Dynamic Process Isolation

Black Hat Asia 2022

Martin Schwarzl
Graz University of Technology

Kenton Varda
Cloudflare Inc.

Michael Schwarz
CISPA Helmholtz Center for Information Security

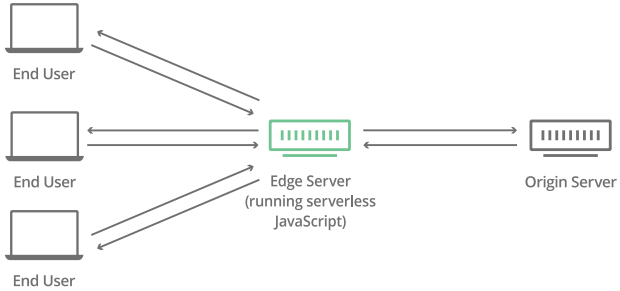
Pietro Borrello
Sapienza University of Rome

Thomas Schuster
Graz University of Technology

Andreas Kogler
Graz University of Technology

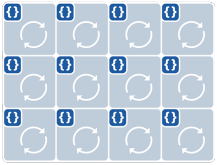
Daniel Gruss
Graz University of Technology

- Cloudflare Workers is one of the top edge-computing systems



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

- Cloudflare Workers is one of the top edge-computing systems
- Single-process design



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

- Cloudflare Workers is one of the top edge-computing systems
- Single-process design
- Vulnerable to Spectre attacks?

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

- Cloudflare Workers is one of the top edge-computing systems
- Single-process design
- Vulnerable to Spectre attacks?
- No local timers

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

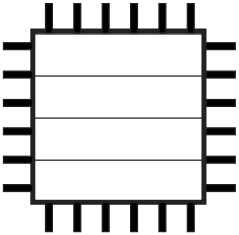
- Cloudflare Workers is one of the top edge-computing systems
- Single-process design
- Vulnerable to Spectre attacks?
- No local timers
- Number of memory, (sub-)requests, runtime is limited

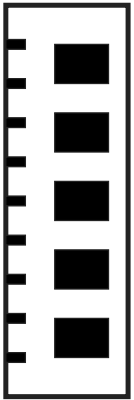
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

- Cloudflare Workers is one of the top edge-computing systems
- Single-process design
- Vulnerable to Spectre attacks?
- No local timers
- Number of memory, (sub-)requests, runtime is limited
- If an attack is possible, we need a low-overhead solution!

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

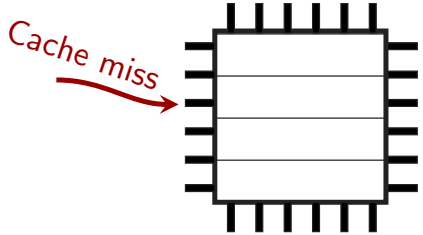
```
maccess(i);

maccess(i);
```

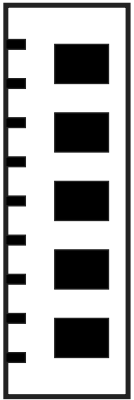




Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

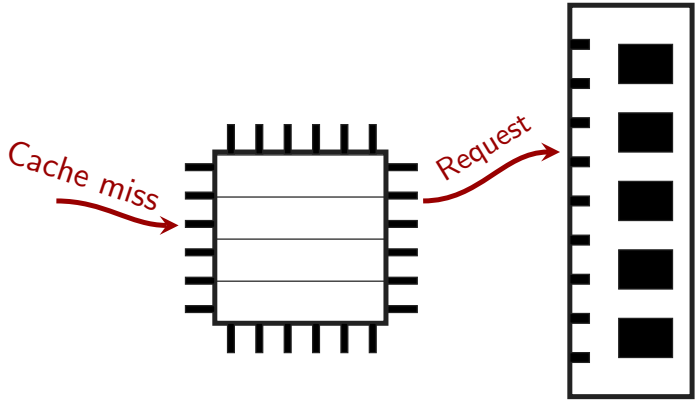
```
maccess(i);

maccess(i);
```





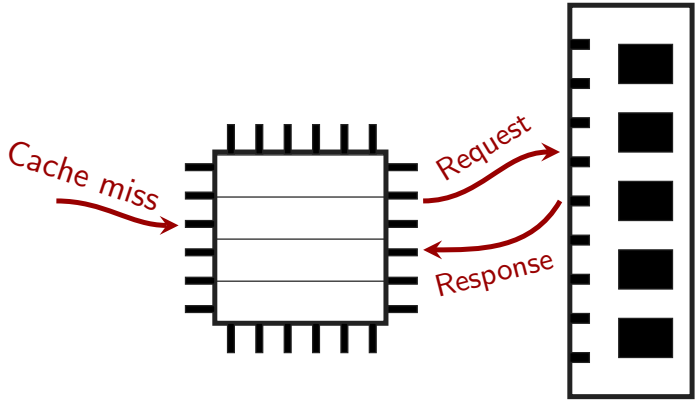
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
maccess(i);

maccess(i);
```

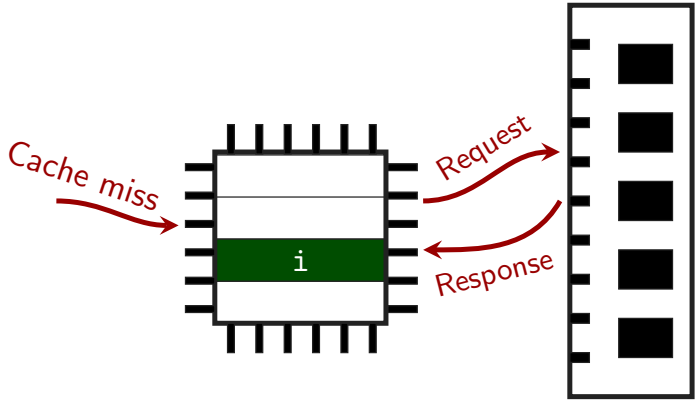


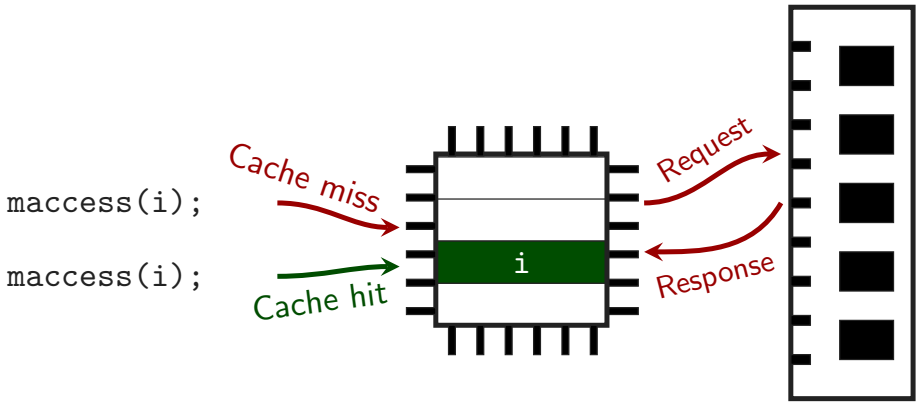
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

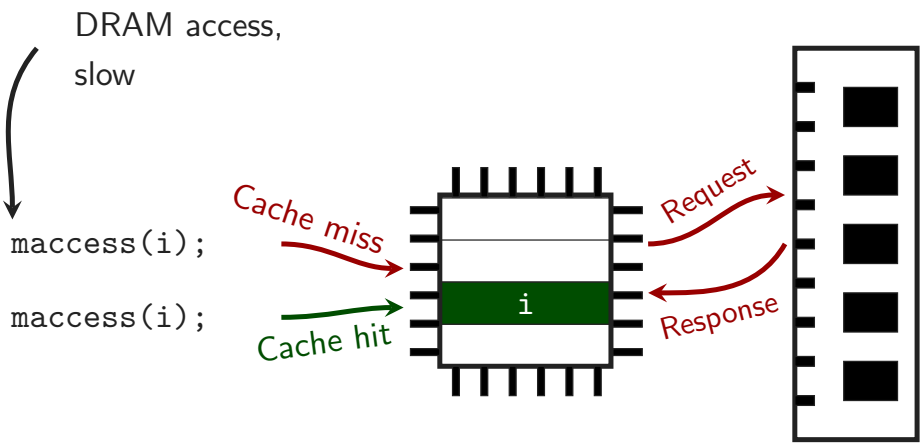
```
maccess(i);

maccess(i);
```



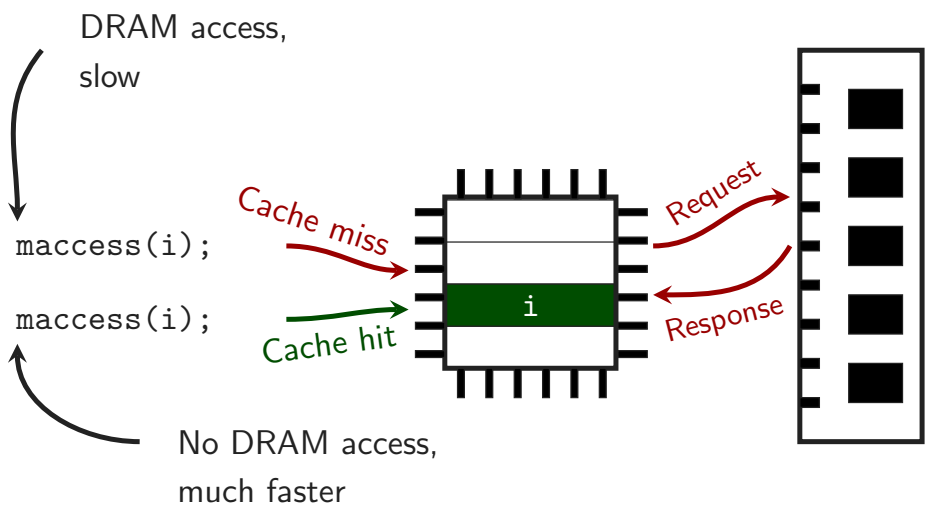
```
maccess(i);

maccess(i);
```




maccess(i);
maccess(i);
Cache miss
Cache hit
i
Request
Response



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



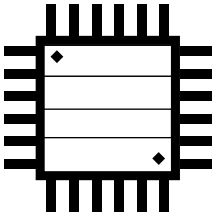
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Shared Memory

ATTACKER

flush

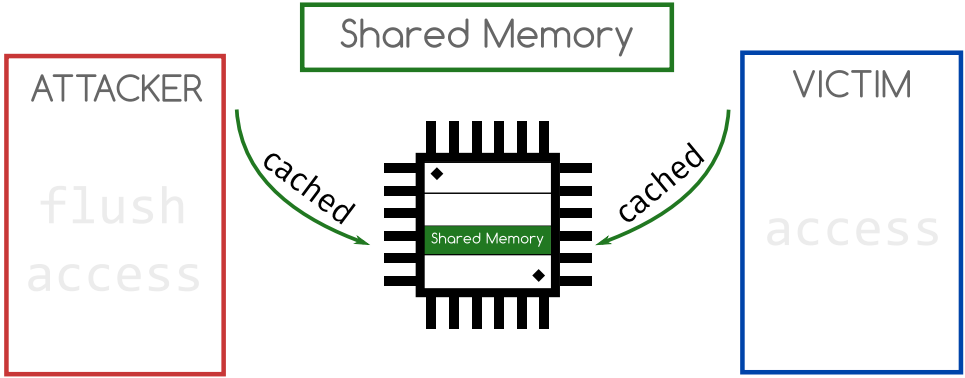
access



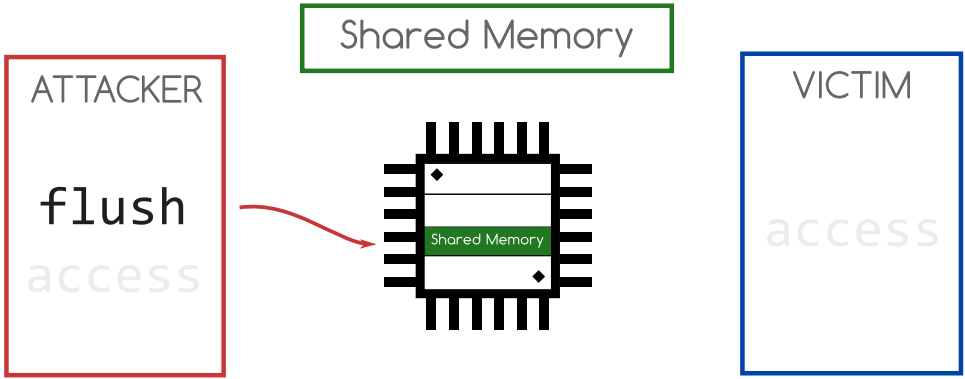
VICTIM

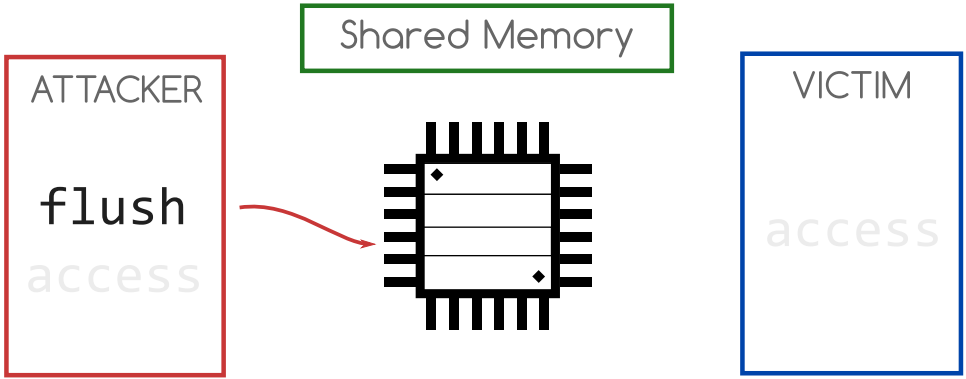
access

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

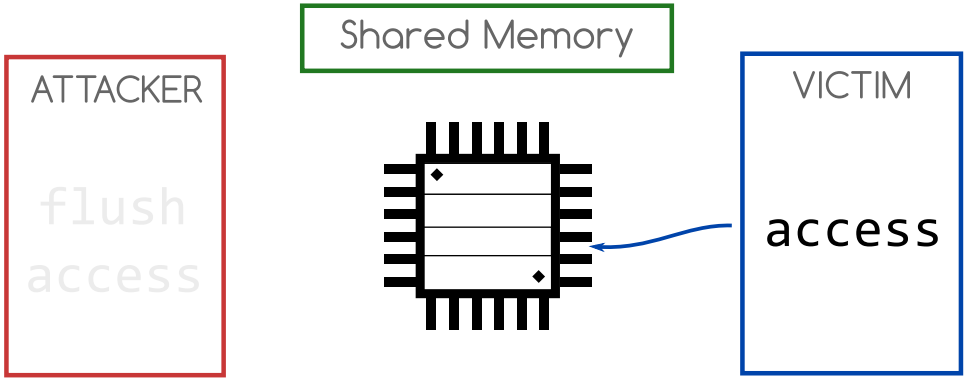


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

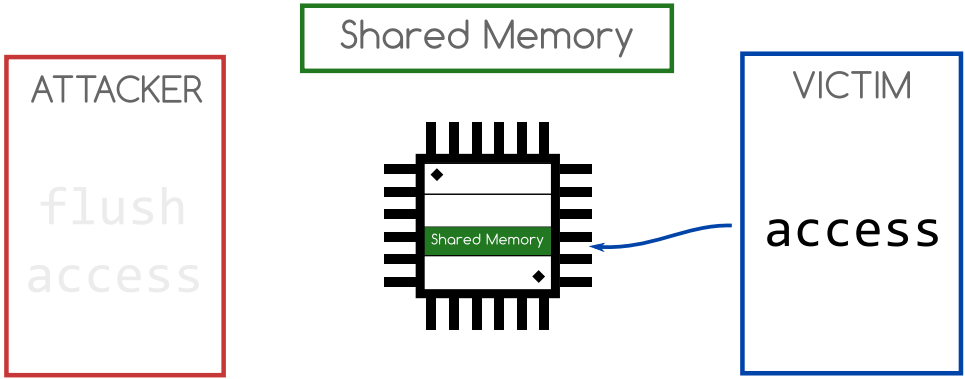

Shared Memory
ATTACKER
flush
access
Shared Memory
VICTIM
access

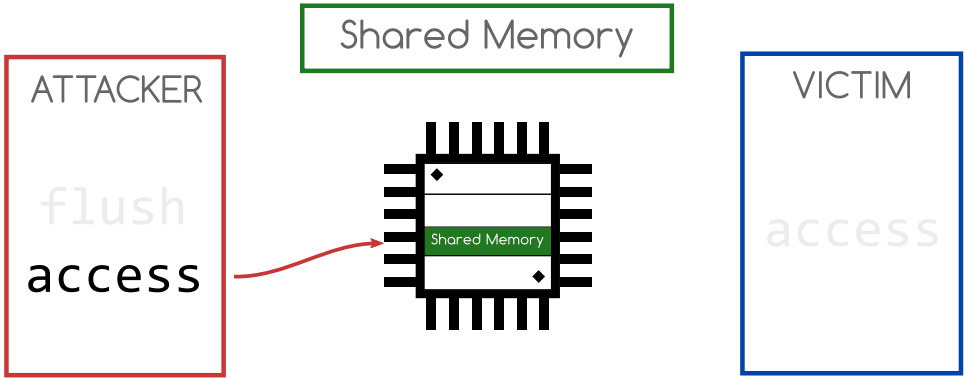


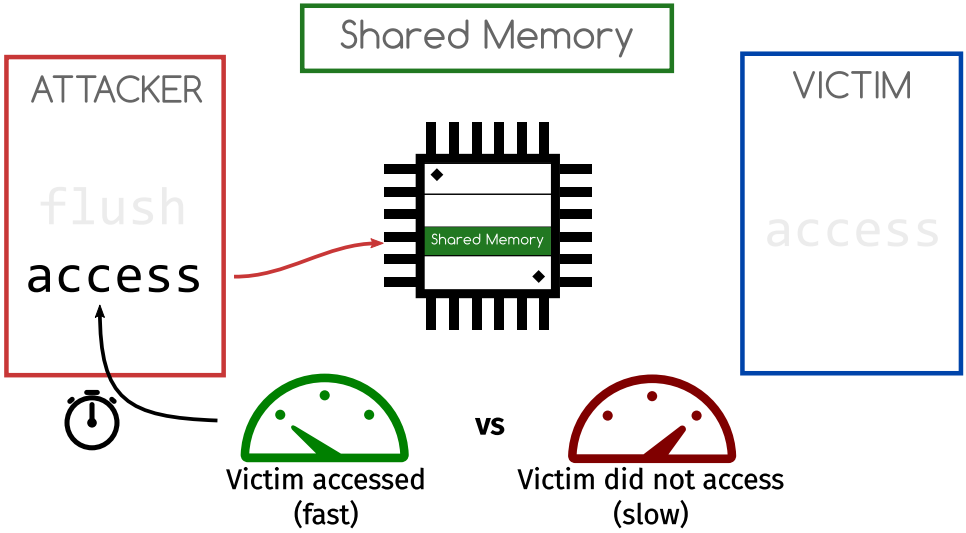
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)


Shared Memory
ATTACKER
flush
access
Shared Memory
VICTIM
access


Shared Memory
ATTACKER
flush
access
Shared Memory
VICTIM
access



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT gadget

```
if (index < data_size)
{
  y = lut[data[index]*4096]
}
```


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
index = 0
```

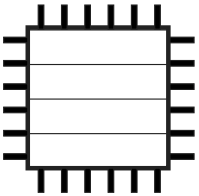
Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

```
if (index < 4)
```

then → `lut[data[index]]`

else → `{}`



Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)

```
index = 0
```

```
if (index < 4)
```

then

```
lut[data[index]]
```

else

Speculate

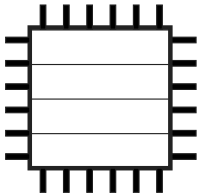
{}

Lookup Table

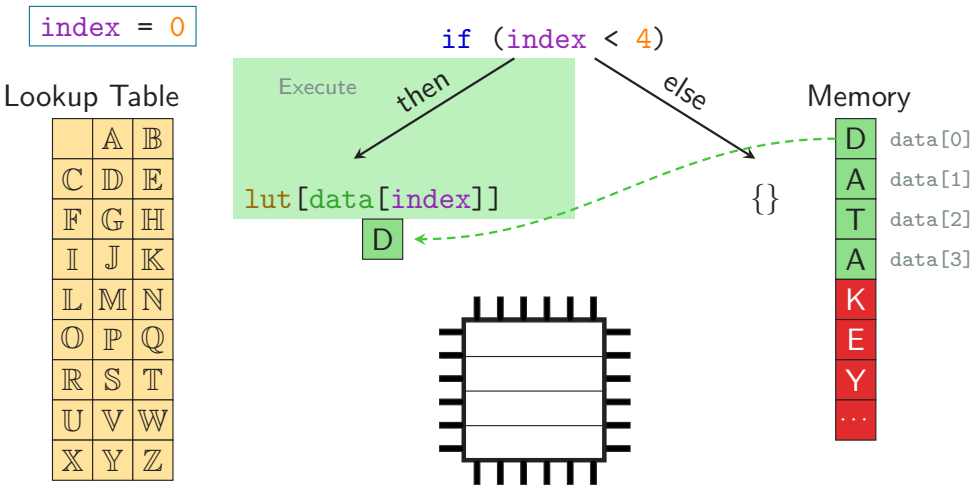
	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	

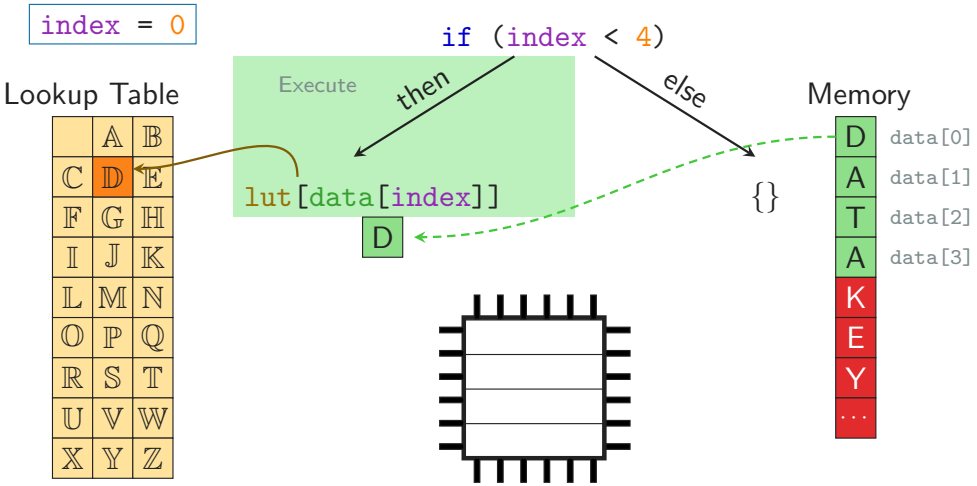


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

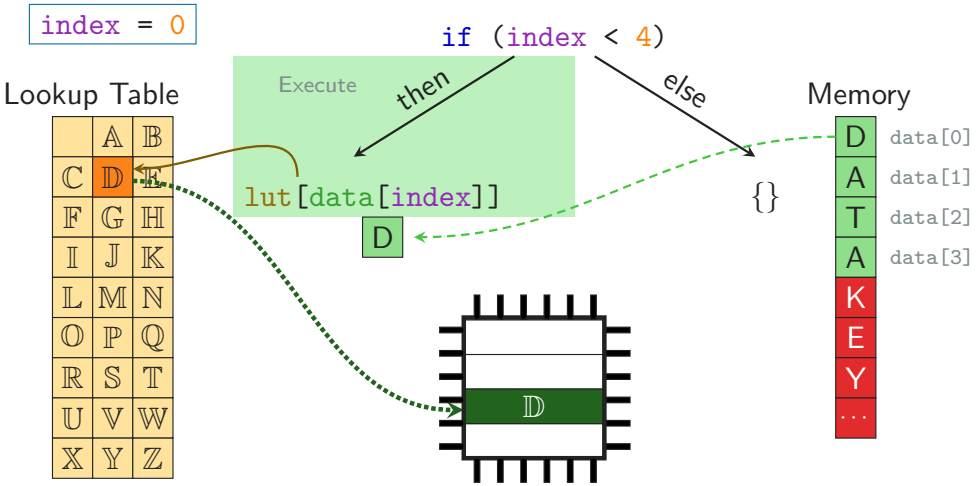


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



Spectre-PHT (aka Spectre Variant 1)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
index = 1
```

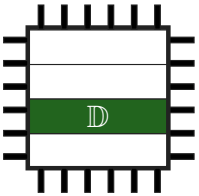
Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

```
if (index < 4)
```

then → `lut[data[index]]`

else → `{}`



Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
index = 1
```

```
if (index < 4)
```

Speculate

then

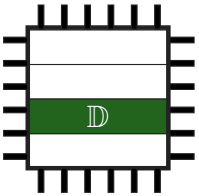
else

```
lut[data[index]]
```

{}

Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

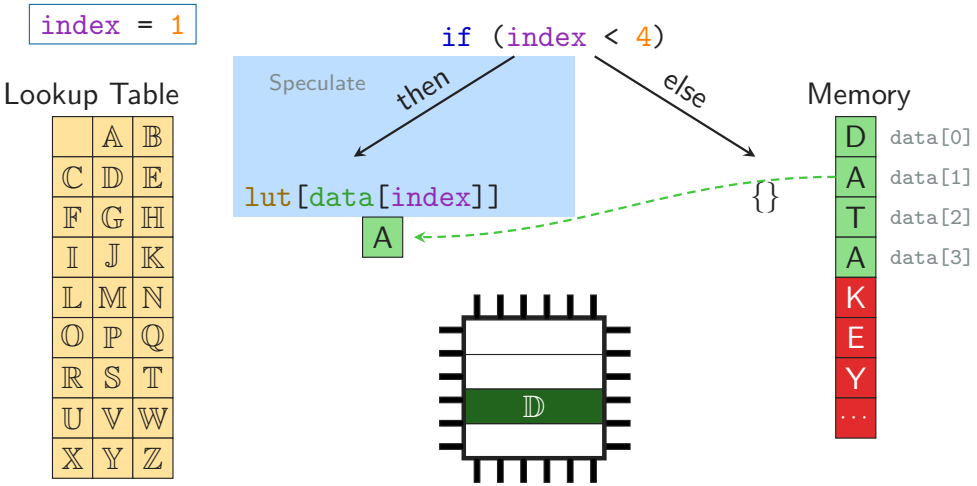


Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	

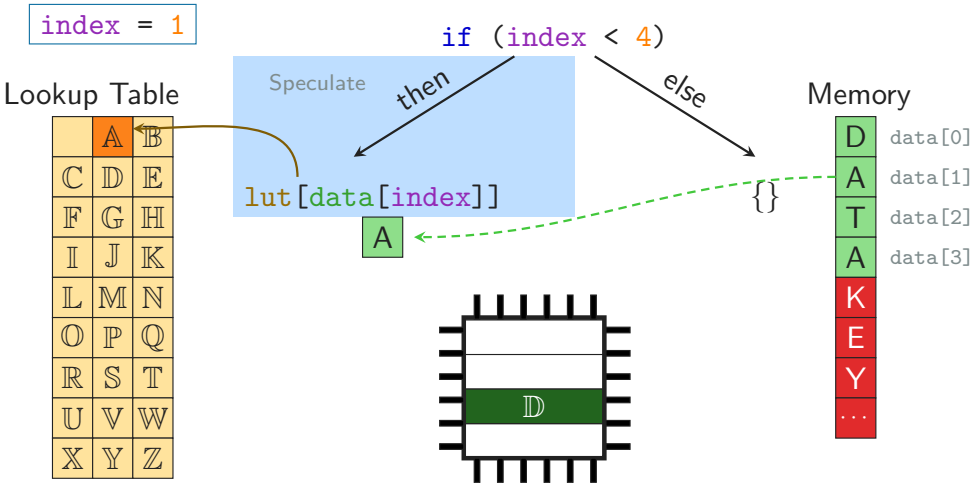
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



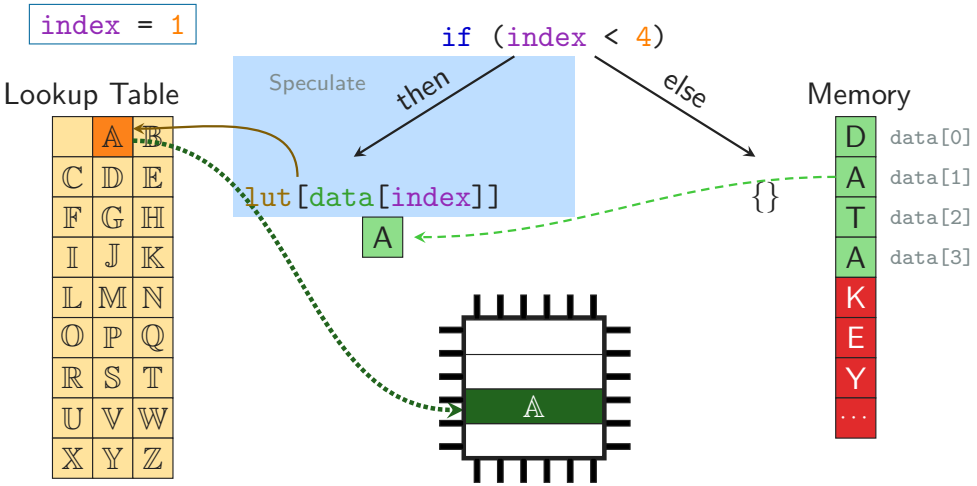
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
index = 1
```

```
if (index < 4)
```

then

Execute

```
lut[data[index]]
```

else

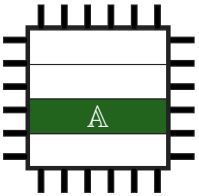
{}

Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)

```
index = 2
```

```
if (index < 4)
```

Speculate

then

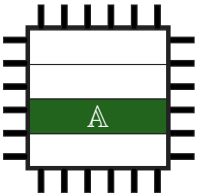
```
lut[data[index]]
```

else

{}

Lookup Table

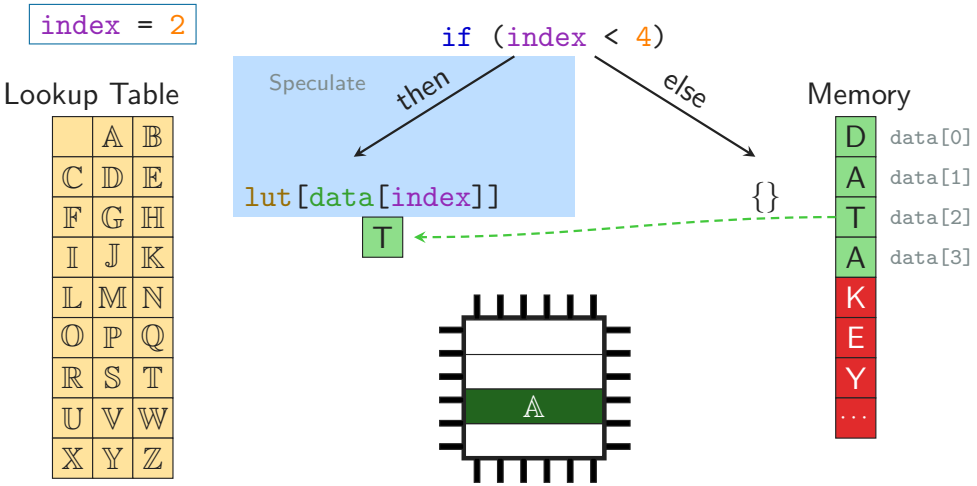
	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z



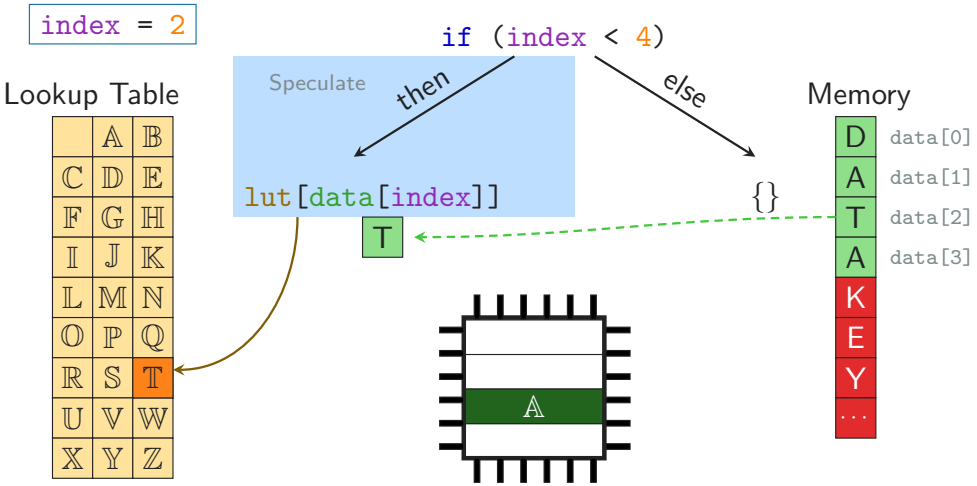
Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

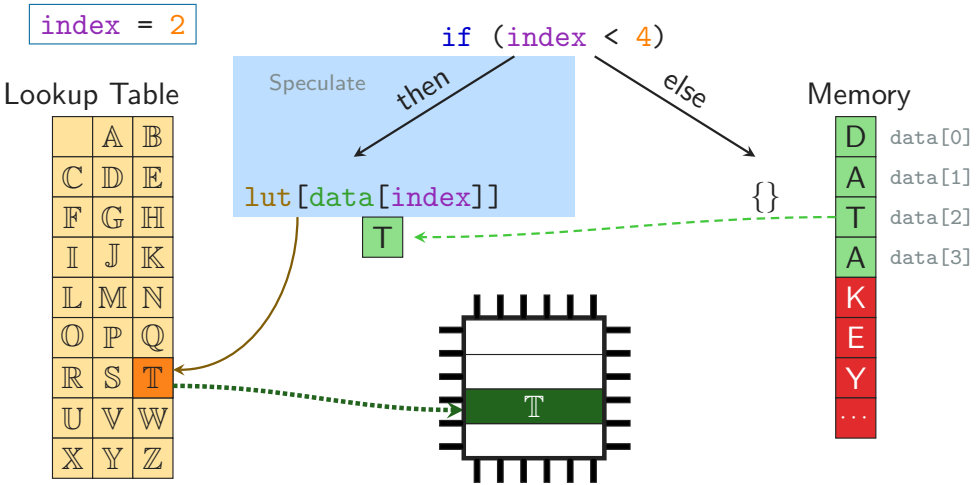


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
index = 2
```

```
if (index < 4)
```

Execute

then

```
lut[data[index]]
```

else

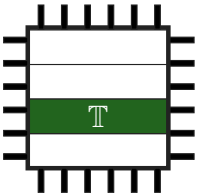
{}

Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	



Spectre-PHT (aka Spectre Variant 1)

```
index = 3
```

```
if (index < 4)
```

Speculate

then

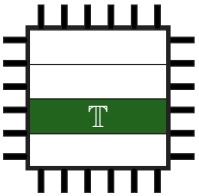
```
lut[data[index]]
```

else

{}

Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

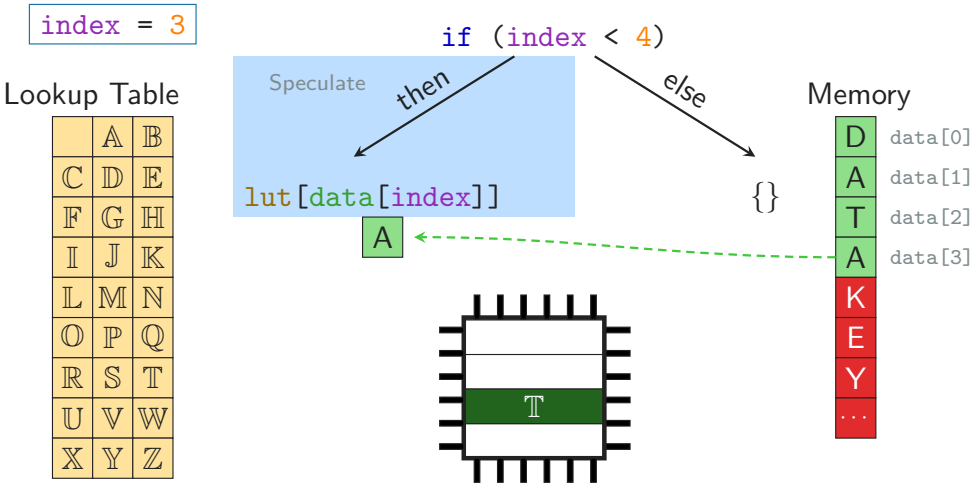


Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	

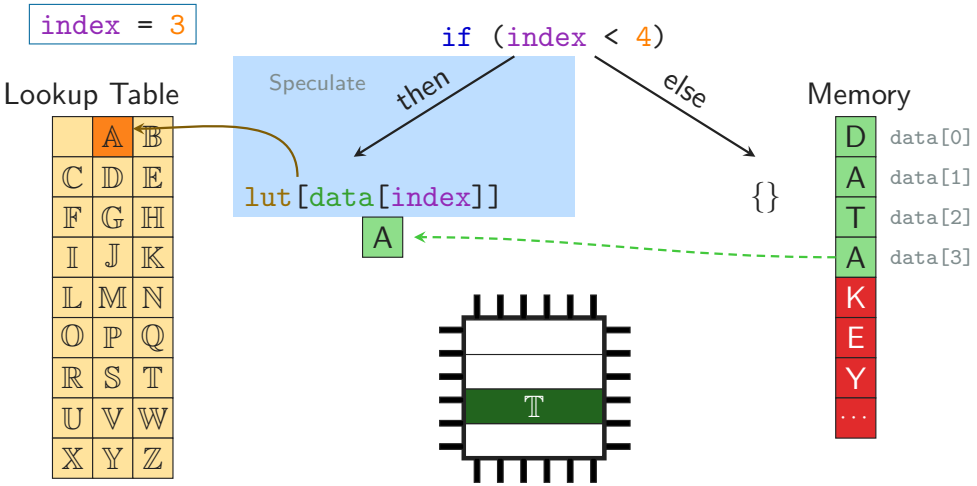
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



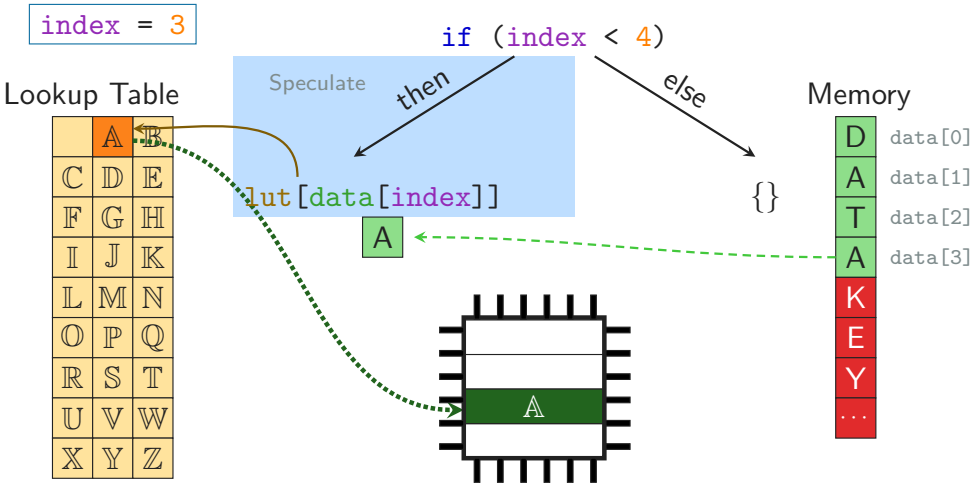
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
index = 3
```

```
if (index < 4)
```

Execute

then

else

```
lut[data[index]]
```

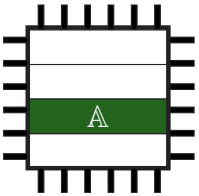
{}

Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
index = 4
```

```
if (index < 4)
```

Speculate

then

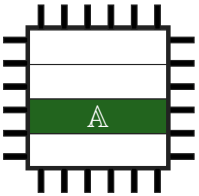
```
lut[data[index]]
```

else

{}

Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

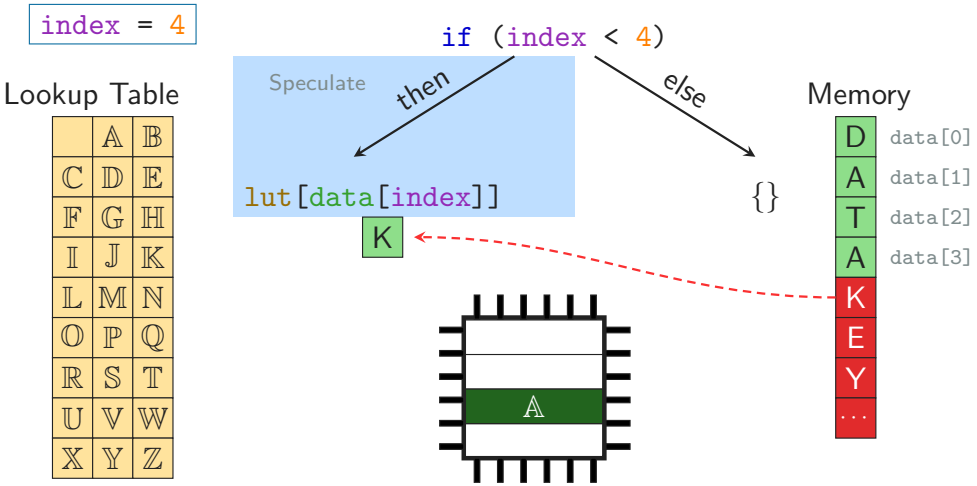


Memory

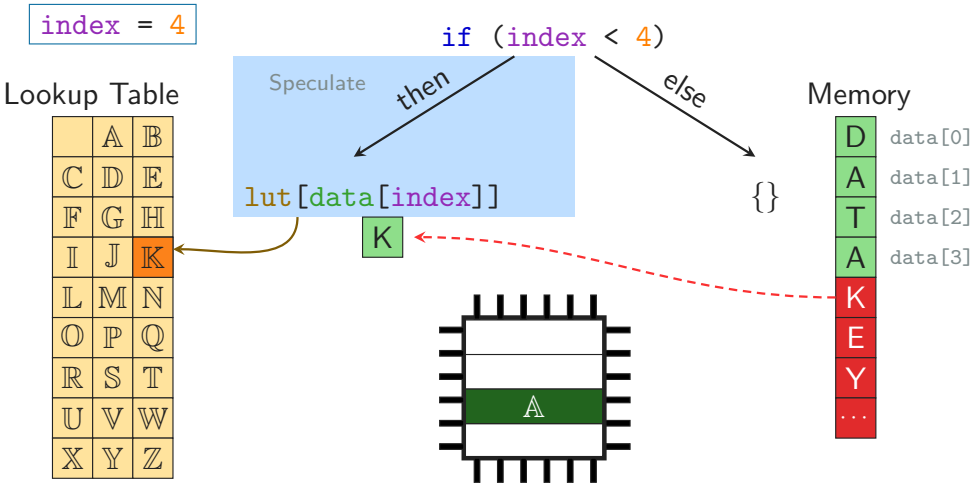
D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)

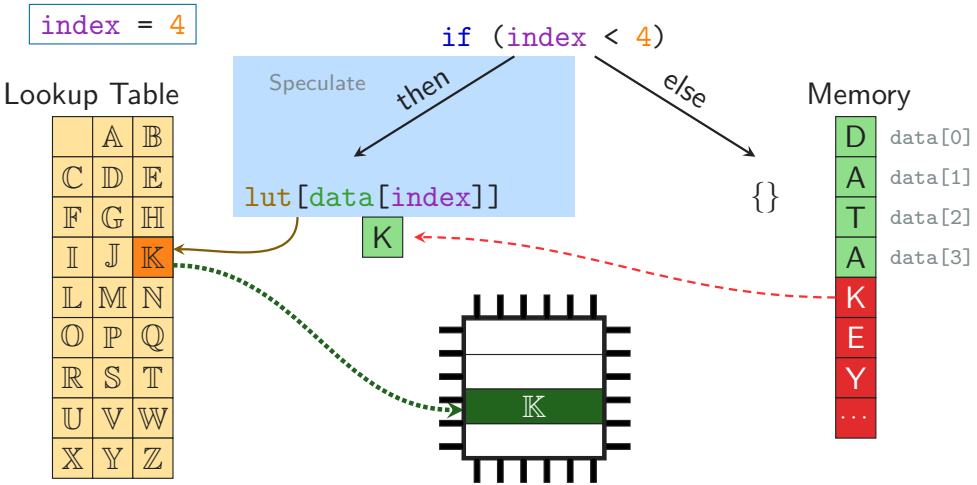


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre-PHT (aka Spectre Variant 1)



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

```
index = 4
```

Lookup Table

	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

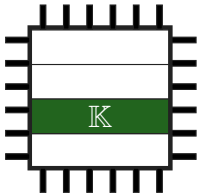
```
if (index < 4)
```

then

```
lut[data[index]]
```

else

Execute {}



Memory

D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Application-Level

Spectre Mitigations

Application-Level

System-Level

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Spectre Mitigations

Application-Level

System-Level

Hardware-Level

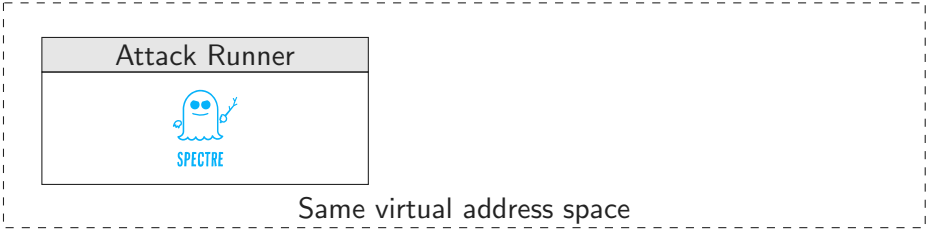
- Freeze the time during execution

- Freeze the time during execution
- No native code

- Freeze the time during execution
- No native code
- No shared-memory

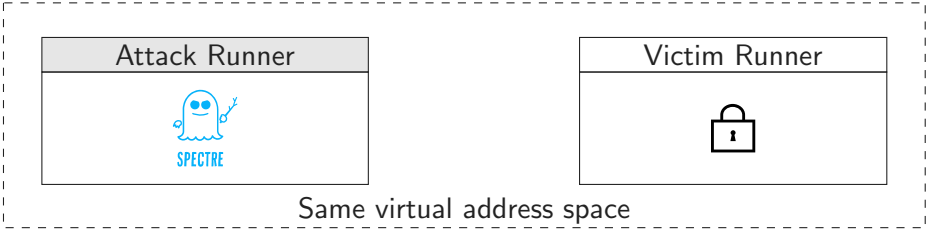
- Freeze the time during execution
- No native code
- No shared-memory
- No multithreading

High-level Attack Overview



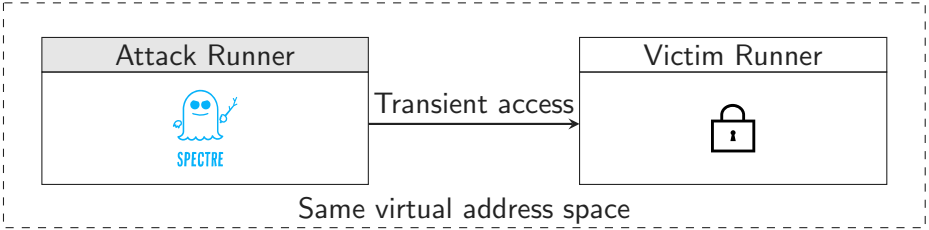
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

High-level Attack Overview



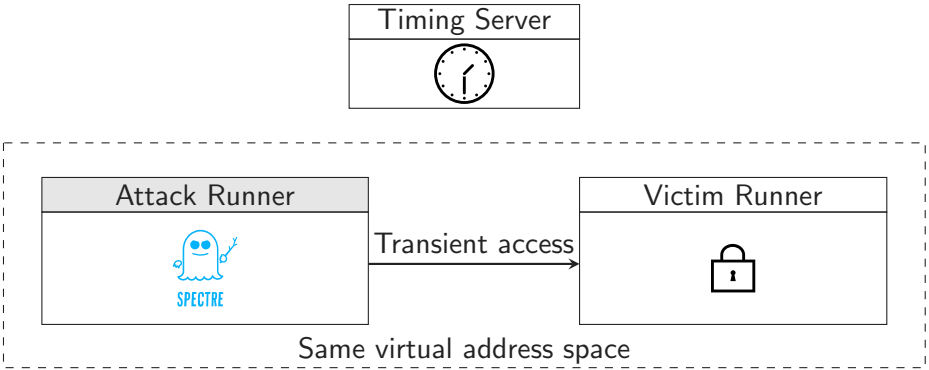
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

High-level Attack Overview



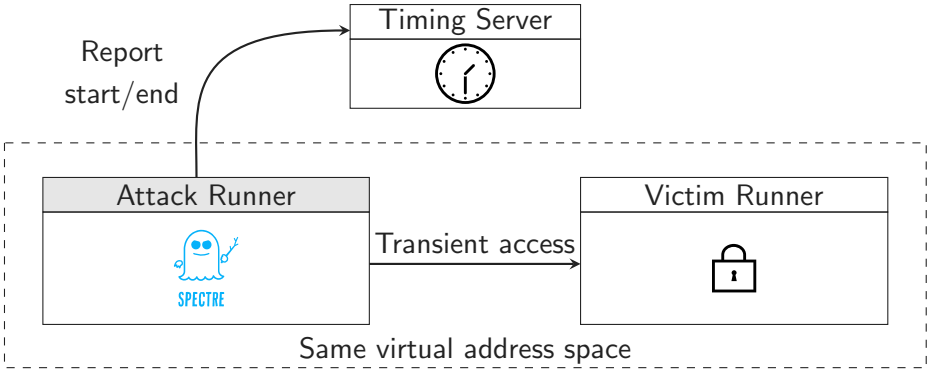
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

High-level Attack Overview

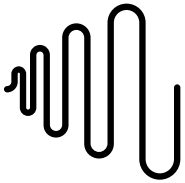


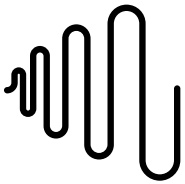
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

High-level Attack Overview

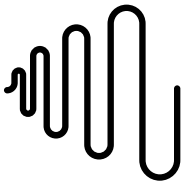


- **Amplify** timing either:

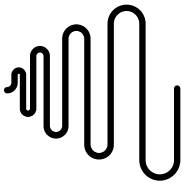




- **Amplify** timing either:
  - Encode secret into **multiple** cache lines



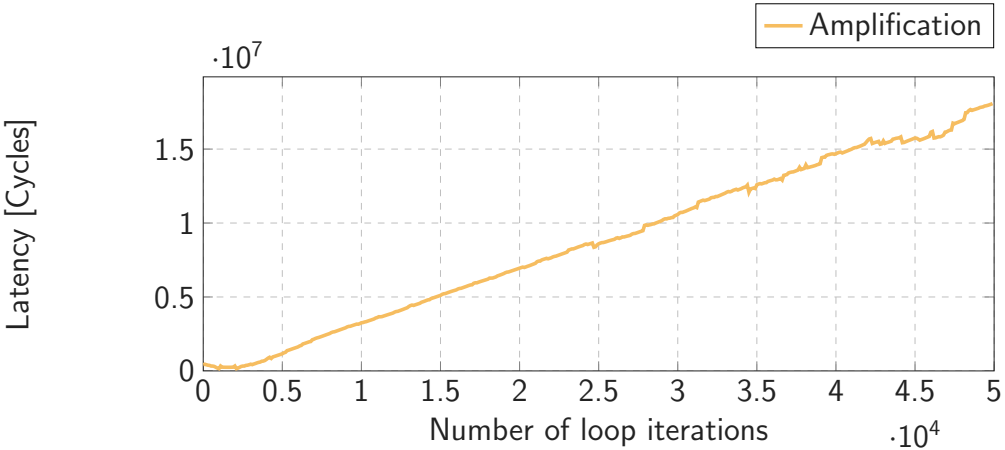
- **Amplify** timing either:
  - Encode secret into **multiple** cache lines
  - Loop over gadget *n* times



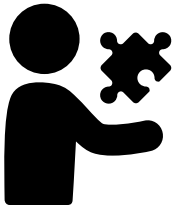
- **Amplify** timing either:
  - Encode secret into **multiple** cache lines
  - Loop over gadget *n* times

```
for (int i = 0; i < N; i++) {
  // evict A and B
  // ...
  if (secret_bit) { access A } // transient
  else { access B }
  access A
}
```

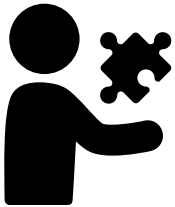

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



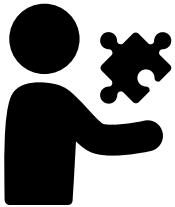
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



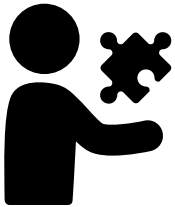
- No native code execution



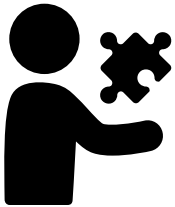
- No native code execution
  - Cannot **Flush & Reload** memory



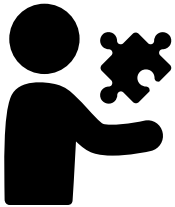
- No native code execution
  - Cannot **Flush & Reload** memory
  - Cannot even build **eviction sets**



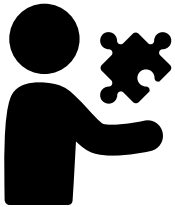
- No native code execution
  - Cannot **Flush & Reload** memory
  - Cannot even build **eviction sets**
  - ✓ Evict the whole cache iterating over a huge array



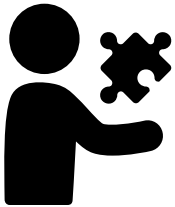
- JavaScript is optimized and deoptimized



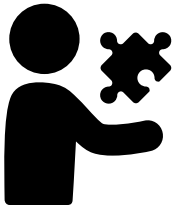
- JavaScript is optimized and deoptimized
  - Use **assumptions** on variables



- JavaScript is optimized and deoptimized
  - Use **assumptions** on variables
  - Deoptimize if **invalidated**



- JavaScript is optimized and deoptimized
  - Use **assumptions** on variables
  - Deoptimize if **invalidated**
  - ✓ Hide the attack behind a **mispredicted** branch



- JavaScript is optimized and deoptimized
  - Use **assumptions** on variables
  - Deoptimize if **invalidated**
  - ✓ Hide the attack behind a **mispredicted** branch
  - ✓ **Prevent** inlining and further optimizations using huge functions

- JavaScript objects cannot index the whole memory

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

- JavaScript objects cannot index the whole memory
- To leak from different workers you need **64-bit addresses**

- JavaScript objects cannot index the whole memory
- To leak from different workers you need **64-bit addresses**

✓ Use **ArrayBuffers** to leak the whole 64-bit address space

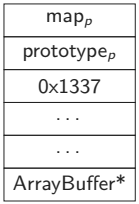
Speculative Type Confusion I

- JavaScript objects cannot index the whole memory
- To leak from different workers you need **64-bit addresses**

✓ Use **ArrayBuffers** to leak the whole 64-bit address space

✓ Create objects of different types and **confuse** branch predictor

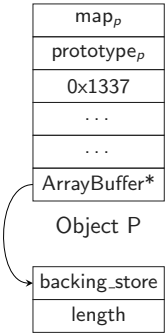
Speculative Type Confusion II



Object P

```
obj = {
  i: 0x1337,
  ...,
  ptr: new ArrayBuffer(0x1000),
};

if (obj instanceof ObjP) {
  leak(obj.ptr[0])
}
```

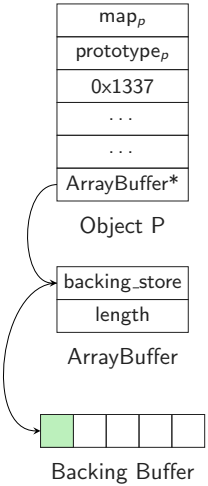

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)




```
obj = {
  i: 0x1337,
  ...,
  ptr: new ArrayBuffer(0x1000),
};

if (obj instanceof ObjP) {
  leak(obj.ptr[0])
}
```


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

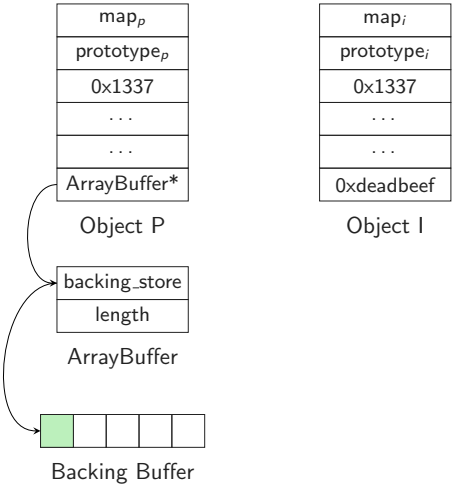
Speculative Type Confusion II




```
obj = {
  i: 0x1337,
  ...,
  ptr: new ArrayBuffer(0x1000),
};

if (obj instanceof ObjP) {
  leak(obj.ptr[0])
}
```


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

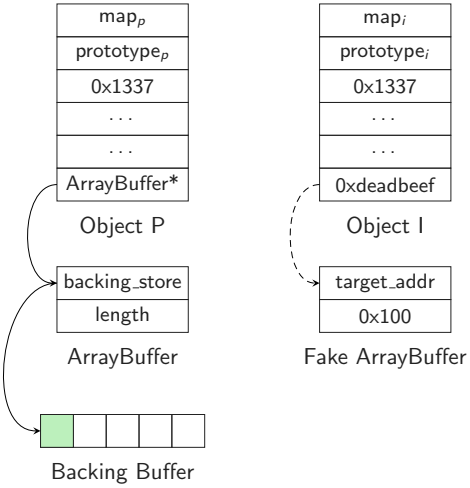
Speculative Type Confusion II




```
obj = {
  i: 0x1337,
  ...,
  ptr: 0xdeadbeef,
};

if (obj instanceof ObjP) {
  leak(obj.ptr[0])
}
```


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

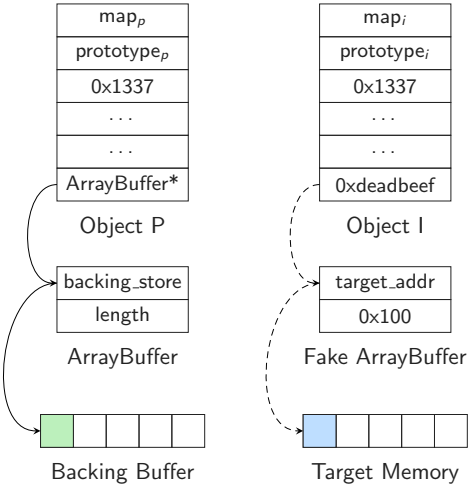
Speculative Type Confusion II




```
obj = {
  i: 0x1337,
  ...,
  ptr: 0xdeadbeef,
};

if (obj instanceof ObjP) {
  leak(obj.ptr[0])
}
```


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

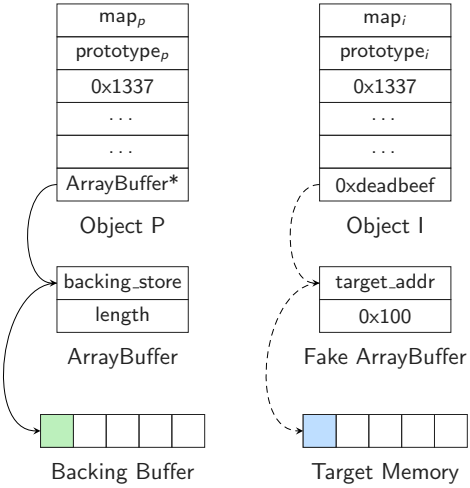
Speculative Type Confusion II




```
obj = {
  i: 0x1337,
  ...,
  ptr: 0xdeadbeef,
};

if (obj instanceof ObjP) {
  leak(obj.ptr[0])
}
```


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

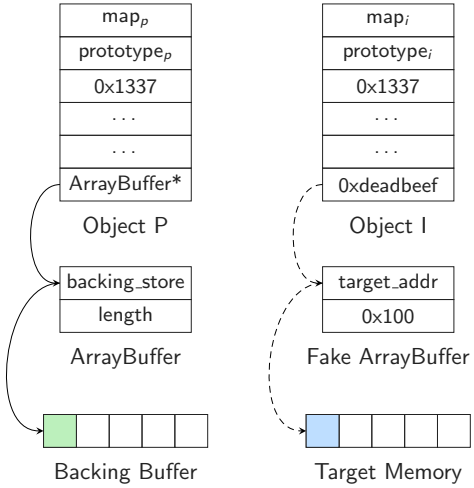
Speculative Type Confusion II




```
obj = {
  i: 0x1337,
  ...,
  ptr: 0xdeadbeef,
};

if (obj instanceof ObjP) {
  leak(obj.ptr[0])
}
```


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Speculative Type Confusion II




```
obj = {
  i: 0x1337,
  ...,
  ptr: 0xdeadbeef,
};

if (obj instanceof ObjP) {
  leak(obj.ptr[0])
}
```


Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

DEMO

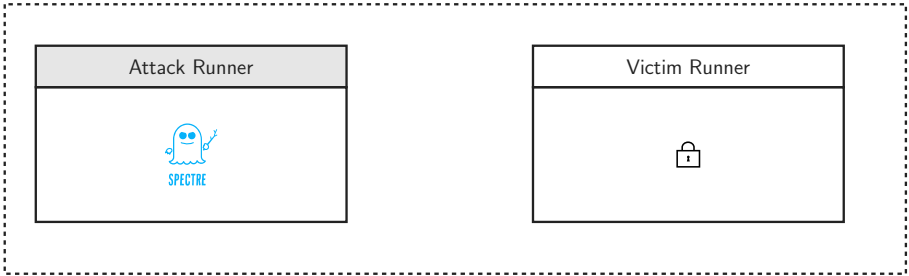
- **Leak** 120 bit/hour over the network

- **Leak** 120 bit/hour over the network
- Works within the offered runtime of 30 seconds

- **Leak** 120 bit/hour over the network
- Works within the offered runtime of 30 seconds
- Use speculative **type confusion** to create 64-bit leak primitive

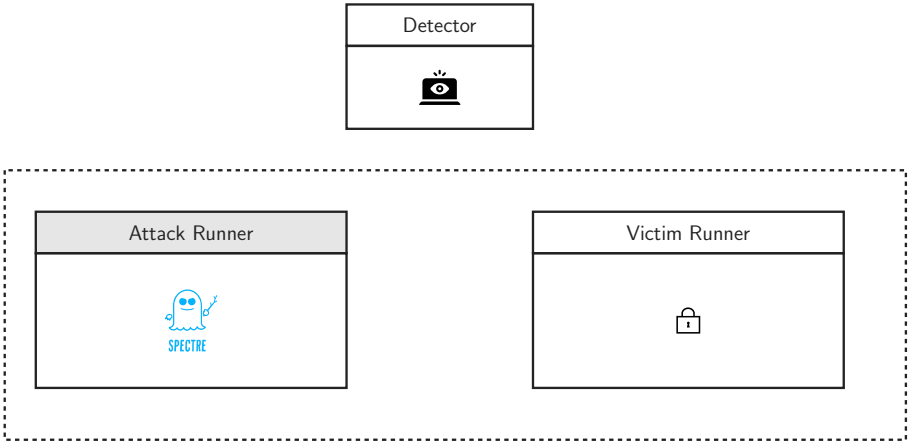
Defense

Dynamic Process Isolation - High Level Idea



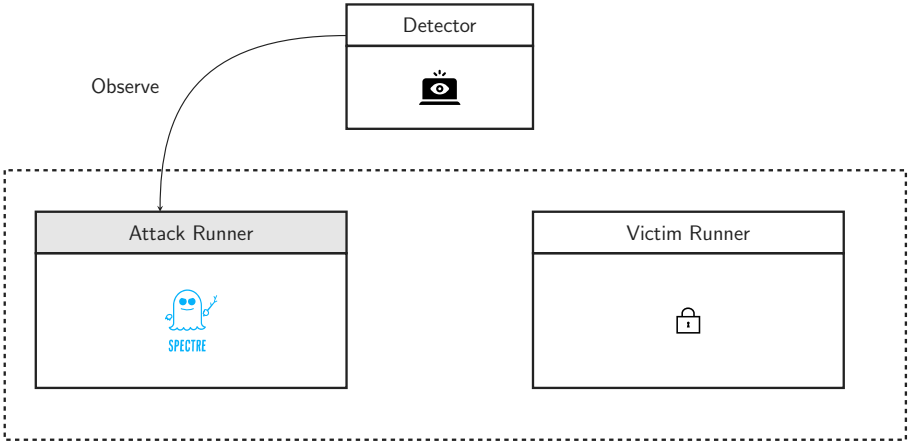
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Dynamic Process Isolation - High Level Idea



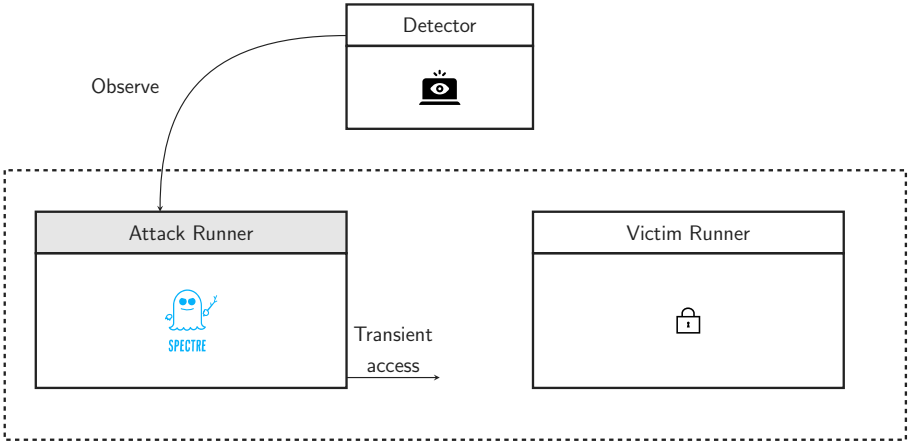
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Dynamic Process Isolation - High Level Idea



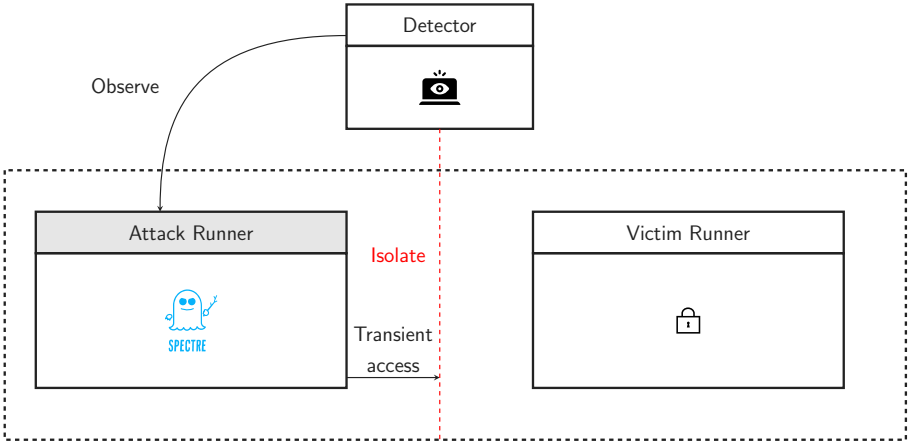
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Dynamic Process Isolation - High Level Idea



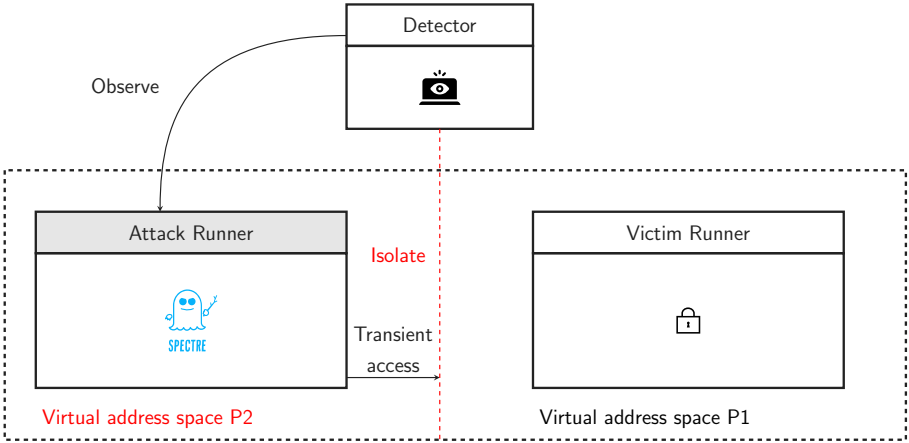
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Dynamic Process Isolation - High Level Idea



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

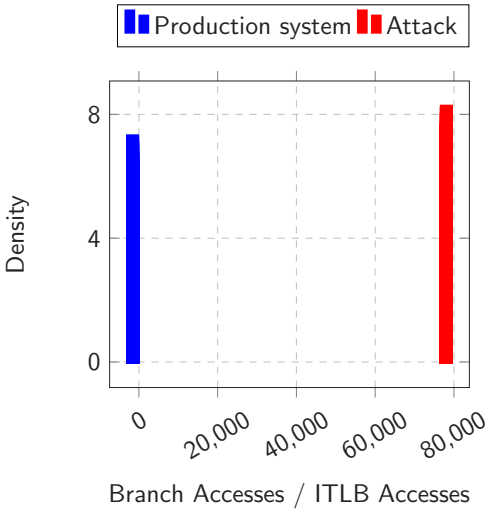
Dynamic Process Isolation - High Level Idea



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

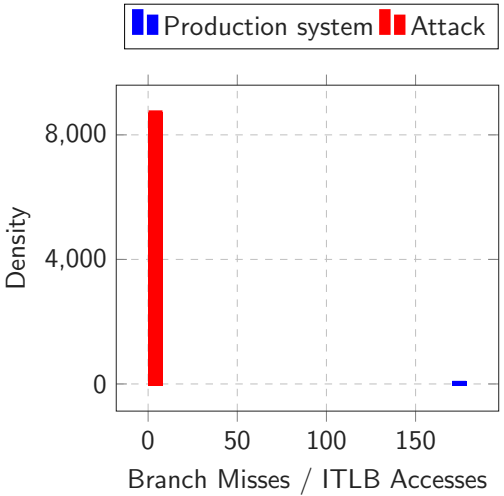
- How to observe the Attacker?

- How to observe the Attacker?
  - ✓ Performance Monitoring Counters



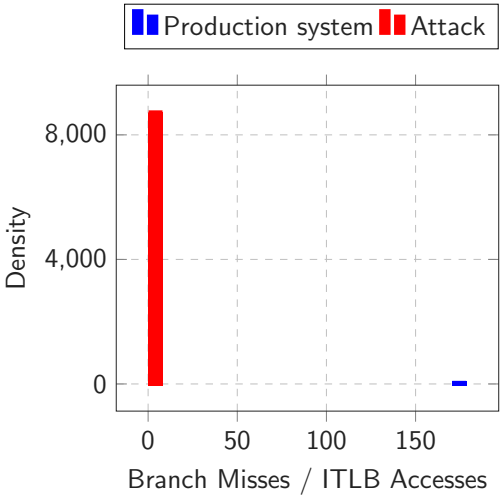
- How to observe the Attacker?
  - ✓ Performance Monitoring Counters
- Branch **Misses**

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



- How to observe the Attacker?
  - ✓ Performance Monitoring Counters
- Branch **Misses**
- Branch **Accesses**

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



- How to observe the Attacker?
  - ✓ Performance Monitoring Counters
- Branch **Misses**
- Branch **Accesses**
- Per **Code** Executed

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

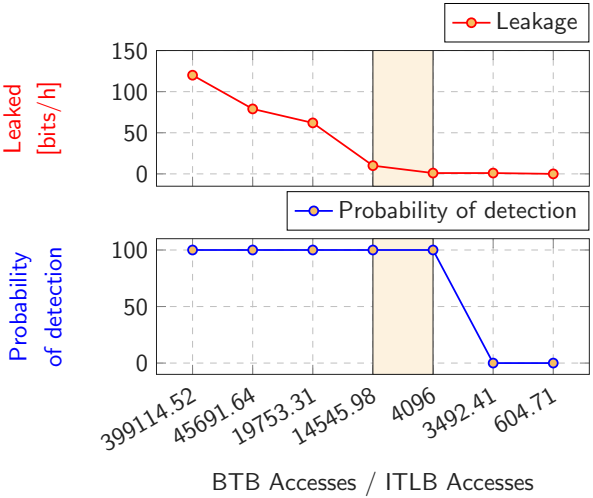
- What is the overhead?

- What is the overhead?
- Different **Interfaces**
  - **PERF**
  - **rdmsr**
  - **rdpmc**

- What is the overhead?
- Different **Interfaces**
  - **PERF**
  - **rdmsr**
  - **rdpmc**

✓ **rdpmc** → 2% overhead

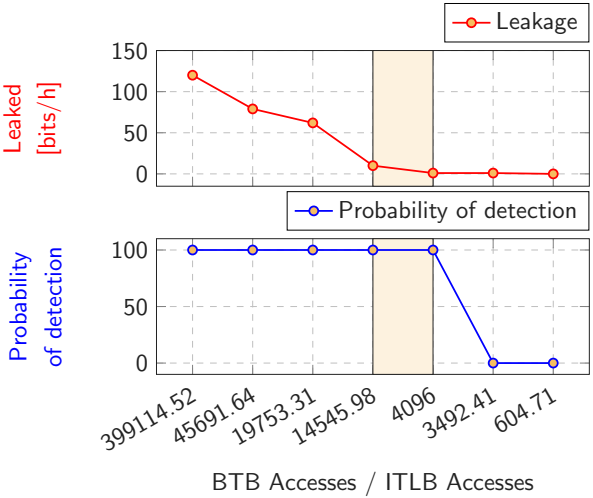
Dynamic Process Isolation - Success Rate



- What is the success rate?

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

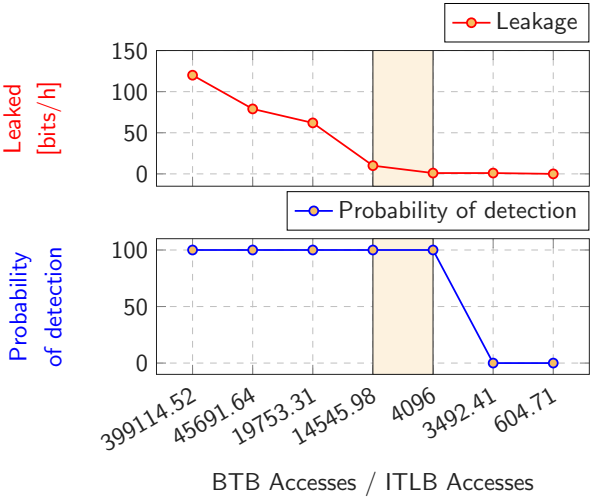
Dynamic Process Isolation - Success Rate



- What is the success rate?
- Detection vs leakage rate

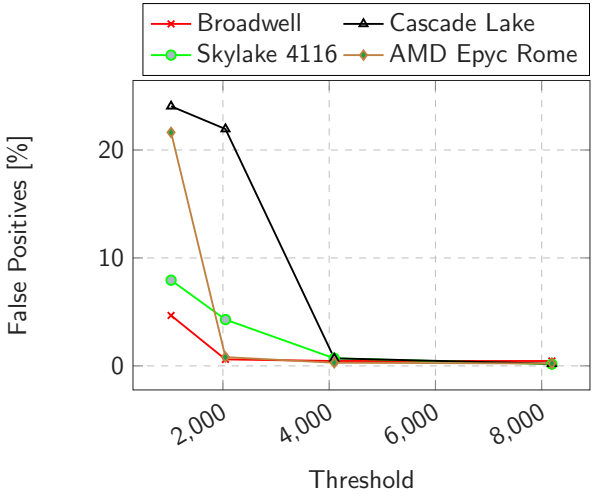
Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

Dynamic Process Isolation - Success Rate



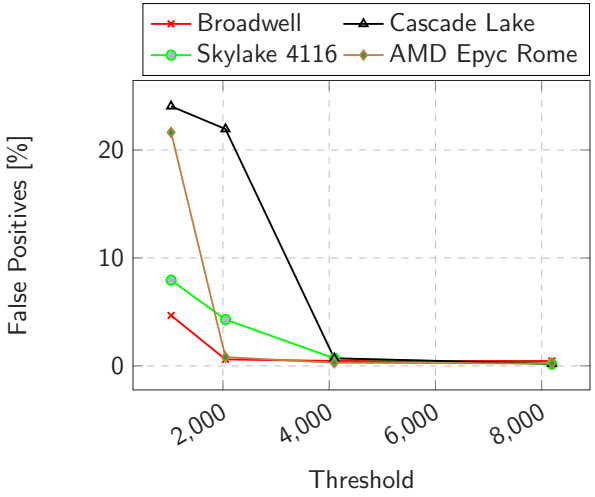
- What is the success rate?
- Detection vs leakage rate
- ✓ Strong **reduction**

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



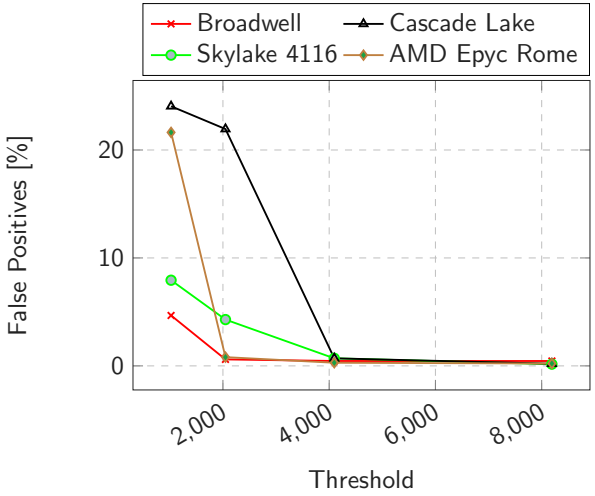
- Are there false positives?

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



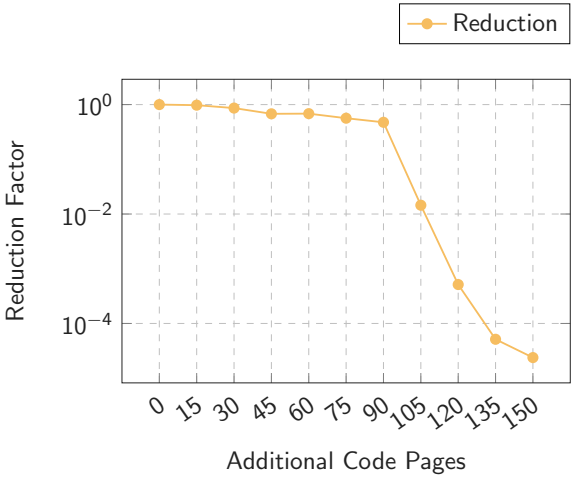
- Are there false positives?
- Different CPUs

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



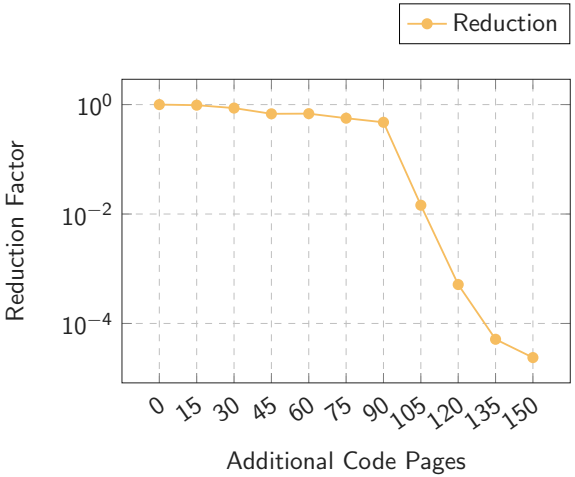
- Are there false positives?
- Different CPUs
- ✓ 4096 → 0.61%

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



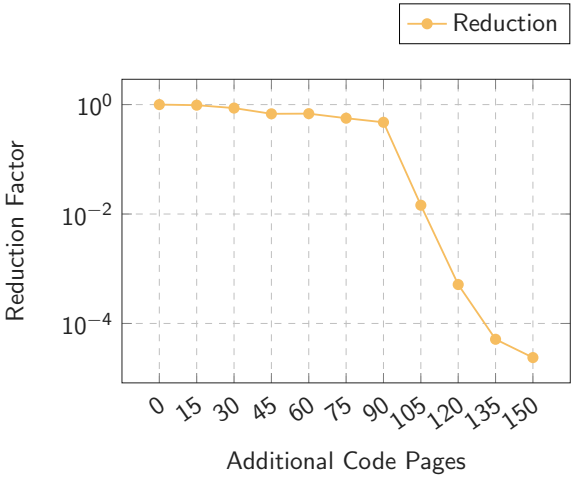
- Can the observer be tricked?

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



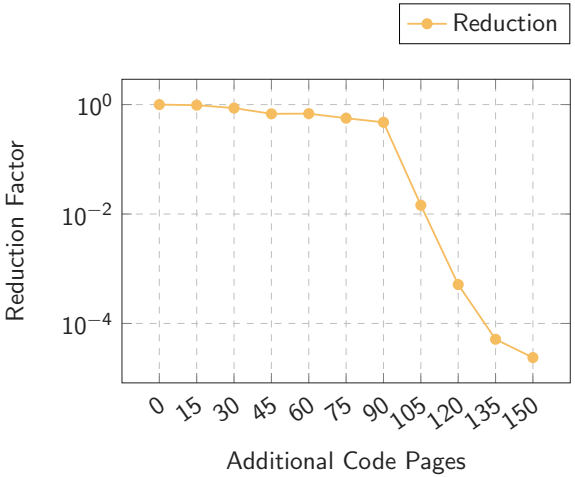
- Can the observer be tricked?
- Additional Code Pages

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



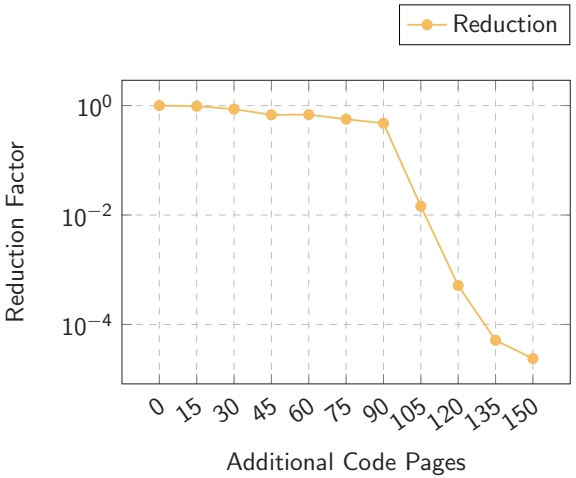
- Can the observer be tricked?
- Additional Code Pages
- Increase ITLB Accesses

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)



- Can the observer be tricked?
- Additional Code Pages
- Increase ITLB Accesses
- Reduction factor

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

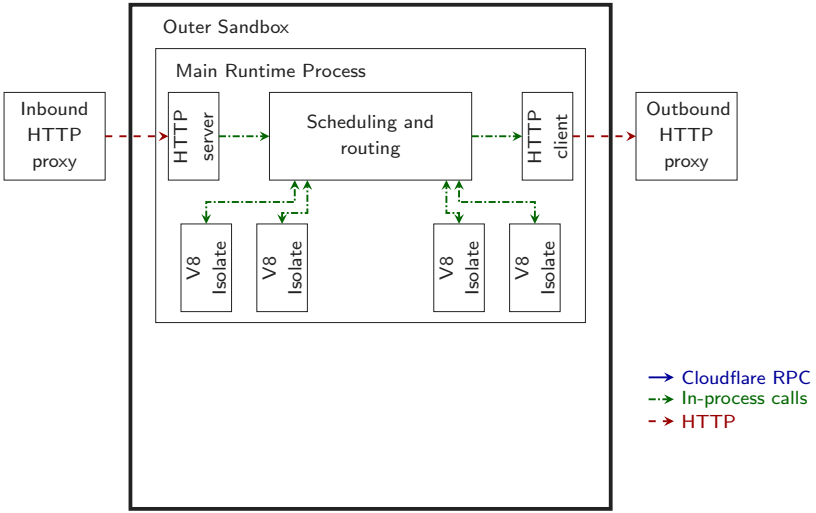


- Can the observer be tricked?
- Additional Code Pages
- Increase ITLB Accesses
- Reduction factor

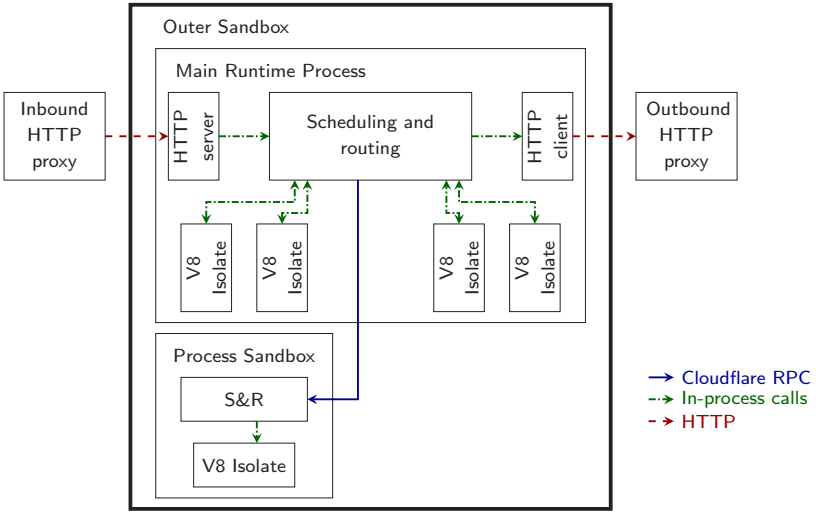
✓ Not **compiled** by JavaScript

Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

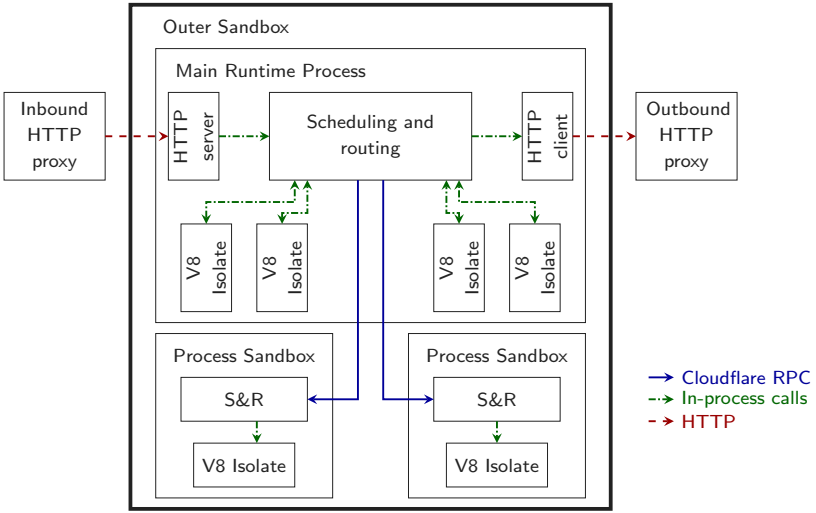
Dynamic Process Isolation - Implementation



Dynamic Process Isolation - Implementation



Dynamic Process Isolation - Implementation



Pietro Borrello (@borrello_pietro) Andreas Kogler (@0xhilbert)

- Remote Spectre attacks were possible on Cloudflare Workers

- Remote Spectre attacks were possible on Cloudflare Workers
- Our solution detects all state-of-the-art Spectre attacks

- Remote Spectre attacks were possible on Cloudflare Workers
- Our solution detects all state-of-the-art Spectre attacks
- **Production-deployed** with a low false-positive rate

- Remote Spectre attacks were possible on Cloudflare Workers
- Our solution detects all state-of-the-art Spectre attacks
- **Production-deployed** with a low false-positive rate

Dynamic Process Isolation

Martin Schwarzl
Graz University of Technology
martin.schwarzl@iaik.tugraz.at

Pietro Borrello
Sapienza University of Rome
borrello@diag.uniroma1.it

Andreas Kogler
Graz University of Technology
andreas.kogler@iaik.tugraz.at

Kenton Varda
Cloudflare
kenton@cloudflare.com

Thomas Schuster
Graz University of Technology
thomas.schuster@student.tugraz.at

Daniel Gruss
Graz University of Technology
daniel.gruss@iaik.tugraz.at

Michael Schwarz
CISPA Helmholtz Center for Information Security
michael.schwarz@cispa.saarland