ESORICS 2019

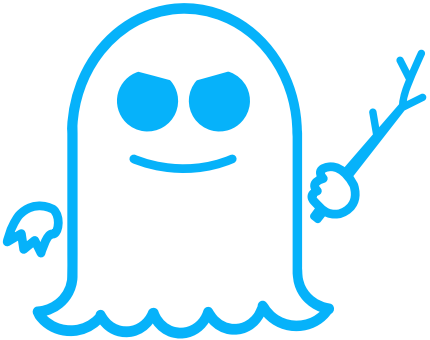
NetSpectre

Read Arbitrary Memory over Network

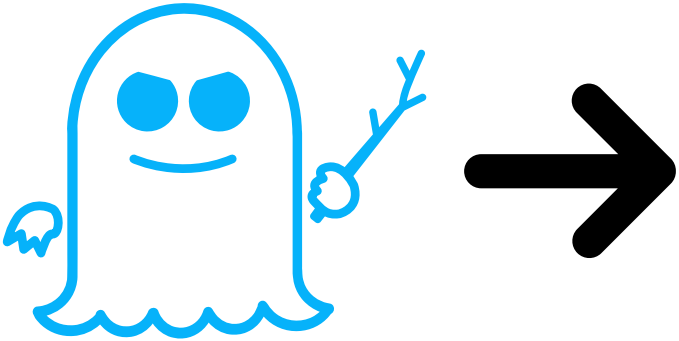
Michael Schwarz[1]
Martin Schwarzl[1]
Moritz Lipp[1]
Jon Masters[2]
Daniel Gruss[1]

[1]Graz University of Technology [2]Red Hat

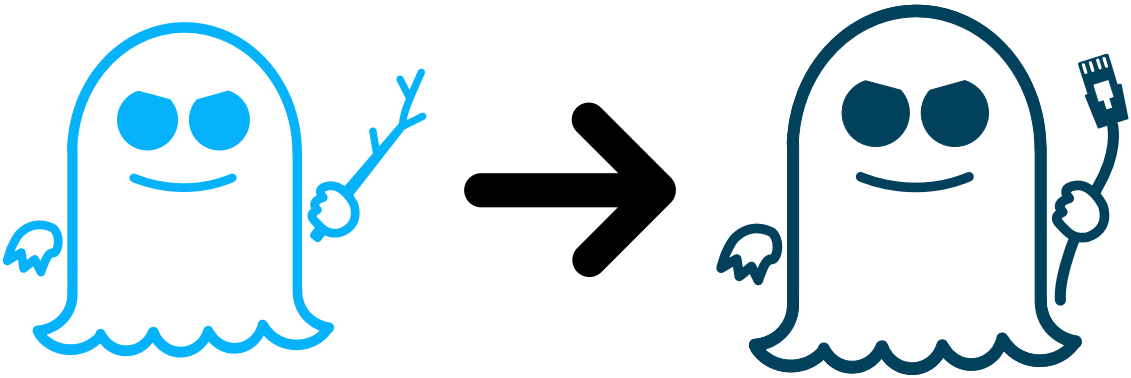
Outline



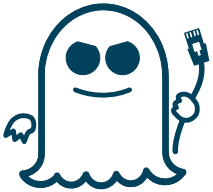
Outline



Outline

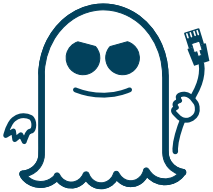


The goal



We want to build a Spectre attack which...

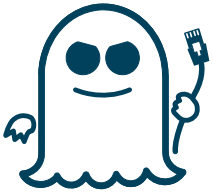
The goal



We want to build a Spectre attack which...

- is capable of leaking secrets from a remote system

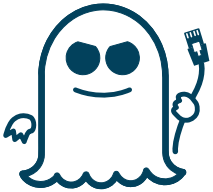
The goal



We want to build a Spectre attack which...

- is capable of leaking secrets from a remote system
- has neither physical access nor code execution on system

The goal



We want to build a Spectre attack which...

- is capable of leaking secrets from a remote system
- has neither physical access nor code execution on system
- does not rely on software vulnerabilities

CVSS v3 for CVE-2017-5753 (Spectre)

Attack Vector

Network	Adjacent Network	**Local**	Physical

CVSS v3 for CVE-2017-5753 (Spectre)

Attack Vector

Network	Adjacent Network	**Local**	Physical

Attack Complexity

Low	**High**

CVSS v3 for CVE-2017-5753 (Spectre)

Attack Vector

Network	Adjacent Network	**Local**	Physical

Attack Complexity

Low	**High**

Privilege Required

None	**Low**	High

CVSS v3 for CVE-2017-5753 (Spectre)

Attack Vector

Network	Adjacent Network	**Local**	Physical

Attack Complexity

Low	**High**

Privilege Required

None	**Low**	High

User Interaction

None	Required

```
index = 0
```

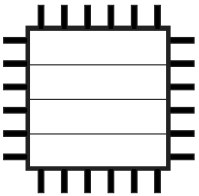
Shared Memory

	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 → `glyph[data[index]]`

else → `{}`



Memory

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

Spectre-PHT (aka Spectre Variant 1)

```
index = 0
```

Shared Memory

	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)
```

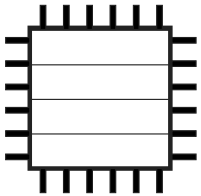
then

```
glyph[data[index]]
```

else

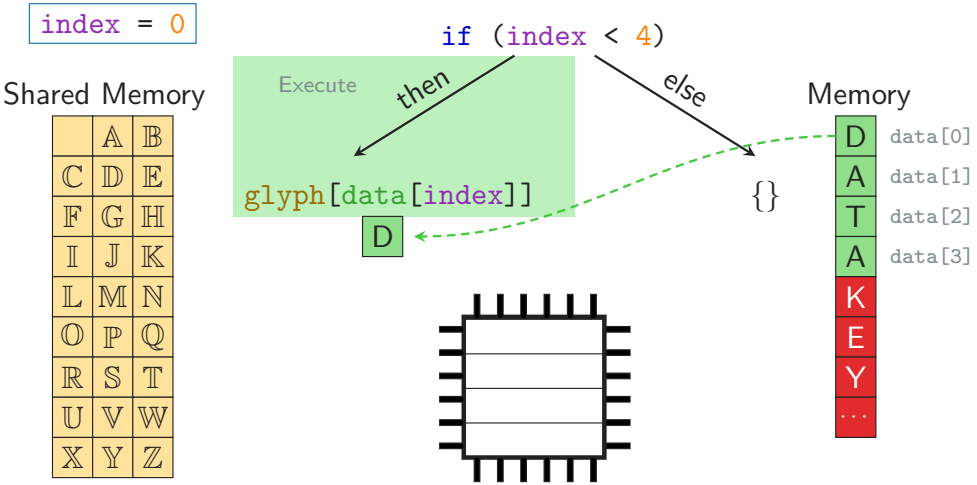
Speculate

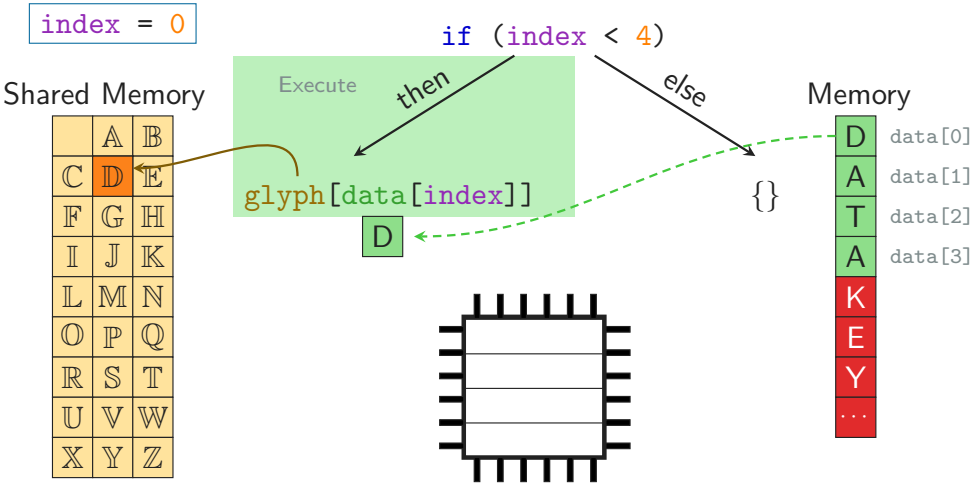
{}

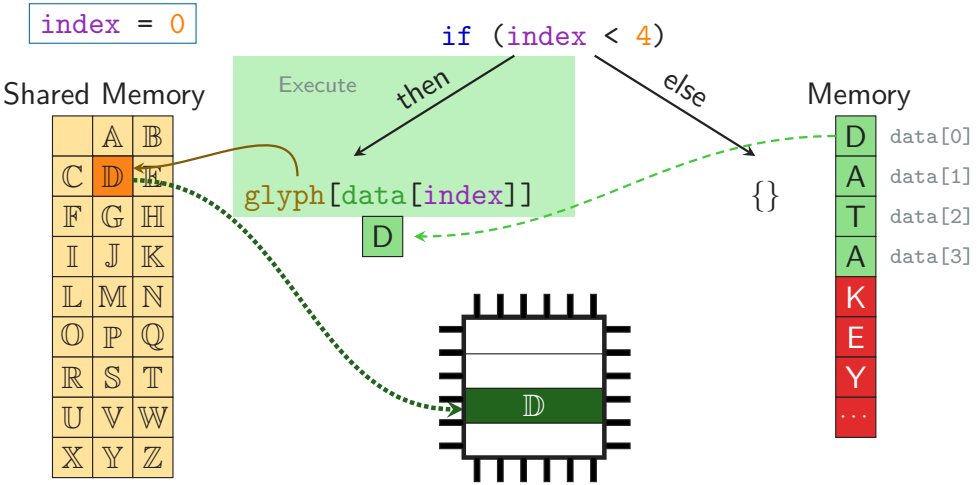


Memory

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


index = 0
if (index < 4)
then
else
Execute
glyph[data[index]]
D
{}
Shared Memory
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
A
T
A
K
E
Y
...
data[0]
data[1]
data[2]
data[3]


index = 0
if (index < 4)
then
else
Execute
Shared Memory
Memory
glyph[data[index]]
D
{}
data[0]
data[1]
data[2]
data[3]


index = 0
if (index < 4)
then
else
Execute
Shared Memory
Memory
glyph[data[index]]
D
{}
data[0]
data[1]
data[2]
data[3]

```
index = 1
```

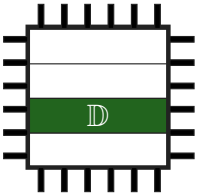
Shared Memory

	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 → `glyph[data[index]]`

else → `{}`



Memory

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

```
index = 1
```

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

Speculate

then

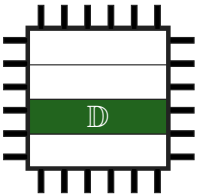
```
glyph[data[index]]
```

else

{}

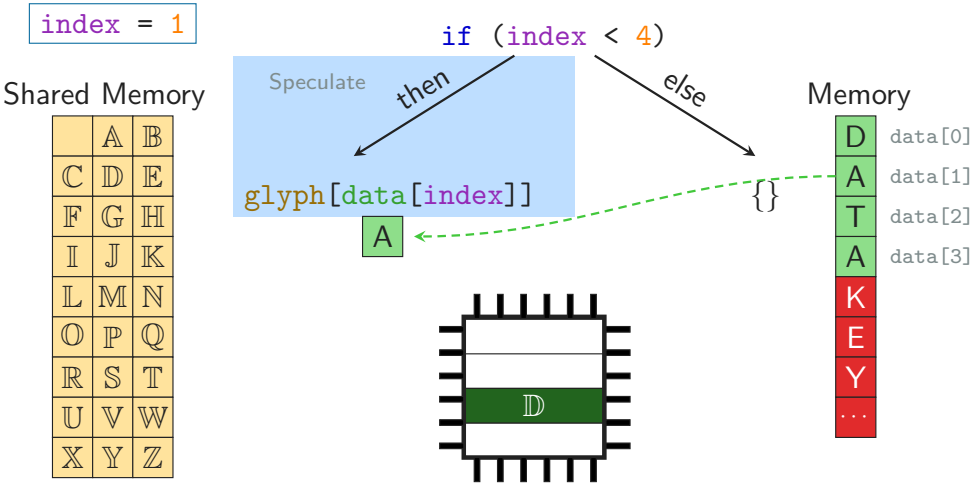
Shared Memory

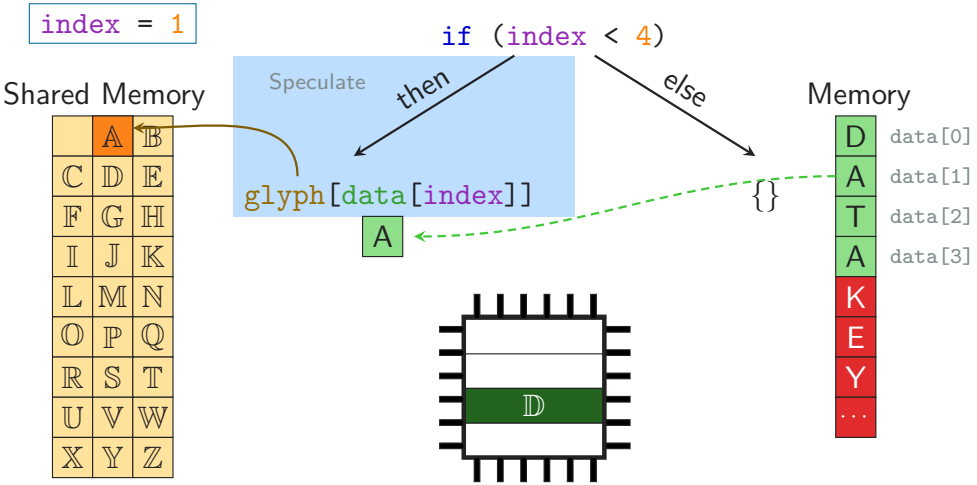
	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z



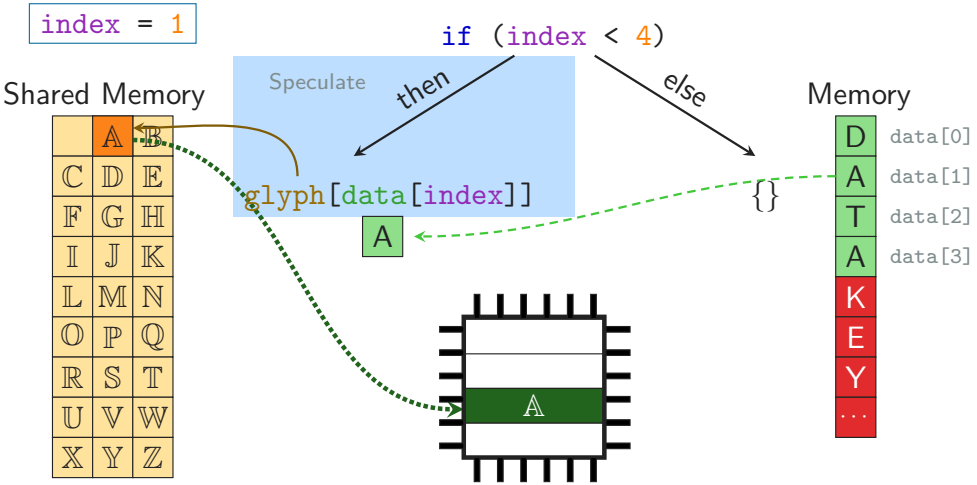
Memory

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


index = 1
Shared Memory
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)
Speculate
then
else
glyph[data[index]]
A
{}
Memory
D
A
T
A
K
E
Y
...
data[0]
data[1]
data[2]
data[3]
D


index = 1
if (index < 4)
then
else
Speculate
Shared Memory
Memory
glyph[data[index]]
A
{}
data[0]
data[1]
data[2]
data[3]
D
A
T
A
K
E
Y
...
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
D

Spectre-PHT (aka Spectre Variant 1)



```
index = 1
```

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

Execute

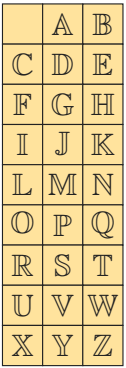
then

```
glyph[data[index]]
```

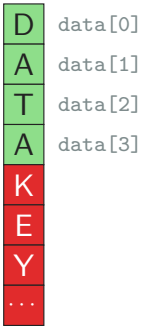
else

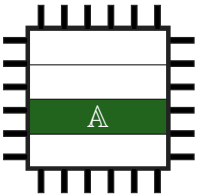
{}

Shared Memory



Memory





```
index = 2
```

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

Speculate

then

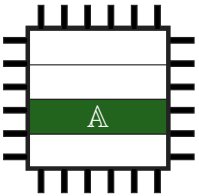
```
glyph[data[index]]
```

else

```
{}
```

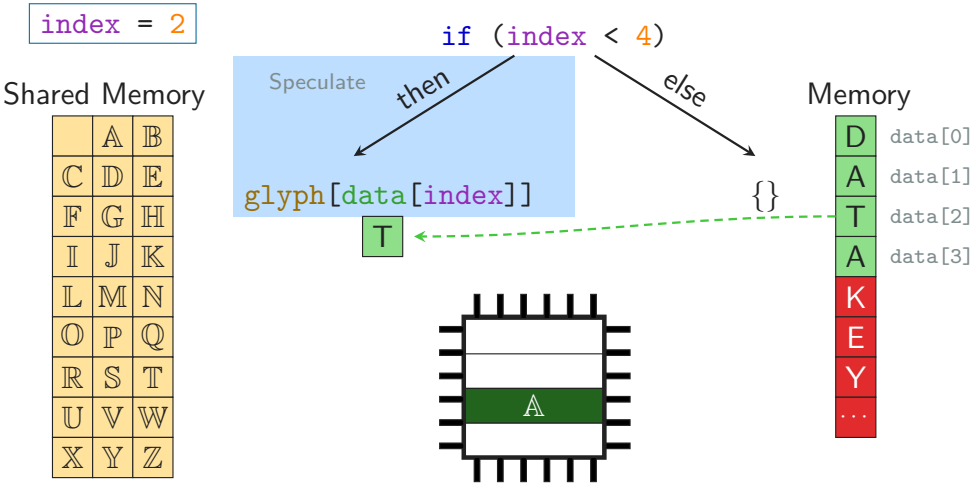
Shared Memory

	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

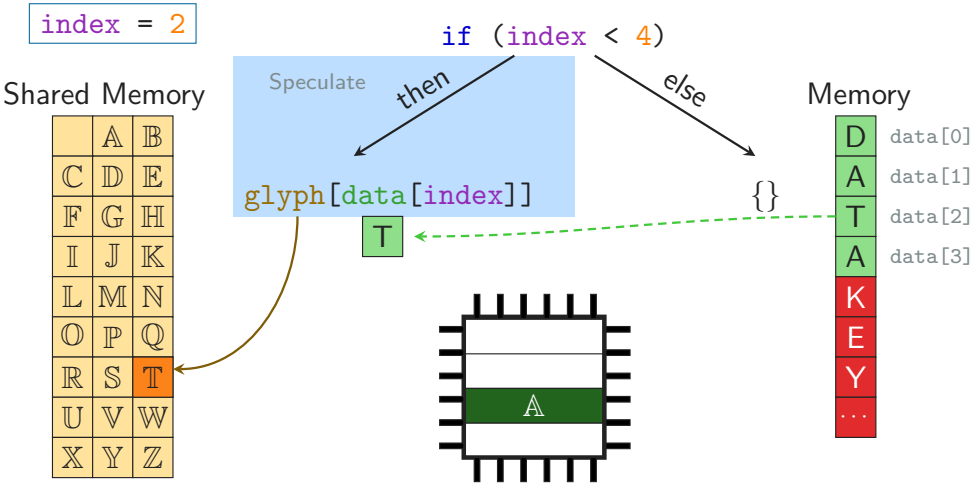


Memory

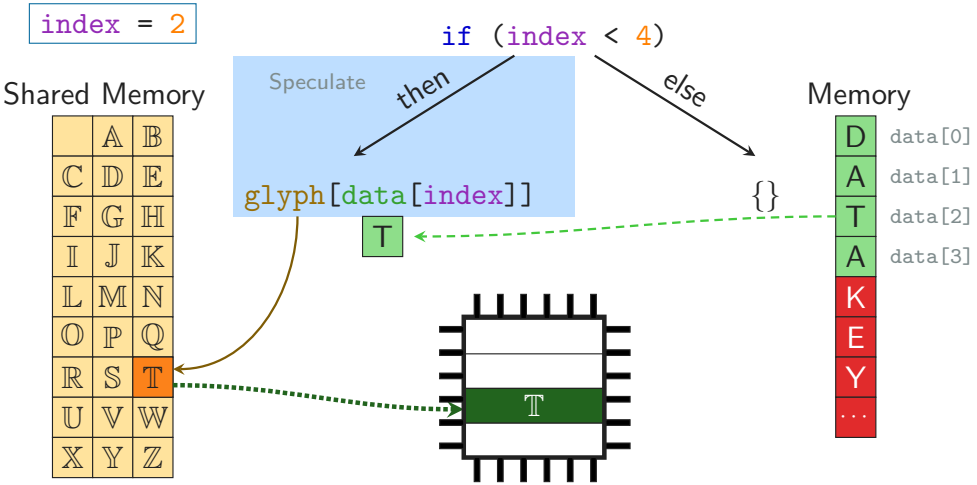
D	data[0]
A	data[1]
T	data[2]
A	data[3]
K	
E	
Y	
...	


index = 2
Shared Memory
if (index < 4)
Speculate
then
else
glyph[data[index]]
T
{}
Memory
D
A
T
A
K
E
Y
...
data[0]
data[1]
data[2]
data[3]
A

Spectre-PHT (aka Spectre Variant 1)



Spectre-PHT (aka Spectre Variant 1)



```
index = 2
```

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

Execute

then

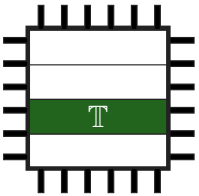
else

```
glyph[data[index]]
```

{}

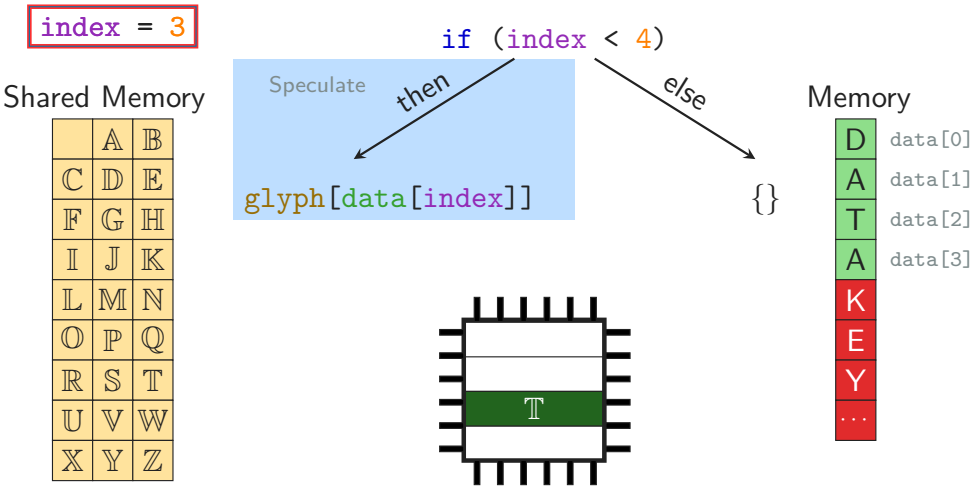
Shared Memory

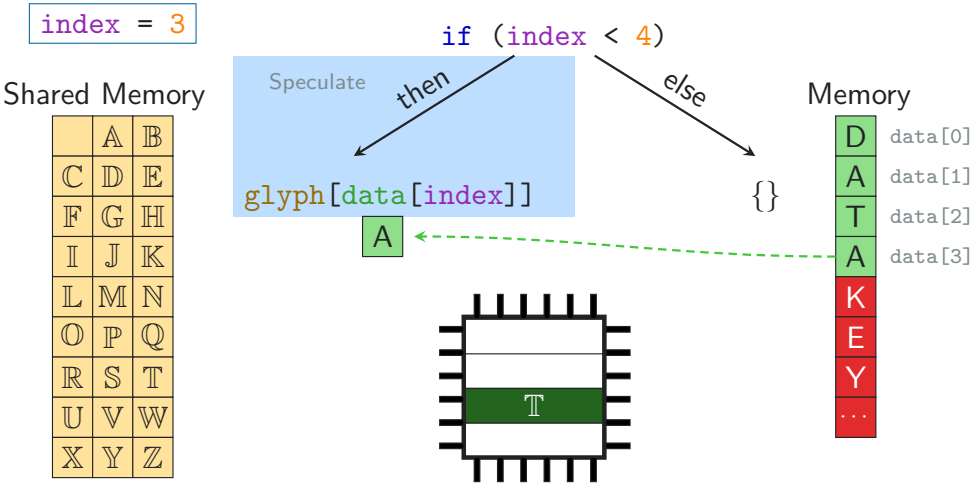
	A	B
C	D	E
F	G	H
I	J	K
L	M	N
O	P	Q
R	S	T
U	V	W
X	Y	Z

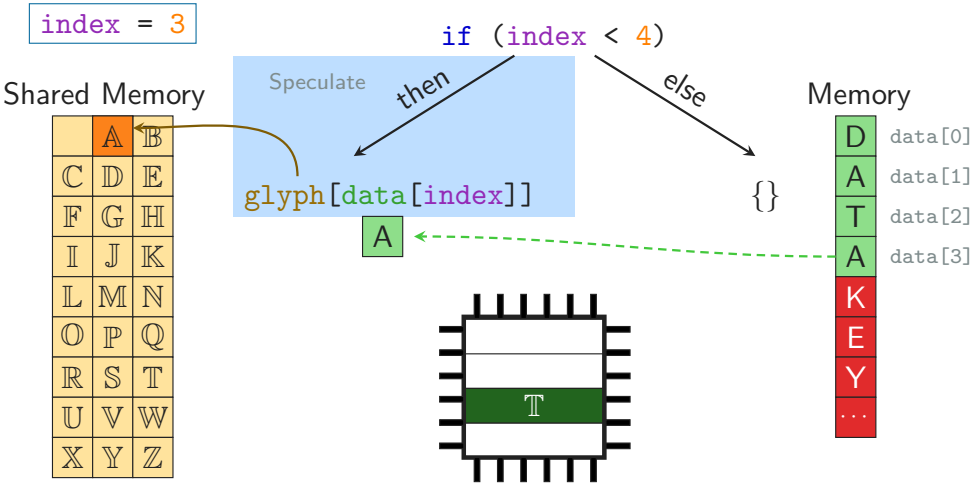


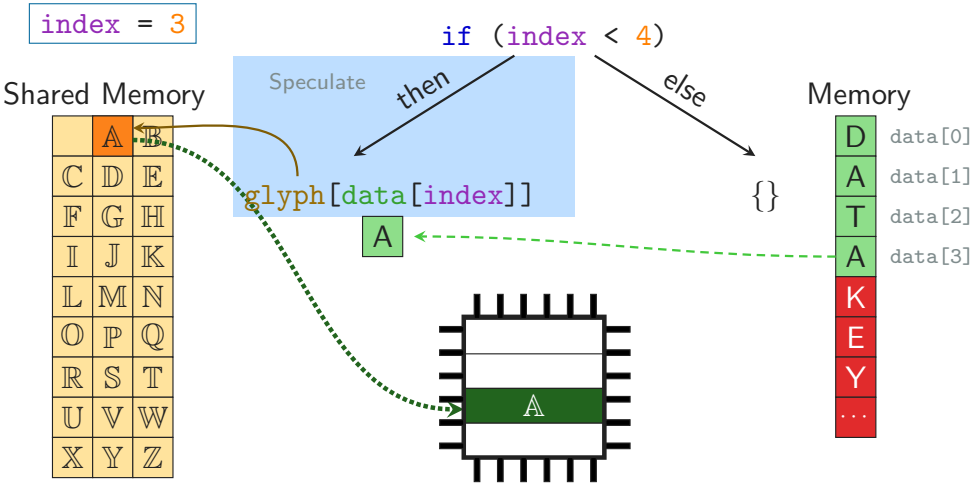
Memory

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


index = 3
if (index < 4)
Speculate
then
else
glyph[data[index]]
{}
Shared Memory
Memory
data[0]
data[1]
data[2]
data[3]
T


index = 3
Shared Memory
if (index < 4)
Speculate
then
else
glyph[data[index]]
{}
A
Memory
D
A
T
A
K
E
Y
...
data[0]
data[1]
data[2]
data[3]
𝕋


index = 3
if (index < 4)
then
else
Speculate
Shared Memory
Memory
glyph[data[index]]
{}
A
data[0]
data[1]
data[2]
data[3]
D
A
T
A
K
E
Y
...


index = 3
if (index < 4)
then
else
Speculate
Shared Memory
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
glyph[data[index]]
A
{}
Memory
D data[0]
A data[1]
T data[2]
A data[3]
K
E
Y
...
A

```
index = 3
```

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

Execute

then

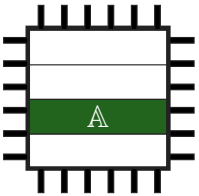
```
glyph[data[index]]
```

else

{}

Shared Memory

	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	
...	

```
index = 4
```

Shared Memory

	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)
```

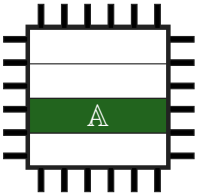
Speculate

then

```
glyph[data[index]]
```

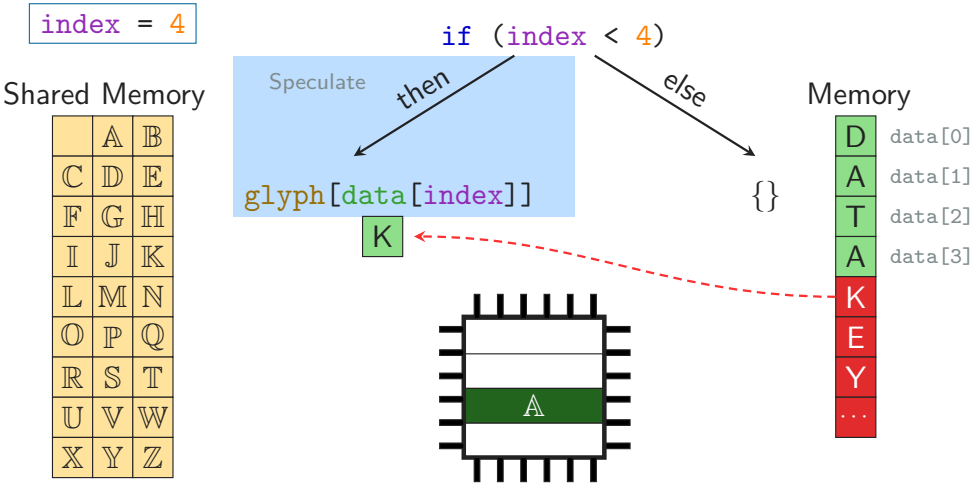
else

```
{}
```

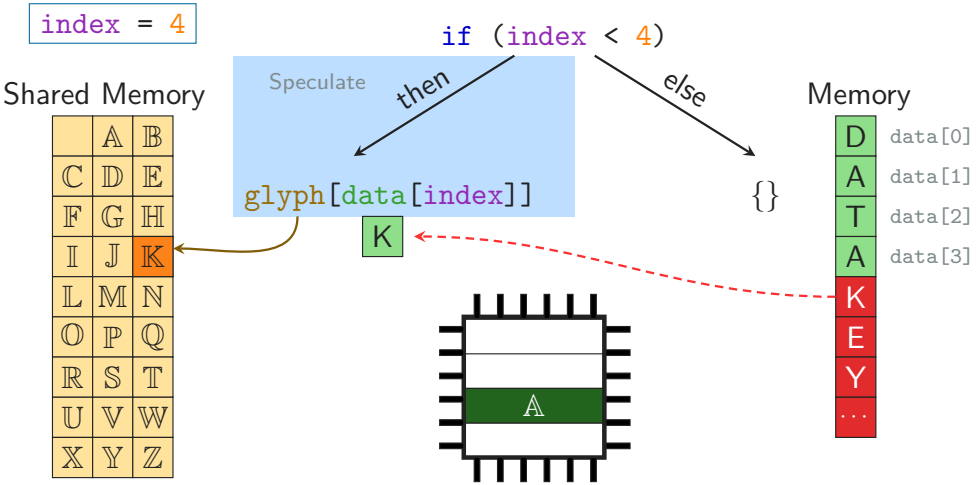


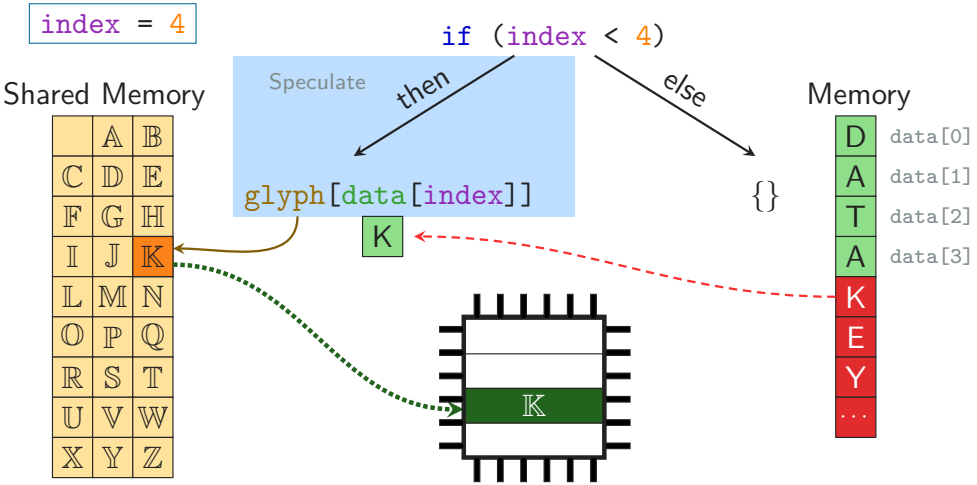
Memory

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


index = 4
if (index < 4)
Speculate
then
else
glyph[data[index]]
{}
K
Shared Memory
Memory
data[0]
data[1]
data[2]
data[3]

Spectre-PHT (aka Spectre Variant 1)




index = 4
if (index < 4)
Speculate
then
else
Shared Memory
glyph[data[index]]
K
{}
Memory
D
A
T
A
K
E
Y
...
data[0]
data[1]
data[2]
data[3]
K

Spectre-PHT (aka Spectre Variant 1)

```
index = 4
```

Shared Memory

	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

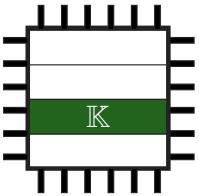
```
if (index < 4)
```

then

```
glyph[data[index]]
```

else

Execute {}



Memory

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

Spectre without code execution is complicated

Spectre without code execution is complicated

- Which branch can be exploited

Spectre without code execution is complicated

- Which branch can be exploited
- Cannot observe the cache state

Spectre without code execution is complicated

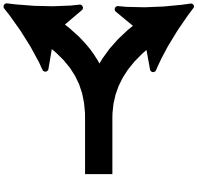
- Which branch can be exploited
- Cannot observe the cache state
- Spectre gadgets will be different

Spectre without code execution is complicated

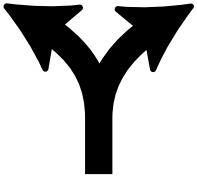
- Which branch can be exploited
- Cannot observe the cache state
- Spectre gadgets will be different
- No timing measurement on the attacked system

Spectre without code execution is complicated

- Which branch can be exploited
- Cannot observe the cache state
- Spectre gadgets will be different
- No timing measurement on the attacked system
- How to select the data to leak

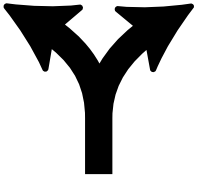


- No code can be injected



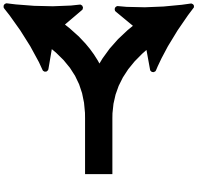
- No code can be injected
- Public interface (API) accessing data

Exploiting Branches



- No code can be injected
- Public interface (API) accessing data
- Branches in API can be mistrained remotely

Exploiting Branches



- No code can be injected
- Public interface (API) accessing data
- Branches in API can be mistrained remotely
- Attacker only calls the API via network requests

API Example

```
def check_user_privileges(user_id):
  [...]
  if user_id < len(users):
    if test_bit(privileges, user_id) == True:
      admin = True

  return SUCCESS
```

API Example

```
def check_user_privileges(user_id):
  [...]
  if user_id < len(users):
    if test_bit(privileges, user_id) == True:
      admin = True

  return SUCCESS
```

Bounds check

API Example

```
def check_user_privileges(user_id):
  [...]
  if user_id < len(users):
    if test_bit(privileges, user_id) == True:
      admin = True

  return SUCCESS
```

Bounds check

Speculative out-of-bounds read

API Example

```
def is_admin():
  return admin
```

API Example

```
def is_admin():
  return admin
```

API Example

```
def is_admin():
  return admin
```

- If bit in array was set $\rightarrow$ `admin` is cached

API Example

```
def is_admin():
  return admin
```

- If bit in array was set → `admin` is cached
- If bit was not set → `admin` is not cached

API Example

```
def is_admin():
  return admin
```

- If bit in array was set → `admin` is cached
- If bit was not set → `admin` is not cached
- Observe cache state via function execution time

Timing Measurement

- Cannot measure time directly on the attacked system

Timing Measurement

- Cannot measure time directly on the attacked system
- Network latency depends on API execution time

Timing Measurement

- Cannot measure time directly on the attacked system
- Network latency depends on API execution time

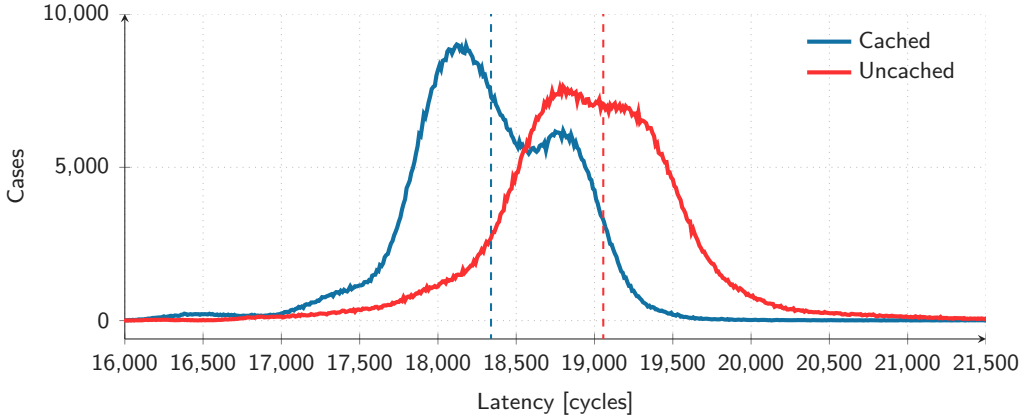
→ Measure the network roundtrip time

Timing Measurement

- Cannot measure time directly on the attacked system
- Network latency depends on API execution time

→ Measure the network roundtrip time

- Reveals whether the variable is cached


10,000
5,000
0
Cases
Cached
Uncached
16,000
16,500
17,000
17,500
18,000
18,500
19,000
19,500
20,000
20,500
21,000
21,500
Latency [cycles]

Resetting Cache State

- After measuring variable is always cached

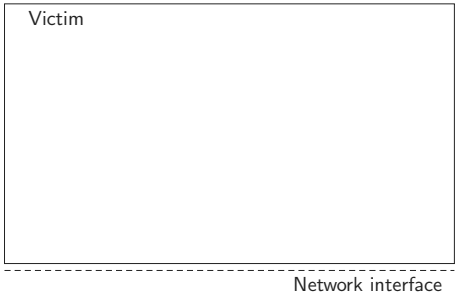
Resetting Cache State

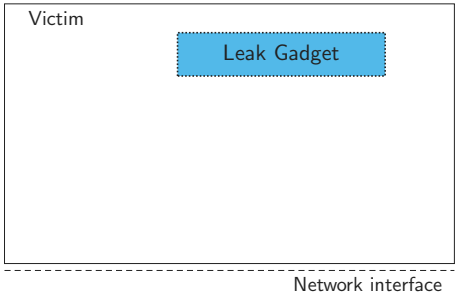
- After measuring variable is always cached
- How do we evict the variable?

- After measuring variable is always cached
- How do we evict the variable?
- Constantly evict the cache via a file download

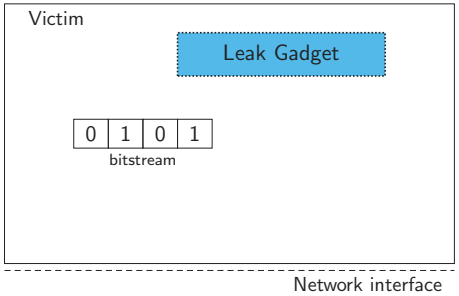
- After measuring variable is always cached
- How do we evict the variable?
- Constantly evict the cache via a file download
- Thrash+Reload → crude form of Evict+Reload

NetSpectre - The Big Picture

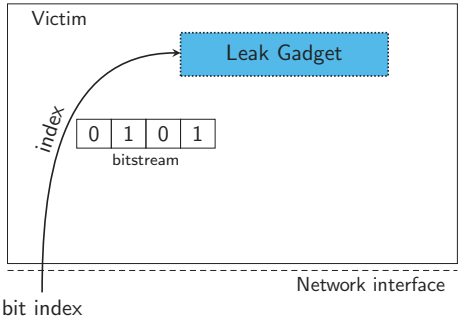





```
if (x < bitstream_length)
   if(bitstream[x])
      flag = true
```

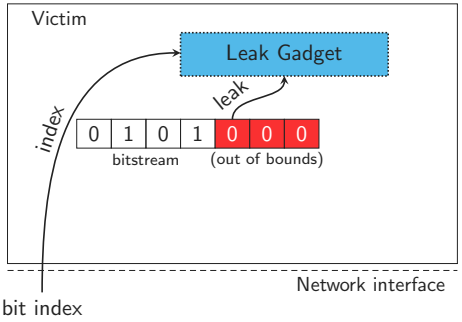



```
if (x < bitstream_length)
   if(bitstream[x])
      flag = true
```

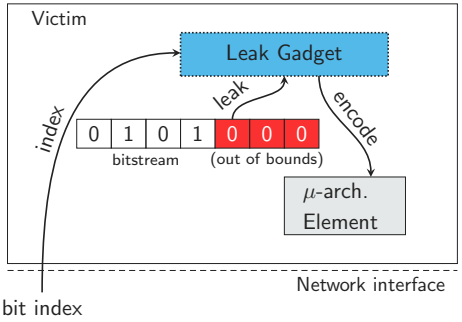



```
if (x < bitstream_length)
   if(bitstream[x])
      flag = true
```

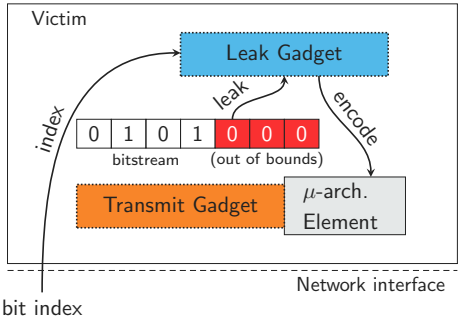
NetSpectre - The Big Picture




```
if (x < bitstream_length)
   if(bitstream[x])
      flag = true
```

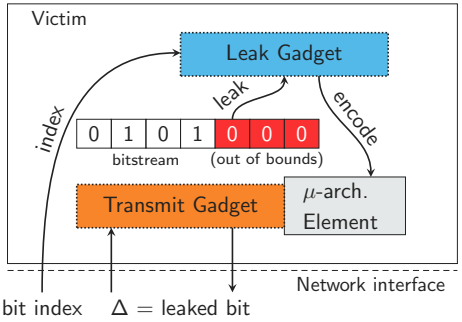



```
if (x < bitstream_length)
   if(bitstream[x])
      flag = true
```



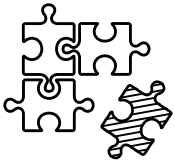

```
if (x < bitstream_length)
   if(bitstream[x])
      flag = true
```

```
send(flag)
```



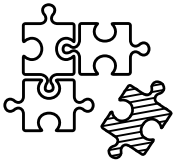

```
if (x < bitstream_length)
   if(bitstream[x])
      flag = true
```

```
send(flag)
```



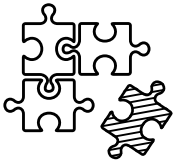
- Mistrain branch predictor with in-bounds requests

Combining Everything

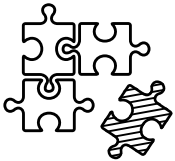


- Mistrain branch predictor with in-bounds requests
- Evict everything from cache via file download

Combining Everything



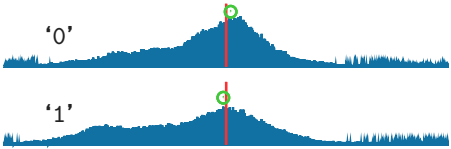
- Mistrain branch predictor with in-bounds requests
- Evict everything from cache via file download
- Leak a bit: do nothing (‘0’) or cache a memory location (‘1’)



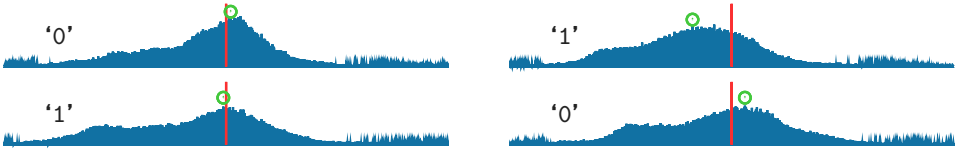
- Mistrain branch predictor with in-bounds requests
- Evict everything from cache via file download
- Leak a bit: do nothing (‘0’) or cache a memory location (‘1’)
- Measure function latency which uses the memory location

Leaking byte 'd' (0)

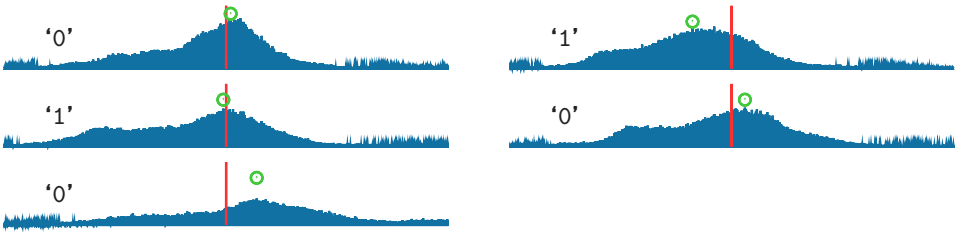
Leaking byte ’d’ (01)



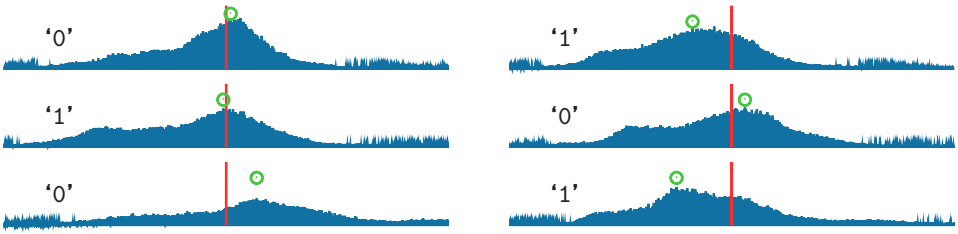
Leaking byte ’d’ (011)



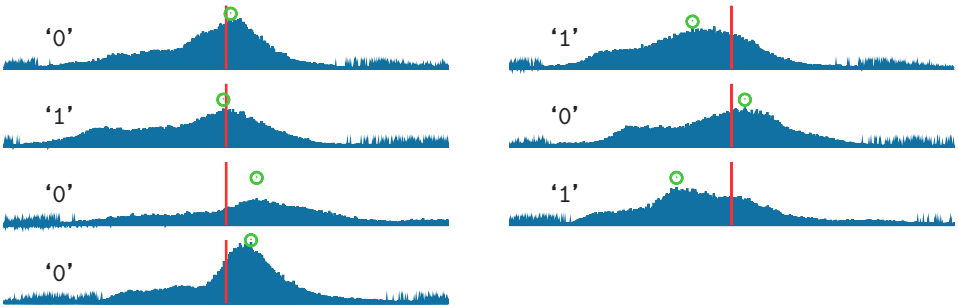
Leaking byte ’d’ (0110)



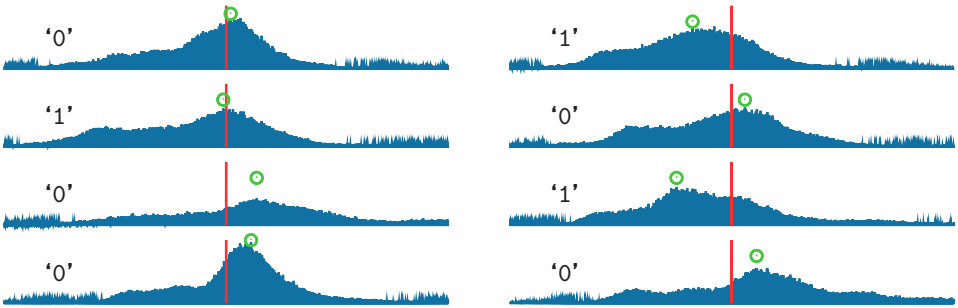
Leaking byte ’d’ (01100)



Leaking byte ’d’ (011001)



Leaking byte ’d’ (0110010)



Leaking byte ’d’ (01100100)

Attack Targets

- Several possible attack targets

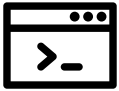
- Several possible attack targets
- Different impacts depending on target

- Several possible attack targets
- Different impacts depending on target

Web/FTP Servers
(user gadget)

- Several possible attack targets
- Different impacts depending on target

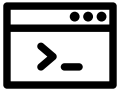
Web/FTP Servers
(user gadget)



SSH Daemons
(user gadget)

- Several possible attack targets
- Different impacts depending on target

Web/FTP Servers
(user gadget)



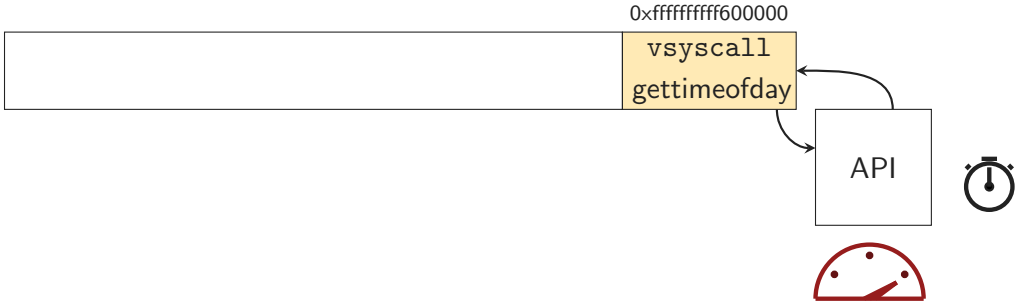
SSH Daemons
(user gadget)

Network Drivers
(kernel gadget)

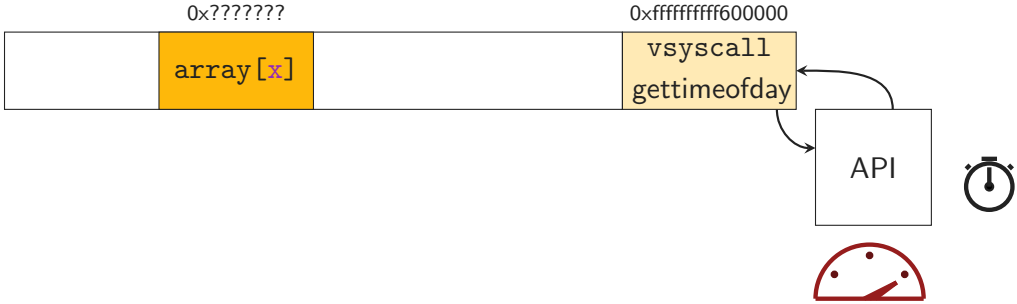
- No indirection, simple array access

- No indirection, simple array access

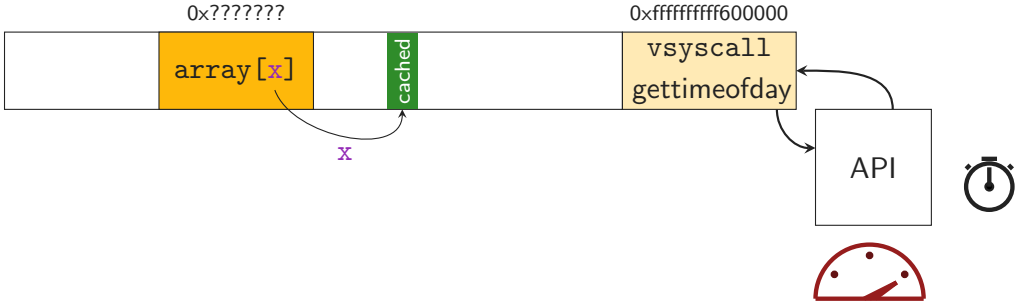
```
if (x < array_length)
  y = array[x];
```


0xffffffffff600000
vsyscall
gettimeofday
API

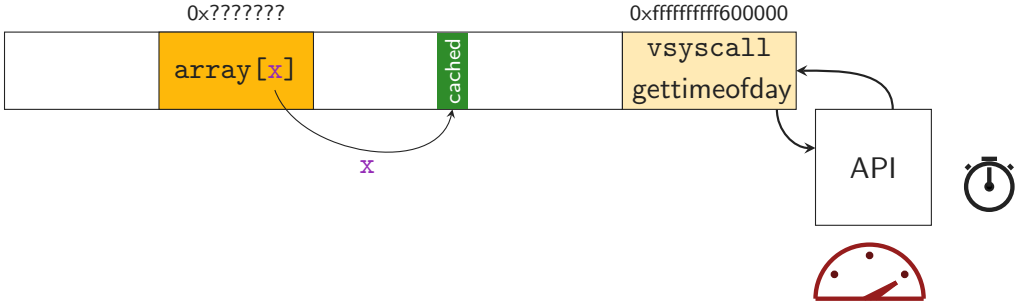
ASLR Break Idea



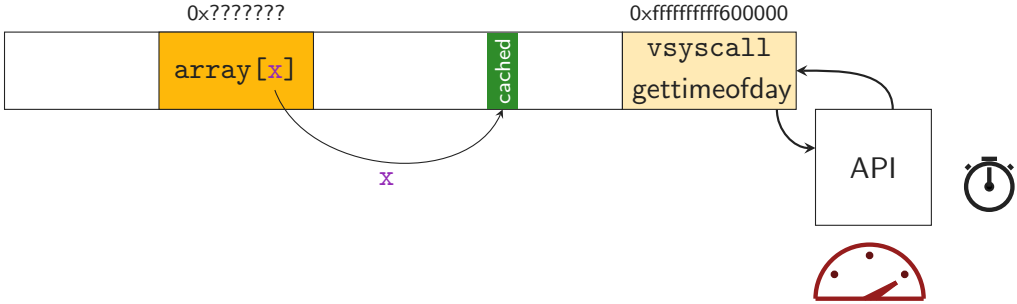
ASLR Break Idea



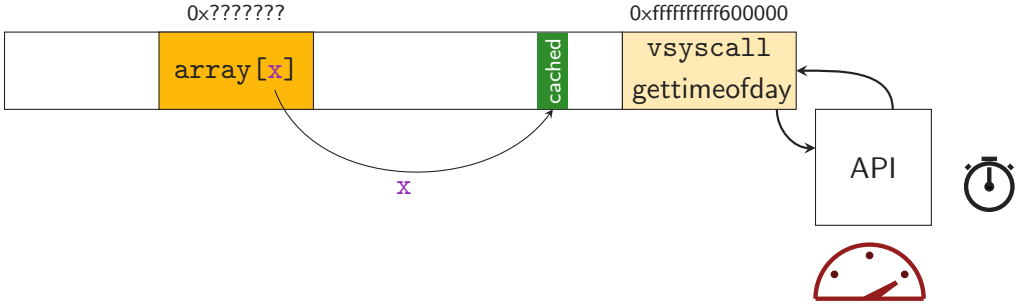
ASLR Break Idea



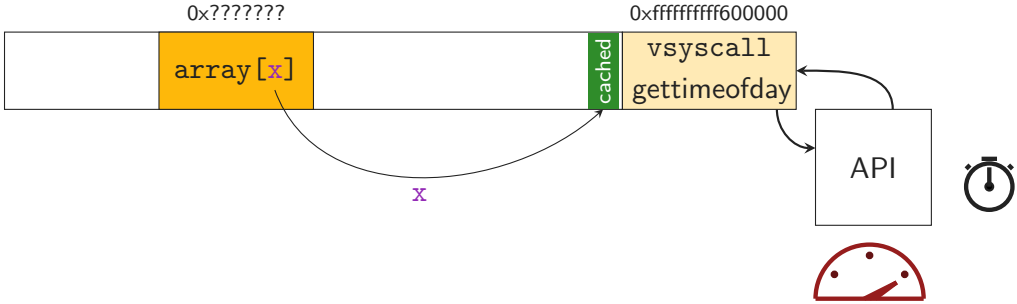
ASLR Break Idea



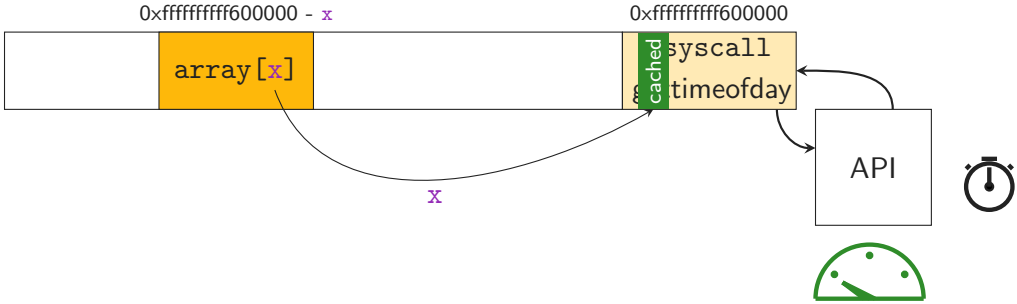
ASLR Break Idea



ASLR Break Idea



ASLR Break Idea



- 256-bit instructions need a lot of power

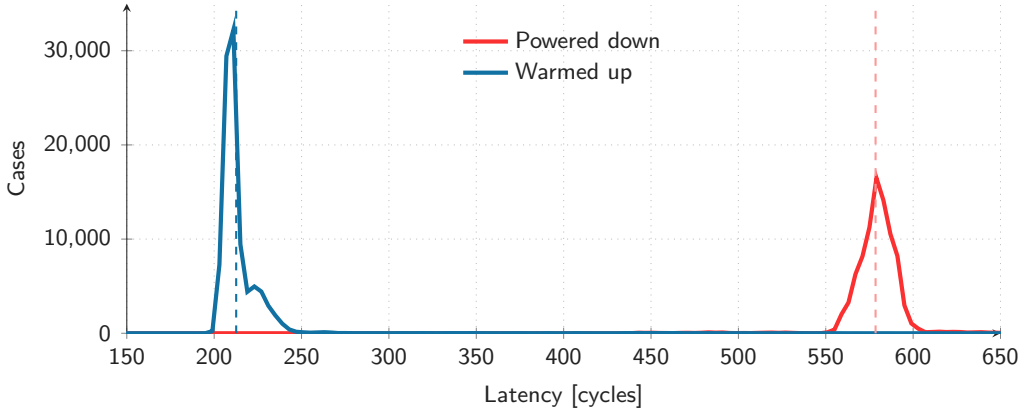
- 256-bit instructions need a lot of power
  - → On Intel, disabled by default, enabled on first use

- 256-bit instructions need a lot of power
  - → On Intel, disabled by default, enabled on first use
- Requires some time to power up

- 256-bit instructions need a lot of power
  - → On Intel, disabled by default, enabled on first use
- Requires some time to power up
- Measure execution time of AVX instruction

- 256-bit instructions need a lot of power
  - → On Intel, disabled by default, enabled on first use
- Requires some time to power up
- Measure execution time of AVX instruction

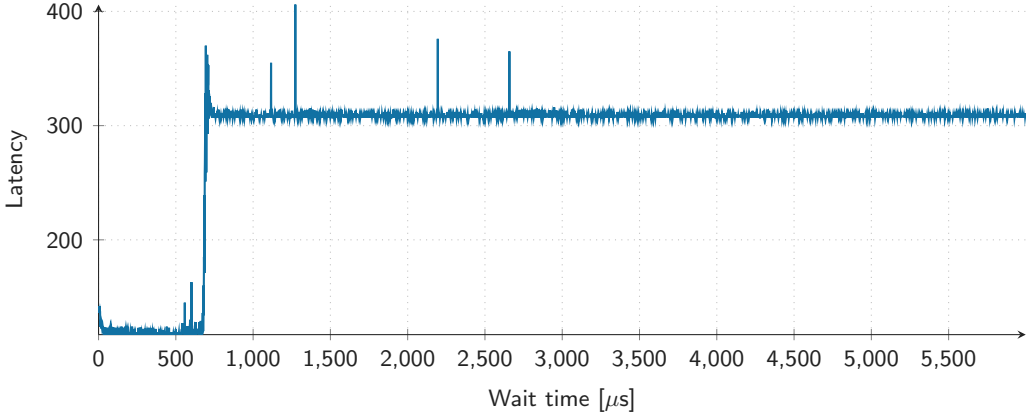
→ Leak timing information

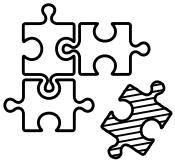

Powered down
Warmed up
Cases
30,000
20,000
10,000
0
150
200
250
300
350
400
450
500
550
600
650
Latency [cycles]

AVX Leak Gadget

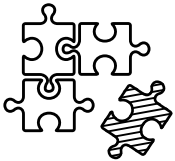
```
if (x < bitstream_length)
    if(bitstream[x])
        _mm256_instruction();
```

AVX Cooldown

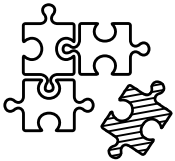




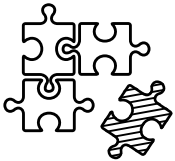
1. Mistrain branch predictor with in-bounds requests



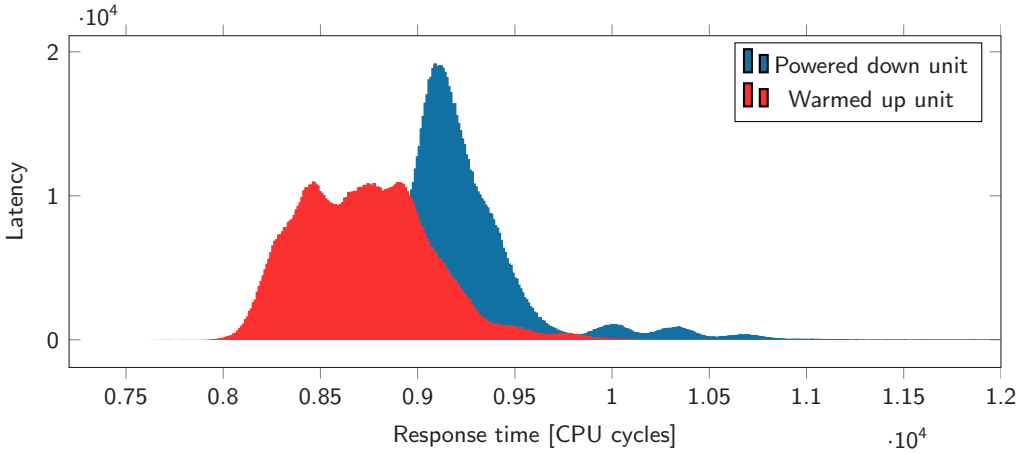
1. Mistrain branch predictor with in-bounds requests
2. Wait for AVX unit to power off (1ms)



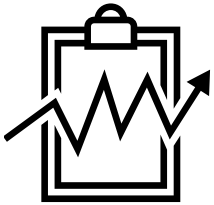
1. Mistrain branch predictor with in-bounds requests
2. Wait for AVX unit to power off (1ms)
3. Leak a bit: do nothing (‘0’) or power AVX unit (‘1’)



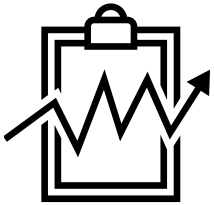
1. Mistrain branch predictor with in-bounds requests
2. Wait for AVX unit to power off (1ms)
3. Leak a bit: do nothing (‘0’) or power AVX unit (‘1’)
4. Measure function latency which uses AVX instruction


·10^4
2
1
0
Latency
Powered down unit
Warmed up unit
0.75
0.8
0.85
0.9
0.95
1
1.05
1.1
1.15
1.2
Response time [CPU cycles]
·10^4

- Local Network (1 000 000 measurements/bit)

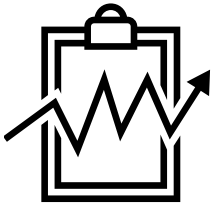


- Local Network (1 000 000 measurements/bit)



30 min/byte

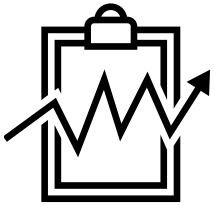
- Local Network (1 000 000 measurements/bit)



30 min/byte

8 min/byte

Attack Results

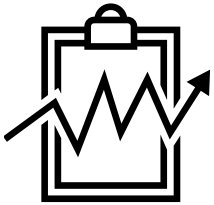


- Local Network (1 000 000 measurements/bit)

30 min/byte

8 min/byte

- Cloud (20 000 000 measurements/bit)



- Local Network (1 000 000 measurements/bit)

30 min/byte

8 min/byte

- Cloud (20 000 000 measurements/bit)

1 h/bit

- NetSpectre requires a fast and stable network connection

Limitations

- NetSpectre requires a fast and stable network connection
  - Local networks

- NetSpectre requires a fast and stable network connection
  - Local networks
  - Data centers (VM to VM attack)

Limitations

- NetSpectre requires a fast and stable network connection
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)

- NetSpectre requires a fast and stable network connection
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)

$\rightarrow$ possible in the near future?

Limitations

- NetSpectre requires a fast and stable network connection
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)

$\rightarrow$ possible in the near future?

- Attack speeds can be drastically improved

Limitations

- NetSpectre requires a fast and stable network connection
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)

$\rightarrow$ possible in the near future?

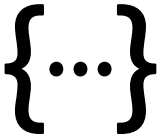
- Attack speeds can be drastically improved
  - Better signal processing/filtering

Limitations

- NetSpectre requires a fast and stable network connection
  - Local networks
  - Data centers (VM to VM attack)
- Internet speeds improve (e.g., fiber, 5G)

$\rightarrow$ possible in the near future?

- Attack speeds can be drastically improved
  - Better signal processing/filtering
  - Dedicated measuring hardware



- Speculative execution leaks secrets without exploiting bugs
- Spectre attacks are not limited to local attackers
- Spectre attacks have a larger impact than assumed

ESORICS 2019

NetSpectre

Read Arbitrary Memory over Network

Michael Schwarz[1]
Martin Schwarzl[1]
Moritz Lipp[1]
Jon Masters[2]
Daniel Gruss[1]

[1]Graz University of Technology [2]Red Hat