Securing Cloud Infrastructure Against Co-Resident DoS Attacks Using Game Theoretic Defense Mechanisms

Harkeerat Singh Bedi
Department of Computer Science
University of Memphis
Memphis, TN 38152
1-423-987-4629
hsbedi@memphis.edu

Sajjan Shiva
Department of Computer Science
University of Memphis
Memphis, TN 38152
1-901-678-5465
sshiva@memphis.edu

ABSTRACT

Evolution in cloud services and infrastructure has been constantly reshaping the way we conduct business and provide services in our day to day lives. Tools and technologies created to improve such cloud services can also be used to impair them. By using generic tools like nmap, hping and wget, one can estimate the placement of virtual machines in a cloud infrastructure with a high likelihood. Moreover, such knowledge and tools can also be used by adversaries to further launch various kinds of attacks. In this paper we focus on one such specific kind of attack, namely a denial of service (DoS), where an attacker congests a bottleneck network channel shared among virtual machines (VMs) co-resident on the same physical node in the cloud infrastructure. We evaluate the behavior of this shared network channel using Click modular router on DETER testbed. We illustrate that game theoretic concepts can be used to model this attack as a two-player game and recommend strategies for defending against such attacks.

Categories and Subject Descriptors

C.2.0 [**Computer-communication Networks**]: General – *Data communications, Security and protection.*

General Terms

Design, Security

Keywords

Cloud computing infrastructure, denial of service (DoS), game theory, cyber security

1. INTRODUCTION

The impact of denial of service attacks in our present cyberspace and economy cannot be neglected. Recent events [1, 2, 3, and 4] indicate the ease with which these attacks can be organized and executed. With the help of social networking websites, user-friendly software like Low Orbit Ion Cannon (LOIC) can be rapidly and effortlessly shared to conduct denial of service attacks on target systems and networks. Using such tools in IRC mode (HiveMind), clients running LOIC can be controlled by a single attacker to conduct highly effective and efficient distributed denial of service attacks that can take down even some of the biggest cyber entities.

As the cloud service providers (CSPs) begin to offer cheaper technology and computing resources to the cyber community, the decision of large and small enterprises to move into the cloud becomes easier. As more of these entities move to the cloud, the utility of conducting such online attacks also increase.

Recent research by Ristenpart et al. [5] shows that by using generic network testing tools, an attacker can successfully identify a target VM on the cloud with a high probability and instantiate VMs co-resident to the target VM to conduct a variety of attacks. In this work, we focus on the possibility of the attacker conducting a denial of service by congesting a network queue shared by all the VMs. We emulate and evaluate the shared queue on DETER testbed and using game theory model this attack scenario as a two player game.

The weakness of traditional network security solutions is that they lack a quantitative decision framework. As game theory deals with problems in which multiple players with contradictory objectives compete with each other, it can provide a mathematical framework for modeling and analyzing network security problems. As an example, a system administrator (defender) and an attacker can be viewed as two competing players participating in a game. In addition, game theory has the capability of examining large possible scenarios before taking the best action; hence, it can considerably enhance the decision making process of the system administrator.

Cloud computing service providers like Amazon's EC2 (Elastic Compute Cloud), Microsoft's Azure and Rackspace's Mosso provide users with the ability to rent computational resources for hosting and offering their services efficiently and cost-effectively to their customers. Users can rent these computational resources in the form of virtual machines (VMs) which run on servers controlled by these providers. We illustrate a game theoretic security model which can be used for defending the virtual machines deployed by such service providers against various kinds of denial of service attacks. As a case study, our model is illustrated using Amazon's EC2.

Users interested in using Amazon's EC2 begin by creating an account and setting up their billing preferences. Users can then instantiate their required number of virtual machines for hosting their services and applications. Each of these VMs is called an *instance* and when instantiated, is deployed on a physical machine. These machines or servers have the ability to run various VMs simultaneously. EC2 uses Xen Hypervisor for virtual machine monitoring, which is the dominant virtual machine (Domain 0) per each physical node and is responsible for system resource allocation to other virtual machines running on that node. The physical machines deployed by EC2 for providing such cloud computing services are distributed geographically at various locations known as *regions*. Each of these regions further has *availability zones* which are intended to be isolated from each other in case of failure.

When requesting for cloud resources, users can specify the region and availability zone they would like their VMs to be instantiated in. They can also specify the instance type of the VMs they are interested in. EC2 currently offers several kinds of instances (e.g. small, large, extra-large, etc.) with varying pricing schemes ranging from approximately \$0.10 per hour to \$2.60 per hour [6]. These are varied based on the amount of resources (memory, CPU, etc.) they offer in a particular type of instance [7].

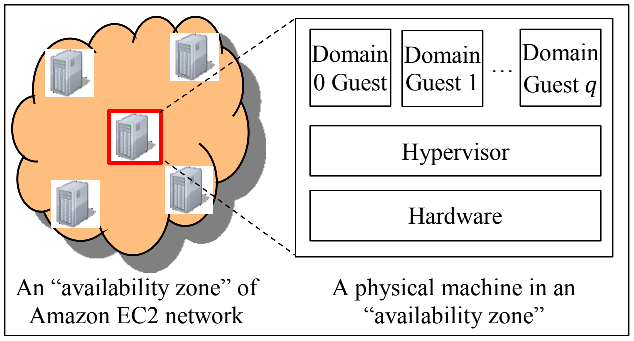
Domain 0 Guest
Domain Guest 1
…
Domain Guest q
Hypervisor
Hardware
An "availability zone" of Amazon EC2 network
A physical machine in an "availability zone"

Figure 1: Illustration of a region and a physical machine in Amazon EC2 network

In Figure 1, the cloud on left depicts an availability zone in Amazon EC2 network. The nodes in the cloud represent physical machines which are used to instantiate new VMs as requested by the users. The layered block diagram on right depicts one such physical machine which can be used to run VMs. The bottom layer represents the hardware resources of the physical machine which are shared among all the VMs. These resources include system hardware like main memory, secondary storage, CPU, network interface cards (NIC), etc. The layer above the Hardware layer represents the Hypervisor. The Hypervisor is a virtual machine monitor which provides the instantiated VMs with access to the physical system hardware and also provides isolation among co-resident VMs. Amazon's EC2 uses Xen hypervisors [8]. The blocks above the Hypervisor layer represent the various VMs running in the physical machine. The first block "Domain 0" represents a privileged guest VM which is used to manage other guests VMs (domain 1… domain q). The domain 0 VM manages the physical resource partitioning of the other guest VMs. In EC2, traffic for all VMs in a physical machine is routed through the Domain 0 VM [5].

In this work we focus on a particular kind of denial of service attack which can take place when malicious VMs co-resident with a victim VM use the shared network channel unfairly such that the victim VM is depleted of network resources. In this work we focus on Xen hypervisor which is used by Amazon's EC2 and the shared network channel here is the queue in the NIC installed in the physical machine.

2. Problem Formulation

Recent work by Ristenpart et al. [5] demonstrates how the internal cloud infrastructure of such service providers (they use Amazon's EC2) can be mapped and the location of a particular VM can be identified. They further show that new VMs can be instantiated until one is placed co-resident with the victim VM. They explore how such placement can be used to mount cross-VM side-channel attacks or cause denial of service attacks on the victim VM.

In this work we focus on the impact of such placement when it is performed by the attacker to launch denial of service attacks on a victim VM and propose a game model to defend against the same.

Figure 2 illustrates the network virtualization architecture as implemented by Xen hypervisor [14]. We use Amazon's EC2 along with this network virtualization architecture of Xen hypervisor to explain the potential denial of service attack and our proposed game model.

The Domain 0 VM uses the NIC Driver to interface directly with the network interface card (NIC) of the physical machine to transmit and receive data packets and driver control. The solid arrows in Figure 2 depict transfer of packet data and driver control between the Domain 0 and the NIC. Moreover, all traffic from other guests VMs (Domain Guest 1, Domain Guest 2, and Domain Guest 3) is also routed through Domain 0 (depicted by dashed arrows in Figure 2). All guest VMs commonly share the "transmit and receive queue" (T/R Queue) in the NIC during their data transmissions. This T/R Queue is susceptible to potential network congestion which can be exploited by denial of service attacks.

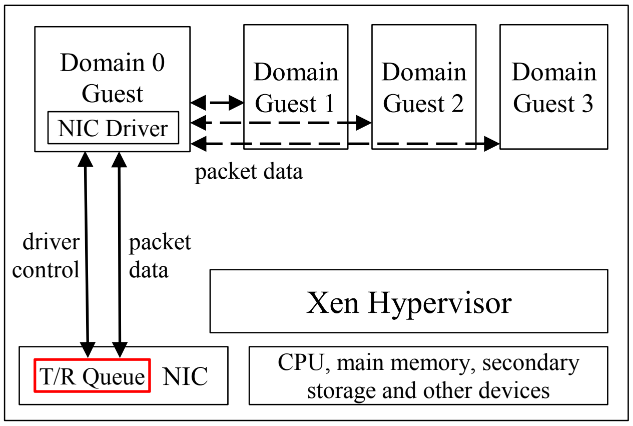

Figure 2: Network virtualization architecture of a physical machine running a Xen hypervisor

An attacker can perform DoS attacks on a victim VM by placing several VM instances as co-resident to the victim VM and then proceed to deplete the available shared network resources. The malicious VMs can send bogus traffic across such shared communication channels such that the victim VM does not get its fair share shared channel space. In our case, the shared network channel is the T/R Queue which is present in the NIC of the physical machine which is running on Xen hypervisor.

To obtain such VM placements which are co-resident to the victim VM, the attacker begins by enumerating a potential set of target victims. Then by using EC2's DNS service which provides means to map public IP addresses to private IP addresses, the attacker is able to infer which of these targets belong to a particular availability zone and instance type [5]. Once the availability zone and instance type of a target VM is identified. The attacker rents several VM instances from the cloud service provider (CSP) which can then be instantiated as required. A VM instance depending on the instance type can cost the attacker anywhere from approximately \$0.10 per hour (small instance: e.g. EC2's m1.small) to \$0.40 (larger instance: e.g. EC2's m1.large).

The attacker then repeatedly runs probe instances in the target zone and target type until he is successful in placing his VM as co-resident with the target VM. Co-placement can be verified by performing network based co-residence checks (e.g. matching

Domain 0 IP address, small packet round-trip times) or by performing cross-VM covert channel communication [5]. The more VMs the attacker can rent the more probes he can perform simultaneously and thus the faster he can place one of his VM's as co-resident with the target VM. Moreover by employing more malicious VMs, the attacker needs to send less bogus traffic per VM over the shared network channel to cause a DoS attack. This also makes it harder for the CSP to identify the attack. However, renting more VM's has a higher cost to the attacker. Thus there exists a tradeoff for the attacker such that employing more VMs makes his attack easier to implement and execute, however it also becomes more expensive.

Thus the attacker's action set includes, a) identifying the number of VMs to rent from the CSP to conduct the DoS attack and b) deciding the bitrate of bogus traffic to send from each rented VM.

The Domain 0 Guest (Dom0), on the other hand aims to handle such by throttling the traffic bitrate of VMs which use more than their fair share of the shared network channel. That is, if a VM tries to use more than their fair share of bandwidth, their traffic would be dropped. In this case, Dom0 uses a firewall that functions based on a threshold value which is used to guide its traffic dropping policies. If the bandwidth used by a VM crosses this threshold value, its traffic is dropped. The use of firewall also incurs a tradeoff. That is, if Dom0 uses a low threshold value then even legitimate VMs will have their traffic dropped. If Dom0 uses a high value, then several malicious VMs may not have their traffic dropped. This would eventually lead to degraded performance for the legitimate VMs.

Hence, the action set of the defender includes deciding the optimum threshold for the firewall, so that it can drop maximum bogus traffic from malicious VMs and at the same time not punish legitimate traffic.

2.1 Assumptions

Our proposed model is based on the following assumptions. The T/R Queue is shared among all guest VMs and is susceptible to congestion [14]. We assume that the amount of time required for packet processing by the NIC is a constant average for each incoming packet. Hence, as per Little's law, the queue usage (queue dynamic length) is directly proportional to the incoming traffic rate.

A single attacker controls all of the attacking VMs. We assume that the victim VM is a legitimate VM. Traffic from all VMs (legitimate and attacker) use UDP as their transport protocol. All traffic per VM is represented as one flow. It should be noted that even if this assumption is not true, i.e. traffic consists of both UDP and TCP flows, our proposed solution will still defend against DoS attacks caused by UDP traffic.

There is infinite bandwidth available on the channel between 1) the NIC Driver in Domain 0 and the T/R Queue in NIC and 2) the transmission channels between the Domain 0 and other guest VMs. Moreover, Domain 0 is able to process all of the incoming packets.

Dom0 monitors the shared network channel usage among the various VMs co-residing on a single physical machine. Our model would also work if this monitoring functionality was carried out externally by the CSP instead of Dom0. In this case, the game model would be executed by CSP instead of the Dom0 VM.

Dom0 has no knowledge of whether the traffic is coming from the attacker or a legitimate VM. Its belief on the legitimacy of the traffic decreases with the increase of its bitrate. A packet from traffic from a VM is dropped in either of the two situations: 1) the traffic does not pass the firewall rule, 2) there is congestion in the queue and it overflows.

We do not consider the case where an attacker might spoof the source address uniquely for each packet in a single flow. Moreover, spoofing can be avoided by using anti-spoofing methods such as ingress filtering [9] or link-layer security protocols [10, 11].

In the next section we determine the behavior of the shared T/R queue under congestion and verify the same using on DETER.

3. Modeling the shared T/R queue behavior

In this work, we assume that the bottleneck T/R queue which is shared among all the VMs co-residing on the same physical machine is generic in nature and employs the widely used drop-tail queue management algorithm. A queue implementing drop-tail mechanism accepts incoming packets as long as it has available space. Once the queue is full, it begins to drop packets at its tail until it has enough space to accept new incoming packets.

Queues implementing drop-tail do not identify individual flows and react similarly to each flow during congestion. That is, during congestion packets from each flow are dropped at similar rates.

3.1 Congestion analysis

For a given drop-tail queue q, we represent its capacity, which is the maximum packets it can hold before it gets full and begins to drop packets as q_{max}. Based on Little's law, we define the dynamic queue length q_l as: $q_l = d \cdot \lambda$

Here d represents the constant average delay which each incoming packet incurs during its processing in the router and λ represents the cumulative incoming packet arrival rate. That is, $\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_n$ and n is the total number of flows. The maximum rate of incoming packets the queue q can handle before it begins to drop them:

$$\lambda_{max} = \frac{q_{max}}{d}$$

Thus, in terms of bitrate, the maximum traffic the queue can handle before it begins to drop packets is $\partial = \lambda_{max} \cdot p_{size}$. Here p_{size} represents the size of each packet. We define the overall packet dropping rate during congestion φ as:

$$\varphi = 1 - \frac{\partial}{\lambda} \qquad \text{(i)}$$

Being drop-tail in nature, the queue q reacts similarly to each incoming flow and thus φ also represents the percentage of traffic dropped by each individual flow. Congestion occurs when $d \cdot \lambda > q_{max}$.

3.2 Emulation using DETER and Click:

We verify the above behavior of a queue which uses drop-tail algorithm for its management using the publicly available testbed DETER [30] and open source router software Click [31].

Figure 2 illustrates the process where co-residing VMs share a common T/R queue among them. To replicate their behaviour on DETER, we setup the following network topology as illustrated in Figure 3. Nodes Source 1, Source 2 and Source 3 emulate the co-resident VMs (Domain Guest 1, Domain Guest 2 and Domain Guest 3) which aim to share a common T/R queue to send and receive data. Switch SW emulates the Domain 0 Guest which forwards the traffic from Domain 1, 2, and 3 to the NIC which houses the T/R queue as shown in Figure 2. On DETER node Gateway emulates the network router which houses the T/R queue which is shared by all VMs and is prone to congestion. Node Sink emulates an external machine to which the co-resident VMs send and receive data.

These nodes on DETER are connected to each other as shown in Figure 3. Each link has a capacity of 100 Mbits/sec. Node Gateway is machine with dual 3Ghz Intel Xeon processor and 2 GB of RAM. To emulate the behavior of the drop-tail T/R queue which is prone to congestion in node Gateway, we implemented a custom router configuration using Click in kernel level.

We create a queue q with a total capacity $q_{max} = 30$ packets and use a delay of $d = 0.01$ seconds. We use the network testing tool Iperf [17] to send and receive traffic among nodes. In the above case $\lambda_{max} = 30/0.01 = 3000$. We send UDP traffic where each datagram (packet) is of the size $p_{size} = 0.01176$ Mbits. Thus the maximum bitrate queue q can handle is $\partial = 3000 * 0.01176 = 35.3$ Mbits/sec.

We run three sets of experiments (Set 1, Set 2 and Set 3) where nodes Source 1, Source 2 and Source 3 send UDP traffic with different bitrates to Sink. During the execution of each set, 30 instances of UDP data transfer was performed from the three Source nodes to Sink. Each instance was 240 seconds in duration. In each instance, the traffic bitrates represented in Table 1 were sent. That is, Source 1, Source 2 and Source 3 send UDP traffic at the rate of 10, 20 and 60 Mbits/sec respectively. The cumulative bitrate of traffic sent during each set was 90 Mbits/sec.

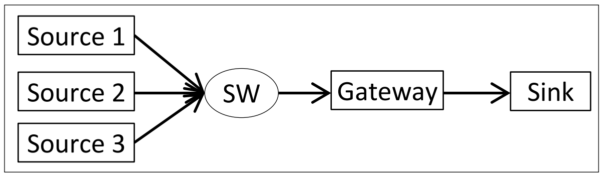

Figure 3: Network topology on DETER

Table 2 represents the average packet drop rate for each of the three sets for each of the source nodes. These drop rates for each set are an average of the 30 instances executed in each set.

Table 1: UDP Sender's Traffic Bitrate

	Source 1 (Mbits/sec)	Source 2 (Mbits/sec)	Source 3 (Mbits/sec)
Set 1	10	20	60
Set 2	10	25	55
Set 3	10	30	50

Table 2: Average Packet Drop Rate

	Source 1	Source 2	Source 3
Set 1	59.77	60.97	60.27
Set 2	59.77	60	60.77
Set 3	59.7	60.7	60.5

As observed in Table 1, the cumulative traffic sent during each set was at the rate 90 Mbits/sec. Thus, based on equation (i), the packet dropping rate during congestion $\varphi = 1 - \frac{35.3}{90} = 60.7\%$.

As illustrated in Table 2, we observed similar results by performing our emulation on DETER. We observed that the average packet drop rates for each flow during each set execution were statistically similar and our aforementioned analysis is validated. In the following sections we profile the behavior of legitimate VMs followed by a discussion of our proposed defense game model.

4. Modeling the attack scenario

This model extends our prior work in this domain [12, 13] where we defend target system against various kinds of denial of service attacks in different network systems. We model the present attack scenario as a two player zero-sum static game between the attacker which hosts malicious VMs and the defender which maintains the firewall settings for the T/R queue. We begin by profiling the behavior of a legitimate and malicious VMs followed by defining the game in the next section.

4.1 Behavior of legitimate VMs

We define legitimate VMs as the ones which share the common network channel (T/R Queue) fairly with other co-resident VMs. Fairness implies that each legitimate VM controls their traffic bandwidth such they do not exceed their share of the queue usage. We assume that the target VM is a legitimate VM and will not exceed its fair share of the T/R queue.

We consider that there are l legitimate VMs co-resident on a single physical machine and each share the T/R queue fairly with others. We model the traffic sending rate of each VM based on a Normal Distribution, i.e. $S_i : \mathcal{N}(s_l, \sigma_l^2)$, $i = 1,2,\ldots,l$ where S_i represents the sending rate of the i^{th} legitimate VM, s_l is the mean value of a legitimate VM's sending rate, and σ_l is the standard deviation.

Hence, the total traffic bandwidth when no attack is taking place is $T^{na} = S_1 + S_2 \ldots + S_l$. By basic laws of probability, we have $T^{na} : \mathcal{N}(l \cdot s_l, l \cdot \sigma_l^2)$. We assume that the total bandwidth (∂) the T/R queue can handle before if begins to overflow and drop packets is greater than the cumulative traffic transmitted by all the legitimate VMs (T^{na}). That is, $\partial > T^{na}$ with a high probability.

4.2 Behavior of malicious VMs

We define malicious VMs as the ones which co-reside with other legitimate VMs on a physical machine and do not share the T/R queue with other VMs fairly. That is, they use more than their fair share of queue in order to induce congestion in the T/R queue and make it drop traffic from the legitimate VMs. In this model, we consider that a single attacker controls all the attacking VM, whose count is denoted by m. We define the attack as a distributed denial of service (DDoS) attack when the attacker employs more than one attacking VM ($m > 1$) and a denial of service (DoS) attack when the attacker employs only one attacking VM ($m = 1$).

5. A game between attacker and defender

Based on the behavior of the legitimate and malicious VMs defined in the previous section, we design the game model between the attacker and defender with the following actions sets. The attacker's action set includes a) deciding the number of malicious VMs to use to conduct the attack and b) deciding the bitrate of traffic through each of them to congest the T/R queue. The defender's action set includes a deciding the firewall threshold value to prevent the T/R queue from congestion from the malicious VMs.

We assume that attacker sends equal amount of traffic from each employed malicious VM and denote it as s_A. Thus, during an attack the total traffic sent across the T/R queue $T^a = (S_1 +$

$S_2 + \ldots + S_l) + m \cdot s_A$. When the total traffic rate exceeds the capacity of the T/R queue ($T^a > \partial$), congestion takes place and packets begin to drop from the queue leading to denial of service. Now, we define the components which attribute to the payoff of the attacker and defender.

5.1 Game payoffs for attacker and defender

The defender, which is the Dom0 VM in our attack scenario, uses a firewall to drop traffic from malicious VMs and at the same time allow traffic from the legitimate VMs. When no defense mechanism is in place, that is, no firewall is installed by Dom0; all traffic is allowed to pass through the T/R queue unrestricted. In this case if $T^a > \partial$, only a percentage of traffic from each VM is allowed by the T/R queue. We denote this percentage by ω and it is the same for each VM.

Thus, if a VM i sends s_i bitrate of traffic, only $s_i \cdot \omega$ bitrate will pass the queue during congestion. We assume that the queue is shared in a fair and equitable manner and thus $\omega = \frac{\partial}{T^a}$. That is, equal percentage of traffic is dropped from each VM.

In real world scenarios, it is well observed that definition of traffic from a node (or a flow in particular) as being alive or dead is largely based on the application and transport protocols used. It is common to consider a flow (or traffic from a node) as dead if it is below a certain threshold. In this work we define the minimum threshold for traffic from a legitimate VM to be considered as alive by γ. If a CSP does not require such a minimum threshold, this model can be used by setting γ to 0.

Let l_a be the number of legitimate VMs whose bitrate is greater than γ and hence are considered alive. We get $l_a = l \cdot P[S_i > \frac{\gamma}{\omega}]$, where $P[S_i > \frac{\gamma}{\omega}]$ gives the probability that the bitrate of traffic from a legitimate VM S_i is greater than the minimum threshold γ.

The payoff of the attacker depends on how well he is able to conduct the attack by maximizing the dropping of traffic from legitimate VMs at the same time minimizing his costs of renting VMs and executing the attack. Similarly, the payoff for the defender (Dom0) which controls the firewall is based on how well he is able to protect the legitimate VMs by minimizing their traffic drops, at the same time filtering out maximum malicious traffic. Thus we define the following components which attribute to the overall payoff of the attacker and defender.

Payoff component 1: Ratio of the average queue consumed by the attacker to the total queue in use:

$$p_{c_1} = \frac{m \cdot \omega \cdot s_A}{\partial} = \frac{m \cdot s_A}{l \cdot s_l + m \cdot s_A} \tag{1}$$

The higher this ratio, the more queue is being used by the attacker, thus the value of this component is directly proportional to the attacker's payoff.

Payoff component 2: Ratio of lost legitimate VMs to the total number of legitimate VMs: $p_{c_2} = \frac{l - l_a}{l} = P\left[S_i < \frac{\gamma}{\omega}\right]$

$$= P\left[S_i < \frac{\gamma(l \cdot s_l + m \cdot s_A)}{\partial}\right] \tag{2}$$

As discussed above, we consider a legitimate VM as lost when its traffic bitrate is below the minimum bitrate threshold γ. As this ratio gets higher, more legitimate VMs experience queue congestion and drops in their traffic. Thus, the value of this component is directly proportional to payoff of the attacker.

Payoff component 3: Attacker's cost of renting VMs:

$$p_{c_3} = m \tag{3}$$

This component quantifies the number of VMs the attacker rents and initializes from the CSP to perform the denial of service attack. The value of this component is inversely proportional to the payoff of the attacker and hence part of his objective includes keeping the value of this component as low as possible.

Thus the overall payoff of the attacker is defined as:

$$P_a = p_{c_1} \cdot w_{c_1} + p_{c_2} \cdot w_{c_2} - p_{c_3} \cdot w_{c_3} \tag{4}$$

And, being a zero-sum game, the total payoff of the defender is defined as: $P_d = -p_{c_1} \cdot w_{c_1} - p_{c_2} \cdot w_{c_2} + p_{c_3} \cdot w_{c_3}$ (5)

Here, the variables $w_{c_1}, w_{c_2}, w_{c_3}$ represent the weights of each component which can be adjusted based on the discretion of the CSP. For example, the variable w_{c_3} can be set as \$0.92 if the attacker is renting extra-large standard on-demand instances from US East (Virginia) region [6]. Furthermore, by using different values for weights in equations 4 and 5, the CSP can also address attacker scenarios which are not zero-sum in nature.

5.2 Modeling the game inspired defense firewall

Equations 4 and 5 represent the overall payoff of the attacker and defender when no defense mechanism is employed by the defender. That is, Dom0 does not use a firewall to throttle malicious traffic which use more than their fair share of the queue.

In this section we model the behavior of the firewall which the defender will use to throttle malicious traffic and at the same time prevent dropping traffic from legitimate VMs. The traffic dropping behavior of the firewall is modeled based on a sigmoid function $F(x)$ as follows: $F(x) = \frac{1}{\left(1 + e^{-\beta\left(\frac{x-M}{\partial}\right)}\right)}$

This function drops traffic from a VM based on the T/R queue usage which is directly proportional to the traffic's incoming rate. Here M represents the bitrate for which the drop rate is 0.5 and the variable β represents the scaling factor. Incoming traffic rate is represented by x and queue capacity by ∂. The firewall drops the packets from a flow with rate x with a probability $F(x)$. It is worth pointing out that some traffic from a legitimate VM might also get dropped at the firewall. We consider that the Dom0 controls the value of M, which is its defense action.

Recall that s_l represents the expected traffic rate of a legitimate VM. Let the average rate of legitimate VM traffic passing through the firewall be s_l'. We have $s_l' = s_l \cdot \left(1 - F(s_l)\right)$. On the other hand, the average bitrate of traffic from attacking VMs passing through the firewall is $s_A' = s_A \cdot (1 - F(s_A))$. If we replace s_A by s_A' and s_l by s_l' in Equations (1) and (2), the new payoff components we obtain are:

Payoff component 1: Ratio of the average queue consumed by the attacker to the total queue in use: $p_{c_1'} = \frac{m \cdot s_A'}{l \cdot s_l' + m \cdot s_A'}$ (6)

Payoff component 2: Ratio of lost legitimate VMs to the total number of legitimate VMs: $p_{c_2'} = P\left[S_i < \frac{\gamma'}{(1 - F(s_l))}\right]$ (7)

Where, $\gamma' = \begin{cases} \frac{\gamma(l \cdot s_l' + m \cdot s_A')}{\partial}, & in\ congestion: \quad (l \cdot s_l' + m \cdot s_A') > \partial \\ \gamma, & in\ no\ congestion: (l \cdot s_l' + m \cdot s_A') \leq \partial \end{cases}$

Payoff component 3: Attacker's cost of renting VMs:

$$v_{c'} = m \qquad (8)$$

Thus the attacker and defender's cumulative payoffs are computed by using these new components in equations 4 and 5. We use the notion of Nash equilibrium to determine the best strategy profile of these two players. Each player has the goal to maximize his/her payoff. The attacker needs to choose optimal values for m and s_A, and the defender needs to choose the best value for M in the sigmoid function to be used by the firewall. The Nash equilibrium of this game is defined to be a pair of strategies (s_A^*, m^*, M^*), which simultaneously satisfy the following two relations:

$$V^a_{(s_A^*, m^*, M^*)} \geq V^a_{(s_A, m, M^*)} \forall s_A, m \qquad (9), \text{and}$$

$$V^d_{(s_A^*, m^*, M^*)} \geq V^d_{(s_A^*, m^*, M)} \forall M \qquad (10)$$

Thus by using the above NE equations (9 and 10), Dom0 can optimally decide the threshold values (M) for its firewall such that it can drop maximum traffic from attacker's malicious VMs while protecting the traffic from the legitimate VMs and thus defending against such denial of service attacks.

6. Related Work

Significant research has been carried out in the domain of network congestion and denial of service attacks [15]. Lau et al. [16] experimented with various queuing algorithms to determine which queuing method in the target router could provide better management of the bandwidth during a DDoS attack. Andersen [18] proposed a proactive protection against DDoS attacks, by imposing overhead on all transactions to actively prevent attacks from reaching the server. Their architecture generalizes the Secure Overlay Services (SOS) to choose a particular overlay routing. The set of overlay nodes are used to distinguish legitimate traffic from the attack traffic. Yaar et al. [19] proposed a flow based mitigation filter for DDoS flooding attacks called Stateless Internet Flow Filter (SIFF). This approach uses a per-flow state, where the flows are classified into two categories privileged flows, and unprivileged flows with the goal of protecting privileged packets from unprivileged packet flows. NetFence [20] uses a congestion feedback mechanism to enable robust congestion policing inside the network. The DoS victims can use the secure congestion policing feedback as capability tokens to suppress unwanted traffic and recover from attacks.

Cloud computing infrastructure being the new paradigm shift is corporate and personal computing has significant room for improvement. Wang et al. [21] in their recent work present a measurement study to characterize the impact of virtualization on the networking performance of the Amazon Elastic Cloud Computing (EC2) data center. Their results show that virtualization can cause significant throughput instability and abnormal delay variations even when the data center is only lightly used. Hao, et al. [22] propose a new data center architecture where users are allows to share physical hardware resources, but network resources are isolated and shared in a controlled manner similar to that of enterprise networks. Ristenpart et al. [5] explain in depth the potential of a target VM running in a cloud data center to be identified geographically by an attacker and its vulnerability to various kinds of attacks.

Recently, researchers have started exploring the applicability of game theory to model network security problems as multiplayer games [23, 24, 25, 26, 27, 28]. Xu et al. [29] proposed a game-theoretic model to defend a web service under DoS attack. They use several metrics to measure the performance of their system. Wu et al. [12] perform similar research where they primarily focus on DoS/DDoS attacks launched using UDP based traffic. Bedi et al. [13] focuses on mitigating DoS and DDoS attacks for TCP-friendly flows using a game theoretic approach.

7. Conclusion

In this paper, we demonstrated the potential of denial of service attacks caused due to congestion of a shared router queue in the physical machines of a cloud service provider. These attacks are made possible by instantiating malicious VMs and making them co-resident with the victim VM on the physical machine. We emulate and evaluate the behavior of this shared router queue the Click modular router on DETER testbed. We model this attack scenario as a two player game and design a model to determine the best strategies for the defender to defend against such attacks.

We are currently verifying the effectiveness of our proposed game model by performing extensive mathematical simulation and emulation. We perform simulation using MATLAB where consider a sub-set of the attacker's and defender's possible action set. For example, we consider the following scenario. The target physical machine on Amazon's EC2 (prone to attack) can run upto 10 VMs at any given time. The bitrate capacity of the transmit and receive (T/R) queue this physical machine is 1 Gb/s. The attacker has the resources to rent up to 10 VMs from Amazon's EC2. The attacker can vary the traffic bitrate of each of its rented VM between 1Mb/s to 100Mb/s. The defender can vary the firewall threshold between 1 Mb/s to 1000 Mb/s.

Based on the set of payoffs obtained for each player, we then identity the Nash equilibrium whose co-ordinates denote the best strategy for the defender (optimum firewall threshold bitrate) to maximize the dropping of bogus traffic from malicious VMs. Hence by performing the above kind of simulation, we aim to illustrate that by using the Nash equilibrium strategy obtained by our game model, the defender is able to obtain a better payoff when compared to the defender choosing any other defense strategy.

We are also working on verifying the applicability of our proposed defense solution by performing emulations using DETER testbeds and Click modular router. We create a topology where each of the various nodes behave as VM instances placed under one physical machine. These nodes share a T/R queue which is emulated by another node which acts as a gateway to the Internet. We run Click modular router on this gateway node in kernel level to monitor the bitrates of traffic coming from the other nodes. Our implementation of Click also includes the capability to drop traffic based on the firewall threshold variable defined in our model.

We aim to study the potential of the applicability of our proposed game defense solution on other cloud service infrastructure like Microsoft's Azure and Rackspace's Mozzo, as we only consider the effect of generic user traffic on such machines to provide immunity against congestion based DoS attacks.

8. REFERENCES

[1] Anonymous activists target Tunisian government sites, http://www.bbc.co.uk/news/technology-12110892, Retrieved on February 12, 2011.

[2] Hackers Shut Down Government Sites, http://www.nytimes.com/2011/02/03/world/middleeast/03hackers.html?_r=2, Retrieved on February 12, 2011.

[3] WikiLeaks Reports Attack on Its Web Site. Retrieved on December 3, 2010, http://thelede.blogs.nytimes.com/2010/11/28/wikileaks-reports-attack-on-its-web-site/

[4] 'Operation Payback' Attacks Target MasterCard and PayPal Sites to Avenge WikiLeaks, http://thelede.blogs.nytimes.com/2010/12/08/operation-payback-targets-mastercard-and-paypal-sites-to-avenge-wikileaks/, Retrieved on February 12, 2011.

[5] Ristenpart, T. and Tromer, E. and Shacham, H. and Savage, S., "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," 16th ACM CCS, pp. 199-212, 2009.

[6] Amazon EC2 Pricing, http://aws.amazon.com/ec2/pricing/, retrieved on February 17, 2012.

[7] Amazon EC2 Instance Types, http://aws.amazon.com/ec2/instance-types/, retrieved on February 17, 2012.

[8] Barham, P. and Dragovic, B. and Fraser, K. and Hand, S. and Harris, T. and Ho, A. and Neugebauer, R. and Pratt, I. and Warfield, A., Xen and the art of virtualization, ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 164-177, 2003.

[9] Ferguson, P. and Senie, D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, 2000.

[10] IEEE Standard 802.1X. http://www.ieee802.org/1/pages/802.1x.html, 2001.

[11] Liu, X. and Yang, X. and Lu, Y., "To Filter or to Authorize: Network-Layer DoS Defense Against Multimillion-node Botnets," in ACM SIGCOMM 2008, Seattle, Aug.2008.

[12] Wu, Q. and Shiva, S. and Roy, S. and Ellis, C. and Datla, V., On Modeling and Simulation of Game Theory-based Defense Mechanisms against DoS and DDoS Attacks. 43rd Annual Simulation Symposium, April 11-15, 2010.

[13] Bedi, H., Roy, S., Shiva, S., Game Theory-based Defense Mechanisms against DDoS Attacks on TCP/TCP-friendly Flows. IEEE Symposium on Computational Intelligence in Cyber Security (CICS), part of (SSCI). Paris, France, 2011.

[14] Rixner, S., Network Virtualization: Breaking the Performance Barrier, ACM Queue, vol. 6, no. 1, Jan. 2008.

[15] Mirkovic, J. and Reiher, P., A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, vol. 34, no. 2, pp.39–53, 2004.

[16] Lau, F. and Rubin, S.H. and Smith, M.H. and Trajkovic, L., Distributed denial of service attacks, In IEEE International Conference on Systems, Man, and Cybernetics, vol. 3, pp. 2275-2280, 2000.

[17] Tirumala, A. and Qin, F. and Dugan, J. and Ferguson, J. and Gibbs, K. Iperf: The TCP/UDP bandwidth measurement tool. [Online]. Available: http://sourceforge.net/projects/iperf

[18] Andersen, D., Mayday: Distributed filtering for internet services, in Proc. of the 4th Usenix Symposium on Internet Technologies and Systems, vol.4, 2003.

[19] Yaar, A. and Perrig, A. and Song, D., Siff: A stateless internet flow filter to mitigate DDoS flooding attacks, in proc. of IEEE Symposium on Security and Privacy, pp. 130–143, 2004.

[20] Liu, X. and Yang, X. and Xia, Y. NetFence: preventing internet denial of service from inside out. ACM SIGCOMM CCR. Vol. 40. Pages: 255-266. 2010.

[21] Wang, G. and Ng, T.S.E., The impact of virtualization on network performance of amazon EC2 data center, INFOCOM, 2010.

[22] Hao, F. and Lakshman, TV and Mukherjee, S. and Song, H., Secure Cloud Computing with a Visualized Network Infrastructure, in proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 2010.

[23] Roy, S. and Ellis, C. and Shiva, S. and Dasgupta, D. and Shandilya, V. and Wu, Q., A survey of game theory as applied to network security, 43rd Hawaii International Conference on System Sciences, 2010.

[24] Charilas, D.E. and Panagopoulos, A.D. A survey on game theory applications in wireless networks. Computer Networks, 2010.

[25] Liu, P. and Zang, W. and Yu, M. Incentive-based modeling and inference of attacker intent, objectives, and strategies. ACM TISSEC, vol. 8(1), pp. 78-118, 2005.

[26] Lye, K. and Wing, J.M. Game strategies in network security. International Journal of Information Security, vol. 4 (1), pp. 71-86, 2005.

[27] Manshaei, M.H. and Zhu, Q. and Alpcan, T. and Basar, T. and Hubaux, J.P. Game Theory Meets Network Security and Privacy. ACM Transactions on Computational Logic, 2010.

[28] Shen, S. and Yue, G. and Cao, Q. and Yu, F. A Survey of Game Theory in Wireless Sensor Networks Security. Journal of Networks, vol. 6, no. 3, pp. 521-532, 2011.

[29] Xu, J. and Lee, W., Sustaining availability of web services under distributed denial of service attacks, IEEE Transactions on Computers, vol. 52, no. 2, pp. 195–208, 2003.

[30] Benzel, T. and Braden, R. and Kim, D. and Neuman, C. and Joseph, A. and Sklower, K. and Ostrenga, R. and Schwab, S. Experience with DETER: A testbed for security research. 2nd International Conference on TRIDENTCOM, 2006.

[31] Kohler, E. and Morris, R. and Chen, B. and Jannotti, J. and Kaashoek, M.F., The Click modular router, ACM TOCS, vol. 28, no. 3, pp. 263-297, 2000.