Protecting Cloud-based CIs: Covert Channel Vulnerabilities at the Resource Level

Tsvetoslava Vateva-Gurova[1], Salman Manzoor[1],
Ruben Trapero[2], and Neeraj Suri[1]

[1] Technische Universität Darmstadt, CS Dept., Germany
{vateva,salman,suri}@deeds.informatik.tu-darmstadt.de,
[2] Atos Research and Innovation, Spain
ruben.trapero@atos.net

Abstract. Critical Infrastructures (CIs) increasingly leverage Cloud computing given its benefits of on-demand scalability, high availability and cost efficiency. However, the Cloud is typically characterized by the co-location of users from varied security domains that also use shared computing resources. This introduces a number of resource/architecture-level vulnerabilities. For example, the usage of a basic shared storage component, such as a memory cache, can expose the entire Cloud system to security risks such as covert-channel attacks. The success of these exploits depends on various execution environment properties. Thus, providing means to assess the feasibility of these attacks in a specific execution environment and enabling postmortem analysis is needed.
While attacks at the architectural level represent a potent vulnerability to exfiltrate information, the low-level often get neglected with techniques such as intrusion detection focused more on the high-level network/middleware threats. Interestingly, cache-based covert-channel attacks are typically not detectable by traditional intrusion detection systems as covert channels do not obey any access rights or other security policies. This paper focuses on the information provided at the low architectural level to cope with the cache-based covert-channel threat. We propose the usage of feasibility metrics collected at the low level to monitor the core-private cache covert channel and, infer information regarding the probability of a covert-channel exploit happening. We also illustrate the applicability of the proposed feasibility metrics in a use case.

Keywords: Information Leakage, Scheduling, Side Channels, Covert Channels, Feasibility

1 Introduction

For its cost efficiency, high availability and on-demand scaling, Cloud computing has become prevalent for myriad applications. According to Gartner, Inc. by 2020 [2], a "no-cloud" policy will be rare in the corporate world. Moreover, a survey conducted in 2014 by Healthcare Information and Management Systems

Society (HIMMS) Analytics [4] confirms the increasingly wide-spread adoption of Cloud services in critical sectors such as Healthcare. As Critical Infrastructures (CIs) increasingly leverage the Cloud computing for its scalability, large-scale connectivity and high availability, consequently, the security vulnerabilities of the Cloud affect the security of CIs driving the need for sophisticated preventive measures.

Covert- and side-channel attacks (CCAs and SCAs) have long been considered a system security threat, and their prevalence is increasing given the shared resources model employed by the Cloud[3], as demonstrated in [16, 11, 23]. The lack of approaches to address SCAs and CCAs practically without introducing significant performance overhead or, the need of additional hardware which increases the financial cost worsen the situation. SCAs and CCAs can pose a threat to any system performing, e.g., sensitive cryptographic operations, including critical infrastructure hardware that provides shared access to users of different security domains. As the Cloud computing becomes prevalent, the risk of covert-channel attacks or side-channel attacks stemming from the usage of shared hardware should not be underestimated.

Varied resources exhibiting observable side effects can be used as a covert or side channel[4]. SCAs and CCAs can be conducted by analyzing the physical effects of the hardware being used, e.g., power analysis (cf., [15, 5]), acoustic noise (cf., [3]) or electromagnetic emanations (cf. [1]), or through spying software. While the attacks based on the analysis of the physical effects of a device might require additional measuring equipment or proximity to the device being attacked, the software-based approaches are more practical. They can be conducted, e.g., by periodically timing the accesses to a resource that is being used by both an attacker and a victim, and do not require special privileges for the attacker. Whereas an SCA is characterized by an attacker spying on the victim's usage of the shared resource, a CCA is characterized by two cooperating attackers using the shared resource to transmit information. In the latter case, one of the attackers accesses the shared resource in a specific predefined way to transfer the data to the other attacker.

With our focus on core-private caches as a covert channel (CC), it has already been demonstrated that the properties of the execution environment affect the feasibility of SCAs and CCAs [19, 26]. Among these properties is the CPU scheduling that might allow for a fine-granular observation of the victim's cache usage [18, 20, 26, 7]. There is a strong correlation between the scheduling of the attacker directly after the victim in a side-channel attack, and the success of the core-private side-channel attacks employing the Prime+Probe strategy. Thus, available scheduling traces can be used to provide valuable information regarding the feasibility of a core-private cache-based CCA or SCA for a given system, as analyzing the effect of the scheduling algorithms on these attacks directly seems to be infeasible due to the involved indeterminism. In addition, abusing

[3] CCAs and SCAs represent a threat to the Cloud settings despite the presumed secure isolation among the Cloud users.

[4] The terms covert channel and side channel are used interchangeably within the paper.

the scheduler is often a prerequisite for a successful covert-channel exploit, as the attacker needs frequent observations on the activities of the victim or the cooperating attacker. This is usually achieved through Inter-processor Interrupts (IPIs). For instance, the authors of [26] use IPIs to collect fine-granular data on the victim's cache usage. Such information can be applied to assess the feasibility of SCAs and CCAs in a particular system and conduct post-mortem analysis without ignoring the load on the system and the properties of the execution environment.

Contributions & Paper Organization

This paper addresses the effects the scheduling has on the covert-channel exploitation using the Prime+Probe strategy [16, 26, 10]. While contemporary research often focuses on Flush+Reload-based SCAs and CCAs [24, 8], this coverage is limited, as this class of attacks depends on the usage of features such as memory deduplication which are disabled in the most Cloud settings [22, 21]. The CPU scheduling enabling proper synchronization upon the usage of the covert channel is a cornerstone for the success of the conducted attack. As the core-private cache can be seen as a memory-less communication channel, each process intervening with the cache erases the cache footprint and with that the data conveyed by the footprint. In this case the attacker and the victim are not synchronized which affects the quality of the obtained data.

Having this in mind, we propose the monitoring of three scheduling-related metrics to infer the feasibility of a cache-based exploit. The frequency of interprocessor interrupts and busy waiting and the number of successive scheduling of two processes on the same CPU core are valuable sources of information regarding the probability of an attack happening. To this end, we propose to use the scheduling traces for post-mortem analysis or feasibility assessment of a covert-channel exploit. We conduct our analysis based on a case study by considering the feasibility of a L1 covert-channel attack based on the Prime+Probe strategy.

Paper Contributions: Overall, our contributions are threefold:

1. We conduct a systematic abstract information-level characterization of scheduling considerations for side- and covert-channel attacks,
2. We propose architectural level metrics, based on IPIs, busy waiting and successive scheduling to assess the feasibility of a cache exploit (in Section 4),
3. We demonstrate of the applicability of the proposed metrics to monitor the CC in a case study (in Section 5).

Key Findings: The experimental results demonstrate the utility of the chosen metrics as indicators for possible successful transmission over the cache as a covert channel. This ascertains the usefulness of the usage of the low-level information to extract feasibility metrics for cache-based covert-channel exploits.

2 Related Work

This section reviews the related work in the area of side- and covert-channel attack approaches, and discusses the state-of-the-art research related to the scheduling effect on these attacks.

Among the most popular strategies for conducting SCAs and CCAs are Prime+Probe (employed in, e.g., [16, 26, 10]) and Flush+Reload (demonstrated in, e.g., [24, 8]). They have been applied to leak information at different cache levels. Prime+Probe had been leveraged only to target exploiting the core-private caches [26, 23] due to the previously unknown or undocumented Last Level Cache (LLC) addressing scheme of modern processors in the past. However, the reverse engineering of the LLC complex addressing scheme, described in [13, 25], made even the Prime+Probe attacks targetting the LLC practical [10].

The Flush+Reload attacks represent a powerful mean for exfiltrating secret data. However, their practical applicability is limited due to the assumptions behind this class of attacks. Flush+Reload attacks rely on the usage of features such as memory deduplication that enables the usage of the same memory pages by two processes that might be adversaries. As these attacks can be easily prevented by disabling the memory deduplication feature, they are not in the focus of our work. Actually, this measure has already been applied by many Cloud providers, as noted in [22, 21].

The effect of the synchronization between the attackers in a CCA and the attacker and the victim in an SCA has already been studied. Hu [6] proposed to use fuzzy time system clocks to disable the proper synchronization and reduce the capacity of a CC. However, this approach is not practical, as it poses restrictions on the execution environment of other processes, as well.

The authors of [11] also discussed the varied factors that influence the feasibility of cache-based CCAs, among them the synchronization of the covert communication. Based on that observation, the authors defined three types of errors in the CC in [11] distinguishing between substitution errors, insertion errors and deletion errors.

In [26], Zhang et al. abused the scheduler by issuing interrupts to obtain fine-granularity of their observations on the victim's actions. The synchronization (i.e., the scheduling effect) has been reported as a challenge or even feasibility aspect for conducting SCAs and CCAs in varied papers e.g., [26, 11, 19]. The scheduling policy is used even as a defense mechanism against cache-based SCAs in the research described in [18].

Although works related to the impact of scheduling on the accesses to the CC exist, there is no comprehensive, systematic analysis on the topic that might be applied on varied systems to analyze the covert-channel threat using the scheduling traces.

3 System and Attacker Models

We now present the system model serving as a basis for the paper. Additionally, the relevant SCA and CCA attacker models are detailed. This paper focuses on

the core-private cache as a CC between two processes. Thus, our system model, depicted in Fig. 1, consists of the CPU and the core-private cache, as being part of the CPU and providing the basis for the covert communication.

A hierarchy of caches, targeting frequently used data, is used to alleviate the speed bottleneck caused by the slow accesses to the main memory. Current architectures are characterized by more than one cache levels where the Level 1 (L1) cache is the fastest and the smallest cache. It is divided into L1 data cache and L1 instruction cache and is core-private. The Level 2 (L2) cache is unified (i.e., storing data and instructions) and, might be core-private or shared depending on the architecture. The largest and slowest is the LLC. Some architectures might also have only L1 (core-private) and last level (shared) caches. As the CPU scheduling is the focus of this work, the CPU scheduler assigning the processes to the CPU cores and with that granting the access to the respective core-private cache, is a significant part of the system model.

Due to the timing differences for the data accesses, it can be inferred where the requested data has been stored (main memory or certain cache level). If the data has to be fetched from the main memory and to be copied into the cache, accessing it takes longer. This is referred to as a cache miss. On the contrary, if the data is already in the cache, accessing it is faster and is referred to as a cache hit. Namely the timing differences in case of a cache hit and cache miss can be analyzed and exploited to leak secret information though the cache as a CC. Through such an analysis, by frequently sampling the cache, an attacker can infer the cache access pattern or cache footprint of another process and can deduce information regarding other process' cache usage which can be further analyzed. In such a scenario, the attacker can be an arbitrary non-privileged process using the same core-private cache as the other process.

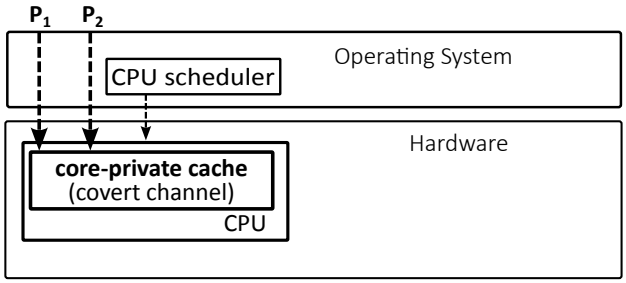

Fig. 1: System model.

This paper focuses on the usage of the CC in both SCAs and CCAs. Although these attacks use the same communication channel, there are slight differences in their attacker models. On one hand, a SCA is characterized by a victim and by an attacker spying on the victims usage of the cache to leak, e.g., secret keys, whereas in the CCA two attackers exchange information over the cache, e.g., a message. In both cases, the leaked information is in the form of cache footprints

that are being collected during the attack, and analyzed after that to decode their meaning. For simplicity, from now on, the party leaving the cache footprint (e.g., the victim in an SCA) is referred to as a *sender* for both types of the attack, and the party decoding the cache footprint (the attacker in an SCA) is called the *receiver* for both attacks. Any other process that interacts with the cache causes noise in the channel from the perspective of the sender and the receiver.

4 CPU Scheduling and Feasibility Metrics

This section discusses on the effect of the scheduling on the exploitability of the cache as a covert channel using the Prime+Probe strategy.

4.1 Scheduling Considerations

The proposed metrics are related to the scheduling of the attacker process(es). Thus, in the next paragraphs we present scheduling issues relevant for the successful attack deployment, and detail what are the scheduling considerations that have to be taken into account for the metrics derivation.

Abusing the Scheduler. Zhang et al. demonstrated a L1 side-channel attack in a Cloud scenario in [26]. In their paper, the authors propose using a third party (an additional VM) to issue IPIs, so that the the attacker can be scheduled frequently enough after the victim. This enables the attacker to collect fine-granular observations over the victim's cache usage. This approach plays a major role in the demonstrated attack. This is, however, only possible if the third party process can interrupt the victim process frequently enough.

From this perspective, non-preemptive scheduling approaches tend to be more secure against such attacks. Still, due to performance implications, the non-preemptive scheduling approaches are almost obsolete in user space of the modern operating systems nowadays. In contrast, the attacker process could even try to influence on which CPU core it gets scheduled, to adjust under given circumstances its priority or to be scheduled according to a chosen scheduling scheme [9]. Although these possibilities are beneficial from a performance and usability point of view, they increase the risk of abusing the scheduler to achieve better synchronization over the usage of the covert channel.

Synchronisation and Atomicity of Operations. As already discussed, the synchronization between the involved parties in a side-channel attack and a covert-channel attack, i.e., between the victim and an attacker in the SCA scenario and between the two cooperating attackers in the CCA scenario, is crucial for the success of the attacks. Ideally, the receiver is scheduled directly after the sender. Moreover, the receiver has to be scheduled after the sender has completed its operation to retain the atomicity of the operations. If the sender gets preempted in the middle of the sending, the receiver will not be able to decode the data.

The same applies if the sender uses the CPU for too long while performing more than one operations. Then, the receiver would not be able to collect all the possible observations. The times the receiver and sender should be scheduled depend on the execution times of the sending and receiving operations to achieve optimal attack results. It is also important that the receiver and sender processes are of roughly the same priority so that they have similar chances to be scheduled.

4.2 Low-Level Feasibility Metrics

Having in mind the scheduling effect on the feasibility of side-channel attack or covert-channel attack, we propose to monitor the scheduling related metrics to exfiltrate information regarding the usage of the covert channel. For this purpose, we propose using three feasibility metrics. They are briefly discussed below:

Successive scheduling (SS). The successive scheduling of two processes can be used as an indicator for the feasibility of the usage of the core-private cache as a covert channel. As the receiver has to be scheduled directly after the sender to obtain information, this metric can be applied as a feasibility indicator. We can monitor SS for a process with suspicious behaviour.

Busy waiting (BW). In a Prime+Probe attack, the synchronisation is usually done through busy waiting. After each measurement, the attacker process yields the CPU and during attacker's busy waiting the victim should get scheduled. This is a very important step in the attack to guarantee the atomicity of operations. Thus, BW can be used not only as a feasibility metric, but also to identify the processes with suspicious behaviour for which the SS metric should be considered.

Interprocessor Interrupts (IPIs). IPIs can be used to abuse the scheduler to grant frequent access to the cache. Thus the number of IPIs over predefined time span can be monitored as an indicator that an attack, as the one described in [26], might be happening. By considering these feasibility metrics, a system administrator can derive a threshold, and monitor for the feasibility of an attack. This is a lightweight approach to monitor ICT systems, among them also CIs, for possible cache-based covert-channel exploits, and enhance the security of these systems.

5 Covert Channel Monitoring and Application Scenario

In this section, we analyze empirically the utility of the proposed feasibility metrics, by conducting post-mortem analysis based on a covert-channel exploit. For that, we deployed a CCA in three different scenarios using varied background load levels, and collected the proposed metrics during the attack. Additionally, for comparison purposes we gathered the feasibility metrics in a non-attacked system with varied background load levels. The experiments were done in Debian Stretch

Operating System. For each experiment, we measured the success of the attack and extracted the proposed metrics using the available hardware performance counters.

5.1 Implementation Details

We implemented a Prime+Probe CCA consisting of a sender and a receiver processes which communicate with each other through the cache footprints left on the L1 cache. For this purpose, the processes had agreed on the meaning of the cache footprint, where each footprint was mapped to either a bit 0, a bit 1 or was invalidated by the receiver. Our attack resembles the attack described in [12], but employed the L1 cache instead of the LLC to covertly transmit information. Similarly to the attack in [12], for sending bit 1, the attacker accesses the whole L1 cache, whereas for sending bit 0 the attacker does not deliberately access the cache. A header and footer are additionally employed to wrap up the transmitted data, so that the receiver could notice where the actual message starts.

The receiver process samples the timestamp counter hardware register (TSC) before and after accessing the cache to be able to infer the cache footprint left by the sender. To enforce executing the instructions in the expected order, we use volatile as a memory barrier, and the CPUID instruction for serialization. The out-of-order execution issue is also considered in [14, 17]. Timor et al. apply double execution to protect the out-of-order part of the CPU by executing each instruction twice in [17]. The scheduler's *setaffinity* option is used to request that the sender and the receiver processes will be executed on the same CPU core and will use the same L1 cache.

5.2 Experiments and Results

We conducted 100 experiments per setting, and in each experiment messages of length 5000 were sent over the CC as packets consisting of 1000 bits payload and 500 bits header and footer each. In each experiment the relevant hardware performance counters were logged simultaneously with the attack to enable extracting the feasibility metrics.

Altogether, we have five settings for five different scenarios. The experiments were deployed by simulating load on the system with the stressing tool stress-ng. Background load levels of 0%, 40% and 80% per CPU core were considered in each of the settings with a running attack, and load levels of 40% and 80% were considered for the non-attacked system. As feasibility metrics, we employed the frequency of successive scheduling (SS) between the sender and the receiver, and the times the receiver process is busy waiting (BW). To assess their applicability, we considered the number of successful transmissions (STs) as an indicator for attack's success.

The relationship between the measurements can be seen in Table 1. The results, summarized in the table, demonstrate the large difference between BWs in an attacked and a non-attacked system. This observation is visible also in Fig. 2 showing the tremendous BWs increase in an attacked system compared to a

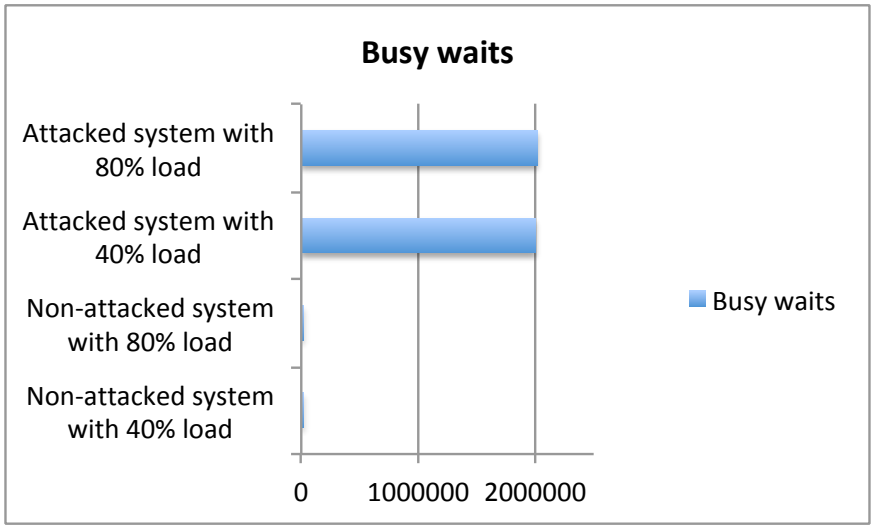

Fig. 2: The bar chart reveals a great difference between the number of BWs in a non-attacked and an attacked system.

non-attacked system. This confirms that BW can be considered as a starting point for the covert-channel feasibility analysis to focus on a suspicious process. As expected and visible from the results, with the load increase, the STs decrease. This is normal due to more contention for the CC. Indirectly, this is also a scheduling effect, as the scheduler grants all the processes CPU time.

The presented results discern that an increased number of issued BWs can be taken a starting point for the further analysis and, can indicate that measurement of the SSs is needed. The BWs metric is particularly suitable for initiating further investigation due to the minimal performance overhead related to its measurement. By using such an approach, we can have an indicator regarding the probability of a covert-channel exploit.

6 Conclusion

Core-private caches represent a convenient and practical way for exfiltrating secret information and endanger ICT systems, including CIs. Attacks abusing the caches as covert channels are hard to be detected, as the caches are easily accessible without any privileges.

To address this threat and enhance the security in CIs and other ICT systems, we proposed the usage of feasibility metrics to assess the probability of a covert-channel exploit happening in the system or, to conduct post mortem analysis. The proposed feasibility metrics can be derived using hardware performance counters, and represent a lightweight way to reason about the possible covert channel threat. To validate our proposal, we demonstrate the applicability of the

Table 1: Results summary.

	CCA employed			No attack	
	0%	**40%**	**80%**	**40%**	**80%**
# Experiments	100	100	100	100	100
STs (overall)	494007	340735	262599	–	–
STs (average per exp.)	4940, 07	3407, 35	2625, 99	–	–
Standard deviation for ST	305	942	1056	–	–
BW (overall)	2024152	2005113	2016318	4	16
BW (average per exp.)	20241.52	20051.13	20163.18	0.04	0.16

proposed metrics by conducting experiments with a L1 covert-channel attack and considering varied scenarios.

Our results discern that the busy waiting and the successive scheduling of the processes can reliably be correlated with the success of a covert-channel exploit using the L1 cache. The proposed metrics help systematically ascertain efficient ways to address such exploits, and to facilitate security enhancement in ICT systems, including CIs.

Acknowledgements.

Research supported in part by EC NECS GA#675320 and CIPSEC GA#700378.

References

1. Agrawal, D., Archambeault, B., Rao, J., Rohatgi, P.: The EM Side-Channel(s). In: Proc. of CHES. vol. 2523, pp. 29–45. Springer-Verlag (2002)
2. Gartner, Inc. : Why a No-Cloud Policy Will Become Extinct. https://www.gartner.com/smarterwithgartner/cloud-computing-predicts/ (2016), Last accessed on 10.07.2018.
3. Genkin, D., Shamir, A., Tromer, E.: RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. Cryptology ePrint Archive, Report 2013/857 http://eprint.iacr.org/ (2013)
4. HIMMS Analytics: 2014 HIMMS Analytics Cloud Survey. https://www.himss.org/file/1308371/download?token=CBkkly5K (2014), Last accessed on 07.07.2018.
5. Hlavacs, H., Treutner, T., Gelas, J.P., Lefevre, L., Orgerie, A.C.: Energy Consumption Side-Channel Attack at Virtual Machines in a Cloud. In: Proceedings of the 2011 IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2011). pp. 605–612 (2011)
6. Hu, W.M.: Reducing Timing Channels with Fuzzy Time. In: Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy. pp. 8–20 (May 1991)

7. Hu, W.M.: Lattice Scheduling and Covert Channels. In: Proceedings of the 1992 IEEE Symposium on Security and Privacy. pp. 52–61. SP '92, IEEE Computer Society, Washington, DC, USA (1992), http://dl.acm.org/citation.cfm?id=882488.884165
8. Irazoqui, G., Inci, M.S., Eisenbarth, T., Sunar, B.: Wait a Minute! A fast, Cross-VM Attack on AES. In: Proc. of the Conference on Research in Attacks, Intrusions and Defenses. pp. 299–319. RAID 2014, Springer International Publishing (2014)
9. Kerrisk, M.: The Linux man-pages project. http://man7.org/linux/man-pages/man7/sched.7.html (2013), Last accessed on 02.07.2018
10. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B.: Last-Level Cache Side-Channel Attacks are Practical. In: Proceedings of the 2015 IEEE Symposium on Security and Privacy. pp. 605–622. SP '15, IEEE Computer Society, Washington, DC, USA (May 2015)
11. Maurice, C., Weber, M., Schwarz, M., Giner, L., Gruss, D., Boano, C.A., Römer, K., Mangard, S.: Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud. In: Proceedings of the 24th Annual Network and Distributed System Security Symposium, NDSS 2017 (2017)
12. Maurice, C., Neumann, C., Heen, O., Francillon, A.: C5: Cross-cores cache covert channel. In: Proc. of the Conference on Detection of Intrusions and Malware & Vulnerability. pp. 46–64. DIMVA 2015, Springer International Publishing (2015)
13. Maurice, C., Scouarnec, N., Neumann, C., Heen, O., Francillon, A.: Reverse Engineering Intel Last-Level Cache Complex Addressing Using Performance Counters. In: Proc. of the Conference on Research in Attacks, Intrusions and Defenses. pp. 48–65. RAID 2015, Springer International Publishing (2015)
14. Mendelson, A., Suri, N.: Designing High-Performance & Reliable Superscalar Architectures: The out of Order Reliable Superscalar (O3RS) Approach. In: Proceedings of the International Conference on Dependable Systems and Networks. pp. 473–481. DSN 2000, IEEE Computer Society (June 2000)
15. Messerges, T., Dabbish, E., Sloan, R.: Investigations of Power Analysis Attacks on Smartcards. In: Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology. pp. 17–17. WOST'99, USENIX Association, Berkeley, CA, USA (1999)
16. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds. In: Proceedings of the 16th ACM Conference on Computer and Communications Security. pp. 199–212. CCS '09, ACM, New York, NY, USA (2009)
17. Timor, A., Mendelson, A., Birk, Y., Suri, N.: Using Underutilized CPU Resources to Enhance Its Reliability. IEEE Transactions on Dependable and Secure Computing 7(1), 94–109 (January 2010)
18. Varadarajan, V., Ristenpart, T., Swift, M.: Scheduler-based Defenses against Cross-VM Side-channels. In: Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14). pp. 687–702. USENIX Association, San Diego, CA (2014)
19. Vateva-Gurova, T., Luna, J., Pellegrino, G., Suri, N.: Towards a Framework for Assessing the Feasibility of Side-channel Attacks in Virtualized Environments. In: Proceedings of the 11th International Conference on Security and Cryptography - Volume 1: SECRYPT, (ICETE 2014). pp. 113–124. SciTePress (2014)
20. Vateva-Gurova, T., Suri, N., Mendelson, A.: The Impact of Hypervisor Scheduling on Compromising Virtualized Environments. In: Proceedings of the 2015 IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2015). pp. 1910–1917 (2015)

21. VMware: Additional Transparent Page Sharing management capabilities and new default settings. Tech. Rep. 2097593, VMware, https://kb.vmware.com/s/article/2097593, Last accessed on 07.06.2018.
22. VMware: Security Considerations and Disallowing Inter-virtual Machine Transparent Page Sharing. Tech. Rep. 2080735, VMware, https://kb.vmware.com/s/article/2080735, Last accessed on 07.06.2018.
23. 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 Workshop on Cloud Computing Security. pp. 29–40 (2011)
24. Yarom, Y., Falkner, K.: FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-channel Attack. In: Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14). pp. 719–732. USENIX Association, San Diego, CA (2014)
25. Yarom, Y., Ge, Q., Liu, F., Lee, R.B., Heiser, G.: Mapping the Intel Last-Level Cache. IACR Cryptology ePrint Archive 2015, 905 (2015)
26. Zhang, Y., Juels, A., Reiter, M., Ristenpart, T.: Cross-VM Side Channels and Their Use to Extract Private Keys. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. pp. 305–316. CCS '12, ACM, New York, NY, USA (2012)