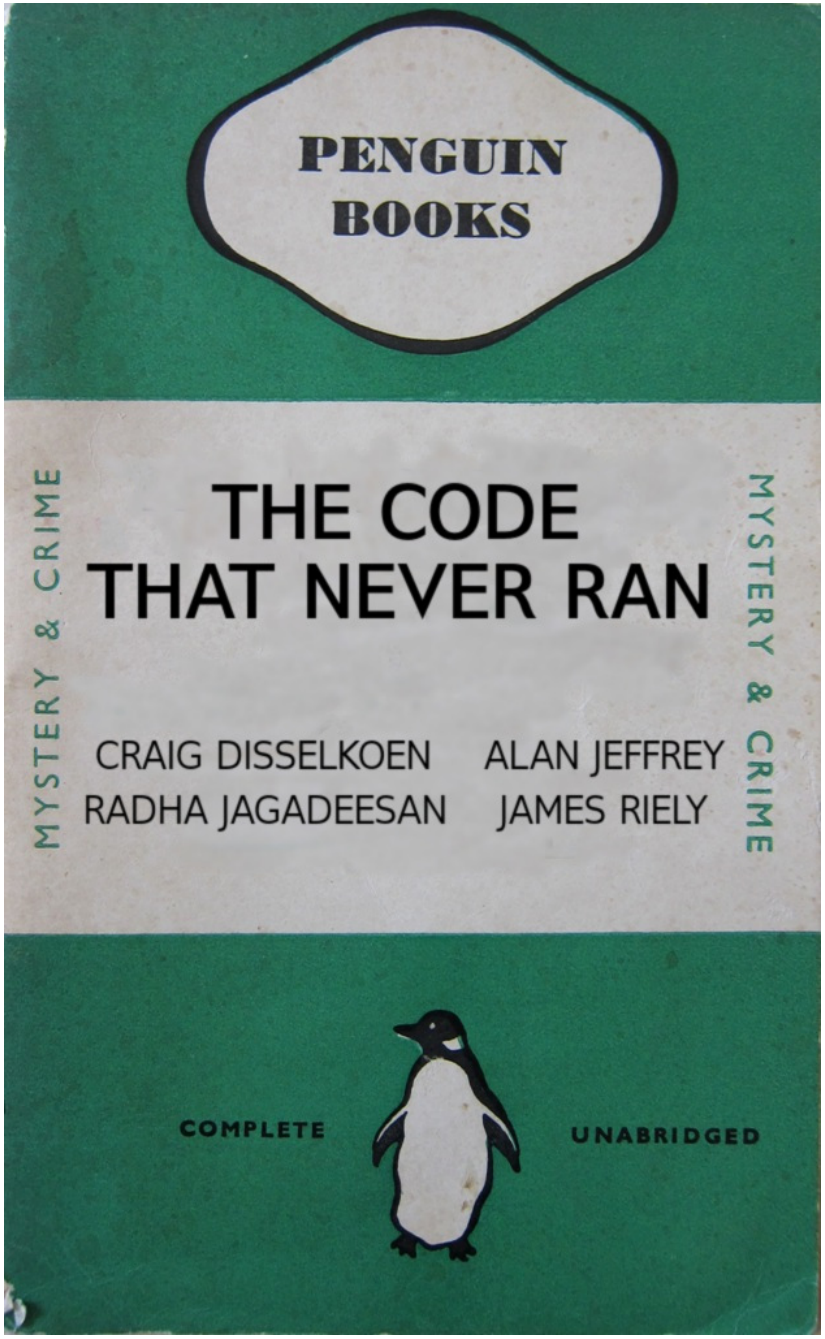


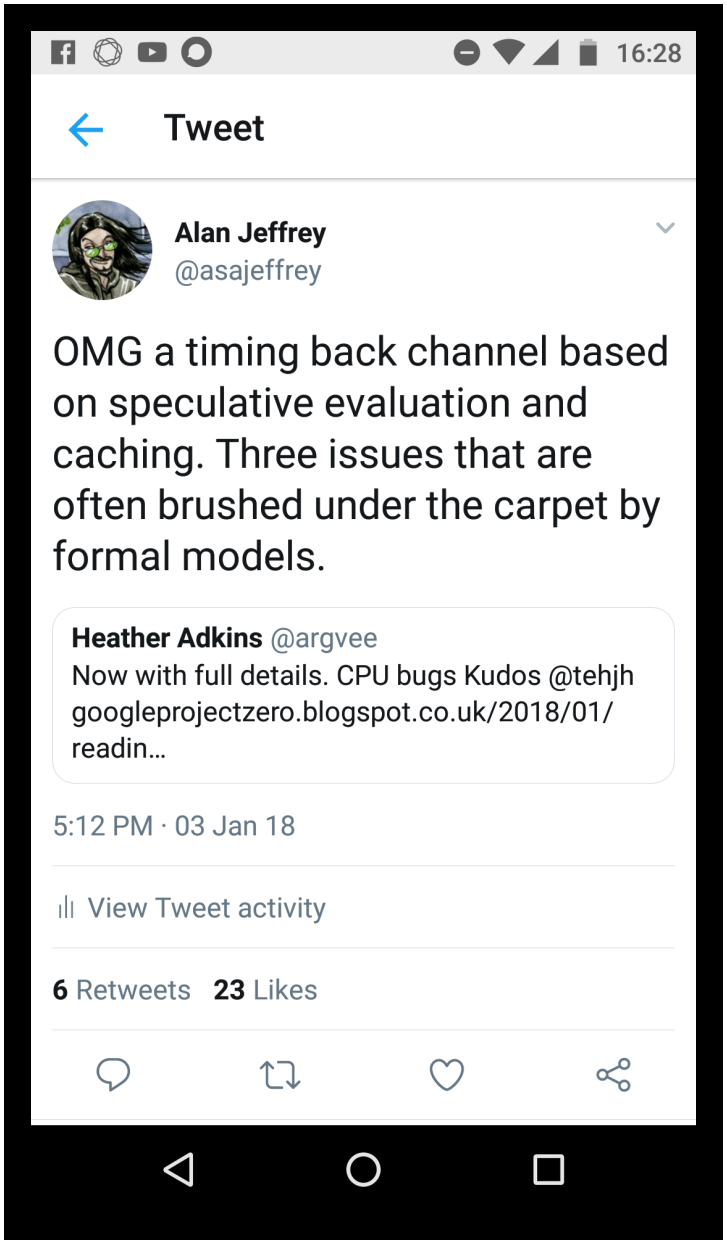
A classic locked-room mystery.
Eve was in the false branch of a conditional the whole time,
how could she do it?

Mozilla Research | DePaul University | U. California San Diego

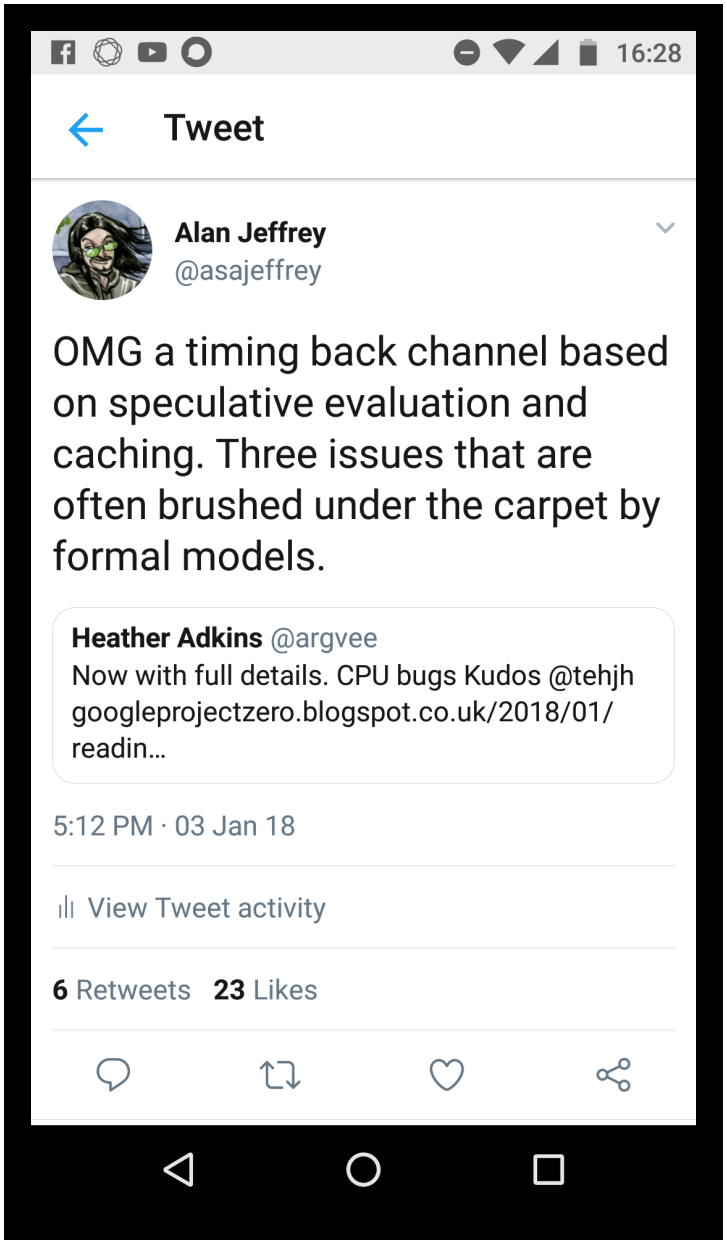
Overview

Introduction
Model
Attacks
Experiments
Conclusions

Why? Spectre!



Why? Spectre!



Attacks bypass dynamic security checks:

```
if (canReadSecret) {
  doStuffWith(SECRET);
}
```

Information flow from `SECRET` even though `canReadSecret` is false.

Most formal models ignore code in branches that aren't taken.

Models that include speculation?

There are some models that include speculation *relaxed memory models*:

- *The Java Memory Model*
  Manson, Pugh and Adve, 2005.
- *Generative Operational Semantics for Relaxed Memory Models*
  Jagadeesan, Pitcher and Riely, 2010.
- *A promising semantics for relaxed-memory concurrency*
  Kang, Hur, Lahav, Vafeiadis and Dreyer, 2017.

Models that include speculation?

There are some models that include speculation *relaxed memory models*:

- *The Java Memory Model*
  Manson, Pugh and Adve, 2005.
- *Generative Operational Semantics for Relaxed Memory Models*
  Jagadeesan, Pitcher and Riely, 2010.
- *A promising semantics for relaxed-memory concurrency*
  Kang, Hur, Lahav, Vafeiadis and Dreyer, 2017.

Question: is there a simple model similar to those of relaxed memory, that can model speculation?

Information flow attacks on speculation

Speculation happens in many places:

- *Speculation in hardware* (branch prediction,. . .)
- *Transactions* (transactional memory,. . .)
- *Relaxed memory* (compiler optimizations,. . .)

Information flow attacks on speculation

Speculation happens in many places:

- *Speculation in hardware* (branch prediction,. . .)
  Attacked by Spectre (Kocher *et al.* 2019).
- *Transactions* (transactional memory,. . .)
  Attacked by Prime+Abort (Disselokoen *et al.* 2017).
- *Relaxed memory* (compiler optimizations,. . .)
  No known attacks.

Question: are there information flow attacks against compiler optimizations?

Contributions

- A simple compositional model.
- Examples.
- Attacks (including a new attack on relaxed memory).
- Experiments (testing practicality of new attacks).

Pomsets

C11-style models are based on *events*
with *labels* (e.g. $(\mathrm{R}\, x\, 3)$ or $(\mathrm{W}\, x\, 3)$)
and *relations* (e.g. happens-before or reads-from).

Pomsets

C11-style models are based on *events*
with *labels* (e.g. $(\mathrm{R}\, x\, 3)$ or $(\mathrm{W}\, x\, 3)$)
and *relations* (e.g. happens-before or reads-from).

Simplest such is *partially ordered multisets* (Gisher, 1988).

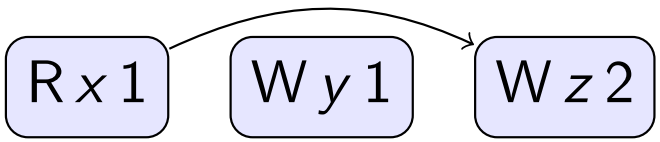
Only one relation, a partial order modelling dependency

Pomsets

C11-style models are based on *events*
with *labels* (e.g. $(\mathrm{R}\,x\,3)$ or $(\mathrm{W}\,x\,3)$)
and *relations* (e.g. happens-before or reads-from).

Simplest such is *partially ordered multisets* (Gisher, 1988).

Only one relation, a partial order modelling dependency, e.g.



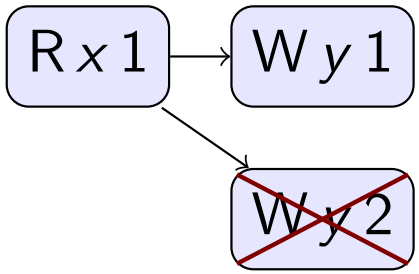
is an execution of $(r := x\,;\; y := 1\,;\; z := r + 1)$.

Pomsets

C11-style models are based on *events*
with *labels* (e.g. (R x 3) or (W x 3))
and *relations* (e.g. happens-before or reads-from).

Simplest such is *partially ordered multisets* (Gisher, 1988).

Only one relation, a partial order modelling dependency, e.g.



is an execution of (`if` $(x)\,\{\,y := 1\,\}$ `else` $\{\,y := 2\,\}$).

Compositional pomset model

First off, straight-line code.

Compositional pomset model

First off, straight-line code.

New idea: put preconditions on events

Compositional pomset model

First off, straight-line code.

New idea: put preconditions on events, e.g.

$$r = 1 \mid \mathrm{W}\, z\, 2$$

is an execution of ($z := r + 1$).

Compositional pomset model

First off, straight-line code.

New idea: put preconditions on events, e.g.

$$\boxed{\mathrm{W}\, y\, 1} \quad \boxed{r = 1 \mid \mathrm{W}\, z\, 2}$$

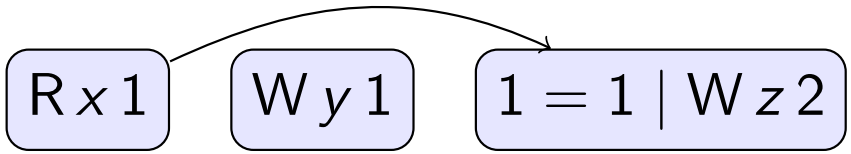
is an execution of ($y := 1 ; z := r + 1$).

Note: no dependency because r does not depend on $y := 1$.

Compositional pomset model

First off, straight-line code.

New idea: put preconditions on events, e.g.



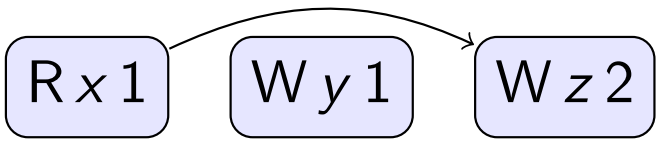
is an execution of $(r := x\,;\ y := 1\,;\ z := r + 1)$.

Note: dependency because r depends on $r := x$.
Also note: performing a substitution $[1/r]$.

Compositional pomset model

First off, straight-line code.

New idea: put preconditions on events, e.g.



is an execution of $(r := x\,;\ y := 1\,;\ z := r + 1)$.

Visualize: elide tautologies

Compositional pomset model

Next, conditionals.

Compositional pomset model

Next, conditionals.

New idea: an execution of `if` $M \{ C \}$ `else` $\{ D \}$ comes from an execution of C *and* an execution of D

Compositional pomset model

Next, conditionals.

New idea: an execution of $\texttt{if}\ M\ \{\ C\ \}\ \texttt{else}\ \{\ D\ \}$ comes from an execution of C *and* an execution of D, e.g.

$$r \neq 0 \mid \mathrm{W}\, y\, 1$$

is an execution of ($y := 1$)
when $r \neq 0$

Compositional pomset model

Next, conditionals.

New idea: an execution of $\texttt{if}\ M\ \{\ C\ \}\ \texttt{else}\ \{\ D\ \}$
comes from an execution of C *and* an execution of D, e.g.

$$r = 0 \mid \mathrm{W}\, y\, 2$$

is an execution of ($y := 2$)
when $r = 0$

Compositional pomset model

Next, conditionals.

New idea: an execution of $\texttt{if}\ M\,\{\,C\,\}\,\texttt{else}\,\{\,D\,\}$ comes from an execution of C *and* an execution of D, e.g.

$$r \neq 0 \mid \mathrm{W}\, y\, 1$$

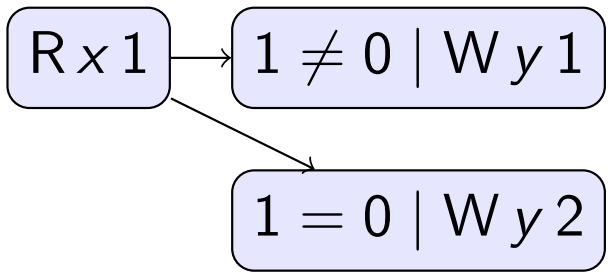
$$r = 0 \mid \mathrm{W}\, y\, 2$$

is an execution of ($\texttt{if}\,(r)\,\{\,y := 1\,\}\,\texttt{else}\,\{\,y := 2\,\}$)

Compositional pomset model

Next, conditionals.

New idea: an execution of $\texttt{if}\ M\,\{\,C\,\}\,\texttt{else}\,\{\,D\,\}$ comes from an execution of C *and* an execution of D, e.g.

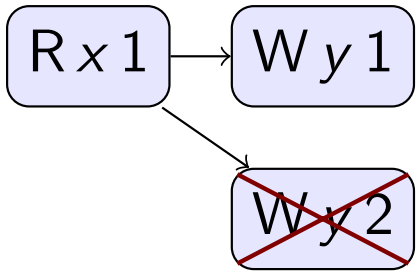


is an execution of $(r := x\,;\ \texttt{if}\,(r)\,\{\,y := 1\,\}\,\texttt{else}\,\{\,y := 2\,\})$

Compositional pomset model

Next, conditionals.

New idea: an execution of $\texttt{if}\ M\,\{\,C\,\}\,\texttt{else}\,\{\,D\,\}$ comes from an execution of C *and* an execution of D, e.g.



is an execution of $(r := x\,;\ \texttt{if}\,(r)\,\{\,y := 1\,\}\,\texttt{else}\,\{\,y := 2\,\})$

Visualize: elide tautologies and cross out unsatisfiables

Compositional pomset model

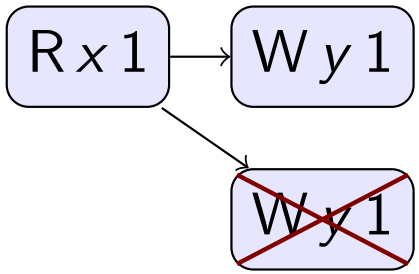
But. . .

Compositional pomset model

But. . . any execution of C should be an execution of $\texttt{if}\ M \{ C \} \texttt{else} \{ C \}$

Compositional pomset model

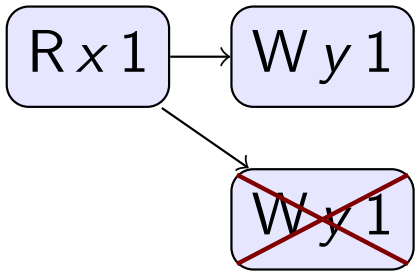
But. . . any execution of C should be an execution of $\texttt{if}\ M\ \{\ C\ \}\ \texttt{else}\ \{\ C\ \}$, e.g.



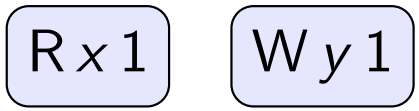
is an execution of $(\texttt{if}\ x\ \{\ y := 1\ \}\ \texttt{else}\ \{\ y := 1\ \})$

Compositional pomset model

But. . . any execution of C should be
an execution of $\texttt{if}\ M\ \{\ C\ \}\ \texttt{else}\ \{\ C\ \}$, e.g.

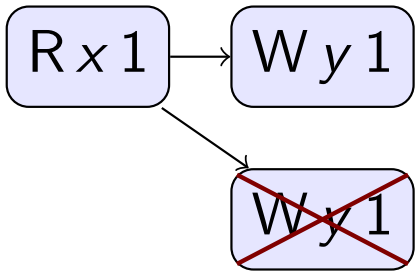


is an execution of $(\texttt{if}\ x\ \{\ y := 1\ \}\ \texttt{else}\ \{\ y := 1\ \})$, but so is

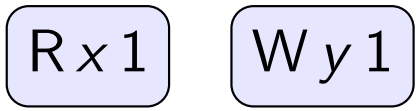


Compositional pomset model

But. . . any execution of C should be
an execution of $\texttt{if}\ M\ \{\ C\ \}\ \texttt{else}\ \{\ C\ \}$, e.g.



is an execution of $(\texttt{if}\ x\ \{\ y := 1\ \}\ \texttt{else}\ \{\ y := 1\ \})$, but so is



New idea: events from different branches can merge.

Compositional pomset model

Lastly, concurrency.

Compositional pomset model

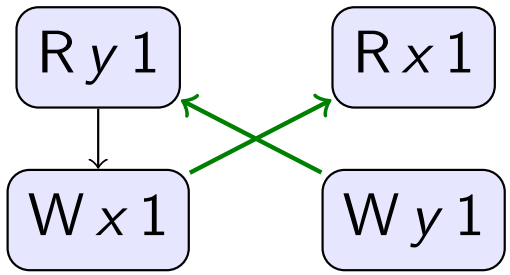
Lastly, concurrency.

Old idea: match reads with matching writes (à la C11)

Compositional pomset model

Lastly, concurrency.

Old idea: match reads with matching writes (à la C11), e.g.



is an execution of $(x := y \;||\; r := x;\; y := 1)$.

Compositional pomset model

Glossed over some details:

- 3-valued pomsets for negative constraints $d \not< e$,
- sanity conditions on reads-from,
- precise rules for dependency,
- variable declaration,
- ...

All in the paper!

Information flow example

Imagine a SECRET, protected by a run-time security check:

$$\texttt{if}\ \texttt{canRead(SECRET)}\ \{\ldots\ \text{use}\ \texttt{SECRET}\ldots\}\ \texttt{else}\ \{\ldots\}$$

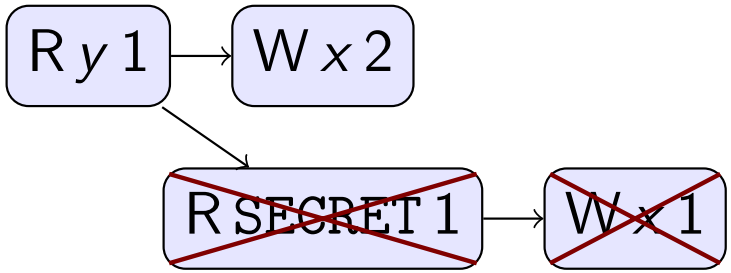
For attacker code canRead(SECRET) is always false

Information flow example

Imagine a SECRET, protected by a run-time security check:

$$\texttt{if canRead(SECRET)}\,\{\ldots\text{use SECRET}\ldots\}\,\texttt{else}\,\{\ldots\}$$

For attacker code canRead(SECRET) is always false, e.g.



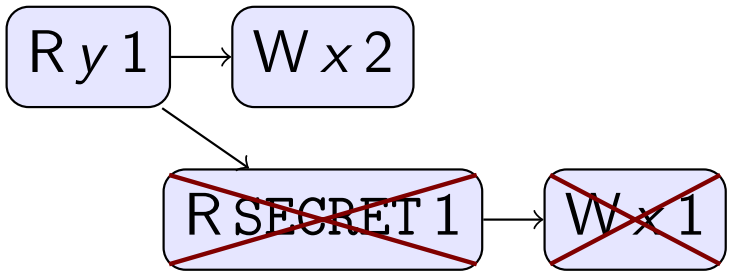
is an execution of
$\texttt{if}\ y\,\{\,\texttt{if canRead(SECRET)}\,\{\,x\texttt{:=SECRET}\,\}\,\texttt{else}\,\{\,x\texttt{:=1}\,\}\,\}$.

Information flow example

Imagine a SECRET, protected by a run-time security check:

$$\texttt{if canRead(SECRET)}\,\{\ldots\text{use SECRET}\ldots\}\,\texttt{else}\,\{\ldots\}$$

For attacker code canRead(SECRET) is always false, e.g.



is an execution of
$\texttt{if}\ y\,\{\,\texttt{if canRead(SECRET)}\,\{\,x\texttt{:=SECRET}\,\}\,\texttt{else}\,\{\,x\texttt{:=1}\,\}\,\}$.

Attacker goal: learn if SECRET is 0 or 1.

Modeling Spectre attack

Spectre uses cache timing to discover if a memory location has been touched.

Modeling Spectre attack

Spectre uses cache timing to discover if a memory location has been touched.

Glossing over a lot of details, this is

$$\texttt{if}\ \texttt{touched}\,(x)\,\{\cdots\}\,\texttt{else}\,\{\cdots\}$$

Modeling Spectre attack

Spectre uses cache timing to discover if a memory location has been touched.

Glossing over a lot of details, this is

$$\texttt{if}\ \texttt{touched}\,(x)\,\{\cdots\}\,\texttt{else}\,\{\cdots\}$$

Modeled with a new action $(\mathsf{T}\,x)$

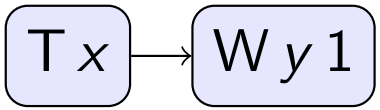
Modeling Spectre attack

Spectre uses cache timing to discover if a memory location has been touched.

Glossing over a lot of details, this is

$$\texttt{if touched}\,(x)\,\{\cdots\}\,\texttt{else}\,\{\cdots\}$$

Modeled with a new action $(\mathsf{T}\,x)$, e.g.



is an execution of $\texttt{if touched}\,(x)\,\{\,y := 1\,\}$.

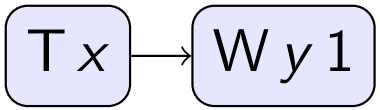
Modeling Spectre attack

Spectre uses cache timing to discover if a memory location has been touched.

Glossing over a lot of details, this is

$$\texttt{if touched}\,(x)\,\{\cdots\}\,\texttt{else}\,\{\cdots\}$$

Modeled with a new action $(\mathsf{T}\,x)$, e.g.



is an execution of $\texttt{if touched}\,(x)\,\{\,y := 1\,\}$.

Require that any event labelled $(\mathsf{T}\,x)$ must be preceded by an event labelled $(\mathsf{R}\,x\,v)$ or $(\mathsf{W}\,x\,v)$.

Modeling Spectre attack

A very simplified Spectre attack:

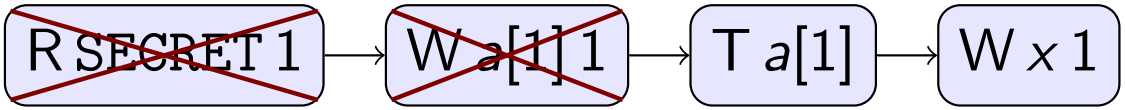
$$\texttt{if canRead(SECRET)}\,\{\, a[\texttt{SECRET}] := 1 \,\}$$
$$\texttt{else if touched}\,(a[0])\,\{\, x := 0 \,\}$$
$$\texttt{else if touched}\,(a[1])\,\{\, x := 1 \,\}$$

Modeling Spectre attack

A very simplified Spectre attack:

$$\texttt{if canRead(SECRET)}\,\{\, a[\texttt{SECRET}] := 1 \,\}$$
$$\texttt{else if touched}\,(a[0])\,\{\, x := 0 \,\}$$
$$\texttt{else if touched}\,(a[1])\,\{\, x := 1 \,\}$$

e.g. with execution



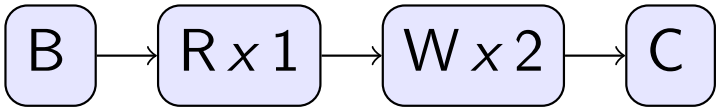
Information flow from SECRET to x.

Modeling Prime+Abort attack

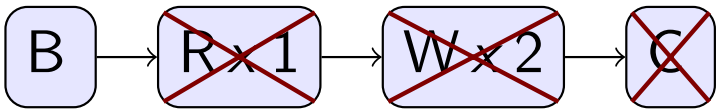
Prime+Abort is an information flow attsck on Intel's transactional memory. So first model transactions

Modeling Prime+Abort attack

Prime+Abort is an information flow attsck on Intel's transactional memory. So first model transactions, e.g.



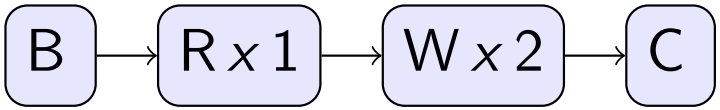
and



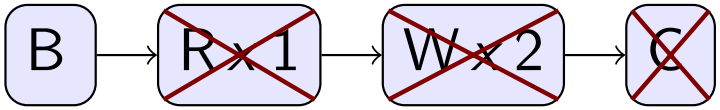
are executions of `begin`; $x := x + 1$; `end`

Modeling Prime+Abort attack

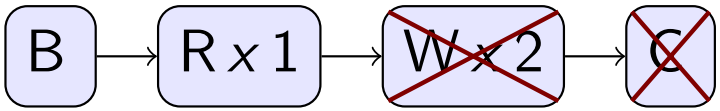
Prime+Abort is an information flow attsck on Intel's transactional memory. So first model transactions, e.g.



and



are executions of `begin`; $x := x + 1$; `end`, but *not*



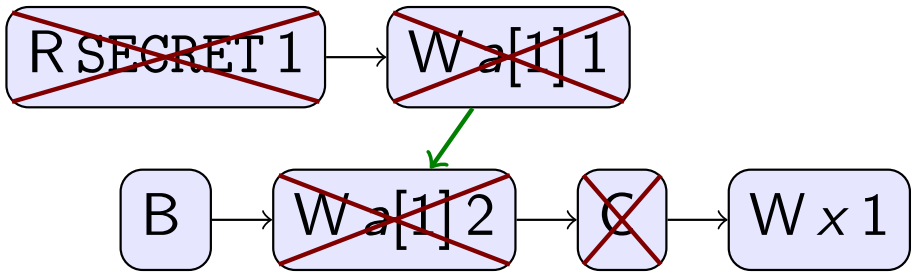
Modeling Prime+Abort attack

Transactions are fine, but not if we add a reason for an abort.

If the attacker knows that every aborted transaction does so because of a read/write or write/write conflict, then in

$$\texttt{if canRead(SECRET)}\,\{\,a[\texttt{SECRET}] := 1\,\}\;||$$
$$\texttt{begin}; a[1] := 2; \texttt{loop}; \texttt{end}; x := 1$$

the transaction aborts only when SECRET is 1.



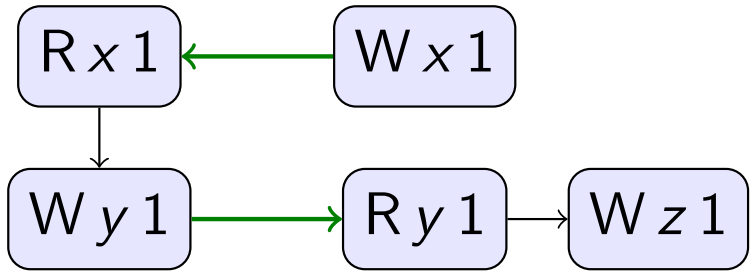
Information flow from SECRET to x.

New store reordering attack

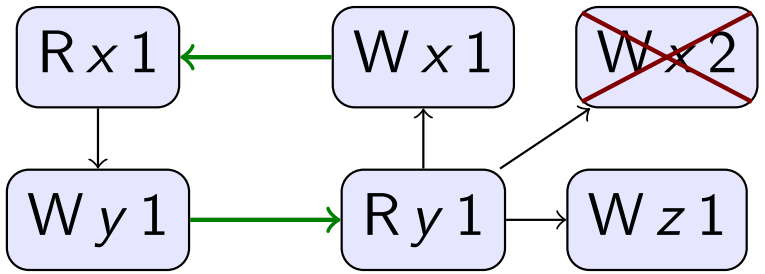
An attack on relaxed memory, *discovered from this model*.

$$y := x \;||\; \texttt{if}\,(y == 0)\,\{\,x := 1\,\}$$
$$\texttt{else if}\,(\texttt{canRead}(\texttt{SECRET}))\,\{\,x := \texttt{SECRET}\,\}$$
$$\texttt{else}\,\{\,x := 1;\, z := 1\,\}$$

If SECRET is 1, there is an execution:



If SECRET is 2, there is no execution:

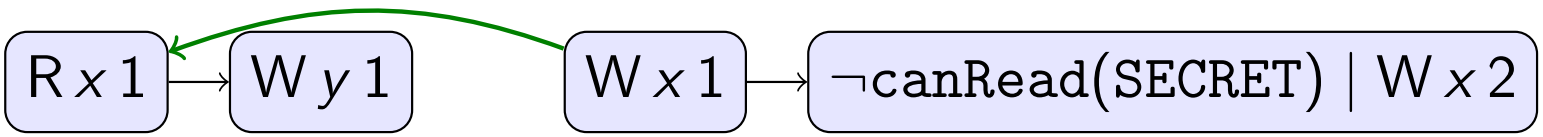


New dead store elimination attack

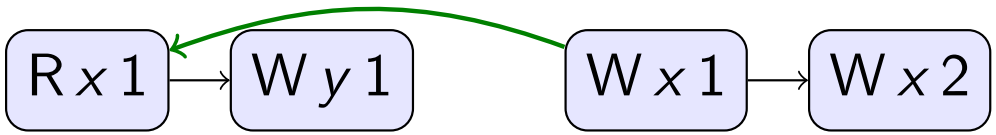
Another attack *discovered from this model*.

$y := x \;||\; x := 1;$
`if (canRead(SECRET)) { if (SECRET) { x := 2 } }`
`else { x := 2 }`

If `SECRET` is 0, there is an execution:



If `SECRET` is 1,
there is an execution:

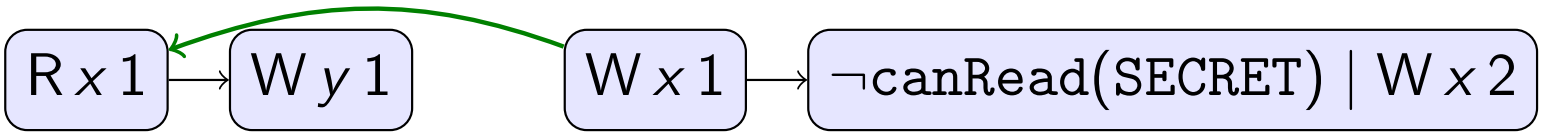


New dead store elimination attack

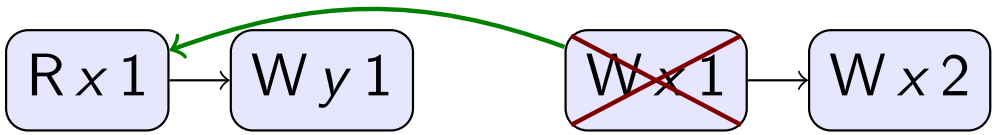
Another attack *discovered from this model*.

$y := x \;||\; x := 1;$
`if (canRead(SECRET)) { if (SECRET) { x := 2 } }`
`else { x := 2 }`

If `SECRET` is 0, there is an execution:



If `SECRET` is 1, and dead store elimination is performed, there is *no* execution:



Implementing the new attacks

Outro goes here