Medusa:
Microarchitectural Data Leakage via Automated Attack Synthesis

- **Daniel Moghimi**
- Moritz Lipp
- Berk Sunar
- Michael Schwarz

29TH USENIX SECURITY SYMPOSIUM

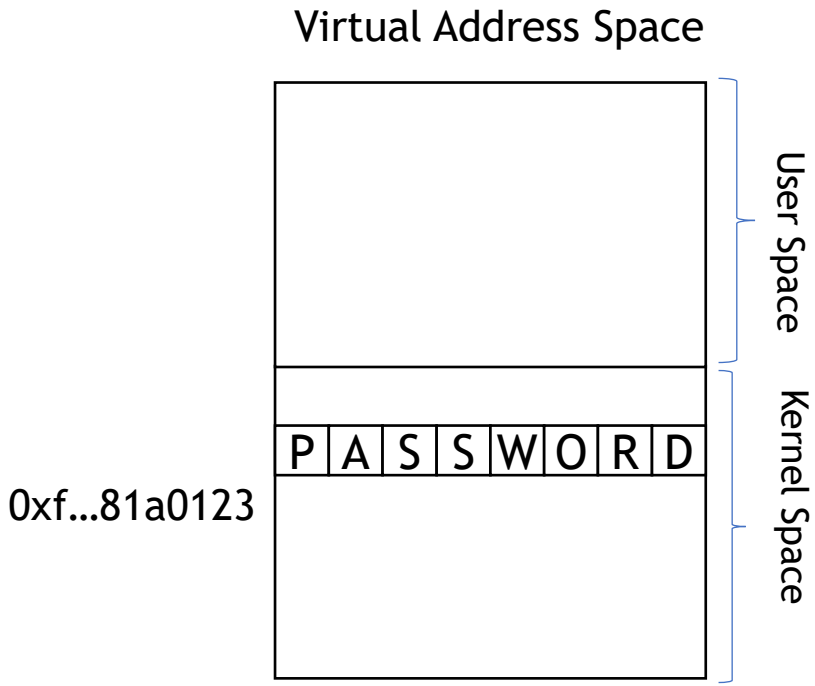
AUGUST 12–14, 2020

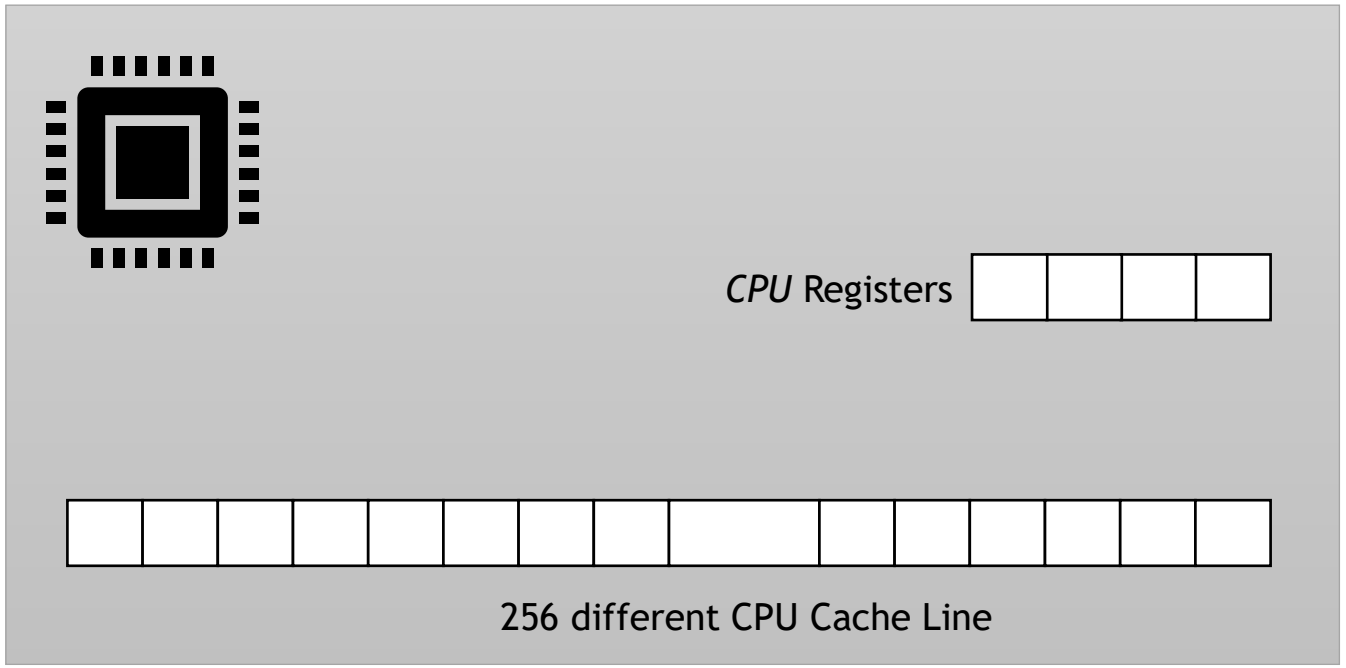
2018: Meltdown Attack?


```
char secret = *(char *) 0xffffffff81a0123;
printf("%c\n", secret);
```

2018: Meltdown Attack?

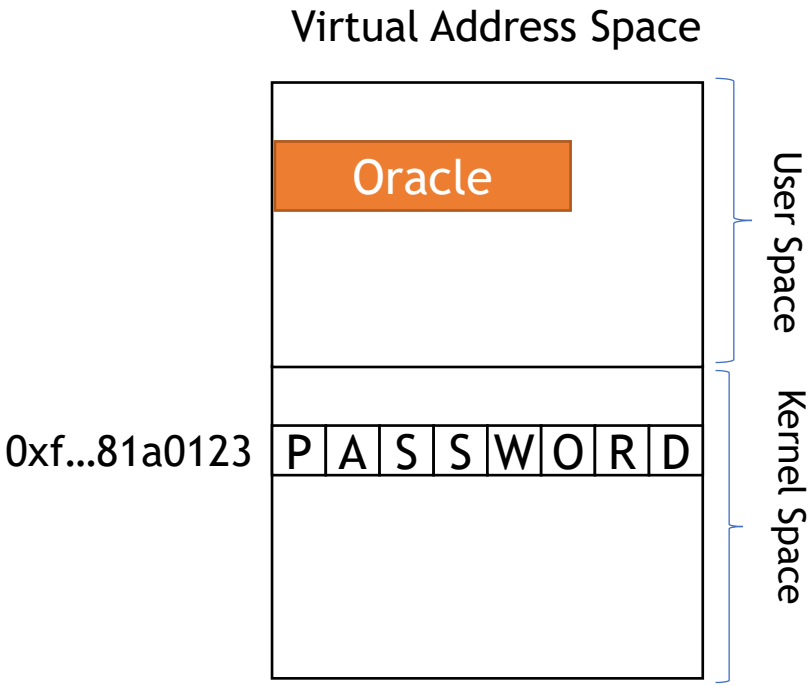
```
char secret = *(char *) 0xffffffff81a0123;
```

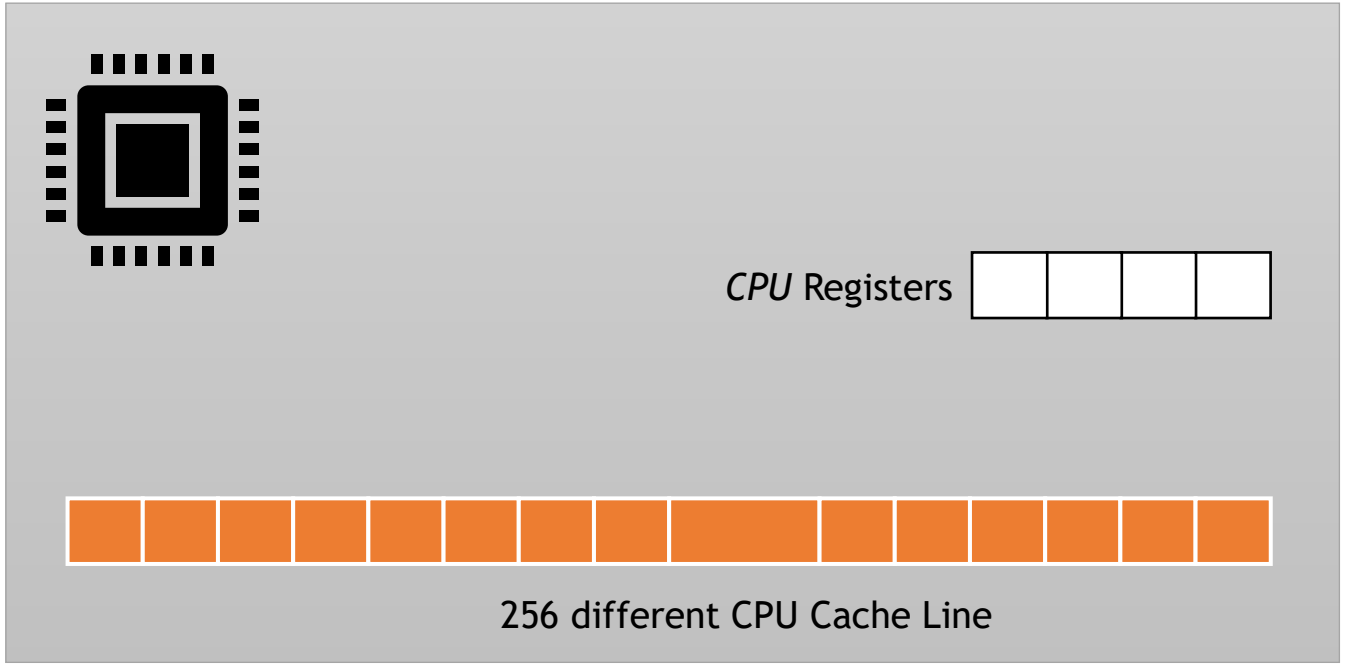




2018: Meltdown Attack?

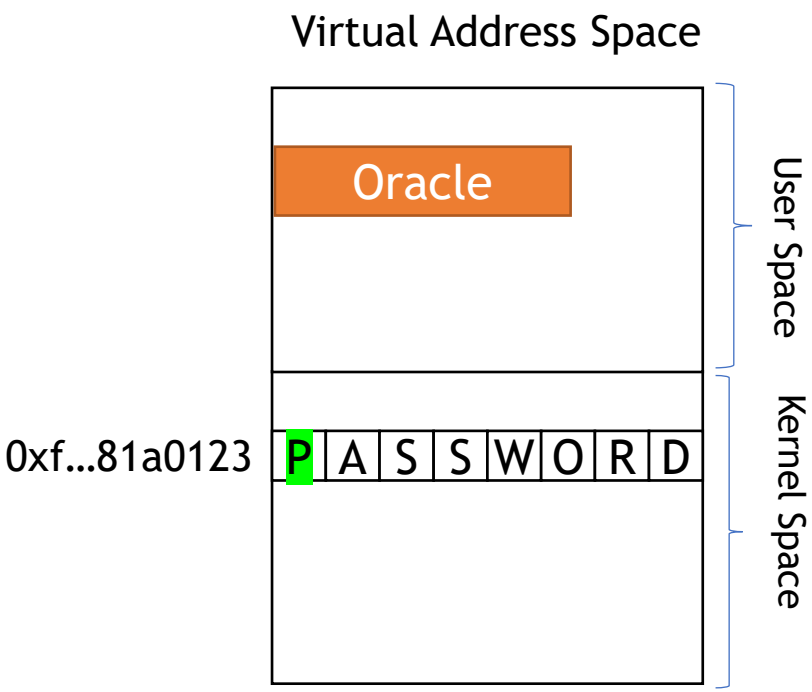
```
char secret = *(char *) 0xffffffff81a0123;
```

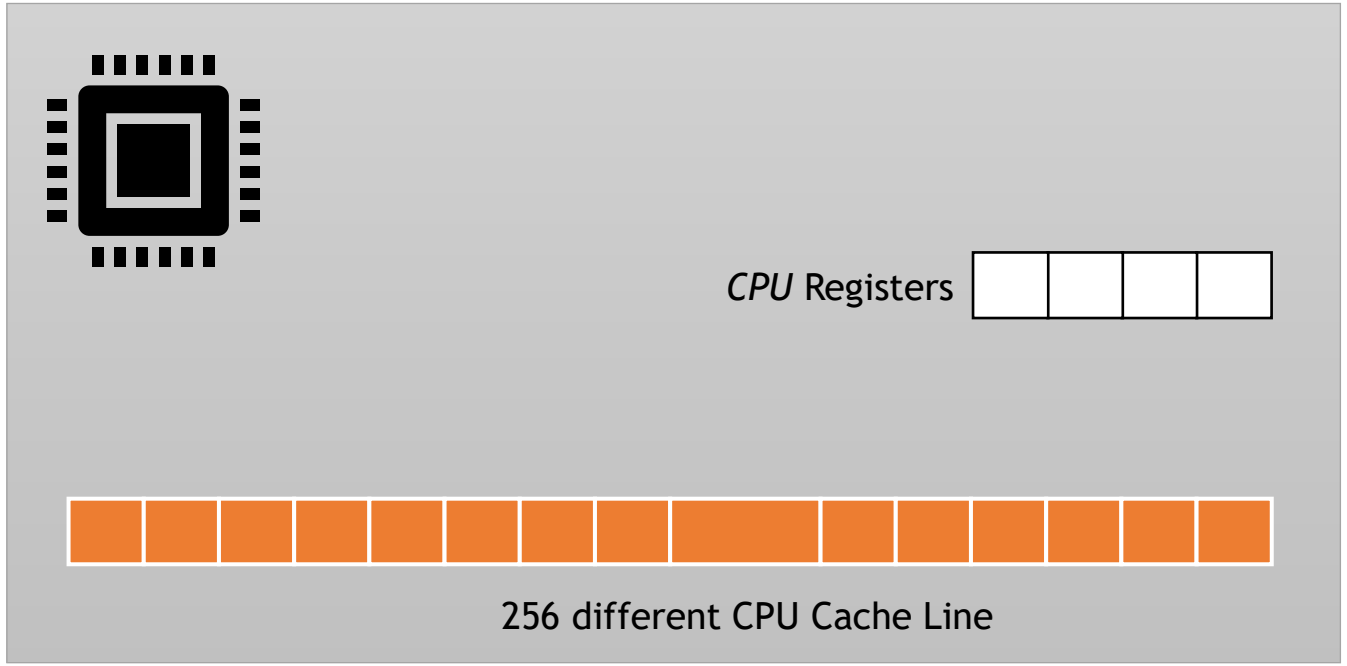




2018: Meltdown Attack? (Step 1)

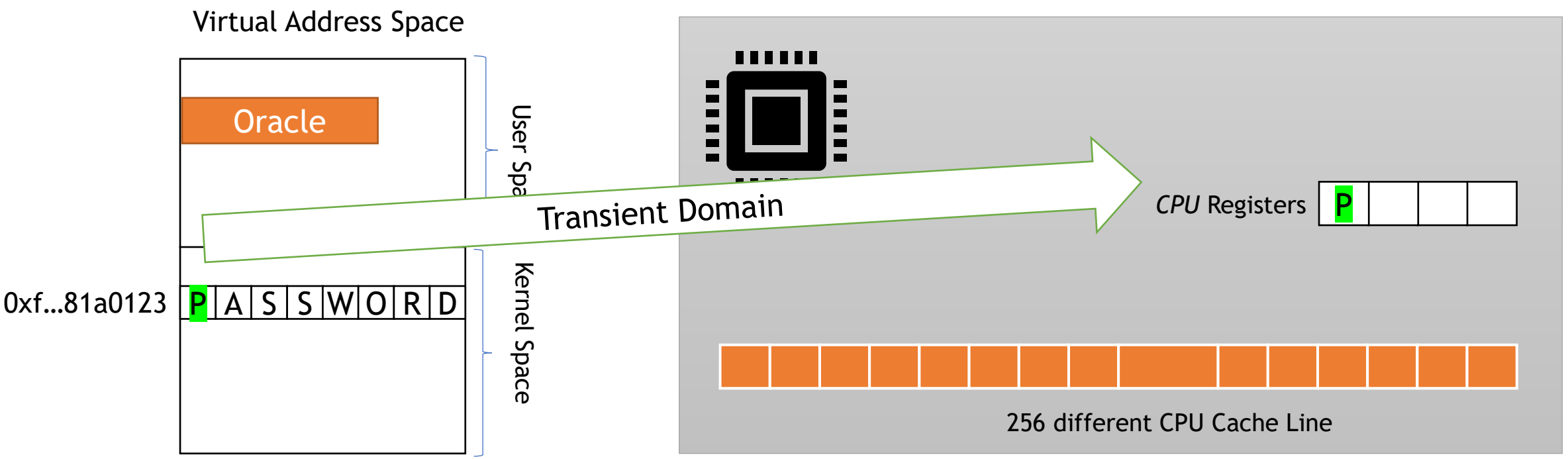
```
char secret = *(char *) 0xffffffff81a0123;
```





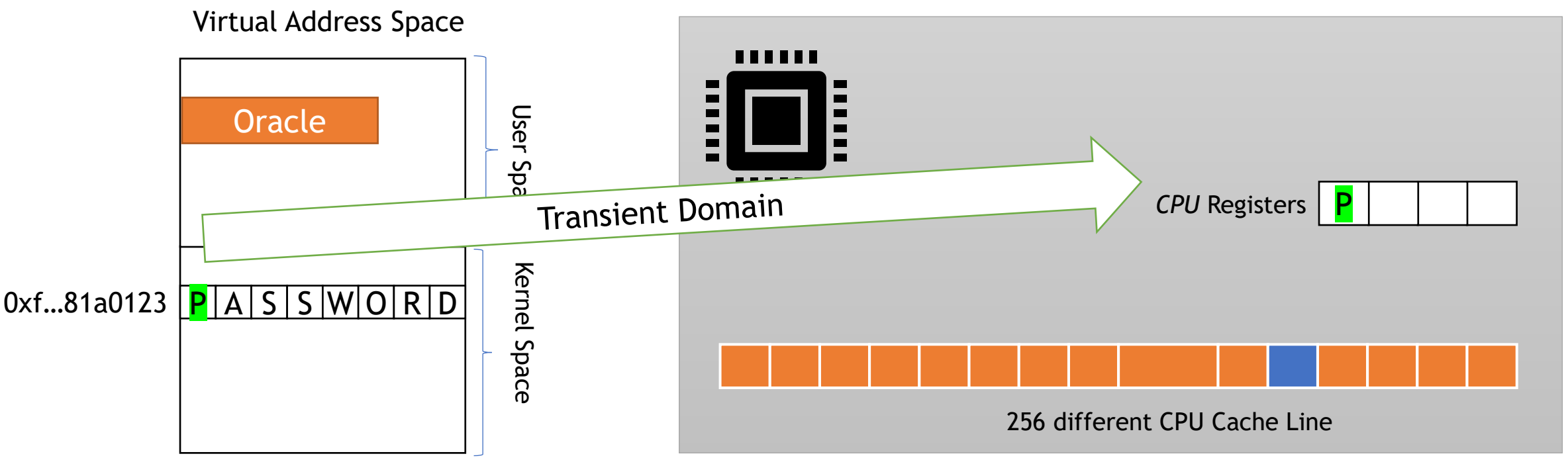
2018: Meltdown Attack? (Step 1)

```
char secret = *(char *) 0xffffffff81a0123;
```



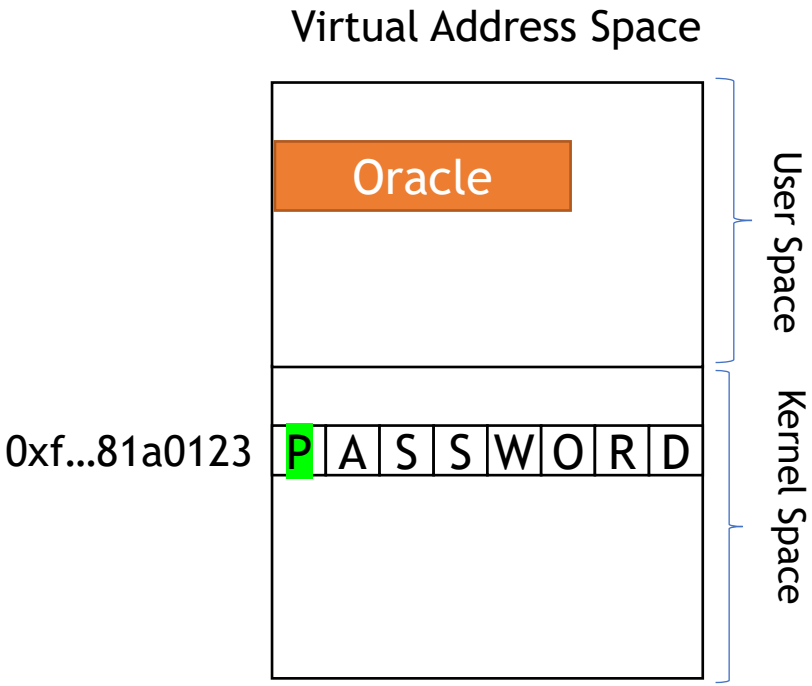
2018: Meltdown Attack? (Step 2)

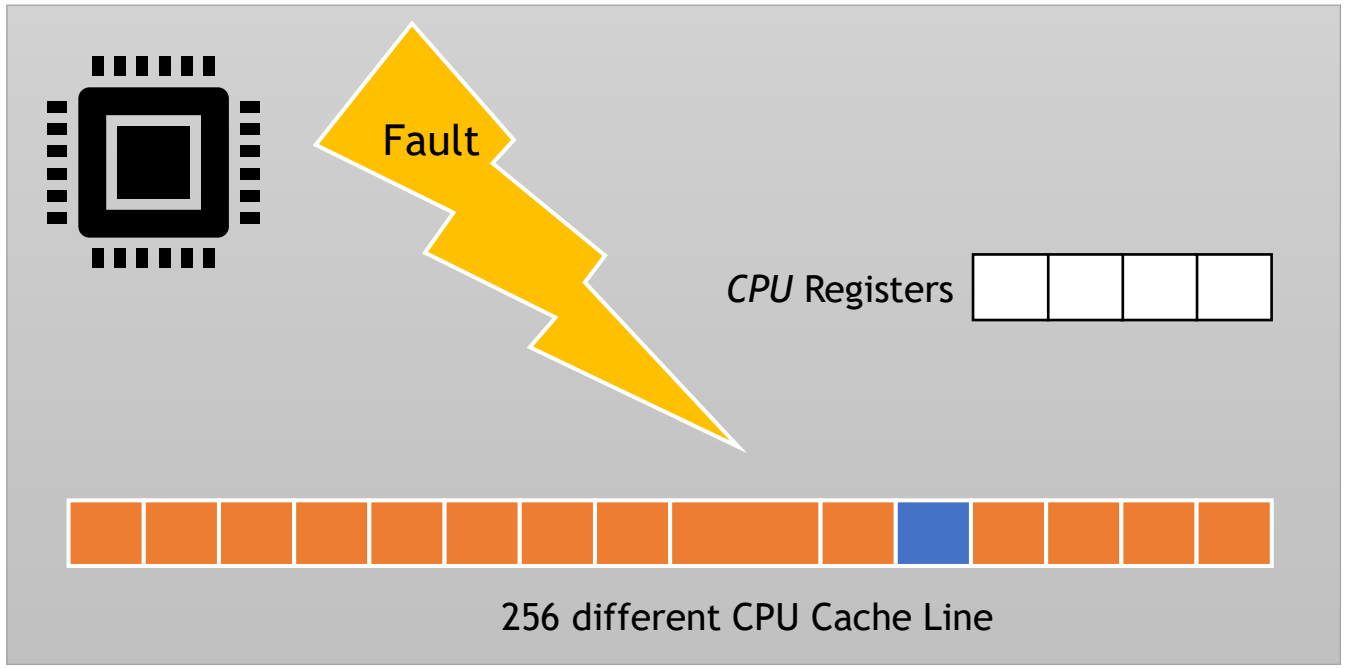

```
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];
```



2018: Meltdown Attack? (Step 2)

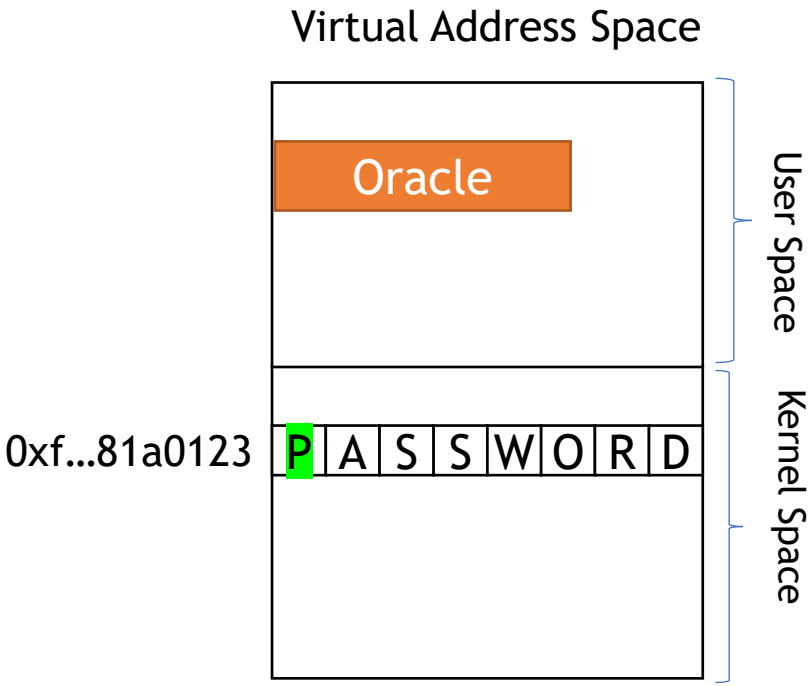

```
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];
```

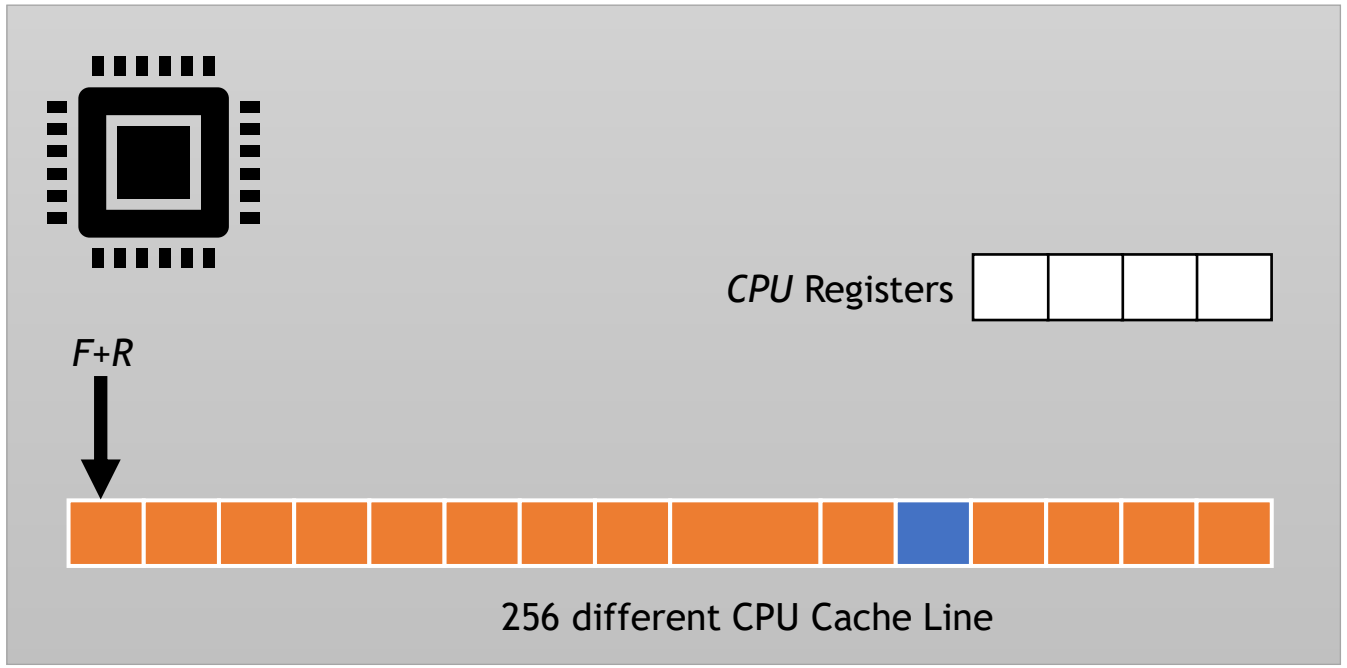




2018: Meltdown Attack? (Step 3)

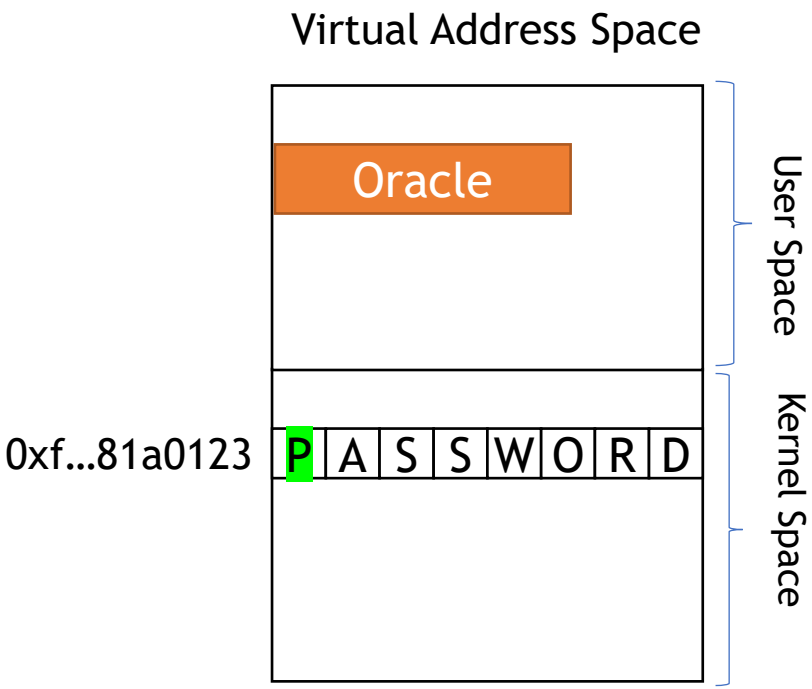
```
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];
```

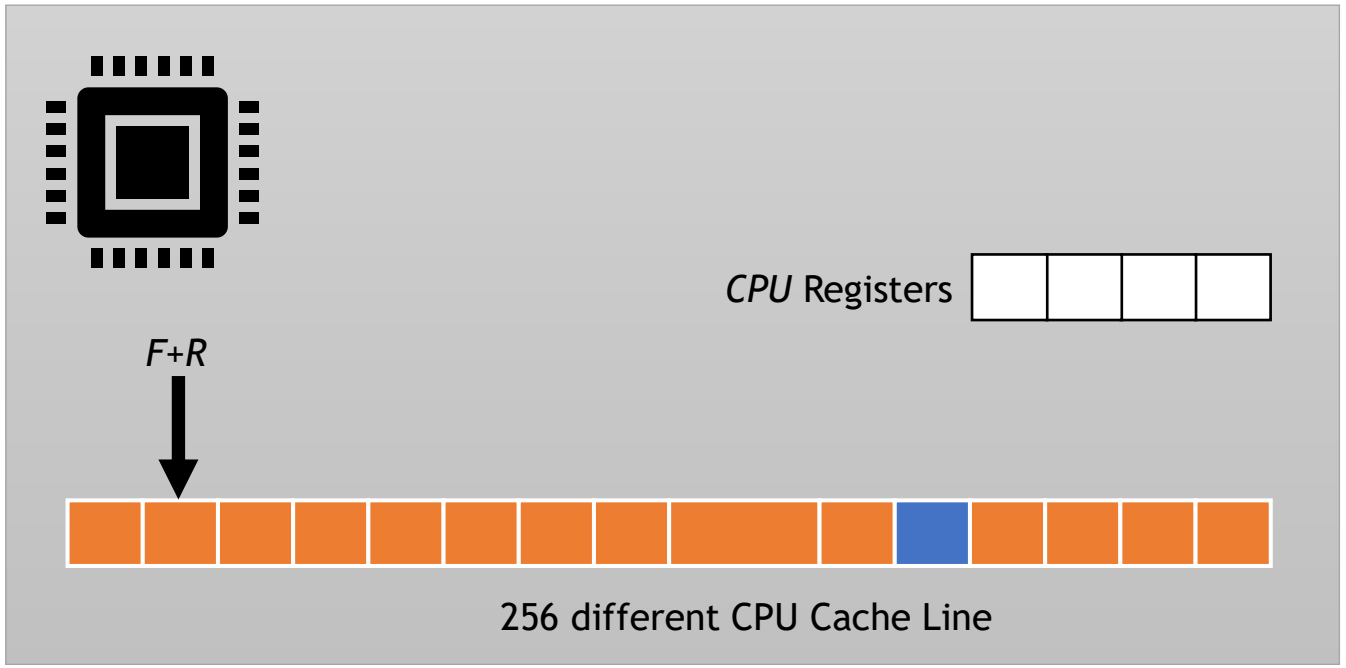




2018: Meltdown Attack? (Step 3)

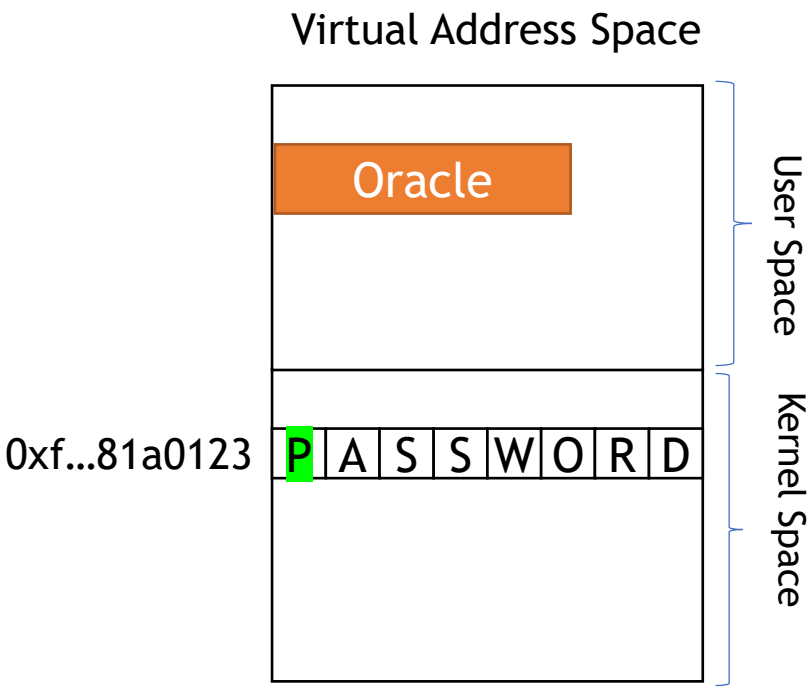
```
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];
```

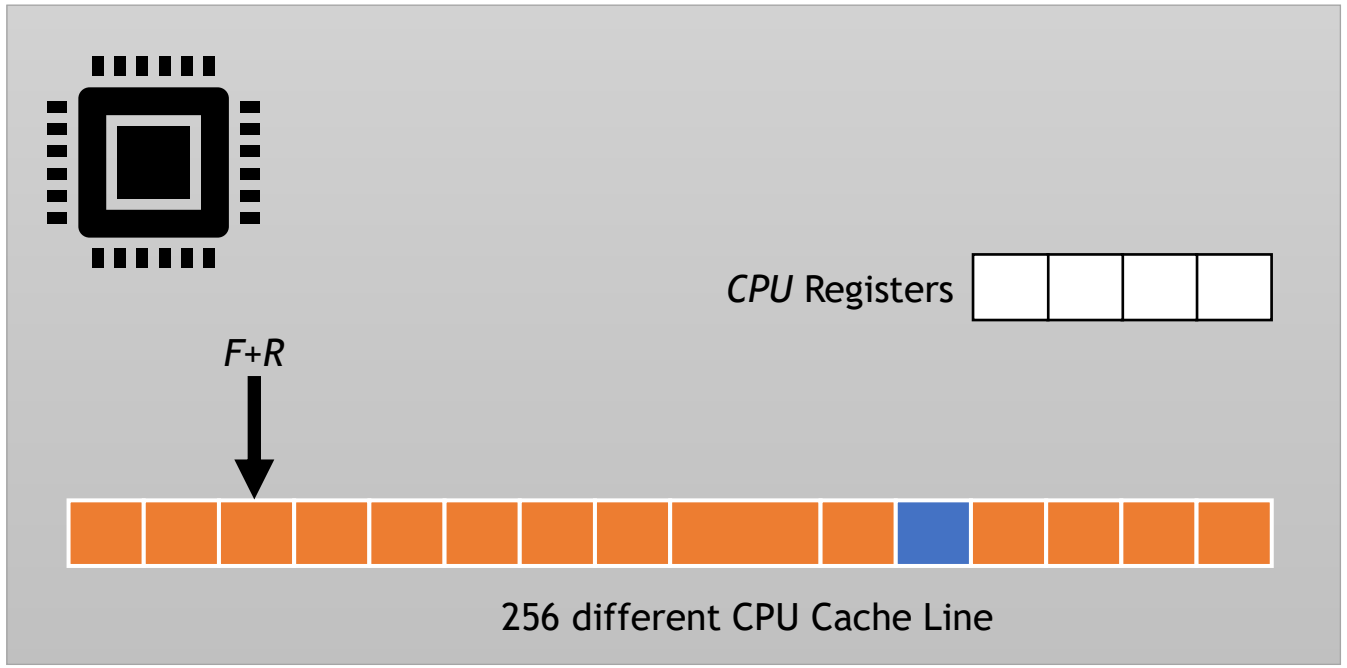




2018: Meltdown Attack? (Step 3)

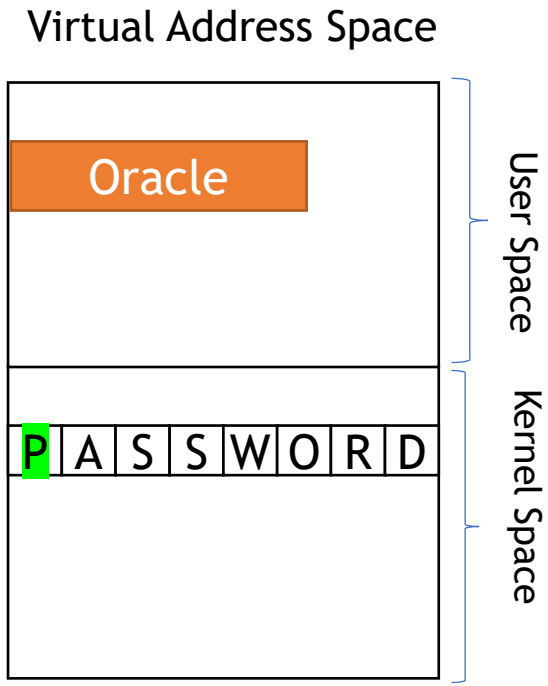
```
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];
```

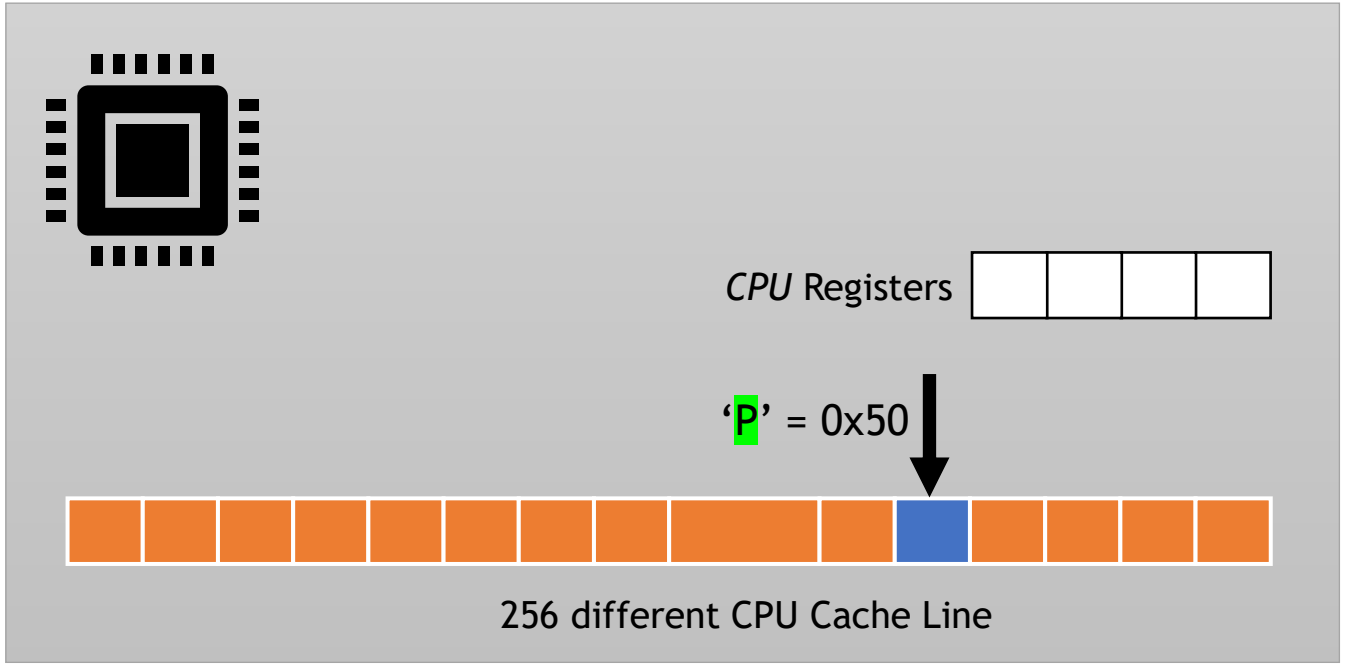




2018: Meltdown Attack? (Step 3)

```
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];
```





Microarchitecture Data Sampling (MDS)

- Meltdown is fixed but you can still leak on the fix hardware.

```
char secret = *(char *) 0xffffffffffffa0123;
```

whatever

- Which part of the CPU leak the data?!

- **Why does it leak?**

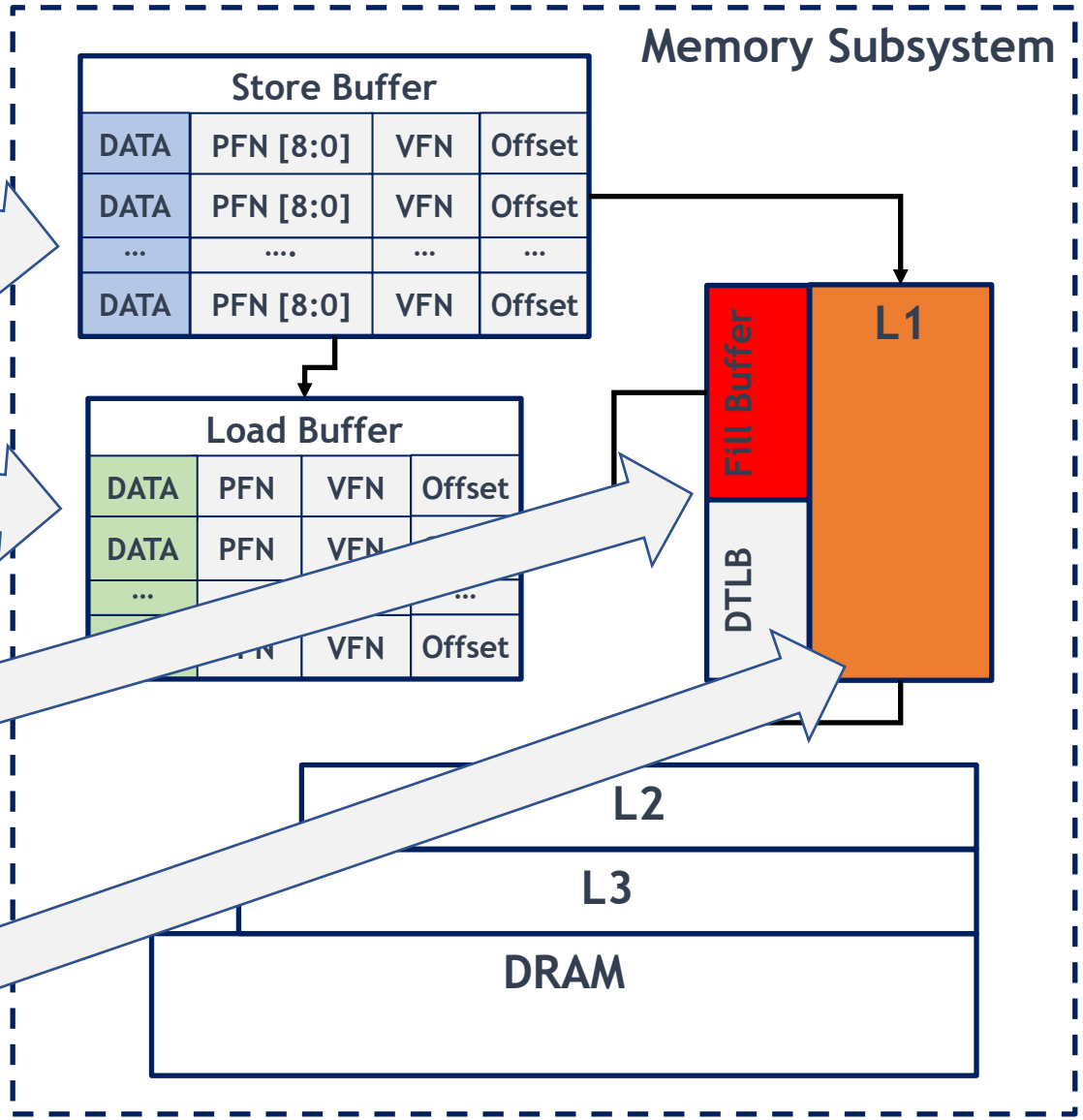
CPU Memory Subsystem – Leaky Buffers

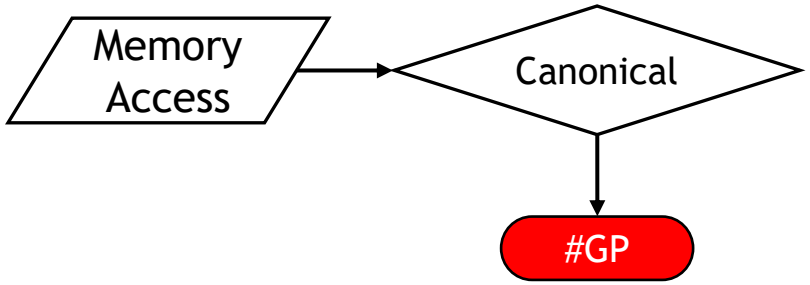
MSBDS

MLPDS

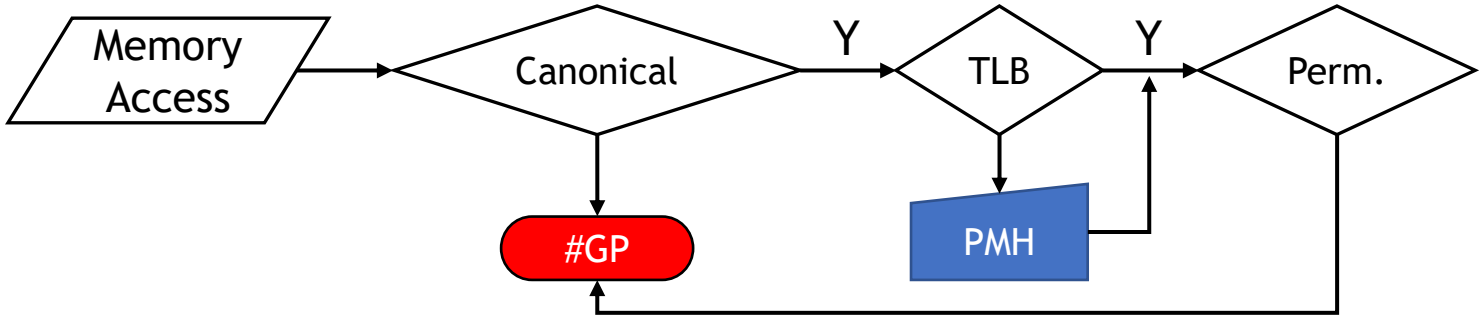
MFBDS

L1TF





Virtual Address

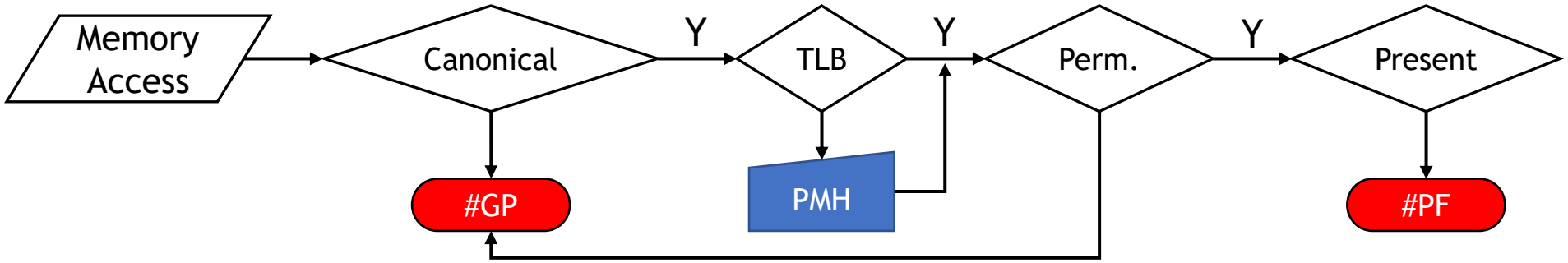


Virtual Address

VFN	Offset

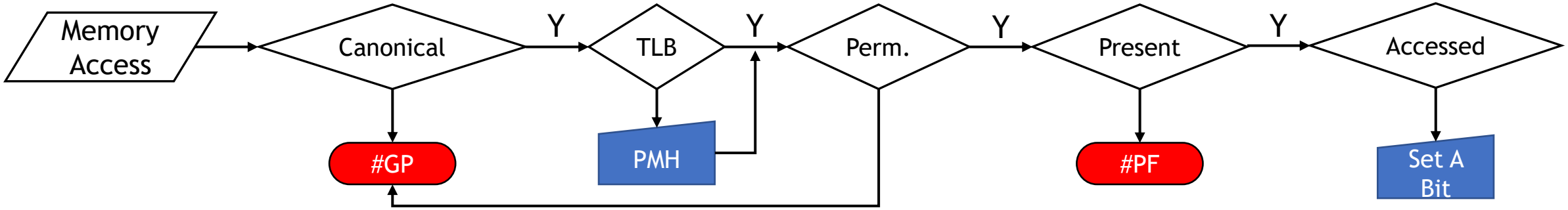
PTE

P	RW	US	…	A	…	Physical Page Number	…



Virtual Address

PTE

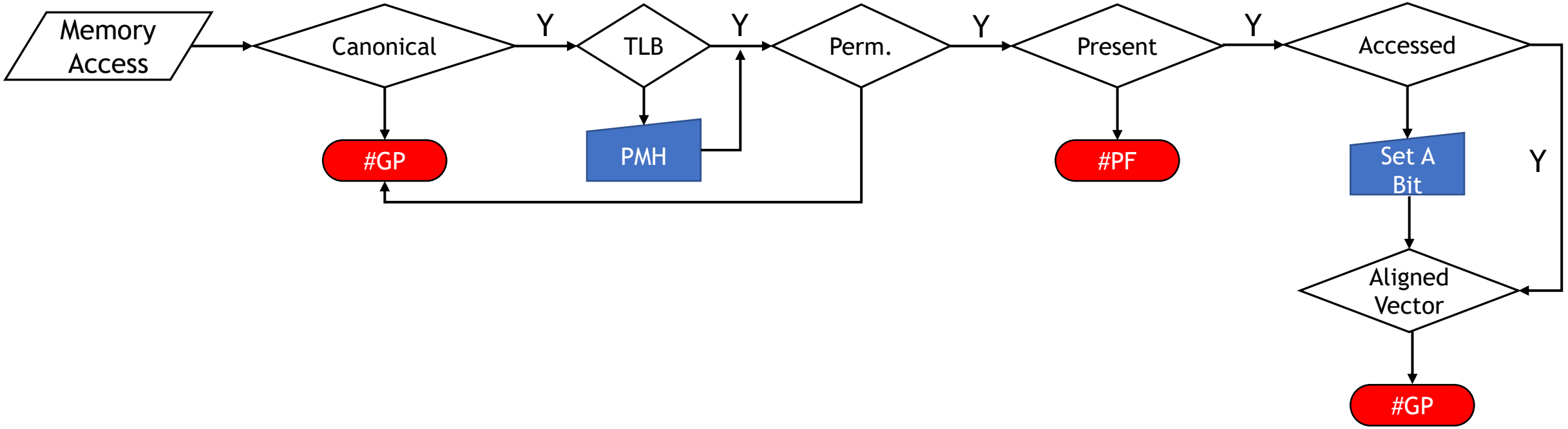


Virtual Address

VFN	Offset

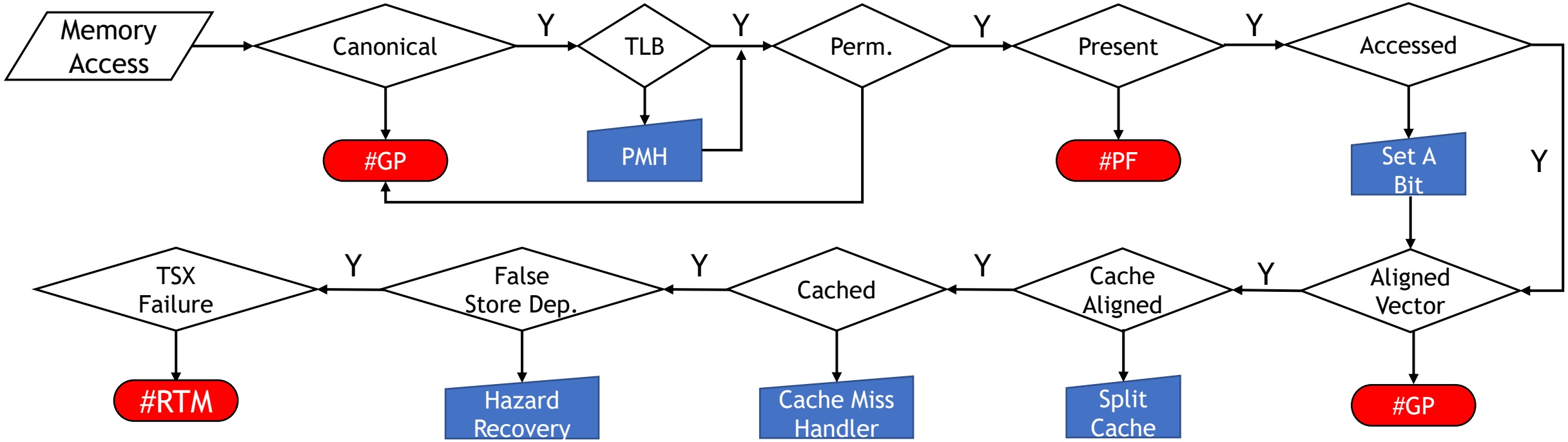
PTE

P	RW	US	…	A	…	Physical Page Number	…



Virtual Address

PTE



Virtual Address

PTE

Challenges with MDS Testing?

- Reproducing attacks is not reliable. It may depend on:
  - massaging the pipeline with other instructions
  - CPU configuration (generation, frequency, microcode patch and etc)

Challenges with MDS Testing?

- Reproducing attacks is not reliable. It may depend on:
  - massaging the pipeline with other instructions
  - CPU configuration (generation, frequency, microcode patch and etc)

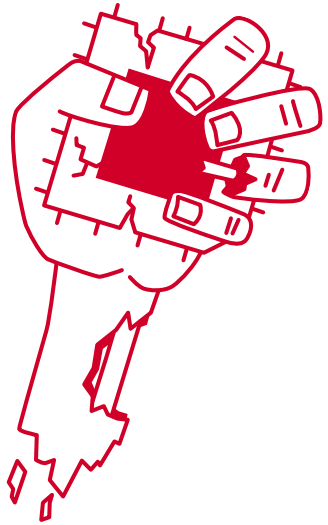
- No public tool to find new variants or to verify hardware patches:
  - Too many things to test (Addressing mode, cache state, assists, and faults)
  - Previous POCs may not work after MC update, but what does it mean?

Challenges with MDS Testing?

- Reproducing attacks is not reliable. It may depend on:
  - massaging the pipeline with other instructions
  - CPU configuration (generation, frequency, microcode patch and etc)

- No public tool to find new variants or to verify hardware patches:
  - Too many things to test (Addressing mode, cache state, assists, and faults)
  - Previous POCs may not work after MC update, but what does it mean?

- Impossible to quantify the impact of leakage:
  - We should care about leakage rate and what data is leaked.
  - My POC is faster than your POC!!



Transynther

Transynther (Fuzzing-based Random MDS Testing)

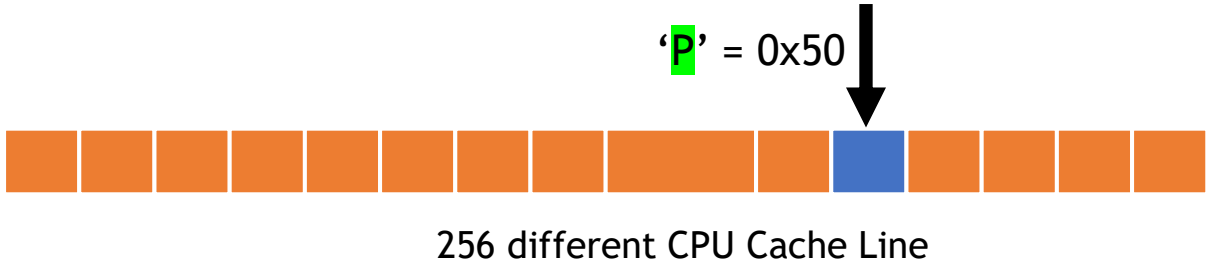
Step 1:

```
char secret = *(char *) 0xffffffff81a0123;
```

Step 2:

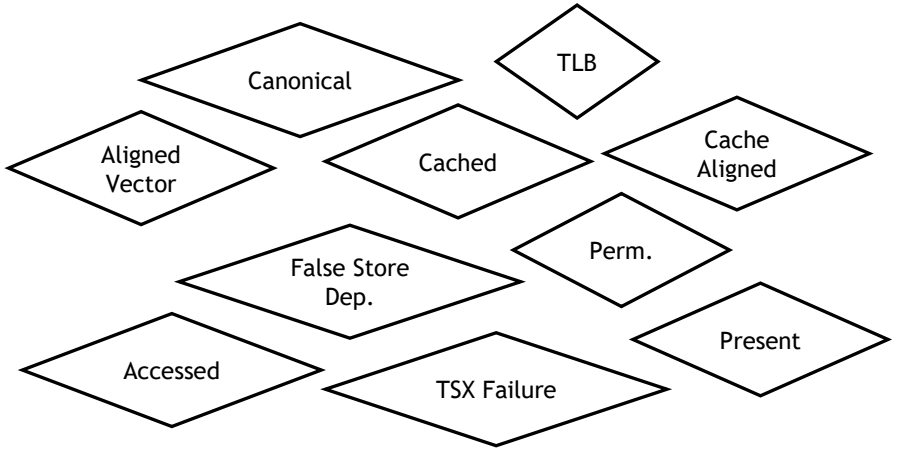
```
char x = oracle[secret * 4096];
```

Step 3:



Transynther (Fuzzing-based Random MDS Testing)

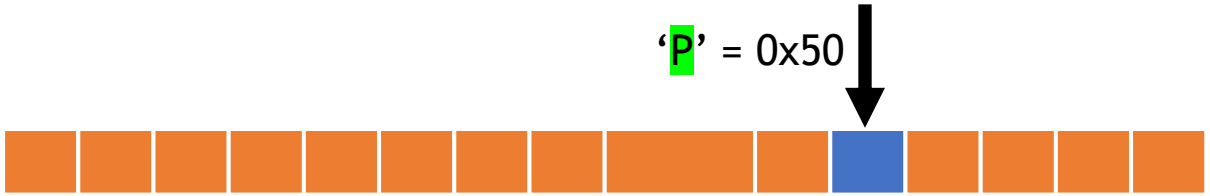
Step 1:



Step 2:

```
char x = oracle[secret * 4096];
```

Step 3:

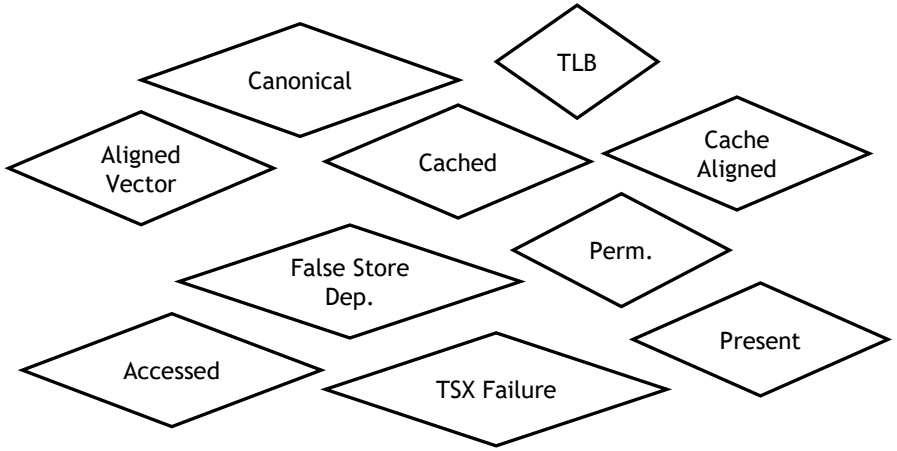


256 different CPU Cache Line

Transynther (Fuzzing-based Random MDS Testing)

Step 0: Buffer Grooming

Stores Same Thread: 0x41424344	Loads Same Thread: 0x51525354	Stores Hyper Thread: 0x61626364	Loads Hyper thread Thread: 0x71727374

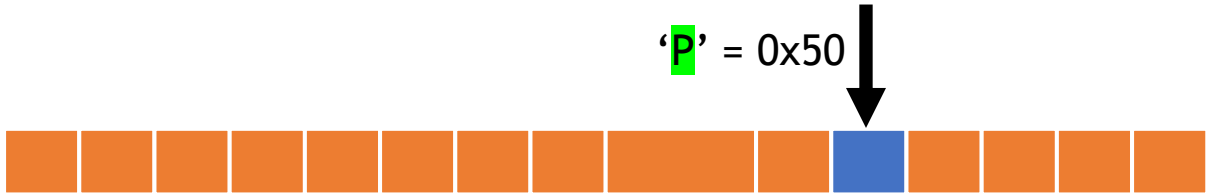


Step 1:

Step 2:

```
char x = oracle[secret * 4096];
```

Step 3:



256 different CPU Cache Line

Transynther (Fuzzing-based Random MDS Testing)

Step 0: Buffer Grooming

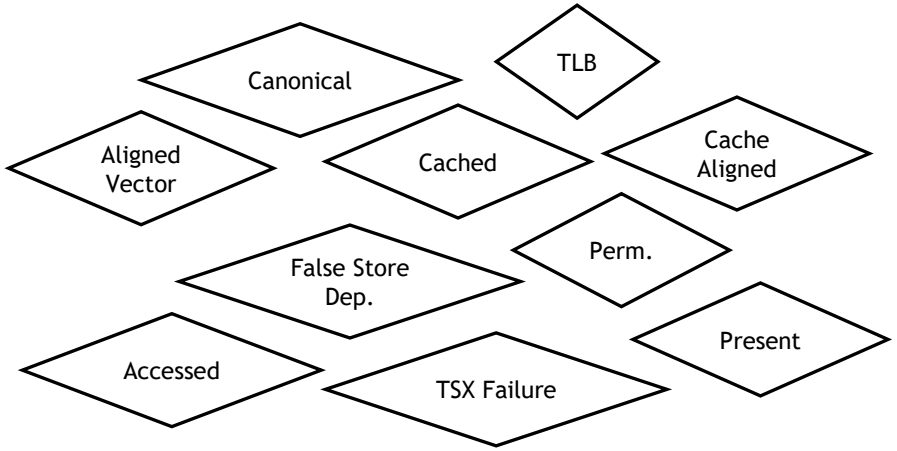
Stores Same Thread: 0x41424344

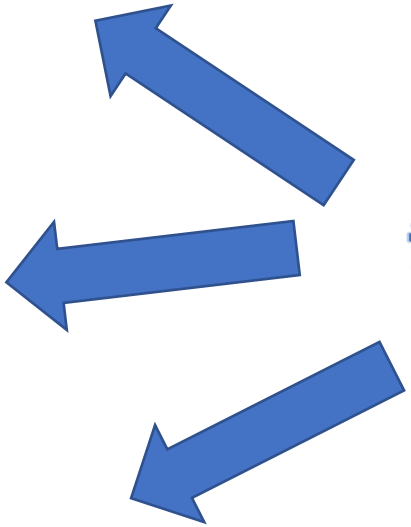
Loads Same Thread: 0x51525354

Stores Hyper Thread: 0x61626364

Loads Hyper thread Thread: 0x71727374

Step 1:



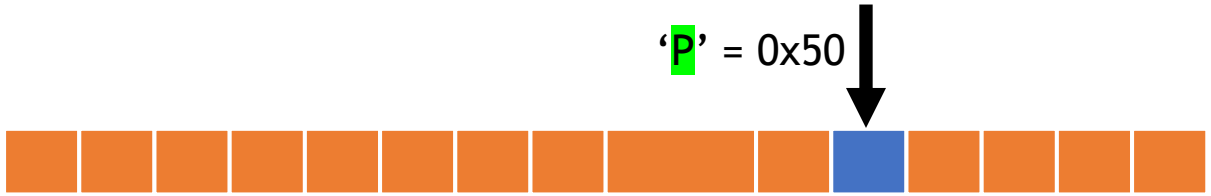


Random instructions

Step 2:

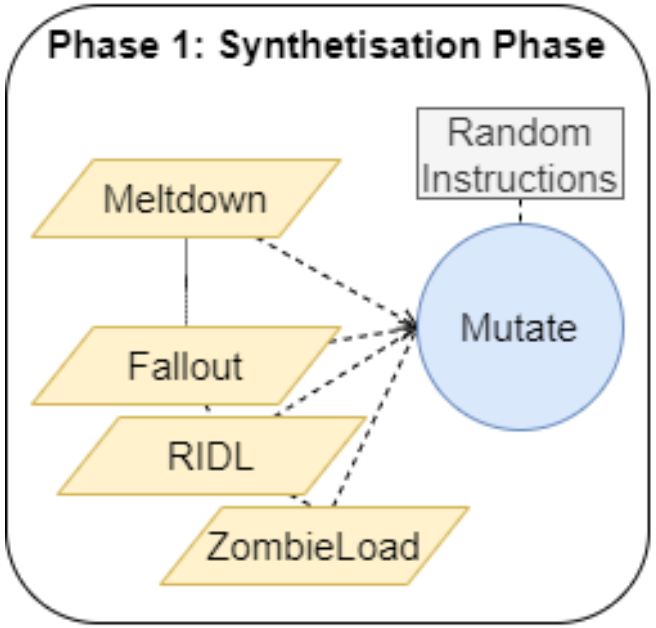
```
char x = oracle[secret * 4096];
```

Step 3:

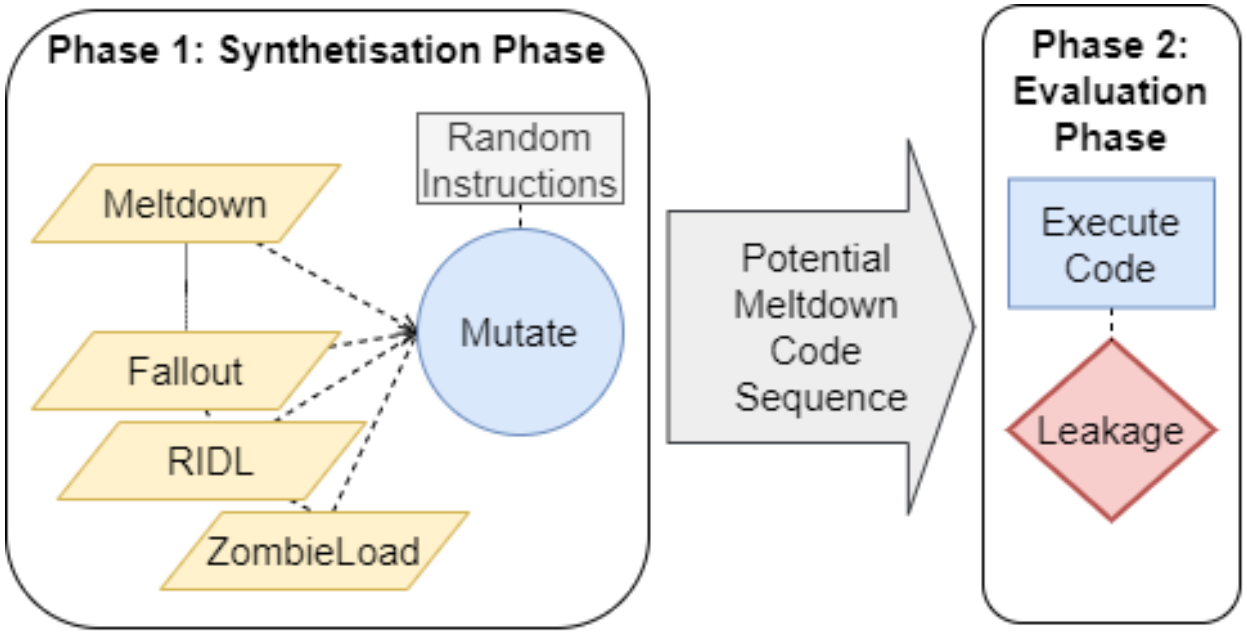


256 different CPU Cache Line

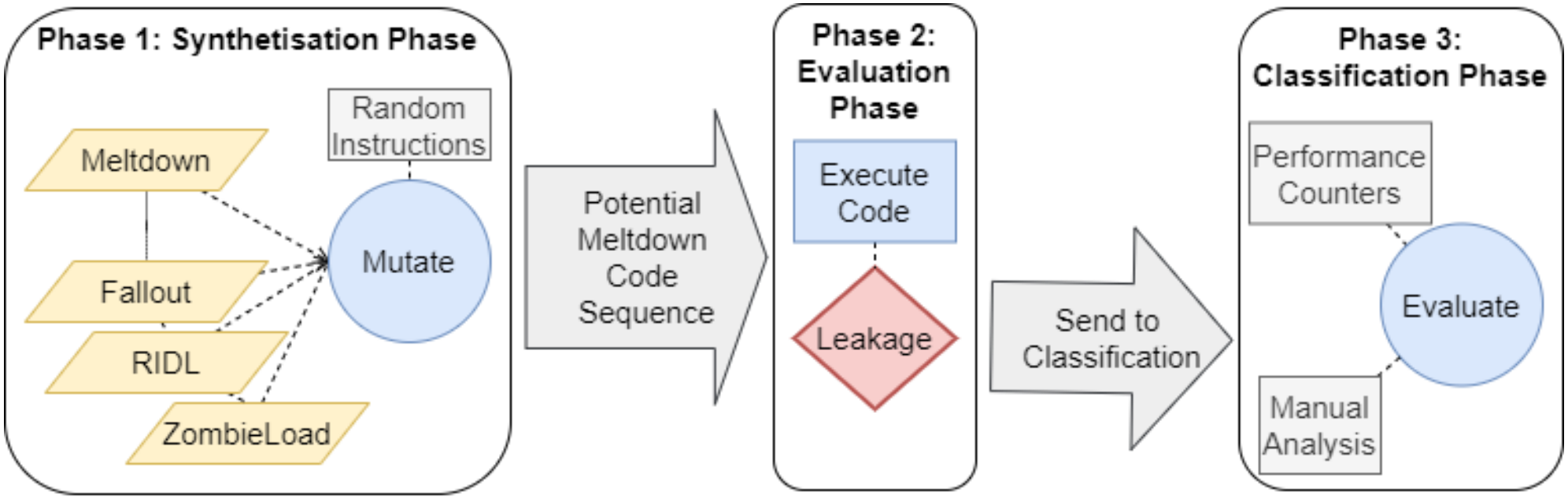
Transynther (Fuzzing-based MDS Testing)



Transynther (Fuzzing-based MDS Testing)



Transynther (Fuzzing-based MDS Testing)



Branch: master ▾ medusa / Transynther / _processed / NC / _ht_ /

Go to file

 danielmgmi committed 05ce08d 3 days ago

History

..

i3-7100_9_0x84_notsx.log_1_1835.asm	init	3 days ago
i3-7100_9_0x84_notsx.log_21829_473.asm	init	3 days ago
i7-7700_9_0x48.log_17_1764.asm	init	3 days ago
i7-7700_9_0x48.log_1_247.asm	init	3 days ago
i7-7700_9_0x48.log_21829_1025.asm	init	3 days ago
i7-7700_9_0x48.log_21829_1240.asm	init	3 days ago
i7-7700_9_0x48.log_21829_2188.asm	init	3 days ago
i7-7700_9_0x48.log_21829_2547.asm	init	3 days ago
i7-7700_9_0x48.log_21829_653.asm	init	3 days ago
i7-7700_9_0xca.log_21829_1175.asm	init	3 days ago
i7-7700_9_0xca.log_21829_541.asm	init	3 days ago
i7-7700_9_0xca.log_21829_561.asm	init	3 days ago

Table 2: Leakage variants discovered by Transynther.

Case	Preparation	Store	Load	Name
①	(access ⊘, random instructions)	-	≮ + 🔒 / ☝ / ⊘	MLPDS
②	(access ⊘, random instructions)	-	AVX ≮ + 🔒 / ☝ / ⊘	MLPDS
③	(access ⊘, random instructions)	-	AVX + 🔒 / ☝ / <✖>	Medusa
④	(access ⊘, random instructions)	-	AVX ⇄ + 🔒 / ☝ / ⊘ / <✖> / ✓	Medusa
⑤	-	store (to load)	🔒 / ☝ / <✖> / ✓	S2L
⑥	(`rep mov` + store, store + fence + load)	store (to load)	🔒 / ☝ / <✖> / ✓	-
⑦	-	store (4K Aliasing) + 🔒 / ☝ / ⊘ / <✖> / ✓	🔒 / ☝	MSBDS
⑧	-	store (4K Aliasing, to load) + 🔒 / ☝ / ⊘ / <✖> / ✓	AVX ⇄ + 🔒 / ☝ / ⊘ / <✖> / ✓	MSBDS, S2L
⑨	(Sibling on/off)	store (random address) + ⊘	🔒 / <✖>	MSBDS
⑩	(Sibling on/off + clflush (store address))	store (Cache Offset of Load) + ⊘	🔒 / <✖>	MSBDS
⑪	(Sibling on/off + repmov (to Load))	store (to Load)	AVX ⇄ + 🔒 / ☝ / ⊘ / <✖> / ✓	Medusa, MLPDS
⑫	-	Store (Unaligned to Load)	🔒 / ☝ / <✖>	Medusa
⑬	(random instructions)	AVX Store (to Load)	<✖>	Medusa, MLPDS, MSBDS
⑭	-	random fill stores	<✖>	MSBDS

<✖> Non-canonical Address Fault ⊘ Non-present Page Fault 🔒 Supervisor Protection Fault ⇄ AVX Alignment Fault
☝ Access-bit Assist ≮ Split-Cache Access Assist ✓ Access without fault or Assist

MDS Attacks - Insights

- Almost any exception/assist can leak from any buffer
- The CPU must flush the pipeline before executing an assist.
- Upon an Exception/Fault/Assist on a Load, Intel CPUs:
  - Execute the load until the last stage.
  - Flush the pipeline at the retirement stage (Cheap Recovery Logic).
  - Continue the load with some data to reach the retirement stage.

- Which data? (Fill buffer, Store Buffer, Load Buffer)
- Which one will be leaked first? (First come first serve)

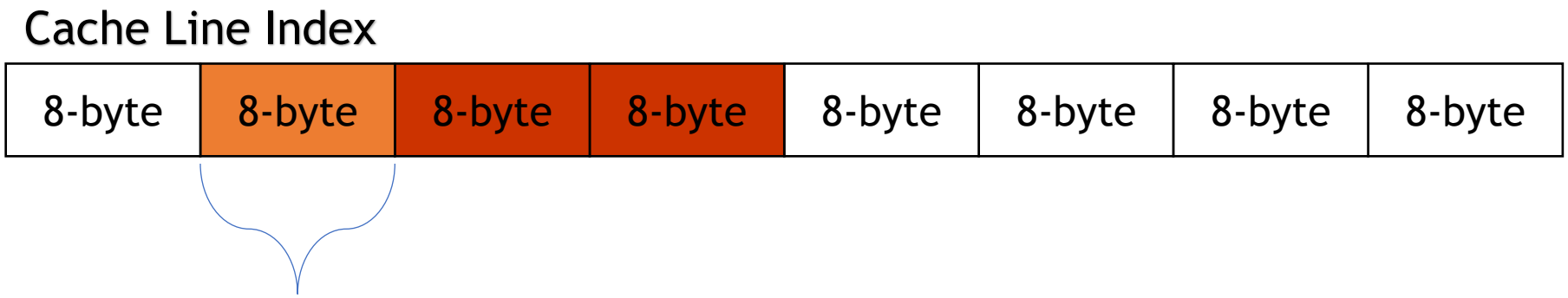
MEDUSA

Medusa Attack

- Medusa only leaks the Write Combining Data
- Implicit WC, i.e., ‘rep mov’, ‘rep sto’, can be leaked.
  - Memory Copy Routines
  - File IO
- Served by a Write Combining Buffer (or just the the Fill Buffer).

- Advantages:
  - Prefiltered data
  - Less Noise
  - More targeted

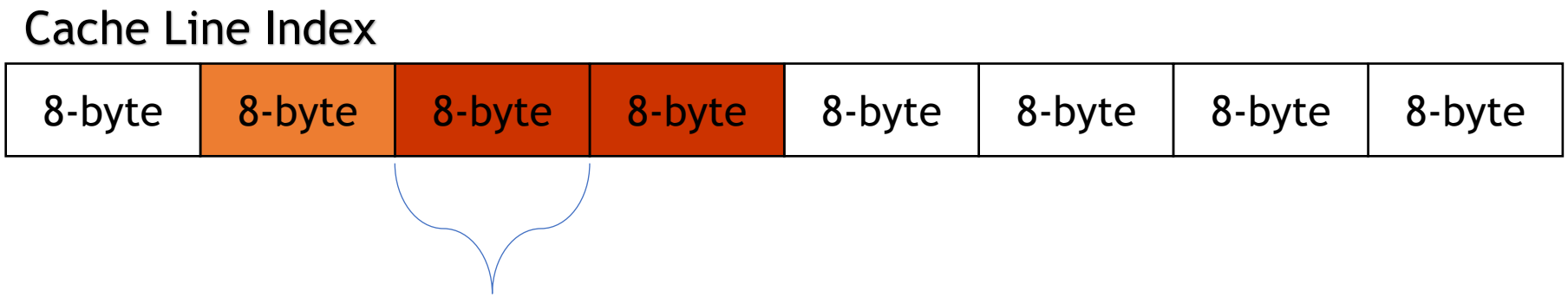
Medusa Attack – V1 Cache Indexing



Faulty Load

An invalid (Non-canon) address:
0x5550000000000008-20

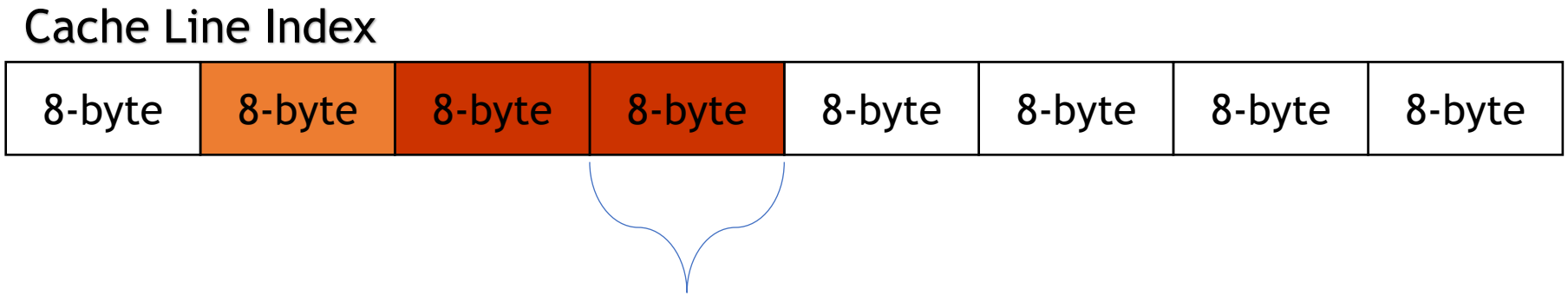
Medusa Attack – V1 Cache Indexing



Faulty Load

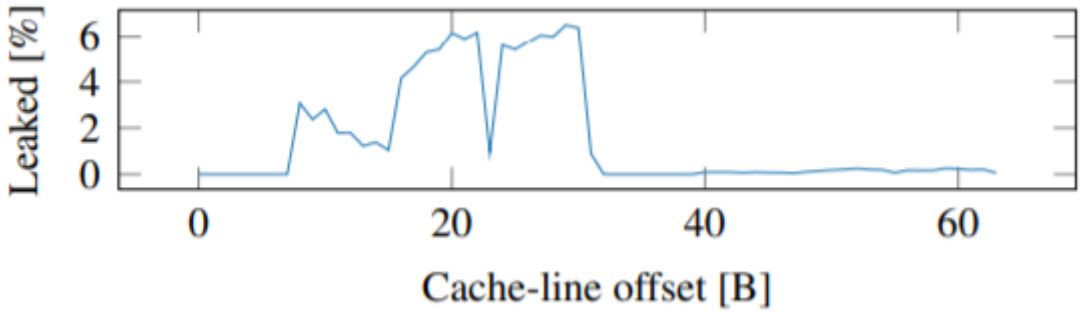
An invalid (Non-canon) address:
0x5550000000000008-20

Medusa Attack – V1 Cache Indexing



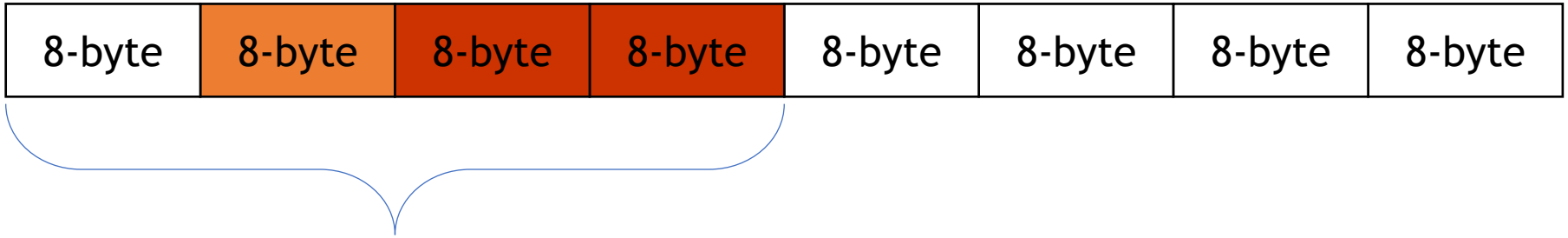
Faulty Load

An invalid (Non-canon) address:
0x5550000000000008-20

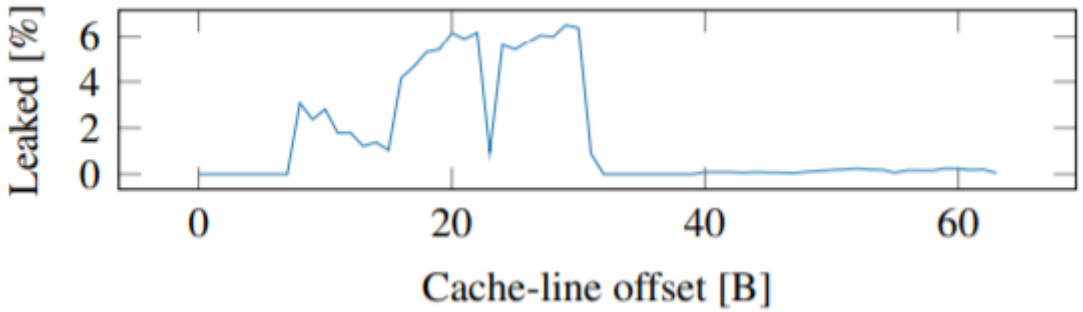


Medusa Attack – V1 Cache Indexing

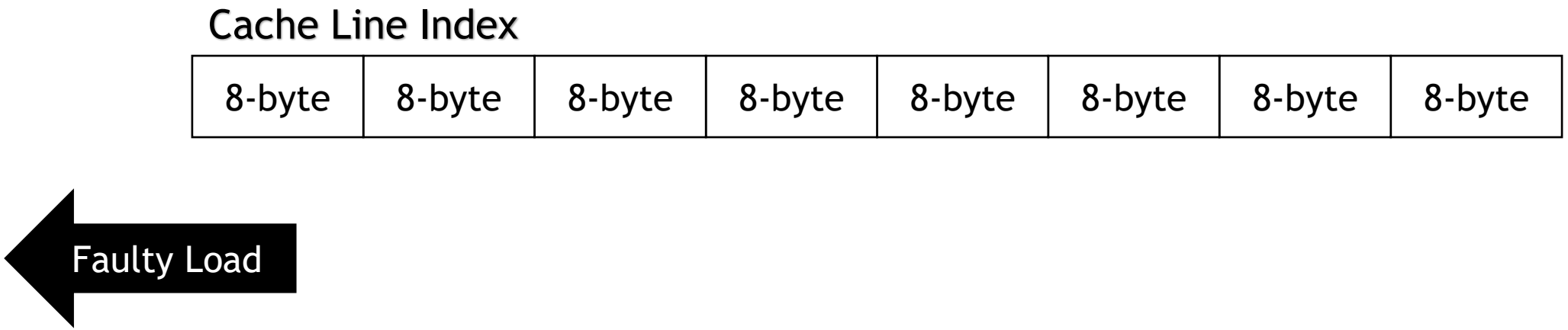
Cache Line Index



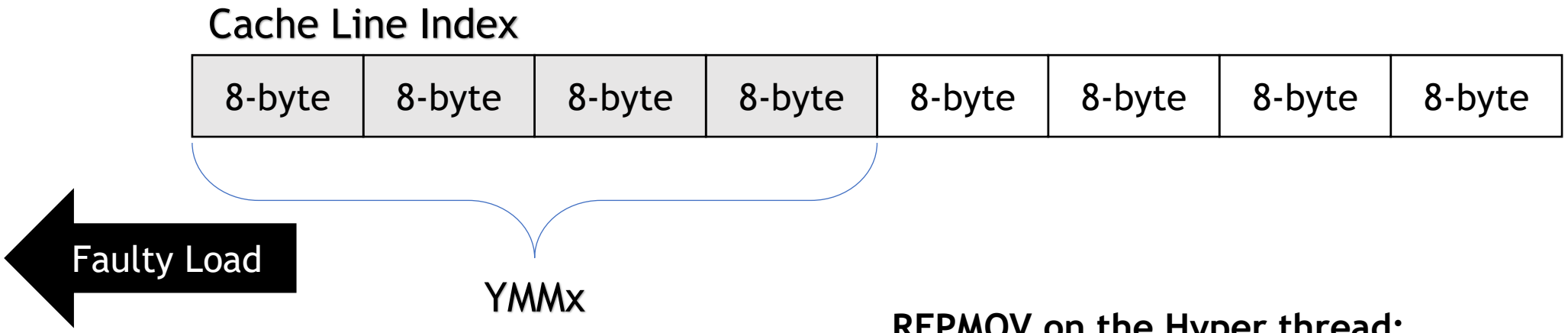
Common Data Bus?!



Medusa Attack – V2 Unaligned S2L Forwarding



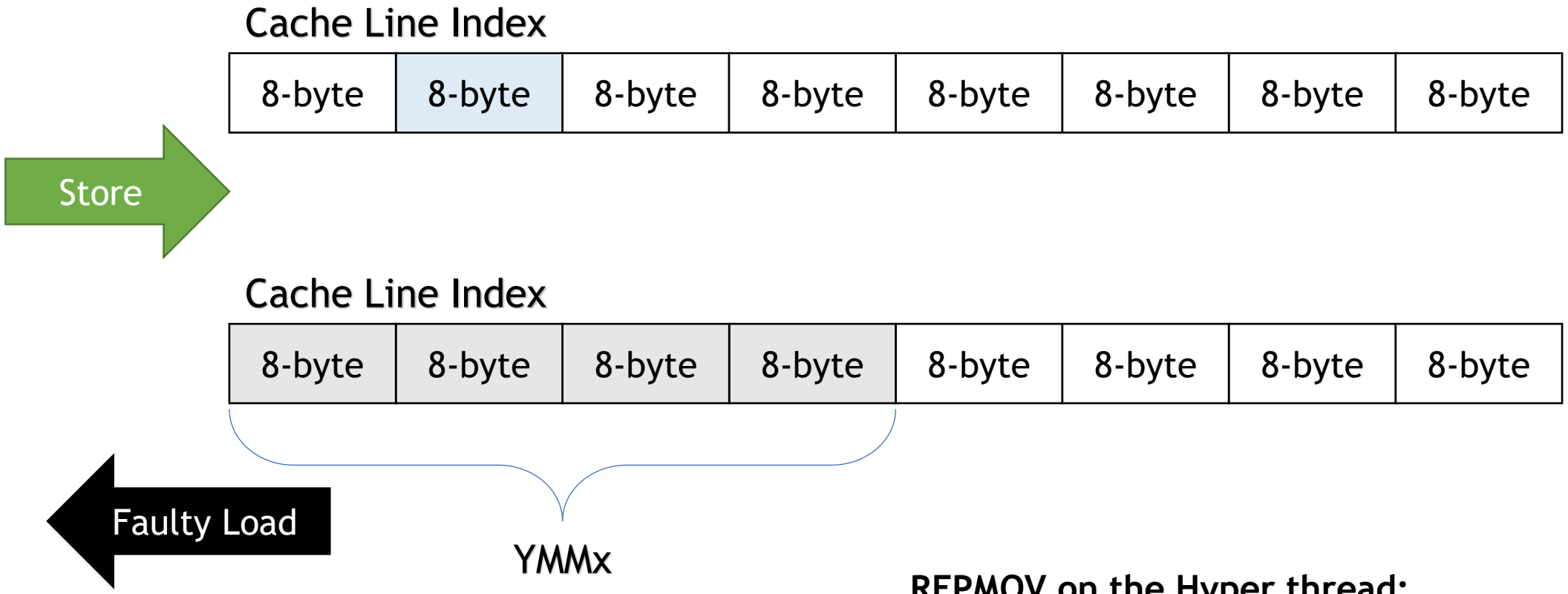
Medusa Attack – V2 Unaligned S2L Forwarding



REPMOV on the Hyper thread:

ABCDEFGH IJKLMNOP QRSTUVWX YZ…

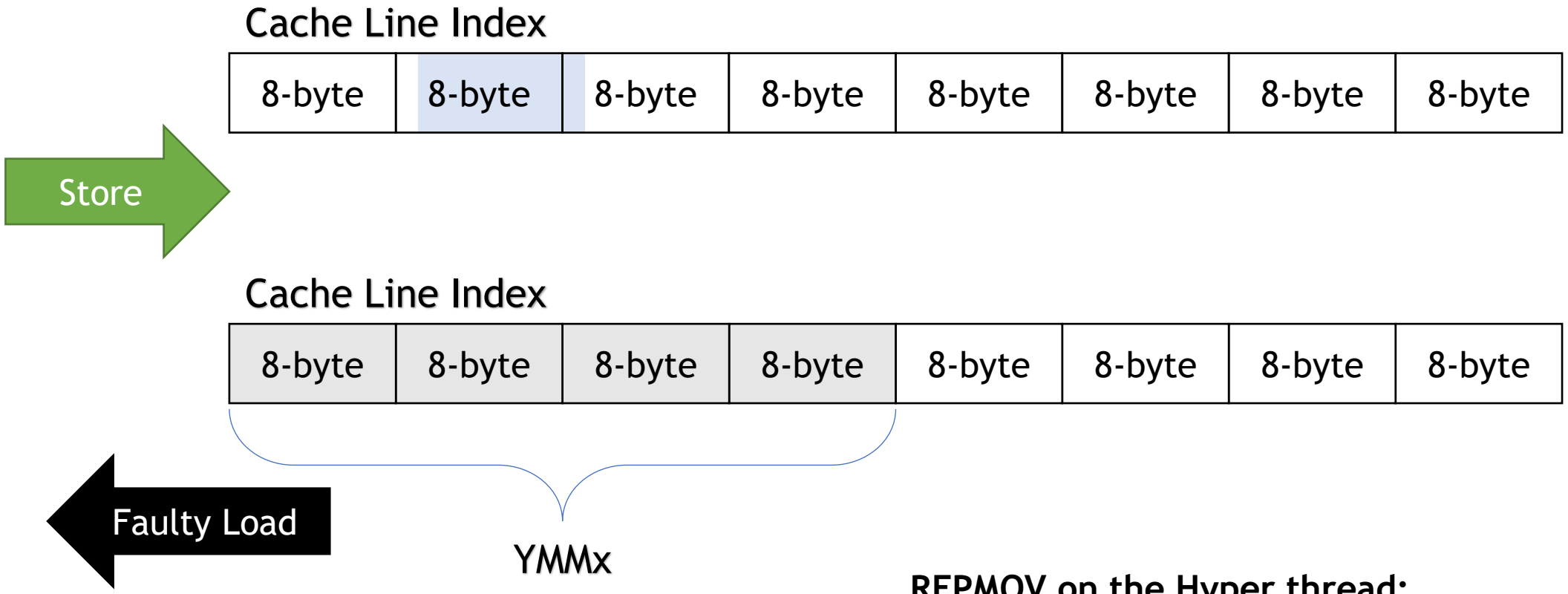
Medusa Attack – V2 Unaligned S2L Forwarding



REPMOV on the Hyper thread:

ABCDEFGH IJKLMNOP QRSTUVWX YZ…

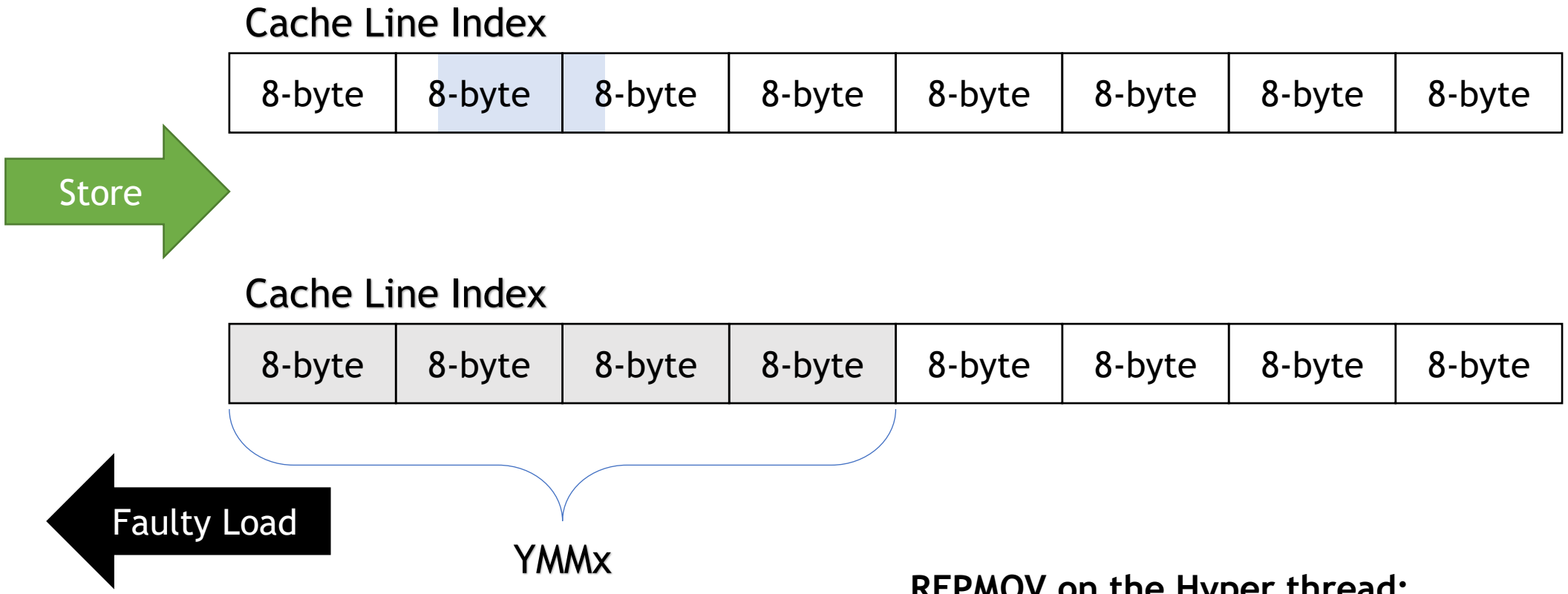
Medusa Attack – V2 Unaligned S2L Forwarding



REPMOV on the Hyper thread:

ABCDEFGH IJKLMNOP QRSTUVWX YZ…

Medusa Attack – V2 Unaligned S2L Forwarding

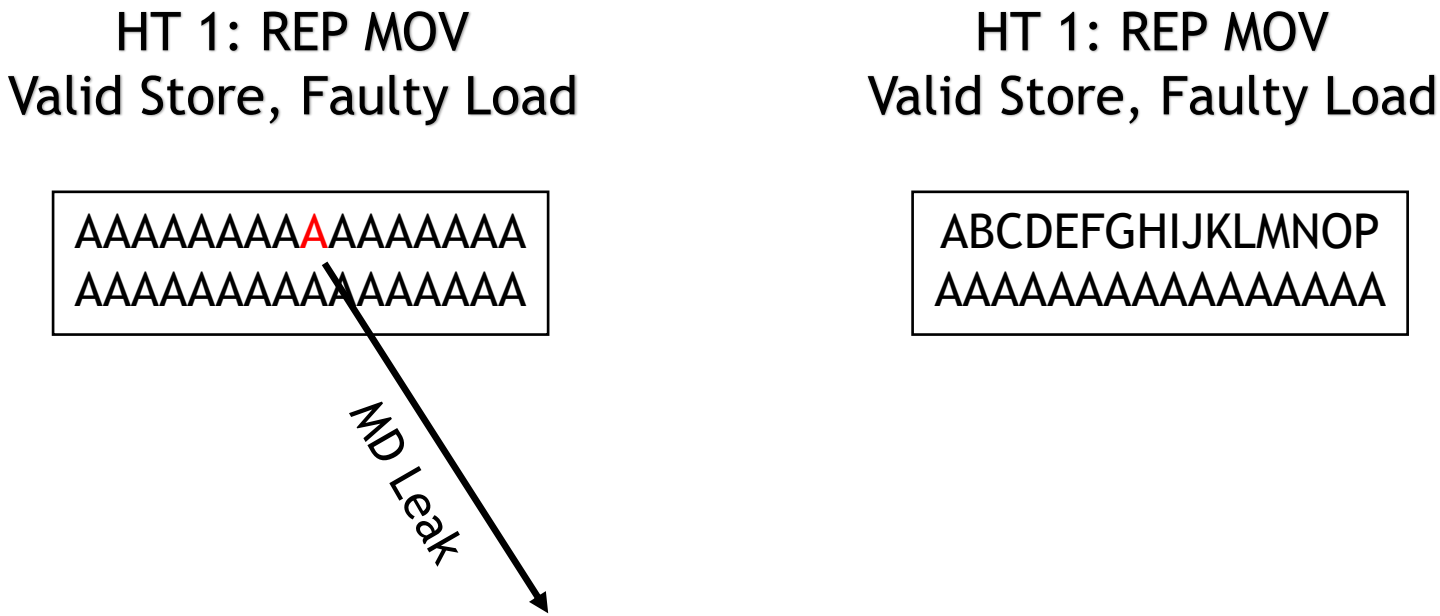


REPMOV on the Hyper thread:

ABCDEFGH IJKLMNOP QRSTUVWX YZ…

Medusa Attack – V3 Shadow *REP MOV*

- A *REP MOV* that fault on the load leaks:
  - the data from the legitimate store address
  - but also the data from the *REP MOV* running on the hyper thread



- A *REP MOV* that fault on the load leaks:
  - the data from the legitimate store address
  - but also the data from the *REP MOV* running on the hyper thread

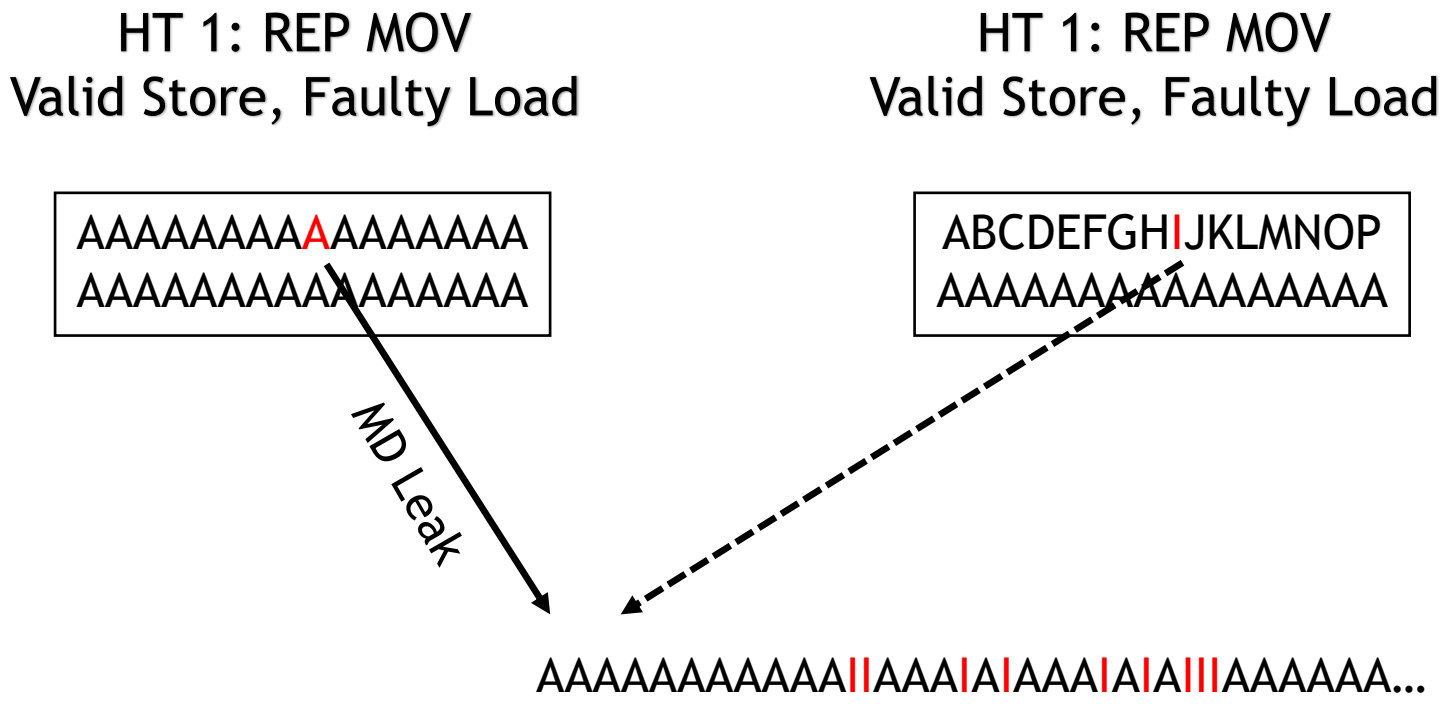
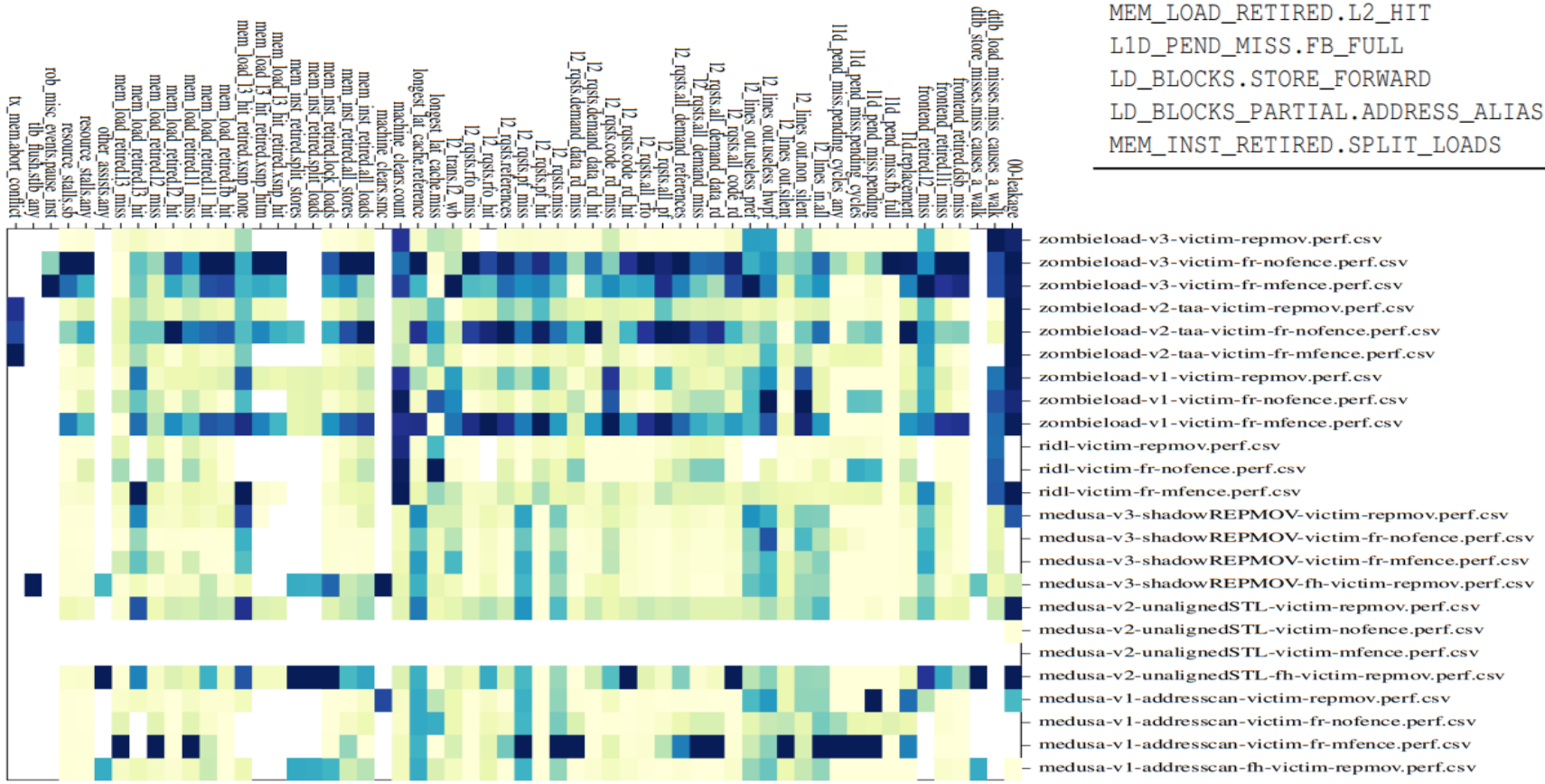


Table 1: The performance counters used in Transynther to identify the active microarchitectural elements.

Counter	Description
MEM_LOAD_RETIRED.FB_HIT	Data loaded from a line-fill-buffer entry.
MEM_LOAD_RETIRED.L1_HIT	Data loaded from the L1 data cache.
MEM_LOAD_RETIRED.L2_HIT	Data loaded from the L2 data cache.
L1D_PEND_MISS.FB_FULL	Data is neither in L1 nor in fill buffer.
LD_BLOCKS.STORE_FORWARD	Store buffer blocks load.
LD_BLOCKS_PARTIAL.ADDRESS_ALIAS	Load blocked by partial address match.
MEM_INST_RETIRED.SPLIT_LOADS	Data spans across two cache lines.



OpenSSL RSA Key Recovery

- OpenSSL Base64 Decoder uses inline Memcpy(-oS)
- Triggered during the RSA Key Decoding from the PEM format:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDmTvQjjtGtnIqMwmmaLW+YjbYTsNR8PGKXr78iYwrMV5Ye4VGy
BwS6qLD4s/EzCzGIDwkWCVx+gVHvh2wGW15Ddof0gVAtAMkR6gRABy4TkK+6YFSK
AyjmHvKCfFHvc9loeFGDyjmwFFkfdwzppXnH1Wwt0OlnyCU1GbQ1w7AHuwIDAQAB
AoGBAMyDri7pQ29NBIfMmGQuFtw8c0R3EamlIdQbX7qUguFEoe2YHqjdrKho5oZj
nDu8o+Zzm5jzBSzdf7oZ4qaeekv0fO+ZSz6CKYLbuzG2IXUB8nHJ7NuH3lacfivD
V4Cfg0yFnTK+MDG/xTVqywrCTsslkTCYC/XZOXU5Xt5z32FZAkEA/nLWQhMC4YPM
0LqMtgKzfgQdJ7vbr43WVVNpC/dN/ibUASI/3YwY0uUtqSjilIghIY7pRohrPJ6W
ntSJw0UAhQJBAOe2b9cfiOTFKXxyU4j315VkulFfTyL6GwXi/7mvpcDCixDLNRyk
uRigmdKjtIUrAX0pwjgXa6niqJ691jExez8CQQCcMZZAvTbZhHSn9LwHxqS0SIY1
K+ZxX5ogirFDPS5NQzyE7adSsntSioh6/LQKBX6BAR9FwtxBPACtwz5F9geZAkA8
a3z0SlvG04aC1cjkgUPsx6wxxbl79F2RhmSKRbvh7JiYk3RQ+L7vJgmWPGu5AcLM
oVPsjmbbkKfJZNTyVOW/AkABepEi++ZQQW0FXJWZ3nM+2CNcXYCtTgi4bGkvnZPp
/1pAy9rjeVJYhb8acTRnt+dU+uZ74CTtfuzUTZLOIuVe
-----END RSA PRIVATE KEY-----
```

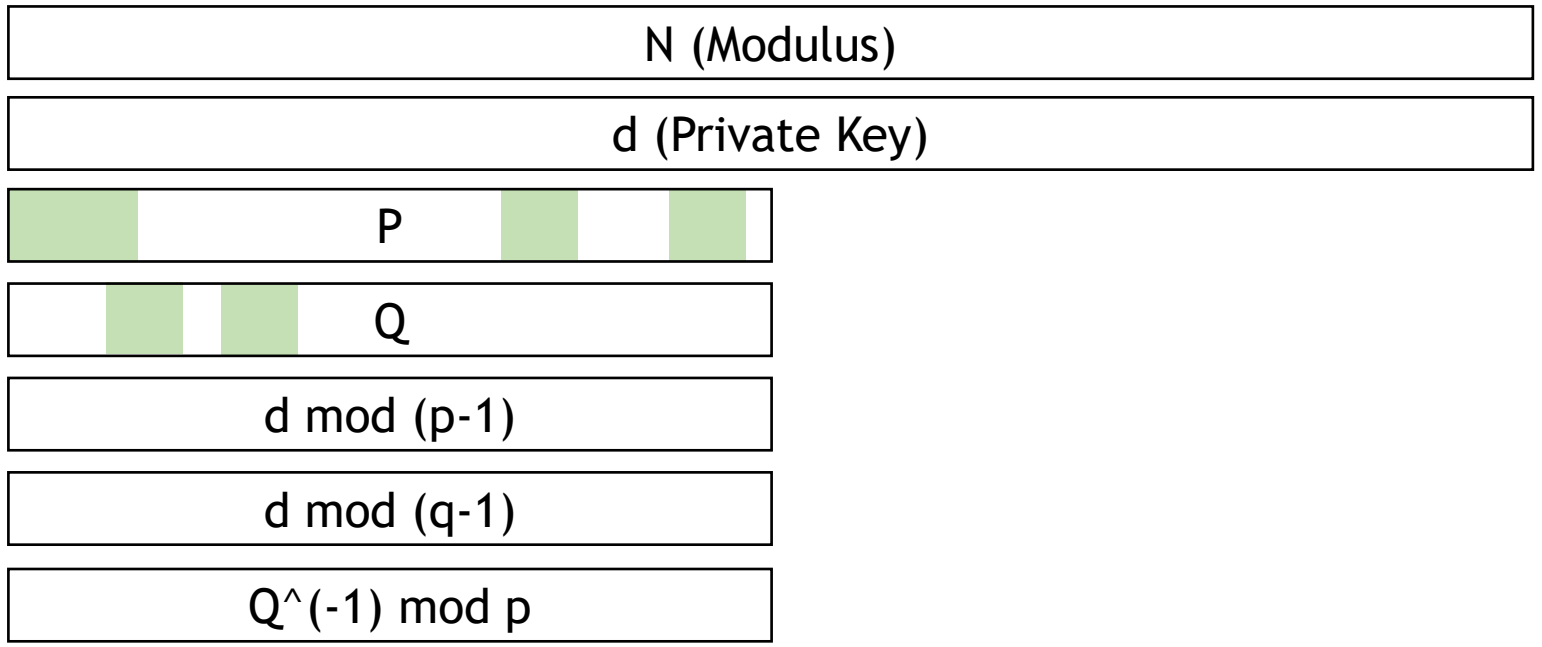
OpenSSL RSA Key Recovery

- OpenSSL Base64 Decoder uses inline Memcpy(-oS)
- Triggered during the RSA Key Decoding from the PEM format:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDmTvQjjtGtnIqMwmmaLW+YjbYTsNR8PGKXr78iYwrMV5Ye4VGy
BwS6qLD4s/EzCzGIDwkWCVx+gVHvh2wGW15Ddof0gVAtAMkR6gRABy4TkK+6YFSK
AyjmHvKCfFHvc9loeFGDyjmwFFkfdwzppXnH1Wwt0OlnyCU1GbQ1w7AHuwIDAQAB
AoGBAMyDri7pQ29NBIfMmGQuFtw8c0R3EamlIdQbX7qUguFEoe2YHqjdrKho5oZj
nDu8o+Zzm5jzBSzdf7oZ4qaeekv0fO+ZSz6CKYLbuzG2IXUB8nHJ7NuH3lacfivD
V4Cfg0yFnTK+MDG/xTVqywrCTsslkTCYC/XZOXU5Xt5z32FZAkEA/nLWQhMC4YPM
0LqMtgKzfgQdJ7vbr43WVVNpC/dN/ibUASI/3YwY0uUtqSjilIghIY7pRohrPJ6W
ntSJw0UAhQJBAOe2b9cfiOTFKXxyU4j315VkulFfTyL6GwXi/7mvpcDCixDLNRyk
uRigmdKjtIUrAX0pwjgXa6niqJ691jExez8CQQCcMZZAvTbZhHSn9LwHxqS0SIY1
K+ZxX5ogirFDPS5NQzyE7adSsntSioh6/LQKBX6BAR9FwtxBPACtwz5F9geZAkA8
a3z0SlvG04aC1cjkgUPsx6wxxbl79F2RhmSKRbvh7JiYk3RQ+L7vJgmWPGu5AcLM
oVPsjmbbkKfJZNTyVOW/AkABepEi++ZQQW0FXJWZ3nM+2CNcXYCtTgi4bGkvnZPp
/1pAy9rjeVJYhb8acTRnt+dU+uZ74CTtfuzUTZLOIuVe
-----END RSA PRIVATE KEY-----
```

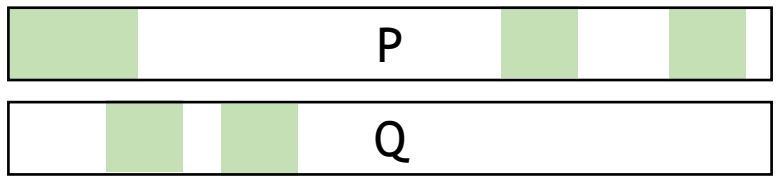
OpenSSL RSA Key Recovery

- OpenSSL Base64 Decoder uses inline Memcpy(-oS)
- Triggered during the RSA Key Decoding from the PEM format:

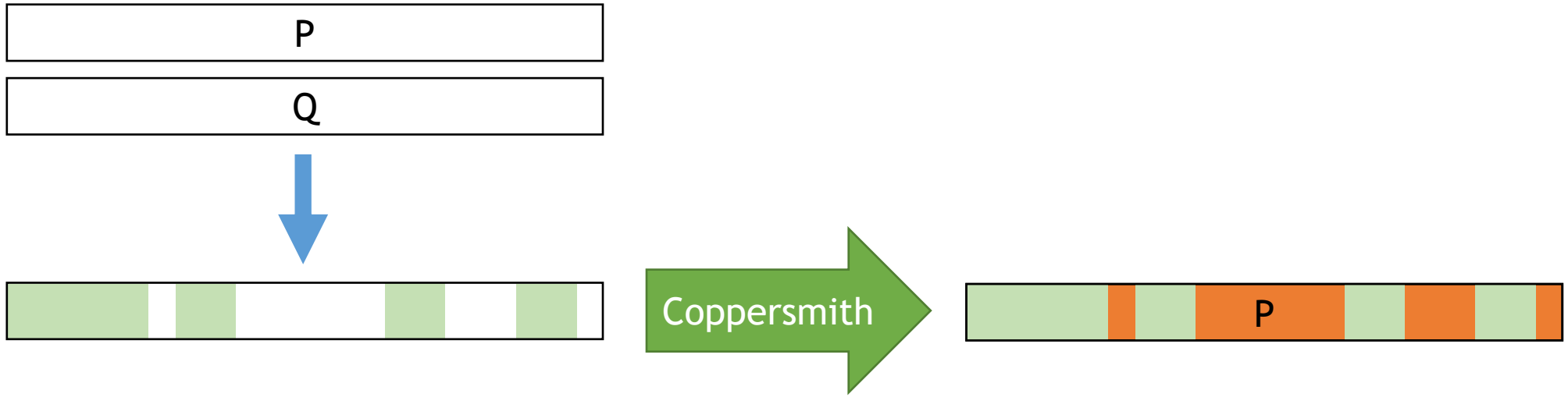


OpenSSL RSA Key Recovery - Coppersmith

- Knowledge of at least $^1/_3$ of P+Q
- Create a n dimensional hidden number problem where n is relative to the number of recovered chunks
- Feed it to the lattice-based algorithm to find the short vector



- Knowledge of at least $^1/_3$ of P+Q.
- Creating a n dimensional hidden number problem where n is relative to the number of recovered chunks.
- Feeding it to the lattice-based algorithm to find the short vector.



Responsible Disclosure

- Medusa
  - June 24, 2019: Reported initial findings to Intel
  - Intel confirmed that WC is part of the fill buffer, but embargoed due to TAA
  - Nov 12, 2019: $$$ Awarded

Conclusion

- Automated Testing for CPU Attacks
  - helps us to understand the root cause of these issues better.
  - can be used to verify hardware mitigations.
  - can help us to improve the leakage rate and understand the impact of attacks better.

- The impact of attacks depend also on the exploitation technique.

Conclusion

- Automated Testing for CPU Attacks
  - helps us to understand the root cause of these issues better.
  - can be used to verify hardware mitigations.
  - can help us to improve the leakage rate and understand the impact of attacks better.
- The impact of attacks depend also on the exploitation technique.

Conclusion

- Automated Testing for CPU Attacks
  - helps us to understand the root cause of these issues better.
  - can be used to verify hardware mitigations.
  - can help us to improve the leakage rate and understand the impact of attacks better.
- The impact of attacks depend also on the exploitation technique.

Responsible Disclosure (Ice Lake)

- MSBDS (Fallout) on Ice Lake
  - November 2019: Intel sent us an Ice Lake Machine (Hardware mitigations)

Responsible Disclosure (Ice Lake)

- MSBDS (Fallout) on Ice Lake
  - November 2019: Intel sent us an Ice Lake Machine
  - March 2019: Tested Transyther on the Ice Lake CPU

Responsible Disclosure (Ice Lake)

- MSBDS (Fallout) on Ice Lake
  - November 2019: Intel sent us an Ice Lake Machine
  - March 2019: Tested Transyther on the Ice Lake CPU

```
11 /** asm.S **/
12       .global  s_faulty_load
13 s_faulty_load :
14  lea  address_normal+0x4822, %r14     //  Store  address
15  lea  address_supervisor +0x822, %r15 //  Load address  (4K alias )
16 // clflush   (%r14)                    //  Uncomment to modify cache state
17 // lock; incl  (%r14)                  //  Uncomment to modify cache state
18  movb $0x41, (%r14)                   //  Store
19  movb (%r15), %al                     //  Faulty  Load
20  lea  oracles ,  %r13                 //  Encode
21  and $0xff,  %rax
22  shlq  $12,  %rax
23  movb (%r13,%rax,1), %al
24  ret
```

Responsible Disclosure (Ice Lake)

- MSBDS (Fallout) on Ice Lake
  - November 2019: Intel sent us an Ice Lake Machine
  - March 2019: Tested Transyther on the Ice Lake CPU
  - Mar 27, 2020: Reported MSBDS Leakage on Ice Lake
  - May 5, 2020: Intel Completed triage
    - MDS mitigations are not deployed properly
      - Chicken bits were not enabled for all mitigations.
      - OEMs shipped with old/wrong microcode.
    - Embargoed till July
  - July 13, 2020: MDS advisory and list of affected CPUs were updated.
    - $$$ Awarded

MC Version	MC Date	Vulnerable	Leakage (bytes/s)		
			clflush	*lock inc*	Unmodified
0x32 (stock)	2019-07-05	✓	577.87	754.99	1.58
0x36	2019-07-18	✓	148.24	529.84	0.62
0x46	2019-09-05	✓	130.15	695.80	0.11
0x48	2019-09-12	✓	271.69	620.07	0.59
0x50	2019-10-27	✓	96.54	542.10	0.25
0x56	2019-11-05	✓	145.46	751.40	0.08
0x5a	2019-11-19	✓	532.40	645.32	0.70
0x66	2020-01-09	✗	0	0	0
0x70	2020-02-17	✗	0	0	0
0x82	2020-04-22	✗	0	0	0
0x86	2020-05-05	✗	0	0	0

Processor	Stepping: All Unless Otherwise Noted	Microarchitectural Store Buffer Data Sampling CVE-2018-12126 / INTEL-SA-00233
		Intel Guidance: Microarchitectural Data Sampling Deep Dive
06_7EH	5	Hardware+MCU

Table 6: List of MDS-affected processors by Family/Model

Family_Model	Step	Processor Families / Processor Number Series	MFBDS	MSBDS	MLPDS
06_7EH	5	10th Generation Intel® Core™ Processor Family based on Ice Lake (U, Y) microarchitecture	No	Yes	No

057	MDS_NO Bit in IA32_ARCH_CAPABILITIES MSR is Incorrectly Set
Problem	MDS_NO bit (bit 5) in IA32_ARCH_CAPABILITIES MSR (10Ah) is set, incorrectly indicating full activation of all MDS (microarchitectural data sampling) mitigations.
Implication	Due to this erratum, the IA32_ARCH_CAPABILITIES MDS_NO bit incorrectly reports the activation of all MDS mitigations actions.
Workaround	It is possible for the BIOS to contain a workaround for this erratum.
Status	For the steppings affected, refer to the Summary Table of Changes.

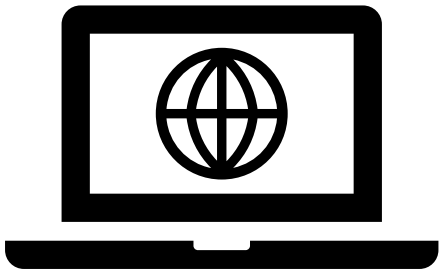
Questions?!

Daniel Moghimi
@danielmgmi

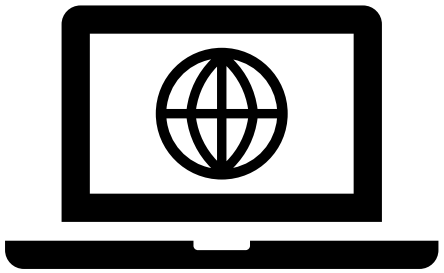
Moritz Lipp
@mlqxyz

Berk Sunar
@berksunar

Michael Schwarz
@misc0110



https://github.com/VernamLab/Medusa



https://github.com/danielmgmi/IceBreak