The Impact of Hypervisor Scheduling on Compromising Virtualized Environments

Tsvetoslava Vateva-Gurova and Neeraj Suri
Dept. of CS, TU Darmstadt, Germany
Email: vateva,suri@deeds.informatik.tu-darmstadt.de

Avi Mendelson
Dept. of CS, Technion, Israel
Email:avi.mendelson@tce.technion.ac.il

***Abstract*—A virtualized environment (VE) is expected to provide secure logical isolation across the co-located tenants encapsulated in the virtual machines. In particular the VE should prevent covert-channels exploitation stemming from the usage of shared resources. However, as sophisticated covert- and side-channel attacks exist, the logical isolation in a VE is often considered insufficient to raise concerns about the security in VEs e.g., the Cloud. Technically, the actual feasibility of such attacks strongly depends on the specific context of the execution environment and the resource allocation schemas used in the virtualization solution. Addressing these VE aspects, we detail the effect of scheduling parameters on the noise (affecting the information leakage) in the covert-channel and empirically validate the impact on the feasibility of covert-channel attacks, using a real VE.**

I. Introduction

Virtualization is the underlying technology in Cloud computing where multiple tenants with different security requirements share resources. The Virtual Machine Monitor (VMM) along with the physically co-located virtual machines (VMs) are the main entities characterizing a virtualized environment (VE). The VMM, also known as the hypervisor, provides low-level abstraction by decoupling the operating system from the hardware state, and enables multiplexing multiple tenants with varied security and functional demands encapsulated in VMs onto a single physical machine. These VMs are expected to provide securely isolated resources to the encapsulated tenants [1], where VE isolation is defined as "the inability of one VM to gain information regarding the co-located VMs, as well as to affect or intervene with their operation" in [2]. On this definition, the paper focuses on the confidentiality aspect of isolation – namely the presumed impossibility to leak information beyond the boundaries of a VM despite the shared usage of resources in a VE. However, covert-channel[1] attacks (CCA) and side-channel attacks (SCA) can be applied to compromise the confidentiality in a VE. While CCAs misuse the covert channel to enable communication across two adversaries, an attacker for SCAs abuses the covert channel to observe the victims operations.

[1]A covert-channel is a communication channel stemming from the usage of shared resources that was not supposed to be used as such by the designers of the system. The existence of a covert-channel is a prerequisite for the successful conduct of side-channel and covert-channel attacks.

A. Problem Statement

Various shared resources have the potential to be used as covert-channels in a VE, though the cache is the one considered most practical. While the feasibility of cache-based CCAs and SCAs, as well as their practical relevance for a VE, has been demonstrated in the security community [3]–[7], little work exists to investigate how the environment affects their feasibility. [2] recently highlighted that the VMM scheduling policy, among other factors, plays a role for compromising the isolation in a VE through a SCA or a CCA. The results shown in [8] reinforce this statement by providing a defense mechanism that alters one of the VMM scheduling parameters to prevent the VE from the cache-based side-channel exploits. However, no work has systematically studied the effect of the multitude of VMM scheduling parameters on the feasibility of cache-based CCAs or SCAs in a VE, and no work provides guidelines on what is the most secure configuration of the hypervisor's scheduler considering the covert-channel attack threat. The lack of such research impedes the better understanding of the threat related to SCAs and CCAs in the VEs as well as the assessment of the strength of the isolation in terms of confidentiality provided by a specific VE.

B. Contributions

On this background, we investigate the effect the VMM scheduling has on the feasibility of covert-channel attacks. For this, we study a set of scheduling parameters, and consider how changing the value of each parameter can affect the exploitability of the cache-based covert-channel in a VE empirically. The environment, and more specifically the VMM scheduling, affects the cache-based covert channel in the same way for both SCAs and CCAs[2]. In addition, a CCA is characterized by two cooperating attackers. In this case the feasibility of the exploit depends less on the capabilities of the attackers than in the case of SCAs which are characterized by an attacker who tries to steal information from a victim through the covert-channel. Thus, we can make stronger statements about the effect of the VMM scheduler on the covert-channel for CCAs than SCAs.

[2]As the effect on CCA/SCA is similar, the terms covert-channel and side-channel are interchangeably used.

Hence, as contributions we: (i) model the cover channel exploit with respect to the way information is conveyed, (ii) identify scheduling parameters that can affect the covert channel, and (iii) empirically ascertain their effect on the feasibility of CCAs by deploying a CCA in a real VE. Our goal is not to propose a novel to the VE exploit, but to investigate under which conditions, e.g., scheduling configuration, the cache-based CCAs are more or less feasible in a VE. Overall, we provide guidance on which scheduler configuration tends to be more secure in regard to the CCAs. The exact feasibility of CCAs naturally depends on the specific implementation.

The paper is structured as follows. Section II investigates the state-of-the-art in the area of SCAs and CCAs and the impact of scheduling on the feasibility of CCA/SCA. Section III describes the system and attacker model for the covert-channel based communication. Section IV discusses the scheduling parameters and their roles in the VE, Section V details the experimental results for their CCA impact.

II. Related work

The relevance of SCAs and CCAs and their practical applicability to the VE has been recognized in the security community [3]–[6], [9]. Much effort has been devoted to enhancing the VE isolation and confidentiality by researching preventive mechanisms against side- and covert-channel attacks [10]–[14]. Despite that Mowery et al. expressed doubts about the feasibility of SCAs on a specific architecture in [15] after an unsuccessful attempt to exploit the system using the cache as a side-channel. According to the work described in [2], various factors characterizing the execution environment such as scheduling policies can impact the feasibility of side-channel exploits by making them harder or easier. However, the effect of these factors has not been extensively studied. With this context, we overview the state-of-the-art in terms of the effect of the hypervisor's scheduler on the feasibility of covert- and side-channel exploits.

The relevance of the scheduling policy for SCAs and CCAs.: In 1992 Hu observed that the scheduling policy plays a major role in exploiting hardware timing covert-channels based on shared CPU usage. Hu addressed the covert-channel threat by proposing a scheduling scheme [16] for the VAX security kernel where each process is characterized by a 64-category secrecy class. A time slot list is used to determine the CPU time allocation and the initial execution order for the processes. To eliminate the covert-channel, the scheduler does not base its decision about the execution order on the readiness of the processes to be executed. In that way alternating execution of the malicious processes which is a prerequisite for the cache-based communication cannot be guaranteed anymore. Although Hu's work is focused on the VAX security kernel, it raised awareness about the relationship between the scheduling scheme and the hardware timing covert-channel exploits.

Recently, Varadarajan et al. addressed the SCA threat by altering the value of one of the VMM's parameters and studying its impact both on the performance and the side-channel exploits [8] while stressing that the effect of the scheduling on the isolation in a VM is not studied extensively in the community. They demonstrated that by altering the rate limit scheduling parameter they can provide a minimum run time guarantee for a particular VM. As a consequence the granularity of the observations the attacker can do is affected. Namely the possibility for frequent preemptions is one of the vulnerabilities that is exploited in the attack described in [5], which leads to the successful execution of a SCA that uses the L1 instruction cache. The work described in [8] actually investigates the impact of one of the VMM's scheduling parameters on the side-channel exploits. *In contrast, our aim is to consider the influence of the spectrum of scheduling parameters and to provide guidelines on the most secure configuration in regard to CCAs in a VE. Unlike [8], we specifically consider CCA versus SCA, as the covert-channel exploits depend less on the capabilities of the attacker and the direct impact of the role of scheduling is discernible.*

In [12] the authors reported the role of scheduling algorithms for the successful exploit of cache-based covert-channels highlighting that deterministic information flow control systems are vulnerable to cache-based timing channels if time-based scheduling is used. To eliminate the covert-channel, the authors propose an instruction-based scheduling scheme. Although the work does not address VE's, the observation that the scheduling scheme can be used as a preventive mechanism against cache-based CCAs raises awareness of the role the scheduler plays in security.

A VMM scheduler vulnerability, though only indirectly related to the covert-channel exploits, is reported also in [17] by Zhou et al. It can be exploited by an attacker residing in a VM to consume more CPU time on the Amazon EC2 Cloud computing infrastructure. Although this work is not directly related to the feasibility of covert-channel attacks and the impact of the configuration of the hypervisor scheduler on that, it demonstrates that the choice of the scheduling mechanism and its parameters is important not only from performance but also from security perspective.

III. System and Attacker Model

A. System Model

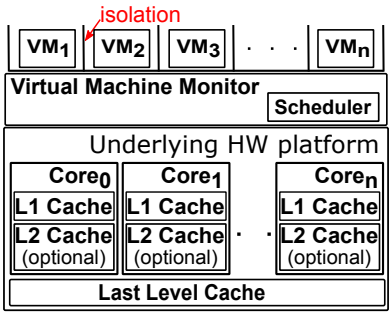

Fig. 1. Isolation in a virtualized environment.

We investigate the impact of the hypervisor scheduler on the cache-based covert-channel attacks that might compromise the confidentiality and with that the isolation in a VE. Our system model is thus focused on the VE, consisting of the VMM and the co-located VMs. The isolation, as shown in Fig. 1, is considered with respect to the logical boundaries of the VMs. Except for the hypervisor scheduler, which is depicted as a part of the VMM, we abstract the role of the other VMM's components and functions. As our focus is on cache-based CCAs, we also model the hierarchy of caches, as can be seen in Fig. 1. For the purpose of better understanding, we highlight the concept of cache set associativity. The cache is divided into cache lines, and the cache lines might be grouped into cache sets depending on the associativity level of the cache. N-way associative cache means that N cache lines are grouped into the same set, and each memory address is mapped to exactly one set, but might be mapped to any of the N cache lines within that set.

Depending on the architecture, each level of cache might be private per core or shared among all CPU cores. Usually the Last Level Cache (LLC) is shared among the cores, and the First Level Cache (L1) is private per core. Some architectures have an L2 cache which can be shared or private. L1 is the smallest and also the fastest cache while the LLC is larger and slower, but still faster than the main memory. If data is needed from the main memory, the processor looks for it in the L1 cache first, then in the other levels of cache, and if it is in none of the caches, it is fetched (copied) from the main memory into the cache and then used. If the data is not in the cache (cache miss), accessing it takes longer. In case of a cache hit (the data is in the cache), the access time is shorter. These timing differences in accessing data enable attackers to use the cache as a covert-channel. Theoretically, each cache level can be used as a covert-channel, but due to the different properties of the caches, the L1-based covert-channel exhibits different properties compared to the LLC-based channel for instance in terms of bandwidth or error rate of the transmitted data. The subsequent paragraphs detail how a cache-based CCA works.

B. Attacker Model

To illustrate how a cache-based CCA works in a VE, we consider two distinct co-located VMs: VM_1 and VM_2, and refer them to as the *sender* and the *receiver*, respectively. We assume that they have no other means of communication except for by using the cache as a covert-channel. As depicted in Fig. 2, VM_1 aims to transmit data to VM_2 by encoding each bit of information as a predefined *cache access pattern*, i.e., by accessing specific cache sets indirectly. For this purpose, the two cooperating VMs have to agree on the meaning of the varied cache access patterns in advance. To infer the cache access pattern and decode the encoded bit, the receiver makes use of the timing difference for fetching data from the main memory (i.e., cache miss) and from the cache (i.e., cache hit).

To perform the attack, the attackers divide the cache sets logically into two parts where each part of the cache corresponds to either bit 1 or bit 0, as shown in Fig. 2, and thus both the receiver and the sender are aware of the meaning of the specific sets. As an initialization step of the attack the receiver accesses both parts of the cache. In that way, the receiver fetches certain data into the cache and a repetitive access to the same data will result in a cache hit and will be fast. If the receiver's data has been evicted from the cache, the repetitive access to it will be slower, as it has to be fetched again from the main memory. Having this in mind, when the sender wants to encode bit 1, he accesses the part of the cache that corresponds to 1 according to the cache access pattern he has agreed on with the receiver. When the sender wants to encode bit 0, he accesses the other part of the cache. Once the sender has encoded the bit of information he wants to transmit, the receiver measures the time for accessing each part of the cache again. The part which corresponds to the longer access time is the part which has been accessed by the sender. In that way, by timing the cache accesses the receiver can infer whether the received bit is 1 or 0. Note that other access patterns can also be successfully applied e.g., the one used in [16].

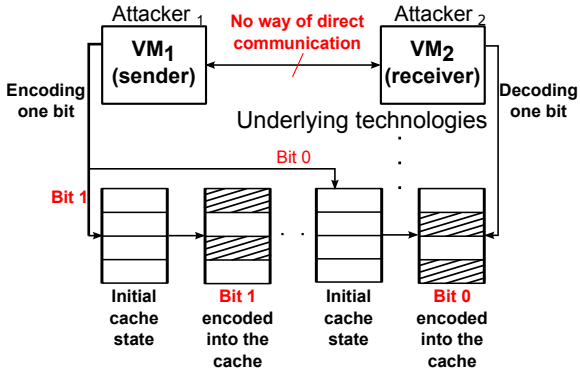

Fig. 2. Transmitting one bit of information.

IV. VMM Scheduler

The VMM scheduler is responsible for scheduling VMs CPUs (VCPUs) on the physical CPUs of the underlying hardware. Moreover, for its resource allocation decisions, the VMM scheduler has to consider not only VMs' performance requirements, but also possible security violations that might be a consequence of the scheduling decisions. The objective of this paper is to investigate how the configuration of the VMM scheduler (the values of the various scheduling parameters) affects the feasibility of CCAs in a VE with regard to the successful transmission of bits through the covert communication channel. Various scheduling algorithms are used by the VMMs e.g., Simple Earliest Deadline First (SEDF) [18], Completely Fair Scheduler (CFS) [19], [20], Borrowed Virtual Time (BVT) [21], Credit scheduler [22], etc. To assess the impact of the scheduling configuration on the covert-channel, we consider fair-share schedulers which represent a widely used solution for VMMs such as KVM [20] and Xen [23]. We

TABLE I
VMM SCHEDULING PARAMETERS.

Parameter	Function
Load balancing	Determines whether automatic load balancing is used
Weight	Denotes the relative CPU allocation among the VMs
Cap	Imposes an absolute limit on max amount of CPU a VM can consume
Time slice	Determines max time period a VM is allowed to run
Rate Limiting	Guarantees min amount of time a VM can run without being preempted

focus on the publicly available Credit scheduler used in Xen [18], [22] —this scheduler is widely used in real-world Cloud scenarios and provides most of the scheduling parameters that are also relevant for many other schedulers used in real-world VEs. It is worth mentioning that parameters with the same conceptual function are often known under different names for different schedulers.

A. Credit Scheduler

The Credit scheduler is a proportional fair-share CPU scheduler. Pursuing its main goal to achieve fairness in terms of resources allocation, the Credit scheduler assigns to each VCPU a certain number of *credits* [17], [24]. With the consumption of CPU time, the VCPU "consumes" credits, thereby reducing the number of its available credits. The scheduler keeps track of the VCPUs that have exceeded their credits. The absolute value of the available credits is not of importance, but only whether a VCPU has exceeded its credits. If a VCPU has run out of credits, its priority is set to "OVER" and it only gets scheduled when the VCPUs with available credits have finished executing. If a VCPU has credits left, its priority is set to "UNDER" and the VCPU can get scheduled until its credits are depleted. Periodically, the VCPUs are given new credits and their available credits are updated.

B. Scheduling Parameters

The scheduling parameters we are considering are load balancing, weight, cap, time slice and rate limiting. They are part of the Xen's Credit scheduler, but are also relevant for other scheduling schemes, though known under different names. Table I gives an overview of their functions.

Load balancing: Among the advantages of the Credit scheduler is its ability to automatically balance the load between the available CPUs. If all the VCPUs running on a certain CPU have exceeded their credits, the scheduler checks whether there is a VCPU with available credits waiting to be scheduled. The system administrator still has the choice to use automatic load balancing, or to manually balance the load by assigning VCPUs to CPUs. The latter option, also called CPU pinning or fixing the CPU affinity, is used as a restriction to limit the CPU cores a VCPU can use. The load balancing or CPU pinning parameter is specified per VCPU.

Weight: The *weight*, also known as shares, is assigned per VM. It denotes the relative CPU allocation time of a VM with regard to the other VMs. A VM with a weight twice as high as the weight of another VM receives twice as much CPU time as the other VM. The scheduler uses the weight value for the calculation of the credits that a VCPU gets.

Cap: The Credit scheduler supports both work-conserving and non-work-conserving modes. This feature is controlled by the *cap* parameter, also known as limit. It imposes a limit on the maximum amount of CPU a VM can consume. A VM with a cap 0 is not capped. This corresponds to scheduler's work-conserving mode in which the CPU is idle only if no VCPU is waiting to be scheduled. If the cap is set to a non-zero value which represents the absolute limit on the amount of CPU a VM can use even if more processing resources are available, the scheduler is in non-work-conserving mode. The cap limits the consumption of a CPU cycles for the VCPUs and with that might increase the time for completion of a task. The cap value is expressed in percentage per VM.

Time slice: The time slice, also referred to as quantum, determines the maximum period of time a VM is allowed to run before being interrupted by the scheduler. If the execution time of a VM exceeds the time slice, the VM gets preempted and another VM is scheduled. The time slice represents the maximum time a VM is allowed to run, but does not guarantee that a VM will run without being interrupted that long. The time slice parameter is set once for all the VMs. For instance, if it is set to 30 ms, each VM gets preempted every 30 ms, and the credits of all runnable VMs are recalculated. Thus, the value of the time slice usually has performance implications —for latency-sensitive workloads having a long time slice can be devastating, whereas for computationally intensive workloads it can have positive effect.

Rate limiting: The rate limiting guarantees minimum amount of time a VM can run without being preempted. This can be disabled by setting its value to 0. Similar to the time slice, the rate limiting value can have impact on system's performance depending on the workload requirements.

V. EMPIRICAL EVALUATION

We experimentally study whether certain scheduling configurations result in a noisier covert-channel and, consequently, in a more secure VE. For each of the scheduling parameters introduced in Section IV, we assess how a change in the value affects the CCAs in terms of bandwidth and error rate.

A. Experimental Setup and Implementation Details

We use two physically co-located VMs: VM_1 and VM_2. A sender process running in VM_1 transmits information to the receiver process running in VM_2 via the L1 cache. In order to assess the influence of the scheduling parameters on the feasibility of CCAs, we conducted experiments considering both minimal and higher third-party workload. For most of the cases, the scheduling's effect is observable without third-party workload. This helps us draw meaningful conclusions on the impact of the scheduling without the need to filter

out the noise caused by third parties. However, in other cases, the effect of the scheduling is better shown if higher load is present in the VE. From a practical point of view, we consider selectively chosen scheduling configurations that are representative of broad scheduling configurations and are expected to result in different levels of noise in the covert-channel. The empirical assessment is done in a real VE. Please note that the implementation details described in this section are specific to our setup and the use of other setup requires adjustments of the implementation.

Hardware: The experiments are conducted on commodity hardware: Intel®Quad Core™ i7-4770 CPU@3.4GHz. Each CPU core has 8-way associative L1 cache with a data cache size of 32KiB. Each cache line is 64 Bytes long. As it has been demonstrated that advanced CPU features such as SMT might enable an easier CCA [25], we have disabled the Intel® Hyper-Threading Technology.

Virtualization and Scheduler: We use Xen 4.3 hypervisor [26] as a virtualization solution. In Xen terminology all the VMs are referred to as guests. Apart from the hypervisor, Xen uses a guest domain with special privileges, called Dom0 which can be seen as a service console. It is used to create or destroy other guests, to adjust the scheduling parameters, etc.

Approach and Algorithm: In our CCA the sender and receiver exploit the timing difference for accessing data from the L1 cache and the lower cache levels or the main memory (cf., Section III). As they cannot access L1 directly, they allocate buffers of integers of the size of the L1 (32 KB) which are also aligned by the cache line size (64 Bytes). Through its buffer, the sender evicts one cache line from every even cache set to transmit bit `0` and one cache line from each odd cache set to transmit bit `1`, by accessing those parts of the buffer that are mapped to the respective cache sets. Then, the receiver determines which parts of the cache have been evicted by measuring the time for accessing one cache line for each of the odd and even cache sets to decode sender's access pattern.

Our L1 is physically tagged and virtually indexed. As the index bits of the memory address indicate the cache set which the data should be mapped to, the attackers can use the virtual addresses of their respective buffers to access specific cache sets. By accessing one integer (4 Bytes) from the allocated buffer, the receiver fetches a 64 Byte chunk (cache line) of data from main memory which corresponds to 16 integers into L1. This scheme allows us to speculate that accessing the receiver's buffer with consecutive indices will result in cache hits even if the sender has previously evicted the respective cache lines. In this case, if the receiver should measure 16 cache misses, most probably he will measure 1 cache miss and 15 cache hits. To diminish this effect, our receiver accesses only one cache line per set and "moves" to the next cache set.

As the difference between a L1 cache miss and hit is ~ 6 clock cycles, for a single cache line the receiver can hardly measure it when decoding the sender's cache access pattern. To overcome this problem, the receiver accumulates clock cycle measurements for accessing one cache line in every even cache set and one cache line in all odd cache sets. Although this approach might not be adequately precise to work for specific implementations, it is sufficient for assessing and analyzing the effect of the scheduling configuration on the covert-channel.

The role of synchronization: A prerequisite for a successful attack is that the receiver times his access to the L1 directly after the sender has encoded one bit of information into it in the form of access pattern. If meanwhile another process or the sender access the L1 cache, the previously encoded bit of information is lost. In our implementation, the receiver and sender are synchronized using POSIX sockets to guarantee that the receiver will not access its buffer (and the cache) before the sender has finished encoding data into it. The implemented synchronization represents the ideal case for measuring the effect of the scheduling on the CCAs, as it eliminates the noise stemming from the incapabilities of the attackers to synchronize. The employed synchronization is just a tool that enables drawing valid conclusions about the scheduling's effect. Our experiments are comprised of a training and a testing phase.

A training phase is employed to determine how many consumed clock cycles represent bit `0` and how many clock cycles represent bit `1`. We notice that the difference in the clock cycle consumption for transmitting a `0` and a `1`, although observable, is not particularly large. This is expected, as we use L1 cache as the covert-channel and the penalty for a cache miss is moderate. Still, the difference is measurable, as we consider accumulated clock cycles consumed for accessing all the odd/even cache sets versus single set or line accesses.

In the testing phase, we measure the error rate and the bandwidth of the communication in regard to a predefined string of bits that the sender tries to encode for varied scheduling configurations. In each experiment the sender encodes `0`s for 1000 iterations. Then, `1`s are encoded for the remaining iterations. For each scheduling configuration we run altogether 1000 experiments with more than 10000 iterations per experiment. The measured error rate is the ratio of the number of correctly transmitted bits to all the conducted measurements on the receiver's side and is given in percentage. The bandwidth indicates the number of correctly transmitted bits per time unit (second). We do not apply any error correction mechanisms.

B. Evaluating the role of Scheduling Parameters

The remainder of the section describes the results of the empirical evaluation for the scheduling parameters (cf., Section IV). In each experimental set we vary a single parameter. All the presented results represent the time difference for accessing the even and the odd cache sets per single iteration explicitly for our setup. Due to space limitations, we show the detailed plots only for load balancing and weight.

*1) **Load balancing**:* In our VE Dom-0 is also competing for resources with the sender and receiver VMs, and we consider its role as a third-party workload when we study the impact

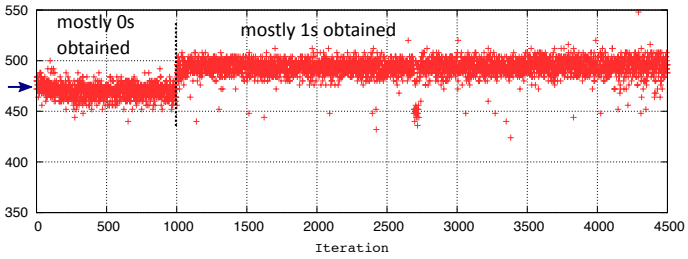

Fig. 3. The sender and the receiver are pinned to the same CPU core.

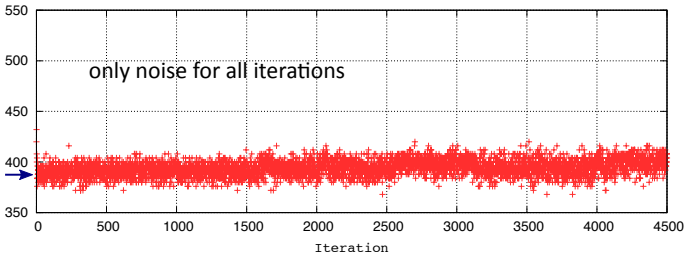

Fig. 4. The sender, receiver and Dom-0 are pinned to distinct CPU cores.

of the load balancing parameter on the covert-channel. We distinguish four relevant configurations.

- **C1:** Sender and receiver are pinned to the same CPU core and Dom-0 is pinned to another CPU core
- **C2:** Sender and receiver are automatically load balanced and Dom-0 is pinned
- **C3:** Sender, receiver and Dom-0 are pinned to the same CPU core
- **C4:** Sender and receiver are pinned to distinct cores

Expectations: We speculate that the load balancing parameter has the highest impact on the cache-based covert-channel and with that on the feasibility of CCAs. This holds in particular when private per core cache e.g., L1 is exploited for the attack. We expect that C4 represents the worst case from attacker's perspective. Logically, in this case the communication channel between the two cooperating attackers is unavailable which results in noise on the receiver's side. Analogously, we expect that the optimal case for attacker's communication is C1, as for the whole experiment duration the sender and the receiver share the same L1, whereas Dom-0 uses the L1 cache of another CPU core. Hence, the receiver and the sender communicate over a channel that is constantly available without interference from the operations of Dom-0.

Results and discussion: We measured the lowest error rate of 39% and the highest bandwidth of 90bps for C1. Fig. 3 clearly shows clock cycles increase after the 1000th iteration which remains for the rest of the experiment. It corresponds to the transition from sending `0`s during the first 1000 iterations to `1`s after that. In C4, on the other hand, this transition is not visible (cf., Fig. 4). The measured error rate in this case is almost 99%, and the bandwidth is below 1bps. All the receiver measurements consist of noise or data resulting from third-party processes but not from the actions of the sender, as the L1-based channel is unavailable in this case. This explains the lower clock cycle consumption indicated by the arrow in Fig. 4. The exhibited error rate for C3 is 72%, and the bandwidth is 9bps. As can be seen, C3 results in a less noisy communication than C4 and a more noisy communication than C1. This is logical as the covert-channel is available for the whole time of the experiments, but the sender and the receiver compete for the cache with a third-party (Dom-0). This explains the variation in the consumption of the clock cycles. C2 is almost as noisy as the worst case configuration (error rate of 97% and bandwidth below 1bps). In this case there is no guarantee that the attackers will share the communication channel even for a limited time due to the automatic load balancing. Actually, during the experiments they get automatically scheduled on distinct CPU cores. Due to space restriction, we omit the plots of the communication between the attackers for C2 and C3. To guarantee the availability of the channel for the rest of the experiments, we use C1 with regard to the load balancing.

*2) **Weight**:* The weight specifies the relative CPU allocation time of a VM with respect to the other VMs. To investigate its impact on the covert-channel, we distinguish four scheduling configurations.

- **C1:** Both weights are 256
- **C2:** Sender's weight is 512, receiver's weight is 256
- **C3:** Receiver's weight is 512, sender's weight is 256
- **C4:** Receiver's weight is 512, sender's weight is 256, Dom-0's weight is 256

Expectations: The optimal case for exploiting the covert-channel should be when the sender and receiver VMs are equally weighted. Then they have nearly equal chances to be scheduled. If the weight value is higher for the sender, the sender will be scheduled more frequently than the receiver, as the processes for the sender and the receiver are of roughly equal magnitude in terms of execution time. Then, the sender will encode bits that are likely not to be received. Analogously, if receiver's weight value is higher than sender's weight value, the chance that the receiver will be scheduled more frequently is higher. The receiver will try to decode the state of the cache that is a result of his own actions.

Results and Discussion: Due to the applied synchronization mechanism, we do not notice any significant difference in terms of error rate in C1-C3 when we consider minimal workload on the VMs. With its synchronization mechanism our implementation guarantees the alternating execution of the sender and the receiver processes even in the cases when one of them is scheduled more frequently. Both parties will have to wait for the confirmation of the other party that a bit has been encoded or decoded before being able to proceed. This is the reason why the error rate is only moderately affected in the case of minimal workload (38%, 42% and 44% for C1, C2 and C3, respectively). Fig. 5 represents the best case of equal weights for the attacker and the sender. The slightly increased error rate can be noticed in Fig. 6 - 7, as indicated by the boxes. The bandwidth of the covert-channel communication, on the other hand, is significantly lower for C2-C3 (32bps and

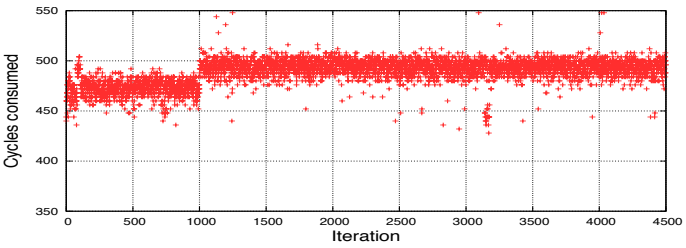

Fig. 5. The sender and the receiver exhibit the same weight values.

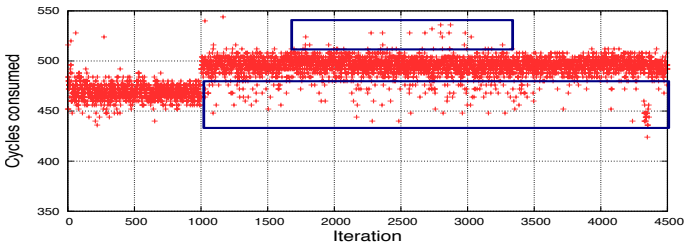

Fig. 6. The sender weight value is 512, the receiver weight value is 256.

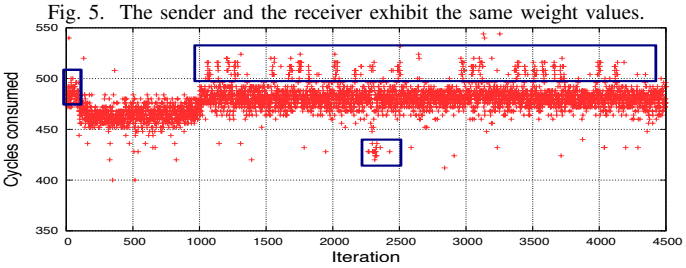

Fig. 7. The sender weight value is 256, the receiver weight value is 512.

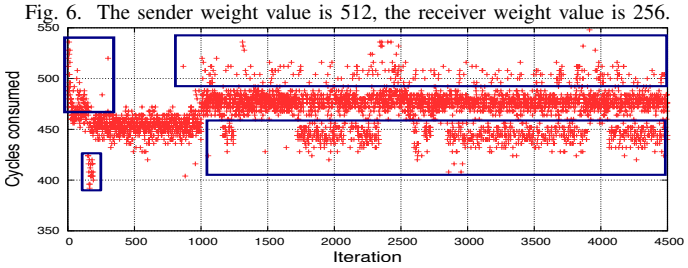

Fig. 8. The sender weight value is twice the weight of the receiver and Dom-0.

29bps, respectively) compared to C1 (89bps). In C2-C3 one party gets scheduled more frequently and has to wait for the the other one to complete its operations. Then the time spent without encoding/decoding a bit is significantly longer.

In the presence of third-party workloads (C4), the error rate also gets affected by the value of the weight parameter. If the weight value of the sender's (receiver's) VM is sufficiently higher than the weight value of the receiver's (sender's) VM, the third-party workload can get scheduled in between the attackers processes for send and receive and the error rate in the covert-channel will be increased. The receiver will decode after the sender has encoded a bit, but sender's cache access pattern will be overwritten by the access pattern of the third-party workload. We observed an increased error rate of 78% which is noticeable also from Fig. 8 for C4. As pointed out by the boxes in the figure, in this case we observe a much noisier communication than for configurations C1-C3.

*3) **Cap**:* The cap parameter can evoke VMs preemption if the VM has exceeded its CPU usage. We consider four relevant configurations.

- **C1:** The sender's and receiver's caps are 0
- **C2:** The sender and the receiver are capped with 40
- **C3:** The sender's cap is 1 and the receiver is not capped

Expectations: Our expectation is that attackers' optimal case is if none of them is capped (C1) or when one/both of them are capped but the cap value is sufficiently high so that they can finish encoding or decoding the data, without being interrupted. In such cases, we expect no influence on the feasibility of the covert-channel exploit considering both the bandwidth and error rate. However, we speculate that low cap values are disadvantageous for the attackers as they will get interrupted before having finished their tasks.

Results and Discussion: As expected, for C1 and C2 we measured an error rate of 38% and 41%, respectively, and obtained a communication bandwidth of 81bps for C1 and 78bps for C2. In these cases, the sender and the receiver are not interrupted before being able to complete their operations. As we speculated, when the sender gets only 1% of the CPU, the sender gets preempted before being able to encode a bit of information. This does not deteriorate the error rate due to the synchronization between the attackers, but results in a very low bandwidth(10bps) for the covert-channel communication.

*4) **Time slice**:* The time slice shows the maximum allowed time for a VM to run and is the same for all the VMs. Representative are the values listed below.

- **C1:** The lowest possible value - 1ms
- **C2:** The highest possible value - 1000ms
- **C3:** The default value - 30ms

Expectations: We speculate that the time slice affects the covert-channel in relation to sender's and receiver's execution times. If it is set to a very low value, i.e., lower than receiver's or sender's execution time, or both, this will result in interrupting the sender while encoding one bit or the receiver while decoding one bit. In both cases this hardens the covert-channel exploit. In contrast, if the time slice parameter is set to a higher value than the execution times of both the sender and receiver, it should not affect the CCA and the covert-channel. The latter case is optimal from attacker's perspective.

Results and Discussion: Analogous to the cases with the weight and the cap, due to the synchronization mechanism the error rate for all time slice configurations is similar. This changes, as in the case of the weight parameter, if third-party workload is involved. Also for the time slice, we have noticed that with the variation of its value, the number of transmitted bits within certain time interval varies. The bandwidth for C1 is 8bps whereas for C2 it is 79 bps. In the latter case no time is wasted for waiting for the other party to finish after being interrupted which increases the achieved bandwidth.

*5) **Rate limiting**:* This parameter guarantees minimum amount of time a VM can run without being preempted. To study its impact on CCAs' feasibility, we consider its lowest (100) and highest (500000) possible values.

Expectations: We expect that the impact of the rate limiting on the covert-channel depends on its relation to the execution times of the sender and the receiver processes. If the rate limiting is set to a value that is as high as the execution time of the sender or the receiver, this guarantees that the sender or the receiver will manage to encode or decode one bit of information, respectively, before being preempted. This scenario depicts the most lucrative case for the attackers. Setting the rate limiting to a value lower than their execution times is not expected to have an effect on the covert-channel exploit. The reason is that the rate limiting parameter imposes a minimum runtime guarantee for the VMs, but after the rate limiting time has elapsed, the VMs are not definitely preempted. Values larger than the execution times of the attacker are expected to harden the attack as they impose the sender or the receiver to run longer than needed.

Results and Discussion: We observe that by setting the rate limiting to 500000 the error rate is not increased, but the bandwidth is affected, as the value exceeds attacker's execution times. Thus, it guarantees that they will not be preempted for a longer period than needed. Logically, this hardens the attack and reduces the bandwidth. By setting rate limiting to the lowest possible value of 100, the guarantee that the receiver and the sender will not be preempted is not viable, but this affects neither the error rate nor the communication bandwidth. This happens as the rate limiting value does not affect the time the attackers are allowed to run before being preempted.

VI. Conclusion and Summary

This paper analyzes the influence of the VMM scheduling configuration on the feasibility of covert-channel exploits in a VE. It empirically shows the role of the scheduling parameters and how their values affect the covert communication channel in terms of error rate and achieved bandwidth. We have shown that all the parameters have an impact on the CCA's feasibility.

Depending on the implementation, the scheduling configuration affects the bandwidth, the error rate or both. Based on the conducted empirical study, we have noticed that the load balancing is the parameter that has the highest impact on the covert-channel because it controls its availability. Logically, the optimal configuration for the attackers is when they are pinned to the same CPU core and any additional workload is deployed to the other cores. Automatic load balancing makes the VE much more secure with respect to CCAs. If the sender and the receiver are equally weighted, their communication exhibits the highest bandwidth. A high difference in the weights of the sender and the receiver has a negative effect on the bandwidth of their communication. Low values for the cap and the time slice also affect the covert-channel communication and reduce the bandwidth, as they lead to attacker's preemption. The same bandwidth reduction occurs in case of very high values for the rate limiting parameter. Then the attackers are guaranteed too long execution time that makes their communication slower.

Based on our results, for a more secure VE, we recommend using the hypervisor's automatic load balancing feature. Better security can be also achieved if the VMs are differently weighted. If this degrades the VM performance, altering the VMs weights can be used. The choice of an adequately low value for the cap and the time slice and a relatively high value for the rate limiting parameter is also recommended despite possible performance implications.

Acknowledgements

Research supported by DFG GRK 1362.

References

[1] G. Popek and R. Goldberg, "Formal requirements for virtualizable third generation architectures," *CACM*, vol. 17, no. 7, pp. 412–421, 1974.
[2] T. Vateva-Gurova, J. Luna, G. Pellegrino, and N. Suri, "Towards a framework for assessing the feasibility of side-channel attacks in virtualized environments," in *Proc. of SECRYPT*, 2014, pp. 113–124.
[3] Y. Xu et al., "An exploration of L2 cache covert channels in virtualized environments," in *Proc. of CCSW*, 2011, pp. 29–40.
[4] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: high-speed covert channel attacks in the cloud," in *USENIX Security*, 2012, pp. 9–9.
[5] Y. Zhang, A. Juels, M. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in *Proc. of CCS*, 2012, pp. 305–316.
[6] T. Ristenpart et al., "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *CCS*, 2009, pp. 199–212.
[7] Y. Yarom and K. Falkner, "Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack." *IACR Cryptology Archive*, 2013.
[8] V. Varadarajan et al., "Scheduler-based Defenses against Cross-VM Side-channels," in *USENIX Security*, 2014, pp. 687–702.
[9] H. Hlavacs et al., "Energy Consumption Side-Channel Attack at Virtual Machines in a Cloud," in *Proc. of CGC*, 2011.
[10] T. Kim, M. Peinado, and G. Mainar-Ruiz, "STEALTHMEM: system-level protection against cache-based side channel attacks in the cloud," in *USENIX Security*, 2012, pp. 11–11.
[11] P. Li, D. Gao, and M. Reiter, "Mitigating access-driven timing channels in clouds using StopWatch," in *Proc. of DSN*, 2013, pp. 1–12.
[12] D. Stefan et al., "Eliminating Cache-Based Timing Attacks with Instruction-Based Scheduling," in *ESORICS*, 2013, pp. 718–735.
[13] J. Shi et al., "Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring," *DSN-W*, pp. 194–199, 2011.
[14] R. Martin, J. Demme, and S. Sethumadhavan, "Timewarp: Rethinking timekeeping and performance monitoring mechanisms to mitigate side-channel attacks." in *Proc. of ISCA*, 2012, pp. 118–129.
[15] K. Mowery, S. Keelveedhi, and H. Shacham, "Are AES x86 cache timing attacks still feasible?" in *Proc. of CCSW*, 2012, pp. 19–24.
[16] W.-M. Hu, "Lattice scheduling and covert channels," in *Proc. of Symposium on Security and Privacy*, 1992.
[17] F. Z. et al., "Scheduler vulnerabilities and coordinated attacks in cloud computing," in *Proc. of NCA*, 2011, pp. 123–130.
[18] Jacob Mathai , "Scheduling in Xen," http://wiki.xen.org, Xen Project. Last accessed on 01.11.2014.
[19] M. Jones, "Inside the Linux 2.6 Completely Fair Scheduler," http://www.ibm.com/developerworks/library/l-completely-fair-scheduler/, IBM. developerWorks. Last accessed on 03.12.2014.
[20] P. Kadam and N. Alone, "Review on KVM Hypervisor," *IJRTE*, vol. 3, no. 4, pp. 54–59, 2014.
[21] K. Duda and D. Cheriton, "Borrowed-virtual-time (BVT) Scheduling: Supporting latency-sensitive threads in a general-purpose scheduler," in *Proc. of SOSP*, 1999, pp. 261–276.
[22] Xen Project, "Credit Scheduler," http://wiki.xen.org/wiki/Credit_Scheduler.
[23] Barham, Paul et al., "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.
[24] L. Cherkasova et al., "Comparison of the Three CPU Schedulers in Xen," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 2, pp. 42–51, 2007.
[25] C. Percival, "Cache missing for fun and profit," in *BSDCan*, 2005.
[26] "Xen Project," http://www.xenproject.org/, Last accessed on 13.10.2014.