MASCAB: a Micro-Architectural Side-Channel Attack Bibliography

http://www.github.com/danpage/mascab

D. Page

September 7, 2018

Contents

1 Miscellaneous

1.1 History

MASCAB:BerLanSch:12 D.J. Bernstein, T. Lange, and P. Schwabe. "The Security Impact of a New Cryptographic Library". In: *Progress in Cryptology (LATINCRYPT)*. LNCS 7533. Springer-Verlag, 2012, pp. 159–176.

> See Section 3 for a detailed description of by-design security against micro-architectural attack in the NaCl library.

MASCAB:Bernstein:08 D.J. Bernstein. "The Salsa20 family of stream ciphers". In: *New Stream Cipher Designs: The eSTREAM Finalists*. LNCS 4986. Springer-Verlag, 2008, pp. 84–97.

> Section 2.3 of this paper presents an argument for security-by-design, wrt. use of look-up tables for S-boxes and more generally, noting for example that "*[a] further argument against S-box lookups is that, on most platforms, they are vulnerable to timing attacks*".

MASCAB:Kocher:96 P.C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology (CRYPTO)*. Springer-Verlag LNCS 1109, 1996, pp. 104–113.

> This paper is evidence that attention to micro-architectural attacks can be identified quite early within the history of academic publications on side-channel attacks more generally. The point is made in various places, but in Section 11 for example the paper states that "*RAM cache hits can produce timing characteristics in implementations of Blowfish, SEAL, DES, and other ciphers if tables in memory are not used identically in every encryption*" thus clearly predicting later, concrete attacks.

MASCAB:KSWH:00 J. Kelsey et al. "Side Channel Cryptanalysis of Product Ciphers". In: *Journal of Computer Security* 8.2-3 (2000), pp. 141–158.

> This paper starts to make more confident predictions about the threat of micro-architectural attacks, stating that "*[w]e believe attacks based on cache hit ratio in large S-box ciphers like Blowfish, CAST, and Khufu are possible*".

MASCAB:Lampson:73 B.W. Lampson. "A note on the confinement problem". In: *Communications of the ACM (CACM)* 16.10 (1973), pp. 613–615.

> This is a seminal paper, exploring various foundational topics: in the abstract, it 1) defines the confinement problem, namely the problem of preventing a process communicating with some other process during execution, 2) defines some terminology, namely a covert channel as "those not intended for information transfer at all, such as the service program's effect on the system load", and a storage channel as "[s]torage of various kinds maintained by the supervisor which can be written by the service and read by an unconfined program, either shortly after it is written or at some later time", 3) defines various necessary conditions for confinement, concluding "[t]here is not likely to be any rigorous way of identifying every channel in any system of even moderate complexity" st. sufficient conditions are hard to provide in a generic manner.

MASCAB:SchWhi:98 B. Schneier and D. Whiting. "Twofish on Smart Cards". In: *Smart Card Research and Applications (CARDIS)*. LNCS 1820. Springer-Verlag, 1998, pp. 265–276.

> This paper is evidence that attention to micro-architectural attacks can be identified quite early within the history of academic publications on side-channel attacks more generally. Two quotes in Section 4.6 stand out, for example: 1) "*[t]iming attacks based on cache hits are not applicable to smart cards as they typically do not have a cache*" and 2) "*[o]n some CPUs the multiply instruction uses an "early out" algorithm*". Clearly the former is reasonable based on the date of publication, even if invalidated wrt. modern examples; the latter acts as a prediction of sorts, later realised in concrete attacks using exactly this behaviour.

MASCAB:SibPorLin:95 O. Sibert, P.A. Porras, and R. Lindell. "The Intel 80x86 Processor Architecture: Pitfalls for Secure Systems". In: *IEEE Symposium on Security & Privicy (S&P)*. 1995, pp. 211–222.

This paper presents an analysis of features and defects in x86, spanning both architecture (or ISA) and micro-architecture (or implementation), focusing on their security implications; there are quite a number of concrete storage- and timing-based covert and side-channels outlined. In addition, some content 1) Section 3.8: (ab)use of high resolution timers, "[a]s with any high-resolution clock, the TSC must be either virtualized or eliminated entirely to reduce the covert channel threat" 2) Section 3.9: (ab)use of performance counters, "[u]nlike the TSC, which measures an external clock, some performance MSRs can measuree effects of other concurrent activities (e.g., cache snoops by another processor) or of previous activities (e.g., cache hits and misses due to cache activity by a previous subject)" 3) Section 3.10: caches, "caches present potential for covert timing channels" 4) Section 4.2 point 6: prefetching, "[p]refetching may fetch otherwise inaccessible instructions in virtual 8086 mode". The latter point as been pitched as an early warning of modern attacks **MASCAB:LSGPHMKGYH:18**; **MASCAB:KHFGGHHLMPSY:18**, but, because prefetching and speculative execution (of instructions) differ considerably, this argument seems reasonable only so far.

1.2 Datasheet

MASCAB:intel˙sdm˙v1:17 Intel. *Intel 64 and IA-32 architectures software developer's manual volume 1: Basic architecture*. Tech. rep. 253665. 2017. URL: `http://software.intel.com/en-us/articles/intel-sdm`

MASCAB:intel˙sdm˙v2:17 Intel. *Intel 64 and IA-32 architectures software developer's manual combined volumes 2A, 2B, 2C, and 2D: Instruction set reference, A-Z*. Tech. rep. 325383. 2017. URL: `http://software.intel.com/en-us/articles/intel-sdm`

MASCAB:intel˙sdm˙v3:17 Intel. *Intel 64 and IA-32 architectures software developer's manual combined volumes 3A, 3B, 3C, and 3D: System programming guide*. Tech. rep. 325384. 2017. URL: `http://software.intel.com/en-us/articles/intel-sdm`

MASCAB:intel˙sdm˙v4:17 Intel. *Intel 64 and IA-32 architectures software developer's manual volume 4: Model-specific registers*. Tech. rep. 335592. 2017. URL: `http://software.intel.com/en-us/articles/intel-sdm`

1.3 Patent

MASCAB:Baker:06 B.S. Baker. "Avoiding cache line sharing in virtual machines". U.S. Patent Number 11/490,785. 2006

MASCAB:BGGKM:07 E.F. Brickell et al. "Protecting a Branch Instruction from Side Channel Vulnerabilities". U.S. Patent Number 11/951,999. 2007

MASCAB:DolAha:08 B. Dolgunov and A. Aharonov. "Memory Randomization For Protection Against Side Channel Attacks". U.S. Patent Number 12/254,225. 2008

MASCAB:Fine:13 K.S. Fine. "Detection of side channel attacks between virtual machines". U.S. Patent Number 14/384,677. 2013

MASCAB:Kavi:16 K.M. Kavi. "Method and apparatus for improving computer cache performance and for protecting memory systems against some side channel attacks". U.S. Patent Number 13/458,145. 2016

MASCAB:LeeWan:10 R.B. Lee and Z. Wang. "Cache Memory Having Enhanced Performance and Security Features". U.S. Patent Number 12/633,500. 2010

MASCAB:MPBC:08 F.X. McKeen et al. "Method and apparatus for preventing software side channel attacks". U.S. Patent Number 11/513,871. 2008

MASCAB:MulReb:13 D. Mukhopadhyay and C.D. Rebeiro. "Resistance to cache timing attacks on block cipher encryption". U.S. Patent Number 14/350,044. 2013

MASCAB:NevSei:05 M. Neve and J.-P. Seifert. "Resisting cache timing based attacks". U.S. Patent Number 11/302,579. 2005

MASCAB:RaiGueShe:13 S. Raikin, S. Gueron, and G. Sheaffer. "Protecting private data from cache attacks". U.S. Patent Number 11/950,963. 2013

MASCAB:Rempel:17 C. Rempel. "Protection against access violation during the execution of an operating sequence in a portable data carrier". U.S. Patent Number 13/581,955. 2017

MASCAB:SebGue:08 J. Sebot and S. Gueron. "Mitigating Branch Prediction and Other Timing Based Side Channel Attacks". U.S. Patent Number 11/950,658. 2008

MASCAB:SetMarDem:18 L. Sethumadhavan, R. Martin, and J. Demme. "Systems and methods to counter side channel attacks". U.S. Patent Number 9,887,833 B2. 2018

MASCAB:StrJenDhe:07 M. Stribaek, J. Jensen, and J.-F. Dhem. "Random Cache Line Refill". U.S. Patent Number 11/943,751. 2007

MASCAB:Trostle:04 J.T. Trostle. "Timing attacks against user logon and network I/O". U.S. Patent Number 10/937,079. 2004

MASCAB:Williamson:12 B.D. Williamson. "Line allocation in multi-level hierarchical data stores". U.S. Patent Number 12/458,690. 2012

1.4 Standard

MASCAB:DOD:85 "Trusted Computer System Evaluation Criteria". In: 5200.28-STD. 1985. URL: `http://fas.org/irp/nsa/rainbow/std001.htm`.

> Section 8.0 of the so-called ORANGE BOOK includes some specific attention to covert channels: it demands a Trusted Computing Base (TCB), which is essentially a security kernel responsible for enforcing policy and supporting isolation, should allow "*the capability to audit the use of covert channel mechanisms with bandwidths that may exceed a rate of one (*1*) bit in ten (*10*) seconds*". This definition of what it terms a "high" bandwidth covert channel is motivated by comparison with the bandwidth of a normal terminal: it says "*[i]t does not seem appropriate to call a computer system "secure" if information can be compromised at a rate equal to the normal output rate of some commonly used device*".

MASCAB:NCSC:93 "A Guide To Understanding Covert Channel Analysis Of Trusted Systems". In: NCSC-TG-030. 1993. URL: `http://fas.org/irp/nsa/rainbow/tg030.htm`

1.5 Survey

MASCAB:AciSeiKoc:07 O. Acıiçmez, J.-P. Seifert, and Ç.K. Koç. "Micro-Architectural Cryptanalysis". In: *IEEE Security & Privacy* 5.4 (2007), pp. 62–64

MASCAB:GYCH:18 Q. Ge et al. "A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware". In: *Journal of Cryptographic Engineering (JCEN)* 8 (1 2018). See also `http://eprint.iacr.org/2016/613`, pp. 1–27

MASCAB:LyuMis:18 Y. Lyu and P. Mishra. "A Survey of Side-Channel Attacks on Caches and Countermeasures". In: *Journal of Hardware and Systems Security* 2.1 (2018), pp. 33–50

MASCAB:SaiSar:16 R. Vijay Sai and S. Saravanan. "A Review on Security in Cache Memories". In: *Indian Journal of Science and Technology* 9.48 (2016), pp. 1–6

MASCAB:ShaLit:15 A. Shahzad and A. Litchfield. "Virtualization Technology: Cross-VM Cache Side Channel Attacks make it Vulnerable". In: *Australasian Conference on Information Systems (ACIS)*. 2015

MASCAB:Szefer:16 J. Szefer. *Survey of Microarchitectural Side and Covert Channels, Attacks, and Defences*. Cryptology ePrint Archive, Report 2016/479. 2016. URL: `http://eprint.iacr.org/2016/479`

MASCAB:Zhang:17 Y. Zhang. "Cache Side Channels: State of the Art and Research Opportunities". In: *ACM Conference on Computer and Communications Security (CCS)*. 2017, pp. 2617–2619

1.6 Position

MASCAB:BurMutTiw:16 W. Burleson, O. Mutlu, and M. Tiwari. "Who is the major threat to tomorrow's security? You, the hardware designer". In: *Design Automation Conference (DAC)*. 2016

MASCAB:DunBea:18 C. Dunham and J. Beard. "This Architecture Tastes Like Microarchitecture". In: 2018.

This position paper (retrospectively) surveys the concept of an ISA representing an abstraction of the underlying implementation, i.e., that of decoupling the ISA from the microarchitecture. The authors argue in favour of this approach, vs. the alternative of exposing microarchitectural features in the ISA: focusing in particular on the original RISC design, they cite design decisions such as a) use of branch delay slots or hints (vs. aggressive prediction and speculation), and b) SIMD (e.g., Intel SSE) with fixed length vectors (vs. true vector designs, e.g., Cray, with arbitrary length vectors), as mistakes in hindsight. While not totally incompatible with **MASCAB:Heiser:17** (e.g., this paper is focused mainly on access to, vs. transparency about, functionality and resources in the microarchitecture), there is a fairly clear contrast in philosophy. Specifically, 1) here, the argument is that treating the ISA as an abstraction affords flexibility; this can be leveraged to satisfy market demands wrt. metrics such as performance and compatibility, 2) there, the argument is that said abstraction and flexibility is a problem: *it* represents at least a complicating factor in producing secure software implementations, because important aspects of the execution semantics for a given instruction cannot be known or, therefore, relied upon.

MASCAB:GeYarHei:18 Q. Ge, Y. Yarom, and G. Heiser. "No security without time protection: we need a new hardware-software contract". In: *Asia-Pacific Workshop on Systems (APSys)*. 2018.

This paper could be viewed as a follow-up to, or at least related to the ideas in **MASCAB:Heiser:17**. Sections 1 and 2 outline the central problem, and so requirements, at a high level. The goal is to prevent (unintended, or unauthorised) information flow between security domains. As such, the platform and kernel must, in combination, enforce time protection or isolation (cf. memory protection); to do so, the hardware must offer mechanisms that allow the kernel to either 1) strictly partition and/or 2) cleanly reset pertinent (micro-)architectural resources. The choice of mechanism is quite context dependent, but, for example, and again at a high level, the former can cater for off-core resources (e.g., LLC, which is used concurrently by multiple cores), whereas the latter can cater for on-core resources (e.g., BTB, whose state might otherwise be retained and probed across security domains). Some subtle caveats apply to both, including a) differences between virually and physically indexed state, and b) need for a constant-time (or padded, worst-case) reset operation. Sections 3 and 4 survey existing reset mechanism in x86 and ARM, and provide evidence of their efficacy; per **MASCAB:GeYarHei:17**, which offers a related analysis, neither is ideal. Finally, Section 5 introduces a set of five design properties for ISAs that allow them to satisfy requirements 1) and 2) above; these are termed an augumented ISA (aISA), and capture a (more) security, or at least information leakage, aware contract between hardware and software per**MASCAB:Heiser:17**.

MASCAB:Heiser:17 G. Heiser. "For Safety's Sake: We Need a New Hardware-Software Contract!" In: *IEEE Design & Test* 35.2 (2018), pp. 27–30.

This position paper explicitly proposes an idea implicit suggested in a variety of other work. The idea is basically greater transparency from or stronger guarantees by the ISA: exposing micro-architectural detail (e.g., temporal properties of execution) normally abstracted and hidden by the ISA, the premise is that more robust verification (e.g., wrt. secure execution) can result. Although the paper offers few concrete details, there are some high-level suggestions for what a so-called Augmented ISA (AISA) *could* include. One could point at the announcement by ARM of a new processor mode in Armv8.4-A as an example of this idea, guaranteeing that most (notably *not* memory access) instructions have data-independent execution time; this forms part of the ARM IoT Security Manifesto[a] (e.g., see Page 10 re. "the rise of side-channels "). However, although it seems unclear[b] *why*, the original[c] announcement was later amended to remove this feature[d].

[a] `http://pages.arm.com/iot-security-manifesto.html`
[b] `http://twitter.com/Kensan42/status/946681596001902593`
[c] `http://web.archive.org/web/20171108010153/http://community.arm.com/processors/b/blog/posts/introducing-2017s-extensions-to-the-arm-architecture`
[d] `http://community.arm.com/processors/b/blog/posts/introducing-2017s-extensions-to-the-arm-architecture`

MASCAB:RKLMR:04 S. Ravi et al. "Security as a new dimension in embedded system design". In: *Design Automation Conference (DAC)*. 2004, pp. 753–760

MASCAB:RRKH:04 S. Ravi et al. "Security in embedded systems: Design challenges". In: *ACM Transactions on Embedded Computer Systems* 3.3 (2004), pp. 461–491

MASCAB:SimChiAnd:18 L. Simon, D. Chisnall, and R. Anderson. "What you get is what you C: Controlling side effects in mainstream C compilers". In: *IEEE European Symposium on Security & Privicy*

(Euro-S&P). 2018, pp. 1–15.

> Although not a position paper per se, there is a clear point made in content. The argument, also made elsewhere, is that writing secure implementations requires cooperation with the development toolchain; this is true of both software *and* hardware, and all elements of said toolchain, e.g., compiler *and* assembler. The point is illustrated using two examples, namely challenges wrt. 1) developing constant-time software implementations, and 2) secure erasure of stack content (or memory content generally). The former is clearly pertinent to side-channel attack, in that any countermeasure implemented by the developer must then be transformed reliably by the toolchain; any overly agressive optimisation by the compiler, for example, can impact robustness of said countermeasure. As a step toward solution of this issue, the paper first introduces implicit invarients: these are properties the developer *wants* their implementation to retain after translation. The paper argues that implicit invarients need to be communicated by the developer, and presents various explicit controls to do so. For example, a dedicated `__builtin_ct_choose` LLVM built-in is used to signal a need for constant-time selection between two inputs (i.e., as an alternative to a ternary operator).

MASCAB:Tiri:07 K. Tiri. "Side-channel Attack Pitfalls". In: *Design Automation Conference (DAC)*. 2007, pp. 15–20.

> This paper sort of surveys the threat of side-channels generally, e.g., to capture and disseminate challenges to the hardware design community. However, it has a focus on micro-architectural attacks in Section 2.

1.7 Book

MASCAB:AciKoc:09 O. Acıiçmez and Ç.K. Koç. "Microarchitectural Attacks and Countermeasures". In: *Cryptographic Engineering*. Ed. by Ç.K. Koç. Springer, 2009. Chap. 18, pp. 475–504

MASCAB:RebMukBha:15 C. Rebeiro, D. Mukhopadhyay, and S. Bhattacharya. *Timing Channels in Cryptography: A Micro-Architectural Perspective*. Springer, 2015

1.8 Thesis

MASCAB:Aciicmez:07:a O. Acıiçmez. "Advances in side-channel cryptanalysis: Microarchitectural attacks". PhD thesis. Oregon State University, 2007

MASCAB:Bao:17 C. Bao. "Hardware Attacks And Mitigation Techniques". PhD thesis. University of Maryland, 2017.

> In particular, see Chapter 4 (relating to use of 3D integration as a countermeasure against cache-based attacks).

MASCAB:Brumley:11 B.B. Brumley. "Covert Timing Channels, Caching, and Cryptography". PhD thesis. Aalto University, 2011

MASCAB:DAntoine:15 S.M. D'Antoine. "Exploiting Processor Side Channels to Enable Cross VM Malicious Code Execution". MA thesis. Rensselaer Polytechnic Institute, 2015

MASCAB:Doychev:16 G. Doychev. "Tools for the Evaluation and Choice of Countermeasures against Side-Channel Attacks". PhD thesis. Universidad Politécnica De Madrid, 2016

MASCAB:Falzon:16 K. Falzon. "On the Use of Migration to Stop Illicit Channels". PhD thesis. Technischen Universität Darmstadt, 2016

MASCAB:Frigo:17 P. Frigo. "Practical Microarchitectural Attacks from Integrated GPUs". MA thesis. Delft University of Technology, 2017

MASCAB:Gallais:11 J.-F. Gallais. "Microarchitectural Side-Channel Attacks". PhD thesis. University of Luxembourg, 2013.

> In particular, see Chapter 2 (relating to cache-based attacks), and Chapter 4 (relating to the peripheral topic of hardware Trojans).

MASCAB:Gay:08 R. Gay. "Interrupt-related Covert Channels from an Attackers Perspective". MA thesis. Aachen University, 2008

MASCAB:Grabher:10 P. Grabher. "Processor Design Techniques for Efficient and Secure Execution of Cryptographic Algorithms". PhD thesis. University of Bristol, 2010.

> In particular, see Chapter 5 (relating to cache-based attacks on low-power designs), and Chapter 6 (relating to cache-based attacks on instruction caches).

MASCAB:Green:17 M. Green. "Implicit Cache Lockdown on ARM: An Accidental Countermeasure to Cache-Timing Attacks". MA thesis. Worcester Polytechnic Institute, 2017

MASCAB:Gruss:17 D. Gruss. "Software-based Microarchitectural Attacks". PhD thesis. Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology, 2017

MASCAB:Huang:10 B. Huang. "Cache-collision timing attacks against AES-GCM". MA thesis. University of Delaware, 2010

MASCAB:Irazoqui:17 G. Irazoqui. "Cross-core Microarchitectural Side Channel Attacks and Countermeasures". PhD thesis. Worcester Polytechnic Institute, 2017

MASCAB:Josyula:15 S.P. Josyula. "On the Applicability of a Cache Side-Channel Attack on ECDSA Signatures: The FLUSH+RELOAD attack on the point multiplication in ECDSA signature generation process". MA thesis. Blekinge Institute of Technology, 2015

MASCAB:Krak:17 R.S. Krak. "Cycle-Accurate Timing Channel Analysis of Binary Code". MA thesis. University of Twente, 2017

MASCAB:Lipp:16 M. Lipp. "Cache Attacks and Rowhammer on ARM". MA thesis. Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology, 2016

MASCAB:Liu:16 F. Liu. "Cache Side Channel Attacks and Secure Cache Architectures". PhD thesis. Princeton University, 2016

MASCAB:Maurice:15 C. Maurice. "Information leakage on shared hardware: Evolutions in recent hardware and applications to virtualization". PhD thesis. Institut des Sciences et des Technologies de Paris (ParisTech), 2015

MASCAB:Neve:06 M. Neve. "Cache-based Vulnerabilities and SPAM analysis". PhD thesis. Université catholique de Louvain, 2006

MASCAB:Schroder:17 F. Schröder. "Security of Cache Replacement Policies under Side-Channel Attacks". MA thesis. Technischen Universität Berlin, 2018

MASCAB:Svedenborg:14 S.V. Svedenborg. "Exploring Instruction Cache Analysis On ARM". MA thesis. Norwegian University of Science and Technology, 2014

1.9 Software

MASCAB:CacheGrab "CacheGrab". URL: `http://github.com/nccgroup/cachegrab`

MASCAB:CTTK "CTTK: Constant-Time Toolkit)". URL: `http://github.com/pornin/CTTK`

MASCAB:Mastik Y. Yarom. "Mastik: a micro-architectural side-channel toolkit". URL: `http://cs.adelaide.edu.au/~yval/Mastik/`

MASCAB:TRICL C. Percival. "TRICL: a timing attack resistant cryptographic library". URL: `http://www.daemonology.net/tricl`

2 Covert channels

2.1 Negative use-cases

MASCAB:NagAbu:17 H. Naghibijouybari and N. Abu-Ghazaleh. "Covert Channels on GPGPUs". In: *IEEE Computer Architecture Letters* 16.1 (2017), pp. 22–25.

This paper constructs a contention-based covert channel, using either L1 or L2 caches on a specific, Nvidia-based GPU; this permits communication within or between constituent streaming multi-processors.

3 Side-channels

3.1 Caches (data *and* instruction)

3.1.1 Negative use-cases: attacks using data-cache (access-based, methodology)

MASCAB:ABFPY:16 T. Allan et al. "Amplifying Side Channels Through Performance Degradation". In: *Annual Computer Security Applications Conference (ACSAC)*. 2016, pp. 422–435.

There is a central idea in this paper, then used in a second step so as to mount an (improved) attack on OpenSSL-based ECDSA. Essentially, the idea is to execute a process dedicated to aggressively evicting content associated with a target; this increases the latency of memory accesses and thus execution time, so degrading performance. The *reason* to do is is that increased execution time thereby allows a higher-fidelity measurement within, for example, PRIME+PROBE style, access-driven cache attacks wrt. the LLC.

MASCAB:BanGulKre:10 E. Bangerter, D. Gullasch, and S. Krenn. "Cache Games – Bringing Access Based Cache Attacks on AES to Practice". Cryptology ePrint Archive, Report 2010/594. 2010. URL: `%5Curl%7Bhttp://eprint.iacr.org/2010/594%7D`.

This is an older version of **MASCAB:GulBanKre:11**; although the newer version is most relevant, this older version remains important in specific cases due to (minor) differences in content.

MASCAB:DKPT:17 C. Disselkoen et al. "PRIME+ABORT: A Timer-Free High-Precision L3 Cache Attack using Intel TSX". In: *Computer and Communications Security (CCS)*. 2017, pp. 51–67.

Within the context of access-driven cache attacks, this paper presents an attack technique which harnesses the Intel transactional memory, or TSX feature. The technique moniker PRIME+ABORT hints at the underlying idea: it is basically similar to PRIME+PROBE, but, rather than interrogate residency of some target address by timing the execution of a "probe" access, it makes use of TSX. More specifically, the attacker primes TSX wrt. the target address st. access to it by the target process causes a transaction abort; this essentially triggers the attack process, which can then act accordingly. Section 3 overviews several strategies in more detail, but some high-level advantages and disadvantages are common to most: a) there is no need for a timer, also implying the side-channel involves less noise, b) there is no need for synchronisation with or detection of the target process: the attack process can essentially just wait, until it is triggered *precisely* when the target process performs an access, c) it is harder to cope with (i.e., multiplex) multiple target addresses. Note that Figure 1 offers a neat comparison of different attack styles, wrt. a variety of properties, and that **MASCAB:JanLeeKim:16** use the same sort of technique to target KASLR.

MASCAB:GruMauWag:15 D. Gruss, C. Maurice, and K. Wagner. "FLUSH+FLUSH: A Stealthier Last-Level Cache Attack". In: *CoRR* abs/1511.04594 (2015). URL: `http://arxiv.org/abs/1511.04594`.

Although the title suggests a focus on the latter, this paper makes three main contributions: it 1) analyses the use of performance counters as a means of (dynamically, at run-time) detecting cache-based side-channel attack; this suggests measuring last-level cache misses and references is enough, 2) formulates a metric for said detection that normalises wrt. the size of kernel (since an attack process exhibits behaviour in the cache via a tight attack loop), and 3) following roughly the same model as FLUSH+RELOAD, an alternative attack strategy, namely FLUSH+FLUSH, is presented: the strategy hinges on the fact that `clflush` has address-dependent execution time (longer if an inclusive flush is required, shorter if not: this is sort of the opposite to a cache hit and cache miss, since it needs longer to flush if the data is resident), so it is possible to replace the reload memory access with a second use of `clflush` and hence a) has a faster attack loop since `clflush` has lower latency than memory access (although this also implies lower fidelity, since the difference in latency is also less), b) is less invasive, in the sense flushes do not trigger prefetching (vs. a reload, which, since it is a memory access, *does*), and c) is more stealthy, since flushes do not count as cache misses. The attack is evaluated by applying it in a range of contexts, e.g., to perform input distinguishing (Section VI) wrt. GTK, and recovery of cryptographic keys (Section VII) wrt. AES as implemented by OpenSSL. There are some further insights into `clflush` itself in Section $VIII$, and a detailed comparison of countermeasure techniques within Section IX.

MASCAB:GruSprMan:15 D. Gruss, R. Spreitzer, and S. Mangard. "Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches". In: *USENIX Security Symposium*. 2015, pp. 897–912.

> In a sense, this paper generalises and adds some degree of automation to FLUSH+RELOAD attack style, access-driven cache attacks (as hinted by the title, wrt. inclusive LLCs). A central idea stems from DPA-style template attacks: the attack is in two phases, namely 1) a profiling (or characterisation) phase (Section 3.1) where, for each event of interest, FLUSH+RELOAD is applied to each address in the (controlled) target; this yields a template wrt. the event, which is essentially a profile of related cache activity, then 2) an exploitation phase (Section 3.2) where the real target is executed: FLUSH+RELOAD is applied to each address, and the resulting profile compared (using a suitable similarity metric) to the corpus of templates st. the event can be identified. The concept of an event is described in the abstract: it could be a) high-level, e.g., a key-press within some application, or b) low-level, e.g., a table access, for example. The paper harnesses the concept to mount various concrete attacks, e.g., on both Linux and Windows UI (Sections 5.1 and 5.3), and AES as implemented by OpenSSL (Section 5.4).

MASCAB:GulBanKre:11 D. Gullasch, E. Bangerter, and S. Krenn. "Cache Games – Bringing Access-Based Cache Attacks on AES to Practice". In: *IEEE Symposium on Security & Privacy (S&P)*. 2011, pp. 490–505.

> This paper extends the state-of-the-art in access-driven cache attacks in several ways. In particular, it a) introduces a new attack strategy, which is FLUSH+RELOAD-esque in the sense it uses a combination of `clflush` and (timed) access to memory, but targets the L1 cache, b) makes use of a form of performance degradation, forcing the target to execute for an abnormally short time quantum: by (ab)using the Linux CFS, the attack process forces this behaviour so as to isolate as few as a single memory access by the target and so obtain very "clean" samples. The paper provides some additional contributions, among which are use of a neural network in order to combat the effect of noise. In combination, the performance degradation and neural network approaches mean the attack can succeed in a totally asynchronous setting: this improves over EVICT+TIME *and* PRIME+PROBE per **MASCAB:TroOsvSha:10**.

MASCAB:OsvShaTro:06 D.A. Osvik, A. Shamir, and E. Tromer. "Cache Attacks and Countermeasures: The Case of AES". In: *Topics in Cryptology (CT-RSA)*. LNCS 3860. Springer-Verlag, 2005, pp. 1–20.

> This is an older version of **MASCAB:TroOsvSha:10**; although the newer version is most relevant, this older version remains important in specific cases due to (minor) differences in content.

MASCAB:Percival:05:a C. Percival. "Cache Missing For Fun And Profit". 2005. URL: `http://www.daemonology.net/papers/htt.pdf`.

> This is an older version of **MASCAB:Percival:05:b**; although the newer version is most relevant, this older version remains important in specific cases due to (minor) differences in content.

MASCAB:Percival:05:b C. Percival. "Cache Missing For Fun And Profit". 2005. URL: `http://www.daemonology.net/papers/cachemissing.pdf`.

> This is a seminal paper, which 1) introduces the PRIME+PROBE attack strategy, targeting either L1 or L2 cache of a Pentium 4 core using an attack process that executes in quasi-parallel (via x86-based HypherThreading) with the target process, and 2) applies this to the RSA implementation in OpenSSL; since this uses the CRT and sliding window exponentiation, some effort is required to perform key recovery (i.e., the attack recovers ~ 300 of 512 bits of p and q, so an approach is needed to recover the rest via a post-processing phase). This alone is important (contemporary attacks were against symmetric targets such as block ciphers), but the paper also includes various additional details of interest: a central example is the detailed attack process in Figure 1 that a) copes with influence of the prefetch system, b) copes with latency of `rdtsc` by artificially elongating the critical path using extra, dependent instructions.

MASCAB:TroOsvSha:10 E. Tromer, D.A. Osvik, and A. Shamir. "Efficient Cache Attacks on AES, and Countermeasures". In: *Journal of Cryptology* 23.1 (2010), pp. 37–71.

> This paper offers a number of seminal contributions. For example, it 1) formalises terminology wrt. synchronous vs. asynchronous attacks, 2) introduces both the EVICT+TIME (Section 3.4) *and* PRIME+PROBE (Section 3.5) attack strategies, applying variants of them to AES implementations within OpenSSL and the Linux `dm-crypt` disk encryption system; both implementations depend on quasi-parallel execution (in this case via x86-based HypherThreading), and target activity in the L1 (and to a lesser extent L2, or at least not shared between core and so not LLC) cache, 3) offers a detailed survey of potential countermeasures (Section 5), and 4) makes several wider predictions and comments, such as the fact "bank collisions (e.g., in Athlon 64 processors) likewise cause timing to be affected by low address bits".

MASCAB:YarFal:14 Y. Yarom and K. Falkner. "FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-channel Attack". In: *USENIX Security Symposium*. 2014, pp. 719–732.

> This paper introduces the FLUSH+RELOAD attack strategy (or, as **MASCAB:GulBanKre:11** describe but do not name what is basically the same approach, *re*introduces): crucially, this approach a) targets the LLC (modulo issues such as inclusivity), so b) permits inter-core attacks st. an attack process executing on one core can attack a target process executing on another; this allows relaxation of previous restrictions on core co-residency, and thus enables cross-VM attacks (modulo issues such as shared memory, via de-duplication). The attack strategy could be viewed as a variant of PRIME+PROBE, in the sense it assumes access to a shared memory region but uses the x86-based *clflush* instruction to prime (i.e., *invalidate*) specific cache lines; Figure 4 supports a detailed overview of the attack process kernel, noting the need for memory fences to serialise memory accesses (and the caveat wrt. using `cpuid` instead, which may in fact be virtualised). Section 4 applies the strategy to the GnuPG implementation of RSA, probing addresses in modular squaring and multiplication functions to distinguish the two and hence recover bits of a 2048-bit exponent based on which they are invoked.

3.1.2 Negative use-cases: attacks using data-cache (access-based, vs. cryptography)

MASCAB:BHLY:16 L.G. Bruinderink et al. "Flush, Gauss, and Reload - A Cache Attack on the BLISS Lattice-Based Signature Scheme". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 9813. Springer-Verlag, 2016, pp. 323–345

MASCAB:GarBru:17 C.P. García and B.B. Brumley. "Constant-Time Callees with Variable-Time Callers". In: *USENIX Security Symposium*. 2017, pp. 83–98.

> This paper mounts a FLUSH+RELOAD style, access-driven cache attack on OpenSSL-based ECDSA. Although this has been targeted previously, the paper proceeds in a somewhat different manner: a) noticing a supposed countermeasure (a constant-time implementation of binary EEA) was not activated due to a bug wrt. propogation of flags (embedded as meta-data in the multi-precision integer object), b) attacking the binary EEA (vs. scalar multipliplication) in ECDSA, and thus recovering inputs (the ephemeral key, or nonce k) via a shift and subtraction sequence in turn recovered from the cache attack (cf. **MASCAB:AciGueSei:07** and **MASCAB:AraThu:07** who use side-channels based on branch prediction and power analysis with the same goal), c) enhancing the robustness of this approach by employing performance degredation **MASCAB:ABFPY:16**, d) recovering the ECDSA signing key from a set of ephemeral keys using a final, lattice-based cryptanalytic step.

MASCAB:GarBruYar:16 C.P. García, B.B. Brumley, and Y. Yarom. "Make Sure DSA Signing Exponentiations Really are Constant-Time". In: *Computer and Communications Security (CCS)*. 2016, pp. 1639–1650.

> An observation, namely identification of a bug, enables contributions of this paper. OpenSSL included a new flag to each multi-precision integer representation, indicating whether or not it is deemed security-critical (e.g., is an RSA private exponent); the flag is used to select either a variable- or constant-time exponentiation implementation. This property *should* be propagated through arithmetic operations, but, it turns out, is *not* due to a bug (or at least poor API design). This fact is harnessed in an attack against OpenSSL DSA implementation, combining use of a) performance degradation, b) a FLUSH+RELOAD style, access-driven cache attack wrt. the LLC, plus c) a lattice-based key recovery phase; the attack is applied both to TLS and SSH.

MASCAB:LGSM:16 M. Lipp et al. "ARMageddon: Last-Level Cache Attacks on Mobile Devices". In: *USENIX Security Symposium*. 2016, pp. 549–564.

> The application of access-driven, cache-based side-channel attacks to an ARM processor implies various challenges: for example, a) ARM use a random replacement policy, which complicates the challenge of interrogating residency of some target data (i.e., by introducing noise), b) the ARM-based analogy of the x86 Time-Stamp Counter (TSC) can only be used in user mode *if* permission is granted from kernel mode, c) a dedicated flush instruction can only be executed in kernel mode, and d) ARM use an LLC that is not inclusive (i.e., data resident in the L1 cache is not guaranteed to be resident in the LLC), which differs from various alternatives. This paper addresses these challenges, mounting PRIME+PROBE and EVICT+RELOAD style, access-driven cache attacks on the Android touch-screen interface (in order to recover tap and swipe events), and BouncyCastle AES implementation.

MASCAB:YarGenHen:16 Y. Yarom, D. Genkin, and N. Heninger. "CacheBleed: A Timing Attack on OpenSSL Constant Time RSA". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS

9813. Springer-Verlag, 2016, pp. 346–367.

To address **MASCAB:Percival:05b** Intel contributed an implementation of the scatter-gather countermeasure to OpenSSL; it basically patched the sliding window based modular exponentiation function as used by RSA. The original attack was able to distinguish between pre-computed values (thus inferring bits of the exponent used to select them) because they mapped to different cache lines; scatter-gather interleaves said values (in a sense transposing them in memory), st. the *same* cache lines are accessed irrespective of the value. This is based on intuition which suggests the latency of access leaks no information below cache line granularity. This paper effectively defeats the scatter-gather countermeasure. Using the fact (per **MASCAB:Bernstein:05** and **MASCAB:TroOsvSha:10**) bank conflicts alter the latency of access to the L1 cache, the intuition above is invalidated: an attack process intentionally constructed to probe a specific bank can monitor access to that bank by the target (by virtue of the resulting difference in latency of access), and so infer associated access *within* some cache line. This approach is harnessed to mount a concrete attack, which requires some post-processing to cope with the (relatively) low number of exponent bits recovered and thereby perform key recovery.

MASCAB:ZanXiaZha:16 X. Zhang, Y. Xiao, and Y. Zhang. "Return-Oriented Flush-Reload Side Channels on ARM and Their Implications for Android Devices". In: *ACM Conference on Computer and Communications Security (CCS)*. 2016, pp. 858–870.

The focus of this paper is the application of FLUSH+RELOAD-style, access-driven cache attacks on ARM-based devices executing Android; the L2 cache (i.e., LLC) is used as a target. A range of challenges are address, but there are effectively three main contributions: 1) Although user-mode processes cannot directly control the caches, the OS provides the `clearcache` system call to *clean* the data cache (i.e., write-back) and *flush* the instruction cache; in a sense, this permits `clflush`-like behaviour for the instruction cache only. A characterisation of this system call, and user-mode timers (since ARM also lacks a user-mode `tdtsc`)) is presented. 2) FLUSH+RELOAD attacks are them mounted using `clearcache`, targeting the instruction cache. The most interesting aspect here is the use of Return Oriented Programming (ROP) gadgets as a means of reloading: calling a "full" target function is problematic in the sense a) a sane caller context (e.g., parameters) must be constructed, b) the achievable granularity of the attack may be to low to capture the function if the associated execution time is short. Instead, the idea is to use ROP gadgets as a proxy for the function itself: these need no context and, by definition, will have a short execution time (of a few cycles typically: automatically returning after a few instructions). 3) The attack is leveraged to target some real-world Android use-cases, including detection of hardware and UI events.

3.1.3 Negative use-cases: attacks using data-cache (access-based, vs. UI)

MASCAB:Hornby:16 T. Hornby. "Side-Channel Attacks on Everyday Applications: Distinguishing Inputs with FLUSH+RELOAD". In: *BlackHat USA*. 2016. URL: `http://www.defuse.ca/side-channel-attacks-on-everyday-applications.htm`.

This paper shifts the target of side-channel attack from key recovery to privacy; the main contribution is a PRIME+PROBE style, access-driven cache attacks used as an input distinguisher (e.g., to determine web-site access by Links, or which PDF loaded by libpoppler, or whether a TrueCrypt volume contains a hidden volume). This alone is interesting, but Section 3.1 offers an additional contribution, (briefly) outlining an approach to automate selection of probe addresses.

3.1.4 Negative use-cases: attacks using data-cache (access-based, vs. TEE)

MASCAB:BMDKCS:17 F. Brasser et al. "Software Grand Exposure: SGX Cache Attacks Are Practical". In: *CoRR* abs/1702.07521 (2017). URL: `http://arxiv.org/abs/1702.07521`.

This paper describes an attack on a target process (so, more accurately, a target enclave) which executes with support from SGX. In line with the SGX security model, the attacker is able to control the surrounding system; per Figure 2, for example, it thus mounts a PRIME+PROBE style (using either L1 or L2 caches, but not LLC), access-driven cache attack on the target. On one hand, doing so is intuitively feasible: each enclave is isolated wrt. various potential interactions, but also share micro-architectural resources with the surrounding system. As a result, Intel exclude such attacks from the SGX security model, so in a sense view them as out of scope (e.g., see **MASCAB:CosDev:16**, Section 6.6). However, on the other hand, some complicating factors exist; the paper addresses these to provide practical evidence of this intuition. Much of the complication stems from noise. For example, in addition to the inherent, experimental noise wrt. timing measurement, pollution of the cache state will degrade the validity of said measurements: 1) there is inherent intra-process pollution, 2) other processes may use the same core as the target process and thus cause inter-process pollution, and 3) any interrupt will cause an Asynchronous Enclave Exit (AEX), activity related to which may pollute the cache. Section 4.3 describes a series of mitgations for these issues. Some are fairly obvious (e.g., limiting those processes scheduled to execute on the target core), but some fairly novel: use of performance counters to offer noise-free measurement of cache behaviour avoids disadvantages in **MASCAB:UhsGeoVer:08**, for example, due to the SGX security model. There has been some critique[a] of the attack target, i.e., the RSA implementation provided in the Intel SDK for SGX. Specifically, the target is assumed to make use of an insecure, fixed-window exponentiation rather than a hardened variant (cf. Section 1, Page 2) also provided; the criticism is that the former is intended for public-key operations so key recover is moot. Arguably there are some counter-arguments: 1) if the API allows such mistakes, the API is badly designed (or at least could or *should* be better), 2) although valid, such an argument does not (necessarily) detract from the high-level take away point and techniques involved. In respect to the latter, in Section 6 the paper also includes an attack against the non-cryptographic workload of genomic processing.

[a] See, e.g., `http://jbeekman.nl/blog/2017/03/sgx-side-channel-attacks`.

MASCAB:Brumley:15 B.B. Brumley. "Cache Storage Attacks". In: *Topics in Cryptology (CT-RSA)*. LNCS 9048. Springer-Verlag, 2015, pp. 22–34.

This paper focuses first on a covert channel predicated on the NS bit in ARM-based caches: based on the storage channel paradigm, the idea is for a channel to be formed between a (insecure) sender and (secure) receiver by 1) the receiver priming the cache with (insecure) content, 2) the sends evicts some of that content, say wrt. an address x, thus replacing with with (secure) content of their own, 3) the receiver probes the cache: when they access address x this is disallowed, causes an interrupt and thus completes transmission. This general approach is then translated into the side-channel setting, where an access-driven attack using a similar principle is presented; an advantage of this is the strong signal is exhibits (vs. use of timing to interrogate residency), but a disadvantage seems to be the need to use physical addresses since the probe stage need to specifically access x (vs. some congruent x') to violate the access permissions.

MASCAB:GESM:17 J. Götzfried et al. "Cache Attacks on Intel SGX". In: *European Workshop on System Security (EuroSec)*. 2017, 2:1–2:6.

As noted in Section 1.2, this paper was developed concurrently with **MASCAB:SWGMM:17** and **MASCAB:BMDKCS:17**; whatever the timing, it *seems* to represent the first peer-reviewed publication relating to cache-based side-channel attacks on SGX. The context is similar to **MASCAB:BMDKCS:17**, in that the goal is a key-recovery attack on a workload executed in an SGX enclave; the target is a T-tables style AES implementation realised in OpenSSL already known to be vulnerable. A PRIME+PROBE style, access-driven cache attack is used (with an elimination-based **MASCAB:NevSei:06** strategy for key-recovery) wrt. the L1 cache. Versus **MASCAB:BMDKCS:17**, there are also similarities in terms of the approach, e.g., wrt. performance counters for noise-free measurement, and attack-friendly scheduling and core allocation by a malicious kernel. Section 4.2 offers some interesting insights: 1) although the Intel ASCI **MASCAB:intel_sdm_v3:17**, Section 42.6 feature can prevent leakage from the enclave via performance counters, their use by a malicious kernel for probing avoids this restriction, and 2) the Intel SDK for SGX utilises an implementation of AES that does not harness AES-NI, which would be the obvious countermeasure against such attacks, which seems an odd choice.

MASCAB:MogIraEis:17 A. Moghimi, G. Irazoqui, and T. Eisenbarth. "CacheZoom: How SGX Amplifies The Power of Cache Attacks". In: *CoRR* arXiv:1703.06986 (2017). URL: `http://arxiv.org/abs/1703.`

06986.

This paper present an attack strategy termed CacheZoom; it operates within the context of SGX, allowing attack of a target enclave, applying an access-driven cache attack based on Prime+Probe to the L1 cache. By mounting a key-recovery attack on an AES implementation that executes within the target enclave, a) the remit matches that of **MASCAB:GESM:17**; **MASCAB:BMDKCS:17**, and b) elements of the strategy are similar: the attacker-controlled kernel manipulates the scheduler to determine which core the target enclave and attack process execute on, for example. However, the strategy *also* differs significantly in the sense that it intentionally interrupts the target enclave; by doing so on (upto) a per instruction basis, noise that may impact on inspection of the cache state (i.e., the probe step). This is achieved using the APIC deadline mode, and, per Section 6.1.1, can eliminate the effect of pre-warming countermeasures because the per instruction fidelity allows it to be skipped or even "undone" before execution of AES itself.

MASCAB:SWGMM:17 M. Schwarz et al. "Malware Guard Extension: Using SGX to Conceal Cache Attacks". In: *CoRR* abs/1702.08719 (2017). URL: http://arxiv.org/abs/1702.08719.

This paper describes a malware concept, where an attack process executes within an SGX enclave. The idea is to mount a Prime+Probe style, access-driven cache attack on some target process (e.g., as executed in another enclave); per Figure 2, for example, the attack hinges on the fact that accesses from the attack enclave will interact, wrt. the LLC, with those from the target. On one hand, doing so is intuitively feasible: each enclave is isolated wrt. various potential interactions, but also share micro-architectural resources with the surrounding system. As a result, Intel exclude such attacks from the SGX security model, so in a sense view them as out of scope (e.g., see **MASCAB:CosDev:16**, Section 6.6). However, on the other hand, some complicating factors exist; the paper addresses these to provide practical evidence of this intuition. For example: a) Within an enclave, the x86 Time-Stamp Counter (TSC) is not available; to provide a high-resolution timer a counter-based thread (see also **MASCAB:LGSM:16**) is used, which, via some optimisation, offers *sub*-cycle accuracy. b) The address space of an enclave (formed from a sub-set of the virtual address space of the associated process) is backed by protected (e.g., encrypted) physical page frames; since the attacker cannot translate virtual addresses to their physical equivalent, eviction of specific cache sets (as required for Prime+Probe) is more complicated. The paper develops a novel approach for deriving the cache set for a given virtual address; this is based on use of timing differences that occur due to bank conflicts during access to physical memory. Beyond this, some more subtle points seem important to stress: 1) it seems fair to *not* classify this as an attack on SGX per se, but rather on the premise that SGX cannot be subverted for malign purposes, 2) an enclave is, by definition, a black box: isolating an enclave from other components is a positive in benign use-cases, but then becomes a negative in malign alternatives, e.g., since the malware cannot be detected by conventional means, and 3) there has been some critique[a] of the attack target, in the sense that although mbedTLS harnesses a constant-time modular multiplication the exponentiation attacked is based on square-and-multiply (despite mbedTLS including side-channel resistant alternatives); although a reasonable criticism, it does not seem to detract from the concept *or* impact of the points above.

[a] See, e.g., http://jbeekman.nl/blog/2017/03/sgx-side-channel-attacks.

MASCAB:ZSSLH:16 N. Zhang et al. "TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices". Cryptology ePrint Archive, Report 2016/980. 2016. URL: http://eprint.iacr.org/2016/980.

Set within the context of ARM TrustZone, this paper essentially shows the NS bit does *not* prevent a non-secure attacker mounting cache-based attacks on a secure target. Put another way, the NS bit prevents access, but *not* eviction: the non-secure attacker can interrogate residency of content relating to the secure target, by using the approach of timing (controlled) eviction as per Prime+Probe. The paper harnesses this fact to mount concrete attacks on OpenSSL-based AES, via the L1 cache (so avoiding issues such as inclusivity), using an interrupt-based (vs. the x86 time stamp counter, for example) method for timing accesses.

3.1.5 Negative use-cases: attacks using data-cache, (time-based)

MASCAB:BHNS:10 B.B. Brumley et al. "Consecutive S-box Lookups: A Timing Attack on SNOW 3G". In: *Information and Communications Security (ICICS)*. LNCS 6476. Springer-Verlag, 2010, pp. 171–185

MASCAB:JiaFeiKae:16 Z.H. Jiang, Y. Fei, and D. Kaeli. "A complete key recovery timing attack on a GPU". In: *High Performance Computer Architecture (HPCA)*. 2016, pp. 394–405.

> This paper presents a cache-based timing attack on AES: per Figure 1, the context is similar to analogous remote timing attacks, except that the target executes AES using a GPU (vs. a CPU). As such, the central observation is that on the experimental Nvidia-based GPU, for example, execution time is correlated with L1 cache behaviour.

MASCAB:LeaZenHaw:09 G. Leander, E. Zenner, and P. Hawkes. "Cache Timing Analysis of LFSR-based Stream Ciphers". In: *Cryptography & Coding*. Springer-Verlag LNCS 5921, 2009, pp. 433–445

MASCAB:TSSSM:03 Y. Tsunoo et al. "Cryptanalysis of DES Implemented on Computers with Cache". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 2779. Springer-Verlag, 2003, pp. 62–76.

> This is a seminal paper, in that it represents one of the first concrete cache-based timing attack on a cryptographic primitive, namely optimised DES (i.e., with wide look-up tables that include the permutations). The paper uses a 1-round toy example (in Figure 1) to explain the attack strategy (Section 2), which includes 1) an acquisition phase, using either the a) non-elimination (or constructive) approach that uses evidence of many cache hits to infer equality (upto cache line granularity) of two S-box indices, a) elimination (or constructive) approach that uses evidence of many cache misses to infer inequality (upto cache line granularity) of two S-box indices, then 2) a cryptanalytic post-processing phase, that uses the key difference produced by the acquisition phase to guide a constrained brute-force search and thus perform key recovery. This is then extended to full DES (Section 3); the experiment reported assumes a cold L1 cache (which is flushed before every DES invocation), and uses the x86 `rdtsc` to measure execution time inline with said invocation (vs. with a separate attack and target process).

MASCAB:TTMM:02 Y. Tsunoo et al. "Cryptanalysis of Block Ciphers Implemented on Computers with Cache". In: *International Symposium on Information Theory and Its Applications (ISITA)*. 2002, pp. 803–806.

> Although the title does not suggest it, this paper presents an attack on the MISTY block cipher.

MASCAB:Zenner:08 E. Zenner. "A Cache Timing Analysis of HC-256". In: *Selected Areas in Cryptography (SAC)*. LNCS 5381. Springer-Verlag, 2008, pp. 199–213.

> Perhaps more interesting than the attack on HC-256 in this paper, is the fact that 1) it applies to a stream cipher, whereas other attacks focus either on block ciphers (e.g., AES) or RSA almost exclusively; in a sense, the paper extends the remit of micro-architectural attacks, 2) Section 5.2 includes a discussion of security-by-design, i.e., the idea of (re)designing a stream cipher to offer explicit by-design vs. after-the-fact, "patched" countermeasures, and, perhaps most importantly, 3) Section 6 generalises the attack model, allowing analysis of any stream cipher; various other papers use **MASCAB:LeaZenHaw:09** or extended **MASCAB:BHNS:10** it.

MASCAB:ZhaWanZhe:09 X.-J. Zhao, T. Wang, and Y.-Y. Zheng. "Cache Timing Attacks on Camellia Block Cipher". Cryptology ePrint Archive, Report 2009/354. 2009. URL: `http://eprint.iacr.org/2009/354`

3.1.6 Negative use-cases: attacks using data-cache, (trace-based)

MASCAB:AciKoc:06 O. Acıiçmez and Ç.K. Koç. "Trace-Driven Cache Attacks on AES". In: *Information and Communications Security (ICS)*. LNCS 4307. Springer-Verlag, 2006, pp. 112–121

MASCAB:NRMW:13 P.H. Nguyen et al. "Improved Differential Cache-Trace Attacks on SMS4". In: *Information Security and Cryptology (INSCRYPT)*. Springer-Verlag, 2013, pp. 29–45

3.1.7 Negative use-cases: attacks using data-cache, (power-based)

MASCAB:BZBMP:05 G. Bertoni et al. "AES power attack based on induced cache miss and countermeasure". In: *Information Technology: Coding and Computing (ITCC)*. 2005, pp. 586–591.

> This paper mounts a (sort of) trace-driven cache attack on AES: the power consumption of a target device is used to distinguish between cache hits and misses, yielding said trace. That said, the strategy, namely 1) execute AES to prime the cache with S-box content, 2) read i-th element of an array whose base address mirrors the S-box, thus evicting the i-th S-box entry, 3) (re)execute AES, and infer use of S-box element i iff. a cache miss is observed, has a distinctly access-driven flavour: the attack model assumes a user mode attacker, and kernel mode cryptographic service as the target. The attack is validated using simulation (via the MIPS-based SimpleScalar framework).

MASCAB:FouTun:06 J.J.A. Fournier and M. Tunstall. "Cache Based Power Analysis Attacks on AES". In: *Australasian Conference on Information Security and Privacy (ACISP)*. LNCS 4058. Springer-Verlag, 2006, pp. 17–28.

> This paper presents two (theoretical): there is no experimental data for either) attacks on AES, based on the assumption that the attacker could monitor power consumption and hence cache behaviour of a target. The attacks target 1) look-ups to the pre-computed S-box within `SubBytes`, and 2) look-ups to the pre-computed `xtime` within `MixColumns`. Both could be characterised as trace-driven, and stem from the attacker searching specific plaintext bytes: when a cache hit is observed, they can form a relationship between known and unknown variables (i.e., the plaintext and key material), and thus allow subsequent cryptanalysis.

MASCAB:MorHin:15 A. Moradi and G. Hinterwälder. "Side-Channel Security Analysis of Ultra-Low-Power FRAM-Based MCUs". In: *Constructive Side-Channel Analysis and Secure Design (COSADE)*. LNCS 9064. Springer-Verlag, pp. 239–254.

> Versus **MASCAB:BZBMP:05**, for instance, where the power consumption of a cache is *simulated*, this paper presents some concrete results for AES executed on a low-power micro-controller: Figure 5 provides a clear illustration of how cache hits and misses can be distinguished in a trace of power consumption, a fact then leveraged in a trace-driven attack.

3.1.8 Negative use-cases: attacks using instruction-cache

MASCAB:AciBruGra:10 O. Acıiçmez, B. Brumley, and P. Grabher. "New Results on Instruction Cache Attacks". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 6225. Springer-Verlag, 2010, pp. 110–124

MASCAB:Aciicmez:07:b O. Acıiçmez. "Yet Another MicroArchitectural Attack: Exploiting I-cache". In: *ACM Workshop on Computer Security Architecture*. 2007, pp. 11–18

MASCAB:AciSch:08 O. Acıiçmez and W. Schindler. "A Vulnerability in RSA Implementations Due to Instruction Cache Analysis and Its Demonstration on OpenSSL". In: *Topics in Cryptology (CT-RSA)*. LNCS 4964. Springer-Verlag, 2008, pp. 256–273

3.1.9 Negative use-cases: attacks using cache pre-fetch

MASCAB:AciBruGra:10 O. Acıiçmez, B. Brumley, and P. Grabher. "New Results on Instruction Cache Attacks". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 6225. Springer-Verlag, 2010, pp. 110–124

MASCAB:Aciicmez:07:b O. Acıiçmez. "Yet Another MicroArchitectural Attack: Exploiting I-cache". In: *ACM Workshop on Computer Security Architecture*. 2007, pp. 11–18

MASCAB:AciSch:08 O. Acıiçmez and W. Schindler. "A Vulnerability in RSA Implementations Due to Instruction Cache Analysis and Its Demonstration on OpenSSL". In: *Topics in Cryptology (CT-RSA)*. LNCS 4964. Springer-Verlag, 2008, pp. 256–273

3.1.10 Positive use-cases: hardware-based countermeasures

MASCAB:CanKopRei:17 P. Cañones, B. Köpf, and J. Reineke. "Security Analysis of Cache Replacement Policies". In: *Principles of Security and Trust (POST)*. LNCS 10204. Springer-Verlag, 2017, pp. 189–209

MASCAB:FucLee:15 A. Fuchs and R.B. Lee. "Disruptive Prefetching: Impact on Side-channel Attacks and Cache Designs". In: *ACM International Systems and Storage Conference (SYSTOR)*. 2015, 14:1–14:12.

> The central idea in this paper is to harness hardware pre-fetchers as a countermeasure against cache-based side-channel attacks: various papers note the negative influence pre-fetching can have on attack success (by polluting the cache state vs. that expected, for example), so this idea attempts to intentionally maximise said influence in a sense. Alterations to two different pre-fetching algorithms are investigated, namely 1) the approach in Section 3.2 randomises the length and order of accesses in the pre-fetched address stream, attempting to balance the degree of randomisation with resulting performance, 2) the approach in Section 3.3 alters the selection of cache set, so as to achieve a uniform set access pattern. Section 4 demonstrates that a combination of 1) and 2) act to prevent a (simulated) PRIME+PROBE-based attack.

MASCAB:KASK:08 G. Keramidas et al. "Non deterministic caches: A simple and effective defense against side channel attacks". In: 12.3 (2008), pp. 221–230.

> This paper repurposes an existing idea, namely cache decay (or, simply, timed cache line invalidation), as a countermeasure against cache-based side-channel attack; originally designed as a mechanism to reduce power consumption, a per line counter disables (and so invalidates) said line once a period of inactivity (the decay interval) passes. The proposal is to reset the counters with a random value, which induces a level of non-deterministic behaviour due to randomisation of the decay interval and hence triggered invalidation. This noise wrt. cache behaviour acts to degrade the validity of inferences based on a given acquisition (e.g., that longer execution time implies more cache misses *solely* due to algorithmic behaviour).

MASCAB:KKEEAPJ:17 M. Kayaalp et al. "RIC: Relaxed Inclusion Caches for Mitigating LLC Side-Channel Attacks". In: *Design Automation Conference (DAC)*. 2017, 7:1–7:6.

> This paper presents RIC, a design concept to secure LLCs vs. cache-based side-channel attack. The idea is to (selectively) relax the inclusivity property of (e.g., x86-based) LLC designs; this is achieved by adding a per line flag to enable RIC, which is controlled to allow a) read-only (e.g., S-box tables, or instructions) or b) thread-local (e.g., any pre-computation for windowed exponentiation) to avoid eviction due to inclusivity. As such, RIC aims to prevent any attack strategy (e.g., FLUSH+RELOAD) that depends on inclusivity; it does so without disadvantages, such as degraded performance, that may be levelled at alternatives such as partitioning.

MASCAB:YGST:17 M. Yan et al. "Secure Hierarchy-Aware cache Replacement Policy (SHARP): Defending against cache-based side channel attacks". In: *International Symposium on Computer Architecture (ISCA)*. 2017, pp. 347–360

3.1.11 Positive use-cases: software-based countermeasures

MASCAB:BanRiz:08 J. Bani and S.S. Rizvi. "Minimizing Cache Timing Attack Using Dynamic Cache Flushing (DCF) Algorithm". In: *Wireless Telecommunications Symposium (WTS)*. 2008, pp. 399–404.

> As outlined in Figure 10 the main proposal in the paper is the idea of randomly flushing the cache, which a) introduces temporal jitter to some extent, but also b) disrupts any attempt to correlate execution time or cache state with key material (i.e., an attack).

MASCAB:ErlAba:07 Ú. Erlingsson and M. Abadi. *Operating system protection against side-channel attacks that exploit memory latency*. Tech. rep. MSR-TR-2007-117. Microsoft, 2007. URL: `http://www.microsoft.com/en-us/research/publication/operating-system-protection-against-side-channel-attacks-that-exploit-memory-latency`.

> This paper considers countermeasures against cache-based attack, under the general assumption that processes will explicitly request (via an API akin to `malloc`) and then use a protected "stealth" region of memory. A set of schemes to manage such regions, thereby (supposedly) preventing an attack: these are 1) partition the region, st. *only* addresses from the stealth region can be resident in a given cache line and/or set, 2) enforce a cache flush when context switching a process that has used a stealth region (st. there may be security-critical content resident in the cache), 3) pre-load (or "warm") the cache with content from the stealth region when context switching.

MASCAB:GGOFWD:09 V. Gopal et al. "Fast and constant-time implementation of modular exponentiation". In: *Reliable Distributed Systems*. 2009. URL: `http://www.cse.buffalo.edu/srds2009/escs2009_submission_Gopal.pdf`.

> See Section *IV* re. scatter-gather technique for layout of pre-computed values in modular exponentiation (cf. RSA).

MASCAB:GSOHLC:17 D. Gruss et al. "Strong and Efficient Cache Side-Channel Protection using Hardware Transactional Memory". In: *Computer and Communications Security (CCS)*. 2017, pp. 217–233.

Within the context of access-driven cache attacks, this paper presents a countermeasure technique which harnesses the Intel transactional memory, or TSX feature; the technique is dubbed CLOAK. The basic idea is to execute the target functionality within a TSX-based transaction: by first warming the cache with the data required, doing so means any deliberate manipulation (e.g., eviction) by an attack process will abort the transaction and hence foil the attack. Conceptually, the approach is similar to AESSE **MASCAB:MulDewFre:10** (and/or similar) which mitigate cold-boot attacks by maintaining a working set of data in the register file; here, the working set is instead maintained in a (or the) cache, and eviction is detected (vs. avoided) via the transaction abort. Some care is required (e.g., wrt. correct warming) to apply CLOAK, but doing so is reality non-invasive: the transaction can be applied to "wrap" target functionality, at a higher level than the functionality itself for example. Section 6 highlights this fact by retrofitting and evaluating CLOAK using previously vulnerable functionality.

MASCAB:Gueron:12 S. Gueron. "Efficient software implementations of modular exponentiation". In: *Journal of Cryptographic Engineering (JCEN)* 2.1 (2012), pp. 31–43.

See Section 7 re. scatter-gather technique for layout of pre-computed values in modular exponentiation (cf. RSA).

MASCAB:Hamburg:09 M. Hamburg. "Accelerating AES with Vector Permute Instructions". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 5747. Springer-Verlag, 2009, pp. 18–32.

This paper explores vector-based (namely SSE and Altivec) implementation techniques for AES. The results demonstrate such implementations can be efficient, while also preventing cache-based side-channel attacks: they essentially represent a class of constant-time countermeasure.

MASCAB:LGYMRHL:16 F. Liu et al. "CATalyst: Defeating last-level cache side channel attacks in cloud computing". In: *High Performance Computer Architecture (HPCA)*. 2016, pp. 406–418.

This paper presents CATALYST, a software system that harnesses the so-called Cache Allocation Technology (CAT) as a countermeasure against LLC-based attack. Intel market CAT as a performance-oriented technology that protects selected LLC content from eviction: some number of Classes of Service (COS) are identified and used to tagged content, which cannot then be evicted by a different COS. In practice, this means any content in the LLC wrt. some performance-critical web-server, for example, could be protected against eviction by other, non-critical processes. CATalyst essentially re-purposes the CAT hardware mechanism, using it as a security-oriented technology. Focusing on cross-VM attacks using the LLC, it offers access to a set of secure pages that are "pinned" (or locked) in a managed cache partition (as supported by CAT). Such pages are is protected wrt. contention, meaning a security-critical COS can be protected against access-driven cache attacks (e.g., FLUSH+RELOAD) that interrogate the residency of content by timing (controlled) eviction.

MASCAB:XTPCL:17 M. Xu et al. "vCAT: Dynamic Cache Management Using CAT Virtualization". In: *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 2017, pp. 211–222.

Although Section II mentions security (i.e., cache-based side-channels) as a use-case, this paper focuses on delivery of general mechanisms for partitioned LLCs based on Intel CAT. The idea is to virtualise, and so allow multiplexing of, the fixed number of CAT partitions; these can then be managed by, e.g., the kernel, wrt. some policy. Clearly that policy might be security-conscious, pitching vCAT as a potentially then generalising the approach of CATalyst **MASCAB:LGYMRHL:16**.

MASCAB:ZhaRei:13 Y. Zhang and M. K. Reiter. "Düppel: retrofitting commodity operating systems to mitigate cache side channels in the cloud". In: *Computer and Communications Security (CCS)*. 2013, pp. 827–838.

This paper presents DÜPPEL, which automatically flushes content security-critical content from the cache; the mechanism focuses on L1 and L2 caches, vs. LLC for example, and realised within the kernel. The idea of cache flushing as a countermeasure against attack is not new: doing so during a context switch means content relating to some target process is always evicted before an attacker executes, and therefore no information can be leaked by the former to the latter via the cache. However, the overhead of flushing is significant; this suggests value in a more "intelligent" policy for doing so, which DÜPPEL offers. More specifically, it a) controls the frequency of flushes st. they can be avoided if there is no chance of leakage anyway, b) allows processes to be marked as security-critical (using an entry in the file system to communicate between user- and kernel-space), which then acts as a hint as to whether a flush is required or not.

MASCAB:ZouReiZan:16 Z. Zhou, M.K. Reiter, and Y. Zhang. "A software approach to defeating side channels in last-level caches". In: *Computer and Communications Security (CCS)*. 2016, pp. 871–882.

> The paper presents CacheBar, a software-only countermeasure against LLC-based attacks: it basically implements security-aware page management. To prevent Flush+Reload attacks it implements a copy-on-*access* policy to (just-in-time) disallow sharing of pages (which enable such an attack strategy); there is an option to reshare (or deduplicate) pages if access to them is temporally non-local (i.e., non-indicative of attack). To prevent Prime+Probe attacks it implements a mechanism to limit the number of lines per set an attacker can access: this places a limit on the ability to observe activity in the set.

3.1.12 Positive use-cases: analysis-based countermeasures (e.g., of source code)

MASCAB:DFKMR:13 G. Doychev et al. "CacheAudit: A Tool for the Static Analysis of Cache Side Channels". In: *USENIX Security Symposium*. 2013, pp. 431–446.

> This paper presents a (theoretical) model that allows analysis of the cache-related behaviour of target and attack processes, and then also implements and evaluates a (concrete) tool based on this; the tool is able to accept x86 binaries as input, and provide guarantees wrt. the leakage possible and hence viability of an attack. The tool itself, termed CacheAudit, employs static analysis based on the use of abstract interpretation (of the binary). Evaluation with AES (per PolarSSL) and Salsa20 allows several general conclusions (e.g., that pre-loading can reduce leakage, or that larger line size can reduce leakage); perhaps the strongest claim is that (the implementation of) Salsa20 provably avoids all attacks (wrt. the model at least). Interestingly, Section 3.4 seems to model an access-driven attacker that is able to interrogate the cache state *on termination* of the target process: it is only able to observe the *final* cache state. This model does not seem to match state-of-the-art strategies for attack, such as Flush+Reload, which are more continuous in nature (i.e., could recover each c_i, not just c_n).

MASCAB:ManWebKoe:17 H. Mantel, A. Weber, and B. Köepf. "A Systematic Study of Cache Side Channels across AES Implementations". In: *Engineering Secure Software and Systems (ESSoS)*. LNCS 10379. Springer-Verlag, 2017, pp. 213–230.

> Having extended the CacheAudit tool of **MASCAB:DFKMR:13** to cope with a larger set of x86 instructions, the main goal of the paper is a systematic evaluation of leakage from deployed AES implementations (wrt. cache-based attacks); it considers LibTomCrypt, mbed TLS, Nettle, and OpenSSL. Various conclusions are presented, with perhaps the most interesting being the interaction with look-up table geometry (e.g., their number, element type, and memory footprint) and security. As an aside, a number of nicely organised micro-surveys are included, such as a) Section 3.1 on static analysis techniques for leakage assessment, and b) Section 6 on related work, especially lineage of countermeasures.

3.1.13 Alternative use-cases

MASCAB:GES:17 B. Gulmezoglu, T. Eisenbarth, and B. Sunar. "Cache-Based Application Detection in the Cloud Using Machine Learning". In: *ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. 2017, pp. 288–300.

> This paper could be viewed as a progression from **MASCAB:IIES:15** in that the goal is generalised: rather than specifically detecting use of cryptographic libraries, it aims to detect use of given applications in general. The idea is similar to **MASCAB:GruSprMan:15** in the sense that profiles of behaviour, wrt. the cache, are captured for each application; rather than comparison with a template, however, machine learning, specifically Support Vector Machines (SVMs), is used as a classifier.

MASCAB:IIES:15:b G. Irazoqui et al. "Know Thy Neighbor: Crypto Library Detection in the Cloud". In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* (1 2015), pp. 25–40.

> This paper presents a novel use of cache-based side-channel: rather than mount an attack per se, the idea is to *identify* an attack target. The goal is to determine whether or not another, co-located target VM is using given cryptographic library (e.g., OpenSSL vs. GnuTLS) of a given version (e.g., 0.9.8k vs. 1.0.0a). This, coupled with another approach that supports recovery of the IP address used by the target VM, equips the attacker with information they can then use to subsequently launch a tailored attack (e.g., Heartbleed, based on identifying a vulnerable version of OpenSSL). The technique to achieve this is actually fairly simple: the idea is to capitalise on memory deduplication, so using a Prime+Probe attack strategy (wrt. LLC) to detect the difference in access time for a) deduplicated (i.e., library is used) and b) non-deduplicated (i.e., library is not used) pages.

3.2 Branch prediction

3.2.1 Negative use-cases: attacks

MASCAB:LSGKKP:16 S. Lee et al. "Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing". In: *CoRR* abs/1611.06952 (2016). URL: `http://arxiv.org/abs/1611.06952`.

This paper presents a mechanism for fine-grained analysis of branch behaviour, of a target enclave by some attacker. The mechanism is similar to **MASCAB:EvtPonAbu:16** in so far as it relies on use of so-called shadow branches overlaid with the target branches st. they contend in various branch prediction structures (e.g., the BTB, which is basically a cache for branch target addresses). Associated variation in execution time can be leveraged to recover the branch behaviour (e.g., branch un/taken status, or branch target address). Rather than a TSC-based approach to destinguish rollback (or not, so longer or shorter execution time), the Last Branch Record (LBR) of Intel processors is used: this is akin to a performance counter for branches, logging information wrt. recent, taken branches for later analysis. Overall, the paper 1) details the construction of how shadow branches, 2) exploits this within various concrete attacks (in Section 4), including a conditional subtraction included in the mbed TLS implementation of Montgomery multiplication; 3) considers various countermeasures against the mechanism, which include a flush-based approach (predicated on hardware support, cf. **MASCAB:GeYarHei:17**), and a software-based approach termed ZIGZAGGER; the latter realises all branches via a form of dispatch, imposes a performance overhead at run-time, but could be automated (by the compiler) at compile-time.

3.3 Speculative and/or out-of-order execution

3.3.1 Negative use-cases: attacks

MASCAB:KHFGGHHLMPSY:18 P. Kocher et al. "Spectre Attacks: Exploiting Speculative Execution". In: (2019). See also `http://spectreattack.com/spectre.pdf`, pp. 19–37.

See also **MASCAB:LSGPHFHMKGYH:18**. Aligning with the co-discovery by Horn[a] of Google Project Zero, the technique captures variant 1 (CVE-2017-5753 or "Bounds Check Bypass") and variant 2 (CVE-2017-5715 or "Branch Target Injection"). There are a wide range of summaries, but that of Bilar[b] is impressively clear and concise. Various high-level explanations also exist; the overview[c] from Raspberry Pi founder Eben Upton is typically clear, though also quite high-level. The attack technique allows an (unprivileged) user process to load from any (mapped) address in the virtual address space of another process. Note that one *could* classify this as an attack stemming from the behaviour of branch predictors; however, it seems more reasonable to cite speculative execution as the root cause and branch predictors simply as one mechanism involved. The first variant (per Section 4) targets conditional branches, e.g., those used to guard array access via a bounds check. The attack itself can be summarised as follows: 1) a target branch is identified within the address space of the target process, 2) the target process is used (e.g., provided with input) st. the branch predictor is trained to believe the branch will be taken, 3) the target process is used (e.g., provided with input) st. the branch is not taken: due to 2), speculative execution means transient instructions from the "branch taken path" are executed, 4) a speculatively executed, transient instruction accesses an address based on a target value, thereby altering the cache state, 5) the cache state is inspected using a FLUSH+RELOAD-based strategy, thus recovering the target value. The second variant (per Section 5) targets indirect branches, e.g., those capable of branching to an address computed and stored within a register (such as x86 `jmp`). The attack itself can be summarised as follows: 1) a target branch is identified within the address space of the target process, 2) a "gadget" is identified within the address space of the target process, 3) the attacker then "poisons" the branch target buffer, i.e., trains it to believe the target branch transfers control-flow to the gadget address, 4) the target process is used (e.g., provided with input) st. the target branch is executed: due to 3), speculative execution means transient instructions from the "gadget" are executed, 4) a speculatively executed, transient instruction accesses an address based on a target value, thereby altering the cache state, 5) the cache state is inspected using a FLUSH+RELOAD-based strategy, thus recovering the target value. There are a few things to note about this strategy, e.g., a) the use of an existing instruction sequence (or gadget) hints at an analogy with Return-Oriented Programming (ROP), b) the training in 3) can be performed using a branch within the address space of the attack process: this is because only a subset of bits are used to index the branch target buffer.

[a] `http://googleprojectzero.blogspot.co.uk/2018/01/reading-privileged-memory-with-side.html`
[b] `http://twitter.com/daniel_bilar/status/948884520316653569`
[c] `http://www.raspberrypi.org/blog/why-raspberry-pi-isnt-vulnerable-to-spectre-or-meltdown/`

MASCAB:LSGPHFHMKGYH:18 M. Lipp et al. "Meltdown: Reading Kernel Memory from User Space". In: (2018). See also `http://meltdownattack.com/meltdown.pdf`, pp. 973–990.

See also **MASCAB:KHFGGHHLMPSY:18**. Aligning with the co-discovery by Jann Horn[a] of Google Project Zero, the technique captures variant 3 (CVE-2017-5754 or "Rogue Data Cache Load"). There are a wide range of summaries, but that of Bilar[b] is impressively clear and concise: Various high-level explanations also exist; the overview[c] from Raspberry Pi founder Eben Upton is typically clear, though also quite high-level. The attack technique allows an (unprivileged) user process to load from any (mapped) address in the associated virtual address space. This is problematic because a) the kernel address space is typically mapped into the virtual address space for each user process (e.g., to reduce the overhead of context switches), so the attack violates user-kernel address space isolation, plus b) the entirety of physical memory may be mapped into the kernel address space (e.g., to allow access to pages allocated to user processes), so the attack violates user-user address space isolation. Note that one *could* classify this as an attack stemming from the behaviour of caches; however, it seems more reasonable to cite speculative execution as the root cause and caches simply as one mechanism involved. The attack itself can be summarised (Section 5.1) as follows: 1) a target value, at an attacker-selected address, is loaded into into a register, 2) a speculatively executed instruction accesses an address based on the target value, thus altering the cache state, 3) the cache state is inspected using a FLUSH+RELOAD-based strategy, thus recovering the target value. This essentially works because the exception caused by 1) is not enforced until 2) has been speculative executed, hence altering the cache state (which is then not reverted); clearly the steps can be iterated. Listing 2 presents the following concise example:

```
retry:
mov al, byte [ rcx ]
shl rax, 0xC
jz retry
mov rbx, qword [ rbx + rax ]
```

although it includes a reset mechanism (to cope with a property of exception handling mechanism), lines 2 and 5 essentially capture steps 1) and 2) from above. Line 2 loads a byte from the address in `rcx` (selected by the attacker), which should be inaccessible and so will cause an exception, into `rax`. Line 5 loads a byte from an address in a probe array (pointed to by `rbx`) at an index dictated by `rax`; this alters the cache state, which can be inspected by FLUSH+RELOAD. Various challenges are address to optimise the attack, e.g., a) use of the retry mechanism mentioned above, b) scattering accesses to the probe array across pages to avoid any influence from the prefetcher, and c) the need to deal with continued execution after the exception (via exception suppression or handling).

[a] `http://googleprojectzero.blogspot.co.uk/2018/01/reading-privileged-memory-with-side.html`
[b] `http://twitter.com/daniel_bilar/status/948884520316653569`
[c] `http://www.raspberrypi.org/blog/why-raspberry-pi-isnt-vulnerable-to-spectre-or-meltdown/`

MASCAB:MaiRos:18 G. Maisuradze and C. Rossow. "Speculose: Analyzing the Security Implications of Speculative Execution in CPUs". In: *CoRR* abs/1801.04084 (2018). URL: `http://arxiv.org/abs/1801.04084`

MASCAB:StePre:18 J. Stecklina and T. Prescher. "LazyFP: Leaking FPU Register State using Microarchitectural Side-Channels". In: *CoRR* abs/1806.07480 (2018). URL: `http://arxiv.org/abs/1806.07480`.

This paper hinges on the use of lazy FPU context switching, which is a performance-enhancing technique for x86 stemming from use of the (separate) x87 FPU. Motivated (originally) by limited use of the FPU, and so needless overhead, the concept basically defers a full context switch of the FPU registers. The FPU is disabled vs. context switched, st. later use causes an interrupt; at this point the context switch is realised, to preserve/restore the FPU state "just in time". Use of this mechanism (which is optional, and arguably of limited value in most modern contexts) implies the FPU state for one user process could be accessed by another *if* the access control mechanism can be bypassed. This is problematic, since extensions such as AES-NI use (logical) SSE-based registers that overlay the (physical) FPU registers: access to the FPU registers could leak AES-NI key material. The LazyFP attack basically does just this, using the Meltdown **MASCAB:LSGPHFHMKGYH:18** strategy to bypass the access control via speculatively executed instructions (which access the FPU registers before the access control can trigger an interrupt). The attack target could be related to **MASCAB:TakNogMor:11**, except of course the attacker uses a normal, user process vs. a malicious hypervisor.

3.4 Arithmetic

3.4.1 Negative use-cases: attacks

MASCAB:AciSei:07 O. Acıiçmez and J.-P. Seifert. "Cheap Hardware Parallelism Implies Cheap Security". In: *Fault Diagnosis and Tolerance in Cryptography (FDTC)*. 2007, pp. 80–91.

> To manage the issue of scale, SMT-based processor cores multiplex some (smaller) number of functional (or execution) units between a (larger) number of hardware-supported threads executed in quasi-parallel; when a complementary instruction mix results (e.g., from composition of one compute-bound and one memory-bound thread), there is little contention. This paper notices the execution units are a shared micro-architectural, albeit one that does not retain state (cf. a cache), and, based on this fact, mounts an attack on RSA as implemented in OpenSSL: the basic idea is that if an attack process saturates the multiplier, timing execution of issued multiplication instructions, it will notice variation iff. a target process attempts concurrent use of the multiplier. Modulo some issues such as synchronisation and noise inherent in similar strategies for cache-based attack, this allows the attacker to recover bits of the exponent by distinguishing between multi-precision square and multiply operations (which perform different limb-wise multiplications, and so make differing use of the associated execution unit). As an aside, the paper also gives some evidence that a longer exponent makes the attack easier (see also **MASCAB:Walter:03**).

MASCAB:AKMJLS:15 M. Andrysco et al. "On Subnormal Floating Point and Abnormal Timing". In: *IEEE Symposium on Security & Privicy (S&P)*. 2015, pp. 623–639.

> IEEE floating point uses a normalised representation; the mantissa will be scaled st. a binary point appears after the most-significant 1 bit, then represented without said bit (which is then implicit). A class of subnormal (or denormal) representations relaxes this, however, avoiding normalisation st. the implicit bit is 0. The values represented can be closer to zero as a result. This paper targets said special-case, 1) noting that arithmetic using them subnormal representations results in (much) longer latency, 2) harnessing the resulting data-dependent execution time to mount a side-channel attack on Firefox (under the pixel stealing model: the idea that an attacker web-page recovers information from a target web-page that should be isolated), and 3) outlining a (carefully designed) countermeasure, namely the library libfixedtimefixedpoint, that "patches" floating point operations with constant-time, fixed point analogues.

MASCAB:GEPT:09 J. Großschädl et al. "Side-Channel Analysis of Cryptographic Software via Early-Terminating Multiplications". In: *Information, Security and Cryptology (ICISC)*. LNCS 5984. Springer-Verlag, 2009, pp. 176–192.

> **MASCAB:SchWhi:98** explains that data-dependent execution time can stem from multiplier hardware that supports early-termination. This is an optimisation where using a multiplier (vs. multiplicand) operand of a certain form implies lower latency than in general (i.e., when using a random multiplier operand). The ARM7TMI includes such hardware, st. a multiplier operand with more contiguous MSBs equal to zero will imply lower latency (ranging between 2 and 5 clock cycles overall). This paper looks at this concrete platform, and 1) presents evidence the early-termination behaviour can be observed in a trace of power consumption (cf. Figure 1), 2) mounts a range of attacks on cryptographic primitives based on this fact (see Section 3), and 3) presents a countermeasure (see Section 4) that "patches" each multiplication instruction with a longer instruction sequence which is constant-time irrespective of the multiplier operand.

MASCAB:KohSha:17 D. Kohlbrenner and H. Shacham. "On the effectiveness of mitigations against floating-point timing channels". In: *USENIX Security Symposium*. 2017, pp. 69–81

MASCAB:KPVV:16 T. Kaufmann et al. "When Constant-Time Source Yields Variable-Time Binary: Exploiting Curve25519-donna Built with MSVC". In: *Cryptology and Network Security (CANS)*. LNCS

10052. Springer-Verlag, 2016, pp. 573–582.

> The paper does not *use* a micro-architectural resource per se, but *relates* to (micro-)architecture in a general sense. It describes an attack on an implementation[a] of the X25519 standard for Curve25519 (as used for ECDH) which follows RFC 7748. The implementation should be constant-time, but in a specific setting is not: on 64-bit platforms, the 32-bit MSVC compiler uses a run-time library (`llmul.asm`) to perform 64×64-bit multiplications and this library is *not* constant-time (it include a short-cut branch to deal efficiently with "short" inputs whose 32 MSBs are zero). This fact is exploited to recover the (secret) scalar multiplier by observing the execution time of (quite a lot of) scalar multiplications. Some of the statements, or at least terminology used in the paper have been a matter of debate; one way to interpret the discussion is as a question wrt. where the cause stems from, e.g., from the RFC, the implementation, the compiler, the (micro-)architecture. Specifically, the paper states that the implementation "follows RFC 7748 requirements"; this was critiqued by Bernstein[b] who quoted the RFC as noting it remains "important that the arithmetic used not leak information about the integers" and thus suggesting the implementation is actually non-compliant.
>
> [a] `http://github.com/agl/curve25519-donna`
> [b] See `http://twitter.com/hashbreaker/status/821067916330303488`.

MASCAB:WalSam:05 C.D. Walter and D. Samyde. "Data dependent power use in multipliers". In: *IEEE Symposium on Computer Arithmetic (ARITH)*. 2005, pp. 4–12.

> Depending on the precise definition, this paper could be deemed at the periphery of micro-architectural attacks: it focuses on the switching activity of multipliers, and so aligns more with general power analysis (e.g., DPA) attacks. However, in doing so it formulates a model so as to assess a variety of multiplier designs (which could be deemed part of a micro-architecture).

3.4.2 Positive use-cases: software-based countermeasures

MASCAB:BenHamouda:11 F. Ben Hamouda. "Exploration of efficiency and side-channel security of different implementations of RSA". 2011. URL: `http://www.normalesup.org/~fbenhamo/files/stage2011/report.pdf`.

> Figure $E.1$ gives a more efficient, constant-time instruction sequence for multiplication (vs. **MASCAB:GEPT:09**) on the ARM7TDMI (or some equivalent with a similar early-terminating multiplier).

MASCAB:RanLinTiw:16 A. Rane, C. Lin, and M. Tiwari. "Secure, Precise, and Fast Floating-Point Operations on x86 Processors". In: *USENIX Security Symposium*. 2016, pp. 71–86

3.5 Network-on-Chip (NoC)

3.6 Miscellaneous

3.6.1 Negative use-cases: attacks

MASCAB:Fomin:16 D.B. Fomin. "A timing attack on CUDA implementations of an AES-type block cipher". In: *Matematicheskie Voprosy Kriptografii* 7 (2 2016), pp. 121–130.

> This paper presents timing attack on AES: there are two distinguishing points vs. related work, namely 1) a GPU- rather than CPU-based platform is targeted; specifically, a CUDA-based implementation is executed on a NVIDIA-based platform, and 2) address-dependent variation in execution time that stems from bank conflicts in the (shared) memory is harnessed; this contrasts, for example, with a traditional, cache-based attack.

MASCAB:JiaFeiKae:17 Z.H. Jiang, Y. Fei, and D. Kaeli. "A Novel Side-Channel Timing Attack on GPUs". In: *Great Lakes Symposium on VLSI (GLSVLSI)*. 2017, pp. 167–172.

> Somewhat similarly to **MASCAB : Fomin : 16**, this paper harnesses bank conflicts (causing data-dependent execution time) in the shared memory of an Nvidia-based GPU to mount a time-based side-channel attack on AES.

3.6.2 Positive use-cases: hardware-based countermeasures

MASCAB:AweAus:17 Z.B. Aweke and T.M. Austin. "Ozone: Efficient Execution with Zero Timing Leakage for Modern Microarchitectures". In: *CoRR* abs/1703.07706 (2017). URL: `http://arxiv.org/abs/1703.07706`.

> Ozone is a system for supporting constant execution time, comprising two components: 1) A processor micro-architecture, prototyped using x86 via gem5. Conceptually, the idea is to isolate execution of security-critical program fragments; in that sense, there is some similarity with the design in **MASCAB:YumSav:15**. To do so, several resources are added, including a) dedicated, Ozone-specific state, and b) data and instruction stratch-pad memories. The security-critical program fragment executes, a) a fixed (micro-)architectural state on entry (e.g., of any branch predictor, meaning it cannot be primed by an attacker), b)*without* the possibility of interruption, c) out of scratch-pad memory (so avoiding cache-based attacks), d) within a WCET limit enforced by a watchdog timer. 2) A compiler, prototyped using LLVM. The goal is to translate an annotated program fragment into a form suitable for execution by the associated micro-architecture; there is an overview of the process in Figure 3, which includes a) performing standard transformations (e.g., if-conversion via `cmov`), b) verification of control-flow properties (e.g., non-existance of data-dependent conditional branches), and c) allocation of scratch-pad memory.

MASCAB:LiuMcG:09 I. Liu and D. McGrogan. *Elimination of Side Channel attacks on a Precision Timed Architecture*. Tech. rep. EECS-2009-15. Electrical Engineering and Computer Sciences, U.C. Berkley, 2009.

> This paper analyses the PRecision Timed Architecture (PRET) as a means of preventing side-channel attack; the PRET processor core is designed for real-time systems, and aims at *predictable* performance. It a) uses hardware-supported SMT, with per-thread scratch-pad memories ensuring inter-thread isolation (e.g., wrt. cache behaviour), and, perhaps more importantly, b) includes a notion of programmer-specified deadlines: an instruction can be annotated with a deadline (e.g., "this must always take x cycles"), which is enforced by the core during execution. Although the latter implies a *stall* for some number of cycles to meet the deadline (which may be observable via power analysis), it can be harnessed as a way to produce constant-time execution; all details aside, one could view this as a form of transparency and predictability wrt. execution time (stemming from the real-time use-case) not evident in many other cores and/or ISAs.

MASCAB:YumSav:15 K. Yumbul and E. Savaş. "Enhancing an Embedded Processor Core for Efficient and Isolated Execution of Cryptographic Algorithms". In: *The Computer Journal* 58.10 (2015), pp. 2368–2387.

> This paper presents an unnamed processor design, based on Tensilica Xtensa LX3 (a 32-bit RISC architecture) by prototype (although of more general applicability). The central idea is isolation of any cryptographic computation within a specially designed secure zone; per Figure 1, this includes a) a unified (data plus instruction) scratch-pad memory, b) a 128-bit cryptographic register file, c) various special-purpose functional units, such as a Block Cipher Unit (BCU). The connection to micro-architectural attacks is the stated aim of a) to avoid cache-based attacks: access to scratch-pad memory will be constant time, by virtue of *not* involving any caches.

MASCAB:ZonYen:16 A.D. Zonenberg and B. Yener. "Antikernel: A Decentralized Secure Hardware-Software Operating System Architecture". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 9813. Springer-Verlag, 2016, pp. 237–256.

> This paper presents a kernel design paradigm (the antikernel), wherein traditional kernel responsibilities are distributed and hence isolated: all interaction between kernel sub-systems is via message passing. To support this, it introduces a hardware implementation (named SARATOGA) with a MIPS-based core that allows hardware multi-threading; the cores are connected via a NoC. The paper is a good example of by-design security against micro-architectural attacks, in the sense 1) although implying a performance impact, the NoC is deterministic st. "no traffic sent by any other host can ever impact the ability of another to communicate and thus there are no timing/resource exhaustion side channels", 2) the L1 cache is essentially partitioned on a thread basis: each of the threads accesses a region of the cache whose separation is enforced in hardware.

3.6.3 Positive use-cases: software-based countermeasures

MASCAB:BraJanBon:15 B.A. Braun, S. Jana, and D. Boneh. "Robust and Efficient Elimination of Cache and Timing Side Channels". In: *CoRR* abs/1506.00189 (2015). URL: `http://arxiv.org/abs/1506.00189`.

> The use of constant-time implementation techniques is a generic approach to mitigation of time-driven side-channel attacks (which include selected cache-based attacks as a subset); irrespective of security-critical used, this hiding countermeasure forces the execution time of a target to be constant and thus prevents leakage of the former via observation of the latter. Realisation of this approach can be difficult; solutions can be platform-specific and fragile wrt. compilation and assembly steps, or minor variations in micro-architecture. This paper takes a different approach, applying the countermeasure at a higher level: at a fine-grained level, guided by programmer annotation of source code, the *kernel* will automatically 1) intelligently pad the execution time of processes to prevent leakage; this needs some care, but essentially forces the worst-case execution time for each process, 2) enforce isolation wrt. shared micro-architectural resources, e.g., using cache flushing and a form of page colouring.

3.6.4 Positive use-cases: analysis-based countermeasures (e.g., of source code)

MASCAB:IraEisSun:16:b G. Irazoqui, T. Eisenbarth, and B. Sunar. "MASScan: Stopping Microarchitectural Attacks Before Execution". Cryptology ePrint Archive, Report 2016/1196. 2016. URL: `http://eprint.iacr.org/2016/1196`.

> As outlined in Section 1, countermeasures against micro-architectural attack tend to be either preventive (i.e., prevent an attack by-design) or reactive (i.e., monitor execution, then react to prevent an attack). This paper takes a slightly different approach by presenting a program analysis tool called MASScan: it applies static analysis to each program provided as input, attempting to classify it as malicious iff. it satisfies metrics associated with such attacks (e.g., matches what is known wrt. attack programs). The metrics are instruction-focused on the whole; for example known x86 cache attack programs (e.g., per those in **MASCAB:Yarom**) utilise `rdtsc` for high-resolution timing, so this otherwise abnormal instruction can be used as evidence that a given program is malicious. The paper pitches MASScan as a means of vetting programs within an app store setting. Using a baseline corpus of 24 attack programs, the false-positive and -negative rates demonstrate the approach is effective up to a point: it miss-classifies 6% of benign programs (e.g., ffmpeg), with the reasons explained in Table 3.

4 Cross-cutting topics

4.1 Performance counters

4.1.1 Negative use-cases: attacks

MASCAB:BhaMuk:15 S. Bhattacharya and D. Mukhopadhyay. "Who Watches the Watchmen?: Utilizing Performance Monitors for Compromising Keys of RSA on Intel Platforms". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 9293. Springer-Verlag, 2015, pp. 248–266.

> At a high level, this paper is arguably best described by relating it to two other papers: 1) it is similar to **MASCAB:UhsGeoVer:08** in that it uses a similar underlying form of leakage: the attacker is assumed to have access to the hardware performance counters, and uses this ability to monitor the branch predictor unit (and thereby branch behaviour of the target process), 2) it is similar to **MASCAB:DKLMQW:98** in that the attack strategy (i.e., Algorithm 4) is more or less the same as a standard RSA timing attack, and targets conditional subtraction (and hence conditional branch) executed by Montgomery multiplication: the clear difference is that it uses a simulated branch predictor to model behaviour, and concrete branch predictor behaviour (collected via performance counters) as a distinguisher. The attack is applied to vanilla RSA, and the PKCS#1-based RSA-OAEP design.

MASCAB:BhaRebMuk:13 S. Bhattacharya, C. Rebeiro, and D. Mukhopadhyay. "Unraveling Timewarp: What All the Fuzz is About?" In: *Hardware and Architectural Support for Security and Privacy (HASP)*. 2013, 8:1–8:8.

> This paper demonstrates that TimeWarp **MASCAB:MarDemSet:12** does not prevent attacks succeeding; specifically, if the epoch time e is less than the time t associated with some operation or event of interest (e.g., a cache-miss for a particular memory access), then it is possible to eliminate the impact of TimeWarp by repeating the operation. Section 5 evaluates this approach in two settings (namely an artificial, access-driven setting, and a real, time-driven attack on CLEFIA), concluding TimeWarp reduces the probability of success (resp. increases the measurements required), but does not *prevent* success.

MASCAB:UhsGeoVer:08 L. Uhsadel, A. Georges, and I. Verbauwhede. "Exploiting Hardware Performance Counters". In: *Fault Diagnosis and Tolerance in Cryptography (FTDC)*. 2008, pp. 59–67.

This paper offers more concrete results that stem from a comment by Tiri **MASCAB:Tiri:07**, Section 2.3, namely that "*performance counters paint a more accurate picture and – if for a moment we ignore the fact that they can only be accessed using privileged ring-0 instructions–, they could enable better and faster attacks than the timestamp counter*". That is, it explores the idea of using performance counters in order to facilitate micro-architectural attack; intuitively, this approach should enable an *extremely* strong attacker due to the range and fidelity of accessible information. The approach is validated using an attack **MASCAB:BonMir:06** on AES as implemented in OpenSSL. While effective, there are a range of caveats in practice: In practice, however, there are a number of caveats: a) access to performance counters is often privileged, meaning that a user mode attacker cannot do so without assistance from the kernel (e.g., via a driver), b) the attacker needs to select a performance counting mode that, as much as possible, associates events specifically with the target process; the normal mode has a process count events wrt. itself, and any modes that count events for *all* processes and the kernel, for example, naturally introduce a lot of noise wrt. the event(s) of interest, and c) performance counters are not typically designed to support precise counting; there is some noise inherent in the results they yield, st. precisely distinguishing fine-grained events (e.g., say n vs. $n+1$ cache misses) is harder than might be expected. In addition, the attack replaces the access-based L2 cache eviction with use of the (privileged) `wbinvd` instruction: this requires support from an additional kernel driver, so could be deemed invasive vs. alternatives.

4.1.2 Positive use-cases: countermeasures

MASCAB:Hu:91 W.M. Hu. "Reducing timing channels with fuzzy time". In: *IEEE Symposium on Security & Privacy (S&P)*. 1991, pp. 8–20.

This paper discusses a concept it terms *fuzzy* time, which could be read as *randomised* time. Implemented as part of the VAX security kernel, fuzzy time aims to prevent covert timing channels by disallowing access to a precise timer; randomising the timer essentially adds noise, and so degrades the quality of measurements and hence channel bandwidth. The paper first surveys the timers available to an attacker: these could be classified as 1) real (Section 4.1 outlines a VAX system-supplied clock, that drives a timer interrupt and is accessible at a memory mapped address), or 2) synthesised (Section 4.2.2, for example, discusses how a timer could be derived from I/O activity). It then outlines how both classes can be randomised in Sections 5.1 and 5.2: the idea is to a) randomise the timer interrupt (initialising the interrupt controller with a random starting point) and reduce the resolution (to 1/10 of a second), and b) embellish the I/O subsystem with features that introduce a random delay between request and acknowledge (or notification) events. Evaluation of the implementation shows only minor degradation wrt. the system performance, while reducing covert timing channel bandwidth by up to two orders of magnitude (to below 10 bits per second).

MASCAB:MarDemSet:12 R. Martin, J. Demme, and S. Sethumadhavan. "TimeWarp: Rethinking Timekeeping and Performance Monitoring Mechanisms to Mitigate Side-channel Attacks". In: *International Symposium on Computer Architecture (ISCA)*. 2012, pp. 118–129.

This paper introduces TimeWarp, a mechanism for controlling access to the high-precision timer required by most micro-architectural attacks. Although it does not cite **MASCAB:Hu:91**, much of the analysis and design is similar: it first identifies three classes of timer, namely 1) hardware-based internal sources (e.g., `rdtsc`), 2) hardware-based external sources (e.g., interrupts stemming from network activity), and 3) software-only examples that allow synthesis of a timer, and then, a) provide an patch to (e.g., in the microcode for) `rdtsc` so it yields randomised behaviour when in a secure mode, b) reason that external sources cannot be delivered fast enough to offer a high enough resolution timer, and c) demonstrate that synthesised timers that use multiply threads (e.g., one to produce periodic events, one to consume them) can be detected by using the existing concept of a hardware race detector. The idea in point a) is to divide time into epochs, whose length varies randomly between 2^{e-1} and $2^e - 1$ cycles (for a configurable e, supplied by the `twrdtsc` instruction); execution of `rdtsc` is deferred until the next epoch boundary then subject to a second random delay of between 0 and $E - 1$ cycles. The timer value then has a third randomisation factor of between 0 and $E - 1$ cycles added to it. There is an emphasis throughout(e.g., Section 4) on the issue of (backward) compatibility: the patched timer must allow as many existing applications to work as possible, while still affording an improvement in security.

MASCAB:PopKli:74 G.J. Popek and C.S. Kline. "Verifiable Secure Operating System Software". In: *National Computer Conference and Exposition*. 1974, pp. 145–151.

This paper makes a case for delivering robust, verifiable guarantees of security (especially protection) in operating system kernels; it offers an overview of how a particular kernel at UCLA is designed specifically to do so. In a sense, it is amazing that such arguments were made and yet (bar notable exceptions such as seL4 **MASCAB:KEHACDEEKNSTW:09**) the same challenges still exist now. Among the discussion of motivation and design rationale, the confinement problem **MASCAB:Lampson:73** is cited as a challenge. It suggests a countermeasure to prevent (or block) such communication channels (e.g., covert channels): the idea is to disallow access to "real time" via a (high enough resolution) timer, only allowing access to a *virtual* timer from each virtual machine (a concept synonymous with process in the kernel discussed). Of course this only addresses timing channels, but, put another way, it means an attack process can only time activity relating to *itself* rather than some other, target process.

MASCAB:VatDasSha:11 B.C. Vattikonda, S. Das, and H. Shacham. "Eliminating Fine Grained Timers in Xen". In: *ACM Cloud Computing Security Workshop (CCSW)*. 2011, pp. 41–46.

The fuzzy time concept was conceived and realised in **MASCAB:Hu:91**. However, this paper notes that it remains unclear how the concept applies in a modern context; in particular, the x86 `rdtsc` instruction means an attack process can access a high-precision, cycle accurate timer from user space alone (i.e., without doing so via the kernel). Focusing on kernels executing under the Xen hypervisor, the paper describes and then analyses the impact of a) virtualising the `rdtsc` instruction using a Xen trap-and-emulate mechanism already present, then b) adjusting the timer value before return to the virtual machine, hence implementing a version of fuzzy time. The bulk of the paper considers masking the timer value LSBs, applying a step function and thereby reducing the resolution. The impact of doing so is limited wrt. performance, but there are limits wrt. correctness: it seems too little resolution impacts, for example, the network sub-system (e.g., the TCP stack uses `rdtsc` to measure RTT). Section 5.1 has a second option, which adds a random perturbation to the timer value while maintaining the monotonically increasing property.

4.2 IDS-like monitoring and detection

MASCAB:AlaSet:15 M. Alam and S. Sethi. "Detection of information leakage in cloud". In: *CoRR* abs/1504.03539 (2015). URL: `http://arxiv.org/abs/1504.03539`

MASCAB:ChiSavYil:16 M. Chiappetta, E. Savaş, and C. Yilmaz. "Real time detection of cache-based side-channel attacks using hardware performance counters". In: *Applied Soft Computing* 49 (2016), pp. 1162–1174.

This paper presents a countermeasure against micro-architectural attacks: the idea is for the kernel to dynamically monitor a process (e.g., using hardware performance counters), and decide whether or not the behaviour it exhibits suggests it is an attack process. Put another way, the idea is similar to an IDS in the sense the kernel detects and then deals with (e.g., terminates) an attack process at run-time; ideally the detection can be efficient enough to prevent an attack succeeding. Three different techniques for classifying processes as an attacker (or not), namely 1) statistical correlation, 2) anomaly detection, 3) supervised learning (using a neural network), with the latter two best described as instances of machine learning. An evaluation based on a Flush+Reload-based attack process highlights a trade-off between detection accuracy and latency, but also negligible overhead of applying each technique; either way, it is clear this attack (or at least this attack process) can be detected soon enough to prevent it succeeding.

MASCAB:Payer:16 M. Payer. "HexPADS: A Platform to Detect "Stealth" Attacks". In: *Engineering Secure Software and Systems (ESSoS)*. LNCS 9639. Springer-Verlag, 2016, pp. 138–154.

This paper describes HexPADS, a system-level countermeasure against micro-architectural attacks: as with **MASCAB:ChiSavYil:16**, the idea is to monitor execution of processes and classify them as malign (or not) based on their behaviour (i.e., it is an IDS of sorts). HexPADS is pitched as a user-space daemon process, although it seems reasonable that it *could* be integrated into a kernel (which reduces some overheads, and potentially offers easier access to some data). Given their remit is similar, a contrast with **MASCAB:ChiSavYil:16** seems a sensible way to describe the content. Positive aspects include the fact that 1) the system gathers and bases decisions on a *range* of data about execution of a given process, including hardware performance counters and the Linux `proc` file system, 2) this diversity allows it to addresses a *range* of attack types, including both cache-based side-channel attacks and Rowhammer attacks. On the other hand, 1) the detection mechanism is based on a rule set per attack type, which is arguably more rigid than say a machine learning based alternative, 2) the description and evaluation is somewhat informal, so a) it is unclear how parameters for the rule sets were derived, b) some aspects of the evaluation (e.g., false-positive rate, or time to detection, which is important in that it determines whether an attack can be prevented from completing) are unclear.

MASCAB:WDWMKW:14 J. Wu et al. "C^2Detector: a covert channel detection framework in cloud computing". In: *Security and Communication Networks* 7.3 (2014), pp. 544–557.

This paper presents a mechanism to detect exploitation of covert channels at run-time; there are two components, namely 1) a (single) hypervisor resident *captor*, which hooks and monitors hypercalls, passing record-based information to 2) a (per virtual machine supervisor (or kernel) resident *detector*, which analyses the records and then classifies associated activity as either a covert channel or not; Markov- and Bayesian-based models are used as classifiers. The mechanism can detect various covert channels, including (per Section 4.3.2) those based on use of the (shared) cache.

5 Peripheral topics

5.1 Caches: designs with general security properties

MASCAB:Inoue:06 K. Inoue. "Lock and Unlock: A Data Management Algorithm for A Security-Aware Cache". In: *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2006, pp. 1093–1096.

This paper is peripheral at best, in the sense it details a cache design intended to prevent buffer overflow attacks; the idea is quite involved, but basically uses the cache as a redundant, hardware-backed stack which can detect manipulation of return addresses. The peripheral connection to cache-based attacks stems from the inclusion of a mechanism for cache line locking: each line can be controlled st. eviction is prevented.

5.2 Caches: designs for low-power

MASCAB:KaxHuMar:01 S. Kaxiras, Z. Hu, and M. Martonosi. "Cache decay: exploiting generational behavior to reduce cache leakage power". In: *International Symposium on Computer Architecture (ISCA)*. 2001, pp. 240–251

MASCAB:KFBM:02 N.S. Kim et al. "Drowsy Instruction Caches: Leakage Power Reduction using Dynamic Voltage Scaling and Cache Sub-bank Prediction". In: *ACM/IEEE International Symposium on Microarchitecture (MICRO)*. 2002, pp. 219–230

MASCAB:KKMBM:02 K. Flautner et al. "Drowsy Caches: Simple Techniques for Reducing Leakage Power". In: *International Symposium on Computer Architecture (ISCA)*. 2002, pp. 148–157

MASCAB:PYFRV:00 M. Powell et al. "Gated-Vdd: A circuit technique to reduce leakage in deep-submicron cache memories". In: *International Symposium on Low Power Electronics and Design*. 2000, pp. 90–95

MASCAB:SubJonTop:13 K.T. Sundararajan, T.M. Jones, and N.P. Topham. "The Smart Cache: An Energy-Efficient Cache Architecture Through Dynamic Adaptation". In: *International Journal of Parallel Programming* 41.2 (2013), pp. 305–330

5.3 Caches: designs that harness partitioning

MASCAB:GonAliVal:95 A. González, C. Aliagas, and M. Valero. "A Data Cache with Multiple Caching Strategies Tuned to Different Types of Locality". In: *International Conference on Supercomputing (ICS)*. 1995, pp. 338–347

MASCAB:JuaRoyNav:95 T. Juan, D. Royo, and J.J. Navarro. "Dynamic Cache Splitting". In: *International Confernce of the Chilean Computational Society*. 1995

MASCAB:KVKSIG:01 S. Kim et al. "Power-aware Partitioned Cache Architectures". In: *International Symposium on Low Power Electronics and Design (ISLPED)*. 2001, pp. 64–67

MASCAB:PetOra:01 P. Petrov and A. Orailoglu. "Towards effective embedded processors in codesigns: customizable partitioned caches". In: *Hardware/Software Codesign (CODES)*. 2001, pp. 79–84

MASCAB:RacPat:03 P. Racunas and Y.N. Patt. "Partitioned First-level Cache Design for Clustered Microarchitectures". In: *International Conference on Supercomputing (ICS)*. 2003, pp. 22–31

MASCAB:RanAdvJou:00 P. Ranganathan, S. Adve, and N.P. Jouppi. "Reconfigurable Caches and Their Application to Media Processing". In: *International Symposium on Computer Architecture (ISCA)*. 2000, pp. 214–224

MASCAB:SanKoz:11 D. Sanchez and C. Kozyrakis. "Vantage: Scalable and efficient fine-grain cache partitioning". In: *International Symposium on Computer Architecture (ISCA)*. 2011, pp. 57–68

5.4 Attacks: results relating to RowHammer

MASCAB:BDGLS:16 F.F. Brasser et al. "CAn't Touch This: Practical and Generic Software-only Defenses Against Rowhammer Attacks". In: *CoRR* abs/1611.08396 (2016). URL: `http://arxiv.org/abs/1611.08396`

MASCAB:BhaMuk:16 S. Bhattacharya and D. Mukhopadhyay. "Curious Case of Rowhammer: Flipping Secret Exponent Bits Using Timing Analysis". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 9813. Springer-Verlag, 2016, pp. 602–624

MASCAB:GruMauMan:15 D. Gruss, C. Maurice, and S. Mangard. "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript". In: *CoRR* abs/1507.06955 (2015). URL: `http://arxiv.org/abs/1507.06955`

MASCAB:KDKFLLWLM:14 Y. Kim et al. "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors". In: *International Symposium on Computer architecuture (ISCA)*. 2014, pp. 361–372

MASCAB:QiaSea:16 R. Qiao and M. Seaborn. "A new approach for rowhammer attacks". In: *Hardware Oriented Security and Trust (HOST)*. 2016, pp. 161–166

MASCAB:VFLGMVBRG:16 V. van der Veen et al. "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms". In: *ACM Conference on Computer and Communications Security (CCS)*. 2016, pp. 1675–1689

5.5 Attacks: results relating to Address Space Layout Randomisation (ASLR)

MASCAB:JanLeeKim:16 Y. Jang, S. Lee, and T. Kim. "Breaking Kernel Address Space Layout Randomization with Intel TSX". In: *Conference on Computer and Communications Security (CCS)*. 2016, pp. 380–392

5.6 Attacks: results relating to test and debug mechanisms (e.g., scan-chain)

MASCAB:HuaChaMis:15 Y. Huang, A. Chattopadhyay, and P. Mishra. "Trace Buffer Attack: Security versus observability study in post-silicon debug". In: *Very Large Scale Integration (VLSI-SoC)*. 2015, pp. 355–360

MASCAB:HuaMis:17 Y. Huang and P. Mishra. "Trace Buffer Attack on the AES Cipher". In: *Journal of Hardware and Systems Security* 1 (1 2017), pp. 68–84.

> See also **MASCAB:HuaChaMis:15**. The ARM trace buffer is essentially an on-chip test and debug mechanism: one can configure it to capture a subset of signals for some period, and then export the result for analysis. In a sense, this yields a problem of the same style as scan-chain based attacks, namely a tension between the needs of observability and security.

5.7 Attacks: use of performance degradation

MASCAB:TsaEtsFei:07 D. Tsafrir, Y. Etsion, and D.G. Feitelson. "Secretly Monopolizing the CPU Without Superuser Privileges". In: *USENIX Security Symposium*. 2007, 17:1–17:18

5.8 Attacks: techniques for key recovery

MASCAB:HenSha:09 N. Heninger and H. Shacham. "Reconstructing RSA Private Keys from Random Key Bits". In: *Advances in Cryptology (CRYPTO)*. LNCS 5677. Springer-Verlag, 2009, pp. 1–17

MASCAB:PatPolSib:12 K.G. Paterson, A. Polychroniadou, and D.L. Sibborn. "A Coding-Theoretic Approach to Recovering Noisy RSA Keys". In: *Advances in Cryptology (ASIACRYPT)*. LNCS 7658. Springer-Verlag, 2012, pp. 386–403

MASCAB:PoeSib:15 B. Poettering and D.L. Sibborn. "Cold Boot Attacks in the Discrete Logarithm Setting". In: *Topics in Cryptology (CT-RSA)*. LNCS 9048. Springer-Verlag, 2015, pp. 449–465

MASCAB:Walter:03 C.D. Walter. "Longer Keys May Facilitate Side Channel Attacks". In: *Selected Areas in Cryptography (SAC)*. LNCS 3006. Springer-Verlag, 2003, pp. 42–57

5.9 Attacks: alterative paradigms

MASCAB:GBHP:16 S. Ghandali et al. "A Design Methodology for Stealthy Parametric Trojans and Its Application to Bug Attacks". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 9813. Springer-Verlag, 2016, pp. 625–647

MASCAB:GruGhi:02 D. Grunwald and S. Ghiasi. "Microarchitectural Denial of Service: Insuring Microarchitectural Fairness". In: *ACM/IEEE International Symposium on Microarchitecture (MICRO)*. 2002, pp. 409–418

MASCAB:RNRDBS:15 L. Riviére et al. "High precision fault injections on the instruction cache of ARMv7-M architectures". In: *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 2015, pp. 62–67

MASCAB:TanSetSto:17 A. Tang, S. Sethumadhavan, and S. Stolfo. "CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management". In: *USENIX Security Symposium*. 2017, pp. 1057–1074.

> Dynamic Frequency and Voltage Scaling (DVFS) is a mechanism to manage the energy consumption of, for example, processor cores; the idea is the core voltage level and/or clock frequency can be dynamically controlled, which allows them to be reduced during periods of relative inactivy in order to reduce energy consumption. On ARM-based platforms using Android, DVFS is managed by the kernel. The paper shows that *if* a malicious kernel driver can gain control over DVFS, this abilility can be leverage to drive the core beyond the stated operational limits; the result is fault(s) during execution, which can be capitalised upon per traditional fault attacks (e.g., based on clock or power glitching: in essence, this is a software-defined fault vs. a fault injected by physical means).

5.10 Verification

MASCAB:ABBDE:16 J.B. Almeida et al. "Verifying Constant-Time Implementations". In: *USENIX Security Symposium*. 2016, pp. 53–70

MASCAB:CGMH:14 D. Cock et al. "The last mile: An empirical study of some timing channels on seL4". In: *Computer and Communications Security (CCS)*. 2014, pp. 570–581.

> Although this paper is interesting for other reasons, a (subtle) point to note is use of the (information theory based) channel matrix to visualise timing channels: points in the matrix (visualised as a heat map) capture the probability of observing each output symbol, given transmission of an input symbol.

MASCAB:MMBGBSLGK:13 T. Murray et al. "seL4: from general purpose to a proof of information flow enforcement". In: *IEEE Symposium on Security & Privicy (S&P)*. 2013, pp. 415–429

5.11 Miscellaneous

MASCAB:AraThu:07 S. Aravamuthan and V.R. Thumparthy. "A Parallelization of ECDSA Resistant to Simple Power Analysis Attacks". In: *Communication Systems Software and Middleware (COMSWARE)*. 2007, pp. 1–7

MASCAB:CosDev:16 V. Costan and S. Devadas. "Intel SGX Explained". Cryptology ePrint Archive, Report 2016/086. 2016. URL: `http://eprint.iacr.org/2016/086`

MASCAB:DKLMQW:98 J.-F. Dhem et al. "A Practical Implementation of the Timing Attack". In: *Smart Card Research and Advanced Applications (CARDIS)*. LNCS 1820. Springer-Verlag, 1998, pp. 167–182

MASCAB:MulDewFre:10 T. Müller, A. Dewald, and F.C. Freiling. "AESSE: A Cold-boot Resistant Implementation of AES". In: *European Workshop on System Security (EUROSEC)*. 2010, pp. 42–47

MASCAB:PieLomVil:16 R. Di Pietro, F. Lombardi, and A. Villani. "CUDA Leaks: A Detailed Hack for CUDA and a (Partial) Fix". In: *ACM Transactions on Embedded Computing Systems (TECS)* 15.1 (2016), 15:1–15:25.

This paper is included mainly to highlight the fact it is *not* about information leakage in the sense of side-channel attacks: the leakage here is (arguably) closer aligned with the concept of data remnants, stemming from inadequate access controlled to various GPU memory structures (by hardware and/or driver infrastructure).

6 Residual unclassified entries

MASCAB:AciGueSei:07 O. Acıiçmez, S. Gueron, and J.-P. Seifert. "New Branch Prediction Vulnerabilities in OpenSSL and Necessary Software Countermeasures". In: *IMA Conference on Cryptography and Coding (IMACC)*. LNCS 4887. Springer-Verlag, 2007, pp. 185–203

MASCAB:AciKocSei:07 O. Acıiçmez, Ç.K. Koç, and J.-P. Seifert. "On the Power of Simple Branch Prediction Analysis". In: *ACM Symposium on Information, Computer and Communications Security*. 2007, pp. 312–320

MASCAB:AciSeiKoc:07:a O. Acıiçmez, J.-P. Seifert, and Ç.K. Koç. "Predicting Secret Keys via Branch Prediction". In: *Topics in Cryptology (CT-RSA)*. LNCS 4377. Springer-Verlag, 2007, pp. 225–242

MASCAB:AciSeiKoc:07:b W. Schindler O. Acıiçmez and Ç.K. Koç. "Cache Based Remote Timing Attacks on the AES". In: *Topics in Cryptology (CT-RSA)*. LNCS 4377. Springer-Verlag, 2007, pp. 271–286

MASCAB:AgoPel:06 G. Agosta and G. Pelosi. "Countermeasures for the Simple Branch Prediction Analysis". Cryptology ePrint Archive, Report 2006/482. 2006. URL: `http://eprint.iacr.org/2006/482.pdf`

MASCAB:AHFG:10 A. Aviram et al. "Determinating Timing Channels in Compute Clouds". In: *ACM Workshop on Cloud Computing Security Workshop (CCSW)*. 2010, pp. 103–108

MASCAB:AlaJayRag:11 J. Alawatugoda, D. Jayasinghe, and R. Ragel. "Countermeasures against Bernstein's remote cache timing attack". In: *IEEE International Conference on Industrial and Information Systems (ICIIS)*. See also `http://arxiv.org/pdf/1403.7297`. 2011, pp. 43–48

MASCAB:AlyElG:13 H. Aly and M. ElGayyar. "Attacking AES Using Bernstein's Attack on Modern Processors". In: *Progress in Cryptology (AFRICACRYPT)*. LNCS 7918. Springer-Verlag, 2013, pp. 127–139

MASCAB:AncChaRoy:14 D.M. Ancajas, K. Chakraborty, and S. Roy. "Fort-NoCs: Mitigating the Threat of a Compromised NoC". In: *Design Automation Conference (DAC)*. 2014, 158:1–158:6

MASCAB:AshGirMen:16 C. Ashokkumar, R. P. Giri, and B. Menezes. "Highly Efficient Algorithms for AES Key Retrieval in Cache Access Attacks". In: *IEEE European Symposium on Security & Privacy (EuroS&P)*. 2016, pp. 261–275

MASCAB:AtiYilSav:13 A.C. Atici, C. Yilmaz, and E. Savaş. "An Approach for Isolating the Sources of Information Leakage Exploited in Cache-Based Side-Channel Attacks". In: *Software Security and Reliability-Companion (SERE-C)*. 2013, pp. 74–83

MASCAB:AtiYilSav:16 A.C. Atici, C. Yilmaz, and E. Savaş. "Remote Cache-Timing Attack without Learning Phase". Cryptology ePrint Archive, Report 2016/002. 2016. URL: `http://eprint.iacr.org/2016/002`

MASCAB:AWHF:10 A. Aviram et al. "Efficient System-enforced Deterministic Parallelism". In: *USENIX Conference on Operating Systems Design and Implementation (OSDI)*. 2010, pp. 193–206

MASCAB:BEPW:10 A. Bogdanov et al. "Differential Cache-Collision Timing Attacks on AES with Applications to Embedded CPUs". In: *Topics in Cryptology (CT-RSA)*. LNCS 5985. Springer-Verlag, 2010, pp. 235–251

MASCAB:Bernstein:05 D.J. Bernstein. "Cache-timing Attacks on AES". 2005. URL: `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`

MASCAB:BGNS:06 E.F. Brickell et al. "Software mitigations to hedge AES against cache-based software side channel vulnerabilities". Cryptology ePrint Archive, Report 2006/052. 2006. URL: `http://eprint.iacr.org/2006/052.pdf`

MASCAB:BhaRebMuk:12 S. Bhattacharya, C.D. Rebeiro, and D. Mukhopadhyay. "Hardware Prefetchers Leak: A Revisit of SVF for Cache-Timing Attacks". In: *IEEE/ACM International Symposium on Microarchitecture Workshops (MICROW)*. 2012, pp. 17–23

MASCAB:BloKru:07 J. Blömer and V. Krummel. "Analysis of Countermeasures Against Access Driven Cache Attacks on AES". In: *Selected Areas in Cryptography (SAC)*. LNCS 4876. Springer-Verlag, 2007, pp. 96–109

MASCAB:BMRM:16 S. Briongos et al. "Modeling Side-channel Cache Attacks on AES". In: *Summer Computer Simulation Conference (SCSC)*. 2016, 37:1–37:8

MASCAB:Bon:06 J. Bonneau. "Robust Final-Round Cache-Trace Attacks Against AES". Cryptology ePrint Archive, Report 2006/374. 2006. URL: `http://eprint.iacr.org/2006/374.pdf`

MASCAB:BonMir:06 J. Bonneau and I. Mironov. "Cache-Collision Timing Attacks Against AES". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 4249. Springer-Verlag, 2006, pp. 201–215

MASCAB:BPSY:14 N. Benger et al. ""Ooh Aah... Just a Little Bit" : A Small Amount of Side Channel Can Go a Long Way". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 8731. Springer-Verlag, 2014, pp. 75–92

MASCAB:BRPG:15 A. Barresi et al. "CAIN: Silently Breaking ASLR in the Cloud". In: *USENIX Workshop on Offensive Technologies (WOOT)*. 2015

MASCAB:BruHak:09 B.B. Brumley and R. Hakala. "Cache-Timing Template Attacks". In: *Advances in Cryptology (ASIACRYPT)*. LNCS 5912. Springer-Verlag, 2009, pp. 667–684

MASCAB:BruTuv:11 B.B. Brumley and N. Tuveri. "Cache-Timing Attacks and Shared Contexts". In: *Constructive Side-Channel Analysis and Secure Design (COSADE)*. 2011

MASCAB:CHBLF:15 S. Crane et al. "Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity". In: *Network and Distributed System Security Symposium (NDSS)*. 2015

MASCAB:CheVen:14 J. Chen and G. Venkataramani. "CC-Hunter: Uncovering Covert Timing Channels on Shared Processor Hardware". In: *IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2014, pp. 216–228

MASCAB:CleCopSut:12 J.V. Cleemput, B. Coppens, and B. De Sutter. "Compiler Mitigations for Time Attacks on Modern x86 Processors". In: *ACM Transactions on Architecture and Code Optimisation (TACO)* 8.4 (2012), 23:1–23:20

MASCAB:CMXHPSZ:14 A. Chen et al. "Detecting Covert Timing Channels with Time-deterministic Replay". In: *USENIX Conference on Operating Systems Design and Implementation (OSDI)*. 2014, pp. 541–554

MASCAB:CosLebDev:16 V. Costan, I. Lebedev, and S. Devadas. "Sanctum: Minimal Hardware Extensions for Strong Software Isolation". In: *USENIX Security Symposium*. 2016, pp. 857–874

MASCAB:CVBS:09 B. Coppens et al. "Practical Mitigation for Timing-Based Side-Channel Attacks on Modern x86 Processors". In: *IEEE Symposium on Security & Privacy (S&P)*. 2009, pp. 45–60

MASCAB:CWKCL:13 C. Chen et al. "Improvement of Trace-driven I-Cache Timing Attack on the RSA Algorithm". In: *Journal of Systems and Software* 86.1 (2013), pp. 100–107

MASCAB:CZRZ:16 S. Chen et al. "Detecting Privileged Side-Channel Attacks in Shielded Execution with Déjá Vu". In: *ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. 2017

MASCAB:DJLAP:12 L. Domnitser et al. "Non-monopolizable Caches: Low-complexity Mitigation of Cache Side Channel Attacks". In: *ACM Transactions on Architure and Code Optimisation (TACO)* 8.4 (2012), 35:1–35:21

MASCAB:DMWS:12 J. Demme et al. "Side-channel Vulnerability Factor: A Metric for Measuring Information Leakage". In: *International Symposium on Computer Architecture (ISCA)*. 2012, pp. 106–117

MASCAB:DMWS:13 J. Demme et al. "A Quantitative, Experimental Approach to Measuring Processor Side-Channel Security". In: *IEEE Micro* 33.3 (2013), pp. 68–77

MASCAB:EvtPon:16 D. Evtyushkin and D. Ponomarev. "Covert Channels Through Random Number Generator: Mechanisms, Capacity Estimation and Mitigations". In: *ACM Conference on Computer and Communications Security (CCS)*. 2016, pp. 843–857

MASCAB:EvtPonAbu:15 D. Evtyushkin, D. Ponomarev, and N. Abu-Ghazaleh. "Covert Channels Through Branch Predictors: A Feasibility Study". In: *Hardware and Architectural Support for Security and Privacy (HASP)*. 2015, 5:1–5:8

MASCAB:EvtPonAbu:16:a D. Evtyushkin, D.V. Ponomarev, and N.B. Abu-Ghazaleh. "Jump over ASLR: Attacking branch predictors to bypass ASLR". In: *ACM/IEEE International Symposium on Microarchitecture (MICRO)*. 2016, pp. 1–13

MASCAB:EvtPonAbu:16:b D. Evtyushkin, D. Ponomarev, and N. Abu-Ghazaleh. "Understanding and Mitigating Covert Channels Through Branch Predictors". In: *ACM Transactions on Architecture and Code Optimisation (TACO)* 13.1 (2016), 10:1–10:23

MASCAB:FioPalSil:08 L. Fiorin, G. Palermo, and C. Silvano. "A Security Monitoring Service for NoCs". In: *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. 2008, pp. 197–202

MASCAB:GalKiz:11 J.-F. Gallais and I. Kizhvatov. "Error-Tolerance in Trace-Driven Cache Collision Attacks". In: *Constructive Side-Channel Analysis and Secure Design (COSADE)*. 2011

MASCAB:GalKizTun:10 J.-F. Gallais, I. Kizhvatov, and M. Tunstall. "Improved trace-driven cache-collision attacks against embedded AES implementations". In: *Information Security Applications (WISA)*. LNCS 6513. Springer-Verlag, 2010, pp. 243–257

MASCAB:GayManSub:15 R. Gay, H. Mantel, and H. Sudbrock. "An Empirical Bandwidth Analysis of Interrupt-Related Covert Channels". In: *International Journal of Secure Software Engineering* 6.2 (2015), pp. 1–22

MASCAB:GiaWan:07 S. Gianvecchio and H. Wang. "Detecting Covert Timing Channels: An Entropy-based Approach". In: *ACM Conference on Computer and Communications Security (CCS)*. 2007, pp. 307–316

MASCAB:GIIES:15 B. Gülmezoglu et al. "A Faster and More Realistic Flush+Reload Attack on AES". In: *Constructive Side-Channel Analysis and Secure Design (COSADE)*. LNCS 9064. Springer-Verlag, 2015, pp. 111–126

MASCAB:Gligor:93 V. Gligor. *A Guide to Understanding Covert Channel Analysis of Trusted Systems*. Tech. rep. ADA276418. 1993. URL: `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA276418`

MASCAB:GMFLM:16 D. Gruss et al. "Prefetch Side-Channel Attacks: Bypassing SMAP and Kernel ASLR". In: *Computer and Communications Security (CCS)*. 2016, pp. 368–379

MASCAB:GNBD:16 R. Guanciale et al. "Cache Storage Channels: Alias-Driven Attacks and Verified Countermeasures". In: *IEEE Symposium on Security & Privacy (S&P)*. 2016, pp. 38–55

MASCAB:Godfrey:13 M. Godfrey. "On the Prevention of Cache-Based Side-Channel Attacks in a Cloud Environment". MA thesis. Queen's University, 2013

MASCAB:GodZul:13 M. Godfrey and M. Zulkernine. "A Server-Side Solution to Cache-Based Side-Channel Attacks in the Cloud". In: *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)*. 2013, pp. 163–170

MASCAB:GolLinCud:84 B.D. Gold, R.R. Linde, and P.F. Cudney. "KVM/370 in Retrospect". In: *IEEE Symposium on Security & Privacy (S&P)*. 1984, pp. 13–23

MASCAB:GraGroPag:07 P. Grabher, J. Großschädl, and D. Page. "Cryptographic Side-Channels from Low Power Cache Memory". In: *IMA Conference on Cryptography and Coding (IMACC)*. LNCS 4887. Springer-Verlag, 2007, pp. 170–184

MASCAB:Gray:94 J.W. Gray. *Countermeasures and tradeoffs for a class of covert timing channels*. Tech. rep. 1783.1-25. 1994. URL: `http://hdl.handle.net/1783.1/25`

MASCAB:HenWilHol:13 R. Hund, C. Willems, and T. Holz. "Practical Timing Side Channel Attacks Against Kernel Space ASLR". In: *IEEE Symposium on Security & Privacy (S&P)*. 2013, pp. 191–205

MASCAB:HKRDVT:15 C. Hunger et al. "Understanding contention-based channels and using them for defense". In: *High Performance Computer Architecture (HPCA)*. 2015, pp. 639–650

MASCAB:Hu:92 W.M. Hu. "Lattice Scheduling and Covert Channels". In: *IEEE Symposium on Security & Privacy (S&P)*. 1992, pp. 52–61

MASCAB:IES:15 G. Irazoqui, T. Eisenbarth, and B. Sunar. "Systematic Reverse Engineering of Cache Slice Selection in Intel Processors". In: *Digital System Design (DSD)*. 2015, pp. 629–636

MASCAB:IGES:16 M.S. Inci et al. "Co-location detection on the Cloud". In: *Constructive Side-Channel Analysis and Secure Design (COSADE)*. LNCS 9689. Springer-Verlag, 2016, pp. 19–34

MASCAB:IGIES:15 M.S. Inci et al. "Seriously, get off my cloud! Cross-VM RSA Key Recovery in a Public Cloud". Cryptology ePrint Archive, Report 2015/898. 2015. URL: `http://eprint.iacr.org/2015/898`

MASCAB:IGIES:16 M.S. Inci et al. "Cache Attacks Enable Bulk Key Recovery on the Cloud". In: *Cryptographic Hardware and Embedded Systems (CHES)*. LNCS 9813. Springer-Verlag, 2016, pp. 368–388

MASCAB:IIES:14 G. Irazoqui et al. "Wait a Minute! A fast, Cross-VM Attack on AES". In: *Recent Advances in Intrusion Detection (RAID)*. LNCS 8688. Springer-Verlag, 2014, pp. 299–319

MASCAB:IIES:15:a G. Irazoqui et al. "Lucky 13 Strikes Back". In: *ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. 2015, pp. 85–96

MASCAB:IraEisSun:14 G. Irazoqui, T. Eisenbarth, and B. Sunar. "Jackpot: Stealing Information From Large Caches via Huge Pages". Cryptology ePrint Archive, Report 2014/970. 2014. URL: `http://eprint.iacr.org/2014/970`

MASCAB:IraEisSun:15 G. Irazoqui, T. Eisenbarth, and B. Sunar. "S$A: A Shared Cache Attack That Works across Cores and Defies VM Sandboxing – and Its Application to AES". In: *IEEE Symposium on Security & Privacy (S&P)*. 2015, pp. 591–604

MASCAB:IraEisSun:16:a G. Irazoqui, T. Eisenbarth, and B. Sunar. "Cross Processor Cache Attacks". In: *ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. 2016, pp. 353–364

MASCAB:JFHR:10 D. Jayasinghe et al. "Remote Cache Timing Attack on Advanced Encryption Standard and countermeasures". In: *Information and Automation for Sustainability (ICIAFs)*. 2010, pp. 177–182

MASCAB:JoyTun:07 M. Joye and M. Tunstall. "Securing OpenSSL against micro-architectural attacks". In: *Security and Cryptography (SECRYPT)*. 2007, pp. 189–196

MASCAB:KadKiy:13 S. Kadloor and N. Kiyavash. "Exploiting Timing Side Channel in Secure Cloud Scheduling". In: *High Performance Cloud Auditing and Applications*. Ed. by K.J. Han, B.-Y. Choi, and S. Song. Springer, 2013. Chap. 6, pp. 147–168

MASCAB:KAPJ:16 M. Kayaalp et al. "A high-resolution side-channel attack on last-level cache". In: *Design Automation Conference (DAC)*. 2016, 72:1–72:6

MASCAB:KarWra:91 P.A. Karger and J.C. Wray. "Storage channels in disk arm optimization". In: *IEEE Symposium on Security & Privacy (S&P)*. 1991, pp. 52–61

MASCAB:KASZ:08 J. Kong et al. "Deconstructing new cache designs for thwarting software cache-based side channel attacks". In: *ACM Workshop on Computer Security Architecture (CSAW)*. 2008, pp. 25–34

MASCAB:KASZ:09 J. Kong et al. "Hardware-Software Integrated Approaches to Defend Against Software Cache-based Side Channel Attacks". In: *High-Performance Computer Architecture (HPCA)*. 2009, pp. 393–404

MASCAB:KEHACDEEKNSTW:09 G. Klein et al. "seL4: Formal Verification of an OS Kernel". In: *ACM Symposium on Operating Systems Principles (SOSP)*. 2009, pp. 207–220

MASCAB:Kemmerer:83 R.A. Kemmerer. "Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels". In: *ACM Transactions on Computer Systems* 1.3 (1983), pp. 256–277

MASCAB:KimPeiMai:12 T. Kim, M. Peinado, and G. Mainar-Ruiz. "StealthMem: System-level Protection Against Cache-based Side Channel Attacks in the Cloud". In: *USENIX Security Symposium*. 2012, 11:1–11:16

MASCAB:KKKH:15 D. Kim et al. "vCache: Architectural Support for Transparent and Isolated Virtual LLCs in Virtualized Environments". In: *ACM/IEEE International Symposium on Microarchitecture (MICRO)*. 2015, pp. 623–634

MASCAB:KopBas:07 B. Köpf and D. Basin. "An Information-theoretic Model for Adaptive Side-channel Attacks". In: *Computer and Communications Security (CCS)*. 2007, pp. 286–296

MASCAB:Lauradoux:05 C. Lauradoux. "Collision attacks on processors with cache and countermeasures". In: *Western European Workshop on Research in Cryptology (WEWoRC)*. 2005, pp. 76–85

MASCAB:LGASSK:09 Y. Liu et al. "Hide and Seek in Time: Robust Covert Timing Channels". In: *European Conference on Research in Computer Security (ESORICS)*. 2009, pp. 120–135

MASCAB:LHMHTS:15 C. Liu et al. "GhostRider: A Hardware-Software System for Memory Trace Oblivious Computation". In: *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 2015, pp. 87–101

MASCAB:LiGaoRei:13 P. Li, D. Gao, and M.K. Reiter. "Mitigating Access-driven Timing Channels in Clouds Using StopWatch". In: *Dependable Systems and Networks Workshop (DSNW)*. 2013, pp. 1–12

MASCAB:LiuLee:13 F. Liu and R.B. Lee. "Security Testing of a Secure Cache Design". In: *Hardware and Architectural Support for Security and Privacy (HASP)*. 2013, 3:1–3:8

MASCAB:LiuLee:14 F. Liu and R.B. Lee. "Random Fill Cache Architecture". In: *IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2014, pp. 203–215

MASCAB:LiuWuLee:15 F. Liu, H. Wu, and R.B. Lee. "Can Randomized Mapping Secure Instruction Caches from Side-channel Attacks?" In: *Hardware and Architectural Support for Security and Privacy (HASP)*. 2015, 4:1–4:8

MASCAB:LYGHL:15 F. Liu et al. "Last-Level Cache Side-Channel Attacks are Practical". In: *IEEE Symposium on Security & Privacy (S&P)*. 2015, pp. 605–622

MASCAB:ManSub:09 H. Mantel and H. Sudbrock. "Formal Aspects in Security and Trust". In: ed. by P. Degano, J. Guttman, and F. Martinelli. Springer-Verlag, 2009. Chap. Information-Theoretic Modeling and Analysis of Interrupt-Related Covert Channels, pp. 67–81

MASCAB:MilMilKul:04 M. Milenkovic, A. Milenkovic, and J. Kulick. "Microbenchmarks for Determining Branch Predictor Organization". In: *Software – Practice & Experience* 34.5 (2004), pp. 465–487

MASCAB:MinTsuTsu:03 K. Minematsu, Y. Tsunoo, and E. Tsujihara. "An Analysis on Success Probability of Cache Attack". In: *Symposium on Cryptography and Information Security (SCIS)*. (in Japanese). 2003

MASCAB:MLSTSAKS:13 M. Maas et al. "PHANTOM: Practical Oblivious Computation in a Secure Processor". In: *ACM Conference on Computer and Communications Security (CCS)*. 2013, pp. 311–324

MASCAB:MNHF:15 C. Maurice et al. "C5: Cross-Cores Cache Covert Channel". In: *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. LNCS 9148. Springer-Verlag, 2015, pp. 46–64

MASCAB:MooSekRei:15 S.-J. Moon, V. Sekar, and M.K. Reiter. "Nomad: Mitigating Arbitrary Cloud Side Channels via Provider-Assisted Migration". In: *ACM Conference on Computer and Communications Security (CCS)*. 2015, pp. 1595–1606

MASCAB:MowKeeSha:12 K. Mowery, S. Keelveedhi, and H. Shacham. "Are AES x86 Cache Timing Attacks Still Feasible?" In: *ACM Cloud Computing Security Workshop (CCSW)*. 2012, pp. 19–24

MASCAB:MSNHF:15 C. Maurice et al. "Reverse Engineering Intel Last-Level Cache Complex Addressing Using Performance Counters". In: *Recent Advances in Intrusion Detection (RAID)*. LNCS 9404. Springer-Verlag, 2015, pp. 48–65

MASCAB:MWSGGBRM:17 C. Maurice et al. "Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud". In: *Network and Distributed System Security Symposium (NDSS)*. 2017

MASCAB:NevSei:06 M. Neve and J-.P. Seifert. "Advances on Access-Driven Cache Attacks on AES". In: *Selected Areas in Cryptography (SAC)*. LNCS 4356. Springer-Verlag, 2006, pp. 147–162

MASCAB:NevSeiWan:06 M. Neve, J.-P. Seifert, and Z. Wang. "A refined look at Bernstein's AES side-channel analysis". In: *ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. 2006, pp. 369–369

MASCAB:NevTir:07 M. Neve and K. Tiri. "On the complexity of side-channel attacks on AES-256: Methodology and quantitative results on cache attacks". Cryptology ePrint Archive, Report 2007/318. 2007. URL: `http://eprint.iacr.org/2007/318.pdf`

MASCAB:OHaTon:05 M. O'Hanlon and A. Tonge. *Investigation Of Cache-Timing Attacks On AES*. Tech. rep. 2005. URL: `http://www.sci-sym.dcu.ie/wpapers/2005/0105.pdf`

MASCAB:OKSK:15 Y. Oren et al. "The Spy in the Sandbox: Practical Cache Attacks in JavaScript and Their Implications". In: *ACM Conference on Computer and Communications Security (CCS)*. 2015, pp. 1406–1418

MASCAB:OKSM:03 K. Ohkuma et al. "Key Inference in a Side-Channel Attack Based on Cache Miss". In: *Symposium on Cryptography and Information Security (SCIS)*. (in Japanese). 2003

MASCAB:Page:02 D. Page. "Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel". Cryptology ePrint Archive, Report 2002/169. 2002. URL: `http://eprint.iacr.org/2002/169.pdf`

MASCAB:Page:03 D. Page. "Defending Against Cache Based Side-Channel Attacks". In: *Information Security Technical Report* 8.1 (2003), pp. 30–44

MASCAB:Page:05 D. Page. "Partitioned Cache Architecture as a Side-Channel Defence Mechanism". Cryptology ePrint Archive, Report 2005/280. 2005. URL: `http://eprint.iacr.org/2005/280.pdf`

MASCAB:PBPMB:17 Q.-S. Phan et al. "Synthesis of Adaptive Side-Channel Attacks". In: *Computer Security Foundations Symposium (CSF)*. 2017, pp. 328–342

MASCAB:PGMSM:16 P. Pessl et al. "DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks". In: *USENIX Security Symposium*. 2016, pp. 565–581

MASCAB:PodDatReb:11 R. Poddar, A. Datta, and C.D. Rebeiro. "A Cache Trace Attack on CAMELLIA". In: *Security Aspects in Information Technology*. LNCS 7011. Springer-Verlag, 2011, pp. 144–156

MASCAB:PolSmaYar:15 J. van de Pol, N.P. Smart, and Y. Yarom. "Just a Little Bit More". In: *Topics in Cryptology (CT-RSA)*. LNCS 9048. Springer-Verlag, 2015, pp. 3–21

MASCAB:RanLinTiw:15 A. Rane, C. Lin, and M. Tiwari. "Raccoon: Closing Digital Side-channels Through Obfuscated Execution". In: *USENIX Security Symposium*. 2015, pp. 431–446

MASCAB:RebMonMuk:10 C.D. Rebeiro, M. Mondal, and D. Mukhopadhyay. "Pinpointing Cache Timing Attacks on AES". In: *International Conference on VLSI Design (VLSID)*. 2010, pp. 306–311

MASCAB:RebMuk:11 C.D. Rebeiro and D. Mukhopadhyay. "Cryptanalysis of CLEFIA Using Differential Methods with Cache Trace Patterns". In: *Topics in Cryptology (CT-RSA)*. LNCS 6558. Springer-Verlag, 2011, pp. 89–103

MASCAB:RebMuk:12 C.D. Rebeiro and D. Mukhopadhyay. "Boosting Profiled Cache Timing Attacks With A Priori Analysis". In: *IEEE Transactions on Information Forensics and Security* 7.6 (2012), pp. 1900–1905

MASCAB:RebMuk:13 C.D. Rebeiro and D. Mukhopadhyay. "Micro-Architectural Analysis of Time-Driven Cache Attacks: Quest for the Ideal Implementation". In: *IEEE Transactions on Computers* 64.3 (2013), pp. 778–790

MASCAB:RMTF:09 C.D. Rebeiro et al. "Cache Timing Attacks on Clefia". In: *Progress in Cryptology (INDOCRYPT)*. LNCS 5922. Springer-Verlag, 2009, pp. 104–118

MASCAB:RPDM:11 C.D. Rebeiro et al. "An Enhanced Differential Cache Attack on CLEFIA for Large Cache Lines". In: *Progress in Cryptology (INDOCRYPT)*. LNCS 7107. Springer-Verlag, 2011, pp. 58–75

MASCAB:RSBSS:16 C. Reinbrecht et al. "Side channel attack on NoC-based MPSoCs are practical: NoC Prime+Probe attack". In: *Integrated Circuits and Systems Design (SBCCI)*. 2016, pp. 1–6

MASCAB:RTSS:09 T. Ristenpart et al. "Hey, You, Get Off of My Cloud! Exploring Information Leakage in Third-Party Compute Clouds". In: *Computer and Communications Security (CCS)*. 2009, pp. 199–212

MASCAB:SadJam:16 M. Sadiquea and D. Jamesa. "A Novel Approach to Prevent Cache-based Side-Channel Attack in the Cloud". In: *Procedia Technology* 25 (2016), pp. 232–239

MASCAB:SavYil:15 E. Savaş and C. Yilmaz. "A Generic Method for the Analysis of a Class of Cache Attacks: A Case Study for AES". In: *The Computer Journal* 58.10 (2015), pp. 2716–2737

MASCAB:SBYLTRM:13 D. Stefan et al. "Eliminating Cache-Based Timing Attacks with Instruction-Based Scheduling". In: *European Symposium on Research in Computer Security (ESORICS)*. LNCS 8134. Springer-Verlag, 2013, pp. 718–735

MASCAB:SCNS:16 S. Shinde et al. "Preventing Page Faults from Telling Your Secrets". In: *ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. 2016, pp. 317–328

MASCAB:SFKD:14 V. Saraswat et al. "Remote Cache-timing Attacks Against AES". In: *Cryptography and Security in Computing Systems (CS2)*. 2014, pp. 45–48

MASCAB:SGLS:77 M. Schaefer et al. "Program confinement in KVM/370". In: *Proceedings of the ACM Annual Conference*. 1977, pp. 404–410

MASCAB:SLKP:17 M.-W. Shih et al. "T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs". In: *Network and Distributed System Security Symposium (NDSS)*. 2017

MASCAB:SprGer:14 R. Spreitzer and B. Gérard. "Towards More Practical Time-Driven Cache Attacks". In: *Workshop on Information Security Theory and Practice (WISTP)*. LNCS 8501. Springer-Verlag, 2014, pp. 24–39

MASCAB:SprPlo:13:a R. Spreitzer and T. Plos. "Cache-Access Pattern Attack on Disaligned AES T-Tables". In: *Constructive Side-Channel Analysis and Secure Design (COSADE)*. LNCS 7864. Springer-Verlag, 2013, pp. 200–214

MASCAB:SprPlo:13:b R. Spreitzer and T. Plos. "On the Applicability of Time-Driven Cache Attacks on Mobile Devices". In: *Network and System Security (NSS)*. LNCS 7873. Springer-Verlag, 2013, pp. 656–662

MASCAB:SSCZ:11 J. Shi et al. "Limiting Cache-based Side-channel in Multi-tenant Cloud Using Dynamic Page Coloring". In: *Dependable Systems and Networks Workshop (DSNW)*. 2011, pp. 194–199

MASCAB:TANA:07 K. Tiri et al. "An Analytical Model for Time-Driven Cache Attacks". In: *Fast Software Encryption (FSE)*. LNCS 4593. Springer-Verlag, 2007, pp. 399–413

MASCAB:TanWeiGuo:14 Y. Tan, J. Wei, and W. Guo. "The Micro-architectural Support Countermeasures against the Branch Prediction Analysis Attack". In: *Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2014, pp. 276–283

MASCAB:TFAF:13 J. Takahashi et al. "Highly Accurate Key Extraction Method for Access-Driven Cache Attacks Using Correlation Coefficient". In: *Australasian Conference on Information Security and Privacy (ACISP)*. LNCS 7959. Springer-Verlag, 2013, pp. 286–301

MASCAB:TLWCS:09 M. Tiwari et al. "Execution Leases: A Hardware-supported Mechanism for Enforcing Strong Non-interference". In: *ACM/IEEE International Symposium on Microarchitecture (MICRO)*. 2009, pp. 493–504

MASCAB:Trostle:98 J.T. Trostle. "Timing Attacks Against Trusted Path". In: *IEEE Symposium on Security & Privacy (S&P)*. 1998, pp. 125–134

MASCAB:TTSKM:06 Y. Tsunoo et al. "Improving cache attacks by considering cipher structure". In: *Journal of Information Security* 5.3 (2006), pp. 166–176

MASCAB:UzeMil:09 V. Uzelac and A. Milenkovic. "Experiment flows and microbenchmarks for reverse engineering of branch predictor structures". In: *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2009, pp. 207–217

MASCAB:V:76 T. van Vleck. "Timing Channels". 1976. URL: `http://www.multicians.org/timing-chn.html`

MASCAB:Valamehr:13 J.K. Valamehr. "Novel Methods of Augmenting High Performance Processors with Security Hardware". PhD thesis. University of California, Santa Barbara, 2013

MASCAB:VarRisSwi:14 V. Varadarajan, T. Ristenpart, and M. Swift. "Scheduler-based Defenses against Cross-VM Side-channels". In: *USENIX Security Symposium*. 2014, pp. 687–702

MASCAB:VZRS:15 V. Varadarajan et al. "A Placement Vulnerability Study in Multi-tenant Public Clouds". In: *USENIX Security Symposium*. 2015, pp. 913–928

MASCAB:WanFerSuh:14 Y. Wang, A. Ferraiuolo, and G.E. Suh. "Timing Channel Protection for a Shared Memory Controller". In: *High-Performance Computer Architecture (HPCA)*. 2014, pp. 225–236

MASCAB:WanLee:07 Z. Wang and R.B. Lee. "New Cache Designs for Thwarting Software Cache-based Side Channel Attacks". In: *International Symposium on Computer Architecture (ISCA)*. 2007, pp. 494–505

MASCAB:WanLee:08 Z. Wang and R.B. Lee. "A Novel Cache Architecture with Enhanced Performance and Security". In: *ACM/IEEE International Symposium on Microarchitecture (MICRO)*. 2008, pp. 83–93

MASCAB:WanSuh:12 Y. Wang and G.E. Suh. "Efficient Timing Channel Protection for On-Chip Networks". In: *IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. 2012, pp. 142–151

MASCAB:WeiHeiStu:12 M. Weiß, B. Heinz, and F. Stumpf. "A Cache Timing Attack on AES in Virtualization Environments". In: *Financial Cryptography and Data Security (FC)*. LNCS 7397. Springer-Verlag, 2012, pp. 314–328

MASCAB:WFZMS:16 Y. Wang et al. "SecDCP: Secure Dynamic Cache Partitioning for Efficient Timing Channel Protection". In: *Proceedings of the 53rd Annual Design Automation Conference*. 2016, 74:1–74:6

MASCAB:WGOHKCS:14 H.M.G. Wassel et al. "Networks on Chip with Provable Security Properties". In: *IEEE Micro* 34.3 (2014), pp. 57–68

MASCAB:Wray:91 J.C. Wray. "An analysis of covert timing channels". In: *IEEE Symposium on Security & Privacy (S&P)*. 1991, pp. 2–7

MASCAB:WuXuWan:12 Z. Wu, Z. Xu, and H. Wang. "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud". In: *USENIX Security Symposium*. 2012, pp. 159–173

MASCAB:WWAS:14 M. Weiß et al. "On Cache Timing Attacks Considering Multi-core Aspects in Virtualized Embedded Systems". In: *International Conference on Trusted Systems (INTRUST)*. LNCS 9473. Springer-Verlag, 2014, pp. 151–167

MASCAB:WZJWGF:15 W. Wu et al. "Warding off timing attacks in Deterland". In: *CoRR* abs/1504.07070 (2015). URL: `http://arxiv.org/abs/1504.07070`

MASCAB:XBJJHS:11 Y. Xu et al. "An Exploration of L2 Cache Covert Channels in Virtualized Environments". In: *ACM Cloud Computing Security Workshop (CCSW)*. 2011, pp. 29–40

MASCAB:XTDYZ:08 Z. Xinjie et al. "Robust First Two Rounds Access Driven Cache Timing Attack on AES". In: *Computer Science and Software Engineering*. 2008, pp. 785–788

MASCAB:XuCuiPei:15 Y. Xu, W. Cui, and M. Peinado. "Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems". In: *IEEE Symposium on Security and Privacy (S&P)*. 2015, pp. 640–656

MASCAB:XuWanWu:15 Z. Xu, H. Wang, and Z. Wu. "A Measurement Study on Co-residence Threat Inside the Cloud". In: *USENIX Security Symposium*. 2015, pp. 929–944

MASCAB:XXHW:12 J. Xiao et al. "A covert channel construction in a virtualized environment". In: *ACM Conference on Computer and Communications Security (CCS)*. 2012, pp. 1040–1042

MASCAB:YarBen:14 Y. Yarom and N. Benger. "Recovering OpenSSL ECDSA Nonces Using the FLUSH+RELOAD Cache Side-channel Attack". Cryptology ePrint Archive, Report 2014/140. 2014. URL: `http://eprint.iacr.org/2014/140`

MASCAB:YGLLH:15 Y. Yarom et al. "Mapping the Intel Last-Level Cache". Cryptology ePrint Archive, Report 2015/905. 2015. URL: `http://eprint.iacr.org/2015/905`

MASCAB:ZDFJ:16 L. Zhang et al. "Statistical Analysis for Access-Driven Cache Attacks Against AES". Cryptology ePrint Archive, Report 2016/970. 2016. URL: `http://eprint.iacr.org/2016/970`

MASCAB:ZhaWan:10 X.-J. Zhao and T. Wang. "Improved Cache Trace Attack on AES and CLEFIA by Considering Cache Miss and S-box Misalignment". Cryptology ePrint Archive, Report 2010/056. 2010. URL: `http://eprint.iacr.org/2010/056`

MASCAB:ZhaZhaLee:16:a T. Zhang, Y. Zhang, and R.B. Lee. "CloudRadar: A Real-Time Side-Channel Attack Detection System in Clouds". In: *Research in Attacks, Intrusions, and Defenses (RAID)*. LNCS 9854. Springer-Verlag, 2016, pp. 118–140

MASCAB:ZhaZhaLee:16:b T. Zhang, Y. Zhang, and R.B. Lee. "Memory DoS Attacks in Multi-tenant Clouds: Severity and Mitigation". In: *CoRR* abs/1603.03404 (2016)

MASCAB:ZhuZhaPan:04 X. Zhuang, T. Zhang, and S. Pande. "HIDE: An Infrastructure for Efficiently Protecting Information Leakage on the Address Bus". In: *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 2004, pp. 72–84

MASCAB:ZJOR:11 Y. Zhang et al. "HomeAlone: Co-residency Detection in the Cloud via Side-Channel Analysis". In: *IEEE Symposium on Security & Privacy (S&P)*. 2011, pp. 313–328

MASCAB:ZJRR:12 Y. Zhang et al. "Cross-VM Side Channels and Their Use to Extract Private Keys". In: *ACM Conference on Computer and Communications Security (CCS)*. 2012, pp. 305–316

MASCAB:ZJRR:14 Y. Zhang et al. "Cross-Tenant Side-Channel Attacks in PaaS Clouds". In: *ACM Conference on Computer and Communications Security (CCS)*. 2014, pp. 990–1003

MASCAB:ZMHS:16 A. Zankl et al. "Towards Efficient Evaluation of a Time-Driven Cache Attack on Modern Processors". In: *European Symposium on Research in Computer Security (ESORICS)*. LNCS 9879. Springer-Verlag, 2016, pp. 3–19

MASCAB:ZWLZZ:15 P. Zhou et al. "Analysis on the parameter selection method for FLUSH+RELOAD based cache timing attack on RSA". In: *China Communications* 12 (6 2015), pp. 33–45

MASCAB:ZZLP:04 X. Zhuang, T. Zhang H.-H.S Lee, and S. Pande. "Hardware Assisted Control Flow Obfuscation for Embedded Processors". In: *Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*. 2004, pp. 292–302