Cloud Computing Security: Hardware-Based Attacks and Countermeasures

Reza Montasari*, Alireza Daneshkhah, Hamid Jahankhani, and Amin Hosseinian-Far

Hillary Rodham Clinton School of Law, Swansea University,
Richard Price Building, Sketty Ln, Sketty, Swansea, SA2 8PP, United Kingdom.
{montasarireza@gmail.com}
http://www.swansea.ac.uk
School of Computing, Electronics and Mathematics, Coventry University,
Priory Street, Coventry, CV1 5FB, United Kingdom.
{ac5916@coventry.ac.uk}
https://www.coventry.ac.uk/
Information Security and Cyber Criminology, Northumbria University,
110 Middlesex Street, London, E1 7HT, United Kingdom.
{hamid.jahankhani@northumbria.ac.uk}
http://www.london.northumbria.ac.uk
Faculty of Business and Law, University of Northampton,
Waterside Campus, University Drive, Northampton, NN1 5PH, United Kingdom.
{Amin.Hosseinian-Far@northampton.ac.uk}
https://www.northampton.ac.uk/

Abstract. Despite its many technological and economic benefits, Cloud Computing poses complex security threats resulting from the use of virtualisation technology. Compromising the security of any component in the cloud virtual infrastructure will negatively affect the security of other elements and so impact the overall system security. By characterising the diversity of cyber-attacks carried out in the Cloud, this paper aims to provide an analysis of both common and underexplored security threats associated with the cloud from a technical viewpoint. Accordingly, the paper will suggest emerging solutions that can help to address such threats. The paper also offers future research directions for cloud security that we hope can inspire the research community to develop more effective security solutions for cloud systems.

Keywords: Cyber security, Cloud security, Virtualisation, Hypervisor attacks, Network attacks, Cyber physical systems, Digital forensics, Computer security, Network security

1 Introduction

Cloud Computing (CC), still an evolving paradigm, has become one of the most transformative computing technologies and a key business avenue, following in

*

the footsteps of main-frames, minicomputers, personal computers, the World Wide Web and smartphones [29] [28] [37] [39]. CC is a shared collection of configurable networked resources (e.g., networks, servers, storage, applications and services) that can be reconfigured quickly with minimal effort [32] [27]. Its vital features have considerably reduced IT costs, contributing to its swift adoption by businesses and governments worldwide.

As a result, CC has drastically transformed the way in which Information Technology (IT) services are created, delivered, accessed and managed. Such a transformation, that offers many technological and economic benefits, has produced substantial interest in both academia and industry. However, despite all its benefits, CC poses numerous security threats with devastating consequences. As a result, many organisations do not move their business IT infrastructure completely to the cloud mainly due to the fears of cloud-related security threats. Some of these fears among others are due to the issues such as processing of sensitive data outside organisations, shared data and ineffectiveness of encryption, etc. [28] [18].

Considering its security requirements (confidentiality, integrity, availability, accountability, and privacy-preservability), this paper presents an analysis of security threats associated with CC. To this end, we identify both common and underexplored cyber-security attacks carried out in Cloud Computing environments (CCEs). Accordingly, we will also propose emerging solutions and lines of defence against each attack vector with a view to mitigating such threats. Our study will also provide insights into the future security perspectives related to the CC. This study only focuses on analysing the technical aspects of cyber-security threats in cloud. To this end, this analysis emphasises the complexity, intensity, duration and distribution of the attacks, outlining the major challenges in safeguarding against each attack. Investigating other security aspects such as organisational, compliance, physical security of data centers, and the way in which an enterprise can meet regulatory requirements is outside the scope of this paper. Similarly, providing an exhaustive list of attack vectors is outside the scope of this study. The remainder of the paper is structured as follow.

The remainder of the paper is structured as follow. Section 2 analyses attack vectors while Section 3 discusses countermeasures. The paper is concluded in Section 4.

2 Attack Vectors

There are certain vulnerabilities associated with computing hardware that attackers can exploit to launch destructive attacks which often go undetected by the existing software countermeasures against embedded device systems within cloud environments. One of the usual characteristics of a computer hardware component can be malfunction or some kinds of abnormal behaviour which can

result in providing a backdoor access to a potential adversary. The following sub-sections analyse the cyber-attacks that can be carried out due to physical hardware flaws.

2.1 Side-Channel Attacks

Side Chanel Attacks (SCAs) are a type of hardware-targeted attacks that is almost impossible to detect. SCAs are based on Side-Channel information that leak through a medium that is not intended for communications. This medium is called a Side-Channel. Side-Channel information can be acquired from an encrypted digital device. SCAs could also stem from a leakage produced by electronic circuits as by-products that render it possible for an adversary without access to circuit, itself, to determine how the circuit operates and what type of data it is processing (heat and electromagnetic leakage are both feasible sources of information for an adversary). SCAs can be very detrimental if proper defence mechanisms are not implemented on a target device. The three major types of side-channel attacks can be categorised based on the leaked information, including: time, trace [22] and access-driven [49]. All the three types acquire sensitive information by observing the execution time or power consumption variations produced through cache hits and misses. However, they tend to vary on the details of the captured information. The time-driven attack observes the aggregate profile and the total number of cache hits and misses. It can be of two types: passive if adversaries have no direct access to the victims machine or active if there is a physical access to the machine [33] [31] [30] [35]. The following sub-sub sections provide an analysis of the variants of SCAs.

Prime+Probe Attacks Through a Prime+Probe Attack, a variation of CB-SCAs, the adversary could potentially attain co-residency and perform load measurement. The theft of sensitive information can be accomplished by exploiting three probable channels: pre-emptive scheduling, hyper-threading, and multi-core. In the first channel, the adversary exploits the context switch between his VM and the victims VM to observe the cache status as the victim had left it. In the second channel, the malicious operation is performed by breaching the CPU core sharing. In this case, the attacker exploits the multi-tenancy, realised with multiple threads running on a single processor. The third stage involves reading the L3 cache that is the only one shared when VMs are allocated to multiple cores instead of multiple threads [22]. Furthermore, by performing a Prime+Probe Attack, the attackers could extract an RSA secret key from a co-located instance in CCEs [19]. A Prime+Probe attack can also be launched against different processor caches such as the L1 data cache, L1 instruction cache and the branch prediction cache. Similarly, adversaries could also exploit Prime+Probe for LLC attacks by leveraging hardware elements that are beyond the control of the CSPs but activated in the VMM for operation reasons [26].

Time-Driven Attacks In a Time-Driven Attack, the attackers extract cipher keys by exploiting side-channel information vulnerabilities triggered by the execution of cryptographic algorithms and data-dependent behaviour of cache memory. A Time-Driven Attack determines the run times of victim processes by exploiting the connection between the secret key and the number of cache misses which in turn establishes the runtime to infer the key. Regardless of the differences between these approaches, the Cache-Based Timing Attacks rely upon the impacts that the number of cache misses have on the execution time of an encryption process. Furthermore, Time-Driven Attacks can be performed against AES in a virtualisation setting [10] [17] [9] [2] [36] [22]. An example of such an attack is the PikeOS Microkernel Virtualization Framework (Wei et al., 2014), which was can be mounted against AES on an actual CPS. Similarly, Address Space Layout Randomization (ASLR), a security technique used to prevent exploitation of memory corruption vulnerabilities, can be bypassed by applying the branch-target buffer.

It is also possible to locate the place in the kernel where codes were run based on the mapping from virtual addresses to the branch-target buffer cache [12]. In addition, a malicious operating system could potentially reverse-engineer the control flow of SGX enclaves via branch-prediction analysis or branch shadowing [25]. Likewise, adversaries might be able to mount Timing Attacks against secret-dependent data access patterns on the sliding-window modular exponentiation implementation [26] [1] [8]. The Scatter-Gather technique, a commonly-implemented method to stop Time-Based Attacks, can also be exploited through a variation of Timing Attack called CacheBleed [48], which takes advantage of cache-bank collisions [13](Intel, 2016) to generate quantifiable timing differences [14]. In general, Time-Driven Attacks are simple to execute since they require less leaked information [6] [2].

Access-Driven Attacks In cases when multi tenancy is employed through Hyper-Threading method, information on cypher algorithms as RSA and AES could potentially be observed by the attackers. One of the most recent attacks is the Access-Driven Attack, via which the attacker can control the cache sets that the cipher process changes. For instance, the attacker is likely to be able to determine which aspects of the lookup tables have been accessed by the cipher. Another type of attack is Boot Integrity Attack, in which adversaries with either logical or physical access are likely to be able to damage boot integrity with bootkits or particular form of malware that exists outside the OS (e.g. within System Management Mode (SMM)).

Considering its location, this kind of malware is predominantly threatening as it can reinfect new OS installations. Susceptibilities have been discovered in the BIOS, UEFI, Master Boot Record (MBR), CPU Management Engines and PCI device option ROMs [46]. Access-driven attacks exploit the connection between the secret key and the cache use of a crypto process. Because the cache is divided

between various processes, an adversary might be able to gain the cache usage of the victim process by monitoring a carefully created process, which executes together with the victim process [24].

2.2 Cache-Based Attacks

Confidential data can be safeguarded against unauthorized access by storing it in an encrypted form and transmitting it over encrypted channels. However, at some point, data need to be decrypted so as to perform the computation. Adversaries could potentially exploit the multi-tenant environment to gain access to physical resources such as memory bus, disk bus, and data and instruction caches in which they can locate decrypted data and the cryptographic keys of well-known algorithms (AES, DES, RSA) and of other VMs instances. An instance of this concerns the shared memory hierarchy of an Intel Pentium 4 with hyper-threading features. Both the L1 and L2 caches with the hyper-threading feature turned on can leak information from one process to the other [33] [31] [41]. This is called a Cache-Based Side-Channel Attack (CBSCA) and is part of a family known as Cross-VM Side-Channel Attacks. This attack evades the logical isolation provided by the hypervisor layer and can be launched by two types of malicious actors including: insider attackers (often cloud employees abusing their privileged position) and malicious customers (that in a first phase must land in the victim server and then initiate the attack).

2.3 Flush+Reload Attacks

By exploiting resource sharing features in virtual environments, adversaries will be able to carry out cross-VM Flush+Reload Attacks against VMs in a hypervisor such as VMware [20]. As a result, they could potentially extract an AES keys in OpenSSL 1.0.1 running inside a victims VM. Likewise, shared memory controllers are susceptible to Flush+Reload Attacks that could exploit memory interferences as timing channels [44]. Similarly, covert channels shared between processor resources could be exploited to facilitate secret communication between malign processes. Trojans and spies could be utilised to compromise Processor Branch Prediction Units [11]. By exploiting Branch Predictor conflicts, adversaries could establish covert channels enabling them to launch Flush+Reload attacks against Computing hardware devices.

2.4 Rowhammer Attacks

If a particular row of a Double Data Rate (DDR) memory bank is constantly activated (opened) and pre-charged (closed) within a Dynamic Random-Access Memory (DRAM) refresh interval, one or more-bit flips take place in physically adjacent DRAM rows to an incorrect value. Such disturbance is known as

Rowhammer [23]. An advanced attacker can exploit the Rowhammer to compromise the DRAM of a computing device. This occurs by evading the defence mechanisms often deployed through traditional security software and features such as memory isolation to conduct the memory disturbance attack. Similarly, a Rowhammer Fault Injection, a recently discovered real-time Microarchitectural Attack, can be launched remotely to gain full access to the DRAM of a CC device. A Rowhammer Attack can pollute system memory, access and alter sensitive data and gain full control of the system.

2.5 Hardware Threading Attacks

Attackers can also exploit hardware threading to examine a competing threads L1 cache usage in real time [14] [36]. Simultaneous multithreading (the sharing of the operation resources of a superscalar processor between multiple execution threads) is a feature implemented into Intel Pentium 4 processors. Under this implementation, the sharing of processor resources between threads spreads beyond the operation units. This denotes that the threads also share access to the memory caches. Such shared access to memory caches can facilitate side channels and enable a malign thread with restricted privilege to scan the operation of another thread. In turn, this results in allowing the attackers to steal cryptographic keys [15] [26] [36]. Additionally, by exploiting side-channel information based on CPU delay, adversaries could potentially mount TBSCAs against the Data Encryption Standard (DES) implemented in some applications. Such cryptanalysis technique applies side-channel information on encryption processing to gather plaintexts for cryptanalysis and infers the information on the extended key from the acquired plaintexts [42]. Through this attack, the adversary will be able to break the cipher with plaintexts [33] [31] [30].

2.6 Data Loss and Data Breach

Data stored in the cloud could be lost because of the hard drive failure, its accidental deletion by CSPs or malicious modification by adversaries, etc. Data loss can have disastrous impacts on enterprises such as bankruptcy. A data breach occurs when a VM accesses data from another VM on the same physical host (when the tenants of the two VMs are different customers). For example, a data breach can be carried out through a Side-Channel Attack, in which adversaries could potentially access data from one VM through another by utilising their shared components such as processors cache.

3 Countermeasures

Shared Technology: Cloud Security Alliance (CSA) (Hubbard and Sutton, 2010) recommends a defence in depth strategy that should include compute, storage,

and network security enforcement and monitoring. According to recommendation, CSPs could deploy robust compartmentalization to ensure that individual customers do not affect the operations of other tenants running on the same CSP. This denotes that customers must not be able to have access to any other tenants actual or residual data, network traffic, etc. Data Loss and Data Breach: Thus, one of the most effective ways to safeguard against data loss is to have in place a proper data backup, which resolves data loss issues.

3.1 Side-Channel Countermeasures

The purpose of a SCA countermeasure must be to hide the leakage or reduce it so that it holds minor or no valuable information against which an adversary is motivated to launch an attack. One of the most widely used defence mechanisms against a SCA relies on rendering the security operation time delay constant or random irrespective of the microarchitecture components utilised [14]. However, implementing constant-time execution code is difficult because optimisations presented by the compiler must be circumvented. Therefore, dedicated constant-time libraries have been introduced to enable security developers to safeguard their applications against SCAs. Hardware partitioning can also be used as a countermeasure to safeguard against SCAs. This hardware partitioning must be based on inactivating hardware threading, page sharing, presenting Hardware Cache Partitions, quasi-partitioning, and migrating VMs within cloud services. Considering that SCAs exploit physical elements of a system, its countermeasures must take the approach of enhancing the security of the system design and development such as that of cache architectures. These cache mechanisms should be offered without vast performance costs.

Furthermore, efficient implementation of Advanced Encryption Standard (AES) algorithm in hardware could be utilised as a defence mechanism against SCAs. There are currently few manufacturers implementing better hardware support in the design of their processor technologies to offer better constant-time cryptography operations. For instance, Intel has introduced AES New Instructions (AES NI), which is a new encryption instruction set that enhances on the AES algorithm and speeds up the encryption of data in the two categories of Intel Xeon processor and the Intel Core processor. AES-NI provides an advantage in relation to speed over other implementations. Moreover, since AES-NI, which consists of seven new instructions, was specifically developed to be constant-time, it provides a better protection against SCAs over some other software implementations.

3.2 Cache-Based Countermeasures

Configurable Cache Architecture could be used as a countermeasure to provide hardware assisted defence against CBSCA [45]. The cache is dynamically divided

into safeguarded regions and can be configured for an application. In partitioned caches, there is a section of the cache that is assigned exclusively to the safeguarded process so as to avert information leakage. Therefore, partitioned cache mechanism can be deployed as a line of defence against CBSCAs. A partitioned cache must be included in devices that are susceptible to SCAs to separate the cache behaviour of one process to another. This will prevent process interference by providing adequate space to store the entire S-box in cache (It will be locked when it is pre-loaded). Segregation does not permit forcible flushing of the cache; furthermore, partitioned cache employs longer cache lines that render attacks more problematic. Similarly, a method called Partition-Locked Cache (PLcache) [45] can be used to deal with cache sharing issues. This method will rely on a fine-grained locking control to isolate only the cache lines that contain important data. By making private partitions only those cache lines that are of interest are locked.

McBits [3] and Bitslice implementation of the AES [4] is another constant time countermeasure. By not using any lookup tables, this implementation could essentially prevent information from leaking out via a side channel. Another countermeasure is to carry out Cache Warming or Pre-Fetching. Time-Driven and Trace-Driven SCAs distinguish cache-miss and cache-hits. Eliminating this distinction can be a robust countermeasure [34]. So as to prevent information from the leakage, one needs to warm up the cache into which the Lookup Tables must be loaded prior to the runtime being initiated. In this situation, no cache misses will occur on condition that data is loaded to cache prior to the runtime. As a result, there will be no leakage of data.

3.3 Rowhammer Countermeasures

To perform a successful Rowhammer attack, adversaries must undertake four steps consisting of identifying the target device and its specific memory architecture characteristics, activating rows in each bank in a swift manner to trigger the Rowhammer vulnerability, accessing the aggressor physical address from userland [40] and exploiting bit flips [23]. In order to counteract a Rowhammer Attack, Rowhammer-induced bit flips must be blocked by altering DRAM, memory controllers or the combination of both. It is important that specific rows not be repeatedly triggered during a specific refresh point if the adjacent rows are not simultaneously refreshed. Targeted Row Refresh (TRR) mode and Maximum Activate Count (MAC) metadata field could also be used by a memory controller as countermeasures to safeguard against Rowhammer Attacks [21]. In the TRR, a memory controller would need the DRAM device to refresh a rows neighbours. In contrast, MAC metadata field specifies the number of activations that a given row can safely cope with before its neighbours require refreshing.

Another countermeasure against Rowhammer Attacks is to use the physical probing of the memory bus via a high-bandwidth oscilloscope. This can be

achieved by determining the voltage on the pins at the DIMM slots [38]. Furthermore, time analysis based on the rowbuffer conflict can be used to determine address pair that is part of the same bank and then apply this address set to rebuild the precise map function automatically [47] [38]. Furthermore, a simple solution requires that DRAM vendors build Rowhammer mitigations internally within a DRAM device, which does not need special memory controller support.

There exist various other methods that can be used to mitigate Rowhammer Attacks. For instance, by constantly refreshing the entire rows, disturbance errors could be eliminated for sufficiently short refresh intervals (RI RIth) [23]. This is despite the fact that regular refreshing might diminish performance and energy-efficiency. Furthermore, a mechanism called Probabilistic Adjacent Row Activation (PARA) Kim et al. (2014), which is implemented in the Memory Controller can also be utilised to prevent DRAM disturbance errors Manufacturers could also retire DRAM cells, identified as victim cells, and remap them to spare cells. The end-users, themselves, could also retire DRAM sells by assessing and utilising system-level techniques for deactivating faulty addresses or remapping defective addresses to reserved addresses [31] [23](Montasari, hardware Kim et al., 2014).

Authors in (Ghasempour et al., 2015) have also suggested a Run-Time Memory Hot Detector (ARMOR) to mitigate Rowhammer attacks. ARMOR is analogous to DRAM in that it is implemented at the memory level. According to the authors in (Ghasempour et al., 2015), ARMOR is capable of detecting all the conceivable Row Hammer errors, screening the activation flow at the memory level and also identify hot rows (specific rows) that might be hammered at run-time.

One method of detecting a Rowhammer attack is to implement the last-level cache counter facility to generate an interrupt after N misses (Aweke et al., 2016). This method involves monitoring the last-level cache misses on a refresh interval and row access with high temporal locality on certain processors such as Intel/AMD (Fournaris et al., 2017). In the event of missing cache surpassing a threshold, a selective refresh could be performed on the vulnerable row. Identifying the hot row (i.e. specific row or aggressor row) and refreshing its neighbouring rows is another technique to counteract a Rowhammer attack.

The CFLUSH command available on user space (userland) for x86 devices can also be used as a countermeasure to evict cache lines associated with the aggressor row addresses among its memory accesses [40] [23]. However, these countermeasures do not appear to be appropriate to deal with Rowhammer Attacks in CCEs. In the context of cloud, Rowhammer Attacks are executed on different attack interfaces (e.g. scripting language based attacks) by deploying web browsers that are activated remotely, a view supported by the authors in [16] [5].

Therefore, it is essential to develop new eviction methods to replace the existing flush instructions so that Rowhammer attacks within cloud environments can be addressed more effectively. These new methods must be able to iden-

tify an eviction set that would comprise of addresses which will be part of the same cache set of the aggressor rows. For instance, this can be accomplished by employing a Time Attack to identify the eviction set. Yet, another eviction method could be based on the reverse engineering analysis of the system that has come under attack [7]. However, this could be a complex task considering the modern Intel processors. Direct Memory Access methods could be utilised to bypass CPUs and their caches to address the Rowhammer attacks [43].

4 Conclusion

In this study, we identified and analysed both common and underexplored hardware-based attacks associated with CCEs. We then made several recommendations with a view to mitigating such attacks. This analysis was based on our own experience as well as various sources in the literature such as official documentations, white papers and existing research articles. As a future work, in order to realise the fullness of our recommended countermeasures, one could perform distinct studies related to the suggested methods. Practical assessments must be performed for each recommendation with a view to determining how effective the countermeasure against a given attack vector is and establish whether or not that mitigation mechanism can be bypassed. It is only by conducting these practical studies that we can truly provide adequate insight on the fullness of these countermeasures.

References

1. Onur Acıiçmez, Çetin Kaya Koç, and Jean-Pierre Seifert. Predicting secret keys via branch prediction. In *Cryptographers Track at the RSA Conference*, pages 225–242. Springer, 2007.
2. Daniel J Bernstein. Cache-timing attacks on aes. 2005.
3. Daniel J Bernstein, Tung Chou, and Peter Schwabe. Mcbits: fast constant-time code-based cryptography. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 250–272. Springer, 2013.
4. Daniel J Bernstein and Peter Schwabe. New aes software speed records. In *International Conference on Cryptology in India*, pages 322–336. Springer, 2008.
5. Sarani Bhattacharya and Debdeep Mukhopadhyay. Curious case of rowhammer: flipping secret exponent bits using timing analysis. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 602–624. Springer, 2016.
6. Joseph Bonneau and Ilya Mironov. Cache-collision timing attacks against aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 201–215. Springer, 2006.
7. Erik Bosman, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. Dedup est machina: Memory deduplication as an advanced exploitation vector. In *2016 IEEE symposium on security and privacy (SP)*, pages 987–1004. IEEE, 2016.
8. David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

9. Bart Coppens, Ingrid Verbauwhede, Koen De Bosschere, and Bjorn De Sutter. Practical mitigations for timing-based side-channel attacks on modern x86 processors. In *2009 30th IEEE Symposium on Security and Privacy*, pages 45–60. IEEE, 2009.
10. Stephen Crane, Andrei Homescu, Stefan Brunthaler, Per Larsen, and Michael Franz. Thwarting cache side-channel attacks through dynamic software diversity. In *NDSS*, pages 8–11, 2015.
11. Dmitry Evtyushkin, Dmitry Ponomarev, and Nael Abu-Ghazaleh. Covert channels through branch predictors: a feasibility study. In *Proceedings of the Fourth Workshop on Hardware and Architectural Support for Security and Privacy*, pages 1–8, 2015.
12. Dmitry Evtyushkin, Dmitry Ponomarev, and Nael Abu-Ghazaleh. Jump over aslr: Attacking branch predictors to bypass aslr. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE, 2016.
13. A. Fog. The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly p. `https://www.agner.org/optimize/microarchitecture.pdf.`, 2020. (Accessed: 16-05-2020).
14. Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, 8(1):1–27, 2018.
15. Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *International workshop on cryptographic hardware and embedded systems*, pages 207–228. Springer, 2015.
16. Daniel Gruss, Clémentine Maurice, Klaus Wagner, and Stefan Mangard. Flush+ flush: a fast and stealthy cache attack. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 279–299. Springer, 2016.
17. David Gullasch, Endre Bangerter, and Stephan Krenn. Cache games–bringing access-based cache attacks on aes to practice. In *2011 IEEE Symposium on Security and Privacy*, pages 490–505. IEEE, 2011.
18. Jay Heiser and Mark Nicolett. Assessing the security risks of cloud computing. *Gartner report*, 27:29–52, 2008.
19. Mehmet Sinan Inci, Berk Gulmezoglu, Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar. Cache attacks enable bulk key recovery on the cloud. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 368–388. Springer, 2016.
20. Gorka Irazoqui, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar. Wait a minute! a fast, cross-vm attack on aes. In *International Workshop on Recent Advances in Intrusion Detection*, pages 299–319. Springer, 2014.
21. JEDEC. Memory Configurations: JESD21-C, 2014. JEDEC Global Standards for the Microelectronics Industry.
22. Taesoo Kim, Marcus Peinado, and Gloria Mainar-Ruiz. {STEALTHMEM}: System-level protection against cache-based side channel attacks in the cloud. In *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, pages 189–204, 2012.
23. Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. *ACM SIGARCH Computer Architecture News*, 42(3):361–372, 2014.

24. Jingfei Kong, Onur Aciiçmez, Jean-Pierre Seifert, and Huiyang Zhou. Hardware-software integrated approaches to defend against software cache-based side channel attacks. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, pages 393–404. IEEE, 2009.
25. Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. Inferring fine-grained control flow inside {SGX} enclaves with branch shadowing. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 557–574, 2017.
26. Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE symposium on security and privacy*, pages 605–622. IEEE, 2015.
27. Peter Mell, Tim Grance, et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Justice, 2011. (Accessed: 15th May 2020).
28. Reza Montasari. An overview of cloud forensics strategy: Capabilities, challenges, and opportunities. In *Strategic Engineering for Cloud Computing and Big Data Analytics*, pages 189–205. Springer, 2017.
29. Reza Montasari and Richard Hill. Next-generation digital forensics: Challenges and future paradigms. In *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*, pages 205–212. IEEE, 2019.
30. Reza Montasari, Richard Hill, Amin Hosseinian-Far, and Farshad Montaseri. Countermeasures for timing-based side-channel attacks against shared, modern computing hardware. *International Journal of Electronic Security and Digital Forensics*, 11(3):294–320, 2019.
31. Reza Montasari, Richard Hill, Simon Parkinson, Amin Hosseinian-Far, and Alireza Daneshkhah. Hardware-based cyber threats: Attack vectors and defence techniques. *International Journal of Electronic Security and Digital Forensics*, 2019.
32. Reza Montasari, Amin Hosseinian-Far, and Richard Hill. Policies, innovative self-adaptive techniques and understanding psychology of cybersecurity to counter adversarial attacks in network and cyber environments. In *Cyber Criminology*, pages 71–93. Springer, 2018.
33. Reza Montasari, Amin Hosseinian-Far, Richard Hill, Farshad Montaseri, Mak Sharma, and Shahid Shabbir. Are timing-based side-channel attacks feasible in shared, modern computing hardware? *International Journal of Organizational and Collective Intelligence (IJOCI)*, 8(2):32–59, 2018.
34. Dan Page. Theoretical use of cache memory as a cryptanalytic side-channel. *IACR Cryptology ePrint Archive*, 2002(169), 2002.
35. Pekka Peltola, Reza Montasari, Fernando Seco, Antonio R Jimenez, and Richard Hill. Multi-platform architecture for cooperative pedestrian navigation applications. In *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE, 2019.
36. Colin Percival. Cache missing for fun and profit, 2005.
37. Randy Perry, Eric Hatcher, Robert P Mahowald, and Stephen D Hendrick. Force. com cloud platform drives huge time to market and cost savings. *TechRepublic. com/White Papers.[en línea] Disponible en: http://whitepapers. techrepublic. com. com/abstract. aspx*, 2009.
38. Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard. {DRAMA}: Exploiting {DRAM} addressing for cross-cpu attacks. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 565–581, 2016.
39. Keyun Ruan, Joe Carthy, Tahar Kechadi, and Mark Crosbie. Cloud forensics. In *IFIP International Conference on Digital Forensics*, pages 35–46. Springer, 2011.

40. Mark Seaborn and Thomas Dullien. Exploiting the dram rowhammer bug to gain kernel privileges. *Black Hat*, 15:71, 2015.
41. Makoto Takahashi, Masaya Nakatani, Soichiro Hiraoka, Hiroshi Ushio, and Akiyoshi Oshima. Component separation device, September 25 2012. US Patent 8,273,302.
42. Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. Cryptanalysis of des implemented on computers with cache. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 62–76. Springer, 2003.
43. Victor Van Der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer: Deterministic rowhammer attacks on mobile platforms. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1675–1689, 2016.
44. Yao Wang, Andrew Ferraiuolo, and G Edward Suh. Timing channel protection for a shared memory controller. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 225–236. IEEE, 2014.
45. Zhenghong Wang and Ruby B Lee. New cache designs for thwarting software cache-based side channel attacks. In *Proceedings of the 34th annual international symposium on Computer architecture*, pages 494–505, 2007.
46. Michael Weiß, Benjamin Weggenmann, Moritz August, and Georg Sigl. On cache timing attacks considering multi-core aspects in virtualized embedded systems. In *International Conference on Trusted Systems*, pages 151–167. Springer, 2014.
47. Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. One bit flips, one cloud flops: Cross-vm row hammer attacks and privilege escalation. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 19–35, 2016.
48. Yuval Yarom, Daniel Genkin, and Nadia Heninger. Cachebleed: a timing attack on openssl constant-time rsa. *Journal of Cryptographic Engineering*, 7(2):99–112, 2017.
49. Yinqian Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Cross-vm side channels and their use to extract private keys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 305–316, 2012.