

Microarchitectural Security

Daniel Gruss

February 20, 2019

Graz University of Technology







Americoin, Americoin God shed his blocks on thee!

Americoin, Americain, Got shed his blocks on thee



ABOUT AGENDA TICKETS VENUE SPONSORS CONTACT US

Paris 9-10 Sep 2017





























Operating System

Untrusted part
Create Enclave

Operating System



Operating System



Operating System



Operating System



Operating System



Operating System



Operating System



Operating System







Protection from Side-Channel Attacks

Protection from Side-Channel Attacks

Intel SGX does not provide explicit protection from side-channel attacks.

Protection from Side-Channel Attacks

Intel SGX does not provide explicit protection from side-channel attacks. It is the enclave developer's responsibility to address side-channel attack concerns.

CAN'T BREAK YOUR SIDE-CHANNEL PROTECTIONS

IF YOU DON'T HAVE ANY

imgflip.com





- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX



- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

Teechain

 $\left[\ldots\right]$ We assume the TEE guarantees to hold





- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

Teechain

[...] We assume the TEE guarantees to hold and do not consider side-channel attacks [5, 35, 46] on the TEE.





- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

Teechain

[...] We assume the TEE guarantees to hold and do not consider side-channel attacks [5, 35, 46] on the TEE. Such attacks and their mitigations [36, 43] are outside the scope of this work. [...]

Raw Prime+Probe trace...¹



¹Michael Schwarz et al. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.

Attacking a weak RSA implementation inside SGX

...processed with a simple moving average...¹



¹Michael Schwarz et al. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.

Attacking a weak RSA implementation inside SGX

...allows to clearly see the bits of the exponent¹



¹Michael Schwarz et al. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.










Power consumption







- Power consumption
- Electro-magnetic radiation





- Power consumption
- Electro-magnetic radiation
- Temperature





- Power consumption
- Electro-magnetic radiation
- Temperature
- Photonic emission





- Power consumption
- Electro-magnetic radiation
- Temperature
- Photonic emission
- Acoustic emissions



- Power consumption
- Electro-magnetic radiation
- Temperature
- Photonic emission
- Acoustic emissions
- \rightarrow Physical access usually relevant, but code execution on device usually not relevant























2004



ŝ

2011



ŝ

2011







2013



































- threat model
- temporal component
- observer effect (destructive measurements)
- spatial component







• Usually no physical access



- Usually no physical access
- Local code



- Usually no physical access
- Local code
- Co-located code



- Usually no physical access
- Local code
- Co-located code
- Different meanings of "remote"



- Usually no physical access
- Local code
- Co-located code
- Different meanings of "remote"
 - 1. Attacker controls code in browser sandbox (e.g., [Ore+15; GMM16])



- Usually no physical access
- Local code
- Co-located code
- Different meanings of "remote"
 - 1. Attacker controls code in browser sandbox (e.g., [Ore+15; GMM16])
 - 2. Attacker cannot control any code on the system






Just a few examples:

 Remote timing attacks on crypto ([Ber04; BB05] and many more)



Just a few examples:

- Remote timing attacks on crypto ([Ber04; BB05] and many more)
- ThrowHammer and NetHammer



Just a few examples:

- Remote timing attacks on crypto ([Ber04; BB05] and many more)
- ThrowHammer and NetHammer
- NetSpectre



TIMING IS EVERYTHING

CPU Cache



_

printf("%d", i); printf("%d", i);

Daniel Gruss — Graz University of Technology















www.tugraz.at

CPU Cache



















Daniel Gruss — Graz University of Technology





Daniel Gruss — Graz University of Technology



• theoretical maximum accuracy of $5.4 \cdot 10^{-44}$ s



- theoretical maximum accuracy of $5.4 \cdot 10^{-44}$ s
- feasible today: $850 \cdot 10^{-21}$ s



- theoretical maximum accuracy of $5.4 \cdot 10^{-44}$ s
- feasible today: $850 \cdot 10^{-21}$ s

Microarchitectural Attacks



- theoretical maximum accuracy of $5.4 \cdot 10^{-44}$ s
- feasible today: $850 \cdot 10^{-21}$ s

Microarchitectural Attacks

• often around nanoseconds



- theoretical maximum accuracy of $5.4 \cdot 10^{-44}$ s
- feasible today: $850 \cdot 10^{-21}$ s

Microarchitectural Attacks

- often around nanoseconds
- sometimes much lower



Daniel Gruss — Graz University of Technology

• in the range of multiple GHz



• in the range of multiple GHz

Microarchitectural Attacks



• in the range of multiple GHz

Microarchitectural Attacks

• usually varying frequency (depending on the attack)

• in the range of multiple GHz

Microarchitectural Attacks

- usually varying frequency (depending on the attack)
- between a few ns (< 1 GHz) and multiple seconds (< 1 Hz) (or even worse)

Daniel Gruss — Graz University of Technology

Intint

• in the range of multiple GHz

Microarchitectural Attacks

- usually varying frequency (depending on the attack)
- between a few ns (< 1 GHz) and multiple seconds (< 1 Hz) (or even worse)
- strongly dependent on the specific attack

Daniel Gruss — Graz University of Technology

Тини

• in the range of multiple GHz

Microarchitectural Attacks

- usually varying frequency (depending on the attack)
- between a few ns (< 1 GHz) and multiple seconds (< 1 Hz) (or even worse)
- strongly dependent on the specific attack
 - device under test = measurement device

Daniel Gruss — Graz University of Technology

<u>Imm</u>

• in the range of multiple GHz

Microarchitectural Attacks

- usually varying frequency (depending on the attack)
- between a few ns (< 1 GHz) and multiple seconds (< 1 Hz) (or even worse)
- strongly dependent on the specific attack
 - device under test = measurement device
 - observer effect

Daniel Gruss — Graz University of Technology

<u>Imbud</u>



device under test = measurement device

- measuring time takes some time
- limits the resolution
- measuring cache hits/misses manipulates the cache state
- virtually all measurements are destructive








• Race condition between attacker and victim (observer effect)



- Race condition between attacker and victim (observer effect)
- Speculative execution



- Race condition between attacker and victim (observer effect)
- Speculative execution
- Prefetching



- Race condition between attacker and victim (observer effect)
- Speculative execution
- Prefetching
- ...





- Race condition between attacker and victim (observer effect)
- Speculative execution
- Prefetching
- ...
- \rightarrow Typically >99.99% precision and recall





Measuring Processor Operations

- Very short timings
- rdtsc instruction: "cycle-accurate" timestamps

```
[...]
rdtsc
function()
rdtsc
[...]
```

- Do you measure what you *think* you measure?
- *Out-of-order* execution \rightarrow what is really executed?

rdtsc	rdtsc	rdtsc
function()	[]	rdtsc
[]	rdtsc	<pre>function()</pre>
rdtsc	function()	[]



• use pseudo-serializing instruction rdtscp (recent CPUs)

- use pseudo-serializing instruction rdtscp (recent CPUs)
- and/or use serializing instructions like cpuid

- use pseudo-serializing instruction rdtscp (recent CPUs)
- and/or use serializing instructions like cpuid
- and/or use fences like mfence

- use pseudo-serializing instruction rdtscp (recent CPUs)
- and/or use serializing instructions like cpuid
- and/or use fences like mfence

Intel, How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures White Paper, December 2010.

AUGUST 22, 2018 BY BRUCE

Intel Publishes Microcode Security

Patches, No Benchmarking Or

Comparison Allowed!

UPDATE: Intel has resolved their microcode licensing issue which I complained about in this blog post. The new license text is here.



Cache Hits



www.tugraz.at

Cache Hits Cache Misses





Temporal Component





Temporal Component







• Flush+Reload had beautifully nice timings, right?



- Flush+Reload had beautifully nice timings, right?
- Well... steps of 2-4 cycles



- Flush+Reload had beautifully nice timings, right?
- Well... steps of 2-4 cycles
 - only 35-70 steps between hits and misses



- Flush+Reload had beautifully nice timings, right?
- Well... steps of 2-4 cycles
 - only 35-70 steps between hits and misses
- On some devices only 1-2 steps!



• We can build our own timer



- We can build our own timer
- Start a thread that continuously increments a global variable



- We can build our own timer
- Start a thread that continuously increments a global variable
- The global variable is our timestamp









1 timestamp = rdtsc();



1 while(1) {
2 timestamp++;
3 }



```
1 while(1) {
2 timestamp++;
3 }
```



1 mov ×tamp, %rcx
2 1: incl (%rcx)
3 jmp 1b



1 mov ×tamp, %rcx 2 1: incl (%rcx) 3 jmp 1b



- $_1$ mov ×tamp , $\ensuremath{\,\% rcx}$
- 2 1: inc %rax
- з mov %rax, (%rcx)
- 4 **jmp** 1b
CPU cycles one increment takes



- $_1~\text{mov}$ ×tamp , ~%rcx
- 2 1: inc %rax
- з mov %rax, (%rcx)
- 4 **jmp** 1b



Modern Processor Design

















• device under test = measurement device





- device under test = measurement device
- $\rightarrow\,$ software defenses are possible



- device under test = measurement device
- $\rightarrow\,$ software defenses are possible
- e.g., make sure attacker can't compute in parallel to victim



- device under test = measurement device
- $\rightarrow\,$ software defenses are possible
- e.g., make sure attacker can't compute in parallel to victim
- how would that work in the physical world?













• physical: different offsets on the chip



- physical: different offsets on the chip
- microarchitectural:



- physical: different offsets on the chip
- microarchitectural:
 - different microarchitectural elements



- physical: different offsets on the chip
- microarchitectural:
 - different microarchitectural elements
 - more significant: huge virtual adress space



- physical: different offsets on the chip
- microarchitectural:
 - different microarchitectural elements
 - more significant: huge virtual adress space
 - 2⁴⁸ different virtual memory locations



- physical: different offsets on the chip
- microarchitectural:
 - different microarchitectural elements
 - more significant: huge virtual adress space
 - 2⁴⁸ different virtual memory locations
 - the location is often (part of) the secret

12	Terminal			× Ciner	r⊈ ∓	Untitled	Document 1	Saue =	: _	+ ×
File Edit View Search Terminal Help								2335 3		
% sleep 2; ./spy 300 7f0 8050	f05140a4000-7f051417b000 r /usr/lib/x86_64-linux-g	r-xp 0x20000 08:02 26 -gnu/gedit/libgedit.so	02 20 111.50							
						I				
in econos Intelatebi 3.		CORS 14 03 2017 20 CORS 14 03 20 CORS 14 00 CORS 14 00	-44-26							
File Edit View Search Terminal Help shark% ./spy []										
onumeroaments;						PlainText 👻	Tab Widds: 2 👻	Ln 1, Col 1		INS

Cache Template Attack Demo

Cache Template²



²Daniel Gruss et al. Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches. In: USENIX Security Symposium. 2015.

Side-Channel Attacks and Fault Attacks?

Physical

• Side-channel attacks





Physical

- Side-channel attacks
- Fault attacks



Physical



- Side-channel attacks
- Fault attacks
- What about cold boot attacks?

Physical



- Side-channel attacks
- Fault attacks
- What about cold boot attacks?

Microarchitectural

Physical



- Side-channel attacks
- Fault attacks
- What about cold boot attacks?

Microarchitectural

• Side-channel attacks

Physical

- Side-channel attacks
- Fault attacks
- What about cold boot attacks?

Microarchitectural

- Side-channel attacks
- Fault attacks







Physical

- Side-channel attacks
- Fault attacks
- What about cold boot attacks?

Microarchitectural

- Side-channel attacks
- Fault attacks
- What about Meltdown/Spectre?









(volatile char) 0; array[84 * 4096] = 0;

Building Meltdown



• Flush+Reload over all pages of the array



Building Meltdown



• Flush+Reload over all pages of the array



• "Unreachable" code line was actually executed

Building Meltdown



• Flush+Reload over all pages of the array



- "Unreachable" code line was actually executed
- Exception was only thrown afterwards



• Out-of-order instructions leave microarchitectural traces



- Out-of-order instructions leave microarchitectural traces
 - We can see them for example through the cache



- Out-of-order instructions leave microarchitectural traces
 - We can see them for example through the cache
- Give such instructions a name: transient instructions


- Out-of-order instructions leave microarchitectural traces
 - We can see them for example through the cache
- Give such instructions a name: transient instructions
- We can indirectly observe the execution of transient instructions



• Add another layer of indirection to test



• Add another layer of indirection to test

• Then check whether any part of array is cached

Building Meltdown



• Flush+Reload over all pages of the array



• Index of cache hit reveals data

Building Meltdown



• Flush+Reload over all pages of the array



- Index of cache hit reveals data
- Permission check is in some cases not fast enough



e01d8130:	20	75	73	65	64	20	77	69	74	68	20	61	75	74	68	6f	used with autho
e01d8140:	72	69	7a	61	74	69	6f	6e	20	66	72	6f	6d	0a	20	53	rization from. S
e01d8150:	69	6c	69	63	6f	6e	20	47	72	61	70	68	69	63	73	2c	ilicon Graphics,
e01d8160:	20	49	6e	63	2e	20	20	48	6f	77	65	76	65	72	2c	20	Inc. However,
e01d8170:	74	68	65	20	61	75	74	68	6f	72	73	20	6d	61	6b	65	the authors make
e01d8180:	20	6e	6f	20	63	6c	61	69	6d	20	74	68	61	74	20	4d	no claim that M
e01d8190:	65	73	61	0a	20	69	73	20	69	6e	20	61	6e	79	20	77	esa. is in any w
e01d81a0:	61	79	20	61	20	63	6f	6d	70	61	74	69	62	6c	65	20	ay a compatible
e01d81b0:	72	65	70	6c	61	63	65	6d	65	6e	74	20	66	6f	72	20	replacement for
e01d81c0:	4f	70	65	6e	47	4c	20	6f	72	20	61	73	73	6f	63	69	OpenGL or associ
e01d81d0:	61	74	65	64	20	77	69	74	68	0a	20	53	69	6c	69	63	ated with. Silic
e01d81e0:	6f	6e	20	47	72	61	70	68	69	63	73	2c	20	49	6e	63	on Graphics, Inc
e01d81f0:	2e	0a	20	2e	0a	20	54	68	69	73	20	76	65	72	73	69	This versi
e01d8200:	6f	6e	20	6f	66	20	4d	65	73	61	20	70	72	6f	76	69	on of Mesa provi
e01d8210:	64	65	73	20	47	4c	58	20	61	6e	64	20	44	52	49	20	des GLX and DRI
e01d8220:	63	61	70	61	62	69	6c	69	74	69	65	73	Зa	20	69	74	<pre> capabilities: it </pre>
e01d8230:	20	69	73	20	63	61	70	61	62	6c	65	20	6f	66	0a	20	is capable of.
e01d8240:	62	6f	74	68	20	64	69	72	65	63	74	20	61	6e	64	20	both direct and
e01d8250:	69	6e	64	69	72	65	63	74	20	72	65	6e	64	65	72	69	indirect renderi
e01d8260:	6e	67	2e	20	20	46	6f	72	20	64	69	72	65	63	74	20	ng. For direct
e01d8270:	72	65	6e	64	65	72	69	6e	67	2c	20	69	74	20	63	61	rendering, it ca
e01d8280:	6e	20	75	73	65	20	44	52	49	0a	20	6d	6f	64	75	6c	n use DRI. modul
e01d8290:	65	73	20	66	72	6f	6d	20	74	68	65	20	6c	69	62	67	es from the libg



×

File Edit View Search Terminal Help						Terminal
	File	Edit	View	Search	Terminal	Help

• Basic Meltdown code leads to a crash (segfault)

- Basic Meltdown code leads to a crash (segfault)
- How to prevent the crash?

- Basic Meltdown code leads to a crash (segfault)
- How to prevent the crash?



Fault Handling



Fault Suppression



Fault Prevention • Intel TSX to suppress exceptions instead of signal handler

```
if(xbegin() == XBEGIN_STARTED) {
 array[secret * 4096] = 0;
 xend();
}
for (size_t i = 0; i < 256; i++) {</pre>
 if (flush_and_reload(array + i * 4096) == CACHE_HIT) {
   printf("%c n", i):
 }
}
```

www.tugraz.at

Meltdown with Fault Prevention

• Speculative execution to prevent exceptions

```
int speculate = rand() % 2;
((size_t)&zero * (1 - speculate));
if(!speculate) {
 char secret = *(char*) address:
 array[secret * 4096] = 0;
}
for (size_t i = 0: i < 256: i++) {</pre>
 if (flush_and_reload(array + i * 4096) == CACHE_HIT) {
   printf("%c\n", i);
 }
}
```

Foreshadow / Foreshadow-NG³



³Jo Van Bulck et al. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In: USENIX Security Symposium. 2018.



L1TF/Foreshadow Demo

Spectre-PHT (v1)





Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)





Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)





Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)





Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)





Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)



Spectre-PHT (v1)






Spectre-PHT (v1)



Spectre-PHT (v1)







Spectre-PHT (v1)









Spectre-STL (v4): Ignore sanitizing write access and use unsanitized old value instead







www.tugraz.at

























Animal* a = bird;





























Animal* a = fish;















Spectre-BTB (v2): mistrain BTB \rightarrow mispredict indirect jump/call







Spectre-BTB (v2): mistrain BTB \rightarrow mispredict indirect jump/call Spectre-RSB (v5): mistrain RSB \rightarrow mispredict return • v1.1: Speculatively write to memory locations

- v1.1: Speculatively write to memory locations
- $\rightarrow\,$ Many more gadgets than previously anticipated n

- v1.1: Speculatively write to memory locations
- $\rightarrow\,$ Many more gadgets than previously anticipated n
 - v1.2: Ignore writable bit

- v1.1: Speculatively write to memory locations
- $\rightarrow\,$ Many more gadgets than previously anticipated n
 - v1.2: Ignore writable bit
- $\rightarrow \,=\, {\sf Meltdown}{\sf -}{\sf RW}$







Shared Branch Prediction State

Classification Tree

www.tugraz.at







JOIN BENEFIT - CONTRIBUTE - DISCOVER - BLOG

Computer Architecture Today

Informing the broad computing community about current activities, advances and future directions in computer architecture.

Let's Keep it to Ourselves: Don't Disclose Vulnerabilities

by Gus Uht on Jan 31, 2019 | Tags: Opinion, Security



CONTRIBUTE

Editor: Alvin R. Lebeck Associate Editor: Vijay Janapa Reddi

Contribute to Computer Architecture Today Mitigations

www.tugraz.at

Table 1: Spectre-type defenses and what they mitigate.



Performance Costs

Table 2: Reported performance impacts of countermeasures

Impact Defense	Performance Loss	Benchmark
InvisiSpec	22%	SPEC
SafeSpec	3% (improvement)	SPEC2017 on MARSSx86
DAWG	2-12%, 1-15%	PARSEC, GAPBS
RSB Stuffing	no reports	
Retpoline	5-10%	real-world workload servers
Site Isolation	only memory overhead	
SLH	36.4%, 29%	Google microbenchmark suite
YSNB	60%	Phoenix
IBRS	20-30%	two sysbench 1.0.11 benchmarks
STIPB	30- 50%	Rodinia OpenMP, DaCapo
IBPB	no individual reports	
Serialization	62%, 74.8%	Google microbenchmark suite
SSBD/SSBB	2-8%	SYSmark®2014 SE & SPEC integer
KAISER/KPTI	0-2.6%	system call rates
L1TF mitigations	-3-31%	various SPEC

Reverse-Engineering the Processors





• new class of software-based attacks


- new class of software-based attacks
- many problems to solve around microarchitectural attacks and especially transient execution attacks



- new class of software-based attacks
- many problems to solve around microarchitectural attacks and especially transient execution attacks
- dedicate more time into identifying problems and not solely in mitigating known problems



Microarchitectural Security

Daniel Gruss

February 20, 2019

Graz University of Technology

Daniel Gruss — Graz University of Technology

References

- Michael Backes et al. Acoustic Side-Channel Attacks on Printers. In: USENIX Security. 2010.
- David Brumley et al. Remote timing attacks are practical. In: Computer Networks 48.5 (2005), pp. 701–716.
 - Daniel J. Bernstein. Cache-Timing Attacks on AES. 2004. URL: http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.
 - Elad Carmon et al. Photonic Side Channel Attacks Against RSA. In: HOST'17. 2017.
 - Daniel Gruss et al. Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript. In: DIMVA. 2016.

Daniel Gruss et al. Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches. In: USENIX Security Symposium. 2015.

- J. Alex Halderman et al. Lest we remember: cold-boot attacks on encryption keys. In: Communications of the ACM (May 2009).
- Michael Hutter et al. The temperature side channel and heating fault attacks. In: International Conference on Smart Card Research and Advanced Applications. Springer. 2013, pp. 219–235.
- Paul Kocher et al. Differential power analysis. In: Annual International Cryptology Conference. Springer. 1999, pp. 388–397.
- Paul Kocher et al. Spectre Attacks: Exploiting Speculative Execution. In: S&P. 2019.
- Emilia Käsper et al. Faster and Timing-Attack Resistant AES-GCM. In: Cryptographic Hardware and Embedded Systems (CHES). 2009, pp. 1–17.

Vladimir Kiriansky et al. Speculative Buffer Overflows: Attacks and Defenses. In: arXiv:1807.03757 (2018).

- Moritz Lipp et al. ARMageddon: Cache Attacks on Mobile Devices. In: USENIX Security Symposium. 2016.
- Moritz Lipp et al. Nethammer: Inducing Rowhammer Faults through Network Requests. In: arXiv:1711.08002 (2017).
- Moritz Lipp et al. Meltdown: Reading Kernel Memory from User Space. In: USENIX Security Symposium. 2018.
- Stefan Mangard et al. Power analysis attacks: Revealing the secrets of smart cards. Vol. 31. Springer Science & Business Media, 2008.
 - Yossef Oren et al. The Spy in the Sandbox: Practical Cache Attacks in JavaScript and their Implications. In: CCS. 2015.
 - Josyula R Rao et al. EMpowering Side-Channel Attacks. In: IACR Cryptology ePrint Archive 2001 (2001), p. 37.

Alexander Schlösser et al. Simple Photonic Emission Analysis of AES. In: CHES'12. 2012.

- Michael Schwarz et al. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.
- Michael Schwarz et al. Automated Detection, Exploitation, and Elimination of Double-Fetch Bugs using Modern CPU Features. In: AsiaCCS (2018).
 - Michael Schwarz et al. NetSpectre: Read Arbitrary Memory over Network. In: arXiv:1807.10535 (2018).
 - Andrei Tatar et al. Throwhammer: Rowhammer Attacks over the Network and Defenses. In: USENIX ATC. 2018.
 - Jo Van Bulck et al. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In: USENIX Security Symposium. 2018.
 - Ofir Weisse et al. Foreshadow-NG: Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution. In: Technical report (2018).