

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

 Creative Commons Attribution-ShareAlike 4.0  
Mozilla Research | DePaul University | U. California San Diego

# 3 January 2018

## Had a day out at the Tate Modern

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

3 January 2018



## The Code That Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

## Humanizing Anecdote

## Spectre

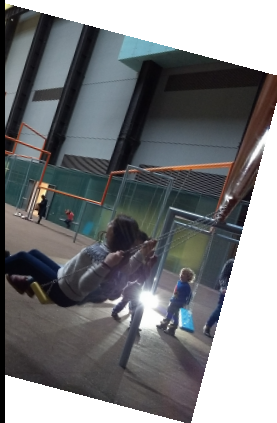
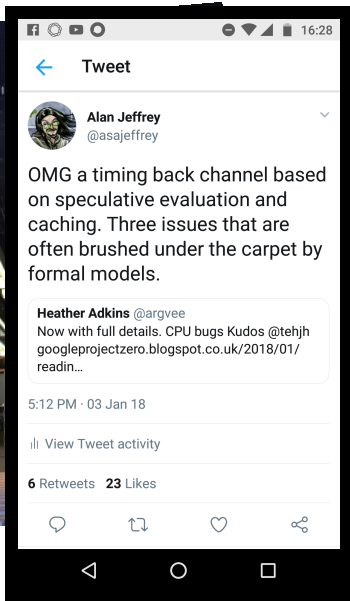
## Optimizations

## Simplified Spectre

TODO

A set of navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

# 3 January 2018



The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

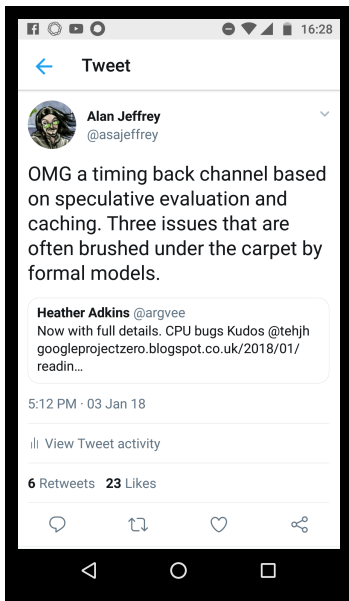
Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO



Attacks bypass run-time security checks.

Can bypass array bounds checks, and read whole process memory.

Can be exploited from JS, so evil.ad.com can read your bank.com data.

Attacks  
*speculative evaluation*  
hardware optimization.

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# Optimizations in hardware

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

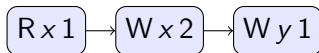
TODD

A lie we tell programmers:

“computers execute instructions one after the other.”

$$x := x + 1; y := 1$$

has execution:



# Optimizations in hardware

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

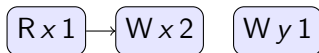
TODD

A lie we tell programmers:

“computers execute instructions one after the other.”

$$x := x + 1; y := 1$$

has execution:



The  $W y 1$  might happen first.



# Optimizations in hardware

The Code That  
Never Ran

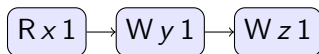
Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Another lie we tell programmers:

“only one branch of an `if` is executed.”

```
if (x) { y:=1; z:=1 } else { y:=2; z:=1 }
```

has execution:



Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# Optimizations in hardware

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

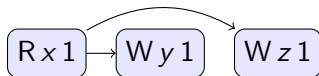
TODO

Another lie we tell programmers:

“only one branch of an `if` is executed.”

```
if (x) { y:=1; z:=1 } else { y:=2; z:=1 }
```

has execution:



`Wz1` might happen before `Wy1`.

# Optimizations in hardware

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

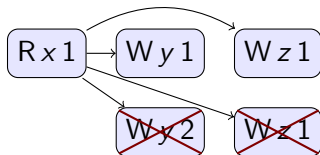
TODO

Another lie we tell programmers:

“only one branch of an `if` is executed.”

```
if (x) { y:=1; z:=1 } else { y:=2; z:=1 }
```

has execution:



`W y 2` might happen, then get rolled back.

# Optimizations in hardware and compilers

The Code That  
Never Ran

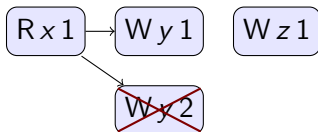
Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Another lie we tell programmers:

“only one branch of an `if` is executed.”

```
if (x) { y:=1; z:=1 } else { y:=2; z:=1 }
```

has execution:



`W z 1` might happen first.

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# Simplified Spectre

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

```
if canRead(SECRET) { ... use SECRET ... } else { ... }
```

For attacker code `canRead(SECRET)` is always false

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# Simplified Spectre

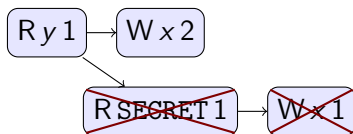
The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

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

```
if canRead(SECRET) { ... use SECRET ... } else { ... }
```

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



is an execution of

```
if y { if canRead(SECRET) { x := SECRET } else { x := 2 } }.
```

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# Simplified Spectre

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

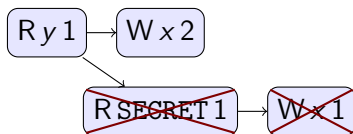
Simplified Spectre

TODO

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

```
if canRead(SECRET) { ... use SECRET ... } else { ... }
```

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



is an execution of

```
if y { if canRead(SECRET) { x := SECRET } else { x := 2 } }.
```

Attacker goal: learn if SECRET is 0 or 1.

# Simplified Spectre

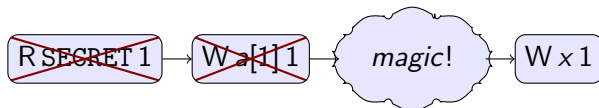
The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

A very simplified Spectre attack:

```
if canRead(SECRET) { a[SECRET] := 1 }  
else if touched(a[0]) { x := 0 }  
else if touched(a[1]) { x := 1 }
```

with execution



Information flow from SECRET to x

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO



# Simplified Spectre

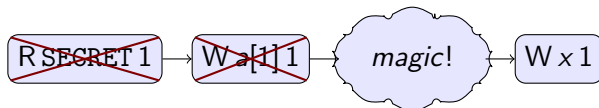
The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

A very simplified Spectre attack:

```
if canRead(SECRET) { a[SECRET] := 1 }  
else if touched(a[0]) { x := 0 }  
else if touched(a[1]) { x := 1 }
```

with execution



Information flow from SECRET to  $x$ ,  
*if* there's an implementation of “magic”.

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# Simplified Spectre

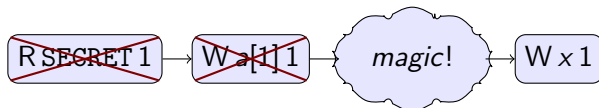
The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

A very simplified Spectre attack:

```
if canRead(SECRET) { a[SECRET] := 1 }  
else if touched(a[0]) { x := 0 }  
else if touched(a[1]) { x := 1 }
```

with execution



Information flow from SECRET to  $x$ ,  
*if* there's an implementation of “magic”.

*Narrator:* there was one.

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# TODO

Start building models, tools etc. which capture the attacks.

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# TODO

Start building models, tools etc. which capture the attacks.

Investigate similar attacks, e.g. on compiler optimizations.

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# TODO

Start building models, tools etc. which capture the attacks.

Investigate similar attacks, e.g. on compiler optimizations.

Make it harder to implement `if (touched(x))`  
(e.g. reduce access to high-precision timers).

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# TODO

Start building models, tools etc. which capture the attacks.

Investigate similar attacks, e.g. on compiler optimizations.

Make it harder to implement `if (touched(x))`  
(e.g. reduce access to high-precision timers).

Process isolation: make sure security boundaries  
line up with process boundaries.

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO

# TODO

The Code That  
Never Ran

Craig Disselkoen,  
Radha Jagadeesan,  
Alan Jeffrey,  
James Riely

Start building models, tools etc. which capture the attacks.

Investigate similar attacks, e.g. on compiler optimizations.

Make it harder to implement `if (touched(x))`  
(e.g. reduce access to high-precision timers).

Process isolation: make sure security boundaries  
line up with process boundaries.

Harden programs, compilers, etc.  
(difficult because it's a large attack surface).

Humanizing  
Anecdote

Spectre

Optimizations

Simplified Spectre

TODO