A Study on Microarchitectural Covert Channel Vulnerabilities in Infrastructure-as-a-Service

Benjamin Semal(✉), Konstantinos Markantonakis, Raja Naeem Akram, and Jan Kalbantner

Royal Holloway University of London, Egham, United Kingdom
benjamin.semal.2018@live.rhul.ac.uk

Abstract. Microarchitectural cross-VM covert channels are software-launched attacks which exploit multi-tenant environments' shared hardware. They enable transmitting information from a compromised system when the information flow policy does not allow to do so. These attacks represent a threat to the confidentiality and integrity of data processed and stored on cloud platforms. Although potentially severe, covert channels tend to be overlooked due to an allegedly strong adversary model. The literature focuses on mechanisms for encoding information through timing variations, without addressing practical considerations. Furthermore, the field lacks a realistic evaluation framework. Covert channels are usually compared to each other using the channel capacity. While a valuable performance metric, the capacity is inadequate to assess the severity of an attack. In this paper, we conduct a comprehensive study on the severity of microarchitectural covert channels in public clouds. State-of-the-art attacks are evaluated against the Common Vulnerability Scoring System in its most recent version (CVSS v3.1). The study shows that a medium severity score of 5.0 is achieved. In comparison, the SSLv3 POODLE (CVE-2014-3566) and OpenSSL Heartbleed (CVE-2014-0160) vulnerabilities achieved respective scores of 3.1 and 7.5. As such, the paper successfully demonstrates that covert channels are not theoretical threats, and that they require the immediate attention of the community. Furthermore, we devise a new and independent scoring system, the Covert Channel Scoring System (CCSS). The scoring of related works under the CCSS shows that cache-based covert channels, although more and more popular, are the least practical ones to deploy. We encourage authors of future cross-VM covert channel attacks to include a CCSS metric in their study, in order to account for deployment constraints and provide a fair point of comparison for the adversary model.

Keywords: Covert channel · Microarchitectural attack · Cloud privacy · Vulnerability study.

1 Introduction

The multi-tenant nature of cloud platforms prompts concerns over the confidentiality and integrity of data [31]. When multiple virtual machines (VMs) are

scheduled on the same hardware platform, they compete with each other for processor resources. Such conflicts can delay the execution of certain instructions, resulting in timing variations to occur during the execution of an application. These timing variations can in turn be exploited by two colluding entities in order to encode and decode binary information. *Microarchitectural covert channel attacks* allow tunneling information out of a compromised system when the security policy does not allow doing so. A sending-end is embedded into the victim's environment, and transmits information to the receiving-end located in the attacker's environment. Furthermore, covert channel attacks are relevant when there is no other mean of leaking information in a non-conspicuous manner, e.g. as part of an advanced persistent threat malware. We note that side channel attacks, which rely on an accidental leakage of information from the victim, are beyond the scope of this study.

Microarchitectural covert channels allegedly rely on a strong adversary model. First, the attacker must infect the victim's instance with a malicious sending-end. Second, the attacker requires co-locating her instance on the same hardware platform as the victim's instance. Researchers devising these attacks tend to focus on new mechanisms for generating timing variations, rather than addressing deployment constraints. Indeed, the trend is to propose covert channels that are always faster and more robust, while assuming an ideal scenario for the attacker. Meanwhile, experts responsible for implementing security policies are free to rebrand microarchitectural covert channels as non-practical exploits, due to the above-mentioned challenges.

This paper investigates the operational constraints of launching a covert channel attack across Infrastructure-as-a-Service (IaaS) instances. To do so, a measurement study on the practicality and severity of these attacks is conducted. The Common Vulnerability Scoring System (CVSS) is used as a support for our analysis. Criteria are discussed in the context of microarchitectural attacks and potentially re-interpreted. Our study shows that microarchitectural covert channels achieve a medium severity score of up to 5.0, discarding the assumption that covert channels aren't practical. In comparison, the MySQL Stored SQL Injection vulnerability [2] achieved a medium severity scrore of 6.4 (CVSS v3.1), and was patched shortly after its disclosure. To this day, there are still no practical countermeasures against severe covert channels released several years ago [30,21]. Secondly, we propose a new evaluation framework dedicated to microarchitectural covert channels, the Covert Channel Scoring System (CCSS). This framework evaluates state-of-the-art attacks, and outlines the effect of operational constraints on the severity score. Among other findings, this evaluation shows that cache-based covert channels achieve the lowest severity scores, despite being increasingly popular. Overall, the paper reveals the existence of a growing gap between academic efforts and the commercial ecosystem.

The contributions of this paper are summarised as follows:

- We propose the first comprehensive study on the practicality and severity of microarchitectural covert channels in IaaS, resulting in CVSS scores ranging from 4.2 to 5.0.

– We devise a new and independent Covert Channel Scoring System, so as to provide a fair and realistic evaluation framework for future research.

The paper is organised as follows. Section 2 provides a background on cloud services, processor architecture, and defines covert channel attacks. In Section 3, the criteria discussed in this paper are defined. Section 4 rates criteria which can be evaluated generically. Section 5 discusses state-of-the-art covert channels, and provides an individual rating for the remaining criteria. Section 6 details the resulting scores. Finally, we conclude in Section 7.

2 Background

2.1 Infrastructure-as-a-Service

Infrastructure-as-a-Service (IaaS) delivers internet-accessible storage, processing, and network resources. The end-user controls every component inside the virtual machine, while the service provider manages servers and orchestrators (or containers). Customers remain in control of the (sensitive) data being processed within the VM, including data from any other service built upon it, i.e. platform or software. Also, the user can interact with the instance as with any other machine (e.g. root access, hardware selection APIs, etc). IaaS minimises the trust that needs to be extended to the cloud provider, and emphasises responsibilities mostly on the customer.

2.2 Processor Organisation

We use the term processor to refer to the processor die, which includes the cores and the last-level cache (LLC). The core contains the CPU along with the level-2

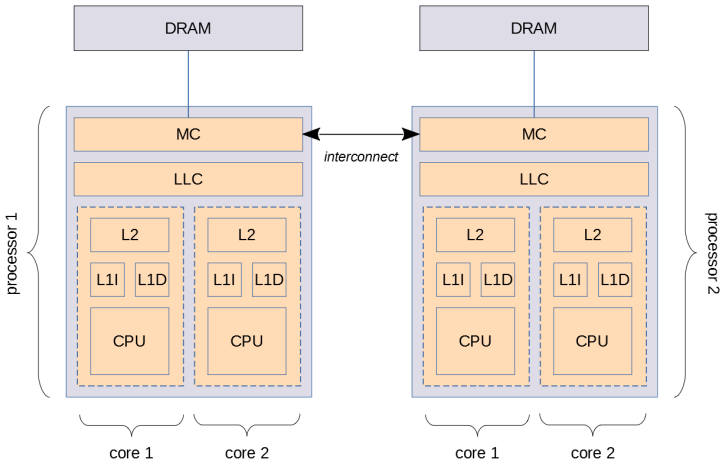

Fig. 1. Processor overview (NUMA configuration).

(L2), level-1 instruction (L1-I) and data (L1-D) caches. The memory bus refers to the front side bus present on older microarchitectures. Since the Nehalem microarchitecture, Intel processors mainly rely on a non-uniform memory access (NUMA) configuration. As a result, the memory bus has been removed, and memory controllers (MC) have been integrated into the processor die. The interconnect allows a processor accessing a region of DRAM that is managed by another processor. Figure 1 provides a representation of a multi-processor multi-core system in the NUMA configuration.

2.3 Scope of the Evaluation

Microarchitectural covert channels exploit vulnerabilities in the implementation of a processor's architecture. In contrast, network covert channels abuse network protocols. This study focuses on the former case. Also, not all covert channels have malicious intents. Nevertheless, we use the terms covert channel and covert channel attack interchangeably, for the sake of simplicity. We assume that a covert channel has a malicious nature. Microarchitectural covert channels differ from microarchitectural side channels in the attack scenario. Although they share the same underlying mechanisms, side channels rely on an accidental leakage of information from the victim, and they do not require compromising the victim's environment a priori. Side channel attacks are beyond the scope of our work. Finally, a covert channel does not necessarily allow communication across VMs. Because this paper focuses on the threat against IaaS, only cross-VM covert channels are considered. In parallel, this study could be applied to environments similar to IaaS, e.g. private clouds. This evaluation is specific to IaaS, and does not account for variations in similar environments. Table 1 surveys relevant covert channel attacks. We note that the respective channel capacities have been calculated here under the binary symmetric model [23].

3 Criteria of Evaluation

This section lists the criteria used to assess the impact of malicious covert channels in IaaS environments. We use the Common Vulnerability Scoring System in its most recent version (CVSS v3.1) [3] as a base for our evaluation metrics. The CVSS is an open industry standard that is widely used in the security community in order to assist responses to threats. While other evaluation frameworks exist, these have been adapting the traditional calculation of the CVSS to specific industrial environments. Certain criteria can be directly applied to all covert channels (**C1** to **C8**), while others are specific to the covert channel considered (**C9** to **C12**). We note that the criteria **C11** and **C12** are not part of the CVSS framework. These have been selected in order to provide an optimal representation of cross-VM covert channels' adversary model.

C1 **Attack vector** evaluates the proximity between the attacker and its target. This criterion can be rated as "network" for remote interaction, "adjacent"

Table 1. Cross-VM covert channel attacks.

Attack	Exploited resource	Bitrate	Error	Capacity
[22]	Last-level cache	0.2 bps	-	-
[32]	Last-level cache	3.2 bps	9.28%	1.77 bps
[30]	Memory bus	343 bps	0.39%	330 bps
[14]	Last-level cache	1.2 Mbps	22%	287 kbps
[16]	Last-level cache	751 bps	5.7%	514 bps
[21]	DRAM row-buffer	596 kbps	0.4%	573 kbps
[17]	Last-level cache	45.25 kbps	0%	45.25 kbps
[24]	Memory order buffer	1.49 Mbps	∼5%	1.06 Mbps
[23]	Memory controller	150 bps	7.8%	90.7 bps

when the attacker needs physical or logical proximity with the target (e.g. Bluetooth), "local" if it relies on user interaction (e.g. social engineering), or "physical" when physical manipulation is required.

C2 Attack complexity assesses the difficulty of exploiting a vulnerability once access to the targeted platform is gained, ranked either as "low" when no specialised access condition exists, or "high" when the attack requires a significant amount of preparation such that it cannot be performed at will. We discuss under this criterion the VM co-location problem.

C3 User interaction indicates whether human interaction other than the adversary is required. As such, this criteria can be rated as either "none" or "required". We discuss under this criterion the trojan insertion problem.

C4 Scope assesses the impact that a vulnerability might have on components other than the one affected by the vulnerability. This metric accounts for the overall system damage caused by the exploitation of the reported vulnerability. Scope can be rated as "changed" when a scope change occurs, or "unchanged" otherwise.

C5 Confidentiality impact assesses the severity of a disclosure of information, as well as the quantity of information that can be leaked. This criterion can be rated as "none", "low" when the attacker can only access a small amount of data and loss of this data does not result in serious consequences, or "high" otherwise.

C6 Integrity impact measures the attacker's capability to tamper with the victim's data. It can be rated as "none", "low" when the amount of data that can be modified is limited and modification of this data does not result in serious consequences, or "high" otherwise.

C7 Exploit code maturity evaluates the state of an attack, from a conceptual exploit to a fully autonomous malware. Exploitability can be rated as "unproven", "proof-of-concept" when the attack has been demonstrated but is not practical, "functional" when the exploit works in most systems where the vulnerability is present but is still not widely accessible, or "high" otherwise.

C8 Report confidence assesses the credibility of the source which reported the vulnerability. This criterion is rated as "confirmed" when originating from

a publication, "reasonable" when multiple non-official sources reported the vulnerability, or "unknown" when a single non-official source is involved.

C9 **Privileges required** evaluates the level of privileges that the adversary must acquire before launching the attack. This criterion can be rated as "none", "low" if privileges that allow performing basic user operations are required (e.g. changing settings), or "high" for administrative privileges. We note that this criterion is relative to the covert channel's sending-end, concealed in the victim's environment.

C10 **Remediation level** accounts for potential countermeasures. This criterion can be rated as "unavailable", "workaround" for non-official mitigation, "temporary fix" for official but not permanent countermeasures, or "official fix" otherwise.

C11 **Hardware configuration** specifies the attacker's proximity with regard to the victim's VM. Covert channels can require both VMs to be scheduled on the same core, on the same processor, or on the same system. Accordingly, hardware configuration can be rated as "core", "processor", or "system". A "system" rating makes for a higher severity score.

C12 **Initialisation** evaluates whether a covert channel attack requires the sender and receiver to perform an initialisation phase before leaking the victim's data. This criterion can be rated as "mandatory" or "optional". In the latter case, the covert channel remains functional in the absence of an initialisation phase, which increases the severity score. The absence of initialisation eases the deployment of the attack and decreases visible side-effects (e.g. large memory footprint).

The confidentiality, integrity, and availability requirements (CR, IR, AR) allow tuning the CVSS evaluation depending on the targeted asset. In the case of a cloud platform, the three requirements are equally important. Therefore, we set these to "medium", i.e. their default value. Furthermore, the availability impact is rated to "none". Covert channel attacks do not aim to compromise the availability of a computing environment.

4 Evaluation of Generic Criteria (C1-C8)

This section discusses the criteria for which the rating can be applied generically to the cross-VM covert channel attacks surveyed in Table 1.

C1. The attack vector is "local". The sending-end is a malicious program running inside the instance of the victim. This trojan must be inserted either using social engineering, or by corrupting the machine image. Independently of the chosen attack vector, user interaction is required. The attack vector is further discussed under requirement **C3**.

C2. Attack complexity is rated as "high". Prior to launching the attack, the adversary must achieve VM co-location, independently of the covert channel considered. Cloud services' application programming interfaces do not allow an

attacker to place an instance at will on a chosen physical machine. VM co-location consists in moving the attacker's VM until it is executing on the same hardware platform as the victim's. Several proposals suggested using networking utilities to map the internal network topology of the data center, allowing an attacker to place two instances on the same platform [22,10,33,25]. Microarchitectural covert channels can later be used to find out whether co-residency is achieved at core-level, package-level, or system-level. While these approaches require some knowledge of the network topology, an adversary can choose instead to directly apply microarchitectural covert channels to detect co-residency. In a purely microarchitectural co-residency attack, the sending-end can broadcast messages on the covert channel, until a receiving-end picks up. Thus targeted co-residency is still possible without access to a reliable network topology of the data centre. Recently, Atya et al. [7] successfully demonstrated this approach on AWS EC2, using the memory bus and the cache as communication mediums.

C3. User interaction is rated as "required". The Amazon Web Service Elastic Compute Cloud (AWS EC2) service is a practical example of means to compromise a victim's instance before its deployment. Amazon Machine Images (AMIs) are the basic unit of the EC2 service. An AMI contains the OS along with libraries, applications, and other components which personalise the instance. Before deploying a VM, a user must choose an AMI, and set permissions for its AWS account(s). AMI selection presents a unique vulnerability: anyone with an AWS account can customise and share an AMI. As a result, an attacker can conceal and distribute a trojan across a large pool of users. While Amazon warns its customers against such practise, it doesn't forbid it. Also, because the AMI contains a tremendous amount of code, it is extremely difficult (if not impossible) to uncover malicious code once it is embedded into the image. Whether trojan insertion is performed using social-engineering, or via machine image corruption, specific actions must be performed by the victim.

C4. The scope metric is rated as "unchanged". The mechanism responsible for enforcing access control over the vulnerable component, also known as the *security authority*, depends on the form of the trojan. For example, if the sending-end is part of a user application (e.g. plugin), the vulnerable component is the affected application (e.g. web-browser) and the security authority is the guest operating system, responsible for enforcing isolation between user applications. However, the covert channel attack does not allow accessing the data of other applications running in the same guest operating system. The same reasoning holds if the sending-end takes the form of a malicious kernel module. The affected component becomes the guest operating system, and the security authority becomes the hypervisor. The sending-end would be able to leak all the information of the guest operating system, but it would not allow accessing the data of other guests under the same hypervisor. Therefore, the fact that data is exfiltrated across virtual machines does not constitute a change of scope. The sole purpose of a covert channel attack is to exfiltrate information, or carry out modifications as instructed by the other communicating entity. Any exploit built on top of

the covert channel attack (e.g. privilege escalation) is beyond the scope of this study.

C5. Confidentiality impact is rated as "high". Covert channels intend to leak a selected amount of information rather than the entire set of system files. However, a successful attack against a public cloud instance can have a significant impact on a victim, such as theft of proprietary information, leakage of personal data, or theft of cryptographic keys. We note that covert channels are only relevant when there is no alternative mean of leaking information in a non-conspicuous manner, e.g. to avoid generating network traffic and associated logs [5]. As such, covert channels constitute an ideal basis for advanced persistent threats, where the attacker employs state-of-the-art techniques in order to maintain long-term intrusion and data exfiltration capabilities. Such an attacker has other incentives than simple financial gain [12].

C6. Integrity impact is rated as "low". The attacker can issue modifications to be applied to the victim's environment, although this requires bi-directional communication, as well as the ability to instruct data tampering operations. Such a covert channel was demonstrated by Maurice et al. [17], who managed to establish a rogue SSH connection between two AWS EC2 instances. Data modification is therefore possible, however it remains a specific case, the primarily objective being data extraction.

C7. Exploit code maturity is rated as "proof-of-concept". The state-of-the-art covert channels surveyed in this paper all demonstrate a functional attack in a virtualised environment. However, researchers rarely disclose their full source-code. Therefore, current microarchitectural covert channels are not directly applicable without a skilled attacker.

C8. As per the CVSS specification [3], disclosure of an exploit in external events such as publications automatically rates the report confidence as "confirmed". A research publication is considered an official source which is corroborated by multiple experts.

To the best of our knowledge, there hasn't been any reported exploit related to cross-VM microarchitectural covert channels. Therefore, a universal approach is adopted in this study. We note that the CVSS v3.1 also provides a set of modified base metrics, allowing the analyst to override base metrics so as to fit the victim's environment specifically. For instance, if the data that was leaked was not considered sensitive, the confidentiality impact can be overwritten to "low". Similarly, if the covert channel allowed modifying data used in critical decision making processes, the integrity impact can be overwritten to "high".

5 Evaluation of Covert Channel-Specific Criteria (C9-C12)

In this section, covert channel attacks surveyed in Table 1 are analysed individually in order to proceed with the criteria evaluation. Results are reported in

Table 3, along with the CVSS and CCSS scores for each attack. We note that the two scores are independent from each other. Additional details on scoring are provided in Section 6.

5.1 Memory Order Buffer

The memory order buffer (MOB) attack [24] exploits a side-effect of write-after-read hazards, called 4k-aliasing. This effect occurs whenever the lower twelve bits of the addresses contained in the load and store registers match, i.e. there is a data dependency between the load and the out-of-order store. This causes the load operation to be re-issued, resulting in the load/store bandwidth to drop. Authors leverage 4k-aliasing to create a covert communication between two hyperthreads. The sender either fills the store buffer with page-aligned addresses to transmit a one, or empties the store buffer to transmit a zero. Concurrently, the receiver probes load operations on every page-aligned addresses. When the load/store bandwidth drops, the receiver will observe a higher latency.

This effect is exploitable only at the thread-level, as it is linked to the load and store buffers located within the CPU. Neither the sender nor the receiver processes require root privileges, and the covert channel works across processes. Therefore, the sending-end can be embedded into a different program than the receiving-end. Both entities need to be scheduled on the same physical core.

With regard to countermeasures, authors acknowledge that disabling SMT is a straightforward way of mitigating the vulnerability. However, they also argue that "hyperthreading is expected to become more popular on IaaS platforms in the near future in order to keep them affordable". Indeed, SMT remains available on dedicated instances or for general-purpose workloads.

5.2 Last-Level Cache

LLC-based covert channels [22,32,14,16,17] derive from the Prime+Probe technique [18]. The receiver initialises the cache by filling it with its own cache lines, waits for the sender to execute, and probes its accesses to the same cache lines. If the sender chooses to modify the cache sets of the receiver, the latter will experience a slower access to its cache lines. Prime+Probe relies on the existence of congruent addresses between the sender and receiver, i.e. virtual addresses that map to the same cache set.

Identifying congruent addresses requires translating virtual pointers into physical addresses, which is performed by accessing the privileged page tables. Alternatively, entities can use the page offset of *huge pages* (e.g. 2 MB) as it is not translated, and it is long enough to include index bits. The communicating entities need to agree on a set of congruent addresses, which cannot be performed in the absence of an existing communication channel. In order to cope with this issue, Maurice et al. [17] suggested using a jamming agreement. Independently of the chosen strategy, LLC-based attacks are not functional without an initialisation phase. LLC-based covert channels are limited to cross-core communication.

Several cloud-oriented mitigation techniques were proposed to tackle LLC-based timing channels, such as cache partitioning or noise injection [13,9,11,26]. Intel Xeon processors support a similar mechanism, i.e. Intel's Cache Allocation Technology [4], which allows locking down portions of the LLC during execution, and ultimately defeat PRIME+PROBE attacks [13]. With a different approach, an auditing technique is suggested by Zhang et al. [34] which consists of using the performance monitoring unit to detect abnormal behaviour.

5.3 DRAM Row-Buffer

The DRAM addressing covert channel [21] exploits the DRAM bank row-buffer to create timing variations on uncached memory accesses. The sending-end allocates memory, and performs memory accesses either in the cache or in the DRAM. When the sender accesses the DRAM, it causes the row-buffer to be updated with the sender's row. Concurrently, the receiver accesses the same DRAM bank as the sender. If the sender evicted the receiver's row from the row-buffer, a row-miss occurs resulting in a higher latency.

Pessl et al. [21] relied on a privileged adversary model in order to access the pagemap file. We note that it is trivial to extend the original author's threat model to remote and unprivileged adversaries. One entity can simply write zeroes and ones on a random memory location, and the other entity scans its memory address space to detect the bit pattern, i.e. consecutive row-hits and row-misses. This approach also enables implementing a covert channel without knowledge of the DRAM addressing function, at the cost of an initialisation phase. This covert channel has the advantage that the communicating entities do not necessarily need to be scheduled on the same processor, as the DRAM memory is shared at a system-level via the interconnect.

Auditing can be used as a mitigation strategy. Indeed, the constant probing to DRAM results in a significant amount of cache-misses, observable by cache-miss counters. Alternatively, these authors proposed restricting access to the `clflush` instruction, which would render the covert channel harder to implement. Semal et al. [23] also suggest enforcing a close-page policy in order to inhibit the effect of the row-buffer.

5.4 Memory Controller

Semal et al. [23] proposed modulating the load on the channel scheduler in order to induce timing variations in the receiver's memory accesses to DRAM. The sender allocates three memory pages, and then reads one byte either in each of the three pages, or in a single page. The receiver observes a higher latency when the sender is increasing the load on the channel scheduler.

Authors demonstrated the attack both with and without privileges. The communicating-entities need to agree on a memory channel. As in the row-buffer attack, this can be achieved by having the sender broadcasting his position. Because the memory controller is accessible at a system level, this attack could be extended to multi-processor configurations. Further research is required to

evaluate the impact of accessing memory regions in external NUMA nodes on the latency variations induced by the sender.

The memory controller covert channel can be addressed with the same countermeasures as the row-buffer one, at the exception of the page policy. Alternatively, the controller can be redesigned in order to enforce temporal [27] or spatial isolation.

5.5 Memory Bus

Wu et al. [30] devised a covert channel based on the memory bus. Authors suggested using atomic operations on exotic memory operations, i.e. operations on cache line-crossing memory regions, in order to trigger a bus lock emulation. The sender either performs an exotic access, or remains idle. Meanwhile the receiver probes its uncached memory accesses. A high latency is observed whenever the sender accesses exotic memory regions.

This attack allows cross-core communication on NUMA architectures, cross-processor communication on front side bus architectures, and it does not require privileges. Furthermore, it is functional without an initialisation phase.

Wu et al. suggest monitoring the cache-miss memory bus lock counters in order to detect performance anomalies at runtime with minimal overhead. Another suggestion consists of enforcing a policy where each tenant can only be neighbour with one other tenant [30]. This approach renders covert channel attacks almost impractical, however the operational cost remains an open-question.

5.6 Summary of Findings

C9. The CVSS v3.1 [3] specifies that exploits which rely on social engineering can be rated as "none". However, the works of Ristenpart et al. [22] and Xu et al. [32] require accessing page tables in order to find congruent addresses, and are thus rated as "high". All remaining covert channels are feasible from a user-level program. These are rated as "none".

C10. The remediation level varies depending on the party that is enforcing countermeasures. Table 2 shows that among the countermeasures proposed in the literature, several rely on an alternative hardware design. This approach has the benefit of being the most efficient, however it is also the hardest to deploy. Furthermore, security by design tends to have a significant performance cost which is not always justified. For instance, Wang et al. [27] suggested a new design of the memory controller which enforces temporal isolation among different security domains. While effective, this technique results in performance cost of up to 150%. As a result, remediation strategies consisting of alternative designs are evaluated as "unavailable".

Other countermeasures have been proposed which can be taken directly by the cloud customer. For example, Zhang et al. [35] devised the *HomeAlone* technique which allows cloud users detecting the presence of a LLC-based timing channel. The victim continuously probes memory accesses to detect anomalies,

Table 2. Remediation level (**C10**) criterion analysis.

C10	Hardware manufacturer	Cloud provider	Cloud customer
Workaround	-	Software partitioning [11], [8], [9]; Noise injection [29], [26]; Auditing hardware counters [34]	Probing memory accesses [35]; Auditing hardware counters [30]
Unavailable	Temporal isolation [27]; Spatial isolation [19], [20], [29], [13]; Restricting `clflush` [21]; Close-page policy [23]	VM clusters [30]	-
Temporary fix	-	Disabling SMT [15]; Dedicated instances [1]	-

and takes reactive measures accordingly. Yet, this approach can result in a high number of false positives depending on the workload. This type of strategy is not official and cannot be generalised to all IaaS users. Therefore, remediation level is rated as "workaround" at the cloud customer level.

The most practical means of deploying countermeasures is if they are enforced by the cloud provider. The AWS EC2 and GCE services propose a type of instance where the user runs on a platform that is isolated from other users [1,6]. Note that these have a significant cost, e.g. an on-demand EC2 a1.2xlarge instance costs 0.204 USD per hour while a dedicated EC2 a1.2xlarge instance costs 2.2162 USD per hour. This approach is valid for running a selected workload only. Cloud providers have reportedly encouraged the disabling of SMT in order to prevent core-level timing channels [15]. We rate these strategies as "temporary fix", as they are recommended by vendors but are only applicable to a set of instances. Researchers also advanced mitigation strategies which can be implemented via the hypervisor. For example, Liu et al. [13] leverage Intel's Cache Allocation Technology to thwart Prime+Probe cache attacks. To the best of our knowledge, this strategy is not applied by cloud providers. As a result, this type of remediation is rated as "workaround".

C11. Ristenpart et al. [22] and Xu et al. [32] use a busy-loop mechanism to synchronise receiver and sender, implying that both VMs share CPU resources. Therefore, these attacks do not meet the requirements for cross-core covert channels. Similarly, the memory order buffer attack [24] requires both entities to share CPU resources. These attacks are set to "core". Remaining LLC-based covert channels are bound to the "processor" rating as the LLC cannot be shared across processors. Note that, as defined in Section 2.2, we use the term processor to refer to the entire processor die. Furthermore, while the memory controller attack [23] exploits a system-level component, the authors didn't demonstrate the attack on a multi-processor system. Similarly, Wu et al [30] assigned different virtual CPUs to each entity without specifying whether these were pinned to dis-

Table 3. Summary of cross-VM covert channel attacks' criteria evaluation and scoring.

Attack	**C9**	**C10**	**C11**	**C12**	CVSS / CCSS
[22]*	*Privileged*	*Workaround*	*Core*	*Mandatory*	4.2 / 1.6
[32]*	*Privileged*	*Workaround*	*Core*	*Mandatory*	4.2 / 1.6
[30]†	*Unprivileged*	*Unavailable*	*Processor*	*Optional*	5.0 / 6.7
[14]*	*Unprivileged*	*Workaround*	*Processor*	*Mandatory*	4.9 / 4.3
[16]*	*Unprivileged*	*Workaround*	*Processor*	*Mandatory*	4.9 / 3.7
[21]¶	*Unprivileged*	*Unavailable*	*System*	*Mandatory*	5.0 / 6.8
[17]*	*Unprivileged*	*Workaround*	*Processor*	*Mandatory*	4.9 / 3.8
[24]‡	*Unprivileged*	*Temporary fix*	*Core*	*Optional*	4.8 / 5.7
[23]§	*Unprivileged*	*Workaround*	*Processor*	*Mandatory*	4.9 / 4.7

*LLC, †Memory bus, ¶Row-buffer, ‡Memory order buffer, §Memory controller.
C1 = *Local*, **C2** = *High*, **C3** = *Required*, **C4** = *Unchanged*, **C5** = *High*, **C6** = *Low*, **C7** = *Proof-of-concept*, **C8** = *Confirmed*.

tinct hardware processors. These covert channels are also rated as "processor". Finally, the DRAM row-buffer [21] attack can transmit data across processors as DRAM memory is shared at system-level. As such, it is rated as "system".

C12. Only the memory bus [30] and the memory order buffer [24] covert channels can be rated as "optional". Every other covert channel requires an initialisation phase, and are thus rated as "mandatory" for this criterion.

6 Severity Scores

6.1 Design of the CCSS Equations

In order to provide a classification of covert channels, we create a new scheme which accounts for criteria **C9**, **C10**, **C11**, **C12**, and the channel capacity. These criteria are specific to the covert channel considered, and provide a point of comparison for the adversary model. The CCSS is by no means a representation of the severity of the attack. Instead, it should be taken as a complement to the CVSS which cannot solely be used to classify cross-VM covert channels.

Criteria scores have been selected such that the scoring equation is as uniform as possible. That is, the five criteria all have the same weight. The motivation behind this decision is that the importance of one factor over another is subjective. For example, one could give a higher weight to the channel capacity, arguing that communication speed and robustness is the most important. From one perspective, this is true. A set of log, data, and application files of a password manager (∼1 GB) would take 99 days 10 hours and 5 minutes at a bitrate of 1 Kb/s to be transmitted, and 2 hours and 23 minutes at a bitrate of 1Mb/s. Cloud instances are rescheduled onto different platforms depending on resource availability and demand. Therefore, the communication speed is critical. However, from another perspective, this is false. Faster communication rates are usually achieved by covert channels that exploit microarchitectural components closer

from the execution units, which can easily be addressed by existing countermeasures (i.e. disabling SMT), or that have been extensively studied and resulted in multiple countermeasure proposals [8,9,11,13,26,28,29,36]. Thus faster covert channels will not necessarily be practical.

Furthermore, improving an evaluation scheme is usually performed over time by comparing the scores with the reality. For instance, the HeartBleed vulnerability was given a medium severity score of 5.0 in CVSS v2. Yet, it could easily be exploited and had significant consequences. It now has a high severity score of 7.5 in the CVSS v3.1 To the best of our knowledge, no covert channel exploit has been reported so far. Therefore, we consider that starting with an impartial scoring equation for the CCSS is the best approach. The scoring equation is,

$$Score = 2 \times (\mathrm{C9} + \mathrm{C10} + \mathrm{C11} + \mathrm{C12} + CapScore) \tag{1}$$

Each CCSS criteria is given a value between 0 and 1: criterion **C9** is scored 0 for "Privileged" and 1 for "Unprivileged"; criterion **C10** is scored 0 for "Temporary fix", 0.5 for "Workaround", and 1 for "Unavailable"; criterion **C11** is given a score of 0 for "Core", 0.5 for "Processor", and 1 for "System"; **C12** is scored 0 for "Mandatory" and 1 for 'Optional"; the capacity score $CapScore$ is modelled as an affine function between the highest and lowest channel capacity observed in this study, such that it outputs a score between 0 and 1,

$$CapScore = 1/(1.06e06 - 1.77) \times Capacity \tag{2}$$

The final CCSS score varies between 0 and 10. Again, the CCSS score is only used for comparing covert channel attacks, and is complementary to the CVSS.

6.2 Results

Figure 2 represents the score of each covert channel under the CCSS and the CVSS. Due to missing information, Ristenpart et al.'s attack [22] was assigned an error rate of 22%, i.e. the maximum error rate observed in this study. Highest

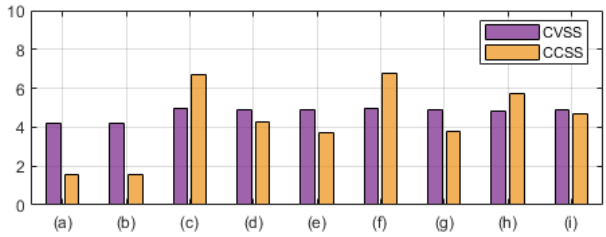

Fig. 2. Scoring of cross-VM covert channel attacks under the CVSS and CCSS: (**a**) = LLC [22], (**b**) = LLC [32], (**c**) = Memory bus [30], (**d**) = LLC [14], (**e**) = LLC [16], (**f**) = Row-buffer [21], (**g**) = LLC [17], (**h**) = Memory order buffer [24], (**i**) = Memory controller [23].

scores are achieved by the memory bus [30] and DRAM row-buffer [21] covert channels. Although not the most recent, these were able to reach high-speed effective communication rates while minimising operational constraints. Meanwhile, LLC-based covert channels tend to achieve lower severity scores, due to the necessity of finding congruent addresses as well as the LLC locality. This shows that future works should emphasise on exploiting system-level resource, while working on making the communication more robust.

The CVSS scores were computed with the CVSS v3.1 equations [3]. According to our study, microarchitectural covert channels achieve a medium severity score ranging from 4.2 to 5.0. It shows that covert channels in IaaS are practical, that they should not be overlooked, and that suitable countermeasures should be devised in the short term in order to tackle timing channel vulnerabilities. More specifically, we suggest addressing the DRAM row-buffer and memory bus covert channels, as cache-based covert and side channel attacks have already been extensively studied.

When comparing the two evaluation frameworks, we observe that the CCSS outlines disparities among covert channel attacks which the CVSS does not. For example, the works proposed by Ristenpart el al. [22] and Wu et al. [30] would both be rated as medium severity vulnerabilities under the CVSS. Yet, the former attack has significant shortcomings including obtaining privileges (**C9**), achieving core-level co-location (**C11**), finding congruent addresses (**C12**), and a low communication speed. Thus the resulting CVSS scoring of the covert channel proposed by Ristenpart et al. [22] as a medium severity vulnerability is not adequate. In comparison, the proposed evaluation framework successfully highlights the benefit of one covert channel over another, with respective scores of 1.6 and 6.7. This shows that the evaluation of microarchitectural covert channels cannot be performed entirely based on the current industry standard, and that the criteria studied in the CCSS should be accounted for when devising new cross-VM covert channel attacks. Authors of future covert channels are encouraged to use the CCSS in order to provide a fair and realistic point of comparison with other works.

7 Conclusion

In this paper, we proposed the Covert Channel Scoring System (CCSS) as a new framework for evaluating microarchitectural covert channels. It allows comparing covert channel attacks based both on their performance (speed and robustness) and their practicality (operational constraints). The analysis revealed that the fastest covert channels are not necessarily the most eminent attacks, as they usually assume a close locality between sender and receiver, or a complex initialisation phase, resulting in lower severity scores. We advocate future works to use the CCSS in order to include a fair comparison metric in their proposal.

Furthermore, we systematically evaluated microarchitectural covert channels in the Infrastructure-as-a-Service ecosystem, using the Common Vulnerability Scoring System in its latest version (CVSS v3.1), and revealing medium severity

scores ranging from 4.2 to 5.0. In comparison, the OpenSSL Heartbleed vulnerability achieved a severity score of 7.5 (CVSS v3.1). Although not as severe, the microarchitectural covert channel threat to IaaS is present and not negligible. In parallel, services built on cloud computing continue offering guarantees on the confidentiality and integrity of their customers' data. The loss of data, e.g. under GDPR requirement, could result in dramatic consequences for the cloud provider, the software provider, and their customers.

References

1. Amazon EC2 dedicated instances. bluehttps://aws.amazon.com/ec2/pricing/dedicated-instances/, last accessed 25 Jul 2020
2. CVE-2013-0375 detail. bluehttps://nvd.nist.gov/vuln/detail/CVE-2013-0375, last accessed 25 Jul 2020
3. CVSS v3 Equations. bluehttps://nvd.nist.gov/vuln-metrics/cvss/v3-calculator/equations, last accessed 25 Jul 2020
4. Improving real-time performance by utilizing cache allocation technology, Intel Corporation (2015)
5. Monitoring your instances using CloudWatch. bluehttps://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-cloudwatch.html, last accessed 25 Jul 2020
6. Sole-tenant nodes. bluehttps://cloud.google.com/compute/docs/nodes, last accessed 25 Jul 2020
7. Atya, A.O.F., Qian, Z., Krishnamurthy, S.V., La Porta, T., McDaniel, P., Marvel, L.M.: Catch me if you can: A closer look at malicious co-residency on the cloud. IEEE/ACM Transactions on Networking **27**(2), 560–576 (2019)
8. Cock, D., Ge, Q., Murray, T., Heiser, G.: The last mile: An empirical study of timing channels on sel4. In: ACM CCS. pp. 570–581 (2014)
9. Godfrey, M.M., Zulkernine, M.: Preventing cache-based side-channel attacks in a cloud environment. IEEE TCC **2**(4), 395–408 (2014)
10. Herzberg, A., Shulman, H., Ullrich, J., Weippl, E.: Cloudoscopy: Services discovery and topology mapping. In: ACM CCSW. pp. 113–122. ACM (2013)
11. Kim, T., Peinado, M., Mainar-Ruiz, G.: STEALTHMEM: System-level protection against cache-based side channel attacks in the cloud. In: USENIX Security. pp. 189–204 (2012)
12. Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. IEEE S&P **9**(3), 49–51 (2011)
13. Liu, F., Ge, Q., Yarom, Y., Mckeen, F., Rozas, C., Heiser, G., Lee, R.B.: Catalyst: Defeating last-level cache side channel attacks in cloud computing. In: IEEE HPCA. pp. 406–418. IEEE (2016)
14. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B.: Last-level cache side-channel attacks are practical. In: IEEE S&P. pp. 605–622. IEEE (2015)
15. Marshall, A., Howard, M., Bugher, G., Harden, B., Kaufman, C., Rues, M., Bertocci, V.: Security best practices for developing windows azure applications. Microsoft Corp p. 42 (2010)
16. Maurice, C., Neumann, C., Heen, O., Francillon, A.: C5: cross-cores cache covert channel. In: DIMVA. pp. 46–64. Springer (2015)
17. Maurice, C., Weber, M., Schwarz, M., Giner, L., Gruss, D., Boano, C.A., Mangard, S., Römer, K.: Hello from the other side: SSH over robust cache covert channels in the cloud. In: NDSS. vol. 17, pp. 8–11 (2017)

18. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: the case of AES. In: CT-RSA. pp. 1–20. Springer (2006)
19. Page, D.: Partitioned cache architecture as a side-channel defence mechanism (2005)
20. Percival, C.: Cache missing for fun and profit (2005)
21. Pessl, P., Gruss, D., Maurice, C., Schwarz, M., Mangard, S.: DRAMA: Exploiting DRAM addressing for cross-cpu attacks. In: USENIX Security. pp. 565–581 (2016)
22. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: ACM CCS. pp. 199–212. ACM (2009)
23. Semal, B., Markantonakis, K., Akram, R.N., Kalbantner, J.: Leaky controller: Cross-VM memory controller covert channel on multi-core systems. EasyChair Preprint no. 2941 (EasyChair, 2020)
24. Sullivan, D., Arias, O., Meade, T., Jin, Y.: Microarchitectural minefields: 4k-aliasing covert channel and multi-tenant detection in IaaS clouds. In: NDSS (2018)
25. Varadarajan, V., Zhang, Y., Ristenpart, T., Swift, M.: A placement vulnerability study in multi-tenant public clouds. In: USENIX Security. pp. 913–928 (2015)
26. Vattikonda, B.C., Das, S., Shacham, H.: Eliminating fine grained timers in Xen. In: ACM CCSW. pp. 41–46 (2011)
27. Wang, Y., Ferraiuolo, A., Suh, G.E.: Timing channel protection for a shared memory controller. In: IEEE HPCA. pp. 225–236. IEEE (2014)
28. Wang, Y., Ferraiuolo, A., Zhang, D., Myers, A.C., Suh, G.E.: SecDCP: secure dynamic cache partitioning for efficient timing channel protection. In: DAC. pp. 1–6 (2016)
29. Wang, Z., Lee, R.B.: New cache designs for thwarting software cache-based side channel attacks. In: ISCA. pp. 494–505 (2007)
30. Wu, Z., Xu, Z., Wang, H.: Whispers in the hyper-space: high-bandwidth and reliable covert channel attacks inside the cloud. IEEE/ACM Transactions on Networking **23**(2), 603–615 (2014)
31. Xiao, Z., Xiao, Y.: Security and privacy in cloud computing. IEEE Communications Surveys & Tutorials **15**(2), 843–859 (2012)
32. Xu, Y., Bailey, M., Jahanian, F., Joshi, K., Hiltunen, M., Schlichting, R.: An exploration of L2 cache covert channels in virtualized environments. In: ACM CCSW. pp. 29–40. ACM (2011)
33. Xu, Z., Wang, H., Wu, Z.: A measurement study on co-residence threat inside the cloud. In: USENIX Security. pp. 929–944 (2015)
34. Zhang, T., Zhang, Y., Lee, R.B.: Cloudradar: A real-time side-channel attack detection system in clouds. In: RAID. pp. 118–140. Springer (2016)
35. Zhang, Y., Juels, A., Oprea, A., Reiter, M.K.: Homealone: Co-residency detection in the cloud via side-channel analysis. In: IEEE S&P. pp. 313–328. IEEE (2011)
36. Zhou, Z., Reiter, M.K., Zhang, Y.: A software approach to defeating side channels in last-level caches. In: ACM CCS. pp. 871–882 (2016)