Host-based DoS Attacks and Defense in the Cloud

Tianwei Zhang
Princeton University
tianweiz@princeton.edu

Ruby B. Lee
Princeton University
rblee@princeton.edu

ABSTRACT

We explore *host-based* DoS attacks, which exploit the shared computing resources in a multi-tenant cloud server to compromise the server's resource availability. We first present a set of attack techniques targeting different types of resources. We show such attacks can significantly affect the performance of co-located VMs, as well as the cloud provider's management services. Then we propose an attack strategy to compromise the availability of the entire datacenter. We show how power-aware optimization techniques can help the attacker achieve his goal faster, with low cost.

We design an effective general-purpose method to defeat memory, network and disk DoS attacks. We use a statistical method to detect changes in the usage of different resources. Once an attack happens, we use resource throttling techniques to identify and thwart the malicious VMs. Our evaluation shows that this defense method can effectively defeat these DoS attacks with negligible performance overhead. We alert the computer architecture community to these catastrophic attacks on the availability of cloud computing resources, to encourage building in better defenses at both the hardware and software levels.

1. INTRODUCTION

The Infrastructure-as-a-Service (IaaS) cloud model provides elastic and cheap resources to customers in the form of virtual machines (VMs). The financial benefits, flexibility and operational stability greatly motivate enterprises to move their services and applications to the cloud. However, malicious customers can also abuse these cheap cloud services to conduct attacks. According to a report from Cloud Security Alliance in 2016 [19], abuse and nefarious use of cloud services have become top threats in cloud computing.

Adversaries can illegally obtain a large number of VMs in a cloud system for cybercrime, at low or no cost in various ways [19]: poorly secured VM deployments or authentication schemes enable adversaries to easily intrude into customers' VMs and take full control of them; public cloud vendors usually provide free cloud service trials that adversaries can use to acquire an arbitrary number of free VMs; adversaries can also procure VM instances using stolen credit cards. Once the adversaries get enough VMs, they can deploy these VMs as botnets to attack other customers, organizations or even the cloud providers. Typical examples of such cloud service misuse include Distributed Denial-of-Service (DDoS) attacks, large-scale email spam and phishing, brute-force password guessing, port scanning, etc. [15]

Traditional *network-based* DDoS attacks compromise the service availability of a victim machine by flooding the machine with superfluous network requests from remote botnets. While these attacks and defenses have been well studied, this paper explores the newer and less studied area of *host-based* DoS attacks. These attacks leverage a basic feature in clouds, multi-tenancy, to conduct availability attacks. Specifically, cloud providers consolidate different VMs on the same cloud server to maximize resource utilization. Different VMs on the same server are logically isolated, but still share the same underlying hardware resources. So a malicious customer can intentionally craft his VMs to abuse the shared resources and affect the availability of different resources of co-located VMs and the host servers.

We focus on how DoS attacks on different resources can severely reduce availability of the entire server including the cloud management software, and even of the entire datacenter, rather than impact just an application or a VM. Also, unlike previous DoS defenses which focus on one specific resource, we show a general solution that works for DoS attacks on different resources.

We show the severity and effectiveness of such host-based DoS attacks at the server level and the datacenter level. At the server level, we show how an attacker can use one VM instance to degrade the performance of all co-located VMs and the cloud provider, by abusing the different layers of shared computing resources, e.g., locking the memory system, or saturating the network and I/O devices.

At the datacenter level, the attacker can use just a few VMs to attack as many cloud servers as possible. To reduce attack cost and increase efficiency, the attacker can shut down redundant attack VMs so that each server has only one malicious VM. This strategy allows the attacker to cover more cloud servers using fewer VMs. Our evaluation also shows that the attacker can compromise a datacenter at lower cost when the cloud provider employs more power-efficient scheduling policies.

To defend against these host-based DoS attacks, we propose a general-purpose method to detect and mitigate these

HASP '17, June 25, 2017, Toronto, ON, Canada

DOI: http://dx.doi.org/10.1145/3092627.3092630

attacks. We design a set of TESTING PROGRAMS to monitor the server's resource usage. When the server is under a host-based DoS attack, we use resource throttling techniques to discover the malicious VMs. Then we can suspend or shut down the VMs to protect the server's availability. This can mitigate the attacks with low performance overhead.

In summary, we make the following contributions:

- Showing DoS attacks on different resources that can cause availability degradation of entire host servers.
- Demonstration of the severity of these attacks.
- An attack strategy to compromise the availability of the entire datacenter, and showing that power-aware scheduling policies make the attacks worse.
- A general-purpose method for defeating malicious VMs using statistical tests and resource throttling.

2. THREAT MODEL AND ASSUMPTIONS

The target of the attacks are the cloud servers and the datacenter. This datacenter deploys IaaS cloud services to the public where customers lease VMs with desired computing resources. The cloud provider allocates VMs to cloud servers in the datacenter. Each server can host multiple VMs from different customers, logically isolated by the hypervisor.

The adversary in our consideration can be an individual hacker, a botnet originator, a competing cloud provider or an organization for committing cybercrime and cyberwarfare. The adversary can launch VMs in the target datacenter. But he cannot select the host servers for his VMs. He is able to take full control of his own VMs, but he does not have root privileges to control the hypervisors or other VMs.

The adversary can illegally launch and control multiple VMs in the datacenter without financial costs via free cloud service trials or credit card fraud. He can also pay to legally rent the VMs. Then there is a cost related to launch the attack. However, the severe damage outweighs the cost: the attacker can just use one tiny VM instance at the cost of several cents per hour to destroy an entire server which may host dozens of large VMs. This amplification effect helps the adversary to launch DoS attacks on the entire datacenter.

3. SERVER-WIDE DOS ATTACKS

We show how an attacker can launch one VM in a cloud server, and craft this VM to degrade the performance of the host server. We first describe attack techniques (Sec. 3.1), then evaluate their impact on co-located VMs and cloud management on the host server (Sec. 3.2 and Sec. 3.3).

3.1 Attack Techniques

A malicious VM can abuse the underlying resources to affect the performance of other VMs that also use these resources. We discuss some known attacks on the memory, network and disk, shared by all the VMs on the server, and show how they can be used to degrade the availability of the entire host server, and even the entire datacenter (Sec. 4).

Memory DoS attacks. While there are many ways to cause a DoS attack on memory availability [23], we describe one of the worst attacks below. Intel processors provide locked atomic operations for managing shared data structures between multi-processors [1]. Before Intel Pentium (P5) processors, the locked atomic operations always generate LOCK signals on the internal buses to achieve operation atomicity. For processor families after P6, the bus lock is transformed into a cache lock: the cache line is locked instead of the bus and the cache coherency mechanism is used to ensure operation atomicity. However, when a program issues locked atomic accesses to unaligned memory blocks, the processor has to fetch two adjacent cache lines and a bus LOCK signal is asserted to prevent other processors from modifying these cache lines. So the adversary VM can keep issuing this "exotic" memory access to lock down all the internal memory buses in the host server to degrade the server's performance [23].

server $\mathbb{A}$	Dell R720: one eight-core, 2.90GHz Intel Xeon E5-2690 processor
server $\mathbb{B}$	Dell R720: two six-core, 2.90GHz Intel Xeon E5-2667 processors
server $\mathbb{C}_1$	Dell R210II: one quad-core, 3.30GHZ Intel Xeon E3-1230v2 processor
server $\mathbb{C}_2$	
server $\mathbb{C}_3$	
Host OS	Ubuntu 14.04 Linux OS with 3.13 kernel, running KVM hypervisor
Guest OS	Ubuntu 14.04 Linux OS with 3.13 kernel

Table 1: Testbed configurations in our experiments

Network DoS attacks. In a cloud server, the hypervisor virtualizes the network devices for VMs. When a VM attempts to send a network packet, the device driver inside the VM passes the packet to the emulated network device in the hypervisor. Then the hypervisor forwards the packet to the physical network device. When a network packet attempts to reach the VM, it will also be routed through the host domain and handled by the physical network device, and then passed to the VM by the hypervisor. The physical device is shared by the host domain and guest VMs. To compromise the target server's network availability, the adversary can flood his own VM with a large number of network packets [7], which can cause congestion in the physical device, as well as deplete the hypervisor's capability to emulate and handle network accesses. This can significantly degrade the host domain and other VM's network performance.

Disk DoS attacks. Disk storage is another type of I/O resource that an attacker can exploit to conduct DoS attacks. Different processes/VMs on the server can issue disk accesses at the same time. To cause disk I/O contention, the malicious VM can keep generating a large volume of accesses to its own disk system [22]. This will cause disk scheduling congestion, thus delaying disk accesses from other VMs.

Another interesting observation is that the disk attack can also cause network contention. Public clouds usually adopt Network Attached Storage (NAS) to manage the VM file system. Cloud providers set up a few separate NAS servers to store the disk system of VMs from other computing servers, and these VMs access their disk system via the network. When the disk attack tries to generate disk contention, it also causes network resource congestion between the computing server and the network storage server.

3.2 Attacking Guest VMs

We evaluate the attack effects on the co-located guest VMs. We exploit three servers from Table 1: servers $\mathbb{A}$ and $\mathbb{B}$ are deployed as the target cloud servers; server $\mathbb{C}_1$ is configured as the disk server for the network-attached storage configuration. On top of the hardware components, a KVM hypervisor is operated to support virtualized guest VMs. The host OS and the guest OS are shown in Table 1.

Memory contention. We launch a malicious VM in the target server, which keeps generating LOCK signals on the internal buses by requesting atomic unaligned memory accesses. On the same server, we launch a set of benign VMs, each of which randomly runs SPEC2006 or PARSEC bench-

marks. We use Hardware Performance Counters to measure the overall instructions per second of the entire server as the performance metric. Figure 1 shows the performance of the server under different CPU utilization, with and without memory DoS attack. We observe that the memory DoS attack can reduce the server's overall IPC by up to 84%.

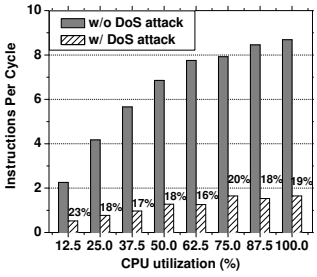


Figure 1: Memory DoS attack.

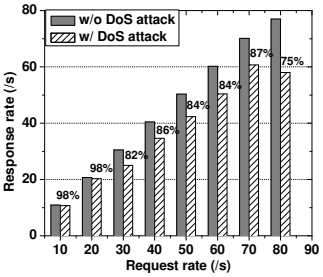
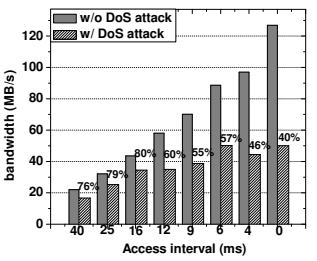


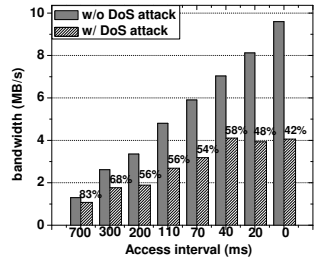
Figure 2: Network DoS attack.

Network contention. We launch a malicious VM on the target server. Then we use a client machine outside the cloud system to keep sending UDP packets to the malicious VM, thus saturating the host server's network resource. We launch a benign VM hosting apache service on the server, and use the httperf [5] benchmark to test its network performance. Figure 2 shows the throughput of the apache server under different request rates. We observe that the network DoS attack can reduce the server's throughout by up to 25%.

Disk contention. We consider two cases: the first one is the local disk system. The second one is the Network File System (NFS) [17], where a network storage server is configured to store the VMs' disks. For each case, we launch one malicious VM which keeps flooding its own file system. We also launch a benign VM on the same server. We use lmbench [2] to test the disk performance of this benign VM. We alter the interval between each disk access to change the disk bandwidth. Figure 3 shows the disk performance without and with disk DoS attacks. We observe the disk DoS attack can affect the co-located VM's disk performance in both the local and network file system. The effect is more significant when the server's disk bandwidth is high: when the access interval is 0, the victim's disk bandwidth will be reduced by up to 60% by the disk DoS attacks.



(a) Local file system



(b) Network file system

Figure 3: Disk DoS attack

3.3 Attacking Cloud Providers

We evaluate the effects of these host-based DoS attacks on the performance of the cloud provider's management services, using the OpenStack cloud software [3].

The OpenStack system is conprised of three types of servers. A `Controller Node` is a centralized server responsible for managing cloud services and coordinating between customers and cloud servers. A `Compute Node` is responsible for hosting customers' VMs. A `Storage Node` is a server to store VMs' disks. We set up the system with one `Controller Node`, two `Compute Nodes` and one `Storage Node`. We use server $\mathbb{C}_1$ from Table 1 as the `Controller Node`, servers $\mathbb{C}_2$ and $\mathbb{C}_3$ as the `Compute Nodes`, and server $\mathbb{B}$ as the `Storage Node` running the Network File System (NFS) to store VMs' disks. We launch a malicious VM on each `Compute Node`. They conduct one of the memory, network or disk DoS attacks. We measure the performance of different tasks conducted by the `Controller Node` under the attacks.

VM launch/termination. When launching a VM, the cloud provider selects the appropriate `Compute Node`, and sets up network and block devices for the VM. Then the `Compute Node` fetches the OS image from the `Storage Node` and boots up the OS. Figure 4a shows the launch performance (launch time) of different VM configurations, when the `Compute Node` has no DoS attacks (*baseline*), or has memory, network, or disk attacks, separately. We see that the memory DoS attack does not affect the VM launch time (up to 30% overhead), as the VM launch is a network-bound workload. The network attack causes up to 8.1× overhead, as it saturates the network resource of a `Compute Node` and delays the hypervisor fetching OS images. The disk attack can incur up to 18.8× overhead, as it saturates the network resource of the `Compute Node` as well as the `Storage Node`.

When terminating a VM, the `Compute Node` saves the OS back to the `Storage Node`. Figure 4b shows the termination performance. The network attack increases the completion time by 3.3× and the disk attack increases the time by 5.1×.

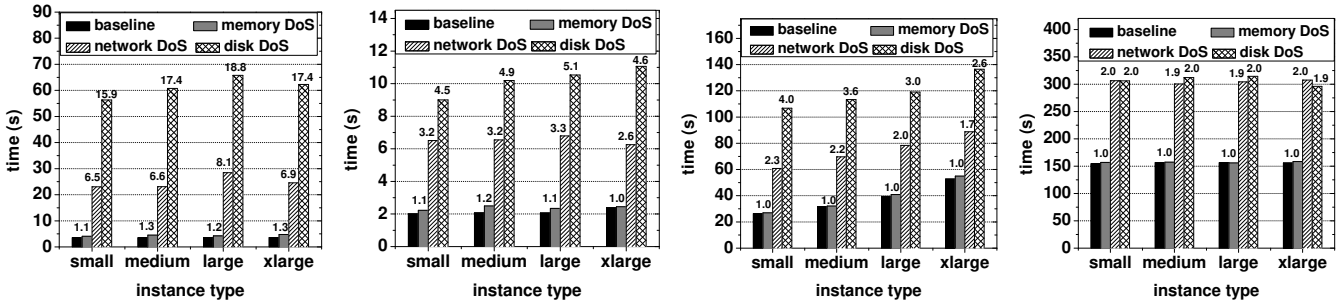
VM migration. The cloud provider migrates the VMs from one `Compute Node` to another for the purpose of resource optimization and workload balancing. Figure 4c shows the live migration completion time of VMs without and with DoS attacks. We observe similar results: the network and disk attacks have more impact on the performance than the memory attacks. The network contention can incur up to 2.3× delay and the disk contention can incur 4× delay.

VM snapshot. The cloud provider takes a VM snapshot, serving as a failover for a VM crash or a new VM image for other customers. The `Compute Node` gets the disk image from the `Storage Node`, and then uploads it to the `Controller Node`. Figure 4d shows the performance impact on VM snapshot due to the DoS attacks. We observe both the network and disk attacks have significant impact (around 2×): the network attack saturates the `Compute Node`'s network resource, and affects its operation of fetching images from the `Storage Node` and sending them to the `Controller Node`. The disk attack saturates the network resources of both `Compute Node` and `Storage Node`, and slows the image transfer between the `Compute Node` and `Storage Node`.

To sum up, the network and disk DoS attacks are effective in degrading the cloud provider's performance. This is because the cloud management services are usually network or disk intensive. So the network or disk contention causes more damage than the memory contention. Memory DoS attacks are more effective when attacking the guest VMs which run memory intensive applications (Figure 1).

4. DATACENTER-WIDE DOS ATTACKS

We now show how much damage an attacker can inflict on a datacenter at minimal cost.



(a) VM launch (b) VM Termination (c) VM migration (d) VM snapshot

Figure 4: DoS attacks in OpenStack

The attacker's strategy is to launch VMs on as many cloud servers as possible. By doing so, each server that hosts the malicious VMs will suffer from host-based DoS attacks. Assume the attacker is allowed to launch at most $\mathbb{N}$ VMs in the cloud within his budget, he can conduct the following procedures to bring the maximum damage to the datacenter.

First he requests to launch $\mathbb{N}$ VMs in the cloud. Then he runs covert-channel attacks in these VMs which can identify his VMs that share the same server [21]. If some of his VMs are placed on the same server, he keeps one VM and reboots the other VMs, which causes these VMs to be placed on other servers. The steps of running covert-channel attacks to discover redundant VMs on the same server, and rebooting these so that they are launched on other servers, is repeated until all the $\mathbb{N}$ VMs are in different host servers. Then each VM can run DoS attacks. We illustrate these steps in detail.

4.1 Impact of Power-aware VM Placement Policies on Launching Attacker VMs

Since customers are not allowed to select host servers for their VMs, the attacker has to launch a large number of VMs in the cloud to cover as many servers as possible. The number of infected servers depends highly on the VM scheduling policy. Some policies schedule the malicious VMs on the same servers so fewer servers get infected; while some policies evenly distribute the malicious VMs on different servers so that these VMs can spread out across the datacenter.

We consider a set of power-aware VM scheduling policies from [8]. The cloud provider uses different strategies to optimize VM placement for power efficiency: (1) during VM launching, the cloud provider tries to allocate these VMs on the smallest number of servers (static policy). (2) During runtime, the cloud provider dynamically checks if one server is overloaded or underloaded. There are several approaches to set such thresholds, e.g., Static Threshold (`THR`), whose utilization threshold is a constant, or Interquartile Range (`IQR`), whose utilization threshold is dynamically adjusted as being negatively linearly correlated to the interquartile range of server utilization in the datacenter. (The cloud provider collects each server's utilization and calculates the interquartile range of all the servers' utilizations. A higher interquartile range will lead to a lower utilization threshold). If a server is overloaded, the cloud provider selects some VMs and migrates them to other servers. There are different approaches for the cloud provider to select VMs for migration, e.g., the Minimum Migration Time (`MMT`) method selects the VM with the shortest migration time, and the Minimum Utilization (`MU`) method selects the VM with the smallest CPU utilization. If a server is underloaded, the cloud provider migrates all the VMs from this server to other servers and shuts down this server to save power.

The cloud provider can combine some of the above methods to define his VM placement policy. For instance, the cloud provider can just perform static VM placement optimization during VM launching (denoted as `STATIC`). It can also perform dynamic VM migration by using `THR` to set utilization threshold and using `MMT` to select VMs for migration (denoted as `THR-MMT`). Other policies (`THR-MU`, `IQR-MMT`, `IQR-MU`) can be defined in a similar way.

Evaluation. We evaluate the impact on the placement of attacker VMs, under the above power-aware placement policies. We use CloudSim [9], a popular large-scale cloud infrastructure simulator, to evaluate the attack strategy. We simulate a cloud system comprising 800 homogenous servers. Each server has 8 physical cores, 32GB memory and 1TB disk. We use real workload traces from the CoMon project in PlanetLab [4], which collected data of CPU utilization from thousands of VMs on servers located at more than 500 places around the world. These workload traces are assigned to VMs launched in the cloud. The attacker launches one VM every minute, and each VM has one virtual CPU, 4GB memory and 25GB storage.

Figure 5 shows the attacker's coverage when the simulated system holds different numbers of VMs (800, 1600, 2400, 3200). The coverage is defined as the ratio of the number of infected servers to the number of active servers. First, we observe a constat coverage in the `STATIC` policy when the number of malicious VMs is large. This is because when the attacker launches malicious VMs, the cloud provider consolidates these VMs on newly bootup servers, and these VMs do not spread to other servers that host victim VMs. The number of infected servers are linearly related to the number of malicious VMs. Second, we observe that the four dynamic policies are more vulnerable to the attacker VMs' coverage of the datacenter's servers than the `STATIC` policy. This is because the dynamic policies also perform runtime optimizations, which migrate and spread the malicious VMs to more active servers and increase the coverage. Third, we observe that the attacker needs fewer VMs to cover the datacenter when the datacenter has fewer victim VMs, as there are fewer active host servers for the attacker to cover.

To sum up, the attacker has lower cost when the cloud provider adopts more advanced power-aware VM placement policies. This should alert datacenter designers to consider such DoS attacks when choosing VM scheduling policies.

4.2 Reducing Co-located VMs

After the initial placement of the attacker VMs, there may be multiple malicious VMs co-locating on the same server.

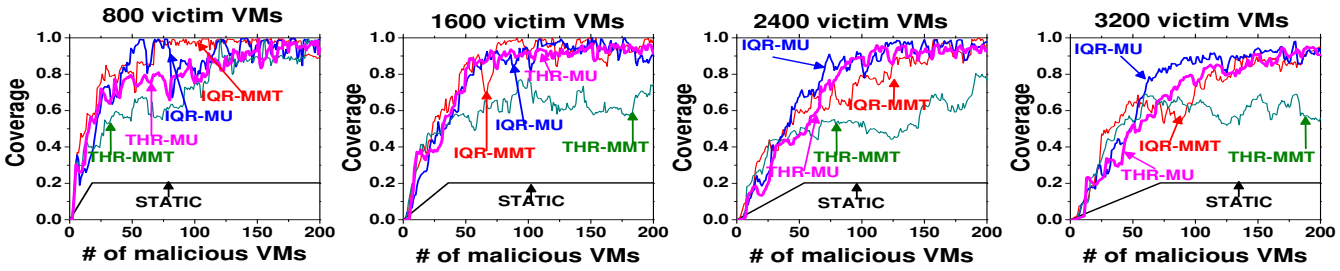


Figure 5: Malicious VMs coverage under different placement policies.

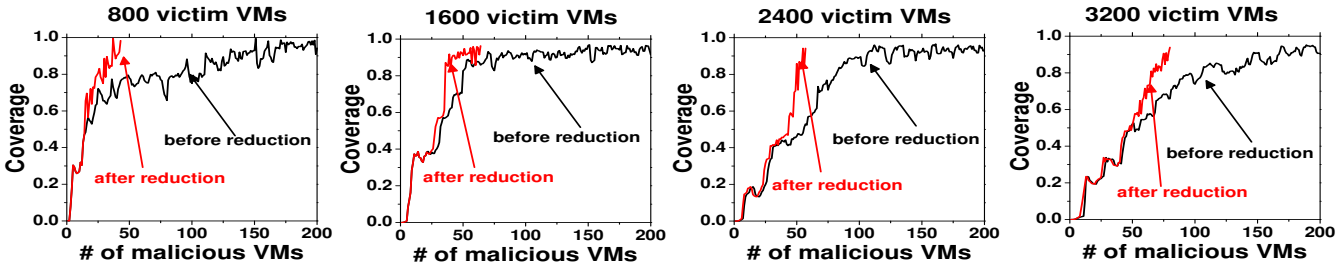


Figure 6: Malicious VMs coverage before and after co-location reduction.

This is not efficient since the attacker only needs one VM on each server to perform the attacks. He needs to identify these co-located VMs and keep only one VM on each server.

The attacker can exploit the covert-channel technique [21] to identify the co-located VMs. Specifically, the attacker selects one VM as a sender, which tries to broadcast a pre-set message to all co-located VMs. To transmit bit "1", this sender conducts DoS attacks for a fixed period. To transmit bit "0", this sender stays idle during the period. All of the other VMs perform regular resource accesses and measure the access time. If one VM observes that its access time follows a pattern that matches the pre-set message, then this VM is receiving the covert-channel message from the sender on the same server. The attacker traverses all the VMs by setting each one as the covert-channel sender, until all co-located VMs are identified and shut down. The feasibility of such covert-channel attacks with high bandwidth and reduced error rate has been shown in public clouds [21].

Figure 6 shows the results of co-location reduction (under `THR-MU` policy). Before checking, the attacker launches 200 VMs in the datacenter, a lot of which are co-located on the same servers. After terminating unnecessary VMs, the attacker only needs to keep around 50 VMs to achieve the same coverage. This can significantly reduce the attack cost.

After a long time, VM allocations in the datacenter may change: VMs can be shut down, launched, or migrated. So it is necessary for the attacker to periodically repeat the launching of attacker VMs and reducing their co-location on the same servers to maintain high infected server coverage.

5. DEFENSE

Although host-based DoS attacks can cause severe loss of availability on a cloud server, detecting them is challenging. First, the malicious VMs behave like normal VMs and never violate the user agreement policy specified by the cloud vendor. Second, the malicious VMs have different ways to deplete the servers' resources and it is hard to use a general-purpose method to monitor all layers of resources. Third, the adversary can use one single tiny VM instance to affect the entire server, and the cloud provider cannot easily identify the malicious VM among all the VMs hosted on the server. Past work offer solutions to defeat DDoS attacks [13, 18]. However, those solutions usually focus on the application level, and target specific victim applications. They cannot be applied to host-based DoS attacks.

We now propose a general-purpose method to defeat these host-based DoS attacks on a cloud server. The key insight of our method is that *the access characteristics to one computing resource (e.g., memory, disk, network) by a known, stable software program will follow a certain probability distribution. If excessive computing resource contention exists in a host-based DoS attack, then the probability distribution of the program's resource usage will change, which can be observed by the cloud provider.* So for each computing resource, the cloud provider can launch a program (dubbed Testing Program) in the host OS of the tested server. This Testing Program has a known stable probability distribution of access characteristics to this resource (denoted as reference samples). At runtime, the cloud provider collects the Testing Program's access characteristics (denoted as monitored samples). If the probability distribution of the monitored samples has a huge divergence from the probability distribution of the reference samples, the cloud provider can suspect that at least one guest VM is executing host-based DoS attacks targeting this computing resource. Note that benign VMs may also alter the Testing Program's behaviors for a short time due to their short bursts of high resource consumption. In order to rule out this kind of false positives, we monitor the Testing Program's behavior constantly and raise the alarm only when we observe that the Testing Program is affected in multiple consecutive measurement windows. Then the cloud provider selectively restricts some of this guest VMs' resource usage and runs the above test to identify the malicious VM(s).

5.1 Methodology

Protecting a server from DoS attacks has three phases: monitoring, pinpointing and mitigation.

Monitoring. The cloud provider can use a set of Testing Programs to test the resource contention on the server.

It first performs offline training by running these TESTING PROGRAMS on a dedicated server with the same hardware configurations as the monitored server, to get nR reference samples of its resource accesses ($[XR_1, XR_2, ..., XR_{nR}]$). Then it launches the same TESTING PROGRAMS on the monitored server and collects nM monitored samples ($[XM_1, XM_2, ..., XM_{nM}]$) periodically.

The TESTING PROGRAMS must be carefully designed to meet the following requirements: (1) they should be sensitive enough to reveal the resource contention caused by host-based DoS attacks; (2) they should have stable resource access probability distributions; (3) they should not impact other VMs' performance. We set the TESTING PROGRAMS for memory, network and disk resources as below:

- *Memory*: the TESTING PROGRAM allocates a memory buffer with the size of Last Level Cache (LLC). It accesses all the data in this buffer sequentially and measures the total time as a sample. It repeats this task for a number of times to form a set of samples.
- *Network*: the TESTING PROGRAM establishes a TCP connection to a remote server and measures the connection time as a sample. It repeats this task for a number of times to form a set of samples.
- *Disk*: the TESTING PROGRAM stores a large file on the disk. It reads all the data in this file in a sequential order and measures the completion time as a sample. It repeats this task for a number of times to form a set of samples.

We use the two-sample Kolmogorov-Smirnov (KS) test [14] to check if the reference and monitored samples belong to the same probability distribution. The KS statistic D is defined in Equation 1 below, where $F_n(X)$ is the empirical distribution function of the samples $[X_1, X_2, ..., X_n]$, and *sup* is the supremum function (i.e., returning the maximum value). We establish the null hypothesis that monitored samples of the TESTING PROGRAMS on the monitored server are drawn from the same distribution as the reference samples from offline training. We can reject such a hypothesis with confidence level $1 - \alpha$ if the KS statistic, $D_{nM,\,nR}$, is greater than predetermined critical values $D^{\alpha}_{nM,\,nR}$. Then the cloud provider can conclude that the monitored server is probably affected by a DoS attack, and trigger the pinpointing phase.

$$D_{nM,\,nR} = \sup_x \mid F_{nM}(XM) - F_{nR}(XR) \mid \qquad (1)$$

Pinpointing. If the KS test rejects the null hypothesis, the cloud provider can conclude that one or more VMs on this server is probably malicious. To identify the malicious VM(s), the cloud provider selects part of the VMs, and throttles down these VMs' execution (i.e., reduce these VMs' impact) on this resource for a short time, while performing the KS test on the TESTING PROGRAM. If throttling down these VMs can significantly reduce the contention on this resource, and the null hypothesis becomes accepted, then the cloud provider can narrow down the malicious VMs to these selected VMs. Otherwise, the malicious VMs will be within the rest of the VMs. The cloud provider can quickly pinpoint the malicious VM(s) using this method combined with binary search. The time complexity is $O(\log n)$, where n is the number of VMs on the monitored server.

We can throttle down VM execution on different layers of resources with the following methods.

- *Memory*: We can use the duty cycle modulation [1] to regulate CPU execution speed. To achieve this, we can modify the register `IA32_CLOCK_MODULATION` to limit one VM's CPU execution to the slowest speed (i.e., 1/16, execute only 1 out of 16 cycles). Then very little memory contention is induced to the server by this VM.
- *Network*: We can set up firewalls to temporarily limit the VM's network connections, releasing its impact on the network resource contention. This can be done by configuring the network settings in the host OS and the physical network routers in the cloud system.
- *Disk*: We can reduce one VM's disk bandwidth to eliminate its impact on disk contention. We can configure the OS filesystem to temporarily set the upper disk limit for one VM (e.g., 1MB). Then this VM has no ability to deplete the disk resource.

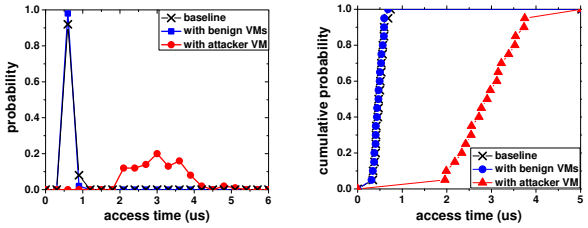
Mitigation. There are multiple ways to thwart the malicious VMs discovered in the pinpointing phase. The cloud provider can keep throttling down these VMs' resource usage and send warnings to the customers. It restores these VMs' executions when they stop the host-based DoS attacks. The cloud provider can also directly suspend or terminate these VMs and block the customers' account.

5.2 Evaluation

We implemented the defense system on the OpenStack platform. Our testbed comprised two cloud servers from Table 1. Server $\mathbb{C}_1$ was configured as the `Controller Node`. Server $\mathbb{A}$ was configured as a `Compute Node` to host VMs.

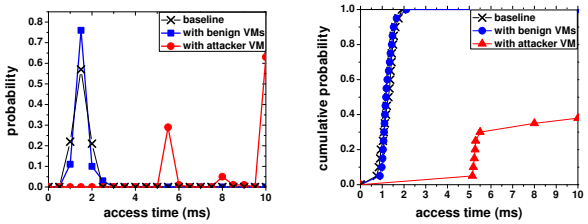
Detection scheme validation. We show how the KS test is used to reveal malicious contention in the monitoring and pinpointing stages. We consider three scenarios: (1) The TESTING PROGRAM runs alone, which produces the reference samples as baseline. (2) The TESTING PROGRAM runs along with 8 benign VMs, running SPEC2006, PARSEC or apache applications. This produces the monitored samples in a benign environment. (3) The TESTING PROGRAM runs along with 7 benign VMs and 1 attacker VM, running the DoS attacks. This generates the monitored samples in an adversarial environment. Figures 7, 8 and 9 show sample probability and cumulative probability of these three cases, under memory, network and disk contention separately. We observe that the distributions of reference samples and monitored samples in the benign case are similar. So the benign VMs have little effect on the TESTING PROGRAMS. However, the monitored samples in the adversarial case have a huge deviation from the baseline, as the attacker VM has significant impact on the TESTING PROGRAM. So the KS values (the largest vertical distances between the cumulative distribution lines of reference and monitored samples) can distinguish the benign case and adversarial case.

In another experiment, we launch eight VMs on the server: one malicious VM conducts a DoS attack, while the remaining benign VMs run SPEC2006, PARSEC or apache programs. Figures 10, 11 and 12 show the KS values of the TESTING PROGRAMS in four stages. In the monitoring stage (I), the monitored server periodically (every 20s) runs the TESTING PROGRAMS and conducts the KS tests. The attacker does nothing so the monitored samples of the TESTING PROGRAMS follow the same probability distribution as the reference samples. In stage II, the attacker begins the attack and the cloud provider immediately detects a high KS value. It repeats the KS test for five times and keeps observing abnormal KS values, thus confirming the attacks.



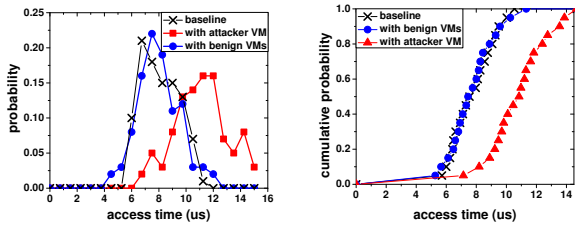
(a) Probability Distribution (b) Cumulative Distribution

Figure 7: Probability distribution and cumulative distribution of the TESTING PROGRAM for memory contention.



(a) Probability Distribution (b) Cumulative Distribution

Figure 8: Probability distribution and cumulative distribution of the TESTING PROGRAM for network contention.



(a) Probability Distribution (b) Cumulative Distribution

Figure 9: Probability distribution and cumulative distribution of the TESTING PROGRAM for disk contention.

Then it triggers the pinpointing stage (III): the server selectively throttles down a set of VMs and performs the KS test. In Figures 10, 11 and 12, the server first throttles down VMs 5, 6, 7, 8, and the KS test shows a low value. The server knows that the attack comes from VMs 5, 6, 7, 8. Then it can throttle down VMs 7 and 8, and the KS value is high, indicating the attack comes from VMs 5 and 6. Then it can throttle down VM 6, and now the KS value is still high. Then it is able to pinpoint the attacker as VM 5. In the mitigation stage IV, the cloud provider shuts down this malicious VM, and the KS value goes back to normal.

Performance. We measure the performance overhead when using the TESTING PROGRAMS to monitor the server. Figures 13, 14 and 15 show the server's performance with and without the monitoring service. We observe the monitoring activity does not incur performance overhead to the server. This is because the TESTING PROGRAMS are designed to be lightweight and they do not affect the server's performance.

Throttling down VMs at the pinpointing stage can delay the VMs' execution. However, this only happens when the server is identified as being attacked, which is relatively rare. Besides, the pinpointing stage can be finished in a very short time: each round only needs less than 1 second, and there are $O(\log n)$ rounds to pinpoint the attacker. So this stage's impact to the system's performance can be ignored.

6. RELATED WORK

Memory Resource Contention. Grunwald and Ghiasi [11] studied the effect of trace cache evictions on the victim's execution with Hyper-Threading enabled in an Intel Pentium 4 Xeon processor. Woo and Lee [20] explored frequently flushing shared L2 caches on multicore platforms to slow down a victim program. They studied saturation and locking of buses that connect L1/L2 caches and the main memory [20]. Moscibroda and Mutlu [16] studied contention attacks on the schedulers of memory controllers. Zhang et al. [23] explored the feasibility of memory DoS attacks on different memory layers in modern cloud servers, and evaluated the attack effects in real cloud settings.

Network Resource Contention. Bedi et al. [7] proposed an attack which causes contention in the Network Interface Controller to degrade the victim's performance. Huang and Lee [12] proposed cascading performance attacks, in which an attacker VM exhausts the I/O processing capabilities of the Xen Dom0, thus degrading the guest victim VM's performance. Similarly, Alarifi and Wolthusen [6] exploited VM migration to degrade Dom0 performance.

Disk Resource Contention. Yang et al. [22] proposed an approach to reverse-engineer the I/O scheduling in the virtualization platform, with which they can design denial of service attacks on the disk I/O resources. Chiang et al. [10] designed a more efficient adaptive attack, which identifies the I/O usage pattern of the victim, and synchronizes the attack phase with the victim.

The above work aim to compromise the service availability of a specific victim machine. Different from these work, our attacks attempt to affect an entire cloud server, including all the guest VMs as well as the cloud management tasks. We also consider datacenter-level DoS attacks: we propose new techniques (e.g., launching and reducing co-located VMs) to show how to economically attack the whole datacenter.

7. CONCLUSIONS

We show host-based DoS attacks on cloud datacenters, and evaluate their damage at two levels. At the server level, we study how a malicious VM can generate malicious contention on different types of resources, i.e., memory, network and disk. We show the malicious VM can also affect the performance of co-located VMs and cloud providers in managing cloud services. At the datacenter level, we conduct large-scale simulation to show how the attacker can use a smaller number of VMs to compromise the availability of more cloud servers, or even the entire datacenter. We show that power-efficient VM scheduling algorithms can in fact make it easier for the attacker! Our results indicate such DoS attacks pose serious availability threats to datacenters.

We then propose a general method to detect different types of host-based DoS attacks. We use a set of TESTING PROGRAMS to monitor the resource usage on the cloud server using statistical methods. Once we detect that a certain type of resource is being depleted by a likely DoS attack, we use our resource throttling technique to quickly pinpoint the attacker VMs, and mitigate the damage. Our evaluation results show this method can effectively protect the cloud servers' availability with little performance overhead.

8. ACKNOWLEDGMENTS

We thank Dr. Zhenyu Wu for invaluable discussions, and the anonymous reviewers for their feedback on this work. This work was supported in part by the National Science Foundation under grant NSF CNS-1218817.

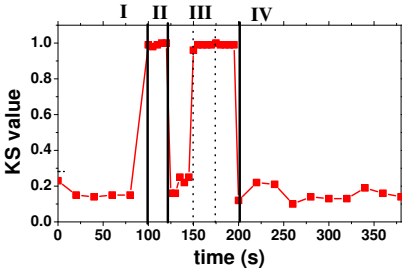


Figure 10: Detecting Memory attacks

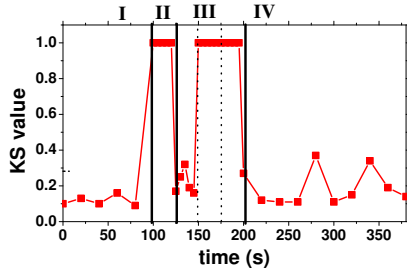


Figure 11: Detecting Network attacks

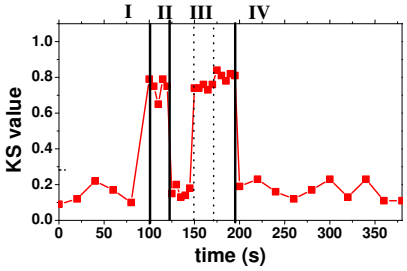


Figure 12: Detecting Disk attacks

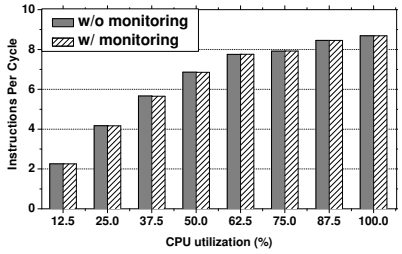


Figure 13: Performance overhead for monitoring memory attacks

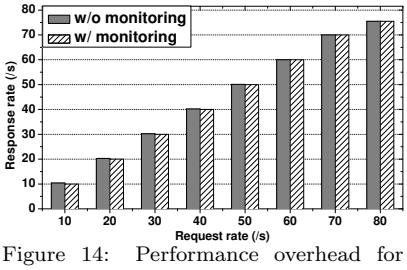


Figure 14: Performance overhead for monitoring network attacks

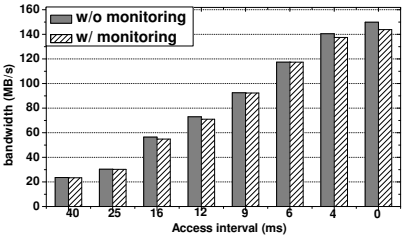


Figure 15: Performance overhead for monitoring disk attacks

9. REFERENCES

[1] Intel 64 and IA-32 architectures software developer's manual, volume 3: System programming guide. http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

[2] Lmbench - tools for performance analysis. http://www.bitmover.com/lmbench/.

[3] Openstack cloud software. http://www.openstack.org/.

[4] Planetlab. https://www.planet-lab.org/.

[5] Welcome to the httperf homepage. http://www.hpl.hp.com/research/linux/httperf/.

[6] S. Alarifi and S. D. Wolthusen. Robust coordination of cloud-internal denial of service attacks. In *Intl. Conf. on Cloud and Green Computing*, 2013.

[7] H. S. Bedi and S. Shiva. Securing cloud infrastructure against co-resident DoS attacks using game theoretic defense mechanisms. In *Intl. Conf. on Advances in Computing, Communications and Informatics*, 2012.

[8] A. Beloglazov and R. Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 2012.

[9] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 2011.

[10] R. Chiang, S. Rajasekaran, N. Zhang, and H. Huang. Swiper: Exploiting virtual machine vulnerability in third-party clouds with competition for i/o resources. *IEEE Trans. on Parallel and Distributed Systems*, 2015.

[11] D. Grunwald and S. Ghiasi. Microarchitectural denial of service: Insuring microarchitectural fairness. In *ACM/IEEE Intl. Symp. on Microarchitecture*, 2002.

[12] Q. Huang and P. P. Lee. An experimental study of cascading performance interference in a virtualized environment. *SIGMETRICS Perf. Eval. Rev.*, 2013.

[13] J. Idziorek, M. Tannian, and D. Jacobson. Detecting fraudulent use of cloud resources. In *ACM Workshop on Cloud Computing Security*, 2011.

[14] F. J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 1951.

[15] R. Miao, R. Potharaju, M. Yu, and N. Jain. The dark menace: Characterizing network-based attacks in the cloud. In *ACM Conference on Internet Measurement Conference*, 2015.

[16] T. Moscibroda and O. Mutlu. Memory performance attacks: Denial of memory service in multi-core systems. In *USENIX Security Symp.*, 2007.

[17] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and implementation of the sun network filesystem, 1985.

[18] P. Shamsolmoali and M. Zareapoor. Statistical-based filtering system against ddos attacks in cloud computing. In *Intl. Conf. on Advances in Computing, Communications and Informatics*, 2014.

[19] Top Threats Working Group. The treacherous 12 cloud computing top threats in 2016. In *Cloud Security Alliance*, 2016.

[20] D. H. Woo and H.-H. S. Lee. Analyzing performance vulnerability due to resource denial-of-service attack on chip multiprocessors. In *Workshop on Chip Multiprocessor Memory Systems and Interconnects*, 2007.

[21] Z. Wu, Z. Xu, and H. Wang. Whispers in the hyper-space: High-speed covert channel attacks in the cloud. In *USENIX Security Symp.*, 2012.

[22] Z. Yang, H. Fang, Y. Wu, C. Li, B. Zhao, and H. Huang. Understanding the effects of hypervisor i/o scheduling for virtual machine performance interference. In *IEEE Intl. Conf. on Cloud Computing Technology and Science*, 2012.

[23] T. Zhang, Y. Zhang, and R. B. Lee. Dos attacks on your memory in the cloud. In *ACM Asia Conference on Computer and Communications Security*, 2017.