shorter by at least δ ($delta > 0$) units of time. Choosing a reasonable δ allows the service to balance between constant migration (which prevents the service from doing any useful work) to no migration at all (which degenerates the strategy into the static one).

We expect a comparative study between these alternative service deployment strategies will elucidate the following points.

- Both "rotate" and the proposed strategy are instances of MTD, and share a similar decision process. The difference between the two is that, for a Cloud-based service with heterogeneous attack surfaces (as modeled by a variety of risk accumulation functions of different VM instances as shown in, for example, Figure 1), "rotate" is oblivious to, while the proposed strategy is aware of, the heterogenity of risk accumulation. For this reason, we call the proposed strategy the *"risk-aware"* (short for "attack-surface heterogeneity-and-dynamics aware") service deployment strategy. A comparison between "rotate" and "risk-aware" shows whether, and to what extent, attack-surface heterogeneity-and-dynamics awareness makes MTD more effective.
- A comparison between the MTD (i.e., "rotate" and "risk-aware") and the "static" strategy shows whether, and to what extent, MTD is effective.

We originally considered both a *targeted* and a *random* attacker model. The difference between the two is that, whereas an attacker of the random model does not know the service deployment and, hence, picks targets *randomly* during each round, an attacker of the targeted model knows exactly the service deployment and precisely attacks the active VM instances during each round. Although a targeted attacker is more capable than a random attacker, the random attacker model is actually more effective for evaluating whether MTD/risk-awareness is effective, for the following reasons.

The random attacker model is more discerning. A static service is doomed under a targeted attack—every attack attempt hits the target. A MTD service fares better due to the amortization of risk accumulation across the pool. However, the favorable result towards MTD under the targeted attack is well expected (and hence uninteresting) and will give an exaggerated impression on how effective MTD actually is. In comparison, under the random attacker model, there is no obvious reason why a MTD service is more resilient than a static one. It is conceivable that, under a random attack, a MTD service takes more hits than a static one—the service does not know whether the VM instance that it chooses to migrate next will also be chosen by the attacker.

The random attacker model is more general. A targeted attacker is one who is either an insider or has studied the target long enough to act like an insider. However, many attacks are preceded by a stage of reconnaissance, during which the attacker acts more like a random attacker. Besides, a targeted attacker is more predictable than a random attacker, which can make a targeted attacker to be detected and countered more predictably; thus, even a targeted attacker may act randomly to circumvent such targeted counterattack.

Therefore, the results presented henceforth are obtained from simulations using a random attacker model.

In the simulation, the whole pool of VM instance is represented by an undirected graph of nodes with neighbors representing mutually replaceable VM instances—mutuality of the replacement relationship gives rise to the undirectness of the graph. We simulate the attack surface heterogeniety by randomly assigning to each node one of the functions shown in Figure 1. The model presented in Section II is simulated as a *game* between an attacker and three defenders, each representing one of the three service deployment strategies (static, rotate, and risk-aware). The game evolves by discrete rounds, each of which proceeds as follows.

- The attacker chooses targets to attack randomly and independently, and is constrained by its budgets on how many targets it can pick during each round.
- Each of the defenders chooses its current active VM instances (i.e., its *assets*), on which the service is deployed. The static defender always chooses the same assets; the MTD defenders (rotate and risk-aware) make probabilistic decisions on whether and where to migrate their assets, as specified in Section III and the beginning of this section.
- For each of the overlapped choices between a pair of attacker/defender (i.e., an asset is being targeted by the attacker in this round), a biased dice is cast based on the particular asset's risk accumulation function, to decide whether the asset is actually captured by the attacker.

The effectiveness of the defenders is measured by the *asset survival rate*, i.e., the percentage of defenders' assets that have not been captured by the attacker yet. The results shown in Figure 2 are conducted with 512 nodes, with different combinations of attacker budgets and initial numbers of defenders' assets. To reduce statistical bias, the simulation is repeated 100 times with different random seeds and the five-number summaries (i.e., the minimum, the 25% quantile, the median, the 75% quantile, and the maximum) are plotted over a period of 250 units of time. The migration threshold δ for the rotate service deployment strategy is chosen to be 4.

B. Results and explanation

The main results shown in Figure 2 are:

- When the service is dense (i.e., has a high ratio of asset over the pool and is shown towards the right of Figure 2)

[1] The source code for the implementation of the strategies and the simulator is publicly available at https://github.com/pw4ever/pw-sim-mtd

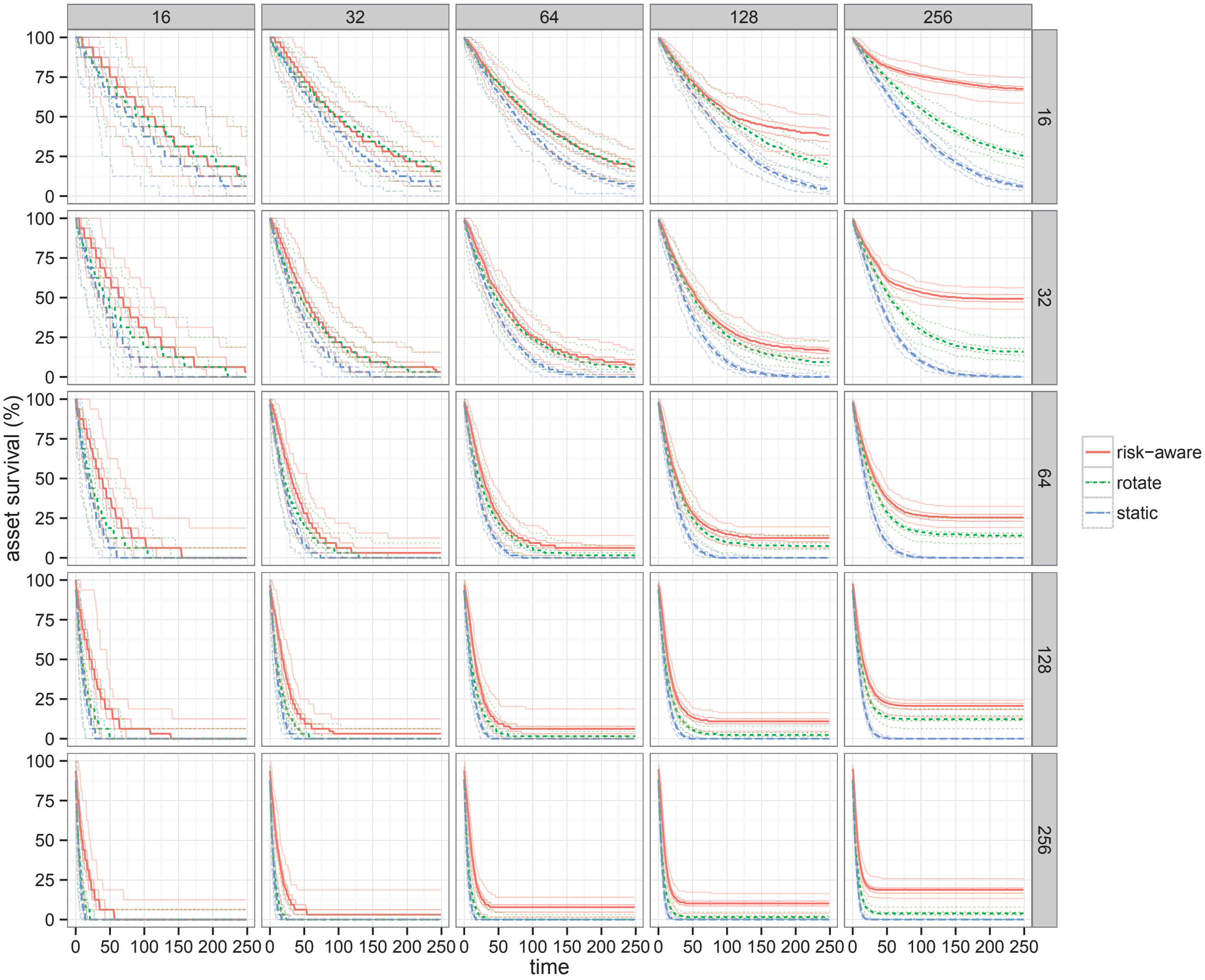

Fig. 2: When the service is dense (i.e., has a high ratio of asset over the pool and is shown towards the right of the figure) and/or the attacker is strong (i.e., has a large attack budget and is shown towards the bottom of the figure), *an MTD service is significantly more resilient than a static one.* When the service is sparse (i.e., has a low ratio of asset over the pool and is shown towards the left of the figure) *and* the attacker is weak (i.e., has a small attack budget and is shown towards the top of the figure), an MTD service *does not* show significant advantage over a static one in resilience (although risk-awareness still helps). Risk (short for *attack-surface heterogeneity-and-dynamics*) awareness helps in improving MTD effectiveness, especially for cases in which a MTD service is significantly more resilient than a static one. The shown results are obtained through a simulation of 512 nodes over 100 random repetitions. Column headers show the number of initial defender assets. Row headers show the attacker budgets. The five-number summaries (i.e., the minimum, the 25% quantile, the median, the 75% quantile, and the maximum) of both static and MTD defenders' asset survival rate are plotted over a period of 250 units of discrete time.

and/or the attacker is strong (i.e., has a large attack budget and is shown towards the bottom of Figure 2), *an MTD service is significantly more resilient (i.e., has a higher asset survival rate) than a static one.*

- When the service is sparse (i.e., has a low ratio of asset over the pool and is shown towards the left of Figure 2) *and* the attacker is weak (i.e., has a small attack budget and is shown towards the top of Figure 2), an MTD service *does not* show significant advantage over a static one in resilience.
- Risk awareness helps in improving MTD effectiveness, especially for the cases in which a MTD service is significantly more resilient than a static one.

An explanation for the results is:

- When the service is dense and/or the attacker is strong, there is a high probability that an attacker (even a random one) will hit a static service. This is demonstrated by the significant changes of the static service's asset survival rate within each column from dropping to (almost) 0% at time 250 for an attacker with a budget of 16 (16/512=1/32 of the pool), to dropping to 0% before time 25 for an attacker with a budget of 256 (256/512=1/2 of the node pool).
- Although an MTD service is equally likely to be hit as a static one, the risk (of the service being compromised by the attacker) is amortized among the pool of replacements through migration. Therefore, over the same period of time, although more assets have been activated (and hence potentially been probed and attacked), fewer has reached a risk level high enough to be compromised.

- When the service is sparse *and* the attacker is weak, the very sparsity serves as adequate camouflage for a static service against the weak attacker, so that MTD does not show significant benefits. Nevertheless, risk awareness helps in improving security, as demonstrated by the better (although only slightly) median asset survival rate of the risk-aware service deployment strategy, even for the cases shown in the top-left corner of Figure 2 (which correspond to sparse services with weak attackers).
- Risk awareness, as embodied in the proposed strategy presented in Section III, helps avoid poor decisions such as migrating an asset from a lowly risky but more used node to a highly risky but less used one. This demonstrates the need and security benefits for an *accurate* attack surface model.

V. Related Works

Moving-target defense (MTD) emerges as a cyber-security research topic since mid-2010 [8, 13, 17]. Mutable Network (MUTE) [3] implements network MTD through random address-hopping and false OS/application probing responses. Secure Overlay Service (SOS) [11] and Rebound Wall [7] propose similar mechanisms of changing system components to thwart specific network attacks. Self-Cleansing Intrusion Tolerant [10, 15] exploits redundancy of computing resources to manage systems' exposure to security threats, much like the refreshing service migration model discussed in Section II-A. Our work studies the more challenging (for the defender) snapshot-and-restore migration model.

Several recent works approach MTD through game theory [5, 19], control theory [16], and genetic algorithm [6]. Zhuang et al. illustrate the design of a MTD system over an example network mission planning system and validate their design by simulating a probabilistic attacker model [20]. In comparison, we explicitly consider the temporal dynamics of the attacker model.

Advancement in virtualization technology makes live migrating of VMs without interrupting services a reality [4, 12]. Previously, live migration are used in load balancing [9, 18] and resource allocation optimization [1, 2]. Liu et al. demonstrate a heterogeneous VM migration system that allows a VM to migrate between two different VMMs [12].

VI. Conclusion

In this paper, we model a Cloud-based service with heterogeneous and dynamic attack surfaces and propose/evaluate a probabilistic service deployment strategy that exploits such heterogeneity and dynamics for making the service more resilient against attacks. Through simulation, we identify the conditions and extent of the proposed MTD strategy's effectiveness in protecting Cloud-based services with heterogeneous and dynamic attack surfaces.

Our work suggests a few directions for future research. First, the benefits of exploiting attack-surface heterogeneity/dynamics awareness in improving service resilience show the need for *accurate* attack surface modeling. Second, the robustness of risk-aware MTD against the deviation of risk estimation from reality should be studied in cases where accurate estimation is not feasible. Last but not least, migration as a security measure has its own costs (in terms of service interruption or consumed communication/computing resources). Although we implicitly consider it by minimizing unnecessary migration through probabilistic and thresholded strategy, a future extension to our work could explicitly model such costs in a scale that is commensurate to the security benefits of MTD.

Acknowledgment. This research is supported in part by NSF grants CNS-1262984, CNS-0916857, DUE-1303325, and Northrop Grumman NGCRC #4301.

References

[1] Resource allocation algorithms for virtualized service hosting platforms. *Journal of Parallel and Distributed Computing*, 70(9):962 – 974, 2010.

[2] Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755 – 768, 2012.

[3] E. Al-Shaer. Toward network configuration randomization for moving target defense. In *Moving Target Defense*, volume 54 of *Advances in Information Security*, pages 153–159. Springer New York, 2011.

[4] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Procs. of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI)*, 2005.

[5] Richard Colbaugh and Kristin Glass. Predictive moving target defense. In *Proc. of Moving Target Research Symposium*, 2012.

[6] Michael Crouse, Errin Fulp, and Daniel Canas. Improving the diversity defense of genetic algorithm-based moving target approaches. In *Proc. of Moving Target Research Symposium*, 2012.

[7] Y. Dai, X. Li, X. Zou, and F. Li. Rebound wall: A novel technology against dos attacks. *Special Issue on System Survivability and Defense against External Impacts, International Journal of Performability Engineering*, 5(1):55–70, 2009.

[8] D. Evans, A. Nguyen-Tuong, and J. Knight. Effectiveness of moving target defenses. In *Moving Target Defense*, volume 54 of *Advances in Information Security*, pages 29–48. Springer New York, 2011.

[9] J. Hu, J. Gu, G. Sun, and T. Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Procs. of 2010 Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, 2010.

[10] Y. Huang and A. Sood. Self-cleansing systems for intrusion containment. In *Procs. of Workshop on Self-Healing, Adaptive, and Self-Managed Systems (SHAMAN)*, 2002.

[11] A. Keromytis, V. Misra, and D. Rubenstein. Sos: secure overlay services. In *Proc. of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications(SIGCOMM)*, pages 61–72, 2002.

[12] P. Liu, Z. Yang, X. Song, Y. Zhou, H. Chen, and B. Zang. Heterogeneous live migration of virtual machines. In *In International Workshop on Virtualization Technology (IWVT)*, 2008.

[13] P. Manadhata and J. Wing. A formal model for a systems attack surface. In *Moving Target Defense*, volume 54 of *Advances in Information Security*, pages 1–28. Springer New York, 2011.

[14] P. Manadhata and J. Wing. An attack surface metric. *IEEE Transactions on Software Engineering*, 37(3):371–386, 2011.

[15] Q. Nguyen and A. Sood. Designing scit architecture pattern in a cloud-based environment. In *Procs. of International Workshop on Dependability of Clouds, Data Centers and Virtual Computing Environments(DCDV)*, 2011.

[16] Jeff Rowe, Karl N. Levitt, Tufan Demir, and Robert Erbacher. Artificial diversity as maneuvers in a control theoretic moving target defense. In *Proc. of Moving Target Research Symposium*, 2012.

[17] F. Sheldon and C. Vishik. Moving Toward Trustworthy Systems: R&D Essentials. *IEEE Computer*, 43:31–40, 2010.

[18] A. Singh, M. Korupolu, and D. Mohapatra. Server-storage virtualization: integration and load balancing in data centers. In *Procs. of the 2008 ACM/IEEE conference on Supercomputing*, 2008.

[19] Yulong Zhang, Min Li, Kun Bai, Meng Yu, and Wanyu Zang. Incentive compatible moving target defense against VM-colocation attacks in clouds. In *Information Security and Privacy Research*. Springer, 2012.

[20] Rui Zhuang, Su Zhang, Scott DeLoach, Xinming Ou, and Anoop Singhal. Simulation-based approaches to studying effectiveness of moving-target network defense. In *Proc. of Moving Target Research Symposium*, 2012.