Frequency-Minimal Utility-Maximal Moving Target Defense Against DDoS in SDN-Based Systems

Saptarshi Debroy, *Member, IEEE*, Prasad Calyam, *Senior Member, IEEE*, Minh Nguyen, Roshan Lal Neupane, *Student Member, IEEE*, Bidyut Mukherjee, *Student Member, IEEE*, Ajay Kumar Eeralla, and Khaled Salah

***Abstract*—With the increase of DDoS attacks, resource adaptation schemes need to be effective to protect critical cloud-hosted applications. Specifically, they need to be adaptable to attack behavior, and be dynamic in terms of resource utilization. In this paper, we propose an intelligent strategy for proactive and reactive application migration by leveraging the concept of 'moving target defense' (MTD). The novelty of our approach lies in: (a) stochastic proactive migration frequency minimization across heterogeneous cloud resources to optimize migration management overheads, (b) market-driven migration location selection during proactive migration to optimize resource utilization, cloud service providers (CSPs) cost and user quality of experience, and (c) fast converging cost-minimizing reactive migration coupled with a 'false reality' pretense to reduce the future attack success probability. We evaluate the effectiveness of our proposed MTD-based defense strategy using a Software-defined Networking (SDN) enabled GENI Cloud testbed for a "Just-in-time news articles and video feeds" application. Our frequency minimization results show more than 40% reduction in DDoS attack success rate in the best cases when compared to the traditional periodic migration schemes on homogeneous cloud resources. The results also show that our market-driven migration location selection strategy decreases CSP cost and increases resource utilization by 30%.**

***Index Terms*—Cloud security, DDoS attack, moving target defense, software-defined networking.**

I. Introduction

With the growing trend of hosting critical applications in, e.g., finance, biotechnology, and healthcare on cloud platforms, there is a need to protect these applications from the security threats of cyber attacks. One of the most common type of cyber attacks targeting cloud infrastructures is the Distributed Denial of Service (DDoS) attack [1] that leads to Loss of Availability (LOA) through starvation of critical application resources serving legitimate users. Lack of adequate defense and recovery strategies to counter against such attacks can impact cloud service provider (CSP) reputation and cause millions of dollars in damages to cloud tenants.

The DDoS attack defense challenges within a cloud infrastructure are more severe than traditional cyber security risks in two ways. Firstly, a cloud infrastructure becomes a 'vulnerability amplifier' to traditional cyber security threats due to the highly elastic nature of the infrastructure resources designed to serve a large population of consumers. Secondly, new means of DDoS attacks exist that specifically target cloud infrastructures in vulnerable areas of application multi-tenancy within a virtual machine (VM), and within an internal network of a CSP. Moreover, traditional 'detect-and-react' defense approaches [2], [3] are largely ineffective in consistently maintaining the Service Level Agreements (SLA) when under DDoS attack due to their lack of: (a) agility in response to attack detection, (b) cost effectiveness for the CSP, and (c) sophistication to tackle intelligent attack strategies. Consequently, as an alternative, the cloud security community and even federal organizations [4] are exploring 'Cyber Agility and Defensive Maneuver' (CAADM) mechanisms that can allow for real-time service restoration through agile cloud resource adaptations once an attack is detected. The same mechanisms can also limit proliferation of detected attacks within the cloud infrastructure through preventive VM resource maneuvers.

Amongst the CAADM strategies, the Moving Target Defense (MTD) based resource obfuscation/adaptation mechanisms can be effective to protect critical cloud-hosted applications. For instance, MTD-based mechanisms can be used to perform both: (i) *proactive* resource adaptation, to detect a DDoS attack and act defensively before major damage is inflicted, and (ii) *reactive* resource adaptation, to act defensively after an attack has occurred. At the same time, MTD-based mechanisms are amenable to leverage the emerging Software-defined Networking (SDN) [5] paradigm to achieve dynamic network resource management.

However, there are three distinct issues that make the design of such MTD-based CAADM strategies in SDN-based systems non-trivial. Firstly, with every dynamic resource adaptation, the CSP encounters cost involving wastage of cloud network/compute/storage resources, which becomes especially prohibitive for proactive adaptations. The alternate approach of

Manuscript received June 14, 2019; revised November 8, 2019, January 23, 2020, and February 9, 2020; accepted February 10, 2020. Date of publication March 4, 2020; date of current version June 10, 2020. This material is based upon work supported by the National Science Foundation under Award Number: CNS-1359125 and Thomson Reuters. The associate editor coordinating the review of this article and approving it for publication was Q. Li. *(Corresponding author: Prasad Calyam.)*

The authors are with the Department of Computer Science, City University of New York, New York, NY 10017 USA, also with the Department of Electrical Engineering and Computer Science, University of Missouri–Columbia, Columbia, MO 65211 USA, and also with the Department of Computer Engineering, Khalifa University, Abu Dhabi, UAE (e-mail: saptarshi.debroy@hunter.cuny.edu; calyamp@missouri.edu; minh.nguyen@hunter.cuny.edu; rlnzq8@mail.missouri.edu; bm346@mail.missouri.edu; ae226@mail.missouri.edu; khaled.salah@kustar.ac.ae).

Digital Object Identifier 10.1109/TNSM.2020.2978425

infrequent adaptations can leave the application vulnerable to DDoS threats. Thus, there is a need to optimize the frequency of proactive adaptations. Secondly, with either proactive or reactive resource adaptations, the legitimate users of a cloud-hosted application will experience service interruptions and quality of experience (QoE) degradations to some extent. Such degradations can be sustained if the resource adaptations are sub-optimal and do not capitalize on the inherent heterogeneity of CSP resources to optimize performance. Thus, there is a need to optimize the adaptations using the elastic resource availability and SDN capabilities in cloud platforms without noticeably impacting the end-user performance. Thirdly, successful MTD-based defense implementations need to possess the potential for deception, wherein a quarantine environment traps the attacker without his/her knowledge to learn more about the attack strategy, while the defense adaptations are progressing to continue service to legitimate users.

In this paper, we address the above fundamental MTD-based defense design issues within SDN-enabled cloud infrastructures. Our MTD-based defense solution is "dual-mode operational" in the sense that it allows for proactive migration of target application in a VM for impending attacks, and triggers reactive migration in the event of an LOA attack detection. Our solution novelty is in the frequency minimization and consequent ideal location selection of the target application across heterogeneous VMs based on LOA attack probability, which in turn minimizes cloud resource wastage without affecting application QoE. The core principle guiding our solution is that the ideal frequency of migration to avert an LOA attack should be frequent enough to avoid vulnerability, i.e., the frequency should minimize the probability of a VM being attacked before migration. To realize this approach, we compare the attack probability and migration interval selection for different attack parameters. We demonstrate that with higher values of attack parameters (signifying stronger attacks), the more frequent migration is necessary.

To find the ideal VM location to migrate, we propose an optimal market-driven scheme that is based upon distributed optimization principles. Our market-driven scheme uses virtual market economics in order to optimize resource allocations during migration. We borrow virtual market economic foundations that have been successful in other large-scale complex networked systems such as, e.g., power grids. We apply a cross-disciplinary strategy in the context of a CSP market where applications and their subscribers behave as buyers who require maximized utility in terms of resources allocated and low cyber attack risk. The CSPs behave as sellers trying to provide optimal resources at low cost. Our scheme also proposes a VM reputation scheme that is based on rewards and penalties given to VMs on a longer time scale, based on their history of thwarting and falling prey to DDoS attack threats. Once an ideal migration location is chosen and the target application is migrated, all the application users are redirected to the chosen destination VM using an SDN controller directing OpenFlow [6] switches within the cloud infrastructure.

As part of our reactive defense mechanism, we devise a dummy-traffic based *false reality* environment to trick attackers into thinking their attack is still in progress, while being quarantined for monitoring and logging. As part of this pretense environment, a dedicated VM is used to create and send dummy user packets to the attacked server. This creates a false perception to the attacker that users are continuing to connect to the attacked server while in actuality the application has already been migrated to a new server, and all legitimate users properly redirected to it. As soon as the attack is detected, migration and *false reality* get triggered simultaneously so as to: (a) prevent attackers from recognizing attack failure, identification of a high-value target and retrying with greater resources, (b) sustain affected services for longer time to collect data to analyze the signature and pattern of attackers, to be prepared for future attacks with minimal increase in the overall CSP cost.

We evaluate our DDoS attack defense scheme using a GENI Cloud [7] testbed that features cyber attack templates affecting different types of cloud-hosted applications ranging from a "just-in-time" news article and video delivery services. These applications provide a unique target use case with multiple vulnerability situations, attack model, user QoE degradations and migration complications. From the collected results, we show how our proactive scheme successfully performs migrations that protect the target applications from DDoS attacks with a very low attack success rate. Using human subject experiments, we also show how our reactive scheme mitigates impact of QoE degradations to the application consumer by a timely migration to a chosen destination VM during a DDoS attack situation. Through extensive simulations of migrations with diverse optimization objectives, we demonstrate that our proposed scheme consistently outperforms other existing greedy schemes which are largely based on centralized optimization principles. Finally, we show that our dummy-based *false reality* pretense environment successfully tricks an attacker with a false sense of success without substantially increasing the overall CSP cost.

The rest of the paper is organized as follows: Section II discusses the related work. Section III discusses the system and attack model and outlines the overall scheme. Section IV outlines the design challenges and optimization problems. Section V presents the analysis of the design and implementation optimizations. Section VI discusses the testbed implementation and performance evaluations. Section VII concludes the paper.

II. Related Work

A. Traditional Cloud DDoS Defense Approaches

Recent works, such as [8] has shown that among different types of cloud attacks, LoA through DoS variants will statistically continue to remain one of the top threats to cloud infrastructures. Some of the traditional works in the literature that target cloud LOA attacks are [9]–[14]. Among these, authors in [9], [10] proposed router filtering approaches to avoid DDoS attacks. In addition, works such as [11], [12] propose different intrusion prevention system (IPS) mechanisms for DDoS avoidance. Further, security strategies against LOA also feature new cache design approaches such as [13], [14]. Similarly, work in [15] monitors shifts in traffic patterns due to

DDoS attacks and guesses the attack source/transit networks to implement LOA defense. *Our work is unique compared to these traditional approaches for defense against DDoS attacks on cloud services because we use dynamic resource allocation mechanisms to adapt to intelligent attacker behavior.*

B. MTD Based CAADM Approaches

There have been prior works on DDoS attack characterization such [16] that prompt actions such as packet discarding. However, MTD based CAADM works are recently gaining momentum in tackling cloud based threats that are predominantly reactive in nature, i.e., the defense scheme kicks in once an attack is identified. Among these, [17]–[21] are notable. In [17], authors propose a shuffling technique of static IP addresses of attacked VMs. Authors in [18] propose a scheme to move around proxy servers to an application server in order to thwart attacks. Other notable works that apply MTD proactively against cyber attacks can be seen in recent works of [19] and [20]; authors in [19] periodically and proactively replace one or more proxies and remap clients to proxies, whereas we propose in a recent work [20], an approach that creates multiple VM-replicas of critical services and assigns VM replicas' IP addresses using address space randomization. In [21], the authors propose a MTD strategy to marginalize the attackers within a small pool of decoy VMs. In recent works such as Graphene [22] and Decima [23], authors have proposed job scheduling techniques on heterogeneous resources using machine learning methodologies for virtual machine selection. Our work differs in terms of the primary optimization objectives focused on virtual machine selection based on reputation estimation during a DDoS attack. *Although most of the other works claim to successfully misdirect the attackers, few of them are proactive or reactive in nature and do not consider addressing situations such as delivering consistency in user QoE before-and-after an attack event.*

C. SDN Enabled Cloud Security Strategies

SDN enabled CAADM works have also been recently proposed for cloud infrastructures which include both MTD and non-MTD based strategies. Among these, works such as, e.g., [24]–[28] are notable. The work in [24] uses network function virtualization capabilities to elastically vary the type and scale of DDoS defense policies in the defense VMs, and SDN is used to flexibly steer suspicious traffic through the defense VMs. A non-MTD based strategy is adopted in [25], where adaptive correlation of analysis is done on attack features in suspicious flows and attacker or victim traffic is throttled. In [26], the authors propose a VM IP address mutation scheme that uses OpenFlow to route cloud users to the target application using the updated IP address. Authors in [27] studied the benefits and overheads of SDN-enabled MTD schemes for VM migration. The closest related work that uses a proactive security strategy using MTD which is similar to our scheme is [28]. Therein, the authors use an online VM migration strategy by predicting impending attacks using attack traffic signature pattern recognition. *However, such works assume a homogeneous VM pool that may or may not guarantee consistent user QoE before and after migration with little-to-no cost benefit analysis from a CSP point of view.*

D. Cloud DDoS Defense Strategies

Among the recent works that propose methods to deflect and divert DDoS attacks, [29]–[32] are notable. An architectural support for defense is utilized in [29], where a centralized control mode is altered with a programmable distributed data plane during attacks in order to adapt traffic routing for attack mitigation. Authors in [30] highlight the research challenges and solution approaches of SDN enabled DDoS defense mechanisms. In [31], the authors provide an overview of a MTD based scheme for preventing DDoS attacks on distributed systems. However, most of such works do not consider strategies to divert future attacks similar to how we consider as part of the overall defense strategy. The most notable existing work that compares to our work is [32], where the authors used decoy VMs to mislead the attackers. They further suggested that there exists a bound on the differences of mean response times between dummy and real VMs for the attacker to notice. *Thus, our work is the first effort to incorporate attacker deception in SDN enabled MTD strategies against DDoS attacks in cloud infrastructures.*

III. System Model and Scheme Overview

A. System and Attack Model

Our system model consists of a cloud application being hosted by the CSP in a VM environment. The CSP internal network connects the application to its consumers/users through a SDN controller that manages Open vSwitches as shown in Figure 1 for dynamic switching and routing. We assume that the SDN controller is logically connected to an authentication server that is utilized for intrusion detection and intruder identification. Thus, the controller has knowledge of traffic from legitimate users and attackers. As shown in Figure 1, the users access the application through the SDN controller driven Open vSwitches. The figure illustrates a scenario after the migration and false reality initiation using a Dummy VM when the regular clients' traffic goes along the *regular path*. The attacker is contained at the VM hosting the target application along the *attack path* with a dummy user load to deceive the attacker that the attack is still successful. We assume that the IP addresses of the VMs hosting applications are hidden from the end-users and managed by the SDN controller.

In order to model the application resource requirements and the CSP resource availability, we assume that each application a_k has QoS requirement bounds $\{Q_k^{max}, Q_k^{min}\}$, where Q_k^{max} and Q_k^{min} are upper and lower bound vectors, each comprising of different QoS metrics such as bandwidth, delay, and jitter. The QoS requirement bounds $\{Q_k^{max}, Q_k^{min}\}$ directly correlate to cloud resource requirement vectors $\{R_k^{max}, R_k^{min}\}$ where each vector is represented by a_k's upper or lower bounds of network/compute/storage resource requirements. Resource provisioning below R_k^{min} violates SLAs (i.e., users cannot access the application properly), and resource provisioning above R_k^{max} does not produce any user-perceivable performance benefits.

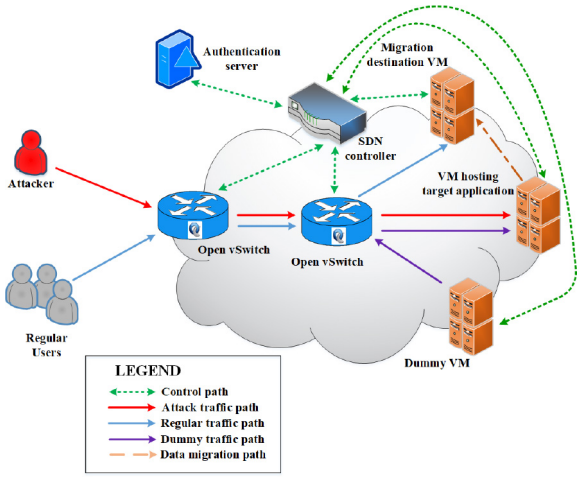


Fig. 1. Proposed MTD based VM migration and false reality initiation with a Dummy VM against DDoS attacks.

We assume that each application has a priority p_k to the CSP that is derived from the "value-at-risk" for that particular service. The CSP decides application priority, and typically mission-critical applications with very strict performance bounds will have higher priority than others. Application priority also drives the application's selection for proactive or reactive migration which is detailed in Section IV. In our model, the SDN controller controls and maintains the resource usage status of all VMs that periodically share status information such as, residual compute/storage capacity with the controller using the *control path* (as shown in Figure 1). Unique from prior works, we assume the VM pool to be heterogeneous, i.e., the VMs have different resource availabilities in terms of levels of compute/storage capacity, network connectivity with varying available bandwidth, and each possessing unique trust/reputation level (e.g., in terms of attack surface) signifying varying ability to handle DDoS attack effects.

We design the scheme for realistic DDoS attacks on cloud environments where the attacker launches both application layer and transport layer attacks to the target application and hosting VM thereby blocking all ports and exhausting all resources (e.g., 'slowloris', 'Slow HTTP POST', and 'Slow Read attack'). In order to optimize the frequency of proactive migration, we analytically model such DDoS attack on a target application to be a Poisson process with exponentially distributed attack inter-arrival times on the VM hosting the target application [33]–[36]. Arrival and departure processes of DoS and DDoS traffic in a network have been typically modeled as Poisson Point Process (PPP) in prior literature [37]–[39]. This is predicated based on the fact that in a data network, any and all traffic is typically modeled as PPP and DDoS traffic regardless of the traffic volume [40]. Our threat model is based on such well known assumptions. Although our mathematical analysis and related results are dependent on the PPP assumption, our proposed overall architecture and solution strategy are valid for modeling the attack to any other distribution (e.g., heavy tailed such as pareto [38]).

In our system and attack model, every VM will experience two states: *Attacked*, when the VM (i.e., the target application it is hosting) is either under attack or being actively probed before an attack, and *Idle*, when the VM is under no active attack. As per Poisson process, the *Attacked* and *Idle* period durations are independent and are exponentially distributed with parameters $\lambda_a = \frac{1}{T_a}$ and $\mu_i = \frac{1}{T_i}$ where T_a and T_i are the expected attack and idle durations. The fraction $\frac{\lambda_a}{\mu_i}$ is called the DDoS 'attack budget' on a particular VM which defines the upper limit of the attack period duration as a fraction of the idle period duration that ensures no attack detection. Such an DDoS attack model on VMs is very similar to the way wireless base stations' transmission pattern is modeled on a particular channel based on its power budget [36]. The overall attack model used for this work is simple, yet representative of general DDoS attacks as modeled in prior works such as [17], [26], [32]. Table I shows the commonly used notations/definitions for the system and attack models.

TABLE I
NOTATIONS USED IN THE SYSTEM AND ATTACK MODELS

a_k	Target application under attack
$\{Q_k^{max}, Q_k^{min}\}$	QoS requirements of a_k
$\{R_k^{max}, R_k^{min}\}$	Resource requirements of a_k
p_k	Priority of application a_k
R_k^J	Allocated resources to a_k at VM v_J
v_J	Candidate VM to host an application
r_J	Reputation of v_J
U_k^J	Utility of a_k when hosted by v_J
MC_k^{IJ}	Migration cost from v_I to v_J
SC_k^I	Snapshot cost at v_I
NC_k^{IJ}	Network cost of from v_I to v_J
Ψ_J	Resource unit price at v_J
DC_k^I	Deployment cost of a_k at v_I
$\lambda_a = 1/T_a$	R.V. for average DDoS attack frequency
$\mu_i = 1/T_i$	R.V. for average idle period frequency
λ_a/μ_i	DDoS attack budget
T_m	VM migration period
γ_J^T	Unit price of one type of resource at VM J

B. Scheme Overview

We propose an SDN-enabled resource adaptation that uses the MTD scheme to cope against DDoS attacks. The SDN controller routing module (as shown in Figure 2) is responsible for managing and performing proactive and reactive migrations. The MTD scheme predominantly adopts a proactive strategy (unless an active attack is detected) where the SDN controller performs dynamic resource adaptations by migrating the applications between VMs. The frequency of the migration is managed by the 'Proactive Migration Location and Frequency Negotiator' as shown in Figure 2. The 'Proactive Migration Location and Frequency Negotiator' is also responsible for new VM location selections while migrating applications. For proactive migration, the module *offline* computes an allocation based on a Least Joint method (discussed in Section IV-B) to identify the best candidate VM with the help of a 'VM Resource Analyzer', and 'Network Bandwidth Analyzer'. During application a_k's migration, the allocated resources (R_k^J) from new VM v_J by the SDN controller is within the range $\{R_k^{max}, R_k^{min}\}$ that also depends on factors such as,

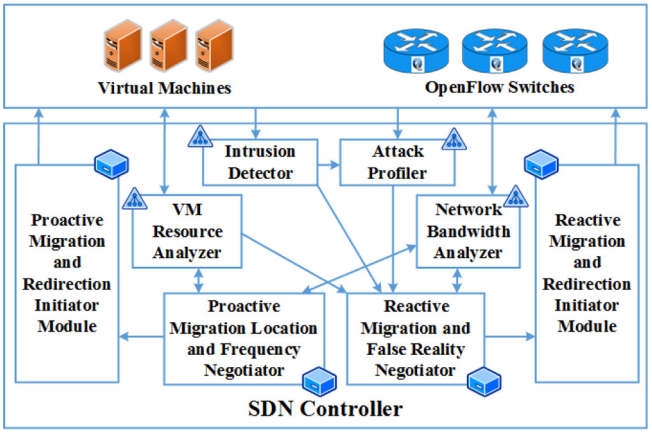


Fig. 2. Software architecture depicting modules responsible for different operations within the SDN controller.

e.g., cloud resource availability at v_J, and application priority p_k. Once the optimal migration frequency and ideal VM location are computed, the SDN controller initiates the migration process through the 'Migration and Redirection Initiator Module', which performs migration by taking a snapshot of the VM currently hosting the migrating application. It also subsequently transfers the snapshot state associated files to the new VM location along the *data migration path* as shown in Figure 1. Upon migration completion, all the application users are re-routed to the new VM using the Open vSwitches.

However, for an active DDoS attack, the entire migration process is preempted by attack detection through an 'Intrusion Detector' that performs intrusion detection, and intruder identification with the help of Authentication server. For the reactive strategy, the 'Reactive Migration and False Reality Negotiator' performs an *online*, fast converging, reactive, and greedy Least Cost approach (discussed in Section IV-B) to decide the new VM location. The SDN controller also ensures that for the user redirection, all the regular users (except for the attackers) are re-routed to the new VM via the Open vSwitches. For the reactive strategy, the SDN controller initiates the false reality through dummy traffic generation from the standby Dummy VMs along the *dummy traffic path* on the VM under attack. The 'Attack Profiler' and 'Network Bandwidth Analyzer' modules feed the 'Reactive Migration and False Reality Negotiator' key information required to calculate just the sufficient amount of dummy traffic to create a pretense that the attack is still succeeding. The 'Reactive Migration and False Reality Negotiator' also performs the cost-benefit analysis of creating a false reality to ensure that the benefits of the pretense over-weigh the associated cost incurred.

IV. Proactive and Reactive Design Challenges

In this section, we formalize the optimization problems for migration frequency, and ideal migration location selection.

A. Problem of Optimizing the Frequency of Proactive Migration

The ideal frequency of VM migration will be such that it is not too infrequent to make the VM vulnerable to DDoS attacks. At the same time, the frequency should not be too often to waste valuable cloud network resources. To solve this frequency optimization problem, we assume the interval between two consecutive migrations of a particular application to be T_m which is infinitely large if there is no DDoS attack, thus minimizing the network resource wastage. However, due to threats of DDoS attacks, T_m needs to be adjusted just enough so that it is less than the DDoS attack inter-arrival time. Thus, the modified optimization problem can be formulated as:

$$\begin{aligned} &maximize \quad (T_m) \\ &s.t. \quad T_m \leq \text{cyber attack inter-arrival time} \end{aligned} \tag{1}$$

T_m will be a function of mean attack period duration and idle period duration (attack inter-arrival times) T_a and T_i, i.e., in other words, T_m will be dependent on λ_a and μ_i, where the exponentially distributed random variable for attack period duration x with mean $T_a = \frac{1}{\lambda_a}$ is given by

$$f_1(x) = \begin{cases} \lambda_a e^{-\lambda_a x} & \forall\ x \geq 0 \\ 0 & \forall\ x < 0 \end{cases} \tag{2}$$

Similarly, the exponentially distributed random variable for idle period duration y with mean $T_i = \frac{1}{\mu_i}$ is given by

$$f_2(y) = \begin{cases} \mu_i e^{-\mu_i y} & \forall\ y \geq 0 \\ 0 & \forall\ y < 0 \end{cases} \tag{3}$$

B. Problem of Selecting the Ideal Migration Location

The objective of the ideal VM selection is a VM v_J's ability to satisfy the application a_k's QoS requirements $\{Q_k^{max}, Q_k^{min}\}$, i.e., in other words its resource requirements $\{R_k^{max}, R_k^{min}\}$. Due to the heterogeneity of the VM pool, we argue that the important factors for such a selection are: (a) v_J's resource (storage/CPU capacity) availability R_J^{avl} that influences the overall application utility U_k^J; (b) the available network bandwidth between the current VM v_I hosting the migrating application and destination v_J that impacts the network cost of migration NC_k^{IJ}; (c) the per unit resource cost Ψ_J at VM v_J that impacts the application deployment cost DC_k^J; and (d) the reputation r_J of VM v_J that impacts the vulnerability of the application to DDoS attacks. The ideal VM location is selected with an aim to maximize the overall CSP utility (U^{CSP}) by using one of the four basic selection schemes: 'Least Cost', 'Least Latency', 'Least Vulnerability' and 'Least Joint', which are described in the following:

Least Cost: Least Cost scheme considers the costs involved in resource price at VM v_J for reserving resources to an application a_k to be migrated. The scheme migrates applications on VMs such that every application is assigned R_k^{min} amount of resources in order to decrease cost. Though the application QoS may not be optimum, the scheme increases U^{CSP} of the system since it reduces operational costs for the CSP.

Least Latency: Least Latency is a utility maximization scheme which maximizes application latency (as perceived at the application user level) by placing applications with close to R_k^{max} resources on the nearest candidate VM from the current VM location. In other words, with other QoS metrics being constant, the overall QoS is inversely proportional to the function of user-perceived latency (greater the latency, lesser is the

QoS). The scheme provides a high U^{CSP} due to increased QoS with latency-awareness between applications and VMs.

Least Vulnerability: Least Vulnerability scheme tries to reduce the future DDoS attack success probability by migrating the application to a VM with very high reputation r_J in terms of DDoS attack history and exposed attack surfaces. This scheme does not mandate the amount of allocated resources and can be combined with Least Cost or Least Latency schemes in order to manage U^{CSP}.

Least Joint: Least Joint scheme combines the optimizations of the above three placement schemes and tries to achieve a higher U^{CSP}. The scheme considers all the factors that influence the overall CSP net utility in terms of: resource availability, application utility, resource price, network latency, and VM reputation.

In our proposed MTD scheme, our objective is to design a Least Joint strategy for *offline* reactive migration that can optimize all the desired factors by finding the ideal migration destination VM. However, for reactive migration, we seek to design an *online* Least Cost scheme in order to compute with a fast converging migration algorithm due to the very small amount of reaction time with a VM under attack. Moreover, establishing an effective *false reality* for DDoS attacks is a challenging proposition. The two main challenges and considerations that need to be accounted includes: (a) the need to build the capability within the reactive migration mechanism in order to seamlessly create an illusion of success with minimal noticeable alteration in attacker QoS, and (b) performing the necessary tasks in a cost-effective way so that the cost of *false reality* for the CSP does not exceed the benefits of keeping the attacker entrapped in a decoy.

V. MTD Strategy Optimization and Analysis

In this section, we present our proactive and reactive migration strategy design, and false reality implementation approach to solve the above optimization problems. We first quantify optimal migration frequency, followed by a market-driven ideal migration location computation, and finally design a cost-effective dummy management based false reality environment implementation.

A. Stochastic Optimization of Migration Frequency

In order to optimize the frequency of proactive migration, let us assume that the random variable representing the attack inter-arrival time be **z** which is the sum of two independent random variables for *Attacked* and *Idle* periods **x** and **y** respectively, i.e., $\mathbf{z} = \mathbf{x} + \mathbf{y}$. Therefore, the distribution of attack inter-arrival time **z** is obtained as:

$$\begin{aligned} f_Z(z) &= f_X(x) * f_Y(y) \\ &= \int_{-\infty}^{+\infty} f_X(z-y) f_Y(y)\, dy \\ &= \begin{cases} \frac{\lambda_a \mu_i \left[e^{-\lambda_a z} - e^{-\mu_i z}\right]}{(\lambda_a - \mu_i)} & \forall\ \lambda_a \neq \mu_i \\ \lambda_a^2 z e^{-\lambda_a z} & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

In order to quantify the optimal T_m, we approach the problem by first calculating the probability of VM getting attacked before migration. Such a probability is expressed as:

$$\begin{aligned} &\text{Prob}\{\text{VM getting attacked before migration}\} \\ &= \text{Prob}\{\text{z} \leq T_m\} \qquad (\text{VM attack being memoryless}) \\ &= \int_{-\infty}^{T_m} f_Z(z)\, dz \\ &= \begin{cases} \int_0^{T_m} \frac{\lambda_a \mu_i \left[e^{-\lambda_a z} - e^{-\mu_i z}\right]}{(\lambda_a - \mu_i)}\, dz & \forall\ \lambda_a \neq \mu_i \\ \int_0^{T_m} \lambda_a^2 z e^{-\lambda_a z}\, dz & \text{otherwise} \end{cases} \\ &= \begin{cases} \frac{\mu_i \left(e^{-\lambda_a T_m} - 1\right) + \lambda_a \left(1 - e^{-\mu_i T_m}\right)}{\lambda_a - \mu_i} & \forall\ \lambda_a \neq \mu_i \\ 1 - e^{-\lambda_a T_m}(\lambda_a T_m + 1) & \text{otherwise} \end{cases} \end{aligned} \tag{5}$$

Now in order to satisfy the condition in optimization Equation (5), probability of VM getting attacked before migration, i.e., $\text{Prob}\{z \leq T_m\}$ needs to be minimized. This reduces the optimization problem from Equation (5) to:

$$minimize\ \ (\text{Prob}\{z \leq T_m\}) \tag{6}$$

However, due to the asymptotic nature of exponentially distributed random variable **z**, the nature of Equation (5) is continuously increasing and asymptotic; and thus does not have any maxima or minima. Therefore, for a particular cyber attack scenario (i.e., with statistical λ_a and μ_i known), the optimal T_m can be evaluated by tuning the desired probability of VM getting attacked before migration, i.e., $\text{Prob}\{z \leq T_m\}$.

B. Market-Driven Migration Utility Optimization

In real-world cloud scenarios, we can view the utility maximization (U^{CSP}) as a CSP-level process which determines ideal resource quantities that provide maximum gain to the CSP. Similarly, the cost minimization process captures the CSP's sentiment to explore the minimum amount it has to pay to maximize the profit out of the bargain for the identified ideal cloud resources. The utility maximization for application migration corresponds to a behavior in which the CSP searches for the ideal VM resource provisioning for each migrating application. Such a search can maximize utility, while the cost minimization step relates to determining a VM location where the least cost of resource consumption is observed. In order to solve this problem, we propose a market-driven "Bid" (B) metric which folds-in the utility maximization and cost minimization into one quantity. It allows for identification of application resource requirements and finds an optimal VM v_{opt} by selecting the VM which produces minimal "Bid" value, where "Bid" value for migrating application a_k from VM v_I to VM v_J is computed by the formulation given below:

$$B_k^{IJ} = U_k^J - MC_k^{IJ} - DC_k^J \tag{7}$$

where for every migrating application a_k, the CSP computes the final "Bid" value $\max(B_k^{IJ})$ by iterating over all VMs. The above Equation (7) consists of several major components that are explained in detail below:

- **Utility** (U_k^J): We model the utility of migrating application a_k to VM v_J as a measure of the migration decision. The VM resource allocation information is supplied via a resource allocation vector (R_k^J) which is a

multi-dimensional vector; each dimension refers to one type of resource available in the VM. The migration information is inherent since we are modeling "Bid" value for every VM. Equation (8) gives the utility which is a dot product of the VM reputation (discussed later) and percentage of maximum resource requirement of a_k that was eventually allocated. Thus, a VM with better reputation and resource allocation closer to the maximum requirement will result in better utility.

$$U_k^J = r_J \times \left(\frac{R_k^J}{R_k^{max}} \right) \tag{8}$$

- **Migration Cost** (MC_k^{IJ}): The migration cost as depicted in Equation (9) captures the cost of migrating a_k from VM v_I to VM v_J. The overall migration cost is the sum of the snapshot cost at VM v_I and the network cost of transferring snapshot files from VM v_I to VM v_J.

$$MC_k^{IJ} = SC_k^I + NC_k^{IJ} \tag{9}$$

- **Deployment Cost** (DC_k^J): The deployment cost or resource usage cost at VM v_J is a measure for the cost associated with accessing the total allocated resources (R_k^J) for a_k from VM *J*. The price vector (γ_J^T) is a multi-dimension vector where each dimension refers to the price of one unit of one type of resource at VM *J*. Hence, the dot product of price vector with resource allocation vector produces the total resource usage cost of accessing R_k^J resources from VM *J*.

$$DC_k^J = \gamma_J^T \times R_k^J \tag{10}$$

- **Quality and resource Constraint**: We use additional constraints for ensuring application users' QoS and corresponding resource requirement satisfaction which focuses on maintaining the user experience even after migration. Through this constraint we ensure that the total allocated resources (R_k^J) for a_k from VM *J* is within a_k's resource requirement bounds $\{R_k^{max}, R_k^{min}\}$ and at the same time, not more than VM v_J's total available resources R_J^{avl}.

$$R_k^J \leq R_J^{avl} \quad \& \quad R_k^{min} \leq R_k^J \leq R_k^{max}$$

- **Reputation** (r_J): We argue that the previous history of a VM in terms of instances of DDoS attacks along with its capability of deflecting future attacks (represented through vulnerability or attack surface) is a critical factor in deciding the VM's suitability to be selected for migration. As the terms 'previous history' and vulnerability are subjective concepts, we translate this into a quantifiable 'VM reputation' which is an objective indicator of how robust a VM is to future DDoS attacks. As the nature and modeling of trust and reputation in cloud environment (both service centric and resource centric) is an active area of research [41] which is independent of a CSP level defense mechanism, we deem further computation and analysis of r_J to be out of the scope of this work.

We assume that the CSP hosting the target application will have deployment/migration cost models for VMs. The actual cost will depend upon the system complexity (i.e., number of different Web services used in the target application services composition) and scale (i.e., number of concurrent users accessing the target application). In our model, the various components (cost, constraints, and reputation) contribute linearly to the final "Bid" value computed by the SDN controller. At the end of first step, we have a $n \times m$ Bid matrix, where n is the number of VMs and m is the number of resources, and reflects the resource allocation vectors for all VMs. Based on the final values of individual VM's Bid matrix, the final decision on the ideal VM location is taken and the migration is initiated. The pseudocode of our market-driven optimization approach is shown in Algorithm 1, which iterates until bids from all VMs are calculated for one migration instance. Upon completion of big calculation, the SDN controller chooses the VM with the highest bid value as the migration destination. Algorithm 2 computes the vector R_k^J that satisfies the application resource requirements and also does not exceed the VM resource availability. If a VM cannot satisfy both conditions, the R_k^J value for that VM is 0. This in turn leads to 0 overall utility and thereby reduces that VM's chances of being selected as the migration location destination.

Algorithm 1: *utilityMaximization* Algorithm

Data: Resource requirement vector $\{R_k^{max}, R_k^{min}\}$ and priority p_k of application to be migrated a_k
Data: Reputation r_J of VM v_J
Data: Snapshot cost SC_k^I of a_k at v_I
Data: Network cost NC_k^{IJ} of migrating a_k from v_I to v_J
Data: Price Ψ_J of one unit of one type of resource at v_J
Data: Allocable resources R_k^J to a_k at VM v_J
Result: Optimal VM v_{opt} and corresponding resource allocation R_k^{opt} that maximizes net utility
for *all candidate VMs J for migration* **do**
  $R_k^J = calculateAllocation(\{R_k^{max}, R_k^{min}\}, p_k)$;
  $U_k^J = r_J \times \frac{R_k^{max}}{R_k^J}$;
  $MC_k^{IJ} = SC_k^I + NC_k^{IJ}$;
  $DC_k^J = \Psi_J \times R_k^J$;
  $B_k^{IJ} = U_k^J - MC_k^{IJ} - DC_k^J$;
end
Find $\bar{J}$ for which $B_k^{I\bar{J}}$ is *maximum*;
Return $v_{opt} = v_{\bar{J}}$;
Return $R_k^{opt} = R_k^{\bar{J}}$;

C. Establishing Dummy Traffic Based False Reality

Statistically, an attacker can differentiate between dummy VM and a real VM hosting an application for multiple users by probing message response times and percentage of dropped packets. In [32], the authors have proved that if the mean response times for the dummy and real VMs are equal to $\frac{1}{rt_d}$, and $\frac{1}{rt_r}$, then the number of responses $\mathcal{K}$ needed for the attacker to distinguish between dummy and real VMs within time T is given as:

$$\mathcal{K} = \arg\min_k \| \left(\frac{rt_r}{rt_d} \right)^k e^{-(rt_r - rt_d)\,T} > C \| \tag{11}$$

Algorithm 2: *calculateAllocation* Algorithm

Data: Resource requirement vector $\{R_k^{max}, R_k^{min}\}$ and priority p_k of application to be migrated a_k
Data: Available resources R_J^{avl} of VM v_J
Result: Allocable resources R_k^J to a_k at VM v_J
if $\left(R_J^{avl} < R_k^{min}\right)$ **then**
  $R_k^J = 0$;
else
  if $\left(p_k \times R_k^{max} < R_k^{min}\right)$ **then**
    $R_k^J = R_k^{min}$;
  else
    $R_k^J = p_k \times R_k^{max}$;
Return R_k^J;

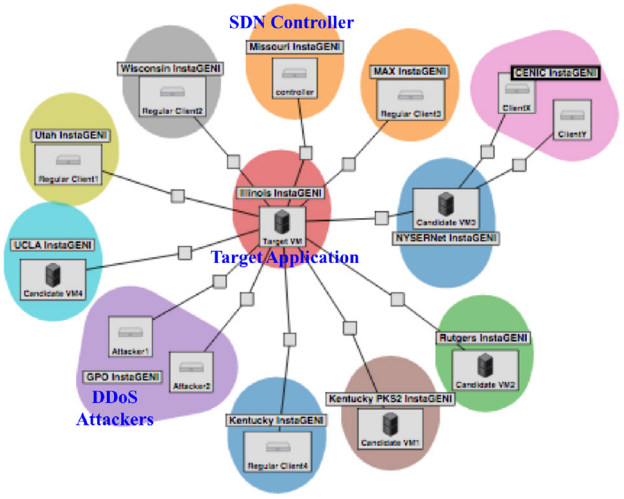


Fig. 3. GENI Cloud testbed topology with VM hosted target application, other candidate and dummy VMs, regular users, DDoS attackers, and SDN controller.

where C is a parameter dependent on the rate of attacker probes and in turn on the attacker's *attack budget*.

Thus, in order for an attacker to quickly distinguish between decoy and real VMs, the attacker has to either: (a) increase the *attack budget*, or (b) rely on the difference between the response times from dummy and real VMs $\frac{1}{rt_d}$ and $\frac{1}{rt_r}$, respectively to be big enough. In other words, greater the difference, higher are the chances of dummy/real VM identification. However, for the first option, increasing the *attack budget* would mean a greater chance of getting detected for the attacker. Thus, in most cases attackers would rely on their ability to precisely differentiate between the response times. Therefore, from the perspective of a CSP, if the false reality can minimize the difference between the dummy and real VM response times, the probability of the attacker detecting a dummy VM within finite amount of time will decrease. Thus, in our MTD with false reality implementation, we will use dummy VMs to generate just the adequate amount of dummy traffic to the VM under attack (after application migration and user redirection) that mimics the realistic traffic pattern of regular users. Such a dummy traffic will ensure that when the attacker returns to intermittent probing periods between prolonged flooding periods, the attacker does not experience any noticeable difference between the behaviors of the VM, with/without the regular users. For simplicity of analysis, we will assume the scenario of a single attacker for analysis, however our propositions will also be valid for multiple attacker scenarios.

Another important consideration while establishing a *false reality* environment is its cost-benefit analysis. Although the benefit of keeping an attacker entrapped and efforts to minimize the chances of future attacks is well motivated, we make a simplistic approach below to quantitatively analyze the cost and benefits of implementing our proposed *false reality* environment. The overall cost of *false reality* implementation ($\mathcal{C}_{\mathrm{FR}}$) to the CSP through dummy VM installation and dummy traffic generation is essentially the cost of CPU utilization ($C_{\mathrm{FR}}^{\mathrm{C}}$) of the VM and the network cost ($C_{\mathrm{FR}}^{\mathrm{N}}$) for dummy traffic generation that can be expressed as:

$$\mathcal{C}_{\mathrm{FR}} = C_{\mathrm{FR}}^{\mathrm{C}} + C_{\mathrm{FR}}^{\mathrm{N}} \tag{12}$$

On the other hand, qualitative benefits of *false reality* are the fact that the regular users experience no or little service interruption boosting the CSP revenue, and continued collection of attack statistics from the VM under attack for more efficent proactive migration strategy design. However, in order to quantify the benefits, we take the approach of measuring the "lost opportunity cost" of *false reality*, i.e., the amount of cloud resource (network and compute) saved by preventing attacker to migrate with the target application and relaunch an attack on the new VM, thereby jeopardizing new reosurces. Such resource saving is in turn equal to the cost of a DDoS attack in terms of cloud resource wastage ($C_{\mathrm{DDoS}}^{\mathrm{C}}$ for compute resource and $C_{\mathrm{DDoS}}^{\mathrm{N}}$ for network resource). Thus, if we ignore the benefits of attack statistics collection which is beyond the scope of this work, then the overall benefits of *false reality* in terms of "lost opportunity cost" is equal to or greater than the cost of DDoS attack, and can be expressed as:

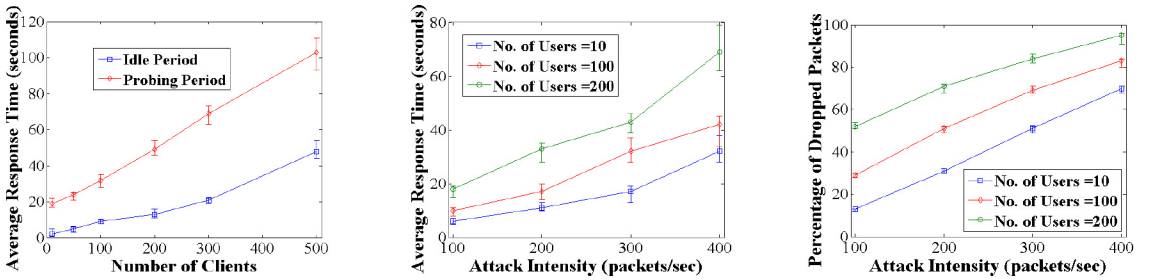
$$\mathcal{B}_{\mathrm{FR}} > \left(C_{\mathrm{DDoS}}^{\mathrm{C}} + C_{\mathrm{DDoS}}^{\mathrm{N}}\right) \tag{13}$$

Therefore, if we compare the costs and benefits of *false reality* from Equations (12) and (13) respectively, implementing *false reality* environment will only be cost effective or optimal, if the CSP defense mechanism obtained from the 'Attack Profiler' satisfies the following conditions:

$$C_{\mathrm{DDoS}}^{\mathrm{C}} + C_{\mathrm{DDoS}}^{\mathrm{N}} > C_{\mathrm{FR}}^{\mathrm{C}} + C_{\mathrm{FR}}^{\mathrm{N}}. \tag{14}$$

VI. Testbed Implementation and Evaluation

In this section, we describe the performance evaluation of our proposed scheme in the GENI Cloud [7] infrastructure. The performance evaluation is three-pronged: (i) we first compare the performance of our proposed frequency-minimal proactive migration scheme (FM-UM) with heterogeneous VM pool against a more periodic MTD scheme that considers a homogeneous VM environment [28], (ii) we then compare the performance of our market-driven utility maximal (FM-UM) migration technique against other one-dimensional migration approaches, such as least cost, least latency, random etc., and



(a) Average response time comparison for different number of user clients

(b) Average response time comparison for different attack intensities

(c) Dropped packet percentage comparison for different attack intensities

Fig. 4. Characteristics of detectable metrics during the DDoS probing stage for different attack intensities.

(iii) finally we showcase the ability of establishing a false reality environment to successfully deceive the attacker against conventional CAADM approaches without false reality.

A. Experiment Setup

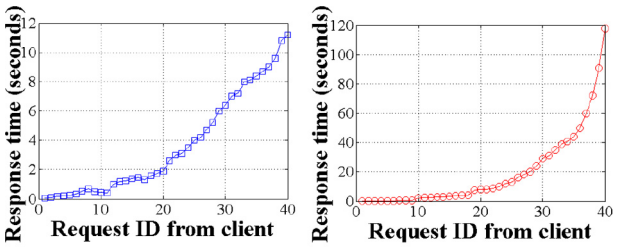
Our experiment setup on a GENI Cloud testbed that we built as shown in Figure 3 consists of the following components:

- *Target application:* We use one VM hosting the DDoS target application at the Illinois InstaGENI location. For one set of experiments, we use a 'Just-in-time news' feed application with a client database. The application supplies the latest news articles via RSS (Rich Site Summary) feeds [42] when it receives HTTP GET requests from the clients. Such an application "Data-at-rest" scenario acts as a ideal targets for DDoS attacks. For another set of experiments with human subjects, we install a 'Just-in-time news' news video server as target application that is used to collect mean opinion scores (MOS) based on user QoE before, during and after attack scenarios.
- *Regular users:* We use four different VM locations generating non-malicious regular user traffic ranging from 1 to 200 users. The users simulate the client side browsers, where they send GET HTTP requests to the target application, and receive article RSS feeds or news video feeds as responses.
- *DDoS attackers:* Two separate VM locations simulate DDoS attacker behavior where they launch application and transport layer attacks on the target application and hosting VM. The consequent impact is that there will be a blocking of all ports and exhaustion of all the application VM related resources. In particular, we install two attackers that use 'slowhttptest' as the attack tool that is a compilation of various DoS attack tools such as 'slowloris', 'Slow HTTP POST', and 'Slow Read attack' (based on TCP persist timer exploit). We use it on two Linux VMs acting as attackers to launch a combined TCP Layer and Application Layer (HTTP request) attack on the target application.
- *Candidate and dummy VMs:* Upto 30 candidate and dummy VMs are installed at different locations simulating varied usage and suitability scenarios (utilization, bandwidth, and reputation) discussed in Sections IV-B and V-C. The simulation randomly assigns the candidate VMs with different values of their properties to form a heterogeneous VM pool. The properties include: storage/compute resource availability, available network bandwidth, a randomly generated binary reputation metric, and a multi-dimensional unit price vector.
- *SDN controller:* The SDN controller along with the software components as shown in Figure 2 is implemented using Python scripts and installed at the Missouri InstaGENI location. For the SDN functionality, we use the open-source POX controller that is available on GENI platform's OVS. For every flow that goes through the OVSes, the forwarding part is dealt at the 'Data Plane'; while the SDN controller governs the 'Control Plane'.

B. Effects of Probing and DDoS Attack

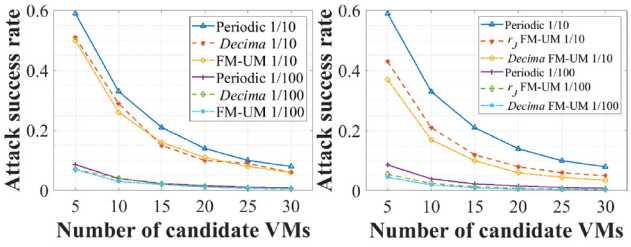
In order to establish the effects of DDoS probing and attack on the target application, we show the attack behavior analysis results in Figure 4. In Figure 4(a), the average response times for the idle and probing periods are compared for different number of users. We can observe that the response time increases during the probing period due to the relatively large number of packets sent by the attacker. Figure 4(b) shows the same average response time metric results for different attack intensities, i.e., number of probing packets/second. We observe that not only the average response time value increases with the attack intensity, but with more users accessing the target application, the attacker is expected to have longer response times for a fixed probing rate. Figure 4(c) shows experiment results comparable to Figure 4(b) for percentage of dropped packets. Similar to the response time, the percentage of dropped packets also increases with the attack intensity, and with higher number of users for a fixed attack intensity.

After the probing, the attackers gradually flood the system with HTTP GET requests to slow down all resources of the VM hosting the target news feed application to stall/disrupt its service. While this attack is being launched, regular users send requests for accessing the resources. The response times for one of the regular users when DDoS attacks are launched from one VM is shown in Figure 5(a), and the response time with attacks from multiple VMs is shown in Figure 5(b). Unlike the probing period, the effects of such attacks in terms of response time are exponential in nature. Thus, the ability of the attackers in minimizing the measurable effects during the probing stage decides the impact of the subsequent DDoS attacks.



(a) DDoS attack with lower intensity (b) DDoS attack with higher intensity

Fig. 5. Response time degradation for low/high DDoS intensity.



(a) Migration comparison without r_J (b) Migration comparison with r_J

Fig. 6. Migration frequency optimization performance.

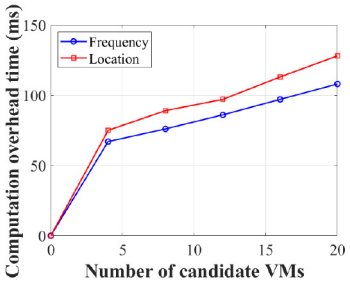


Fig. 7. Computation overhead time characteristics that shows minimal workload for increasing number of candidate VMs.

C. Frequency Optimization Performance Results

We perform proactive migration of the target application using the optimal T_m for FM-UM by fixing the upper bound of tolerable attack success rate (from Equation (5)). We vary the probability of attack on the target application by varying the ratio $\frac{T_a}{T_i}$ that is essentially dependent on the attack budget. We compare the performance improvement over a periodic migration in four cases: (i) migration without candidate reputation r_J, (ii) migration with r_J (assuming 20% of candidate VMs are immune to attacks), (iii) migration on clusters enabled with Decima [23], wherein the model inherently learns the reputation of candidate VMs as well as the attack patterns in order to trigger intermittent migrations, and (iv) migration on clusters enabled with Decima using the optimal T_m for FM-UM. As shown in Figures 6(a) and 6(b), we can observe that for different attack budgets, our proposed proactive migration strategy with optimal migration frequency for clusters performs on par with Decima in similar environment configurations. We also can observe that FM-UM applied on clusters with attack immunity reputation outperforms FM-UM without reputation as well as Decima. Moreover, consideration of clusters enabled with Decima as well as our FM-UM scheme can further improve the system attack resilience compared to only using the FM-UM scheme. Thus, we justify the benefits of our proposed migration frequency optimization and reputation factor consideration. We remark that the proactive migration can be implemented seamlessly in today's cloud platforms that provide sophisticated functionality (e.g., AWS Connector) for virtual machines and related network configuration migrations.

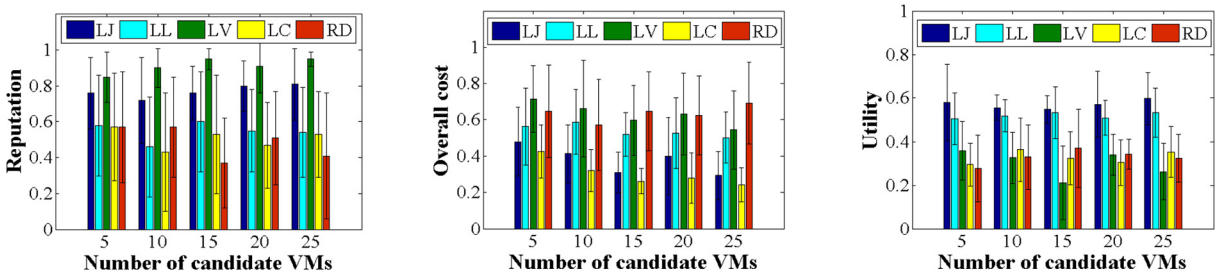
Figure 7 shows the computation overhead time experienced by the SDN controller for running migration frequency optimization and migration utility maximization algorithms. We can see that both characteristics introduce minimal workload even with increasing number of candidate VMs.

D. Utility Optimization Performance Results

In the next set of experiments, we evaluate the performance of the proposed least joint (LJ) based proactive migration against other traditional utility maximization schemes, such as, least cost (LC), least latency (LL), least vulnerability (LV), and random (RD) as explained in Section IV-B. Based on the definition provided in Section IV-B, we program the SDN controller to use these different schemes in order to choose the destination VM from the pool of candidate VMs. Upon selection, we calculate the different market related performance metrics from the selected VM described in Section V-B. Each experiment is run 20 times by randomly changing the VM properties and the average characteristics as shown in Figures 8 and 9.

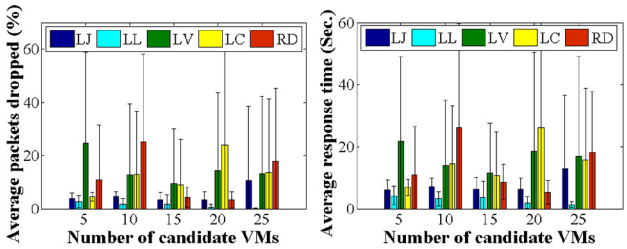
In Figure 8, we show the performance of different proactive migration schemes in terms of maximizing the overall utility, maximizing the reputation of the selected VM, and minimizing the overall cost of the CSP as calculated in Section IV-B. We varied the number of candidate VMs in the system from 5 to 25 to check the generality of the observed characteristics. Figure 8(a) shows that the LV scheme performs the best in terms of choosing the VMs with best reputation scores closely followed by the proposed FM-UM LJ scheme. The observation is consistent for different values of candidate VMs. Similar characteristics can be observed in Figure 8(b) where the best performance is obtained by the LC scheme closely followed by our proposed LJ scheme. The results are consistent given that the one-dimensional LV and LC schemes perform the best as one would expect for the respective metrics (i.e., vulnerability and cost) that the schemes are designed to maximize. However, the proposed LJ method notably performs almost as good as the one-dimensional schemes for their respective metrics. The overall benefit of the proposed LJ scheme is evident in Figure 8(c), where our scheme ensures by 30% better maximum overall utility.

Similar experiment results are shown in Figure 9 corresponding to the QoS metrics such as average packets dropped, and average response time at the user side. In these results, we observe that the LL scheme performs the best when it optimizes the network bandwidth between the old and new VMs. However, similar to Figure 8, the proposed LJ scheme performs almost as good as the LL scheme with the other schemes performing much worse. Figures 8 and 9 overall demonstrate the benefits of LJ as the preferred scheme for VM location selection in case of the proactive migration strategy. This is because of the fact that this scheme optimizes all the properties that determine the suitability of a VM. The



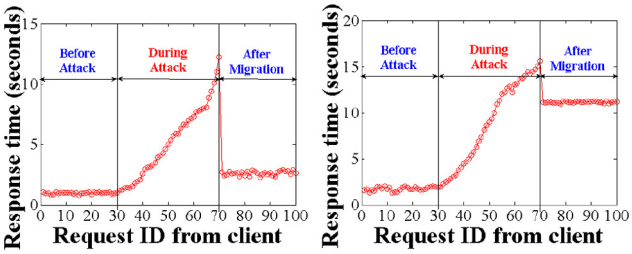
(a) Reputation (b) CSP cost (c) Overall utility

Fig. 8. Comparison of candidate proactive schemes on different performance metrics.



(a) Average packets dropped (b) Average response time

Fig. 9. Comparison of candidate proactive schemes on QoS attributes.



(a) Migration using LC scheme (b) Migration using RD scheme

Fig. 10. Comparison of LC and RD schemes for reactive migration.

other candidate schemes although perform better than the LJ scheme in their respective metrics, no clear favorite emerges among these when all the metrics are jointly considered.

E. Reactive Migration Performance Results

Although we have established that our proposed FM-UM LJ scheme yields best overall performance post migration, such an optimization can only be performed offline in the case of proactive migration. In cases when an active DDoS attack is detected, the CSP needs to initiate a fast converging, greedy, online migration, such as in the LC scheme, in order to migrate the target application and redirect the users quickly/safety without significantly compromising the CSP cost. In order to compare the post-migration QoS effects of the LC and RD reactive migrations, we observe the response time characteristics of one of the regular users as shown in Figures 10(a) and 10(b). Figure 10(a) shows the performance of our proposed LC based reactive migration where the SDN controller chooses Rutgers InstaGENI location as destination VM location because it minimizes the CSP cost. Whereas, if the controller uses a reactive scheme, the NYSERnet InstaGENI location is chosen randomly assuming a homogeneous VM pool. Although not generalized, but we observe that the LC scheme in the process of minimizing the network and compute cost, ends up choosing a VM where such a resource is cheaper as the availability is higher. However, the RD scheme cannot guarantee such a result, and ends up choosing a VM which is busy serving a separate pool of users and having a highly reduced resource availability.

Next, in order to generalize such observations and also to examine if similar effects can be seen when real users are involved, we used the same setup and performed similar experiments, however using a "just-in-time" news video server as the target application (instead of the news article feeds). This time we measure the effects of the attack and subsequent migration using the candidate reactive schemes on human subjects in terms of the user QoE. In this experiment, the DDoS attack is launched while the users are accessing the streaming and then reactive migration is performed by generating different scenarios. The video playback quality before attack, during attack, and after migration is observed and rated using a Mean Opinion Score (MOS) ranking on a scale of 1 (Poor) to 5 (Excellent). We recruit 10 human subjects from a diverse age group of 25-34 with 6:4 proportion of male and female subjects as part of a human subject survey conducted at University of Missouri and using the GENI Cloud testbed.

We created 5 different scenarios, i.e., S1 - S5, where the application is migrated based on random (RD), least joint (LJ), least latency (LL), least cost (LC) with playback from the beginning, and LC with playback from the point of interruption (which was caused by DDoS attack) schemes. Figure 11 shows the average MOS ratings are the highest for scenarios S4 and S5 where the application is migrated using the LC scheme and the ratings are as good as those before the attack. Among the two, the users rated scenario S5 marginally better as the playback was resumed from the point of interruption due to the attack occurrence. Upon interviewing the human subjects, it became evident that the reasons for rating scenario S3 lower was due to the slower reaction time of the LJ scheme for service recovery from the point of interruption due to the attack.

This experiment demonstrates that although in theory the LJ method ensures better overall utility and performance for proactive strategy, a greedy approach, such as LC yields better user QoE due to its quicker response and recovery for the reactive strategy leading to minimal impact of performance. This also demonstrates that our proposed method is light-weight

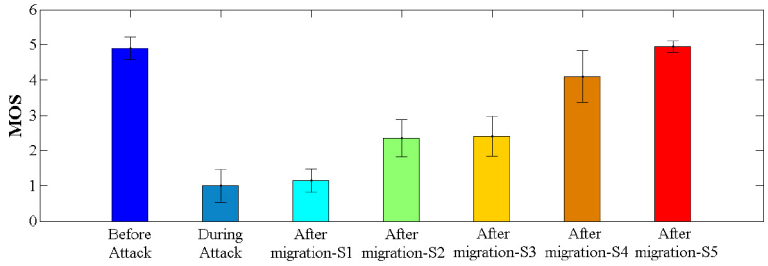
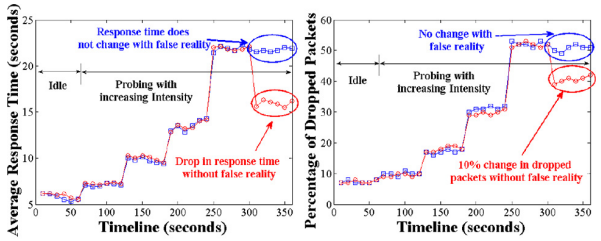


Fig. 11. Comparison of candidate proactive schemes using human subject QoE mean opinion scores.



(a) Average response time (b) Percentage of dropped packets

Fig. 12. Performance comparison of reactive migration strategies with-and-without false reality.

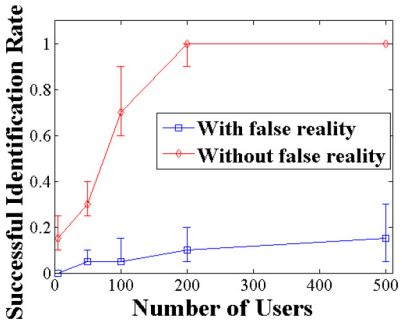


Fig. 13. Comparison of successful identification of dummy traffic by the attacker.

not only in terms of QoS (e.g., latency) overhead but also in QoE perspective, given our heuristic algorithms for service migrations that can be implemented relatively easily within a data center using tools such as AWS Connector (versus across many data centers) without disruption to the application user experience.

F. False Reality Establishment Results

In order to satisfy the necessary conditions for creating the false reality environment, the 'Attack Profiler' (from Section III-B) generates results similar to the one in Figure 4. These results help in determining the amount of dummy traffic that needs to be synthetically created to instantiate the pretense. In our false reality experiment, we use Figure 4 results to initiate false reality and compare the performance of our proposed 'LC-based reactive scheme with false reality' against our previous migration scheme without false reality [43]. The comparisons are made in terms of the chances for the attacker to distinguish between VMs with real and dummy traffic. In Figures 12(a), and 12(b), we compare the average response time and percentage of dropped packets during the idle and probing phases for the different attack intensity settings. We observe that - for the migration without false reality, the difference between response times and percentage of dropped packets before and after migration/redirection is sufficient enough (from Section V-C) for the attacker to detect migration/redirection scenarios. Such a detection suggests to the attacker that a high-value target has been found, and thus increases the chances of a future (more intensive) attack. Whereas, for our proposed scheme with false reality, the change in response time and dropped packets percentage after migration/redirection is negligible. The effectiveness of our false reality environment shown in Figures 12(a) and 12(b) are direct outcomes of the intelligent dummy traffic generation insights received from Figure 4 through the 'Attack Profiler' functionality.

Next, we demonstrate the overall effectiveness of our proposed false reality pretense in terms of the success rate of an attacker in identifying the presence of dummy traffic in Figure 13. The success rate is calculated by the total number of successful dummy identifications for the target application consumption by a small-to-large number of users. The results present an average of 20 reactive migration attempts involving different destination VMs with varied properties. Figure 13 shows that the false reality ensures considerably lower detection success in terms of identifying the dummy traffic patterns. We observe that with small number of users, the benefits of false reality is not pronounced enough as with small amount of traffic generated from a small number of users using the target application. Consequently, the attacker cannot really experience any perceivable difference in response times when the users are redirected. Thus, it is possible for our schemes with-and-without false reality to be virtually indistinguishable. Moreover, schemes without false reality perform well when number of users using the target application is very small. However, with large number of users, the attacker can easily identify the migration/redirection with considerable change in response times and dropped packets (as seen from Figures 12(a) and 12(b)) in the absence of false reality.

Finally in Figure 14, we characterize the benefits of the false reality pretense in terms of the 'lost opportunity cost' of a DDoS attack and compare it with the cost of implementation of the false reality pretense. We perform the comparison in terms of average CPU utilization in order to ascertain whether the creation of false reality is viable for the CSPs. We calculated the CPU utilization of the VM hosting the target application during: idle, probing (200 packets/sec), and flooding periods. We then compared them with the CPU utilization of the standby VM generating dummy traffic for 100, 300, and 500

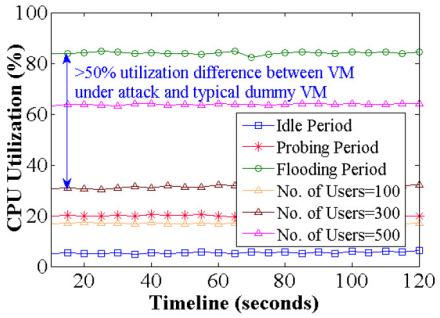


Fig. 14. Benefits and costs of false reality in terms of average CPU utilization.

users, respectively. From Figure 14, it is evident that the utilization of a VM under DDoS attack is more than 50% higher than an average dummy VM generating traffic for around 300 users, thus satisfying Equation (14) for cost effectiveness of the false reality pretense. Interestingly while generating high loads of dummy traffic, i.e., mimicking more than 1000 users' traffic, the dummy VM's CPU utilization might be as high as a VM under attack, thus making false reality less beneficial in those circumstances (at least in terms of CPU utilization cost). These results overall signify the importance of the false reality pretense and provide insights for the cost-effective dummy traffic generation design while creating an illusion of success for the attacker.

VII. Conclusion and Future Work

In this paper, we proposed an intelligent MTD based proactive and reactive VM migration scheme to protect cloud based applications from LOA attacks, such as DDoS. Our proposed scheme optimizes the frequency of migration in order to minimize wastage of network resources, yet limiting attack impact and damages. The scheme also computes the ideal migration location selection based on a market-driven scheme that considers key factors which include candidate VM's capacity, available network bandwidth, and VM reputation in terms of attack history. To the best of our knowledge, our work is one among the very first to propose proactive MTD based VM migration optimization schemes that consider a heterogeneous VM pool and perform trade-offs between the cost of migration upon attack detection. In addition to being effective when a CSP's internal network is infiltrated to launch DDoS attacks, our scheme provides benefits to CSPs in terms of enhanced protection using a false reality pretense to learn intelligent attack behavior and deter future attacks.

In the future, we plan to minimize the cloud management cost of MTD-based solutions to perform trade-offs between system obfuscation and resource management. We also plan to study the implementation of our schemes considering containerized application formats and policy co-ordination across multiple domains to block DDoS attacks closer to the attack source side.

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or Thomson Reuters.

References

[1] P. Hershey and C. Silio, Jr., "Procedure for detection of and response to distributed denial of service cyber attacks on complex enterprise systems," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, Mar. 2012, pp. 1–6.
[2] P. Stavroulakis and M. Stamp, *Handbook of Information and Communication Security*. Heidelberg, Germany: Springer-Verlag, 2010.
[3] A. Kartit, A. Saidi, F. Bezzazi, M. El Marakki, and A. Radi, "A new approach to intrusion detection system," *J. Theor. Appl. Inf. Technol.*, vol. 36, no. 2, pp. 284–289, Feb. 2012.
[4] *Extreme DDoS Defense (XD3)*. Accessed: Mar. 2020. [Online]. Available: http://www.darpa.mil/program/extreme-ddos-defense/
[5] I. Monga, E. Pouyoul, and C. Guok, "Software-defined networking for big-data science—Architectural models from campus to the WAN," in *Proc. High Perform. Comput. Netw. Storage Anal. (SCC)*, Nov. 2012, pp. 1629–1635.
[6] *OpenFlow Switch Specification*. Accessed: Mar. 2020. [Online]. Available: https://www.opennetworking.org
[7] M. Berman *et al.*, "GENI: A federated testbed for innovative network experiments," *Comput. Netw.*, vol. 61, pp. 5–23, Mar. 2014.
[8] *The Notorious Nine: Cloud Computing Top Threats in 2013*, Cloud Security Alliance, Seattle, WA, USA, 2013.
[9] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds," in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implement. (NSDI)*, vol. 2, 2005, pp. 287–300.
[10] A. Yaar, A. Perrig, and D. Song, "FIT: Fast Internet traceback," in *Proc. IEEE 24th Annu. Joint Conf. Comput. Commun. Soc. (INFOCOM)*, vol. 2, Mar. 2005, pp. 1395–1406.
[11] J. Idziorek, M. Tannian, and D. Jacobson, "Detecting fraudulent use of cloud resources," in *Proc. 3rd ACM Workshop Cloud Comput. Security Workshop (CCSW)*, 2011, pp. 61–72.
[12] J. Idziorek and M. Tannian, "Exploiting cloud utility models for profit and ruin," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2011, pp. 33–40.
[13] J. Kong, O. Aciiçmez, J.-P. Seifert, and H. Zhou, "Deconstructing new cache designs for thwarting software cache-based side channel attacks," in *Proc. 2nd ACM Workshop Comput. Security Archit.*, 2008, pp. 25–34.
[14] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *Proc. 34th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2007, pp. 494–505.
[15] J. Yuan and K. Mills, "Monitoring the macroscopic effect of DDoS flooding attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 2, no. 4, pp. 324–335, Oct.–Dec. 2005.
[16] Y. Kim, W. C. Lau, M. C. Chuah, and H. J. Chao, "PacketScore: A statistics-based packet filtering scheme against distributed denial-of-service attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 3, no. 2, pp. 141–155, Apr.–Jun. 2006.
[17] T. E. Carroll, M. B. Crouse, E. W. Fulp, and K. S. Berenhaut, "Analysis of network address shuffling as a moving target defense," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 701–706.
[18] H. Wang, Q. Jia, D. Fleck, W. Powell, F. Li, and A. Stavrou, "A moving target DDoS defense mechanism," *Comput. Commun.*, vol. 46, pp. 10–21, Jun. 2014.
[19] S. Venkatesan, M. Albanese, K. Amin, S. Jajodia, and M. Wright, "A moving target defense approach to mitigate DDoS attacks against proxy-based architectures," in *Proc. IEEE Int. Conf. Commun. Netw. Security (CNS)*, 2016, pp. 198–206.
[20] M. Nguyen, A. Pal, and S. Debroy, "Whack-a-mole: Software-defined networking driven multi-level DDoS defense for cloud environments," in *Proc. IEEE Int. Conf. Local Comput. Netw. (LCN)*, 2018, pp. 493–501.
[21] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell, "Catch me if you can: A cloud-enabled DDoS defense," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, 2014, pp. 264–275.
[22] R. Grandl, S. Kandula, S. Rao, A. Akella, and J. Kulkarni, "GRAPHENE: Packing and dependency-aware scheduling for data-parallel clusters," in *Proc. 12th USENIX Conf. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 81–97. [Online]. Available: http://dl.acm.org/citation.cfm?id=3026877.3026885
[23] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. ACM Special Interest Group Data Commun. (SIGCOMM)*, 2019, pp. 270–288.
[24] S. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *Proc. 24th USENIX Security Symp.*, 2015, pp. 817–832.

[25] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime DDoS defense using cots SDN switches via adaptive correlation analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 7, pp. 1838–1853, Jul. 2018.

[26] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "OpenFlow random host mutation: Transparent moving target defense using software defined networking," in *Proc. Workshop Hot Topics Softw. Defined Netw.*, 2012, pp. 127–132.

[27] P. Kampanakis, H. Perros, and T. Beyene, "SDN-based solutions for moving target defense network protection," in *Proc. IEEE 15th Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, 2014, pp. 1–6.

[28] R. Zhuang, S. Zhang, A. Bardas, S. DeLoach, X. Ou, and A. Singhal, "Investigating the application of moving target defenses to network security," in *Proc. Symp. Resilient Control Syst. (ISRCS)*, 2013, pp. 162–169.

[29] J. Xing, W. Wu, and A. Chen, "Architecting programmable data plane defenses into the network with FastFlex," in *Proc. ACM Wkshp Hot Topics Netw.*, 2019, pp. 161–169.

[30] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.

[31] S. Shetty, X. Yuchi, and M. Song, *Moving Target Defense for Distributed Systems* (Wireless Networks). Cham, Switzerland: Springer, 2016.

[32] A. Clark, K. Sun, and R. Poovendran, "Effectiveness of IP address randomization in decoy-based moving target defense," in *Proc. IEEE 52nd Annu. Conf. Decis. Control (CDC)*, Dec. 2013, pp. 678–685.

[33] L. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," in *Proc. DARPA Inf. Survivability Conf. Expo. (DISCEX)*, vol. 2, 2001, pp. 307–321.

[34] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing, "Automated generation and analysis of attack graphs," in *Proc. IEEE Symp. Security Privacy*, 2002, pp. 273–284.

[35] M. E. Kuhl, J. Kistner, K. Costantini, and M. Sudit, "Cyber attack modeling and simulation for network security analysis," in *Proc. 39th Conf. Winter Simulat. (WSC)*, 2007, pp. 1180–1188.

[36] S. Debroy, S. De, and M. Chatterjee, "Contention based multichannel MAC protocol for distributed cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 12, pp. 2749–2762, Dec. 2014.

[37] S. Yu and W. Zhou, "Entropy-based collaborative detection of DDoS attacks on community networks," in *Proc. 6th Annu. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2008, pp. 566–571.

[38] J. Mirkovic *et al.*, "Automating DDoS experimentation," in *Proc. USENIX Demo*, 2007, pp. 1–8.

[39] J. Pawlick and Q. Zhu, "Proactive defense against physical denial of service attacks using Poisson signaling games," in *Decision and Game Theory for Security*. Cham, Switzerland: Springer Int., 2017, pp. 336–356.

[40] D. P. Bertsekas and R. G. Gallager, *Data Networks*, Cambridge, MA, USA: MIT Press, 1992.

[41] T. H. Noor, Q. Z. Sheng, L. Yao, S. Dustdar, and A. H. H. Ngu, "CloudArmor: Supporting reputation-based trust management for cloud services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 367–380, Feb. 2016.

[42] *Really Simple Syndication*. Accessed: Mar. 2020. [Online]. Available: https://www.rss.com/

[43] S. Debroy, P. Calyam, M. Nguyen, A. Stage, and V. Georgiev, "Frequency-minimal moving target defense using software-defined networking," in *Proc. IEEE Int. Conf. Comput. Netw. Commun.*, Feb. 2016, pp. 1–6.

Saptarshi Debroy (Member, IEEE) received the B.Tech. degree from the West Bengal University of Technology, India, in 2006, the M.Tech. degree from Jadavpur University, India, in 2008, and the Ph.D. degree in computer engineering from the University of Central Florida in 2014. He is currently an Assistant Professor with the Department of Computer Science, City University of New York. His current research interests include cyber security, cloud computing, and wireless networks.

Prasad Calyam (Senior Member, IEEE) received the M.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering, Ohio State University in 2002 and 2007, respectively. He was a Research Director with the Ohio Supercomputer Center. He is currently an Associate Professor with the Department of Computer Science, University of Missouri–Columbia. His research interests include distributed and cloud computing, computer networking, and cyber security.

Minh Nguyen received the M.S. degree in computer engineering from the University of Missouri in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of New York. His current research interests include cyber security and cloud computing.

Roshan Lal Neupane (Student Member, IEEE) received the B.E. degree in computer science and engineering from Visvesvaraya Technological University, India. He is currently pursuing the M.S. degree in computer science with the University of Missouri–Columbia. His current research interests include cloud networking, IoT, and cybersecurity.

Bidyut Mukherjee (Student Member, IEEE) received the B.E. degree in computer technology from RTM Nagpur University, India, in 2015, and the M.S. degree in computer science from the University of Missouri–Columbia in 2017. His current research interests include cloud computing, computer networking, IoT, and cyber security.

Ajay Kumar Eeralla received the M.S. degree in computer science from Clarkson University in 2014, and the M.Tech. degree in systems analysis and computer applications from NITK, Surathkal, India, in 2011. He is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Missouri–Columbia. His current research interests include cyber security, hierarchical access control, cryptography, and formal methods.

Khaled Salah He was with the Department of Information and Computer Science, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. He is a Full Professor with the ECE Department, Khalifa University of Science, Technology and Research, UAE. He joined the ECE Department in September 2010. He has over 145 publications related to the areas of cloud computing and cybersecurity. He was a recipient of Khalifa University Outstanding Research Award 2014, the KFUPM University Excellence in Research Award 2009, and the KFUPM Best Research Project Award 2010.