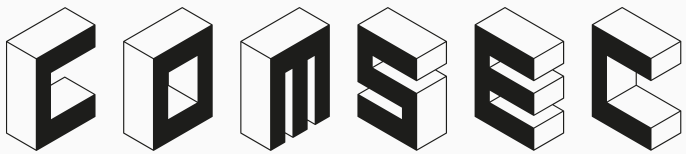
MileSan

Detecting Exploitable Microarchitectural Leakage via Differential Hardware-Software Taint Tracking

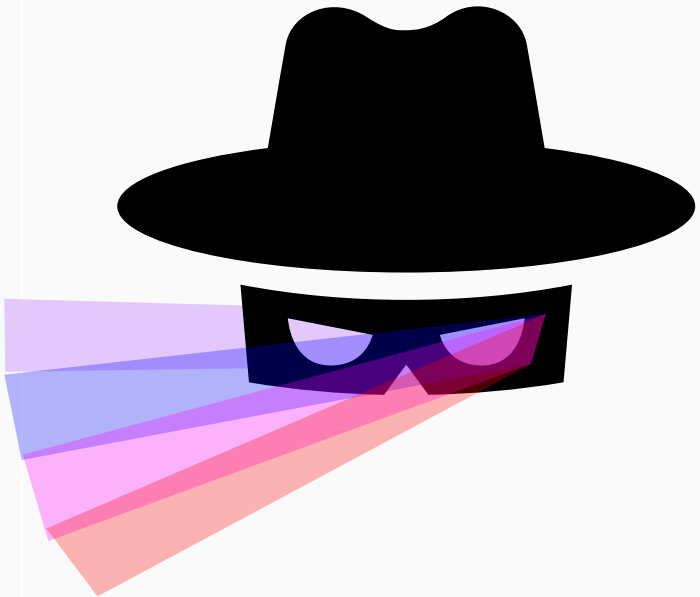
Tobias Kovats, Flavien Solt*, Katharina Ceesay-Seitz, Kaveh Razavi

Presented at CCS'25

ETH Zurich, *UC Berkeley



MileSan

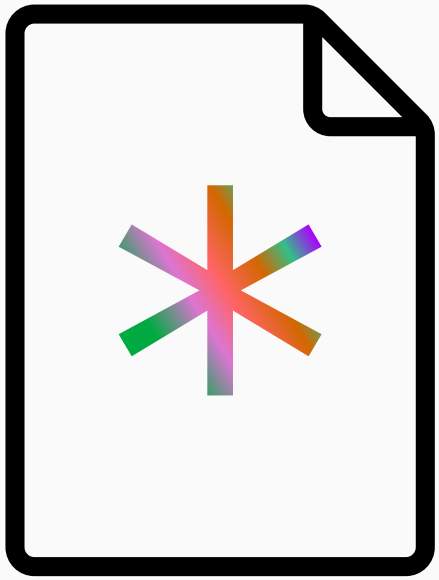


detects arbitrary
exploitable leakage in RTL

MileSan

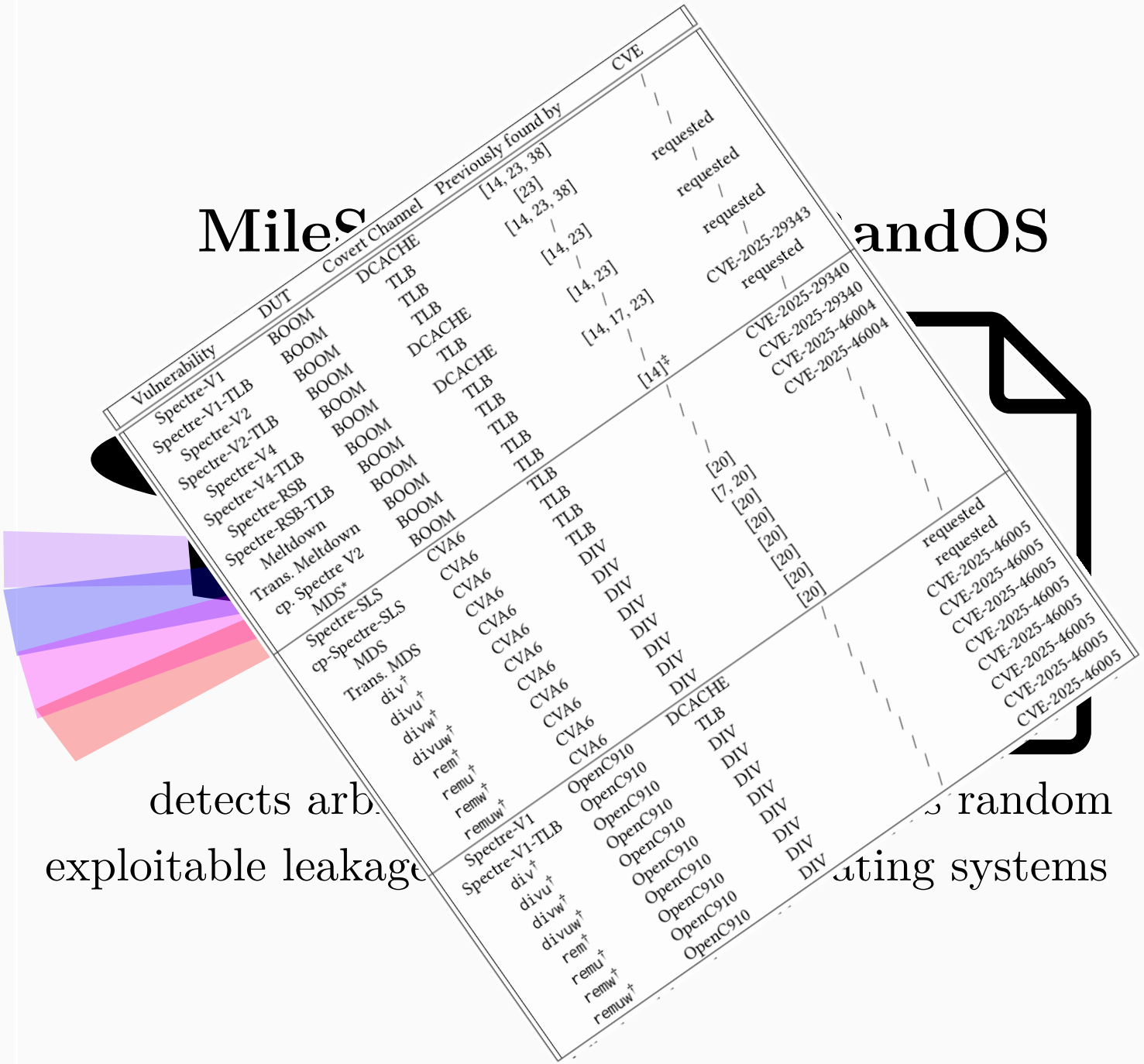
detects arbitrary
exploitable leakage in RTL

RandOS



generates random
operating systems

MileS andOS

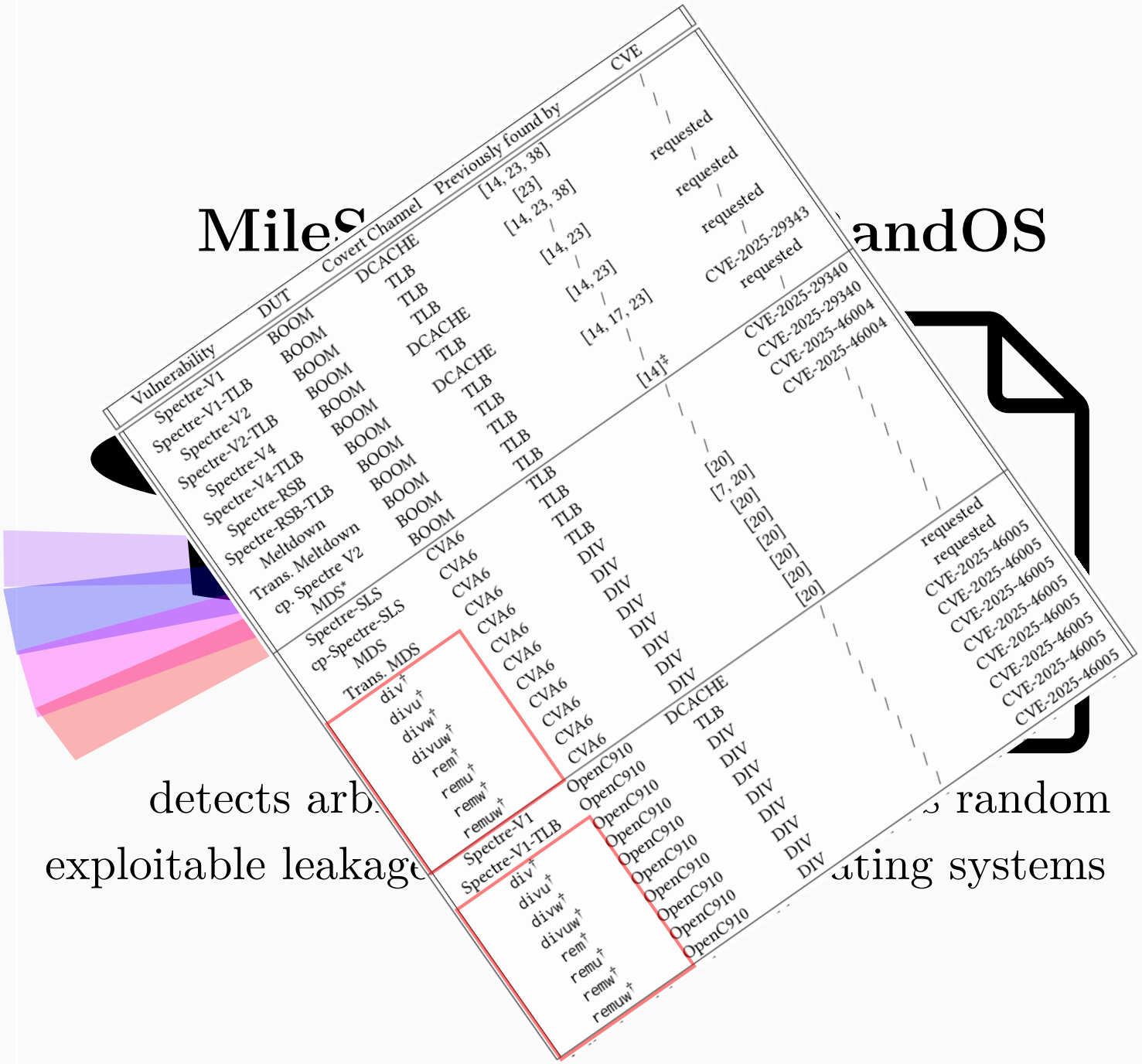


Vulnerability	DUT	Covert Channel	Previously found by	CVE
Spectre-V1	BOOM	DCACHE	[14, 23, 38]	/
Spectre-V1-TLB	BOOM	TLB	[23]	/
Spectre-V2	BOOM	TLB	[14, 23, 38]	requested
Spectre-V2-TLB	BOOM	TLB	/	/
Spectre-V4	BOOM	DCACHE	[14, 23]	requested
Spectre-V4-TLB	BOOM	TLB	/	/
Spectre-RSB	BOOM	DCACHE	[14, 23]	requested
Spectre-RSB-TLB	BOOM	TLB	/	/
Meltdown	BOOM	TLB	[14, 17, 23]	CVE-2025-29343
Trans. Meltdown	BOOM	TLB	/	requested
cp. Spectre V2	BOOM	TLB	[14]‡	/
MDS*	BOOM	TLB	/	CVE-2025-29340
Spectre-SLS	CVA6	TLB	[20]	CVE-2025-29340
cp-Spectre-SLS	CVA6	TLB	[7, 20]	CVE-2025-46004
MDS	CVA6	TLB	[20]	CVE-2025-46004
Trans. MDS	CVA6	DIV	[20]	/
div†	CVA6	DIV	[20]	/
divu†	CVA6	DIV	[20]	/
divw†	CVA6	DIV	[20]	/
divuw†	CVA6	DIV	[20]	/
rem†	CVA6	DIV	[20]	/
remu†	CVA6	DIV	[20]	/
remw†	CVA6	DIV	[20]	/
remuw†	CVA6	DIV	[20]	/
Spectre-V1	OpenC910	DCACHE	/	requested
Spectre-V1-TLB	OpenC910	TLB	/	requested
div†	OpenC910	DIV	/	CVE-2025-46005
divu†	OpenC910	DIV	/	CVE-2025-46005
divw†	OpenC910	DIV	/	CVE-2025-46005
divuw†	OpenC910	DIV	/	CVE-2025-46005
rem†	OpenC910	DIV	/	CVE-2025-46005
remu†	OpenC910	DIV	/	CVE-2025-46005
remw†	OpenC910	DIV	/	CVE-2025-46005
remuw†	OpenC910	DIV	/	CVE-2025-46005

detects arb random

exploitable leakage ating systems

MileS andOS

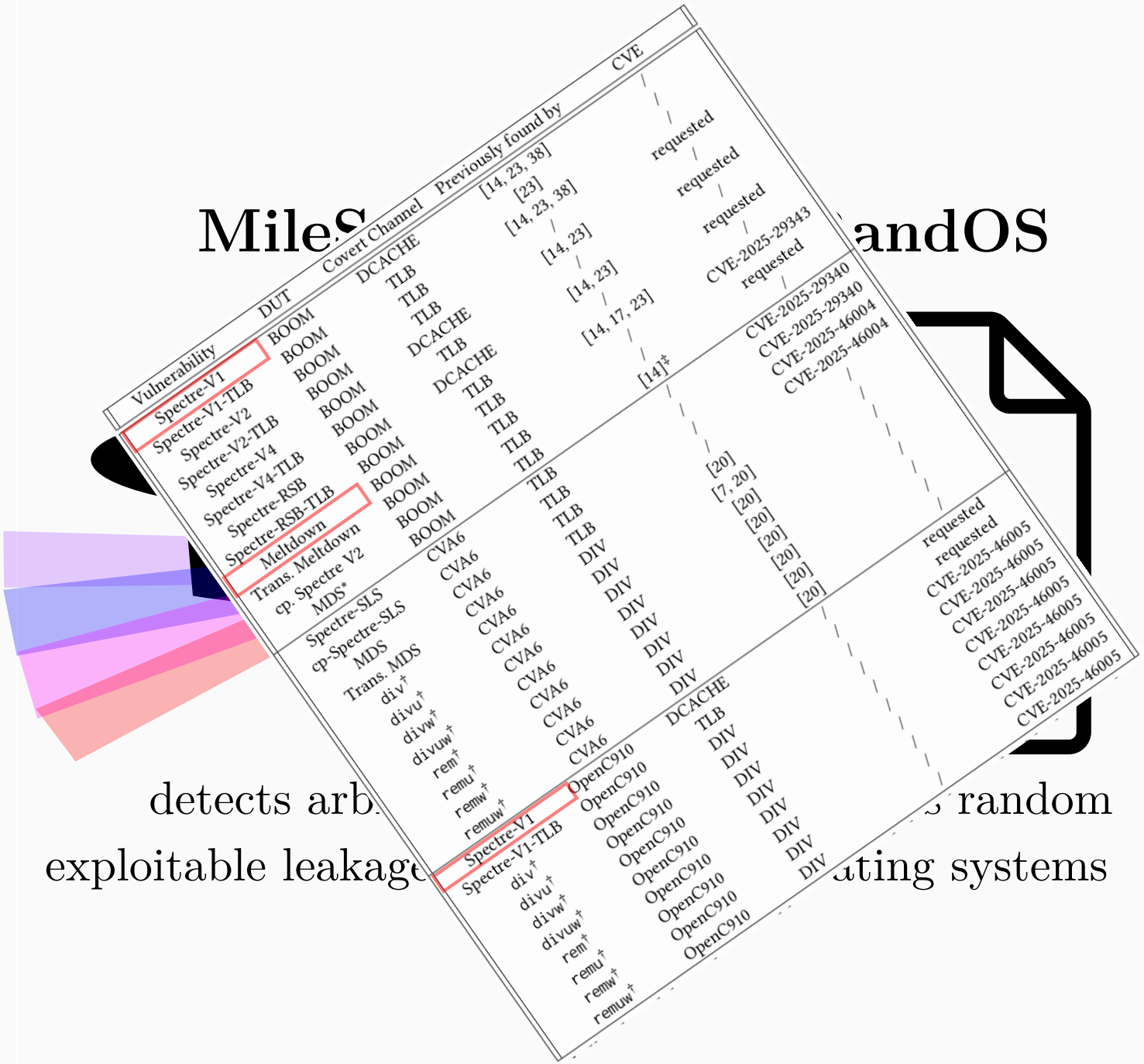


Vulnerability	DUT	Covert Channel	Previously found by	CVE
Spectre-V1	BOOM	DCACHE	[14, 23, 38]	/
Spectre-V1-TLB	BOOM	TLB	[23]	/
Spectre-V2	BOOM	TLB	[14, 23, 38]	requested
Spectre-V2-TLB	BOOM	TLB	/	/
Spectre-V4	BOOM	DCACHE	[14, 23]	requested
Spectre-V4-TLB	BOOM	TLB	/	/
Spectre-RSB	BOOM	DCACHE	[14, 23]	requested
Spectre-RSB-TLB	BOOM	TLB	/	/
Meltdown	BOOM	TLB	[14, 17, 23]	CVE-2025-29343
Trans. Meltdown	BOOM	TLB	/	requested
cp. Spectre V2	BOOM	TLB	[14]‡	/
MDS*	BOOM	TLB	/	CVE-2025-29340
Spectre-SLS	CVA6	TLB	[20]	CVE-2025-29340
cp-Spectre-SLS	CVA6	TLB	[7, 20]	CVE-2025-46004
MDS	CVA6	TLB	[20]	CVE-2025-46004
Trans. MDS	CVA6	DIV	[20]	/
div†	CVA6	DIV	[20]	/
divu†	CVA6	DIV	[20]	/
divw†	CVA6	DIV	[20]	/
divuw†	CVA6	DIV	[20]	/
rem†	CVA6	DIV	[20]	/
remu†	CVA6	DIV	[20]	/
remw†	CVA6	DIV	[20]	/
remuw†	CVA6	DIV	[20]	/
Spectre-V1	CVA6	DCACHE	/	requested
Spectre-V1-TLB	OpenC910	TLB	/	requested
div†	OpenC910	DIV	/	CVE-2025-46005
divu†	OpenC910	DIV	/	CVE-2025-46005
divw†	OpenC910	DIV	/	CVE-2025-46005
divuw†	OpenC910	DIV	/	CVE-2025-46005
rem†	OpenC910	DIV	/	CVE-2025-46005
remu†	OpenC910	DIV	/	CVE-2025-46005
remw†	OpenC910	DIV	/	CVE-2025-46005
remuw†	OpenC910	DIV	/	CVE-2025-46005

detects arb random

exploitable leakage ating systems

MileS andOS

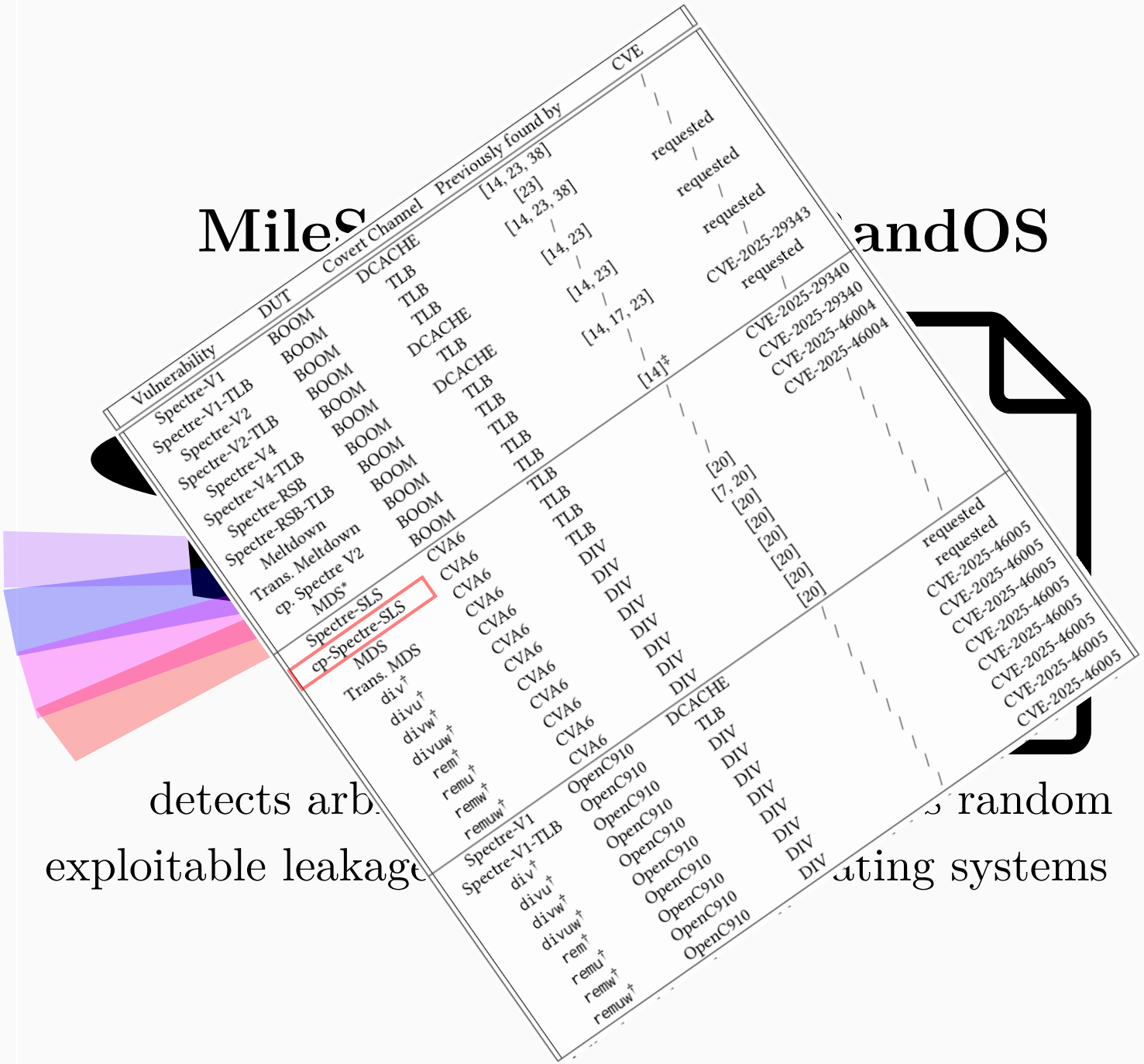


Vulnerability	DUT	Covert Channel	Previously found by	CVE
Spectre-V1	BOOM	DCACHE	[14, 23, 38]	/
Spectre-V1-TLB	BOOM	TLB	[23]	/
Spectre-V2	BOOM	TLB	[14, 23, 38]	requested
Spectre-V2-TLB	BOOM	TLB	/	/
Spectre-V4	BOOM	DCACHE	[14, 23]	requested
Spectre-V4-TLB	BOOM	TLB	/	/
Spectre-RSB	BOOM	DCACHE	[14, 23]	requested
Spectre-RSB-TLB	BOOM	TLB	/	/
Meltdown	BOOM	TLB	[14, 17, 23]	CVE-2025-29343
Trans. Meltdown	BOOM	TLB	/	requested
cp. Spectre V2	BOOM	TLB	[14]‡	/
MDS*	BOOM	TLB	/	CVE-2025-29340
Spectre-SLS	CVA6	TLB	[20]	CVE-2025-29340
cp-Spectre-SLS	CVA6	TLB	[7, 20]	CVE-2025-46004
MDS	CVA6	TLB	[20]	CVE-2025-46004
Trans. MDS	CVA6	DIV	[20]	/
div†	CVA6	DIV	[20]	/
divu†	CVA6	DIV	[20]	/
divw†	CVA6	DIV	[20]	/
divuw†	CVA6	DIV	[20]	/
rem†	CVA6	DIV	[20]	/
remu†	CVA6	DIV	[20]	/
remw†	CVA6	DIV	[20]	/
remuw†	CVA6	DIV	[20]	/
Spectre-V1	OpenC910	DCACHE	/	requested
Spectre-V1-TLB	OpenC910	TLB	/	requested
div†	OpenC910	DIV	/	CVE-2025-46005
divu†	OpenC910	DIV	/	CVE-2025-46005
divw†	OpenC910	DIV	/	CVE-2025-46005
divuw†	OpenC910	DIV	/	CVE-2025-46005
rem†	OpenC910	DIV	/	CVE-2025-46005
remu†	OpenC910	DIV	/	CVE-2025-46005
remw†	OpenC910	DIV	/	CVE-2025-46005
remuw†	OpenC910	DIV	/	CVE-2025-46005

detects arb random

exploitable leakage ating systems

MileS andOS



Vulnerability	DUT	Covert Channel	Previously found by	CVE
Spectre-V1	BOOM	DCACHE	[14, 23, 38]	/
Spectre-V1-TLB	BOOM	TLB	[23]	/
Spectre-V2	BOOM	TLB	[14, 23, 38]	requested
Spectre-V2-TLB	BOOM	TLB	/	/
Spectre-V4	BOOM	DCACHE	[14, 23]	requested
Spectre-V4-TLB	BOOM	TLB	/	/
Spectre-RSB	BOOM	DCACHE	[14, 23]	requested
Spectre-RSB-TLB	BOOM	TLB	/	/
Meltdown	BOOM	TLB	[14, 17, 23]	CVE-2025-29343
Trans. Meltdown	BOOM	TLB	/	requested
cp. Spectre V2	BOOM	TLB	[14]‡	/
MDS*	BOOM	TLB	/	CVE-2025-29340
Spectre-SLS	CVA6	TLB	[20]	CVE-2025-29340
cp-Spectre-SLS	CVA6	TLB	[7, 20]	CVE-2025-46004
MDS	CVA6	TLB	[20]	CVE-2025-46004
Trans. MDS	CVA6	DIV	[20]	/
div†	CVA6	DIV	[20]	/
divu†	CVA6	DIV	[20]	/
divw†	CVA6	DIV	[20]	/
divuw†	CVA6	DIV	[20]	/
rem†	CVA6	DIV	[20]	/
remu†	CVA6	DIV	[20]	/
remw†	CVA6	DIV	[20]	/
remuw†	CVA6	DIV	[20]	/
Spectre-V1	OpenC910	DCACHE	/	requested
Spectre-V1-TLB	OpenC910	TLB	/	requested
div†	OpenC910	DIV	/	CVE-2025-46005
divu†	OpenC910	DIV	/	CVE-2025-46005
divw†	OpenC910	DIV	/	CVE-2025-46005
divuw†	OpenC910	DIV	/	CVE-2025-46005
rem†	OpenC910	DIV	/	CVE-2025-46005
remu†	OpenC910	DIV	/	CVE-2025-46005
remw†	OpenC910	DIV	/	CVE-2025-46005
remuw†	OpenC910	DIV	/	CVE-2025-46005

detects arb random

exploitable leakage ating systems

Problem. Existing pre-silicon microarchitectural fuzzers overfit to known vulnerabilities.

Overfitting in software for triggering leakage

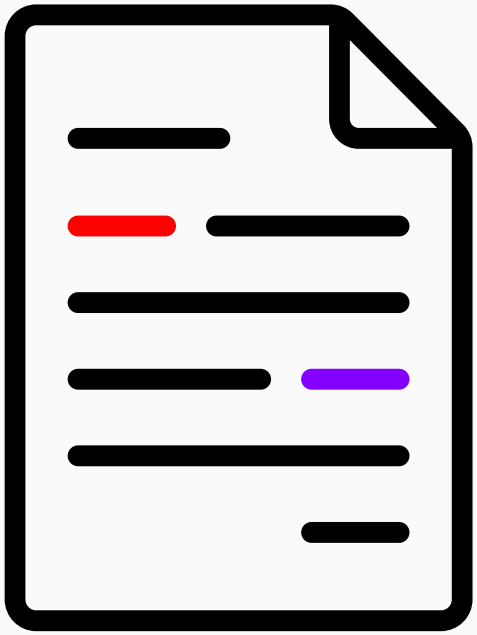
Overfitting in software for triggering leakage

Overfitting in software for triggering leakage

Overfitting in software for triggering leakage

TRAINING_GADGET

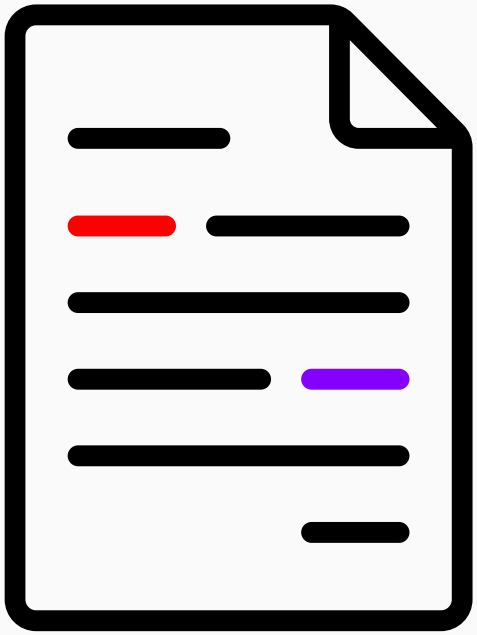
Overfitting in software for triggering leakage



TRAINING_GADGET

LEAKING_GADGET

Overfitting in software for triggering leakage



```
TRAINING_GADGET
LEAKING_GADGET
```

Overfitting in software for triggering leakage

TRAINING_GADGET

LEAKING_GADGET

Overfitting in software for triggering leakage

```
TRAINING_GADGET

LEAKING_GADGET
```

Overfitting in software for triggering leakage

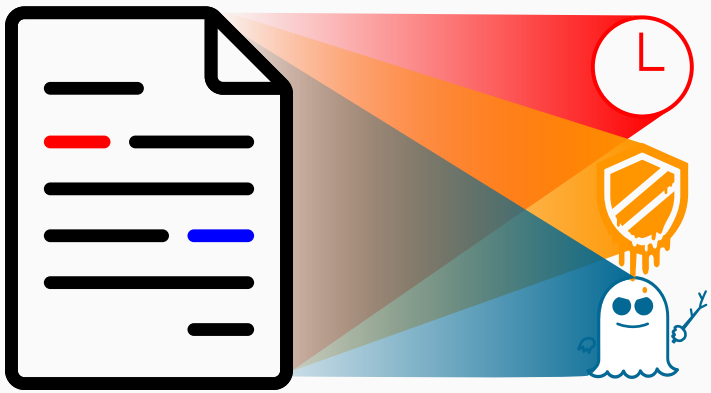
```
TRAINING_GADGET

LEAKING_GADGET
```

Overfitting in software for triggering leakage

```
TRAINING_GADGET
LEAKING_GADGET
```

Overfitting in software for triggering leakage



WhisperFuzz SEC'24

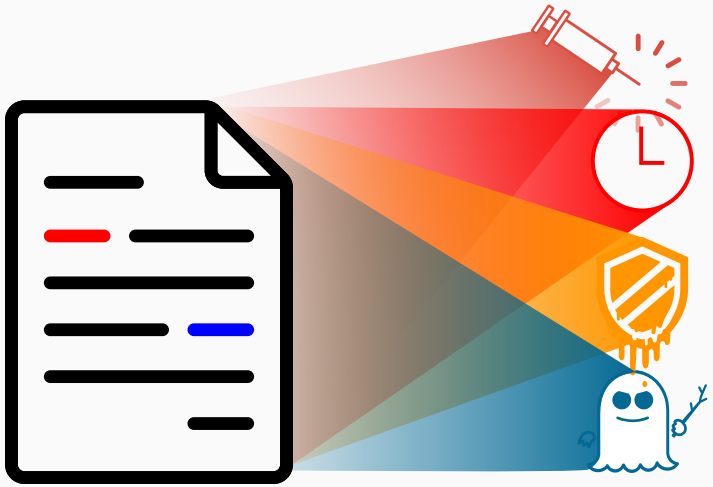
IntroSpectre ISCA'21

Phantom Trails SEC'25

SpecDoctor CCS'22

Specure DAC'24

Overfitting in software for triggering leakage



WhisperFuzz SEC'24

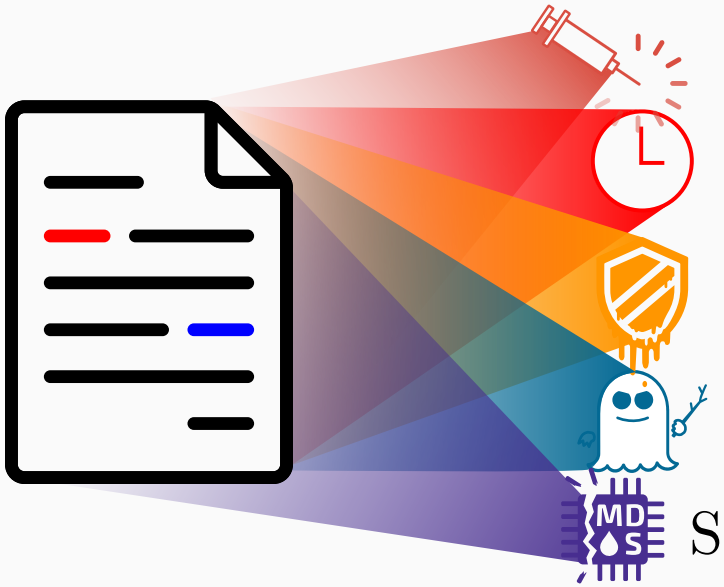
IntroSpectre ISCA'21

Phantom Trails SEC'25

SpecDoctor CCS'22

Specure DAC'24

Overfitting in software for triggering leakage



WhisperFuzz SEC'24

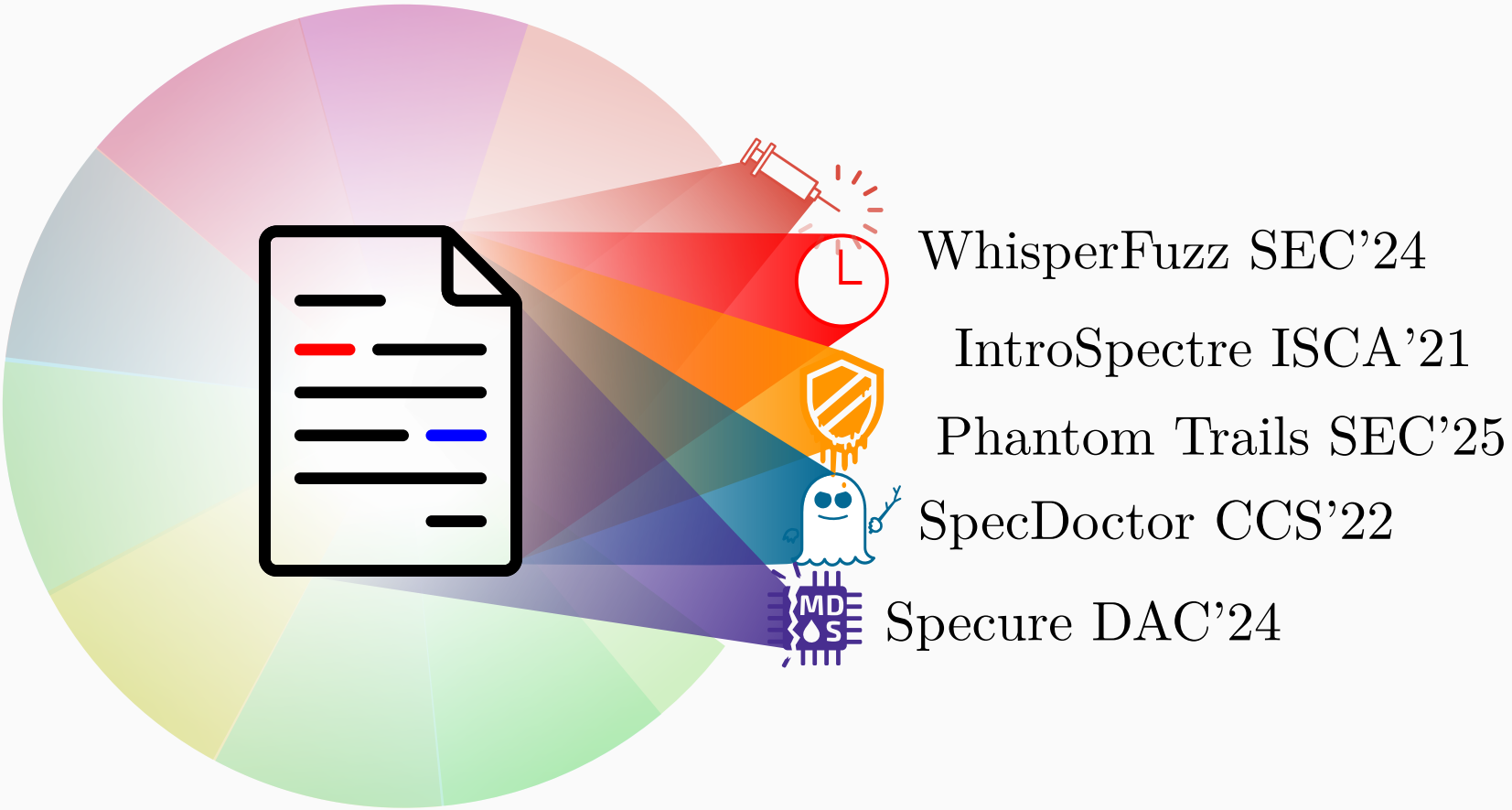
IntroSpectre ISCA'21

Phantom Trails SEC'25

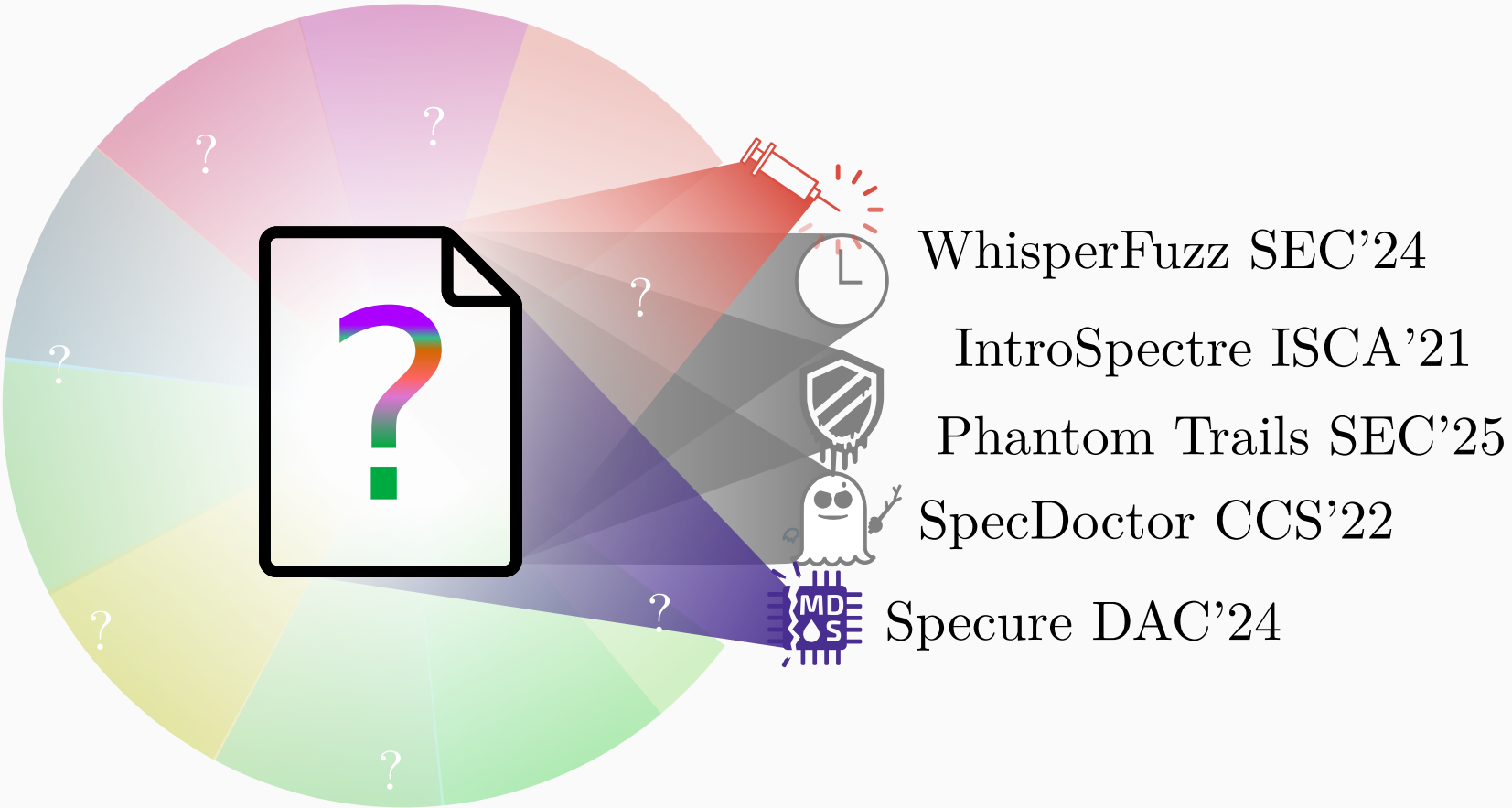
SpecDoctor CCS'22

Specure DAC'24

Overfitting in software for triggering leakage



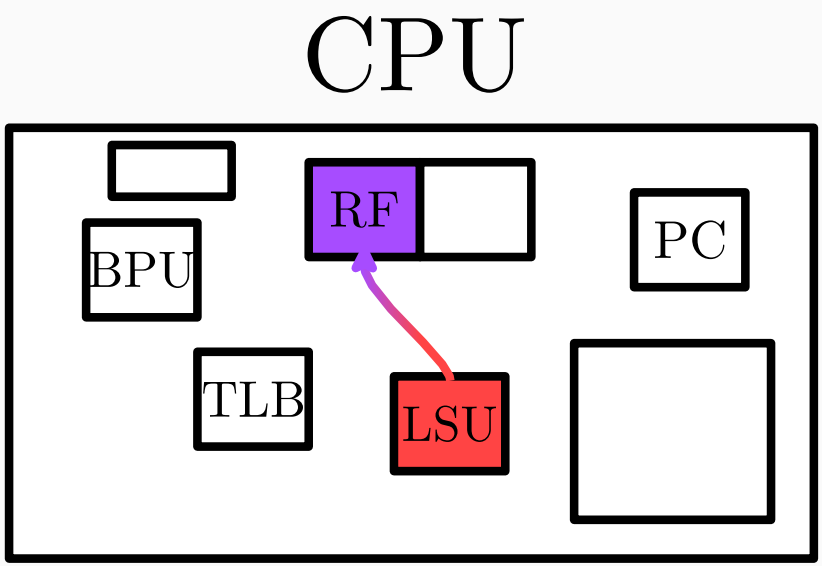
Overfitting in software for triggering leakage



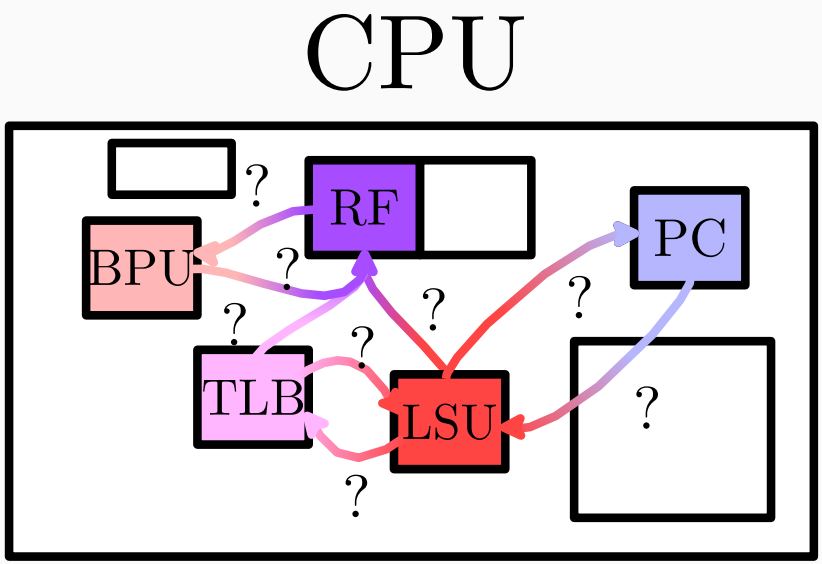
Overfitting in software for triggering leakage

Obs. 1. Test case generation overfits to known vulnerabilities.

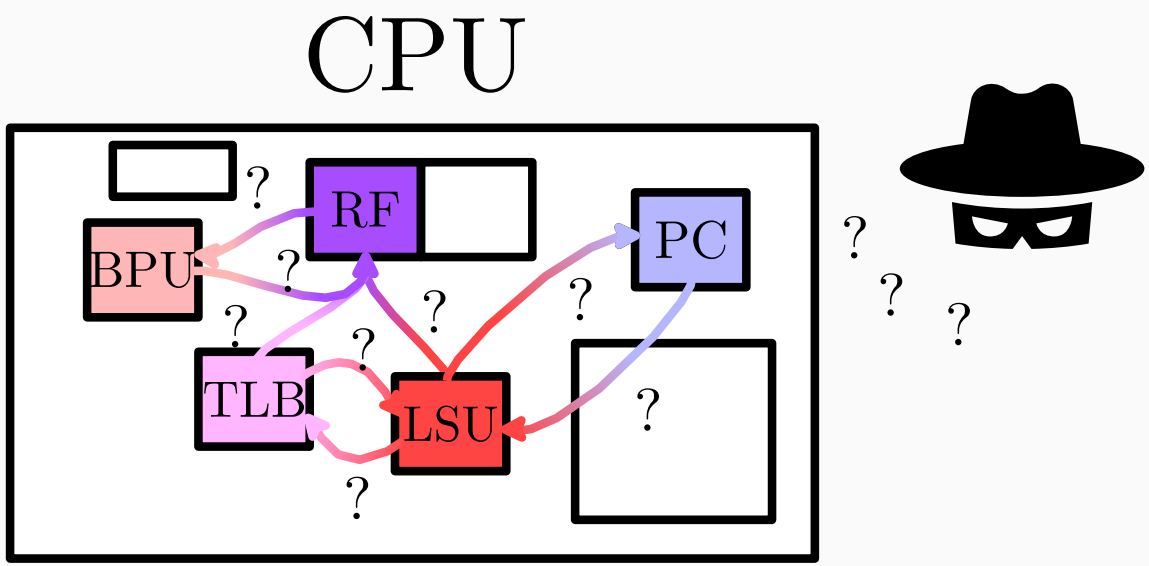
Overfitting in hardware for detecting leakage



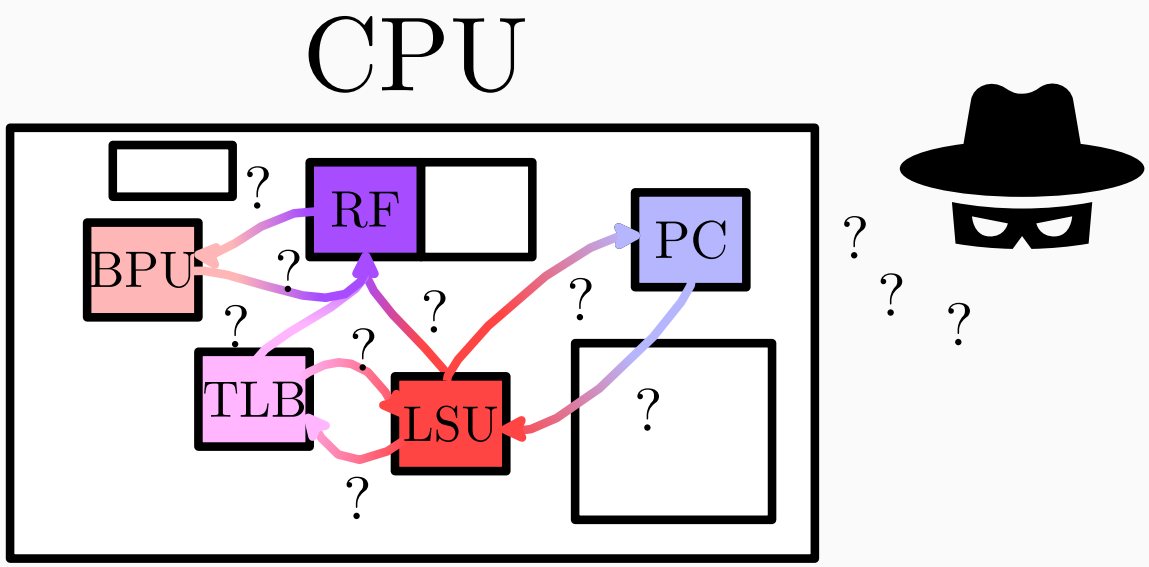
Overfitting in hardware for detecting leakage



Overfitting in hardware for detecting leakage



Overfitting in hardware for detecting leakage

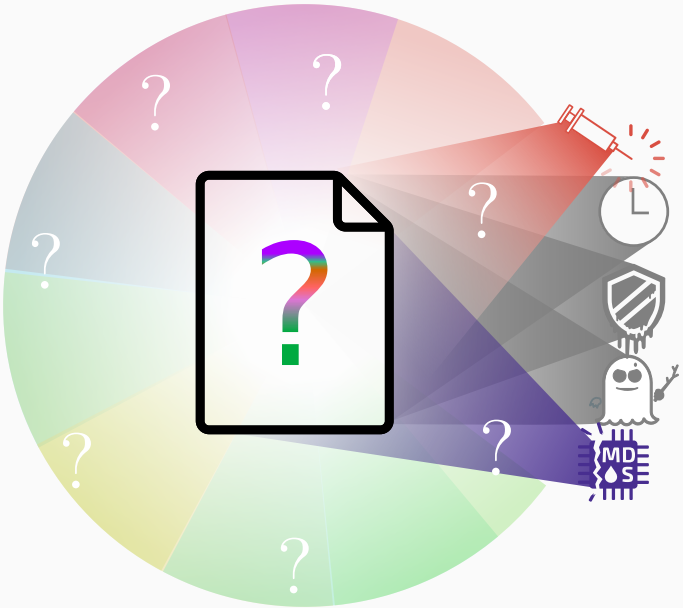


Obs. 2. Leakage detection overfits to known vulnerabilities.

We have mechanisms that **overfit** in

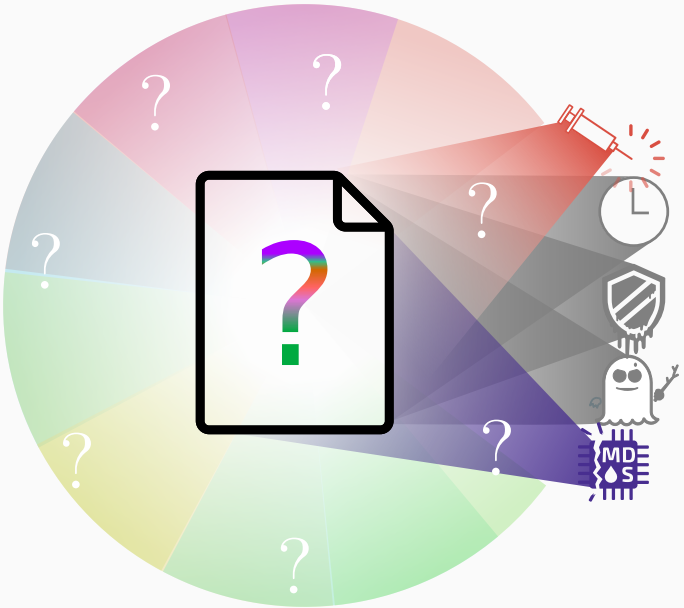
We have mechanisms that **overfit** in

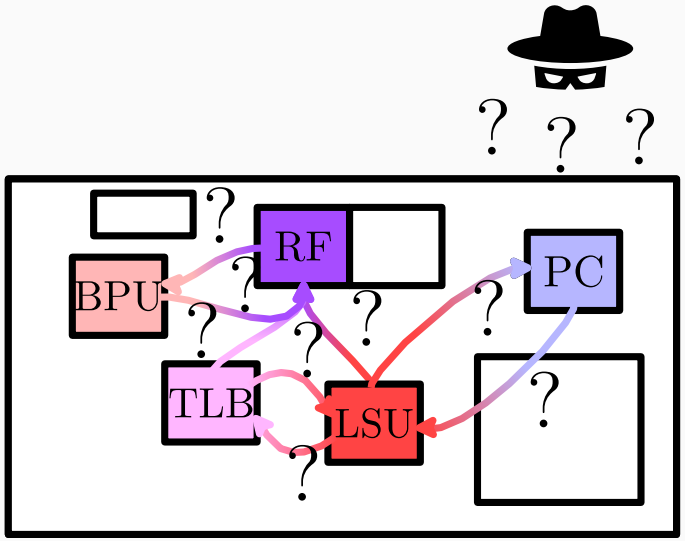
Software



We have mechanisms that **overfit** in

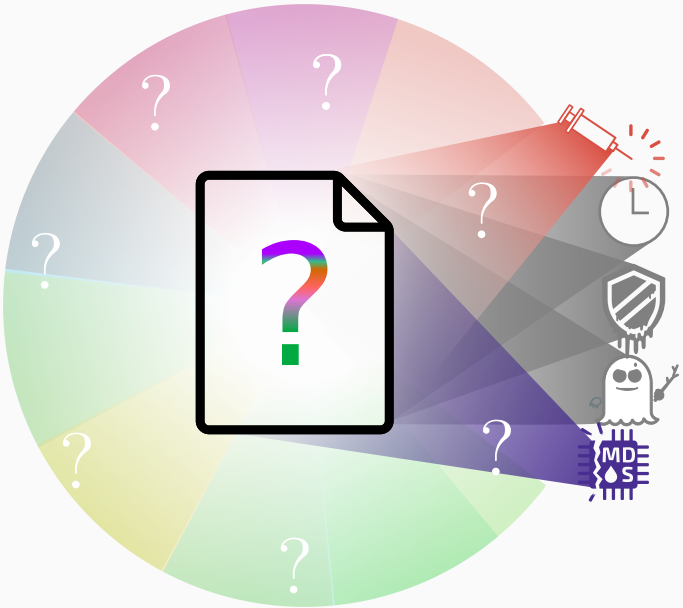
Software and **Hardware**

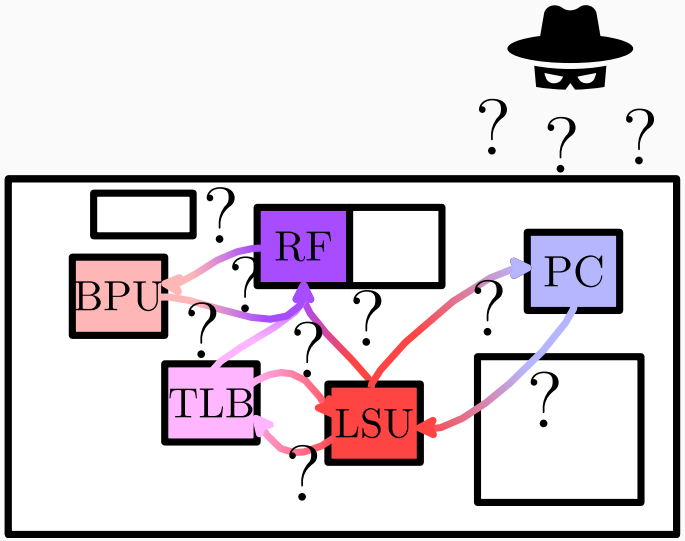




We have mechanisms that **overfit** in

Software and **Hardware**



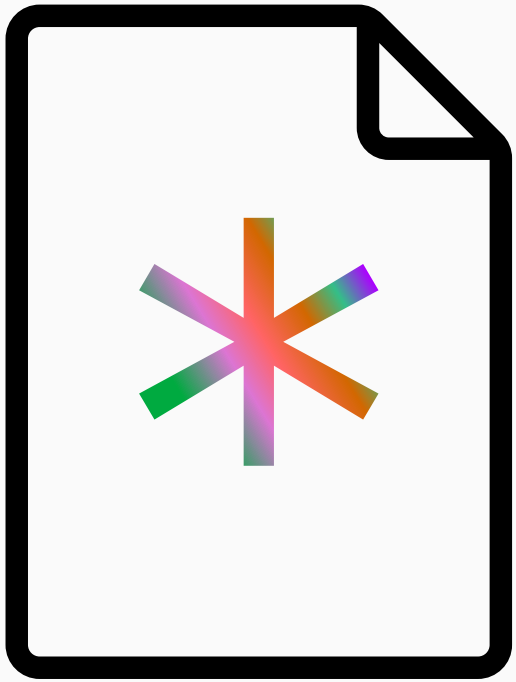


to **known vulnerabilities.**

We *need* mechanisms that **generalize** in

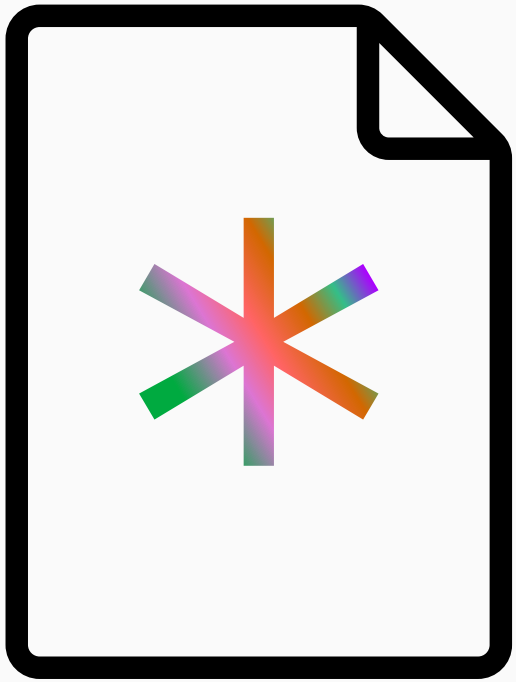
We *need* mechanisms that **generalize** in

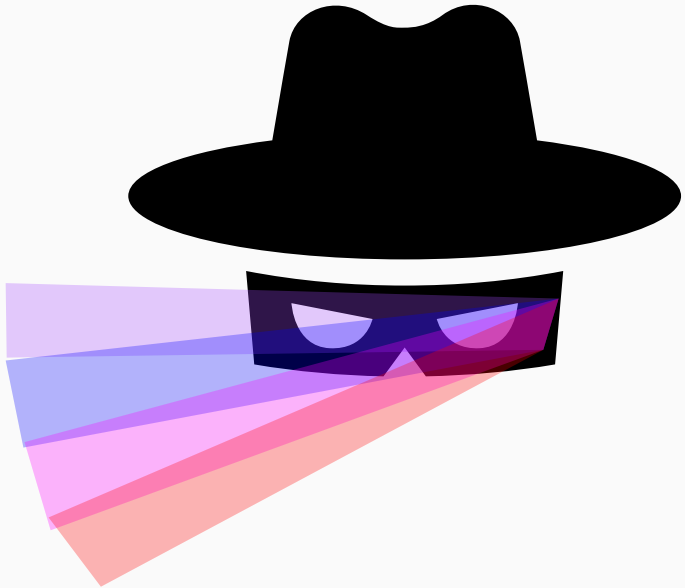
Software



We *need* mechanisms that **generalize** in

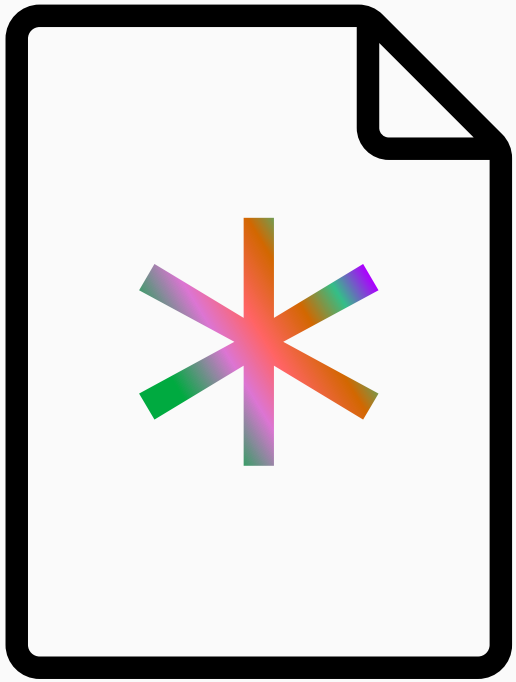
Software and **Hardware**





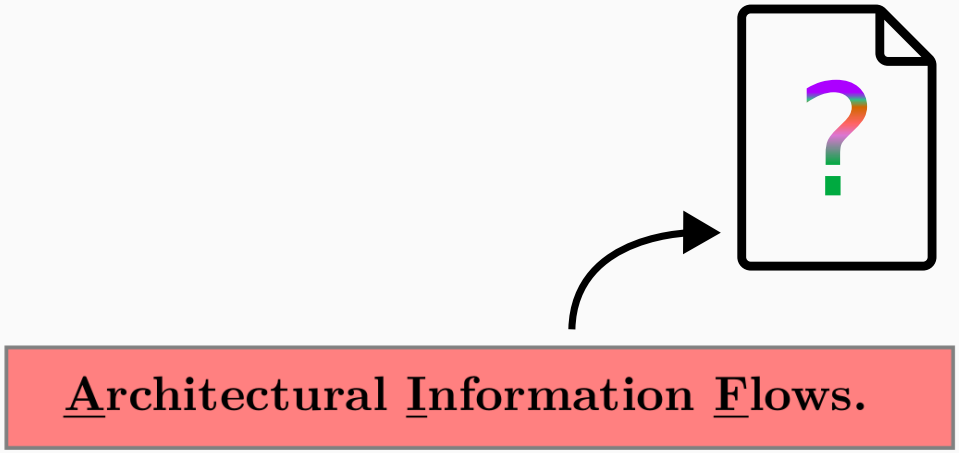
We *need* mechanisms that **generalize** in

Software and **Hardware**

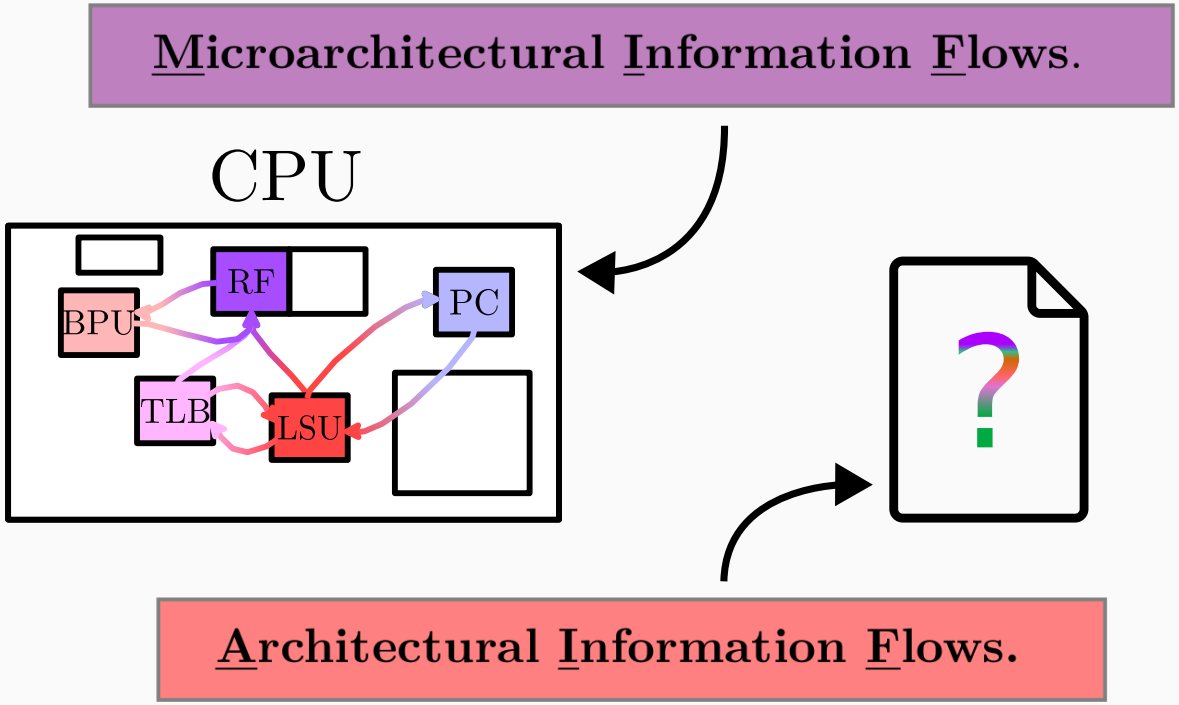


to **arbitrary vulnerabilities**.

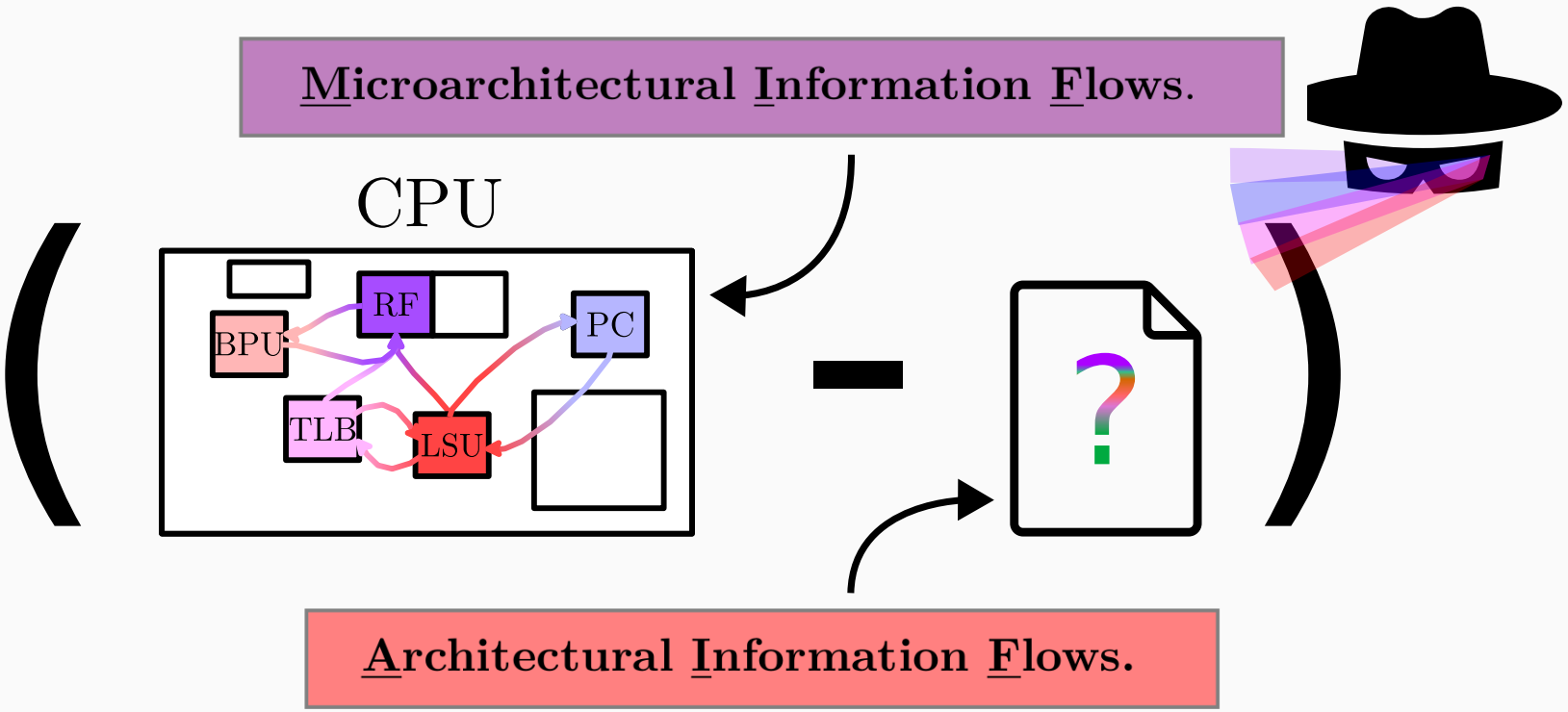
The Microarchitectural leakage Sanitizer



The Microarchitectural leakage Sanitizer



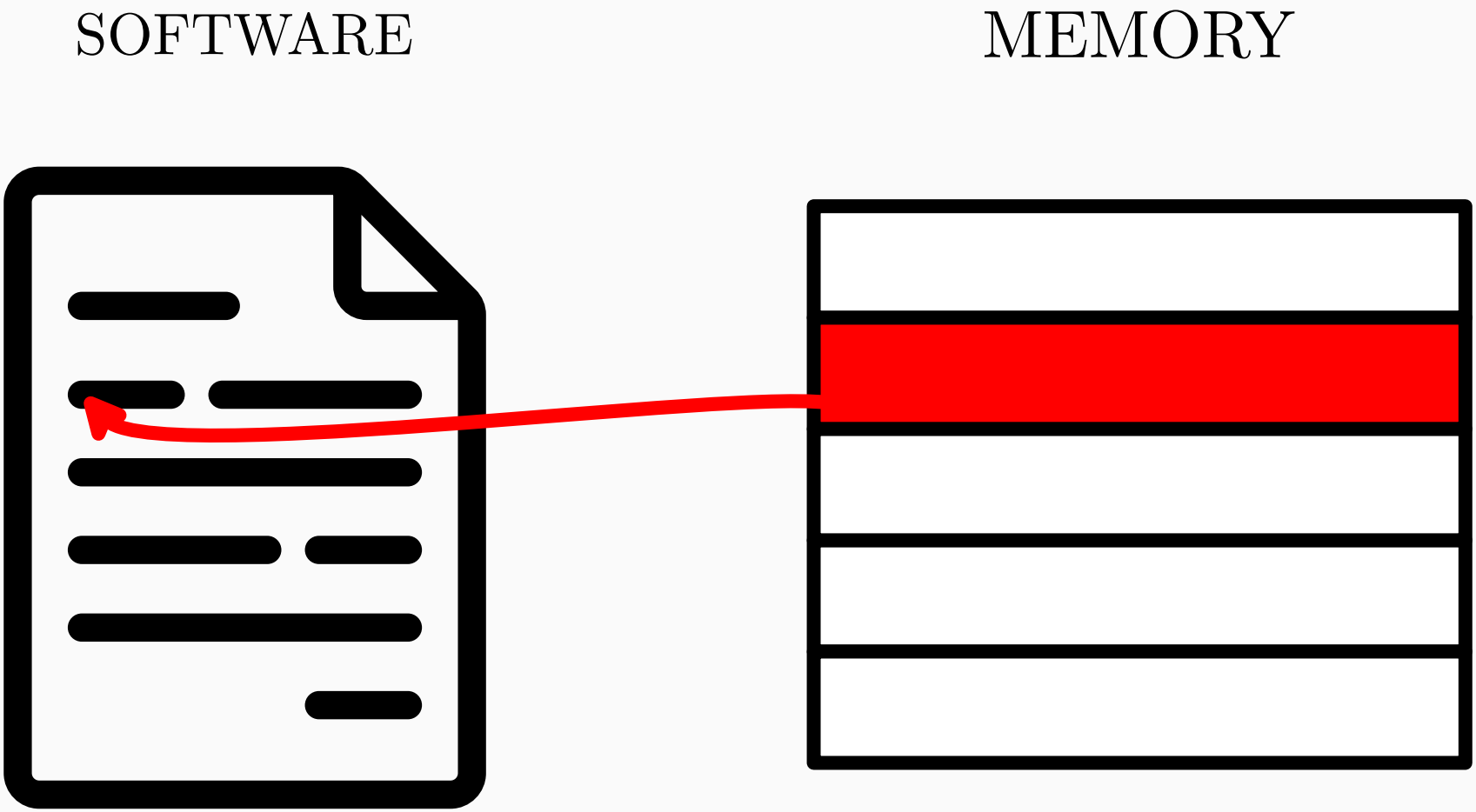
The Microarchitectural leakage Sanitizer



The Microarchitectural leakage Sanitizer

RQ. 1. How can we compute the architectural information flows in hardware?

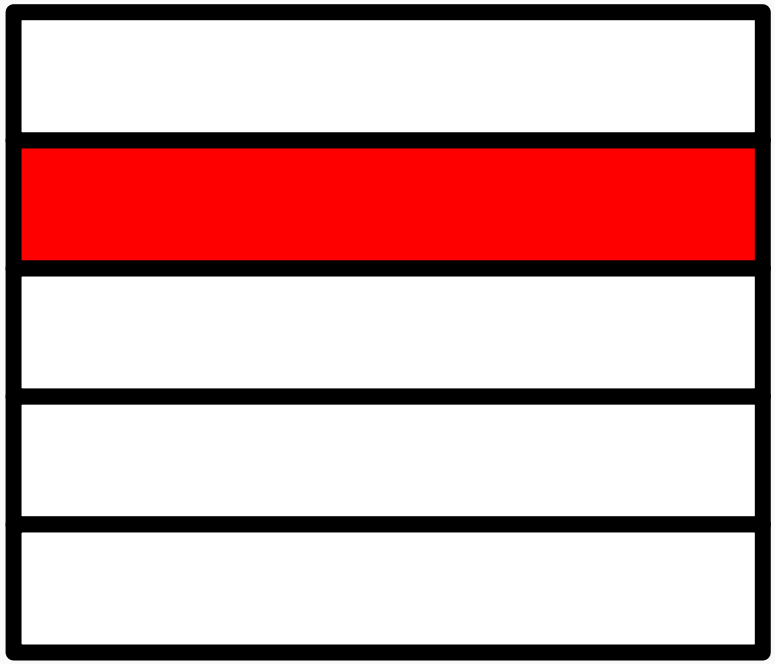
Architectural info flows in software



Architectural info flows in software

SOFTWARE

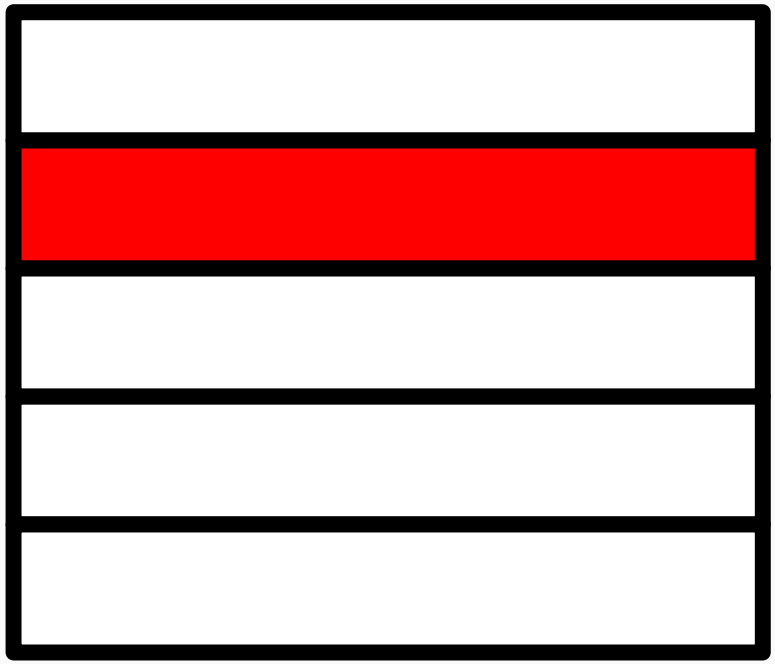
MEMORY



Architectural info flows in software

SOFTWARE

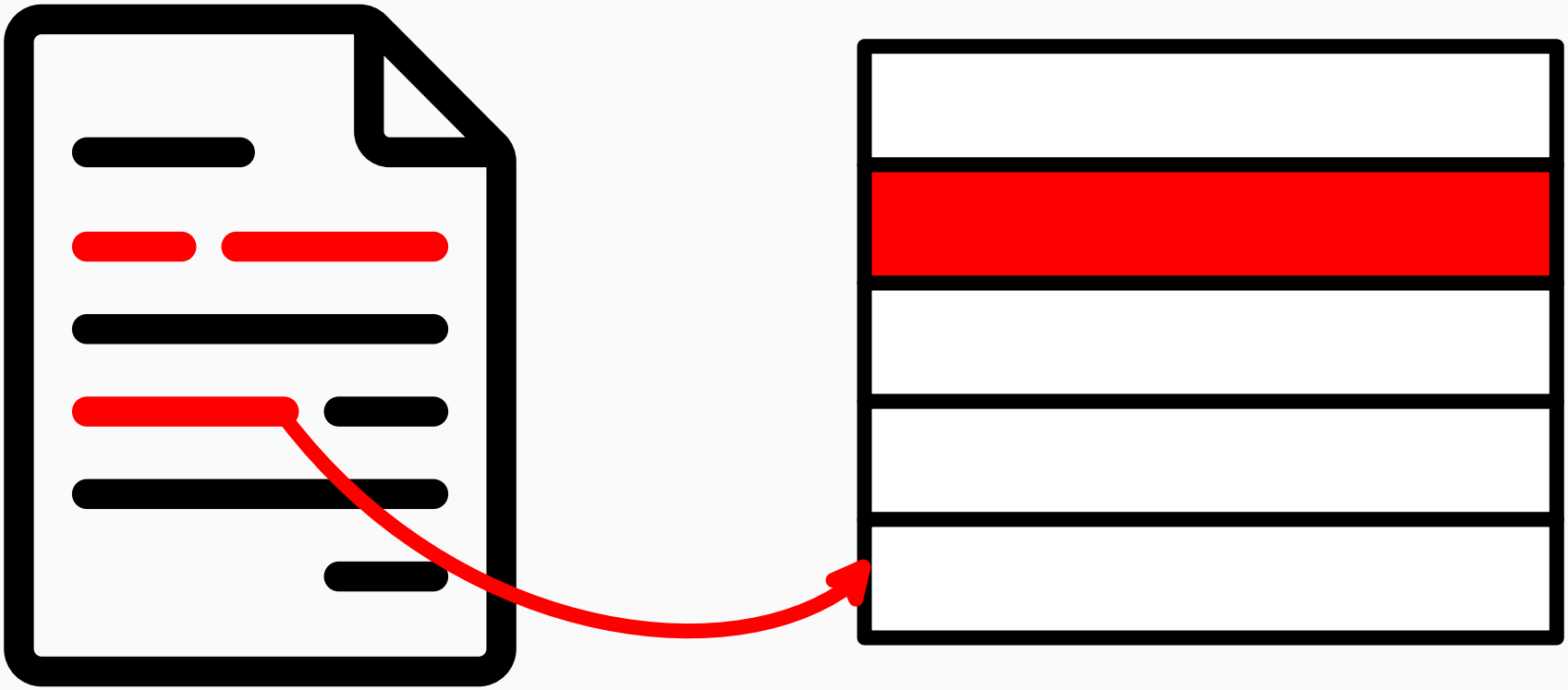
MEMORY



Architectural info flows in software

SOFTWARE

MEMORY

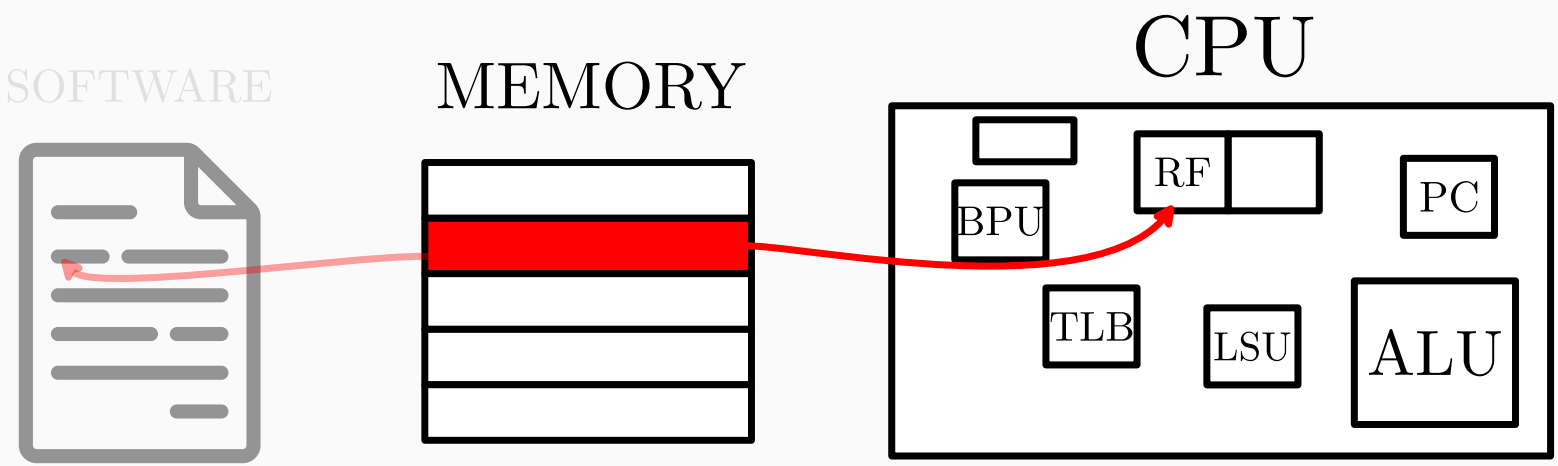


Architectural info flows in software

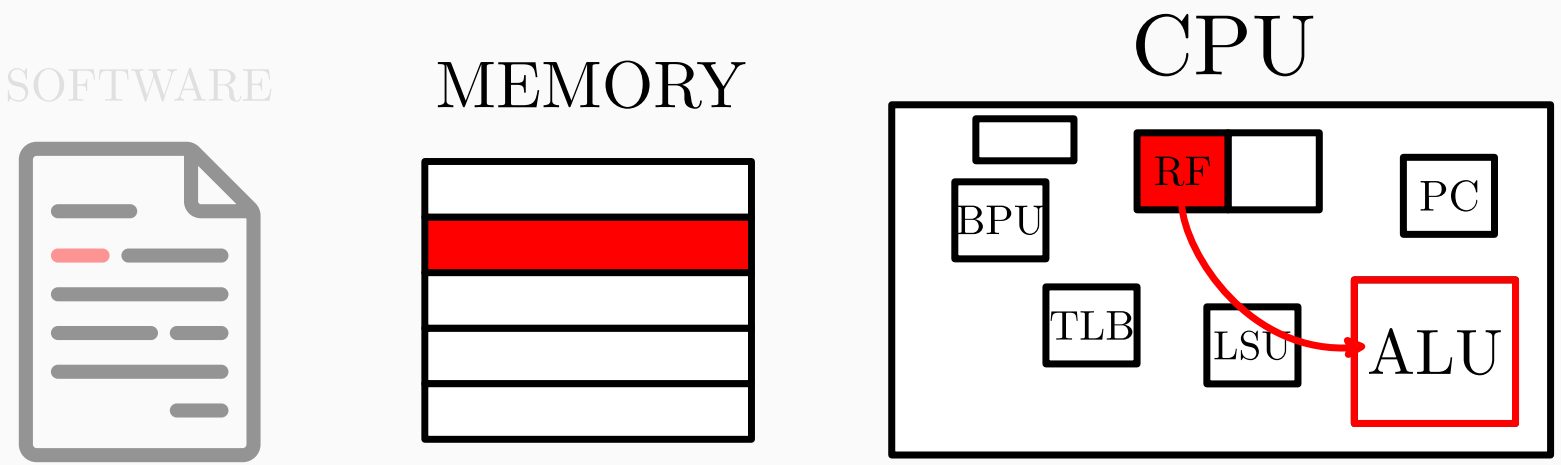
SOFTWARE

MEMORY

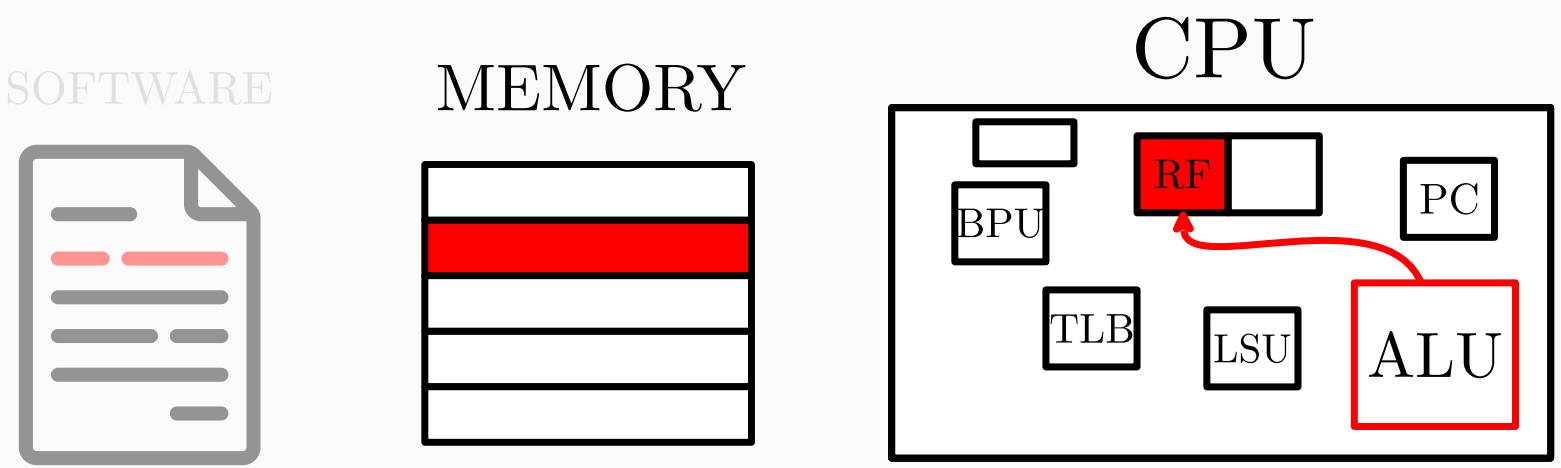
Architectural information flows in *hardware*



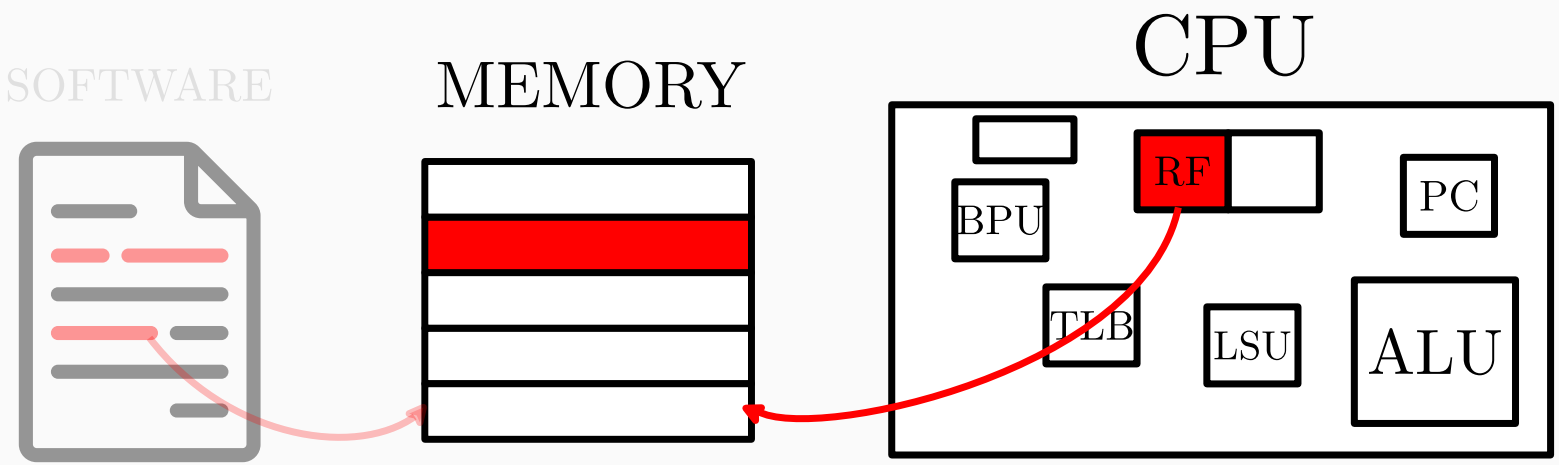
Architectural information flows in *hardware*



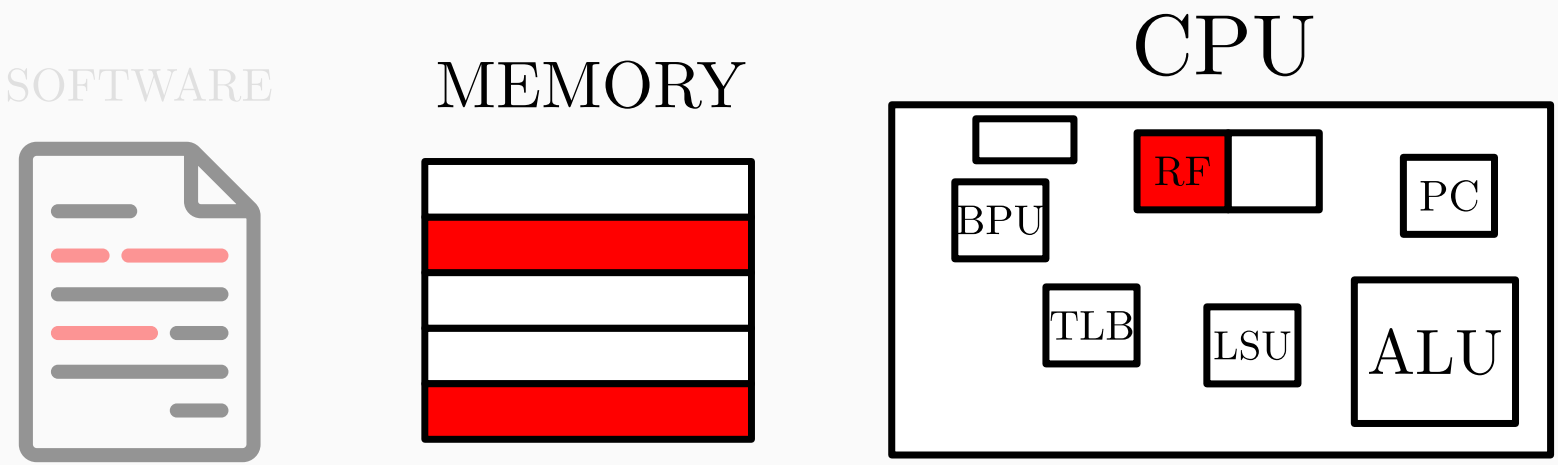
Architectural information flows in *hardware*



Architectural information flows in *hardware*



Architectural information flows in *hardware*



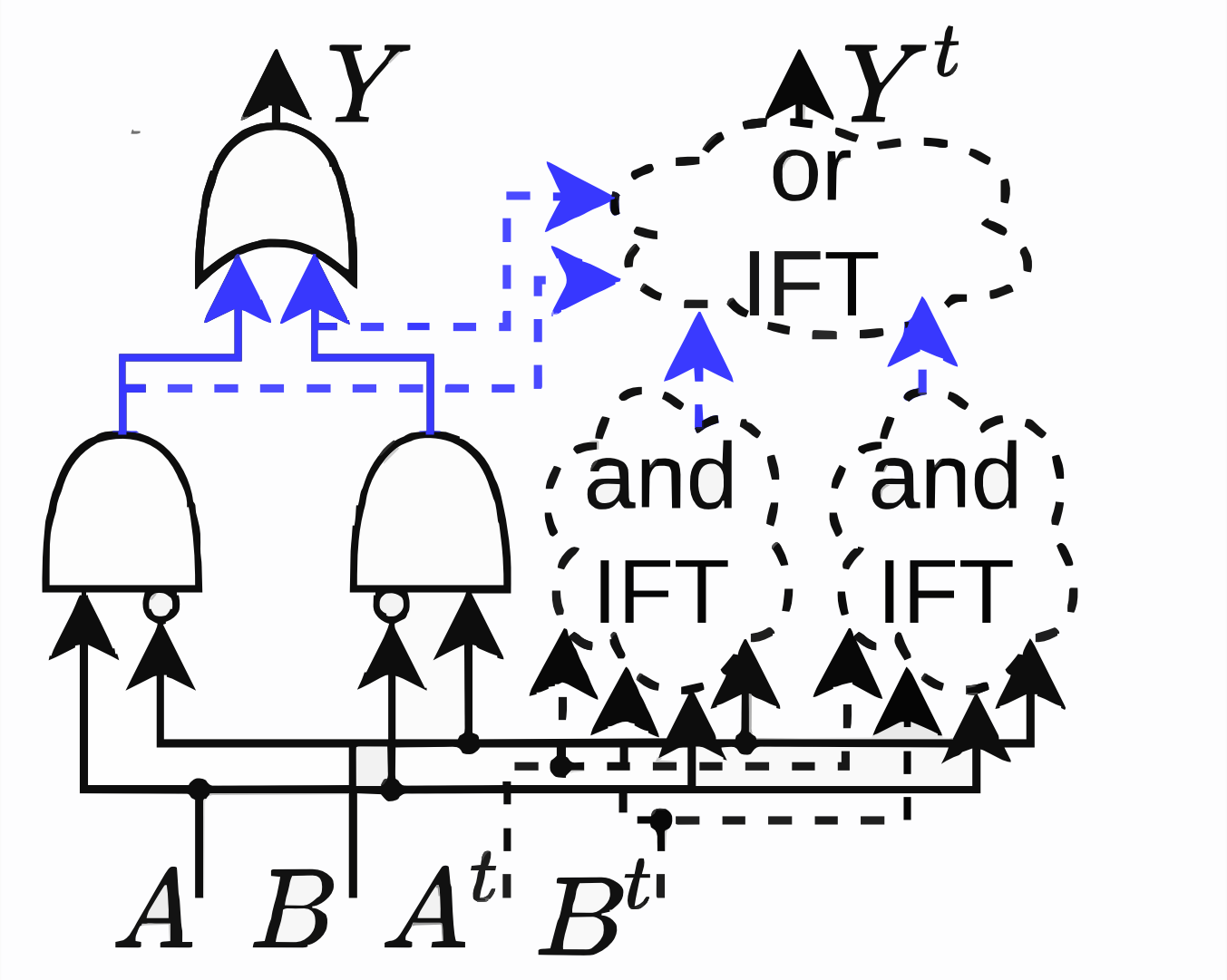
Insight 1. We can compute architectural information flows in *hardware* using *software* IFT.

The Microarchitectural leakage Sanitizer

RQ. 2. How can we compute microarchitectural information flows in hardware?

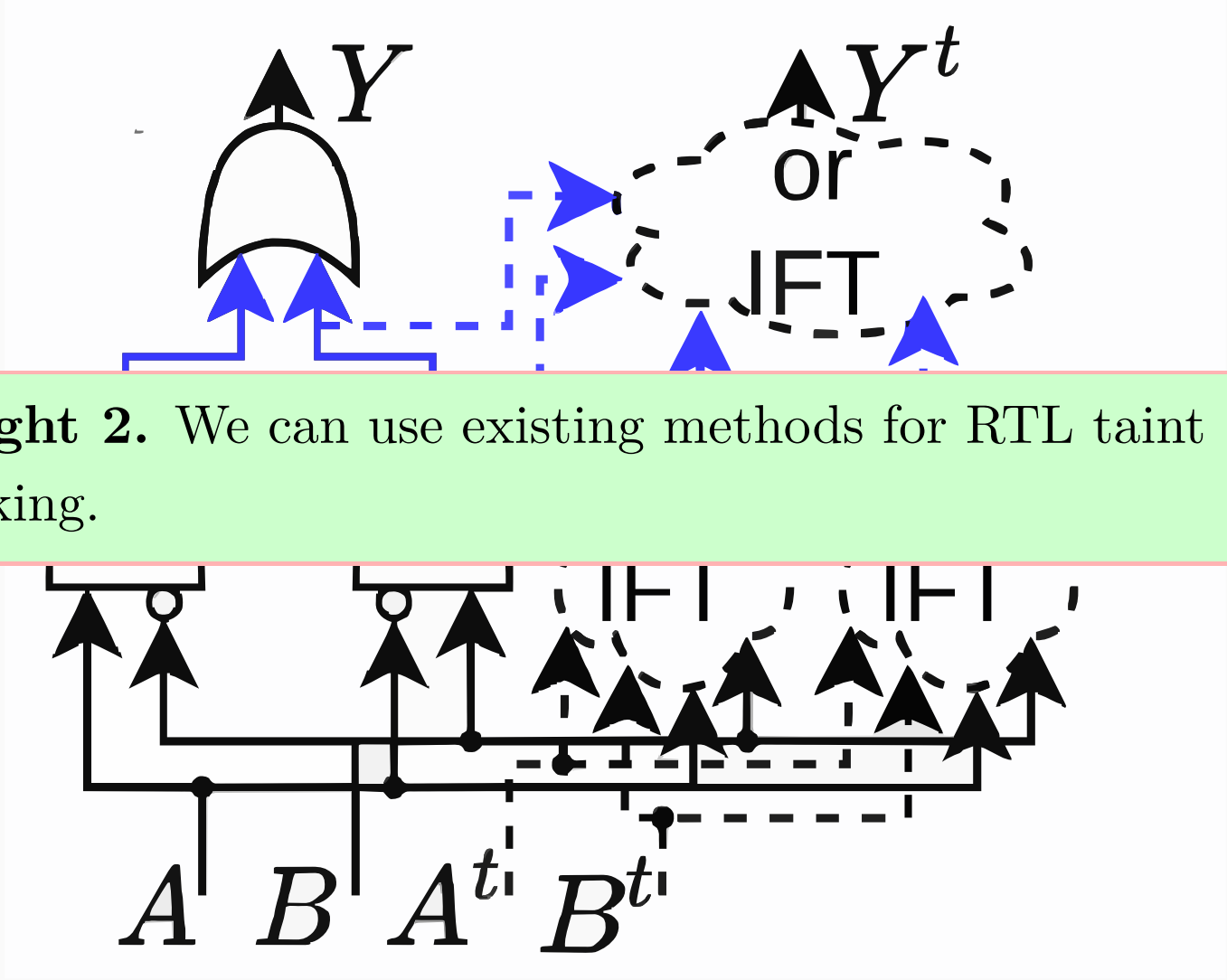
The Microarchitectural leakage Sanitizer

CellIFT SEC'22



The Microarchitectural leakage Sanitizer

CellIFT SEC'22

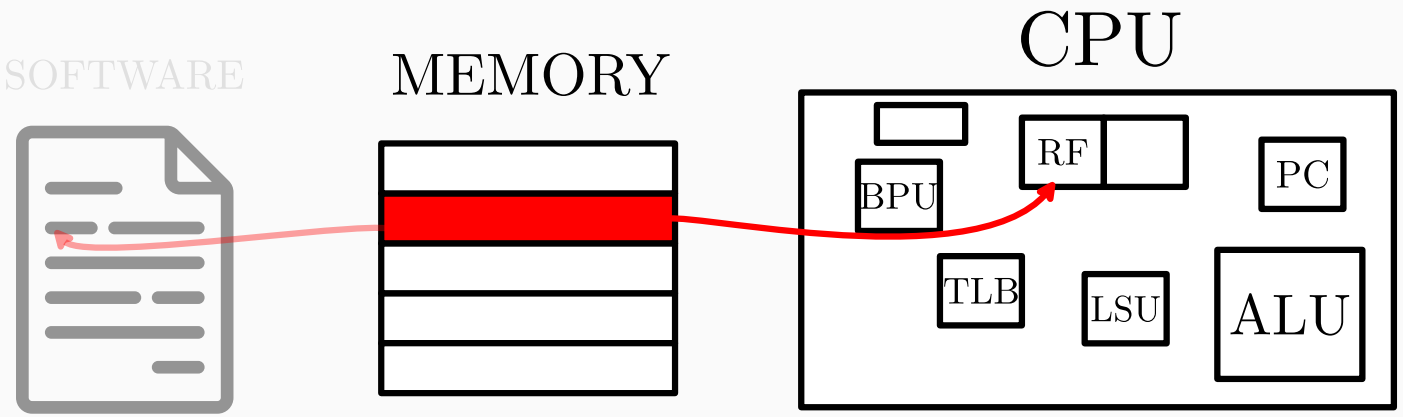


Insight 2. We can use existing methods for RTL taint tracking.

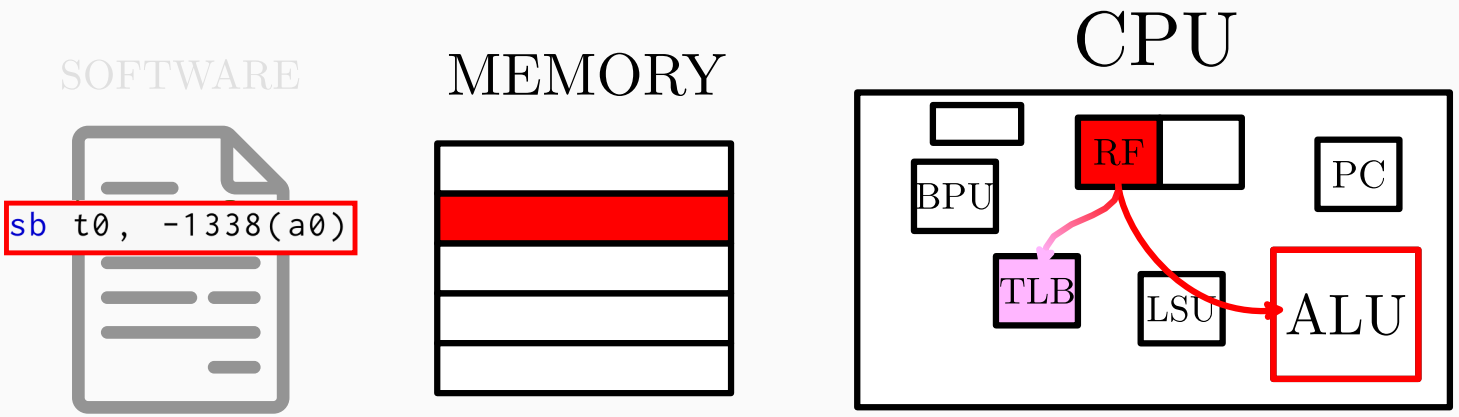
The Microarchitectural leakage Sanitizer

RQ. 3. How can we detect leakage?

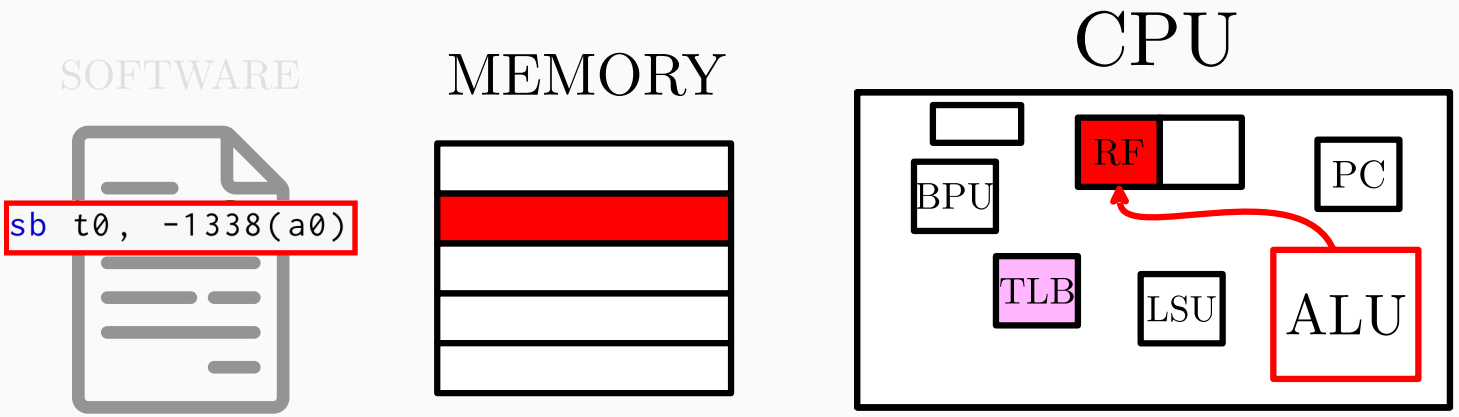
Arch. and Microarch. Info Flows in hardware



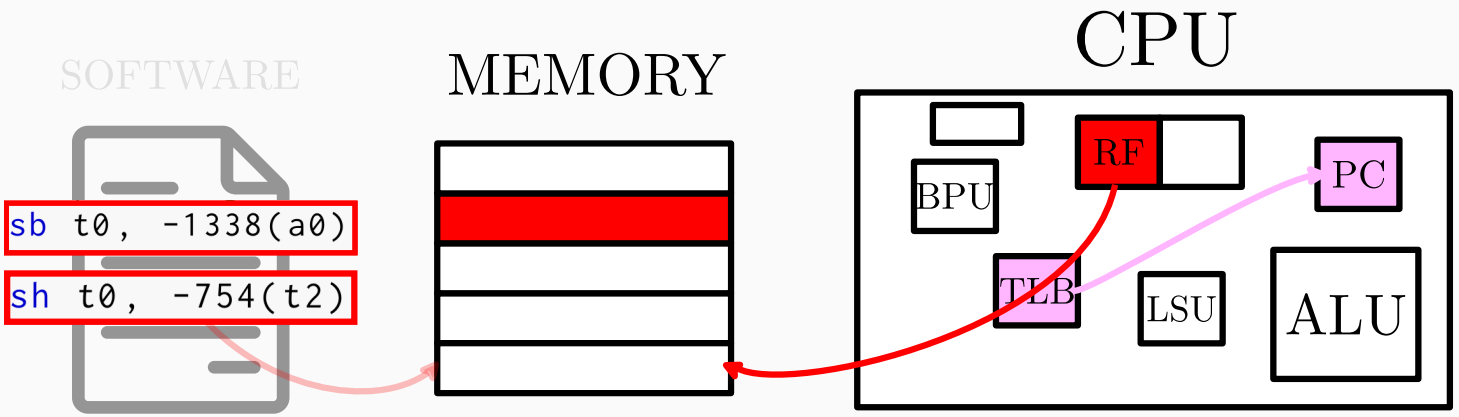
Arch. and Microarch. Info Flows in hardware



Arch. and Microarch. Info Flows in hardware



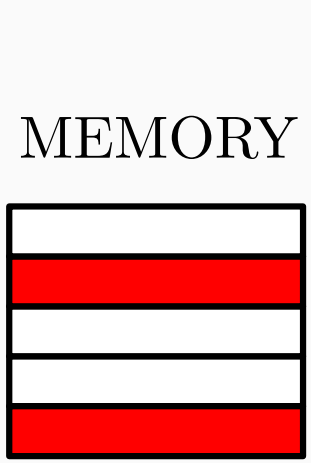
Arch. and Microarch. Info Flows in hardware

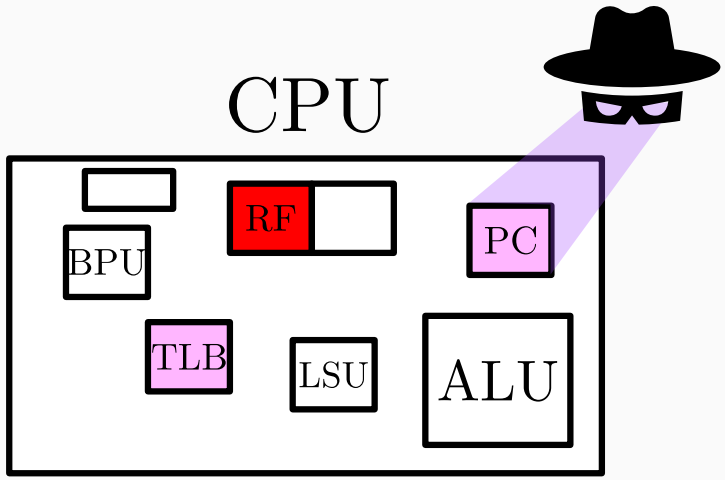


Arch. and Microarch. Info Flows in hardware

CVE-2025-29340

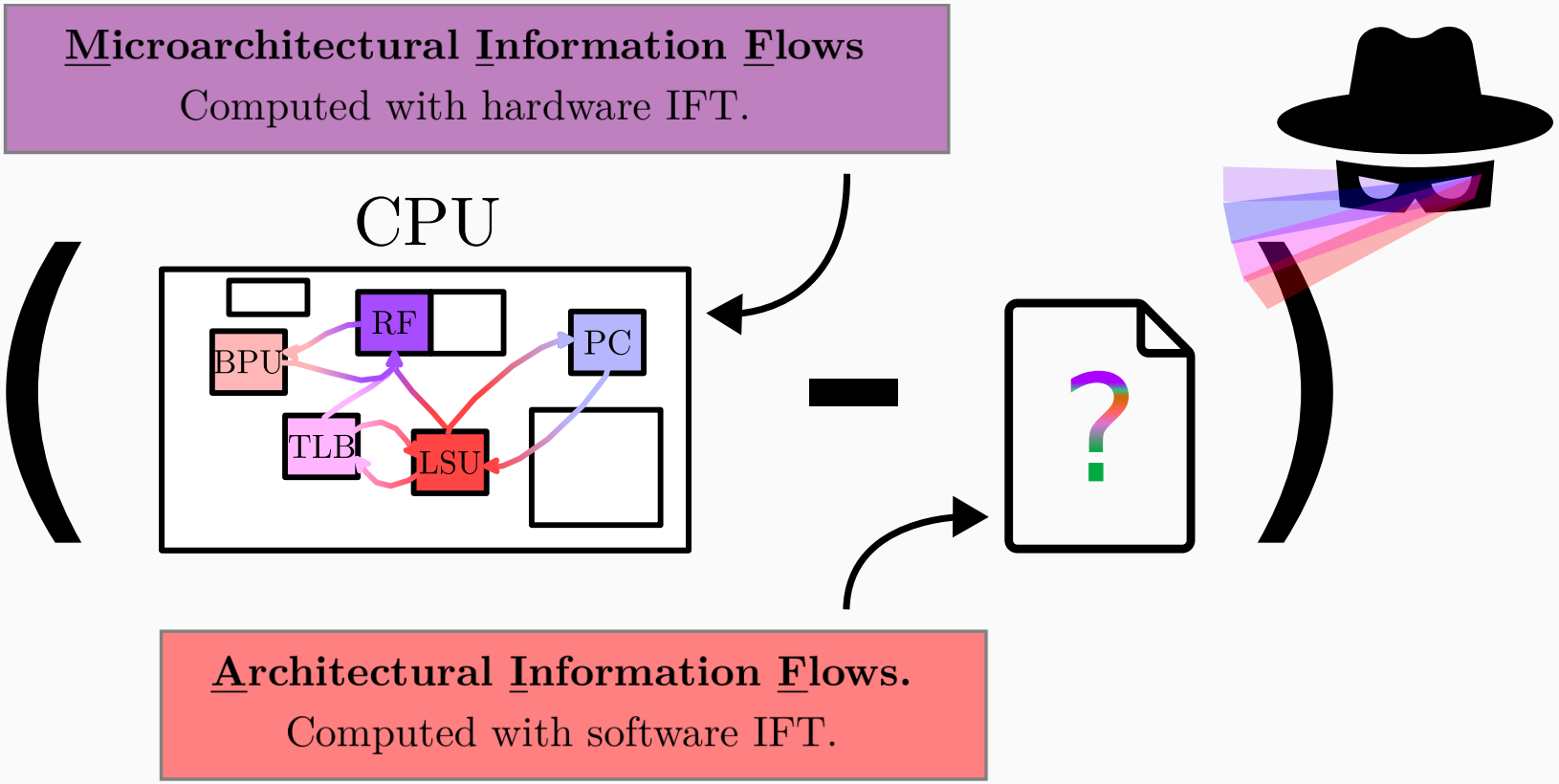
```
# Load some privileged data from S-mode.
0xa244c: lw s2, 456(ra)
...
# SPP=1, so we remain in S-mode.
0xa2dd8: sret
...
# The store executes transiently
# and leaks privileged data to the TLB.
0xa2ddc: sb t0, -1338(a0)
...
# SPP=0, so we go to U-mode.
0xd0f50: sret
...
# A store now leaks from the TLB.
0x67b50: sh t0, -754(t2)
```



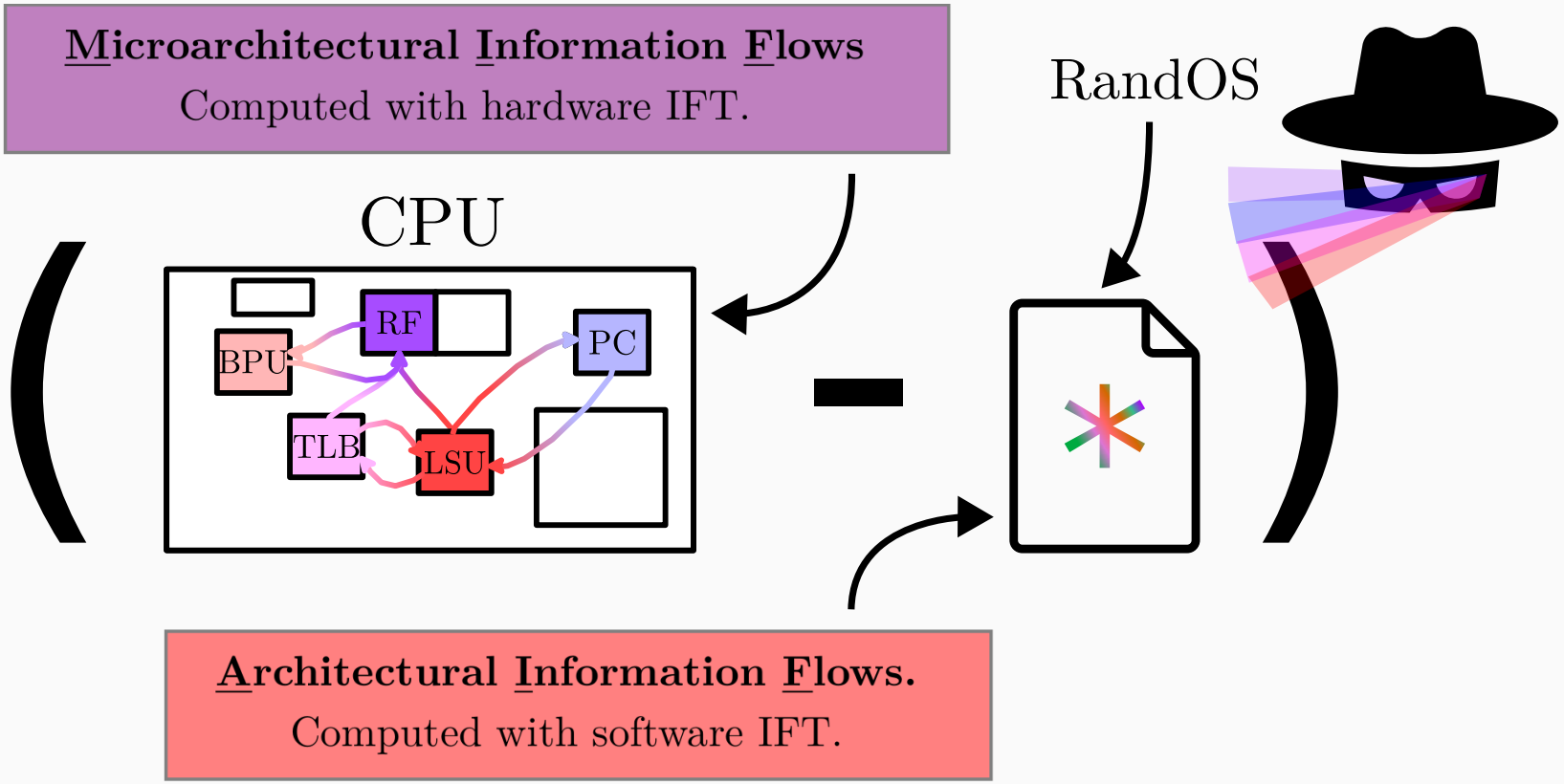


Insight 3. Exploitable leakage is a microarchitectural information flow to the PC!

MileSan and RandOS



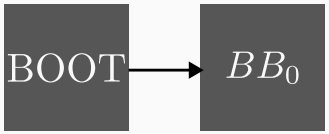
MileSan and RandOS



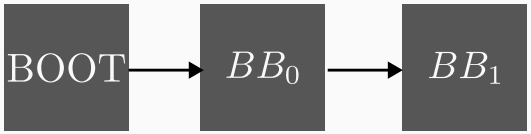
RQ. 4. How should test cases behave?

RandOS: Generating programs

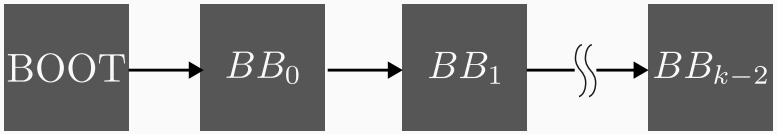
RandOS: Generating programs



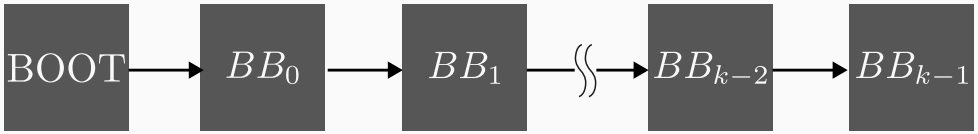
RandOS: Generating programs



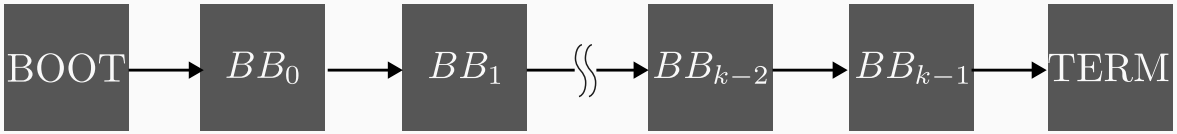
RandOS: Generating programs



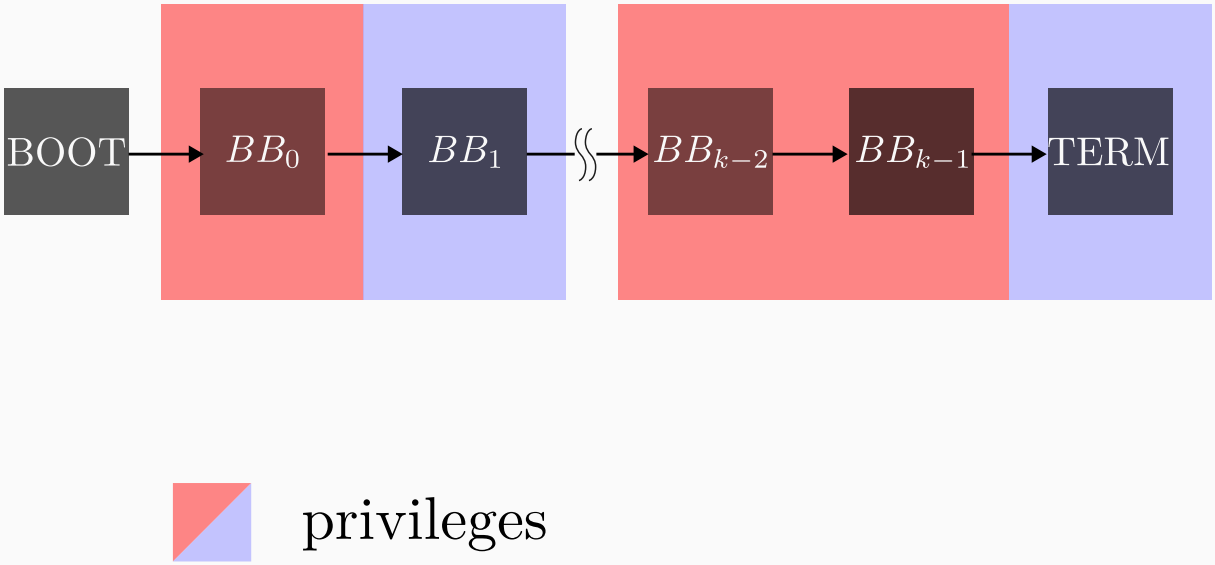
RandOS: Generating programs



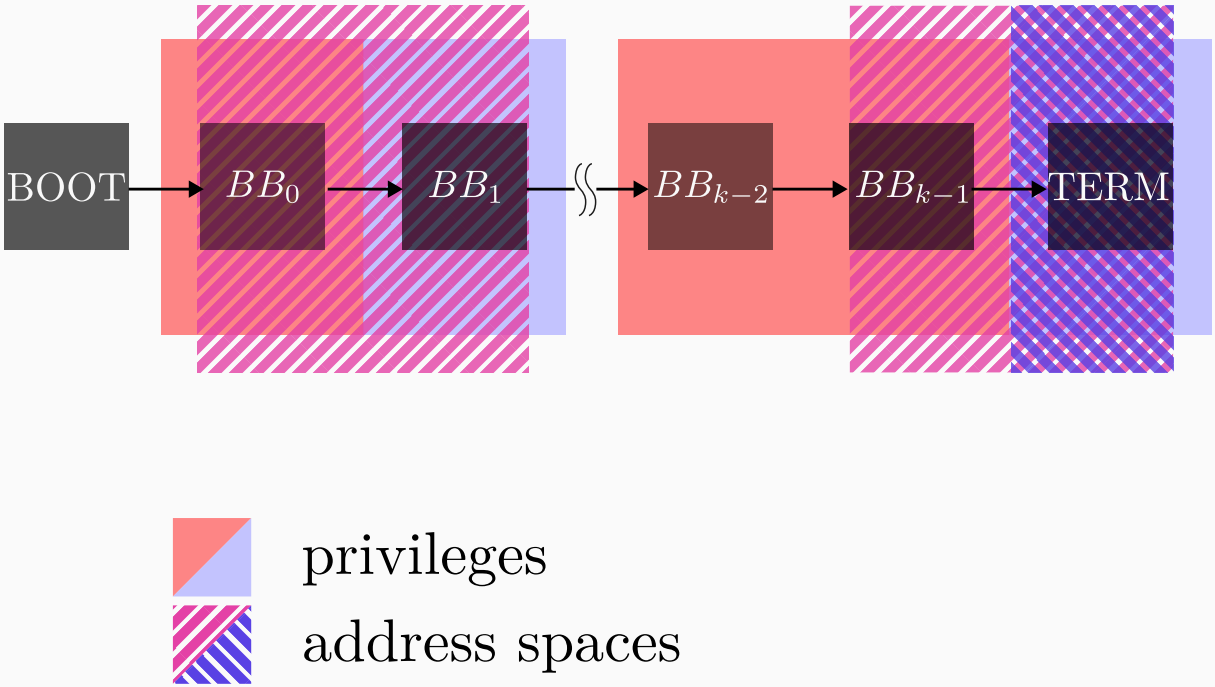
RandOS: Generating programs



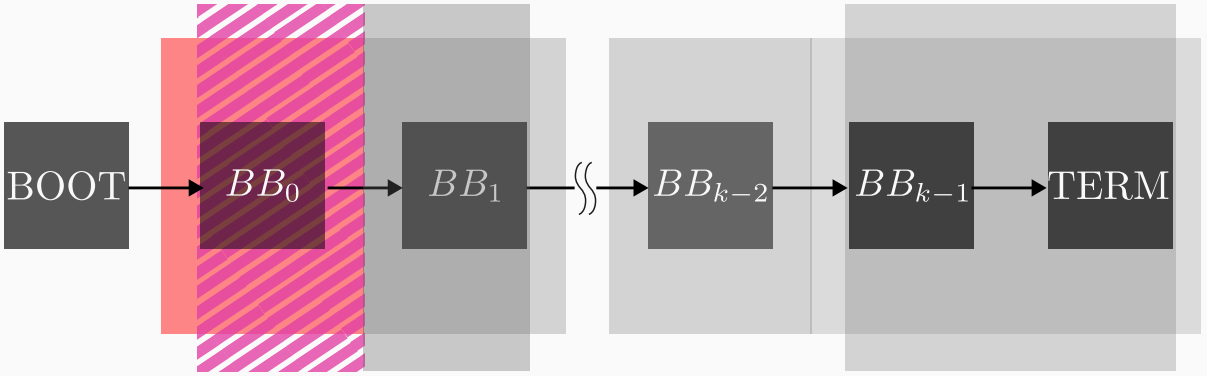
RandOS: Generating programs

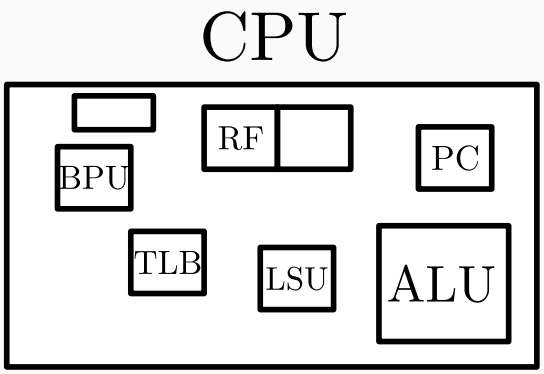


RandOS: Generating programs

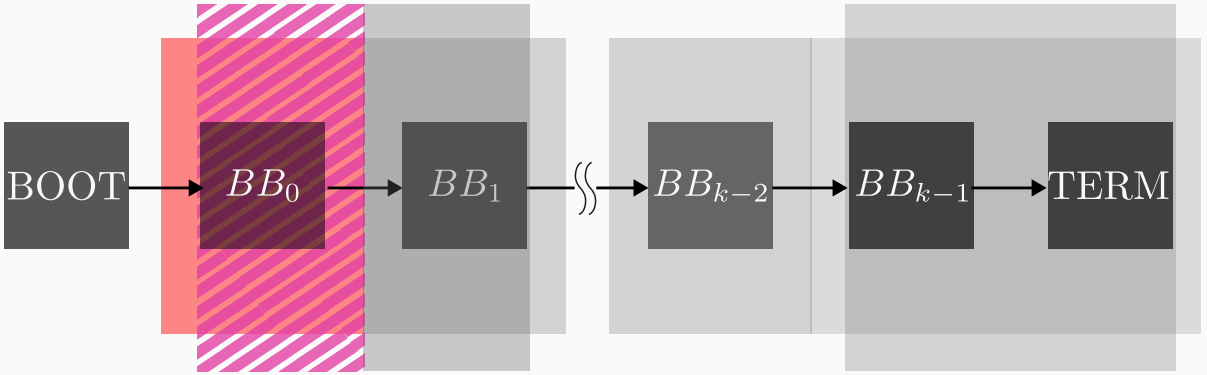


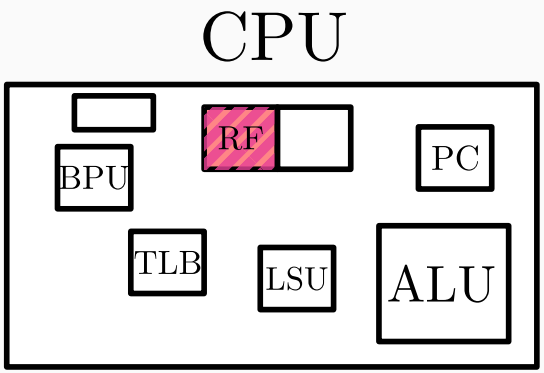
RandOS: Traversing isolation boundaries



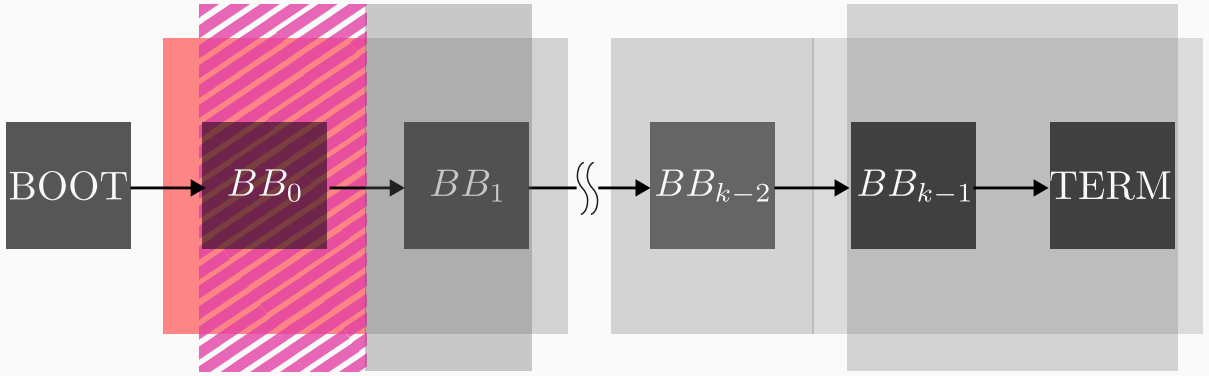


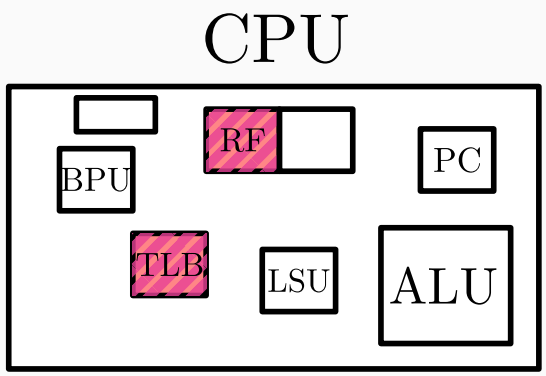
RandOS: Traversing isolation boundaries



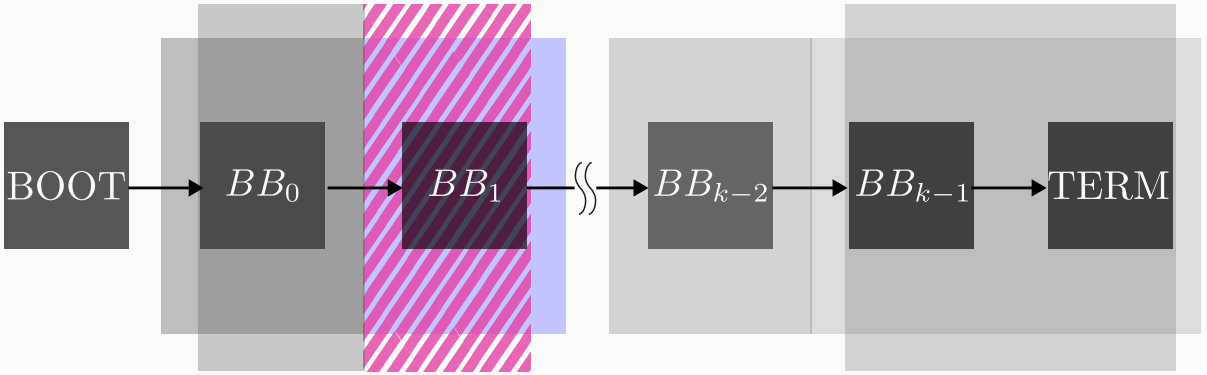


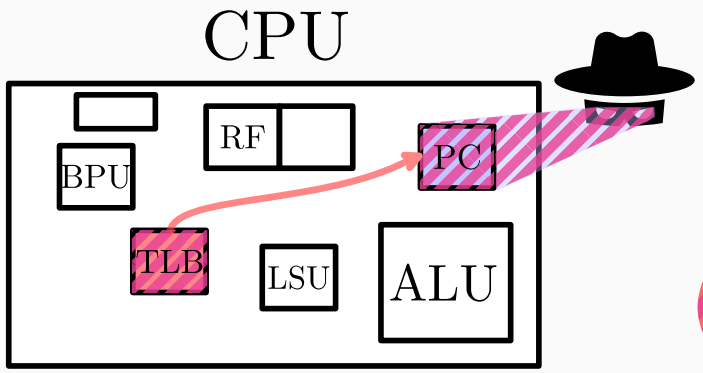
RandOS: Traversing isolation boundaries





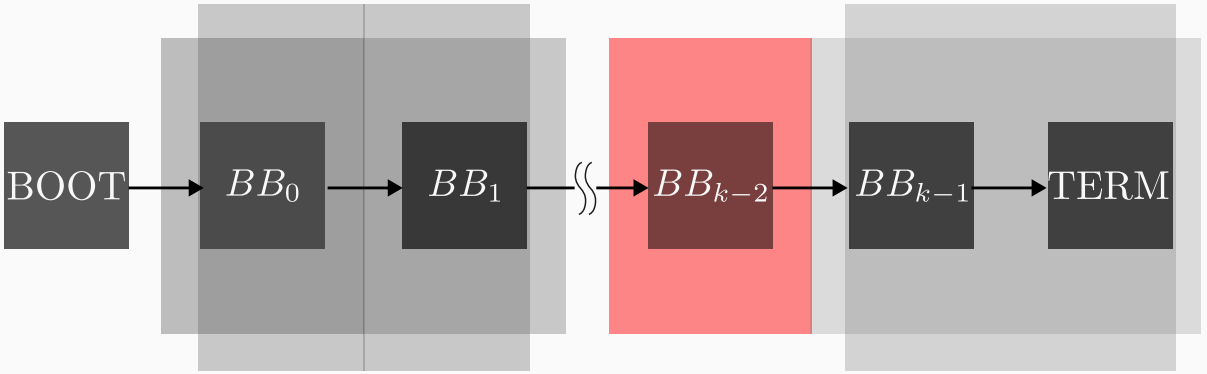
RandOS: Traversing isolation boundaries

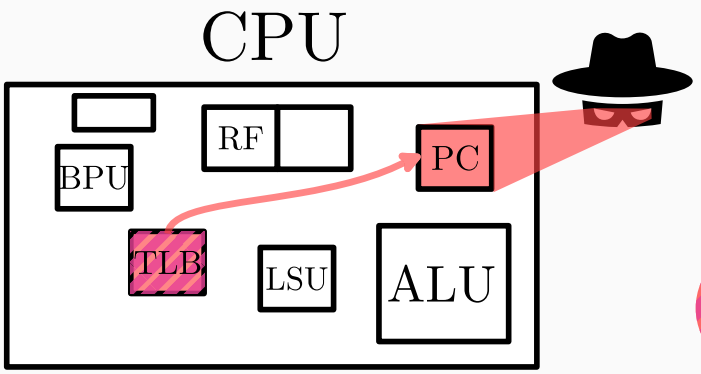




1 cross-privilege

RandOS: Traversing isolation boundaries

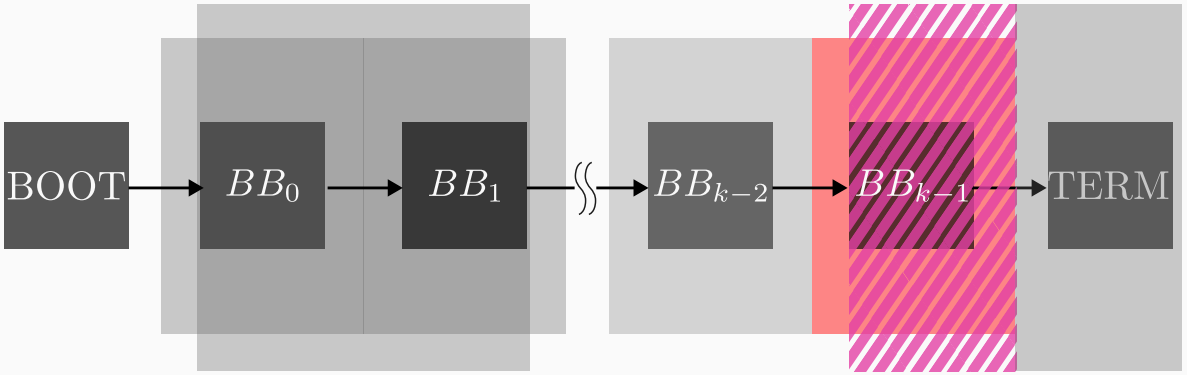


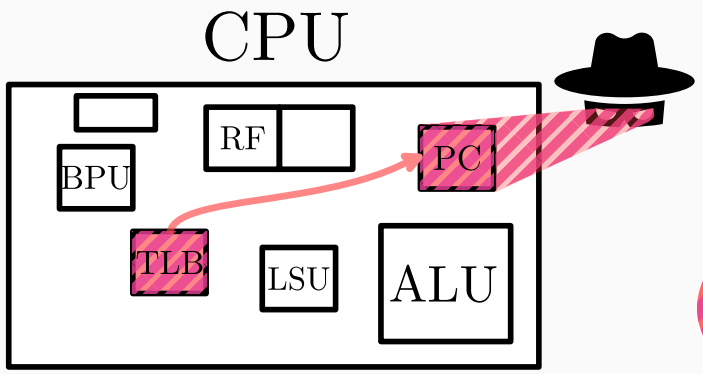


1 cross-privilege

2 cross-addr. space

RandOS: Traversing isolation boundaries



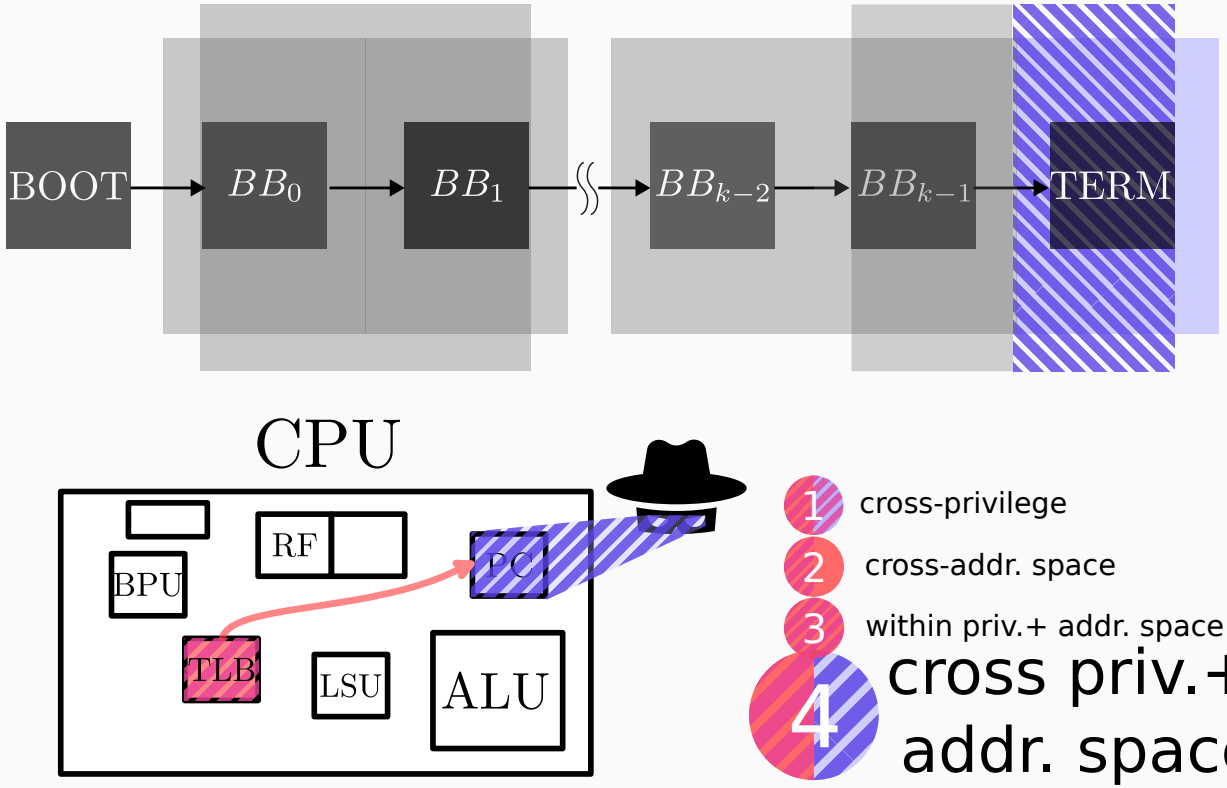


1 cross-privilege

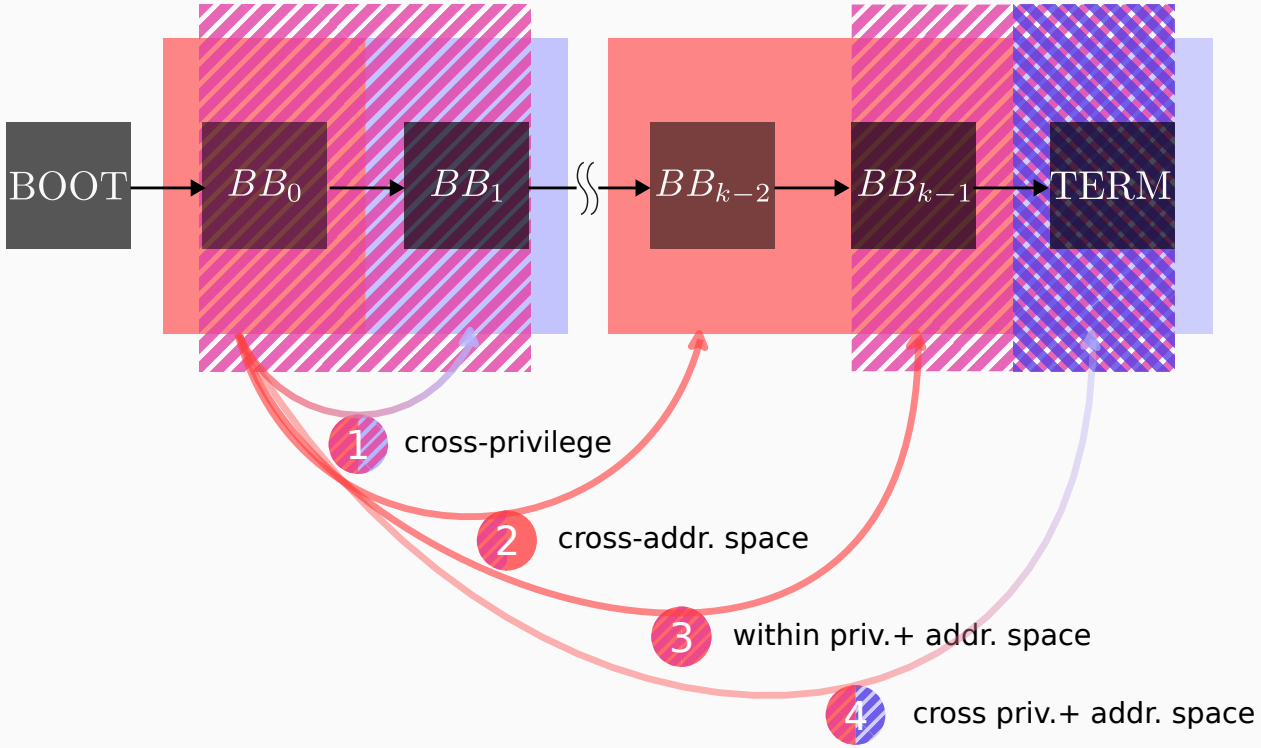
2 cross-addr. space

3 within priv.+ addr. space

RandOS: Traversing isolation boundaries

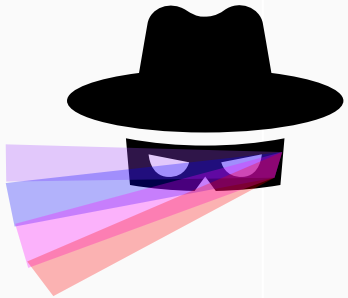


RandOS: Traversing isolation boundaries



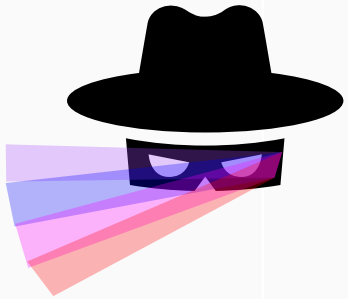
Implementation

MileSan. Hardware IFT: Using CellIFT and HybridIFT. Software IFT: Approx. 15k LoC Python.

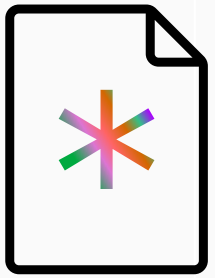


Implementation

MileSan. Hardware IFT: Using CellIFT and HybridIFT. Software IFT: Approx. 15k LoC Python.

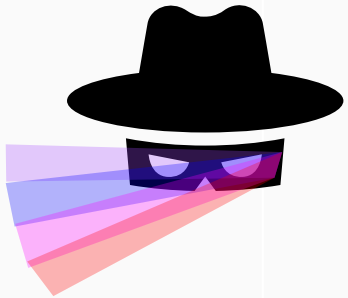


RandOS. Approx. 20k LoC Python.

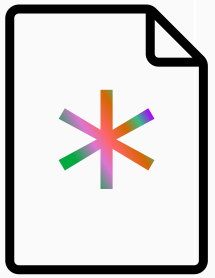


Implementation

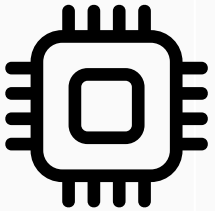
MileSan. Hardware IFT: Using CellIFT and HybridIFT. Software IFT: Approx. 15k LoC Python.



RandOS. Approx. 20k LoC Python.



Tested CPUs. Kronos, Rocket, BOOM, CVA6, OpenC910.



Evaluation

Q1. How fast can MileSan and RandOS find known vulnerabilities?

Evaluation

Q1. How fast can MileSan and RandOS find known vulnerabilities?

Q2. Can MileSan verify PoCs from prior work?

Evaluation

Q1. How fast can MileSan and RandOS find known vulnerabilities?

Q2. Can MileSan verify PoCs from prior work?

Q3. Can MileSan and RandOS find new vulnerabilities?

Evaluation

Q1. How fast can MileSan and RandOS find known vulnerabilities? 4.5× faster.

Q2. Can MileSan verify PoCs from prior work? Prior work reports false positives!

Q3. Can MileSan and RandOS find new vulnerabilities?

Can MileSan and Randos find new vulns.?

Vulnerability	DUT	Covert Channel	
Spectre-V2-TLB	BOOM	TLB	
Spectre-V4-TLB	BOOM	TLB	
Spectre-RSB-TLB	BOOM	TLB	
Trans. Meltdown	BOOM	TLB	1
cp-Spectre V2	BOOM	TLB	1
Spectre-SLS	CVA6	TLB	
cp-Spectre-SLS	CVA6	TLB	1
MDS	CVA6	TLB	1
Trans. MDS	CVA6	TLB	1
Spectre-V1	OpenC910	DCACHE	
Spectre-V1-TLB	OpenC910	TLB	
`div`†	OpenC910	DIV	
`divu`†	OpenC910	DIV	
`divw`†	OpenC910	DIV	
`divuw`†	OpenC910	DIV	
`rem`†	OpenC910	DIV	
`remu`†	OpenC910	DIV	
`remw`†	OpenC910	DIV	
`remuw`†	OpenC910	DIV	

cross-privilege

Can MileSan and Randos find new vulns.?

Vulnerability	DUT	Covert Channel	
Spectre-V2-TLB	BOOM	TLB	
Spectre-V4-TLB	BOOM	TLB	
Spectre-RSB-TLB	BOOM	TLB	
Trans. Meltdown	BOOM	TLB	1
cp-Spectre V2	BOOM	TLB	1
Spectre-SLS	CVA6	TLB	
cp-Spectre-SLS	CVA6	TLB	1
MDS	CVA6	TLB	1
Trans. MDS	CVA6	TLB	1
Spectre-V1	OpenC910	DCACHE	
Spectre-V1-TLB	OpenC910	TLB	
`div`†	OpenC910	DIV	
`divu`†	OpenC910	DIV	
`divw`†	OpenC910	DIV	
`divuw`†	OpenC910	DIV	
`rem`†	OpenC910	DIV	
`remu`†	OpenC910	DIV	
`remw`†	OpenC910	DIV	
`remuw`†	OpenC910	DIV	

cross-privilege

Can MileSan and Randos find new vulns.?

Vulnerability	DUT	Covert Channel	
Spectre-V2-TLB	BOOM	TLB	
Spectre-V4-TLB	BOOM	TLB	
Spectre-RSB-TLB	BOOM	TLB	
Trans. Meltdown	BOOM	TLB	1
cp-Spectre V2	BOOM	TLB	1
Spectre-SLS	CVA6	TLB	
cp-Spectre-SLS	CVA6	TLB	1
MDS	CVA6	TLB	1
Trans. MDS	CVA6	TLB	1
Spectre-V1	OpenC910	DCACHE	
Spectre-V1-TLB	OpenC910	TLB	
`div`†	OpenC910	DIV	
`divu`†	OpenC910	DIV	
`divw`†	OpenC910	DIV	
`divuw`†	OpenC910	DIV	
`rem`†	OpenC910	DIV	
`remu`†	OpenC910	DIV	
`remw`†	OpenC910	DIV	
`remuw`†	OpenC910	DIV	

cross-privilege

Can MileSan and Randos find new vulns.?

Leaking 1Mbps from kernel to user at 2GHz on BOOM

CVE-2025-29343

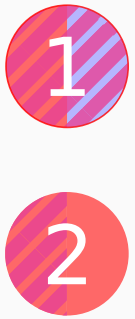
```
...
0x28778: remu a2,s0,gp
0x28780: rem a2,a2,tp
0x287c8: divw ra,gp,a2
# Branch below mispredicted taken.
0x287d0: blt gp,ra,0x28e00
...
# Meltdown gadget executed transiently.
0x28e00: lb sp,-1587(t1)
0x28e04: lh s1,-1010(sp)
```

End-to-end exploit

```
0  la ra, secret
1  la a2, buffer
…  ...              # Evict the TLB, delay t0 results.
6  beqz t0, correct_target
7  ld a0, 0(ra)     # Access secret.
8  andi a0, a0, 1   # Mask out a single bit.
9  slli a0, a0, 12  # Shift secret bit by page offset.
10 add a2, a2, a0   # Add secret bit to buffer.
11 ld a2, 0(a2)     # Encode the secret bit in the TLB.
12 correct_target:
…  ...              # Resume at correct path.
```

Can MileSan and Randos find new vulns.?

Vulnerability	DUT	Covert Channel
Spectre-V2-TLB	BOOM	TLB
Spectre-V4-TLB	BOOM	TLB
Spectre-RSB-TLB	BOOM	TLB
Trans. Meltdown	BOOM	TLB
cp-Spectre V2	BOOM	TLB
Spectre-SLS	CVA6	TLB
cp-Spectre-SLS	CVA6	TLB
MDS	CVA6	TLB
Trans. MDS	CVA6	TLB
Spectre-V1	OpenC910	DCACHE
Spectre-V1-TLB	OpenC910	TLB
`div`†	OpenC910	DIV
`divu`†	OpenC910	DIV
`divw`†	OpenC910	DIV
`divuw`†	OpenC910	DIV
`rem`†	OpenC910	DIV
`remu`†	OpenC910	DIV
`remw`†	OpenC910	DIV
`remuw`†	OpenC910	DIV

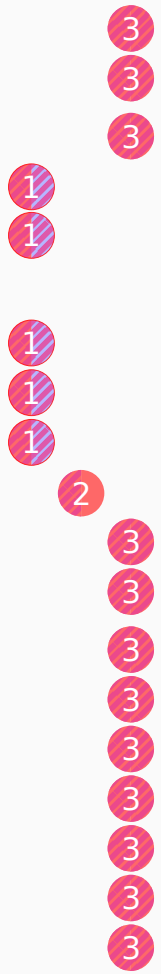


cross-privilege

cross-addr. space

Can MileSan and Randos find new vulns.?

Vulnerability	DUT	Covert Channel
Spectre-V2-TLB	BOOM	TLB
Spectre-V4-TLB	BOOM	TLB
Spectre-RSB-TLB	BOOM	TLB
Trans. Meltdown	BOOM	TLB
cp-Spectre V2	BOOM	TLB
Spectre-SLS	CVA6	TLB
cp-Spectre-SLS	CVA6	TLB
MDS	CVA6	TLB
Trans. MDS	CVA6	TLB
Spectre-V1	OpenC910	DCACHE
Spectre-V1-TLB	OpenC910	TLB
div†	OpenC910	DIV
divu†	OpenC910	DIV
divw†	OpenC910	DIV
divuw†	OpenC910	DIV
rem†	OpenC910	DIV
remu†	OpenC910	DIV
remw†	OpenC910	DIV
remuw†	OpenC910	DIV



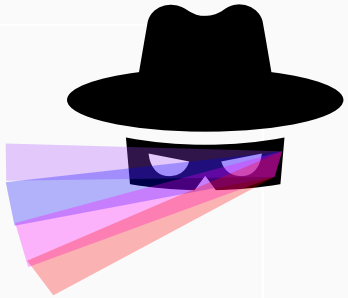
cross-privilege

cross-addr. space

within priv.+ addr. space

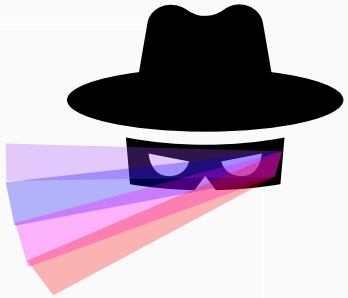
Conclusion

MileSan. Based on fundamental properties of information flows. Works for arbitrary programs and leakage.

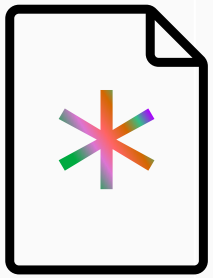


Conclusion

MileSan. Based on fundamental properties of information flows. Works for arbitrary programs and leakage.

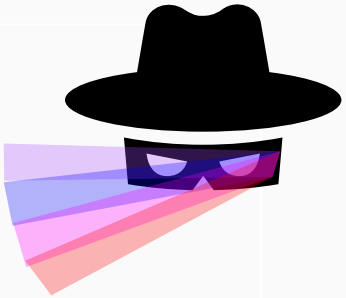


RandOS. Generates random operating systems. Tests all architectural isolation boundaries.

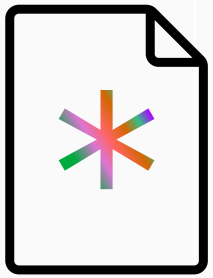


Conclusion

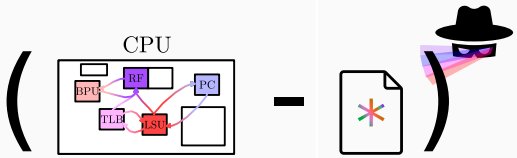
MileSan. Based on fundamental properties of information flows. Works for arbitrary programs and leakage.



RandOS. Generates random operating systems. Tests all architectural isolation boundaries.

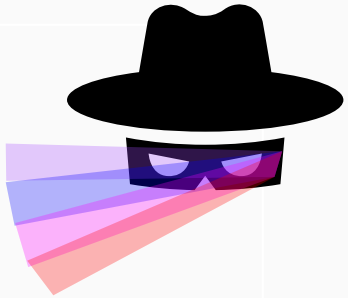


MileSan+RandOS. Found 19 new vulnerabilities on BOOM, CVA6 and OpenC910.

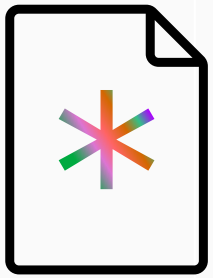


Conclusion

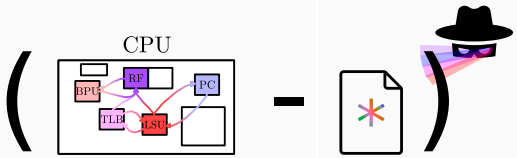
MileSan. Based on fundamental properties of information flows. Works for arbitrary programs and leakage.



RandOS. Generates random operating systems. Tests all architectural isolation boundaries.



MileSan+RandOS. Found 19 new vulnerabilities on BOOM, CVA6 and OpenC910.



more info + source code