Advanced Computer Architecture Imperial College London

Chapter 5 part 1:

Sidechannel vulnerabilities



November 2022 Paul H J Kelly



Side-channels

What can we infer about another thread by observing its effect on the system state?

Through what channels?

How can we trigger exposure of private data?

How can we block side-channels?

Exfiltration

Thread A
(attacker)Thread B
("victim")Core #1Core #2L1D #1L1D #2

Shared L2

 Suppose we control thread A
 Suppose thread B is encrypting a message using a secret key, executing code we know but do not control

How can we program thread A to learn something (perhaps statistically) about B – perhaps the message?

Exfiltration⁴

Thread AThread B(attacker)("victim")



Suppose thread B's encryption algorithm is this simple:

```
For (i=0; i<N; ++i) {
```

```
C[i] = code[P[i]];
```

How can we program thread A to learn something (perhaps statistically) about P ?

Prime and Probe

- This technique detects the eviction of the attacker's working set by the victim:
 - The attacker first primes the cache by filling one or more sets with its own lines
 - Once the victim has executed, the attacker probes by timing accesses to its previouslyloaded lines, to see if any were evicted
 - If so, the victim must have touched an address that maps to the same set

Evict and Time

- This approach uses the targeted eviction of lines, together with overall execution time measurement
 - The attacker first causes the victim to run, preloading its working set, and establishing a baseline execution time
 - The attacker then evicts a line of interest, and runs the victim again
 - A variation in execution time indicates that the line of interest was accessed

- This is the inverse of prime and probe, and relies on the existence of shared virtual memory (such as shared libraries or page deduplication), and the ability to flush by virtual address
- The attacker first flushes a shared line of interest (by using dedicated instructions or by eviction through contention).
- Once the victim has executed, the attacker then reloads the evicted line by touching it, measuring the time taken
- A fast reload indicates that the victim touched this line (reloading it), while a slow reload indicates that it didn't



Flush



Figure 4: Even if a memory location is only accessed during out-of-order execution, it remains cached. Iterating over the 256 pages of probe_array shows one cache hit, exactly on the page that was accessed during the outof-order execution.

https://meltdownattack.com/meltdown.pdf

- Mon x86 the two steps of the attack can be combined by measuring timing variations of the clflush instruction
- The advantage of FLUSH+RELOAD over PRIME+PROBE is that the attacker can target a specific line, rather than just a cache set.

Side channels – shared state[®]

For a side channel to be exploited, we need to identify state that is affected by execution and shared between attacker and victim

If they share a single core:

L1I, L1D, L2, TLB, branch predictor, prefetchers, physical rename registers, dispatch ports...

Separate cores may share caches, interconnect etc



(A survey of microarchitectural timing attacks and countermeasures on contemporary hardware Q Ge, Y Yarom, D Cock, G Heiser - Journal of Cryptographic Engineering, 2018)

How can we trigger co-located execution of the victim?

System call

How can we trigger co-located execution of the victim?

- System call
- 🕨 Release a lock
- SMT threads co-scheduled on same core
- Call it as a function

How can we trigger co-located execution of the victim?

- System call
- 🕨 Release a lock
- SMT threads co-scheduled on same core
- Call it as a function
- Why is calling a function interesting?
 - Language-based security
 - Victim may be an object with secret state and a public access method

- Consider a web browser containing a Javascript interpreter
- Different web pages require Javascript execution for rendering
- Each web page's rendering is done by the browser
- But don't worry, the Javascript engine prevents page A from accessing page B's data
- Eg by array bounds checking:

Language-based security: Bounds checking



Side-channels in speculative execution

- Suppose the bounds check "if" is predicted satisfied
- But i is out of bounds
- So *p points to a victim web page's secret s (like the paypal password I just entered)
- So we can speculatively use s as an index into an array that we do have access to
- And then using timing to determine whether the cache line on which B[s] falls has been allocated as a side-effect of speculative execution



Side-channels in speculative execution

- Suppose the bounds check "if" is predicted satisfied
- But i is out of bounds
- So *p points to a victim web page's secret s (like the paypal password I just entered)
- So we can speculatively use s as an index into an array that we do have access to
- And then using timing to determine whether the cache line on which B[s] falls has been allocated as a side-effect of speculative execution

This is Spectre Variant #1



| E wetter | | | | |
|---|----|---|--|--|
| Fortile controls Fortile controls Fortile controls Fortile controls Fortile F | 14 | 14 unsigned int array1_size = 16; | | |
| <pre>mpige destruit (p _ an) File</pre> | 15 | <pre>15 uint8_t unused1[64];</pre> | | |
| <pre>citize came. context came. context came.context came</pre> | 16 | 16 | | |
| 1 D. 1 D. 2 D. | 17 | 1, | | |
| 5 5 5 6 10 7 10 7 10 7 10 7 10 7 10 7 10 7 10 7 | 18 | 2, Declare valid array1 for victim | | |
| 5. 5. 5. 6. 7. 7 | 19 | 3, to access | | |
| inter * servet = "the Right Works are Separately Decotings.") | 20 | 4, | | |
| <pre>wid = numeric = numer</pre> | 21 | 5, | | |
| Activity can Activity can marine COD (17) INTENDED(01) /* access can what is for a the control of V /* fourt test gaves is value(2) out reme-up in value(1) // | 22 | 6, | | |
| and rankwarphycical, withinks, with scalar(1), inf same(2)) (and rankwarphycical, withinks, withinks, and scalar (1), inf same(2)) (and visual(2, inf scalar), and inf scalar (1), inf scalar (2), in | 23 | 7, | | |
| <pre>int (s < 0; i < 10; i < 1</pre> | 24 | 8, | | |
| 1 we (1, *§, i + 2g, (a)) we (1, *§, i + 2g, (a)) (***) ***************************** | 25 | 9, | | |
| In the contrainty Large gravity study to the contraints list $x + x + x + x + x + x + x + x + x + x $ | 26 | 10, | | |
| 4 + () ((a > 0)) (b) (* for a < 0 + (d > 0)) 4 + (random g > (* (a (d) (minima a * random g))) (* (minima a (minima a)) (* (minima (minima (minima a))) (* (minima (mi | 27 | 11, | | |
|) $\label{eq:constraint} The model, define to Lightly strong up to present string prediction A' for [1+4] (2,200 [100] , [4,2] ((1+20)) < 10) at Eq.$ | 28 | 12, | | |
| <pre>main * a memory & a point (* main Them */ pain * a memory * memory * memory * memory * memory * tends * a memory * memory * main * memory * memory * memory * memory * memory * or (main * or Memory * memory</pre> | 29 | 13, | | |
|) (* Lambs lighted & angeot-lighted results results follow in $[0,1]$; (, 4 , 1); for $(1 \times F_2 + 20)$ (or) (, (, (, (, (, (, (, (, (, (| 30 | 14, | | |
| $\begin{array}{cccccccccccccccccccccccccccccccccccc$ | 31 | 15. | | |
| 10: at (vestid)() = 0 (2 - valid)() = (1) ((vestid)() = 1 = 0) 11: transf, C. Char scarmer (1 = 1 + 2 - values) = 0 (2 - 2 - 2 - 2 - 2 - 2 - 2 - 2 - 2 - 2 | 32 | 16 | | |
| 111 wale(1) < (dist).914.1 112 wale(1) = wale(1)(1) 114 min(1) = wale(1)(1) 115 min(1) = wale(1) = wale(1) 115 min(1) = wale(1) = wa | 33 | }: | | |
| 10 const within a structure structure (new *) wraph); structure to wraphic to within a structure struc | 34 | uint8 t unused2[64]: | | |
| 10: W (equive 3) if constructions, if and it is a substance of the subs | 35 | uint8 t arrav2[256 * 512]: Declare "cana | | |
| [10] priority thanking to (processing), length 20] with (in each of 1) in priority that (in each of a statistical exception of the statistical exception of the statistical exception in priority (his // priority (his // a statistical exception of the statistical exception of the priority (his // priority (his // a statistical exception of the statistical exception of the priority (his // priority (his // a statistical exception of the statistical exception of the priority (his // priority (his // a statistical exception of the statistical exception of the priority (his // priority (his // a statistical exception of the statistical exception of the priority (his // a statistical exception of the statistical exception of the statistical exception of the priority (his // a statistical exception of the statistical exception of the statistical exception of the priority (his // a statistical exception of the statistical exception of the statistical exception of the priority (his // a statistical exception of the statistical exception of the statistical exception of the statistical exception of the priority (his // a statistical exception of the statistical excep | 36 | cached-ness v | | |
| <pre>[10] (man(4) > 10 man(4) (10 7 man(4) ; "7), man(4)); 11 ((man(4) > 6) 21 grav(1)(man(4) math Bally man(4), man(1), man(1)); 12 grav(1)(man(4) math Bally man(4), man(1), man(1)); 13 grav(1)(man(4)); 14 (man(4));</pre> | 37 | char * secret = "The Magic Words are Sque | | |
| The second se | | | | |

In two pages of code: https://gist.github.com/ErikAugust/ 724d4a969fb2c6ae1bbd7b2a9e3d4b

b6

Declare "canary" array2 whose ached-ness we will probe

ords are Squeamish Ossifrage.";

| E spette.c | | | | |
|---|----|-----------------------------------|---|--|
| Minista Antonio | 14 | 14 unsigned int array1_size = 16; | | |
| Weise distribution, at a cost of the cost | 15 | 15 uint8_t unused1[64]; | | |
| International and the second sec | 16 | uint8_t | array1[160] = { | |
| | 17 | 1, | | |
| 1 - 1 - 5 | 18 | 2, | Declare valid array for victim to | |
| - 15. - 15. - 16. - 16. - 17. - 19. - | 19 | 3. | access | |
| $(1,1,2,\dots,2)$, and $(1,1,2,\dots,2)$) $(1,1,2,\dots,2)$ is a set of the | 20 | 4. | | |
| | 20 | ., | | |
| 1 Registri onto | | ⊃, | <pre>41 void victim_function(size_t x) {</pre> | |
| 1 Primaria (2012) 2015 Selecting (2014) Primaria (2014) 2015 Primaria | 22 | 6, | 12 if $(x < appa) (1 < izo) ($ | |
| 1 1.1.0.5 1.1.0.2.5 register and test tests, tests; 1.1.0.2.5 1.1.0.2.5 1 1.1.0.2.5 1.1.0.2.5 1 1.1.0.2.5 1.1.0.2.5 1 1.1.0.2.5 1.1.0.2.5 1 1.1.0.2.5 1.1.0.2.5 | 23 | 7, | 42 II (X Callay1_SIZE) [| |
| <pre>c = remain(c) + k) = f(c) = (remain(c) + (remain(c</pre> | 24 | 8, | 43 temp &= array2[array1[x] * 512]; | |
| M. Lawan, K. and K. S. Sang, Y. Sang, Y. S. Markan, S. Sang, S | 25 | 9, | 11 access "capary" array using data | |
| The constant is $M = X + X + X = M + X + M + M + M + M + M + M + M + M +$ | 26 | 10, | indexed out of bounds | |
| * training * (of pairing a strong a)) (* dations where v) (data strong a) | 27 | 11, | 45 } | |
|) $p_{1}=p_{2}^{-1}$ for ranks, there is a lightly model on the present starting predictions of $p_{1}=p_{1}^{-1}$ for $p_{1}^{-1}=p_{2}^{-1}$ (i.e. $p_{1}^{-1}=p_{1}^{-1}$ (i.e. $p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}$ (i.e. $p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}$ (i.e. $p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}$ (i.e. $p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}$ (i.e. $p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^{-1}=p_{1}^$ | 28 | 12. | | |
| 1 tank 4, "strain 4, popul," proto Theory " part - Margin 2, memory in Report Constraints for " tank 2, "strain 4, parts," in Report 1, popul, "A fill of the 4 competer Language training " protocols", "straints," in Report 1, protocols and straints," in Provide Tank 4, and protocols that protocols = 0.000 (pt) (theorem) and straints, "in Provide Tank 4, and protocols that protocols = 0.000 (pt) (theorem) and straints, "in Provide Tank 4, and protocols that protocols = 0.000 (pt) (theorem) and straints, "in Provide Tank 4, and protocols that protocols = 0.000 (pt) (theorem) and straints, "in Provide Tank 4, and protocols that protocols = 0.000 (pt) (theorem) and straints, "in Provide Tank 4, and protocols that protocols = 0.000 (theorem) and theorem 1, and theorem 1, and theorem 1, and the protocols = 0.000 (theorem) and theorem 1, and theorem 1, and theorem 1, and the protocols = 0.000 (theorem) and theorem 1, and theorem 1, and the protocols = 0.000 (theorem 1, and theorem 1, and theorem 1, and theorem 1, and the protocols = 0.000 (theorem 1, and theorem 1, and theorem 1, and the protocols = 0.000 (theorem 1, and theorem 1, and theorem 1, and theorem 1, and the protocols = 0.000 (theorem 1, and theorem 1, and theorem 1, and theorem 1, and theorem 1, and the protocols = 0.000 (theorem 1, and theorem 1, and theorem 1, and the protocols = 0.000 (theorem 1, and the protocols = 0.000 (theorem 1, and theorem 1, and the protocols = 0.000 (theorem 1, and theorem 1, and t | 29 | 13 | | |
|) and the state of the state o | 20 | 14 | | |
| iii iii (i < 1) | 30 | 14, | | |
| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 31 | 15, | | |
| 1 undel[1] (undel[1]) (undel[1]) 1 undel[1] (undel[1]) (undel[1]) 1 undel[1] (undel[1]) (undel[1]) 1 undel[1] (undel[1]) (undel[1]) | 32 | 16 | | |
| 101 In some per segne 101 In som | 33 | }; | Declare "canary" array whose | |
| interaction | 34 | uint8_t | unused2[64]; cached-ness we will probe | |
| matchings -= (cost parently - doment space state into a pattern */ end (cost parently - cost parently | 35 | uint8_t | array2[256 * 512]; | |
| <pre>iii = (-iii = (-i) (-i) (-iii = (-iii), (-iii = (-iii), (-iii = (-iii), (-iii))))))))))))))))))))))))))))))))))</pre> | 36 | | Secret message, out of bounds of victin | |
| <pre>import (reason that and heading submetty), satisf(), satisf();</pre> | 37 | char * : | <pre>secret = "The Magic Words are Squeamish Ossifrage.";</pre> | |





| E spectra.c | Rare | | 1 |
|--|------|--|--------------|
| Musican contractor Musican contractor Musican contractor Musican contractor | 53 | <pre>void readMemoryByte(size_t malicious_x, uint8_t value[2], int score[2]) {</pre> | L |
| Mendade caterizado (**ren elhano ant cititat */ Regnan caterizado (**ren elhano) Refer Mendade caterizado (**ren elhano) and cititat */ | 54 | <pre>static int results[256];</pre> | |
| Kentr Kentr Kentr Kentr Kentr Kentr Kentr | 55 | <pre>int tries, i, j, k, mix_i, junk = 0;</pre> | |
| <pre>i useigne (n) exempt into a 16; i useign usenest[10]; i useign xerevo[100] + (i i</pre> | 56 | <pre>size_t training_x, x;</pre> | |
| () 2. () 3. () 3. () 3. | 57 | <pre>register uint64_t time1, time2;</pre> | |
| 11 2. 2. 2. 2. 2. 2. 3. | 58 | <pre>volatile uint8_t * addr;</pre> | |
| 2 6. 6 6. 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 | 59 | | |
| 16. 11. 12. Lains (support (10)) 13. Lains (support (10)) 14. Lains (| 60 | for $(i = 0; i < 256; i++)$ | |
| $\label{eq:charge} charge = c^{-1} has Right limits are logarithe locality growth in the charge of the second sec$ | 61 | results[i] = 0; | |
| 1 and a variety fronting (see 1 + 0) { 17 (1 + 0 + 0 + 0 + 0 + 1) (1 + 0 + 0) 18 (1 + 0 + 0 + 0 + 0 + 0) (1 + 0 + 0) 18 (1 + 0 + 0 + 0) (1 | 62 | <pre>for (tries = 999; tries > 0; tries) {</pre> | |
| Augusta rate | 63 | | |
| Martine Config (UT) Traditional (QU) (* association and calles has been been been been been been been bee | 64 | /* Flush array2[256*(0255)] from cache */ | |
| the forms, h, i, h, etc., bank, - bank, - dy stark, brandstark, J, and - dy register scientific, i tools, transj. stark, i stark, i tools, transj. | 65 | for (i = 0; i < 256; i++) | Eluch array2 |
| $\label{eq:alpha} \begin{array}{l} \mbox{we}(x) + d(y) = -2 M_{2}(x) d(x) \\ \mbox{we}(x) + d(x) d($ | 66 | <pre>mm clflush(& array2[i * 512]); /* intrinsic for clflush instruction */</pre> | Flush arrayz |
| (* Time array[130(10, 130)] from terms */ from (+ 5 ± 1 + 20); (- 5 ± 1); (m) official(& array[1 + 532); (* intrinsic for intrinsic for sitility instruments). | 67 | | from the |
| translag r_{i} to the Karnyl size the constraints of the Karnyl size the constraints in the Annexyl size the constraints in the $r_{i} \in \{100\}$ (set) (1) r^{i} being (consists whereas) r_{i} | 68 | /* 30 loops: 5 training runs (x=training x) per attack run (x=malicious x) */ | cache |
| $r^{(2)}$ = 0.1 training (a set estraining x is filled as sullings x is filled (i) $r^{(2)}$ hold (page 3) may time in part the local problem $r^{(2)}$ $r^{(2)}$ (c) (1) (1) = 0.1 mod (p) (r) as sufficient (r) filled (x) for $r^{(2)}$ (r) $r^{(2)}$ = (1) (1) (1) = 0.1 mod (r) | 69 | training x = tries % array1 size; | |
| x - training x - i cA (millions x + logitor all) /* Call no views (*) with function() | 70 | for $(j = 29; j \ge 0; j -)$ { | |
| 11 1 17 The reads. Other is Lightly stored up to present string prediction V 18 to the (1, +), 1, 2, 200, 144.) | 71 | <pre>mm clflush(& array1 size);</pre> | |
| <pre>stat, + (11 × 107) + 11) & 3.35(at a star - (A a regulater, 1 × 101) b tast - (result, A peop); /* 101 p b tast - (result, A peop); /* 101 p tast - (result, a star - (result, A peop); / (result, a star - (result, a star -</pre> | 72 | <pre>for (volatile int z = 0; z < 100; z++) {} /* Delay (can also mfence) */</pre> | |
| <pre>if (1000 += CODE UT NUTDERD M area to envery(1000 h mengi size)) remultiples.j(=; /* code UD - and el to some ten UDs asiae */)</pre> | 73 | | Train the |
| Printer appent & munch-depent result results (allow in (97.7) prior (1) (1) (1) (1) (1) (1) (1) (1) (1) (1) | 74 | /* Bit twiddling to set x=training x if j%6!=0 or malicious x if j%6==0 */ | branch |
| $ \begin{array}{llllllllllllllllllllllllllllllllllll$ | 75 | /* Avoid jumps in case those tip off the branch predictor */ | |
| int (result)(1) = (1 + result)(1) + (3 + 1) (1 (result)(1) = 2 + 3 + result)(1 + 2 + 4) (1 + result) + Case scatters if best (1 + 2 + result) = 5 = 2 + 20 + 7 + 10 + 10 + 10 + 10 + 10 + 10 + 10 | 76 | x = ((j % 6) - 1) & ~0xFFFF; /* Set x=FFF.FF0000 if j%6==0, else x=0 */ | predictor |
| 100 consid[1] consid[1] consid[1] 101 consid[1] consid[1] consid[1] | 77 | x = (x (x >> 16)); /* Set x=-1 if j&6=0, else x=0 */ | |
| <pre>int int and modeline args, int constraints * * args) { int sing * minimum * constituents - (one *) array(1); /* meanit for minimum x */ int sing * minimum * constituents - (one *) array(1); /* meanit for minimum x */ int sing * meanity, one = 0;</pre> | 78 | | |
| $\label{eq:constraint} \begin{array}{llllllllllllllllllllllllllllllllllll$ | | /* Call the victim! */ | |
| 1 | | , , | |
| <pre>prior("framing & project(s", int);</pre> | | <pre>victim_function(x);</pre> | |
| | | | |

| E spectre.c | | | 21 |
|---|------------------|-----------------------------------|---------------------------------|
| Periodia Antonio Periodia Antonio Periodia Antonio Periodia Antonio Periodia Antonio Periodia Antonio (Providence antonio Providence Antonio | Flush array2 | | |
| Mysens edition("gr",en) Mysens editor("gr",en) Myseline editor("gr",en) Myseline editor("gr",en) | | | |
| i pressioner and a second seco | / from the | | |
| <pre>interface in array i.i.i. i.i.i i.i.i.i.i.i.i.i.i.i.i.i i.i.i.i.</pre> | (cacho | | |
| 17 - 16 17 - 36 18 - 56 | Lache | | |
| 12 4. 13 5. 14 6. 15 7. | | | |
| 1 6. 1 6. | Train the | | |
| 10. 17. II. 16. | Irain the | | |
| 1 11, 1 16 1 17 18 19 10 10 10 10 10 10 10 10 10 10 | branch | | |
| sioned a arrayo(2106 + 513); :ther * sammat = "The Regic Harma are Squantian Decifrege."; | branen | | |
| <pre>ident().temp + 4; /> lowed as complier sen's epiteine ast virths function() */ i ident().temp + 4; /> lowed as complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast complier sen's epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast compliant epiteine ast virths function() */ ident().temp + 4; /> lowed ast complised ast compliant epiteine ast virths function()</pre> | predictor | | |
| 1* () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () < () | | | |
| Analysis inde | | 7 | |
| White CAUE (107:7087080.2000) /* assume tame with 10 time or dreamaild */ /* Report best games in united[2] and nummer-up in united[2] */ | Call the | | |
| $\label{eq:constraints} \begin{array}{l} \mbox{traints} \ trai$ | victim | | |
| register intel(+ ton), tin2; collection intel(+ amp; collection intel | Victim | | |
| <pre>result()[1 = 0; = trr (tries = 00; tries.) 0; tries) { = trr (tries = 00; tries.) 0; tries tries) { = trr (tries = 00; tries.) 0; tree tries</pre> | | _ | |
| Wer (1 + 4) + 202 (1+) we obtain(4 wraph(+ 522))) /* intrimate the obtain interaction by an obtain(4 wraph(+ 522)) /* intrimate the obtain one framework of a state of the obtained of the obtained of the obtained one framework on the analysis of the obtained of the obtained of the obtained one framework of the analysis of the obtained of the obtained of the obtained one framework of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the analysis of the obtained of the obtained of the obtained of the obtained of the analysis of the obtained of the analysis of the obtained of the ob | 85 /* Time rea | ads. Order is lightly mixed up to | prevent stride prediction */ |
| $\label{eq:constraint} \begin{aligned} & \text{transmits} \{x \in \text{transmits} \} \text{ and} \\ & \text{transmits} \ x \in t$ | 86 for (i = 0 | ; i < 256; i++) { | |
| (* Attribute is a figure of the strength of | 87 miyi- | ((i * 167) + 13) & 255 | Prohe cache |
| $\tau = \tau < 0$ [(a to a bit)) τ^{2} for $\tau > 1$ (a both or τ^{2} (both or τ^{2}) $\tau = \tau =$ | | | |
| alota function(a); | addr = & | array2[m1x_1 * 512]; | and time |
| $ \begin{array}{llllllllllllllllllllllllllllllllllll$ | 89 time1 = _ | rdtscp(& junk); /* READ TIMER * | accesses |
| 10 table | 90 junk = * | addr; /* MEMORY ACCESS TO TIME */ | |
| 10 A series highest & writehighest results results tailing in [20.5] (1) (2.5.5.2) | 91 time2 = | rdtscp(& junk) - time1; /* READ | TIMER & COMPUTE ELAPSED TIME */ |
| iii the (1, i i j : d dd (i + i) (iii (2, i i j) = monits(1)) = monits(2)) (iii i i + i) iii (i + i) | 97 if (time) | 2 (- CACHE HIT THRESHOLD && mix i | - annavi[tries % annavi size]) |
| | | | |
| (a) (constraint) * Close states if their (1 ≤) if (constraint) = 2 = 200 k · (s) = (1 = 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 | 93 results | s[mix_i]++; /* cache hit - add +1 | to score for this value */ |
| 1 amoli - empirie cumulat - compared 10 manual 11 - compared 10 manu | 94 } | | |
|)) main(int agt,) dami day ** agp) [| | | |
| 11 Along Smilling, a (10, 2)(herest - (200 *) array(1) /* metalli for malining (* ?) 11 (10), A (met)(2), Ber (4), 12 (10), a (10) | Do some statis | tics to | |
| $\label{eq:starting} \begin{array}{c} \mbox{tr} \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$ | | | |
| autoring x = (vine i) arrayi; (* forwart input value into a pointse */ manuf(array(1), *M: A lest;) | find outlier acc | Cess Print the most li | celv |
| <pre>print(Taxing & types(*, int); while (-int = vi); print(*)Sating = while inter v = (e_1, *, (out *) while inter, i); print(*)Sating = while inter v = (e_1, *, (out *) while inter, i); result(*)Sating = while inter v = (e_1, *, v); result(*)Sating = (e_1, *, v); resu</pre> | times | | |
| <pre>put("bit", (count)({bit", i=2 + count)({bit", i=count}); "counts"; counts"; (count)({bit", i=count}); (count)({bit",</pre> | | character values | trom |
| | | the secret messa | G A |
| (1)(0); (2)(0); | | | 5C |

| \$./spectre-gcc00 | |
|--|--|
| Reading 40 bytes: | |
| Reading at malicious_x = 0xffffffffffffffffffff Unclear: 0x54='T' score=998 | 3 (second best: 0x01 score=745) |
| Reading at malicious_x = 0xfffffffffffffffffffff Unclear: 0x68='h' score=997 | 7 (second best: 0x01 score=750) |
| Reading at malicious_x = 0xffffffffffffffffffa Unclear: 0x65='e' score=996 | 6 (second best: 0x01 score=749) |
| Reading at malicious_x = 0xfffffffffffffffffff Unclear: 0x20=' ' score=995 | 5 (second best: 0x01 score=747) |
| Reading at malicious_x = 0xffffffffffffffffffffffffffffffffff | 0 (second best: 0x01 score=716) |
| Reading at malicious_x = 0xfffffffffffffffdfedfd Unclear: 0x61='a' score=997 | 7 (second best: 0x01 score=734) |
| Reading at malicious_x = 0xfffffffffffffffedfe Unclear: 0x67='g' score=999 | 0 (second best: 0x01 score=699) |
| Reading at malicious_x = 0xffffffffffffffffffffffffffffffffff | 7 (second best: 0x01 score=715) |
| Reading at malicious_x = 0xfffffffffffffee00 Unclear: 0x63='c' score=998 | 3 (second best: 0x01 score=741) |
| Reading at malicious_x = 0xfffffffffffffee01 Success: 0x20=' ' score=2 | |
| Reading at malicious_x = 0xfffffffffffffee02 Unclear: 0x57='W' score=978 | 3 (second best: 0x01 score=725) |
| Reading at malicious_x = 0xffffffffffffffee03 Unclear: 0x6F='o' score=996 | 5 (second best: 0x01 score=742) |
| Reading at malicious_x = 0xfffffffffffffee04 Unclear: 0x72='r' score=998 | 3 (second best: 0x01 score=733) |
| Reading at malicious_x = 0xfffffffffffffee05 Unclear: 0x64='d' score=986 | 5 (second best: 0x01 score=741) |
| Reading at malicious_x = 0xfffffffffffffee06 Unclear: 0x73='s' score=999 | 0 (second best: 0x01 score=733) |
| Reading at malicious_x = 0xfffffffffffffee07 Unclear: 0x20=' ' score=997 | 7 (second best: 0x01 score=745) |
| Reading at malicious_x = 0xfffffffffffffee08 Unclear: 0x61='a' score=996 | 5 (second best: 0x01 score=706) |
| Reading at malicious_x = 0xfffffffffffffee09 Unclear: 0x72='r' score=998 | 3 (second best: 0x01 score=697) |
| Reading at malicious_x = 0xffffffffffffffee0a Unclear: 0x65='e' score=995 | 5 (second best: 0x01 score=710) |
| Reading at malicious_x = 0xfffffffffffffee0b Unclear: 0x20=' ' score=997 | 7 (second best: 0x01 score=731) |
| Reading at malicious_x = 0xffffffffffffffee0c Unclear: 0x53='S' score=996 | 5 (second best: 0x01 score=721) |
| Reading at malicious_x = 0xffffffffffffffee0d Unclear: 0x71='q' score=992 | 2 (second best: 0x01 score=731) |
| Reading at malicious_x = 0xffffffffffffffee0e Unclear: 0x75='u' score=997 | 7 (second best: 0x01 score=731) |
| Reading at malicious_x = 0xffffffffffffffee0f Unclear: 0x65='e' score=994 | (second best: 0x01 score=760) |
| Reading at malicious_x = 0xffffffffffffffee10 Unclear: 0x61='a' score=988 | 3 (second best: 0x01 score=714) |
| Reading at malicious_x = 0xffffffffffffffee11 Unclear: 0x6D='m' score=994 | (second best: 0x01 score=728) |
| Reading at malicious_x = 0xffffffffffffffee12 Unclear: 0x69='i' score=998 | 3 (second best: 0x01 score=750) |
| Reading at malicious_x = 0xffffffffffffffee13 Unclear: 0x73='s' score=999 | 0 (second best: 0x01 score=749) |
| Reading at malicious_x = 0xffffffffffffffee14 Unclear: 0x68='h' score=999 | 0 (second best: 0x01 score=687) |
| Reading at malicious_x = 0xffffffffffffffee15 Unclear: 0x20=' ' score=998 | 3 (second best: 0x01 score=750) |
| Reading at malicious_x = 0xffffffffffffffee16 Unclear: 0x4F='0' score=991 | l (second best: 0x01 score=725) |
| Reading at malicious_x = 0xffffffffffffffee17 Unclear: 0x73='s' score=998 | 3 (second best: 0x01 score=734) |
| Reading at malicious_x = 0xffffffffffffffee18 Unclear: 0x73='s' score=999 | 0 (second best: 0x01 score=753) |
| Reading at malicious_x = 0xffffffffffffffee19 Unclear: 0x69='i' score=996 | 5 (second best: 0x01 score=761) |
| Reading at malicious_x = 0xffffffffffffffee1a Unclear: 0x66='f' score=995 | 5 (second best: 0x01 score=743) |
| Reading at malicious_x = 0xffffffffffffffee1b Unclear: 0x72='r' score=996 | 6 (second best: 0x01 score=726) |
| Reading at malicious_x = 0xffffffffffffffee1c Unclear: 0x61='a' score=979 | 0 (second best: 0x01 score=733) |
| Reading at malicious_x = 0xffffffffffffffee1d Unclear: 0x67='g' score=997 | <pre>/ (second best: 0x01 score=723)</pre> |
| Reading at malicious_x = 0xffffffffffffffee1e Unclear: 0x65='e' score=989 | 0 (second best: 0x01 score=750) |
| Reading at malicious x = 0xffffffffffffffee1f Unclear: 0x2E='.' score=971 | l (second best: 0x01 score=696) |

Windows Subsystem for Linux, Windows 10 (1809), gcc 7.3, Intel i7-7500U

22

How bad is this?

Different browser tabs should obviously not run in the same address space!

Is that good enough?

Can I read the operating system's memory?

Can I read other processes' memory?

Side-channels in speculative execution

- Suppose the bounds check "if" is predicted satisfied
- But i is out of bounds
- So *p points to a victim web page's secret s (like the paypal password I just entered)
- So we can speculatively use s as an index into an array that we do have access to
- And then using timing to determine whether the cache line on which B[s] falls has been allocated as a side-effect of speculative execution

Student question

This is Spectre Variant #1



¹⁵ "I just wanted to check if my understanding ²⁵ was correct on how we access the data in the secret address

- We assign an out of bound index that takes *p (and therefore s) to the secret place
- Execution happens because of speculation "branch taken" and therefore within the commit queues we have the message in S now but we can't read it because there was no commit
- To "read it", we do that bit by bit, through accessing some cache data. We know both rows X and X+1 are not in the cache, and try to call one of them through indexing in array B by using a bit of S
- Even though we are in speculative execution still, out-of-order will issue the memory call to the cache and queue it in the LSQ without being written to R.
- But we don't care, because that cache now will have either retrieved X or X+1 line. We determine that by classic probing / timing analysis for valid cache access later in the code and depending on the line that was already cached by the speculative execution of $r = (B[16^*(s\&1)])$; we conclude if that bit of interest in the secret message was 1 or 0
- If the above is correct, we are therefore assuming that branch correction for the speculation will NOT occur before the cache request through r = (B[16*(s&1)]);"







On completion, result is broadcast on CDB with tag that was assigned when it was issued



On completion, result is broadcast on CDB with tag that was assigned when it was issued



On completion, result is broadcast on CDB with tag that was assigned when it was issued



- Load unit initiates load from L1D cache
- Indexes L1D\$ data and tag
- Looks up virtual page number in DTLB
- If tag matches translation, data is forwarded to CDB
- If tag match fails, initiates L2 access



Student question³⁴

Q: could you explain what the operations on the s variable do when using it as an index (r=B[**16*(s&1)**])?

re: "r=B[16*(s&1)])"

s&1 does a Boolean "and" with the bits of a, and the single one-bit "1".

So we get either a zero (if s was even) or one (if s was odd).

I multiplied by 16 to hit a different cache line (supposing that the cache line size is 16).

I chose this one-bit idea so we could talk about just two cache lines (on reflection, maybe it didn't simplify things!).

What happens in the spectre.c code is

```
s = array1[x]
```

```
r = array2[s * 512]
```

where array1 is a char array so array1[x] is an 8-bit value. Thus we ensure that whatever the value of array1[x], the access to array2 hits a distinct cache line.

Student question³⁵

Q: "If so I don't understand why you use this value for an index to another array? Surely you already have the data you need and don't need to probe the cache?"

The interesting case starts with this:

- 1: if (p is in bounds)
- 2: s = *p
- 3: else
- 4: throw bounds error exception
- 5: print s

If p is indeed in bounds, we get to print s - but sadly s isn't a secret, since p was in-bounds.

If p is not in-bounds, we (might) speculatively execute the load instruction to fetch *p, but we discover the branch misprediction and roll back - so we can't print s.

So here's the trick: we do something with s, while we are still on the speculative path, that betrays the secret. Like using the value of s to allocate a cache line. This is what the code on the slide does:

```
1: if (p is in bounds)
```

2: s = *p

```
3: r=B[16*(s&1)]
```

- 4: else
- 5: throw bounds error exception

```
6: print s, r
```

Now, when we speculatively execute line 2, in the out-of-bounds case, s is a secret.

And line 3 results in a load instruction to one of two addresses: B[0] or B[16].

The misprediction is detected as before, at some later point (eg line 6). We roll back, so we can't print s or r. But the cache allocation due to line 3 is still there.

So now we can do a timing analysis to (probably) discover whether B[0] or B[16] was allocated.