# The Beauty or The Beast? Attacking Rate Limits of the Xen Hypervisor

Johanna Ullrich(✉) and Edgar Weippl

SBA Research, Vienna, Austria
{JUllrich,EWeippl}@sba-research.org

**Abstract.** Rate limits, i.e., throttling network bandwidth, are considered to be means of protection; and guarantee fair bandwidth distribution among virtual machines that reside on the same *Xen* hypervisor. In the absence of rate limits, a single virtual machine would be able to (unintentionally or maliciously) exhaust all resources, and cause a denial-of-service for its neighbors.

In this paper, we show that rate limits snap back and become attack vectors themselves. Our analysis highlights that *Xen*'s rate limiting throttles only outbound traffic, and is further prone to burst transmissions making virtual machines that are rate limited vulnerable to externally-launched attacks. In particular, we propose two attacks: Our side channel allows to infer all configuration parameters that are related to rate limiting functionality; while our denial-of-service attack causes up to 88.3 % packet drops, or up to 13.8 s of packet delay.

## 1 Introduction

Cloud computing is here to stay; and has become an all-embracing solution for numerous challenges in information technology: Defending against cyber attacks, countries back up their "digital monuments" in clouds [1]; clouds support censorship evasion [2]; clouds accommodate power-restrained mobile devices with computing [3]; automotive clouds connect a vehicle's sensors and actuators with other vehicles or external control entities for safer and more comfortable driving [4]; and also healthcare applications are hosted in the cloud [5]. Its total market is worth more than 100 billion US dollars [6]; and recently, even the conservative banking sector is jumping on the bandwagon [7,8].

A key technology in cloud computing is virtualization as provided by the *Xen* hypervisor [9] that enables multiple virtual instances to share a physical server [10]; but at the same time, resource sharing provides opportunity for adversarial virtual machines to launch attacks against its neighbors. For example, side channels exploiting shared hard disks [11] or network capabilities [12] allow to check for co-residency of two virtual machines; data might be leaked from one virtual instance to another via covert channels exploiting CPU load [13] or cache misses [14]; an instance might free up resources for itself when tricking the neighbor into another resource's limit [15]; and shared network interfaces allow to infer a neighbor's networking behavior [16,17]. Mitigation follows two principal


I. Askoxylakis et al. (Eds.): ESORICS 2016, Part II, LNCS 9879, pp. 493–511, 2016.
DOI: 10.1007/978-3-319-45741-3_25

directions: On the one hand, dedicated hardware eliminates mutual dependencies and thus the threat of co-residency, but contradicts cloud computing's premise of resource sharing. On the other hand, isolation reduces the impact of a virtual machine's behavior on its neighbors despite resource sharing. With respect to networking, rate limits are introduced as means of isolation in order to throttle a virtual machine's maximum amount of traffic per time interval. This approach is considered to guarantee fair distribution of bandwidth among virtual instances and mitigates denial-of-service of neighbors in case a single instance (accidentally or maliciously) requests all bandwidth. The *Xen* hypervisor provides such a rate limiting functionality [18].

The introduction of a countermeasure should raise the question whether it does not form a new attack vector itself. Throttling network traffic however seems to be such a universal approach that its implementation into the *Xen* hypervisor is barely scrutinized. Solely, [19] investigates rate limiting's quality of isolation; [20] analyzes rate limiting with respect to bandwidth utilization. The paper at hand overcomes this gap and examines the impact of *Xen*'s rate limiting functionality on security. Our analysis reveals that rate limits might protect from co-residency threats, but allow (yet unknown) attacks that are directed against the rate limited virtual machine itself. In particular, we propose a side channel and a denial-of-service attack. The side channel reveals *Xen*'s configuration parameters that are related to the rate limiting functionality, while the denial-of-service attack causes up to 88.3 % of packet loss or up to 13.8s of delay in benign connections. Our results emphasize that *Xen*'s rate limiting snaps back, and revision should be considered.

The remainder of the paper is structured as follows: Sect. 2 provides details on *Xen*'s networking in general and its rate limit functionality in particular, whereas Sect. 3 analyzes this mechanism with respect to security. Section 4 presents our side channel revealing configuration parameters and respective measurement results; Sect. 5 presents three flavors of our denial-of-service attacks and discusses them with respect to their impact on benign connections. It is followed by related work in Sect. 6. Overall results are discussed Sect. 7. Section 8 concludes.

## 2 Background

This section first provides a general overview on Xen's networking architecture. Its rate limiting functionality however throttles only a virtual machine's outbound traffic; thus, we describe a virtual machine's outbound traffic path in a second step. Finally, we focus on the credit-based algorithm eventually throttling a machine's traffic.

***General Networking Architecture:*** The Xen hypervisor follows the approach of paravirtualization; it provides device abstractions to its virtual machines – in terms of Xen virtual machines are called *domains* – so that all sensitive instruction like those for device I/O are redirected over the hypervisor. Paravirtualizing hypervisors do not need specific hardware capabilities; but require modifications

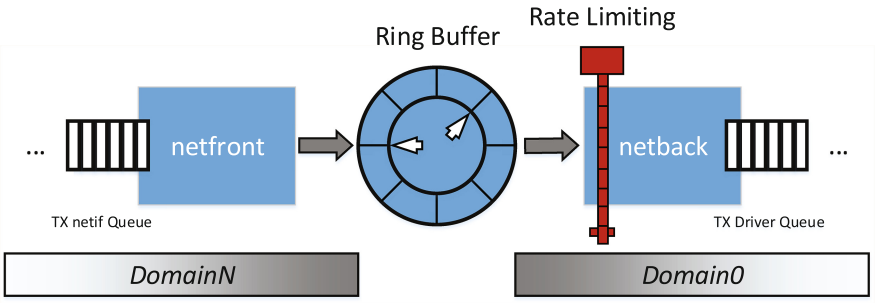


**Fig. 1.** Xen's outbound traffic path

of the operating systems running in the virtual machines [9]. With respect to Xen, the hypervisor in the narrower sense is responsible for CPU scheduling, memory management and interrupt forwarding. The remainder tasks are delegated to *domain0* – a privileged virtual machine with the right to access physical I/O devices and to interact with other (non-privileged) domains. Abstract networking devices consist of two distinct parts: (1) *netfront* devices are provided to non-privileged domains replacing classic network interfaces; (2) its counterpart *netback* resides in *domain0*, multiplexes packets from multiple *netfront* devices and forwards them to the physical network interface card as in standard Linux operating systems [19,21].

***Outbound Traffic Path:*** Packets originating from non-privileged virtual machines (*domainN*) have to pass *domain0* on their way to the physical network; the respective handover path is depicted in Fig. 1. Therefore, Xen provides descriptor rings, i.e., ring buffers, as central points of communication. The ring does not directly contain data; this data is rather stored in buffers that are indirectly referenced via the ring descriptors. Packets pass this path in the following manner. First, packets are enqueued in the virtual machine's network interface TX queue. Then, netfront forwards these packets from the TX queue to the ring buffer, and notifies netback. Netback – being within *domain0* and thus having access to physical drivers – hands them over to the physical network interface card's driver queue and removes them from the ring buffer. Beyond, netback is the place of rate limiting. If a respective virtual machine exceeds its assigned bandwidth quota, netback refrains from taking further packets from the ring buffer and discontinues forwarding for some time. As items are not removed from the ring anymore, the buffer becomes full. As soon as a virtual machine's netfront detects this, it signalizes this fact to the upper networking layers by means of a flag. Packets pending in the ring buffer have to wait for further processing until the next bandwidth quota is received.

***Rate Limiting:*** Rate limiting throttles a virtual machine's bandwidth – however, it confines outbound traffic only – and is configured by means of two parameters [18]. The parameter *rate* defines the respective bandwidth limit in MB/s, while *time window* defines the replenish interval of the rate limiting

algorithm. Its default value is 50 ms. Looking behind the scenes, the algorithm is credit-based[1]. With every packet forwarded from the ring buffer, the respective packet size is subtracted from the remaining credit. In case of lacking credits, two alternatives remain: (1) immediate replenishment of credits and continuation of transmission, or (2) discontinuation and waiting for replenishment of credits at a later point in time. Immediate replenishment is only possible if the last replenishment happened at least the time defined by the parameter *time window* ago. In the alternative case, a timer is set to the time of next replenishment, and packet transmission is rescheduled as soon as credits are regained. According to the parameters *rate* $r$ and *time window* $t$, the credit bytes per interval $c$ calculates to $c = r \cdot t$, and the total amount of available credit is limited to this number. This implies that accumulating unused credits for later transmission is impossible. There is a single exception if $c$ remains below 128 kB, i.e., rates of less than 2.5 MB/s at the default time window, as then jumbo packets might seize up the interface. In such a case, credit accumulation up to 128 kB is allowed.

## 3 Security Analysis

In this section, we perform a manual security analysis of the Xen's rate limit functionality. This analysis reveals distinct characteristics that may serve as attack surface; we describe these characteristics, highlight their implication on security and discuss them with respect to cloud computing. Finally, we provide a high-level overview of our attacks that exploit the found characteristics.

***(1) Unidirectional Bandwidth Limits:*** Xen allows to restrict a virtual machine's outbound bandwidth, but inbound remains unlimited without any chance for change. In consequence, the transmission paths are asymmetric. In principle, asymmetry in bandwidth is a known phenomena, e.g., *Asymmetric digital subscriber line* (ADSL), but Xen's asymmetry appears to contradict its application in cloud services as highlighted by the following analogy. ADSL's asymmetry is concordant with its application in consumer broad-band connections. Consumers typically request more downstream than upstream bandwidth, and thus favoring the first direction (at the expense of the latter) is reasonable. Cloud instances predominantly require higher outbound than inbound bandwidth, e.g., when used as application, web or streaming servers. Xen however performs precisely the opposite and limit's the more utilized outbound direction[2].

Bandwidth is not only unequally distributed, but also differs by magnitudes as in consequence inbound traffic is only limited by the underlying hardware. Outbound bandwidth in public clouds starts from 12.5 MB/s for small cloud instances; assuming a 10-Gigabit physical network in the data center, maximum inbound outperforms maximum outbound bandwidth by a factor up to 100.

[1] Kernel 3.16.0, /net/xen-netback/netback.c.

[2] Cloud providers like Rackspace (see https://www.rackspace.com/cloud/servers) or Amazon EC2 (see https://aws.amazon.com/en/ec2/pricing/) typically do not even charge inbound traffic.

***(2) Susceptibility to Burst Transmissions:*** Xen's algorithm is prone to burst transmissions. A virtual machine transmitting high amounts of traffic shoots its wad at the begin of a time slot, and has to wait for new credits then. At the time of replenishment, further packets might already wait for transmission and cause another burst consuming all credits. In consequence, packets experience latencies when pausing for the next slot; however, these latencies are only experienced by outbound traffic due to the unidirectional bandwidth limitation. In case the outbound traffic exceeds the configured bandwidth for a longer period of time, packets might even be dropped: Packets remain in the ring buffer as a result of credit shortage. As a consequence, netfront cannot forward packets to the ring descriptor anymore and causes a growing backlog in the virtual machines TX queue. If the number of packets becomes larger than this queue's size, packets are dropped. By default, time window is set to 50 ms; according to the documentation *"a good balance between latency and throughput and in most cases will not require changing"* [18]. This implies that the credit-based algorithm is rather coarse-grained as time slots in the virtual machine's traffic are of the same order of magnitude as round trip times, and the bursts are externally observable.

***Attacks Exploiting Rate Limiting:*** We found two attacks exploiting these characteristics – a side channel revealing Xen configuration parameters that are related to its rate limit functionality, and a denial-of-service attack causing significant delays and packets drops in benign connections to third parties. We provide a high-level overview on these attacks, before addressing them in more detail in Sects. 4 and 5.

1. **Side Channel:** Pushing a virtual machine into its outbound traffic limits, leads to burst transmissions that can be observed. By measuring time between two bursts, it is possible to infer the parameter *time window* $t$; by summing up the bytes of a burst, an adversary is able to infer the amount of credits $c$ per interval, and subsequently also calculate the *rate* $r$.
2. **Denial-of-Service Attack:** An adversary might force a virtual machine to spend all its credits; in consequence, a virtual machine has not enough credits left in order to serve benign requests. Respective responses are significantly delayed as they have to wait for credit replenishment, or dropped due to full buffers. This denial-of-service attack is insofar remarkably as it exhausts outbound bandwidth in comparison to ordinary bandwidth exhaustion attacks exhausting inbound bandwidth.

## 4 Side Channel

If a virtual machine requires more bandwidth than assigned, its traffic becomes bursty due to Xen's credit-based rate limit algorithm. An adversary might exploit this behavior to determine a virtual machine's configuration parameters *time window* $t$ and *rate* $r$ by means of the following side channel. The adversary

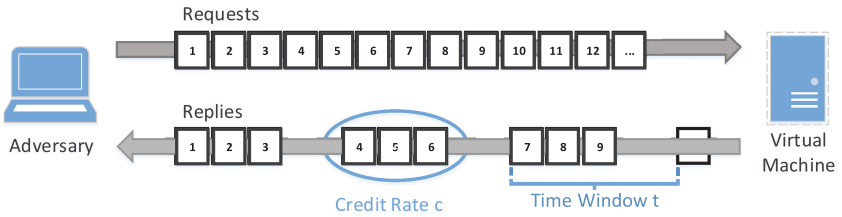


**Fig. 2.** Side channel attack scenario

sends a high number of legitimate requests to the virtual machine. The latter replies according to the chosen protocol; however, the sum of all replies exceeds the assigned bandwidth and outbound traffic becomes bursty as depicted in Fig. 2. The time interval between two bursts is equivalent to the configured *time window* $t$, as the virtual machine receives credits for further transmission immediately after the timer expires. Summing up the size of all packets within a burst allows to determine the victim's *credit rate* $c$. Finally, the adversary is able to calculate the victim's assigned bandwidth (parameter *rate*) $r = c/t$. The side channel is advantageously protocol independent. The only stringent objective is that the virtual machine reliably replies; thus, a wide variety of protocols are worth considering, e.g., ICMP, DNS, etc. The more outbound traffic, the better; the larger the amplification between outbound and inbound traffic, the better; both facilitate to reach the assigned rate limit for outbound traffic.

We evaluated this side channel in our experiments[3]. The virtual machine was limited to 5 MB/s at the default time window of 50 ms. Checking the configuration with *iperf*[4], we measured 4.7 MB/s from the virtual machine to the adversary (throttled outbound traffic), and 117.3 MB/s in the other direction (unthrottled inbound traffic). Attacking the virtual machine, the adversary sent 16 ICMP Echo Request of 1458 bytes, waiting a millisecond before sending the next 16 ICMP Echo Requests causing up to 22.2 MB/s of inbound traffic for the virtual machine. In total, the attack runs for 1000 of such cycles sending in total 16000 Echo Requests. Repeating this attack ten times, we inferred the configuration parameter from the measurements according to the following approaches:

– **Time Window:** The begin of a time window is indicated by a packet following a (larger than usual) pause. Thus, we extracted all packets following a pause of at least 5 ms, and measured the time window between these first packets of subsequent bursts. Rounding off to whole milliseconds, we took the most frequent candidate of all test runs.

[3] For our experiments, we use Xen version 4.4.1 (on Debian 8.2) on an Intel i5-750. On the hypervisor, two virtual machines run Debian 7.9; each guest is pinned to a separate CPU, *domain0* runs on the remaining two CPUs. The two virtual instances were rate limited and bridged via the hypervisor. The adversary ran Debian 8.2 on an Lenovo X200 laptop. The hypervisor and the adversary's laptop were connected via a 1 Gbit/s network switch.

[4] https://iperf.fr/.

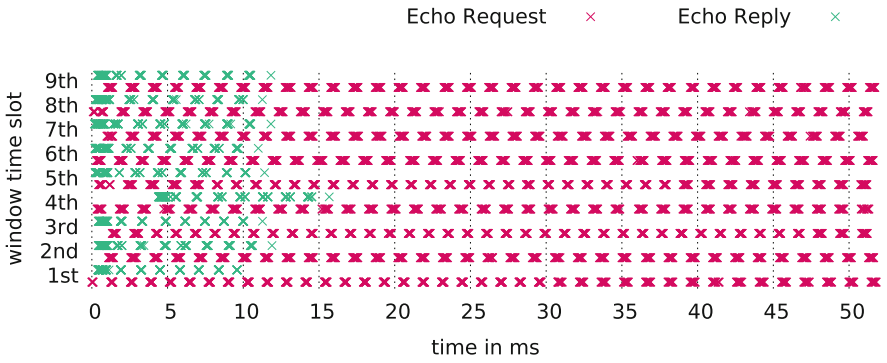


**Fig. 3.** Side channel measurement results using ICMP

– **Credit Rate:** In the previous step, the first packets of bursts have already been determined; the credit rate is now calculated by summing up the size of all packets from this first packet of the burst to the last one. The last packet of the burst is the one right before the first packet of the next burst. Again, the most frequent candidate is taken from all candidates.

This way, we inferred a *time window* $t$ of 52 ms, and a *credit rate* $c$ of 249,318 bytes; the resulting bandwidth $r$ is thus 4.8 MB/s. Figure 3 depicts a network trace of the side channel from the adversary's point of view; for reasons of simplicity, the graph is already slotted in time intervals of 52 ms. While the adversary sends requests in regular intervals, the virtual machines replies predominantly at the begin of a time slot. Afterwards, it remains silent due to lacking further credits. One can also see in the figure that the number of sent replies is high at the begin of a time slot; this is an indicator that all waiting replies are sent at once immediately after credit replenishment. The side channel was measured with different configurations of the virtual machine. First, we altered the bandwidth keeping the time window at the default configuration of 50 ms; results are provided in Table 1. Then, we modified the time window at a fixed bandwidth of 5 MB/s; results are provided in Table 2. The first line of Table 1, and the second line of Table 2 represents the results of the measurement that has been described above.

Our results show that the measured time window is slightly longer than the configured time window. Taking a look into Xen's source code, the time window is strictly speaking the time period for the timer; this additional time of mostly 2 ms might be caused by credit replenishment, packet forwarding, etc. that is necessary after the timer expires. Actual bandwidth appears to be below the configuration parameter; however, our side channel appears to reflect *iperf* measurements well. Measurements for 30 MB/s at 50 ms of Table 1 shows an increased time window; however, evaluation shows two almost equally frequent candidates – 56 ms and 48 ms – both equally distant from the expected 52 ms. Similarly, measurements for 5 MB/s at 20 ms (peaks at 16 ms and 24 ms) as well

**Table 1.** Bandwidth Measurements with Fixed Time Window of 50 ms

| Xen configuration | | Attack parameters | | Side channel | | |
|---|---|---|---|---|---|---|
| Configured bandwidth | iperf (Outbound) | Requests per cycle | Inbound bandwidth | Credit rate $c$ (Measured) | Time window $t$ (Measured) | Rate $r$ (Calculated) |
| MB/s | MB/s | | MB/s | B | ms | MB/s |
| 5 | 4.7 | 16 | 23.3 | 249318 | 52 | 4.8 |
| 10 | 9.4 | 32 | 46.7 | 498636 | 52 | 9.6 |
| 20 | 18.9 | 32 | 46.7 | 998730 | 52 | 19.2 |
| 30 | 27.8 | 48 | 70.0 | 1498824 | 56 | 26.8 |
| 40 | 37.0 | 60 | 87.5 | 1998918 | 52 | 38.4 |

**Table 2.** Time Window Measurements with Fixed Bandwidth of 5 MB/s

| Xen configuration | | Attack parameters | | Side channel | | |
|---|---|---|---|---|---|---|
| Configured time window | iperf (Outbound) | Requests per cycle | Inbound bandwidth | Credit rate $c$ (Measured) | Time window $t$ (Measured) | Rate $r$ (Calculated) |
| ms | MB/s | | MB/s | B | ms | MB/s |
| 70 | 4.8 | 16 | 23.3 | 349920 | 72 | 4.8 |
| 50 | 4.7 | 16 | 23.3 | 249318 | 52 | 4.8 |
| 30 | 4.6 | 16 | 23.3 | 148716 | 32 | 4.6 |
| 20 | 4.9 | 16 | 23.3 | 100602 | 24 | 4.2 |
| 10 | 4.1 | 16 | 23.3 | 49572 | 8 | 6.2 |

as 5 MB/s at 10 ms (peaks at 8 ms and 16 ms) of Table 2 show two such peaks. For the latter however the lower peaks has slightly more candidates. The reason for less quality of the latter two results might be the rather small *time window* $t$. Pauses before first packets of a burst become shorter with decreasing time windows; thus, our algorithm looking for 5 ms pauses might struggle to detect begin packets at such low time windows accurately. This might be overcome by looking for shorter pauses.

## 5 Denial-of-Service

Traffic exceeding the rate limit has to wait for a free time slot in the future; beyond, if the backlog of waiting packets becomes too much, buffers become full and packets are dropped. Deliberately filling the buffers, an adversary might exploit this behavior in order to perform a denial-of-service attack causing significant packet delays or even drops of benign traffic.

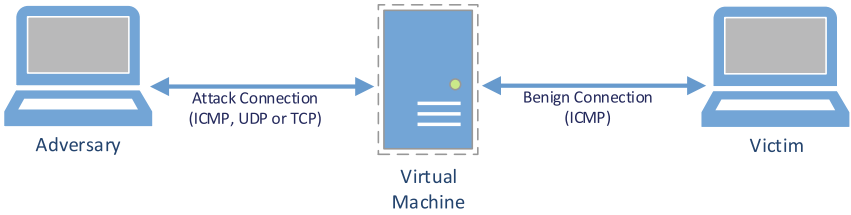


**Fig. 4.** Denial-of-service attack scenario

For evaluation, we extended the measurement setup by an additional host representing the victim[5] as depicted in Fig. 4. The victim had a benign connection to the virtual machine; we decided to probe the virtual machine with ICMP Echo Requests at an interval of 10 ms. In total, these requests (and potentially received replies) require a maximum bandwidth of 19.6 kB/s which is negligible in comparison to the attack traffic. For the adversary, we tested three alternatives for causing the backlog – by means of ICMP Echo Requests, UDP-based traffic amplification and TCP acceleration as described in the following paragraphs.

**ICMP Echo Requests:** As with the side channel, the adversary sends multiple Echo Requests and pauses afterwards for 1 ms repeating both actions in a loop. Echo Requests however bear the drawback of being non-amplifying, i.e., the virtual machine's (maximum) outbound traffic is of the same amount as the inbound traffic from the adversary.

**UDP-Based Traffic Amplification:** A virtual machine might host a service that answers with replies that exceed the requests in size and thus amplifies inbound traffic. An adversary sending numerous such requests is able to trigger more outbound traffic than with ICMP. Susceptible protocols are predominantly UDP-based, e.g., NTP, DNS, SSDP or BitTorrent, and bandwidth amplification factors reach up to 4670.0 [22][6]. For our evaluation, we scripted a simple UDP server that responded with a bandwidth amplification factor of 100. The server ran on the virtual machine; our adversary sent respective UDP requests in the same manner as the ICMP Requests – sending a certain number of UDP requests before pausing for 1 ms repeating both actions in a loop.

[5] The victim ran Ubuntu 14.4 LTS on a Lenovo X60 laptop. The virtual machines were rate limited to 5 MB/s at the default window time of 50 ms.

[6] [22] investigated amplifying protocols with respect to reflective denial-of-service. Such attacks require source address spoofing in order to redirect replies to the victim – a prerequisite that is not necessary for our denial-of-service attack. This implies that (1) there are even more protocols than described in this paper that are susceptible to our attack and (2) ingress filtering does not prevent our attack.

**TCP Acceleration:** TCP connections, e.g., when serving a HTTP request, are frequently asymmetric with respect to transmitted payload; a server is sending amounts of data while the client almost exclusively acknowledges receipt with a couple of bytes. TCP is a reliable protocol, adjust its speed according to given network capabilities and thus does not automatically lead into a denial-of-service attack; but an adversary might intentionally accelerate a TCP connection by means of optimistic acknowledgments [23]. Such optimistic acknowledgments are sent prior the receipt of the respective segment, lead the server to believe in higher available bandwidth and make the server send at a higher speed than normally. For our evaluation, we installed an Apache[7] server on the virtual machine providing a 100 MB file for download. For the adversary, we re-implemented this attack with respect to current TCP implementations as congestion control has significantly changed over the last decade and ran the attack when downloading the previously mentioned file.

**Results for ICMP and UDP-Based Attacks:** Results for the ICMP-based attacks are found in Table 3; results for the UDP-based attack with traffic amplification in Table 4[8]. Inbound bandwidth refers to the traffic that is sent from the adversary to the virtual machine, while the potential outbound bandwidth refers to the bandwidth that would be caused in the reverse direction in the absence of rate limits. The remaining three columns show the average delay of replies to the victim's Echo Requests, the observed maximum delay as well as the relative amount of dropped replies. In both tables, the first line represents the latter values in the absence of an attack for reasons of comparison.

The results highlight the following: (1) All attacks significantly increase the delays by two orders of magnitudes. (2) The higher the ICMP bandwidth, the more packet drops. The average delay however appears to decrease at higher attack bandwidths; and might be an artifact of increased drop rates as less replies were received by the victim and were taken into account for average delay calculation. The maximum delay of *icmp16* might be higher than the remainder for the same reason. (3) UDP-based attacks do not cause any packets drops; average and maximum delay are higher than in the ICMP-based attacks, and appear to be independent of the attack bandwidth. The reason might be that the virtual machine favors ICMP traffic over UDP, and thus drops attack traffic rather than the victim's. However, amplification allows the adversary to reduce the amount of sent traffic in order to gain the same potential outbound bandwidth at the virtual machine; for example, attack *icmp16* leads to the same potential outbound bandwidth as *udp4*. Figure 5 depicts a test run of the ICMP-based attack, Fig. 6 of the UDP-based attack. Both figures show the increased round-trip times of the victim; the first figure further shows packet drops.

**Results for TCP-Based Attack:** The results of our TCP attack are found in Table 5. In comparison to ICMP- or UDP-based attacks, delays are much higher.

[7] https://httpd.apache.org/.

[8] As in the side channel, the results are based on ten test runs each.

**Table 3.** Denial-of-service attack with ICMP echo requests

| ID | Adversary | | | Victim | | |
|---|---|---|---|---|---|---|
| | Requests per cycle | Inbound bandwidth | Potential outbound bandwidth | Average delay | Maximum delay | Dropped replies |
| | | MB/s | MB/s | ms | ms | % |
| noattack | | | | 0.221 | 0.491 | 0 |
| icmp16 | 16 | 23.3 | 23.3 | 14.3 | 65.7 | 67.5 |
| icmp32 | 32 | 46.7 | 46.7 | 23.6 | 49.3 | 85.6 |
| icmp48 | 48 | 70.0 | 70.0 | 21.8 | 51.5 | 83.6 |
| icmp60 | 60 | 87.5 | 87.5 | 16.7 | 52.8 | 88.3 |

**Table 4.** Denial-of-service attack with UDP-based traffic amplification

| ID | Adversary | | | Victim | | |
|---|---|---|---|---|---|---|
| | Requests per cycle | Inbound bandwidth | Potential outbound bandwidth | Average delay | Maximum delay | Dropped replies |
| | | MB/s | MB/s | ms | ms | % |
| noattack | | | | 0.221 | 0.491 | 0 |
| udp4 | 4 | 0.2 | 23.2 | 23.0 | 53.3 | 0 |
| udp8 | 8 | 0.5 | 46.4 | 22.9 | 53.5 | 0 |
| udp12 | 12 | 0.7 | 69.6 | 22.7 | 53.4 | 0 |
| udp15 | 15 | 0.9 | 87.0 | 23.6 | 53.7 | 0 |

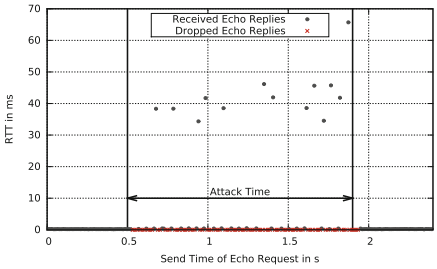


**Fig. 5.** Impact on the victim (icmp16)

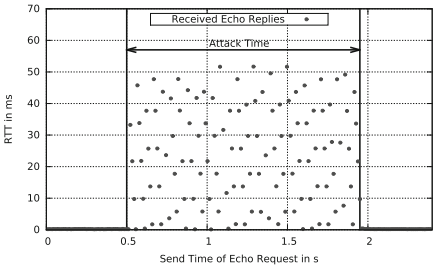


**Fig. 6.** Impact on the victim (udp4)

The average delay is 1625.8 ms, the maximum delay even 13791 ms, i.e., almost 14s. Packet drops are however below the ICMP-based attack: 33.2 %.

Figure 7 shows the sequence numbers of sent TCP acknowledgments and received TCP payload from the adversary's perspective. While the first increases

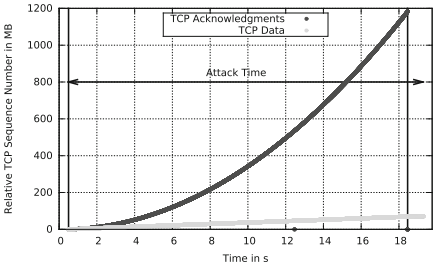


**Fig. 7.** Relative sequence numbers (tcp)

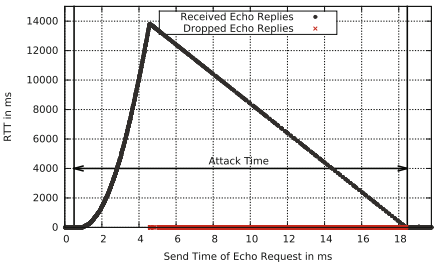


**Fig. 8.** Impact on the victim (tcp)

**Table 5.** Denial-of-service attack with TCP

| ID | Adversary | | | Victim | | |
|---|---|---|---|---|---|---|
| | Requests per cycle | Inbound bandwidth | Potential outbound bandwidth | Average delay | Maximum delay | Dropped replies |
| | | MB/s | MB/s | ms | ms | % |
| noattack | | | | 0.221 | 0.491 | 0 |
| tcp | | 0.01 | 65.8 | 1625.8 | 13791.6 | 33.2 |

exponentially to maximize the virtual machine's congestion window, the latter increases only in a linear manner. This linear increase is caused by the rate limit of 5 MB/s, and provides a first evidence that the virtual machine operates at its networking limits. Moreover, enlarged sections of this figure would clearly depict the bursty transmission and the underlying 50 ms intervals (but were omitted due to space constraints). Figure 8 shows the attack's impact on the victim's round-trip times: Right at the start, round-trip times are as expected less than a millisecond; then, round-trip times start to increase. The maximum recorded delay is 13,791.6 ms. In a third phase, the buffers are full and Echo Replies are dropped at a large-scale. As numerous packets are dropped, the buffer is released and round-trip times decrease back to normal.

## 6 Related Work

Our research is based on three foundations. First, we summarize related work on Xen's network rate limits in general. Then, we discuss known side channels as well as denial-of-service attacks in cloud computing; by now, no approach exploited a hypervisor's rate limit functionality.

***Rate Limit Functionality:*** Related work on Xen's rate limit for networking is rather scarce, and must not be mistaken for (the more frequently discussed) rate limits with respect to CPU scheduling, e.g., [24–26].

Adamczyk et al. [19] investigate rate limits' quality of isolation, i.e., protection against malicious neighbor virtual machines. The authors infer that performance isolation is moderate in case of applied rate limits. Nevertheless, the authors propose round robin for more fairness among co-resident tenants. Mei et al. [20] analyze rate limits with respect to bandwidth utilization, and considers Xen's rate limiting to be *static* as a virtual machine cannot provide spare tokens to neighbors. The authors propose a new bandwidth allocation algorithm that dynamically provides bandwidth based on current and past bandwidth consumption. While this algorithm definitely increases bandwidth allocation, an actively networking virtual machine would increase its assigned bandwidth with time, and lead to starving neighbors. Summarizing, the impact of rate limits on security of the throttled virtual machine themselves has never been raised.

***Side Channels:*** Numerous side channels in cloud computing exploit network timing. Bowers et al. [27] measure file access times to extract data's hardware spread in order to check a cloud provider's hardware redundancy, Benson et al. [28] measure file access times in order to determine the geographic location of files. Multiple side channels check for co-residency, i.e., whether two virtual machines reside on the same physical server: Ristenpart et al. [11] exploit round-trip times, Bates et al. [16,17] packet arrival rates and Herzberg et al. [12] latencies when downloading a file.

Beyond, side channels enabling to spy on neighbors are available. Kadloor et al. [29,30] measure round-trip times in order to infer a neighbor's traffic amount. Bates et al. [16,17] observe a distinct TCP throughput ratio therefore. Herzberg et al. [12] propose address deanonmyization, i.e., discovery of private IP addresses, by exploiting again latencies in file downloads, and further infer the number of intermediate hops by finding the minimal TTL in order to gain a successful connection.

***Denial-of-Service Attacks:*** Alarifi et al. [31] present an attack that forces a virtual instance to migrate to another physical host. This behavior is triggered by co-resident virtual machines riding a workload wave. Ficco et al. [32] propose an attack that stealthily increases resource use (by means of XML-based denial-of-service attack) in order to remain undetected. The authors aim to harm the victim economically as the increased resource use is charged by the provider. Liu et al. [33] discuss under-provision of network links in data centers, e.g., that uplink capacity remains typically smaller than the total subnet bandwidth. An adversary might spot such bottlenecks in clouds and strike it in a concerted action of multiple virtual machines.

Shea et al. [34,35] analyze the impact of TCP SYN floods on virtualized environments and infer that virtualization overhead negatively impacts a host's vulnerability to denial-of-service attacks. Ferriman et al. [36] analyze the impact of denial-of-service attacks on Google App Engine and saw an increased rendering time for a test application. Chonka et al. [37] perform XML-based attacks in order to evaluate their service-oriented traceback architecture. Beyond, distributed denial-of-service attacks striking software-defined networks are known [38].

Our attacks differ with respect to three attributes from the available approaches: (1) Our side channel does not only reveal a victim's network bandwidth, but rather extracts the (more accurate) parameters *credit rate* and *time window* of the applied rate limiting. (2) Both – our side channels as well as the denial-of-service attack – are the very first that exploit the asymmetric behavior of Xen's rate limiting and its susceptibility to burst transmissions in order to gain information about the victim or negatively impact the latter's availability. (3) In contrast to side channels from related work measuring network bandwidth, the adversary neither has to control the measured virtual instance nor be co-resident to the victim instance; the latter holds for the denial-of-service attack, too.

## 7 Discussion

Throttling network bandwidth hinders a virtual machine from claiming all available resources and cutting off supply to neighbor machines. Such rate limits have always been considered as means of security against denial-of-service attacks, but not as an attack vector themselves. Notwithstanding, our work conveys by the example of the popular Xen hypervisor that (1) configuration parameters of rate limits are easily gained through a side channel and that (2) novel denial-of-service attacks exploiting (allegedly protective) rate limits are feasible. In comparison to traditional bandwidth consumption attacks, our denial-of-service attack shows a peculiarity with respect to the point of consumption. A traditional denial-of-service attack jams the virtual machine's *inbound* link, exploiting rate limiting functionality as shown in the paper at hand causes jam on the *outbound* link. In the first case, a virtual machine would not receive any further traffic and might suspect irregularities on the network. In the latter case, it would still obtain requests and would be (mostly) unaware that responses are stuck in the hypervisor. Digging its own grave, it would even answer incoming requests strengthening the attack. Xen's unilateral bandwidth limits (not throttling inbound traffic) is an additional blessing as requests from the adversary are reliably forwarded to the victim. This means that there is in principle no need for traffic amplification; but admittedly, the attack is more likely to succeed with some sort of traffic amplification, e.g., when striking over the Internet with much lower bandwidth.

Our side channel allows to infer all configuration parameters of Xen's rate limits – the rate and the window time. These parameters are sensitive insofar as they allow a more detailed look on network characteristics than conventional means of bandwidth measuring, and serve various attacks. On the one hand, this enables an adversary to plan an attack, e.g., our denial-of-service-attack, more accurately. Further, an adversary once knowing these parameters of a virtual machine would be able to glean the latter's networking behavior; but also benign customers might use the side channel to check compliance of the configuration with their service contract. On the other hand, the side channel may also serve as a way to identify the underlying hypervisor of a virtual machine as Xen. By now, however, it remains unclear whether burst transmissions are just an issue of Xen, or also applicable to other hypervisors and container solutions.

In dependence on the outcome, burst transmissions themselves would imply the use of Xen; otherwise, fingerprinting would have to focus on more subtle differences in bursts among different hypervisors; we plan respective investigations for future work. Beyond, the side channel has potential to be developed further into a covert channel. A limitation is however given by network jitter as an adversary depends on clear distinction between subsequent time slots. This limitation however is only valid for the side channel, not for the denial-of-service attack.

Our denial-of-service attack causes latencies of almost 14s, and packet drops of up to 88.3%. Service degradation is generally undesired, for example, it decreases interactivity [39]; but there are also scenarios beyond the obvious that we would like to highlight. First, virtual machines are remotely synchronized by means of a synchronization protocol like Network Time Protocol (NTP) for purposes of time measurements [40,41]. However, the synchronization algorithm easily looses its stability in case of variable path delays [42], and these delays are heavily increased for a certain period before going back to normal by our attack; further, synchronization is prone to path asymmetry [42], and this asymmetry is also exacerbated by our attack. Synchronization errors are already in the milliseconds in presence of moderate CPU load [41], and will become significantly worse in presence of our attack making accurate time measurements in the clouds a nightmare. Second, temporal lensing was lately introduced as a way of attacking [43]; thereby an adversary performs a reflective denial-of-service attack that concentrates into a single (short, but high-bandwidth) pulse striking the victim. This is achieved by using reflectors with different attack path latencies, i.e., requesting reflectors with long paths before those with shorter ones, with the goal that all replies reach the victim simultaneously. The more reflectors with higher latencies are found, the more the adversary is able to funnel. The longest path latency found by the authors was 800 ms. In case such a reflector resided on a virtual machine, its responses could be delayed up to almost 14s by hitting this virtual machine with our TCP attack. This approach would significantly increase temporal lensing's power of impact by providing seamlessly controllable reflectors.

Finally, our results provide a further explanation to cloud phenomena: [44] measured TCP and UDP performance in the *Amazon EC2* cloud that is known to use the Xen hypervisor, and identified regular bandwidth drops[9]. They seem to occur roughly every 50 ms, and might be a consequence of rate limits. This observation might further be an indicator that rate limits were (and possibly still are) deployed at this major cloud provider; but Rackspace – another public cloud provider also using Xen – might also throttle virtual machines this way as they claim that only outbound traffic is limited [45]. Parenthetically, public cloud providers charge only outbound traffic while inbound remains free. This implies that our denial-of-service attack does not only impact a virtual machine's availability, but also costs the owner actual money and could be used to economically harm somebody.

[9] See Fig. 5 in [44].

In consequence, mitigation is of utter importance; however, none of the following suggestions fully prevents our attacks. (1) Throttling inbound traffic as well would only prevent non-amplifying attacks, but might negatively impact a host's availability. However, providers could choose to apply such limits only in the presence of an attack – provided that adequate detection mechanisms are prevalent. (2) A modification of the credit-based scheduler enabling short spikes (by spending previously saved credits) would increase the effort to overwhelm rate limits and buffers for the adversary. (3) Decreasing the *time window* $t$ makes our side channel more prone to jitter (and thus prevent it) as the time slots cannot be clearly distinguished anymore, but would have a negative impact on performance. Alternatively, the algorithm might be modified in order to be less deterministic, e.g., by randomizing the time window.

## 8 Conclusion

Rate limits are known to guarantee fair bandwidth distribution and to prevent denial-of-service attacks among virtual machines on the same *Xen* hypervisor; but our work shows that rate limits themselves become a vector for externally-launched attacks. The underlying reasons are *Xen*s unidirectional rate limits throttling outbound traffic only, and its susceptibility to burst transmissions. In the paper at hand, we propose two distinct attacks exploiting rate limits. Our side channel reveals configuration parameters that are related to rate limit functionality; our denial-of-service attack causes up to 13.8s of packet delay or up to 88.3 % packet drops. Beyond ordinary service degradation, these latencies may heavily destabilize time synchronization in clouds due to increased path asymmetry and path variability; but may also strengthen temporal lensing attacks due to providing reflectors with controllable path latency. There is indication that popular cloud providers like *Amazon EC2* or *Rackspace* are using *Xen*'s rate limits; thus, a large number of hosts remains conceivably vulnerable.

**Acknowledgments.** The authors thank Peter Wurzinger, and Adrian Dabrowski for many fruitful discussions; Rob Sherwood for sharing the original implementation of optimistic acknowledging and David Lobmaier for reimplementing it with respect to current TCP implementations. Further, the authors are grateful to our reviewers for their comments, especially on the aspect of mitigation.

This research was funded by P 842485 and COMET K1, both FFG - Austrian Research Promotion Agency.

## References

1. With an eye on Russia, Estonia seeks security in cloud computing, December 2015. http://www.firstpost.com/business/with-an-eye-on-russia-estonia-seeks-security-in-cloud-computing-2535650.html
2. Dou, E., Barr, A.: U.S. Cloud Providers Face Backlash From China's Censors, March 2015. http://www.wsj.com/articles/u-s-cloud-providers-face-backlash-from-chinas-censors-1426541126

3. Khan, A., Othman, M., Madani, S., Khan, S.: A survey of mobile cloud computing application models. IEEE Commun. Surv. Tutorials **16**(1), 393–413 (2014)
4. Ericsson, Connected Vehicle Cloud Under The Hood
5. Gilpin, L.: How The Cloud Is Revolutionizing Healthcare, December 2015. http://www.forbes.com/sites/lyndseygilpin/2015/12/01/how-the-cloud-is-revolutionizing-healthcare/
6. Departement of Commerce, 2015 Top Markets Report Cloud Computing - A Market Assessment Tool for U.S. Exporterts (2015)
7. FCA paves the way for cloud computing in UK financial services, November 2015. http://www.out-law.com/en/articles/2015/november/fca-paves-the-way-for-cloud-computing-in-uk-financial-services/
8. Finnegan, M.: How Tesco Bank has adopted AWS cloud as 'business as usual' in eight months, November 2015. http://www.computerworlduk.com/cloud-computing/how-tesco-bank-has-adopted-aws-cloud-as-business-as-usual-in-eight-months-3629767/
9. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP 2003, pp. 164–177 (2003)
10. Mather, T., Kumaraswamy, S., Latif, S.: Cloud security and privacy: an enterprise perspective on risks and compliance. O'Reilly Media Inc., Sebastopol (2009)
11. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: 16th ACM Conference on Computer and Communications Security, pp. 199–212 (2009)
12. Herzberg, A., Shulman, H., Ullrich, J., Weippl, E.: Cloudoscopy: services discovery and topology mapping. In: ACM Cloud Computing Security Workshop, pp. 113–122 (2013)
13. Okamura, K., Oyama, Y.: Load-based covert channels between Xen virtual machines. In: Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 173–180 (2010)
14. Xu, Y., Bailey, M., Jahanian, F., Joshi, K., Hiltunen, M., Schlichting, R.: An exploration of l2 cache covert channels in virtualized environments. In: Proceedings of the 2011 ACM Workshop on Cloud Computing Security Workshop, pp. 29–40 (2011)
15. Varadarajan, V., Kooburat, T., Farley, B., Ristenpart, T., Swift, M.M.: Resource-freeing attacks: improve your cloud performance (at your neighbor's expense). In: ACM Conference on Computer and Communications Security, pp. 281–292 (2012)
16. Bates, A., Mood, B., Pletcher, J., Pruse, H., Valafar, M., Butler, K.: Detecting co-residency with active traffic analysis techniques. In: ACM Cloud Computing Security Workshop, pp. 1–12 (2012)
17. Bates, A., Mood, B., Pletcher, J., Pruse, H., Valafar, M., Butler, K.: On detecting co-resident cloud instances using network flow watermarking techniques. Int. J. Inf. Secur. **13**(2), 171–189 (2014)
18. redhat, 33.10.Limit network bandwidth for a Xen guest (2016). https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/5/html/Virtualization/sect-Virtualization-Tips_and_tricks-Limit_network_bandwidth_for_a_Xen_guest.html
19. Adamczyk, B., Chydzinski, A.: On the performance isolation across virtual network adapters in Xen. In: Proceedings of the 2nd International Conference Cloud Comput. GRIDs Virtual, CLOUD COMPUTING 2011, pp. 222–227 (2011)

20. Mei, L., Lv, X.: Optimization of network bandwidth allocation in Xen. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), pp. 1558–1566, August 2015
21. Li, C., Xi, S., Lu, C., Gill, C.D., Guerin, R.: Prioritizing soft real-time network traffic in virtualized hosts based on Xen. In: 21st IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 145–156, April 2015
22. Rossow, C.: Amplification hell: revisiting network protocols for DDoS abuse. In: Network and Distributed System Security Symposium (NDSS) (2014)
23. Sherwood, R., Bhattacharjee, B., Braud, R.: Misbehaving TCP receivers can cause internet-wide congestion collapse. In: Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS), pp. 383–392 (2005)
24. Xu, Y., Musgrave, Z., Noble, B., Bailey, M.: Bobtail: avoiding long tails in the cloud. In: Presented as Part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2013), pp. 329–341 (2013)
25. Xu, Y., Bailey, M., Noble, B., Jahanian, F.: Small is better: avoiding latency traps in virtualized data centers. In: Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC 2013 (2013)
26. Varadarajan, V., Ristenpart, T., Swift, M.: Scheduler-based defenses against Cross-VM side-channels. In: 23rd USENIX Security Symposium (USENIX Security 2014), pp. 687–702, August 2014
27. Bowers, K.D., van Dijk, M., Juels, A., Oprea, A., Rivest, R.L.: How to tell if your cloud files are vulnerable to drive crashes. In: 18th ACM Conference on Computer and Communications Security, pp. 501–514 (2011)
28. Benson, K., Dowsley, R., Shacham, H.: Do you know where your cloud files are? In: 3rd ACM Cloud Computing Security Workshop, pp. 73–82 (2011)
29. Kadloor, S., Gong, X., Kiyavash, N., Tezcan, T., Borisov, N.: Low-cost side channel remote traffic analysis attack in packet networks. In: IEEE International Conference on Communications (ICC), pp. 1–5, May 2010
30. Kadloor, S., Kiyavash, N., Venkitasubramaniam, P.: Mitigating timing based information leakage in shared schedulers. In: IEEE INFOCOM, pp. 1044–1052 (2012)
31. Alarifi, S., Wolthusen, S.D.: Robust coordination of cloud-internal denial of service attacks. In: 2013 Third International Conference on Cloud and Green Computing (CGC), pp. 135–142, September 2013
32. Ficco, M., Rak, M.: Stealthy denial of service strategy in cloud computing. IEEE Trans. Cloud Comput. **3**(1), 80–94 (2015)
33. Liu, H.: A new form of DOS attack in a cloud and its avoidance mechanism. In: Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop, CCSW 2010, pp. 65–76 (2010)
34. Shea, R., Liu, J.: Understanding the impact of denial of service attacks on virtual machines. In: Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service, IWQoS 2012, pp. 27:1–27:9 (2012)
35. Shea, R., Liu, J.: Performance of virtual machines under networked denial of service attacks: experiments and analysis. IEEE Syst. J. **7**(2), 335–345 (2013)
36. Ferriman, B., Hamed, T., Mahmoud, Q.H.: Storming the cloud: a look at denial of service in the Google App Engine. In: 2015 International Conference on Computing, Networking and Communications (ICNC), pp. 363–368, February 2015
37. Chonka, A., Xiang, Y., Zhou, W., Bonti, A.: Cloud security defence to protect cloud computing against HTTP-DoS and XMLAQ2DoS attacks. J. Netw. Comput. Appl. 34(4), 1097–1107 (2011)

38. Yan, Q., Yu, F.R.: Distributed denial of service attacks in software-defined networking with cloud computing. IEEE Commun. Mag. **53**(4), 52–59 (2015)
39. Sanaei, Z., Abolfazli, S., Gani, A., Buyya, R.: Heterogeneity in mobile cloud computing: Taxonomy and open challenges. IEEE Commun. Surv. Tutorials **16**(1), 369–392 (2014)
40. Lampe, U., Kieselmann, M., Miede, A., Zöller, S., Steinmetz, R.: A tale of millis and nanos: time measurements in virtual and physical machines. In: Lau, K.-K., Lamersdorf, W., Pimentel, E. (eds.) ESOCC 2013. LNCS, vol. 8135, pp. 172–179. Springer, Heidelberg (2013)
41. Broomhead, T., Cremean, L., Ridoux, J., Veitch, D.: Virtualize everything but time. In: USENIX Symposium on Operating Systems Design and Implementation (OSDI 2010) (2010)
42. Ullmann, M., Vogeler, M.: Delay attacks: implication on NTP and PTP time synchronization. In: 2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, October 2009
43. Rasti, R., Murthy, M., Weaver, N., Paxson, V.: Temporal lensing and its application in pulsing denial-of-service attacks. In: 2015 IEEE Symposium on Security and Privacy, pp. 187–198, May 2015
44. Wang, G., Ng, T.S.E.: The impact of virtualization on network performance of amazon EC2 data center. In: INFOCOM, 2010 Proceedings IEEE, pp. 1–9, March 2010
45. Rackspace, Pricing (2016). https://www.rackspace.com/cloud/servers/pricing