# Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript

**Michael Schwarz, Clémentine Maurice, Daniel Gruss, Stefan Mangard**
**Graz University of Technology**

April 2017 — FC 2017

# Outline

- Building a covert channel from a virtual machine without network access to the browser
- Reviving cache attacks in the browser
- No high-resolution timers in JavaScript
- How can we build our own timers?
- How to get a higher resolution than the native timers?

# Covert channel

## What is a covert channel?

- Two programs would like to communicate

# Covert channel

What is a covert channel?

- Two programs would like to communicate but are not allowed to do so

# Covert channel

## What is a covert channel?

- Two programs would like to communicate but are not allowed to do so
  - either because there is no communication channel...

# Covert channel

What is a covert channel?

- Two programs would like to communicate but are not allowed to do so
  - either because there is no communication channel...
  - ...or the channels are monitored and programs are stopped on communication attempts

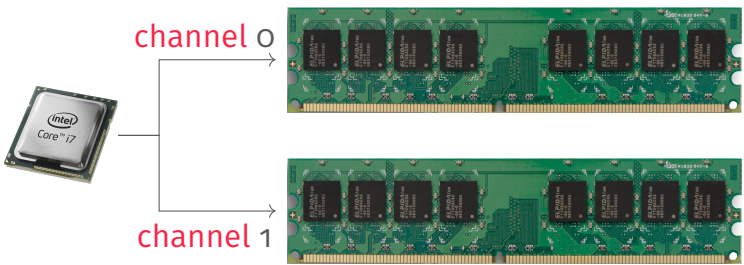
# Covert channel

What is a covert channel?

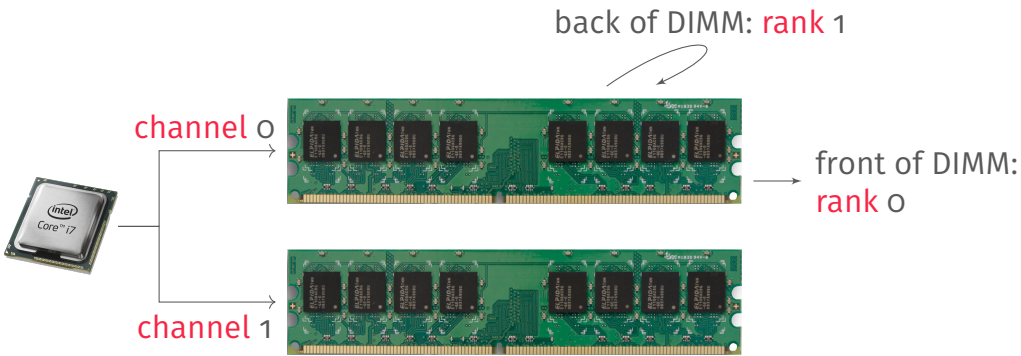
- Two programs would like to communicate but are not allowed to do so
  - either because there is no communication channel...
  - ...or the channels are monitored and programs are stopped on communication attempts
- Use side channels to communicate

# DRAM organization

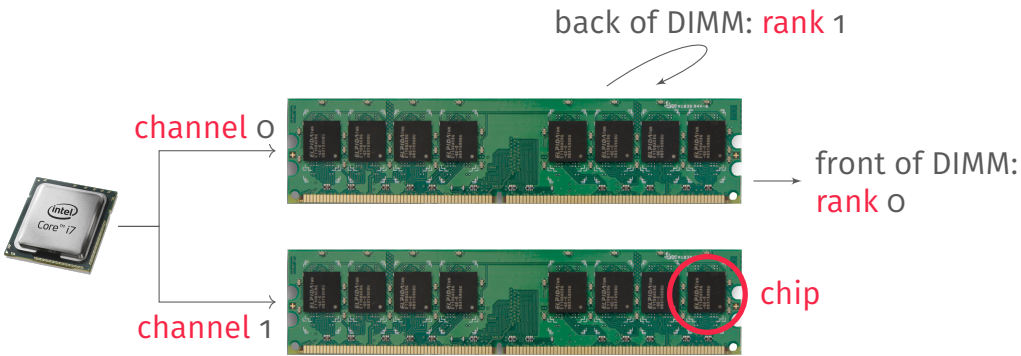
# DRAM organization



# DRAM organization

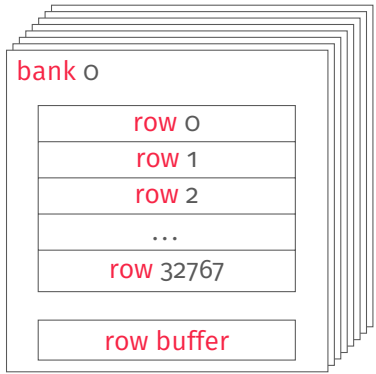


# DRAM organization



# DRAM organization

chip

# DRAM organization

chip


bank 0

row 0
row 1
row 2
…
row 32767

row buffer

64k cells
1 capacitor,
1 transitor each

# The row buffer

- DRAM internally is only capable of reading entire rows

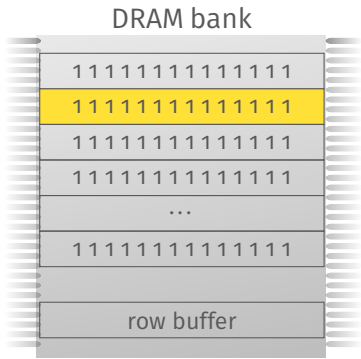
# The row buffer

- DRAM internally is only capable of reading entire rows
- Capacitors in cells discharge when reading them
- Bits are buffered when reading them from the cells
- Then, bits are written back to the cells again

# The row buffer

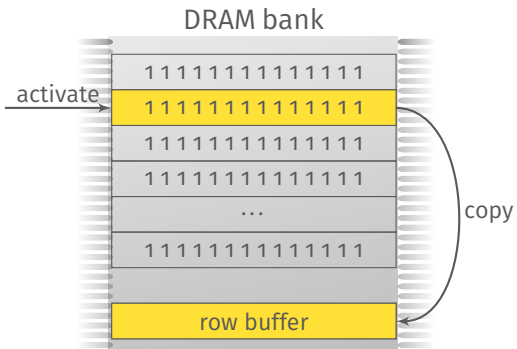
- DRAM internally is only capable of reading entire rows
- Capacitors in cells discharge when reading them
- Bits are buffered when reading them from the cells
- Then, bits are written back to the cells again
- $\rightarrow$ Row buffer

# Reading from DRAM

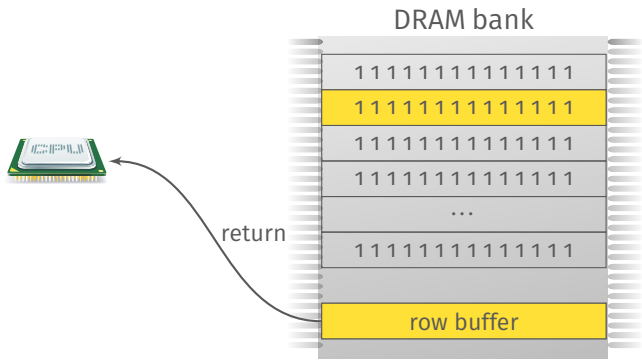




CPU reads row 1, row buffer empty!

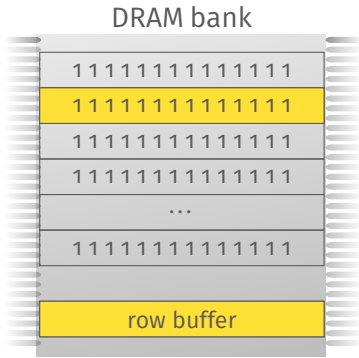
# Reading from DRAM

# Reading from DRAM

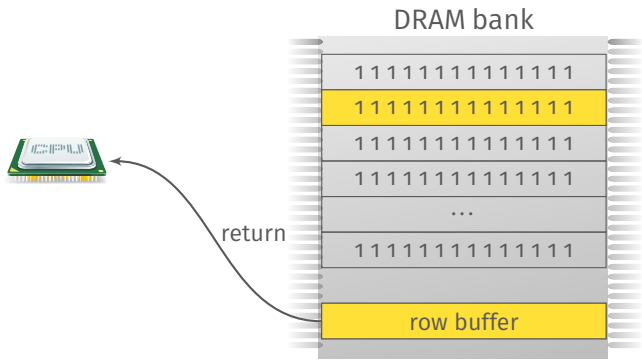


# Reading from DRAM

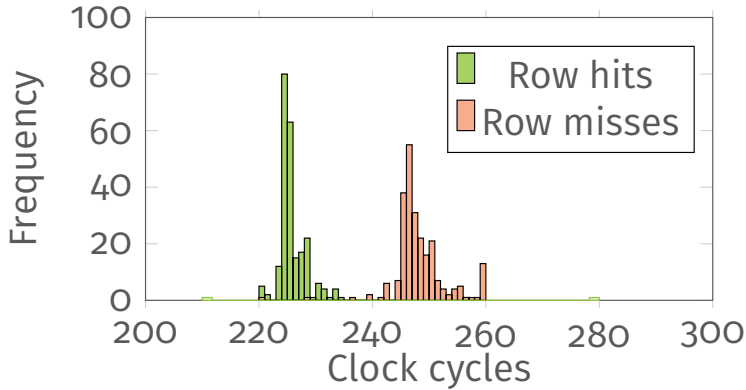

CPU reads row 1, row buffer now full!

# Reading from DRAM




Less work!
Is it faster?

# Timing difference




Row hits ($\approx$ 225 cycles) and row conflicts ($\approx$ 247 cycles)

# Timers in JavaScript

- We need a high-resolution timer to measure such small differences

# Timers in JavaScript

- We need a high-resolution timer to measure such small differences
- Native: `rdtsc` - timestamp in CPU cycles

# Timers in JavaScript

- We need a high-resolution timer to measure such small differences
- Native: `rdtsc` - timestamp in CPU cycles
- JavaScript: `performance.now()` has the highest resolution

# Timers in JavaScript

- We need a high-resolution timer to measure such small differences
- Native: `rdtsc` - timestamp in CPU cycles
- JavaScript: `performance.now()` has the highest resolution

## performance.now()

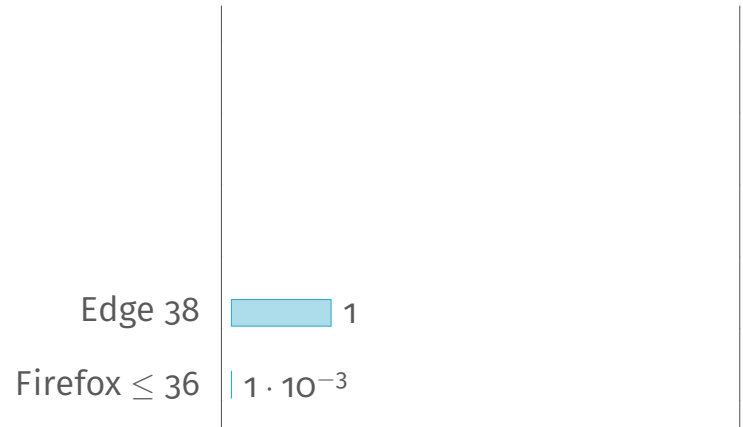
[...] represent times as floating-point numbers with up to microsecond precision.

— Mozilla Developer Network

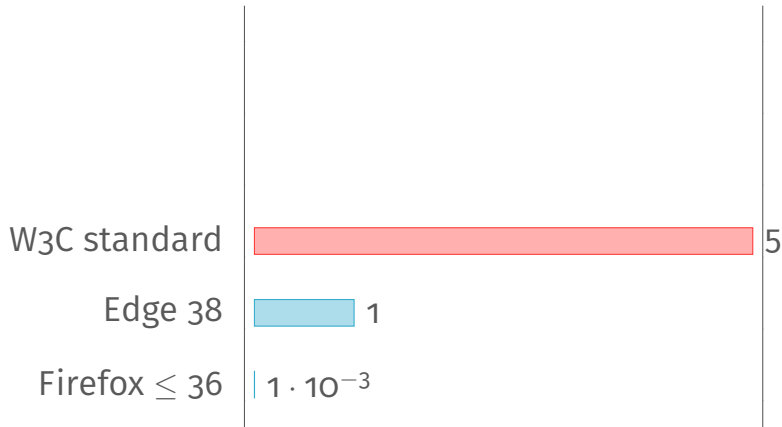
# ...up to microsecond precision?

| | |
|---|---|
| Firefox $\leq 36$ | $1 \cdot 10^{-3}$ |

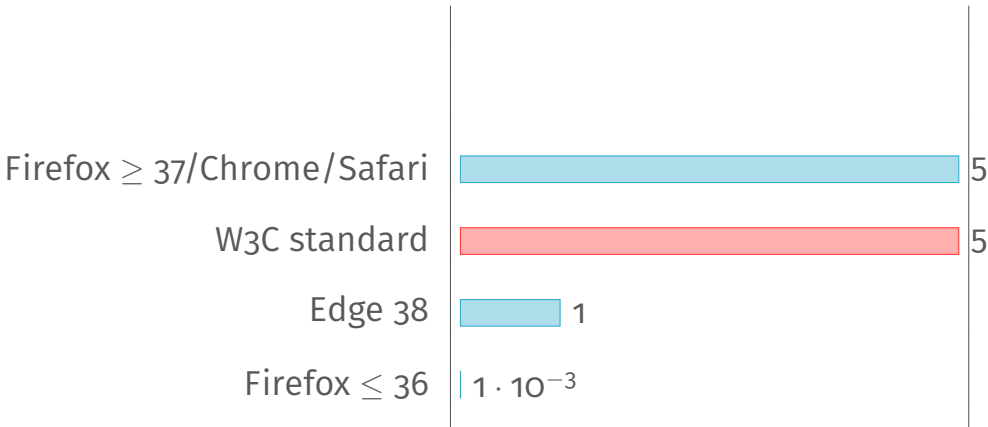
# ...up to microsecond precision?



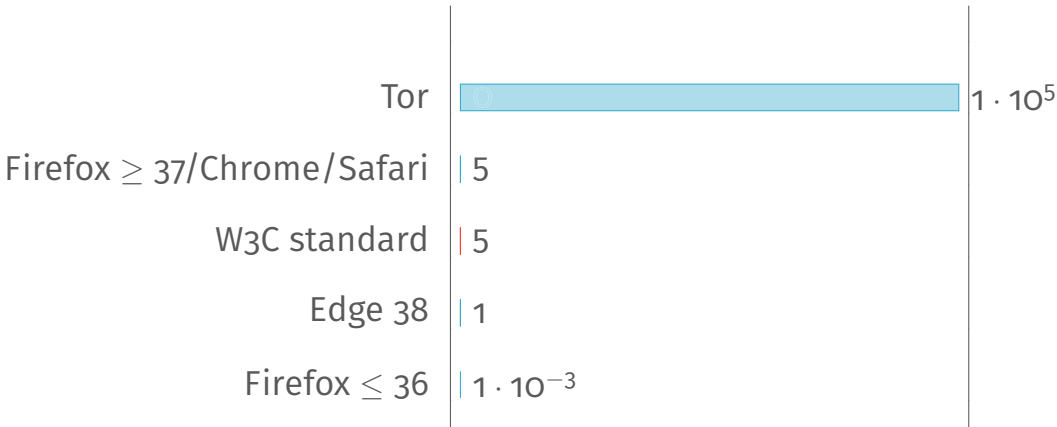
# ...up to microsecond precision?



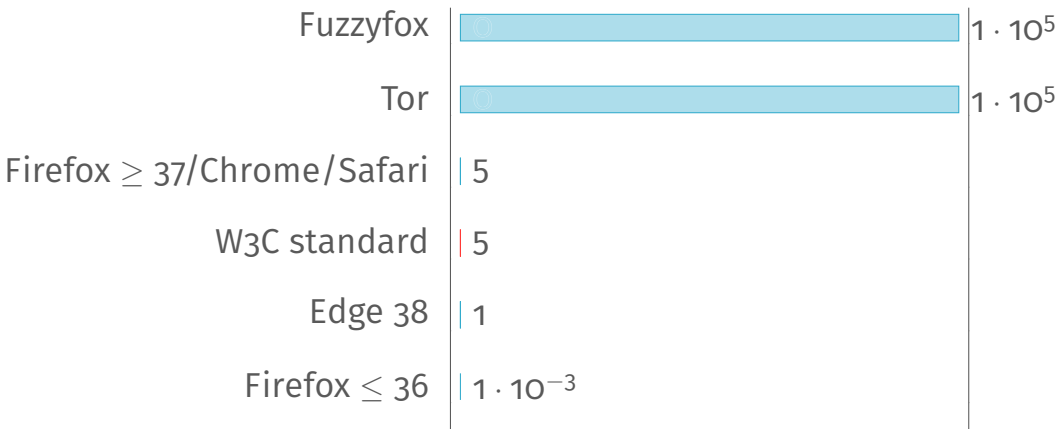
# ...up to microsecond precision?



# ...up to microsecond precision?



# ...up to microsecond precision?



# We require a higher resolution

- Current precision is not sufficient to measure cycle differences

# We require a higher resolution

- Current precision is not sufficient to measure cycle differences
- We have two possibilities

# We require a higher resolution

- Current precision is not sufficient to measure cycle differences
- We have two possibilities
- Recover a higher resolution from the available timer

# We require a higher resolution

- Current precision is not sufficient to measure cycle differences
- We have two possibilities
- Recover a higher resolution from the available timer
- Build our own high-resolution timer

# Recovering resolution - Clock interpolation

- Measure how often we can increment a variable between two timer ticks

# Recovering resolution - Clock interpolation

- Measure how often we can increment a variable between two timer ticks
- Average number of increments is the interpolation step

# Recovering resolution - Clock interpolation

- Measure how often we can increment a variable between two timer ticks
- Average number of increments is the interpolation step
- To measure with high resolution:

# Recovering resolution - Clock interpolation

- Measure how often we can increment a variable between two timer ticks
- Average number of increments is the interpolation step
- To measure with high resolution:
  - Start measurement at clock edge

# Recovering resolution - Clock interpolation

- Measure how often we can increment a variable between two timer ticks
- Average number of increments is the interpolation step
- To measure with high resolution:
  - Start measurement at clock edge
  - Increment a variable until next clock edge

# Recovering resolution - Clock interpolation

- Measure how often we can increment a variable between two timer ticks
- Average number of increments is the interpolation step
- To measure with high resolution:
  - Start measurement at clock edge
  - Increment a variable until next clock edge
- Highly accurate: 500 ns (Firefox/Chrome), 15 µs (Tor)

# Recovering resolution - Edge thresholding

- We can get a higher resolution for a classifier only

# Recovering resolution - Edge thresholding

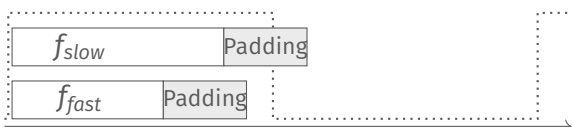
- We can get a higher resolution for a classifier only
- Often sufficient to see which of two functions takes longer

# Recovering resolution - Edge thresholding

- We can get a higher resolution for a classifier only
- Often sufficient to see which of two functions takes longer

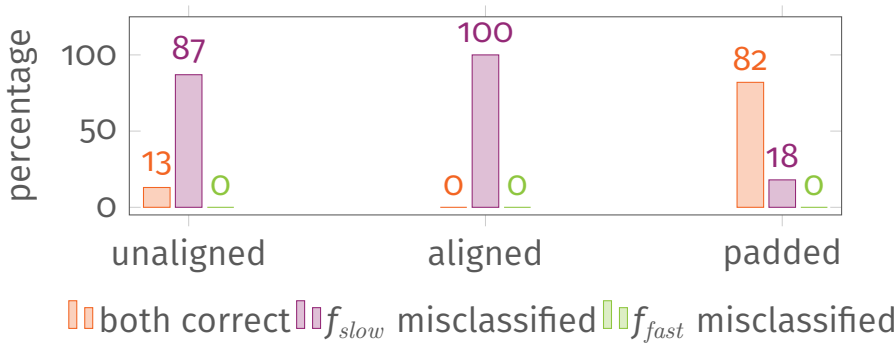
# Recovering resolution - Edge thresholding

- We can get a higher resolution for a classifier only
- Often sufficient to see which of two functions takes longer

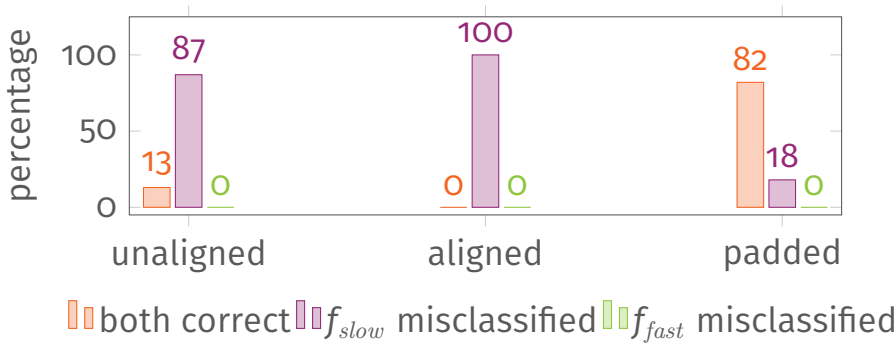



- Edge thresholding: apply padding such that the slow function crosses one more clock edge than the fast function.

# Recovering resolution - Edge thresholding

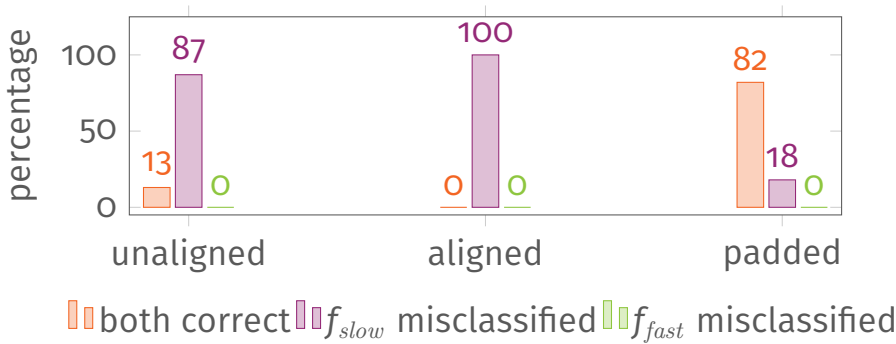


# Recovering resolution - Edge thresholding




- Yields nanosecond resolution

# Recovering resolution - Edge thresholding




- Yields nanosecond resolution
- Firefox/Tor (2 ns), Edge (10 ns), Chrome (15 ns)

# Building a timer

- Goal: counter that does not block main thread

# Building a timer

- Goal: counter that does not block main thread
- Baseline setTimeout: 4 ms (except Edge: 2 ms)

# Building a timer

- Goal: counter that does not block main thread
- Baseline setTimeout: 4 ms (except Edge: 2 ms)
- CSS animation: increase width of element as fast as possible

# Building a timer

- Goal: counter that does not block main thread
- Baseline setTimeout: 4 ms (except Edge: 2 ms)
- CSS animation: increase width of element as fast as possible
- Width of element is timestamp

# Building a timer

- Goal: counter that does not block main thread
- Baseline setTimeout: 4 ms (except Edge: 2 ms)
- CSS animation: increase width of element as fast as possible
- Width of element is timestamp
- However, animation is limited to 60 fps $\rightarrow$ 16 ms

# Building a timer - Web worker

- JavaScript can spawn new threads called web worker

# Building a timer - Web worker

- JavaScript can spawn new threads called web worker
- Web worker communicate using message passing

# Building a timer - Web worker

- JavaScript can spawn new threads called web worker
- Web worker communicate using message passing
- Let worker count and request timestamp in main thread

# Building a timer - Web worker

- JavaScript can spawn new threads called web worker
- Web worker communicate using message passing
- Let worker count and request timestamp in main thread
- Multiple possibilities: `postMessage`, `MessageChannel` or `BroadcastChannel`

# Building a timer - Web worker

- JavaScript can spawn new threads called web worker
- Web worker communicate using message passing
- Let worker count and request timestamp in main thread
- Multiple possibilities: `postMessage`, `MessageChannel` or `BroadcastChannel`
- Yields microsecond resolution (even on Tor and Fuzzyfox)

# Building a timer - Web worker

- Experimental feature to share data: `SharedArrayBuffer`

# Building a timer - Web worker

- Experimental feature to share data: `SharedArrayBuffer`
- Web worker can simultaneously read/write data

# Building a timer - Web worker

- Experimental feature to share data: `SharedArrayBuffer`
- Web worker can simultaneously read/write data
- No message passing overhead

# Building a timer - Web worker

- Experimental feature to share data: `SharedArrayBuffer`
- Web worker can simultaneously read/write data
- No message passing overhead
- One dedicated worker for incrementing the shared variable

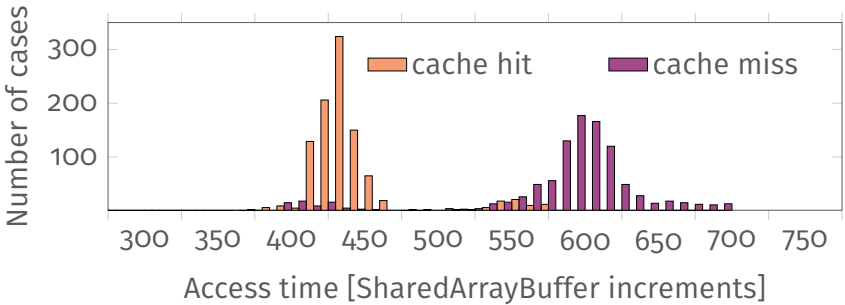
# Building a timer - Web worker

- Experimental feature to share data: `SharedArrayBuffer`
- Web worker can simultaneously read/write data
- No message passing overhead
- One dedicated worker for incrementing the shared variable
- Firefox/Fuzzyfox: 2 ns, Chrome: 15 ns

# Building a timer - Web worker

- Experimental feature to share data: `SharedArrayBuffer`
- Web worker can simultaneously read/write data
- No message passing overhead
- One dedicated worker for incrementing the shared variable
- Firefox/Fuzzyfox: 2 ns, Chrome: 15 ns
- Sufficient for microarchitectural attacks

# Measuring cache timing



# DRAM covert channel

- Sender and receiver agree on a bank (can be hardcoded)

# DRAM covert channel

- Sender and receiver agree on a bank (can be hardcoded)
- Both (native) sender inside VM and JavaScript on host select a different row inside this bank

# DRAM covert channel

- Sender and receiver agree on a bank (can be hardcoded)
- Both (native) sender inside VM and JavaScript on host select a different row inside this bank
- JavaScript measures access time for this row

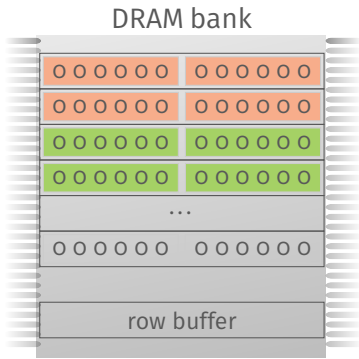
# DRAM covert channel

- Sender and receiver agree on a bank (can be hardcoded)
- Both (native) sender inside VM and JavaScript on host select a different row inside this bank
- JavaScript measures access time for this row
- Sender can transmit 0 by doing nothing and 1 by causing row conflict

# DRAM covert channel

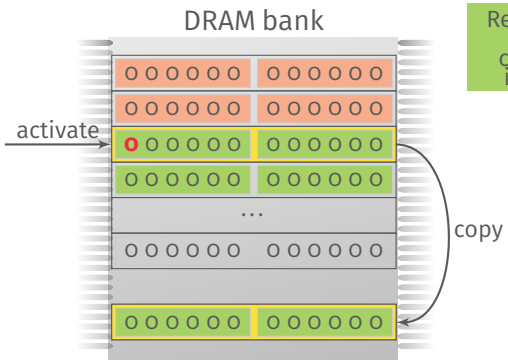
- Sender and receiver agree on a bank (can be hardcoded)
- Both (native) sender inside VM and JavaScript on host select a different row inside this bank
- JavaScript measures access time for this row
- Sender can transmit 0 by doing nothing and 1 by causing row conflict
- If measured timing was “fast” sender transmitted 0.

# Transmitting data



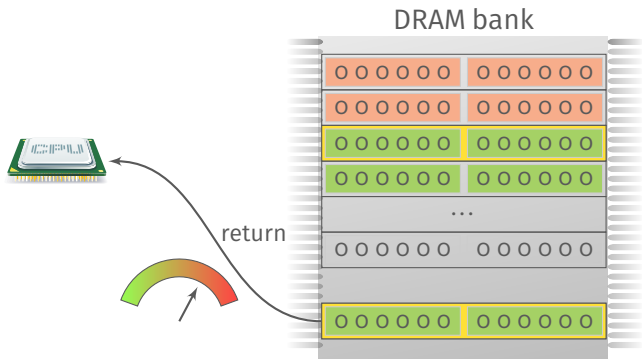

Sender and receiver decide on one bank

# Transmitting data

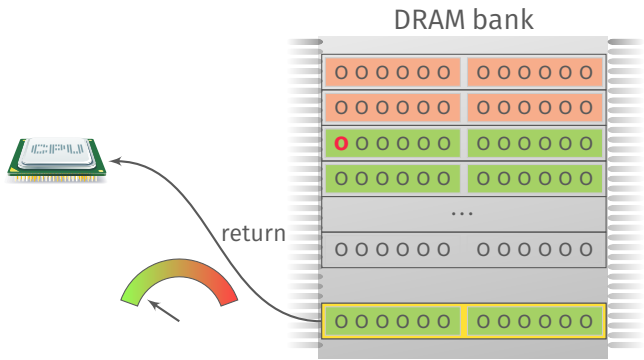



Receiver measures access time to its address

# Transmitting data

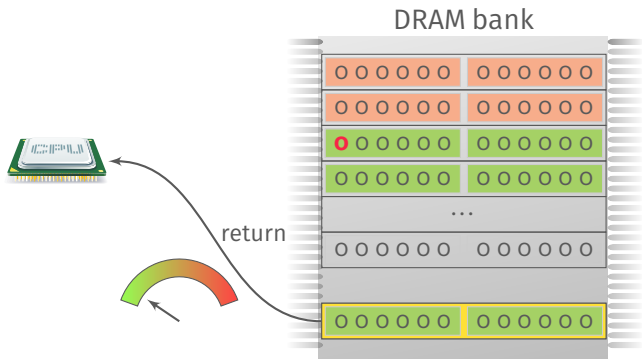


# Transmitting data

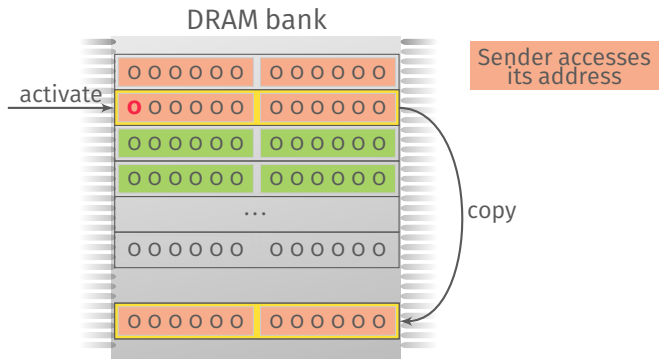



Repeated access always has low access times

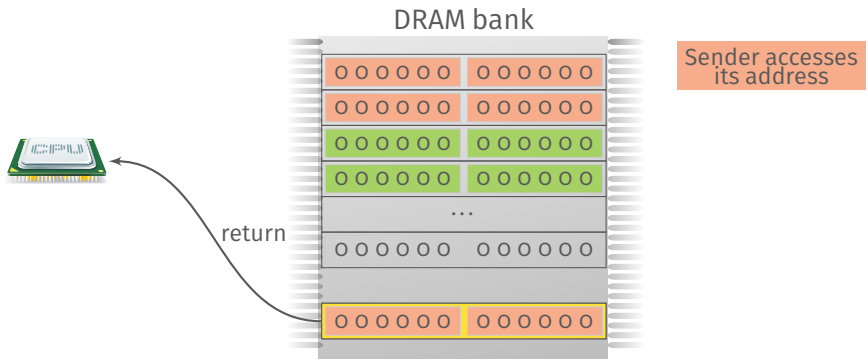
# Transmitting data



# Transmitting data

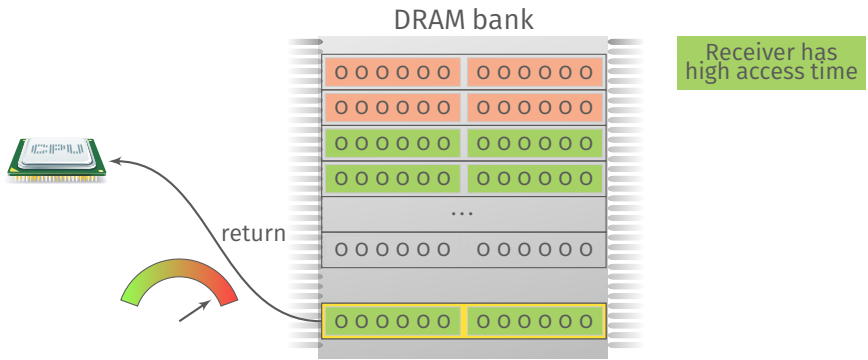



# Transmitting data

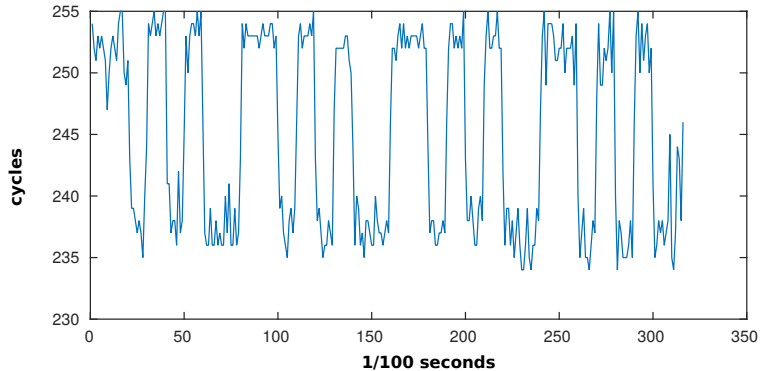


# Transmitting data

# Transmitting data

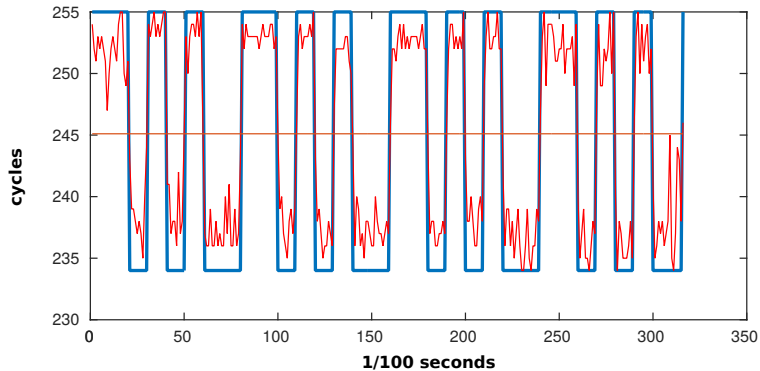


# Measurement




Multiple measurements per bit to have a reliable detection

# Measurement




Multiple measurements per bit to have a reliable detection

# Sending packets


- Communication is based on packets

# Sending packets


- Communication is based on packets
- Packet starts with a 2-bit preamble

# Sending packets


- Communication is based on packets
- Packet starts with a 2-bit preamble
- Data integrity is checked by an error-detection code (EDC)

# Sending packets


- Communication is based on packets
- Packet starts with a 2-bit preamble
- Data integrity is checked by an error-detection code (EDC)
- Sequence bit indicates whether it is a re-transmission or a new packet

# Results

- Transmission of approximately 11 bits/s

# Results

- Transmission of approximately 11 bits/s
- Can be improved using

# Results

- Transmission of approximately 11 bits/s
- Can be improved using
  - Fewer re-transmits

# Results

- Transmission of approximately 11 bits/s
- Can be improved using
  - Fewer re-transmits
  - Error correction

# Results

- Transmission of approximately 11 bits/s
- Can be improved using
  - Fewer re-transmits
  - Error correction
  - Multithreading $\rightarrow$ multiple banks in parallel

# Results

- Transmission of approximately 11 bits/s
- Can be improved using
  - Fewer re-transmits
  - Error correction
  - Multithreading $\rightarrow$ multiple banks in parallel
  - What is possible in native code? 596 kbit/s cross CPU and cross VM

# Responsible disclosure

- We contacted Google, Mozilla, Microsoft and the Tor project

# Responsible disclosure

- We contacted Google, Mozilla, Microsoft and the Tor project
- Microsoft $\rightarrow$ no response

# Responsible disclosure

- We contacted Google, Mozilla, Microsoft and the Tor project
- Microsoft $\rightarrow$ no response
- Google $\rightarrow$ several developers are assigned to the bug report

# Responsible disclosure

- We contacted Google, Mozilla, Microsoft and the Tor project
- Microsoft $\rightarrow$ no response
- Google $\rightarrow$ several developers are assigned to the bug report
- Mozilla $\rightarrow$ Discussing about clock randomization and not shipping `SharedArrayBuffer`

# Responsible disclosure

- We contacted Google, Mozilla, Microsoft and the Tor project
- Microsoft → no response
- Google → several developers are assigned to the bug report
- Mozilla → Discussing about clock randomization and not shipping `SharedArrayBuffer`
- Tor → Working on fuzzy time

# Conclusion

- Just rounding timers is not a solution

# Conclusion

- Just rounding timers is not a solution
- Multithreading allows to build new timers

# Conclusion

- Just rounding timers is not a solution
- Multithreading allows to build new timers
- Shared data comes with great risks

# Conclusion

- Just rounding timers is not a solution
- Multithreading allows to build new timers
- Shared data comes with great risks
- It allows to build timers with nanosecond resolution

# Conclusion

- Just rounding timers is not a solution
- Multithreading allows to build new timers
- Shared data comes with great risks
- It allows to build timers with nanosecond resolution
- Microarchitectural attacks in the browser are possible again

Michael Schwarz, Clémentine Maurice, Daniel Gruss, Stefan Mangard

# FANTASTIC TIMERS AND WHERE TO FIND THEM

## HIGH-RESOLUTION MICROARCHITECTURAL ATTACKS IN JAVASCRIPT