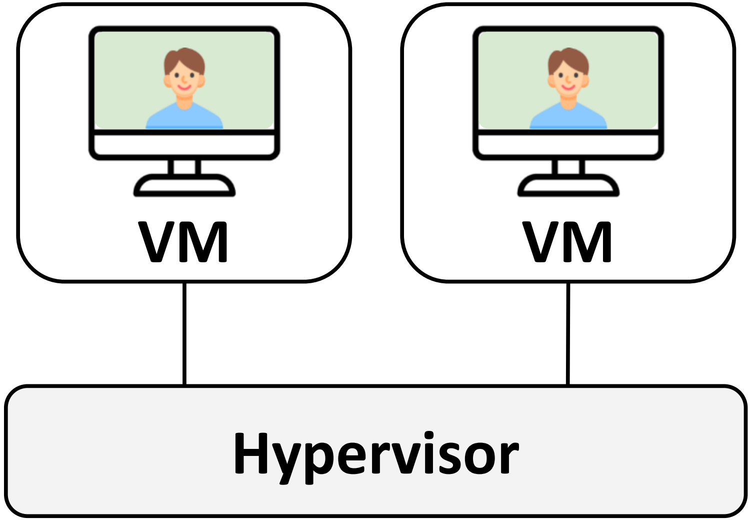
MundoFuzz: Hypervisor Fuzzing with Statistical Coverage Testing and Grammar Inference

Cheolwoo Myung[†], Gwangmu Lee[‡], and Byoungyoung Lee[†]

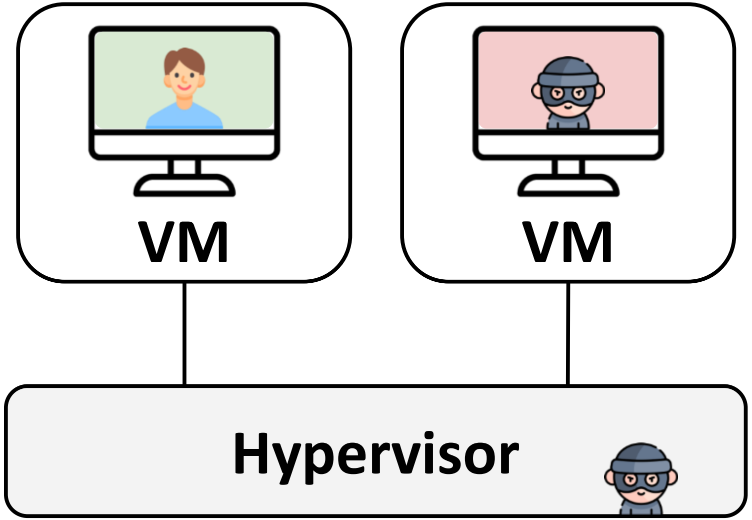
Seoul National University[†], EPFL[‡]

Hypervisor: Manager of Virtual Machine



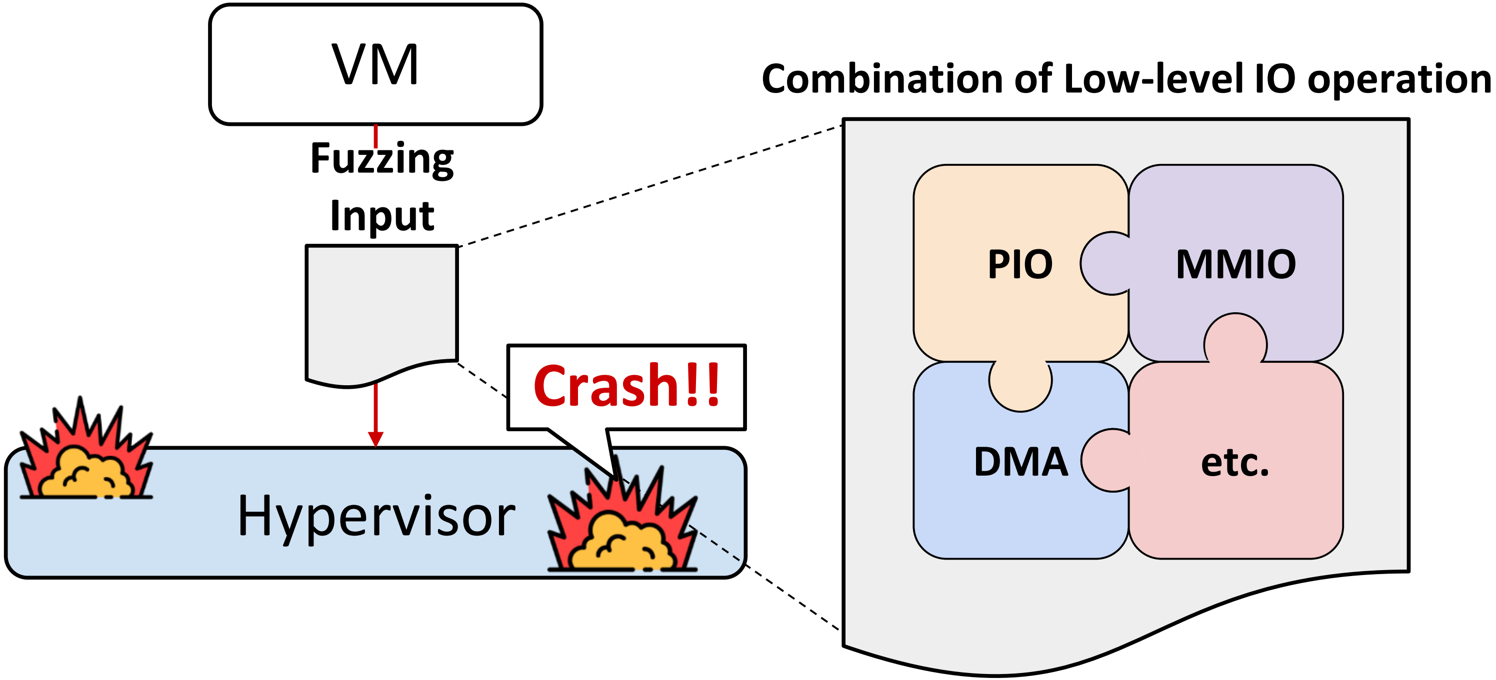
- Allow **remote users** to run guest VMs

Hypervisor can be attacked by Malicious VM



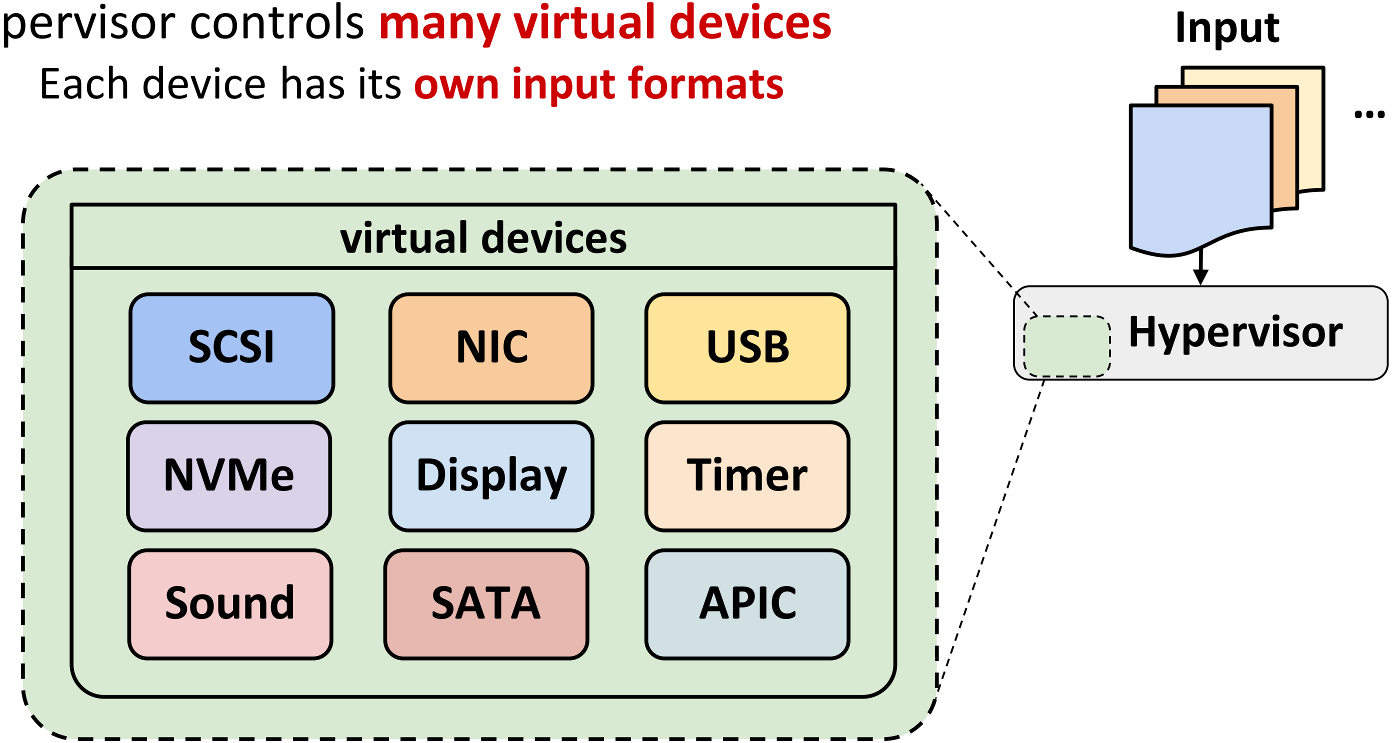
- One of guest VMs can be **malicious**

Fuzzing: Feed Random Inputs to Hypervisor



Motivation: Too many devices, too many formats

- Hypervisor controls **many virtual devices**
  - Each device has its **own input formats**



Limitations of Current Hypervisor Fuzzing

#1. Generating **random inputs** per device

Limitation ⇒ Cannot explore deep states of the devices

#2. Relying on **manual input grammars** per device

Limitation ⇒ Require unacceptable manual work to specify grammar rules

Let's fuzz hypervisor with grammar-awareness using automatic grammar inference!

Overview of MundoFuzz

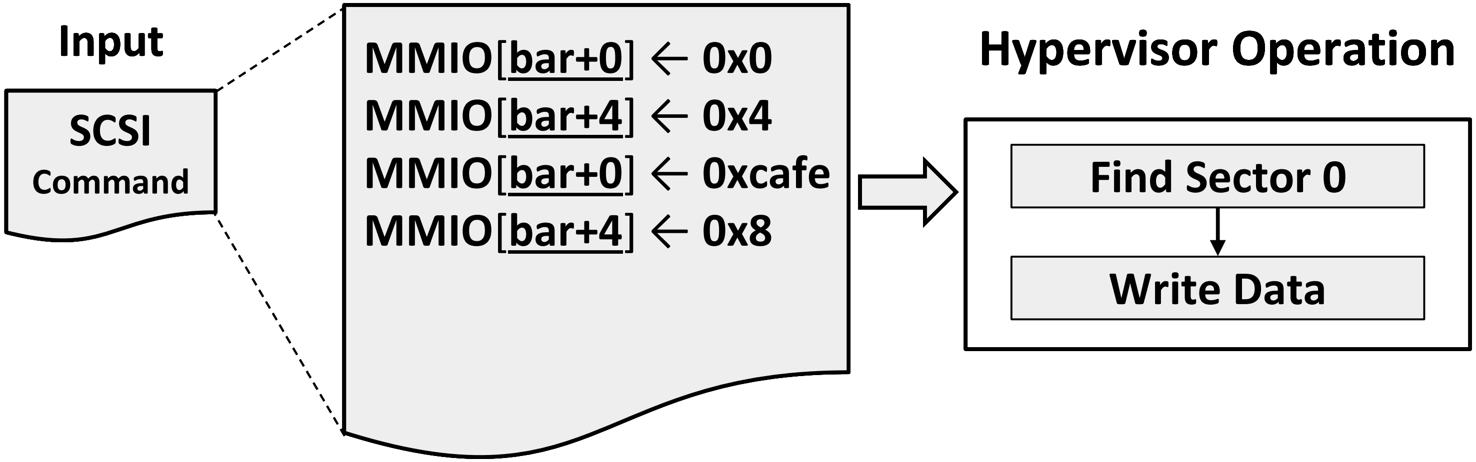
- Augment hypervisor fuzzing capability with automatic grammar inference

- **Challenges** in inferring hypervisor grammars
    - #1. Hypervisor grammars have **hidden input semantics** per device
    - #2. Hardware features of hypervisor introduce **coverage noises**

- **Our approach**
    - **Statistical and differential learning with coverage**

Challenge 1: Hidden Input Semantics

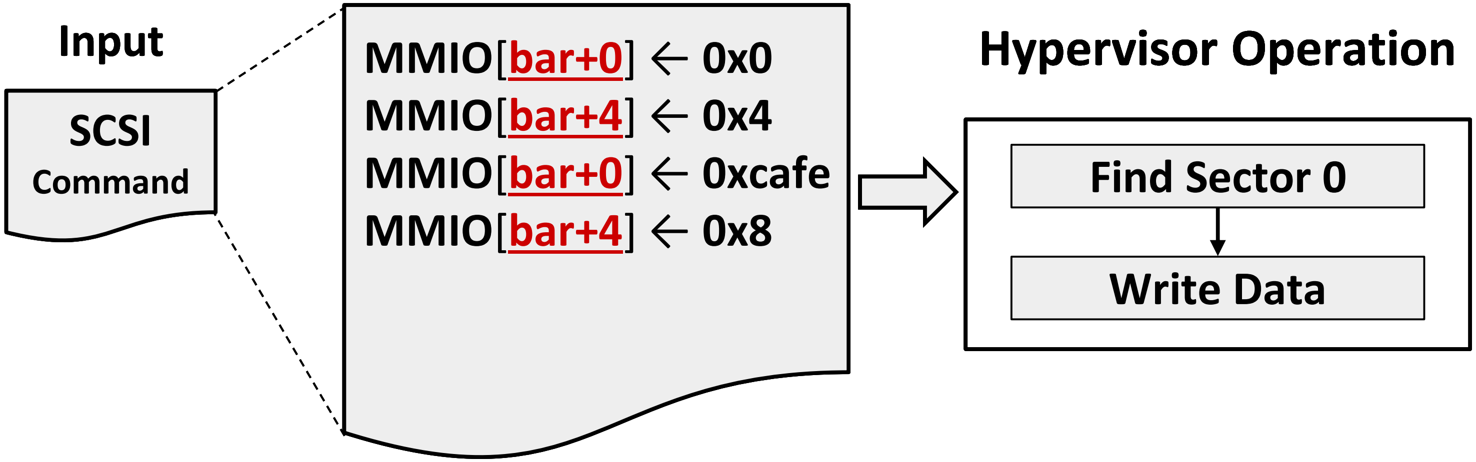
- Too difficult to infer **hidden input semantics** behind the hypervisor input



Example: SCSI command input

Challenge 1: Hidden Input Semantics

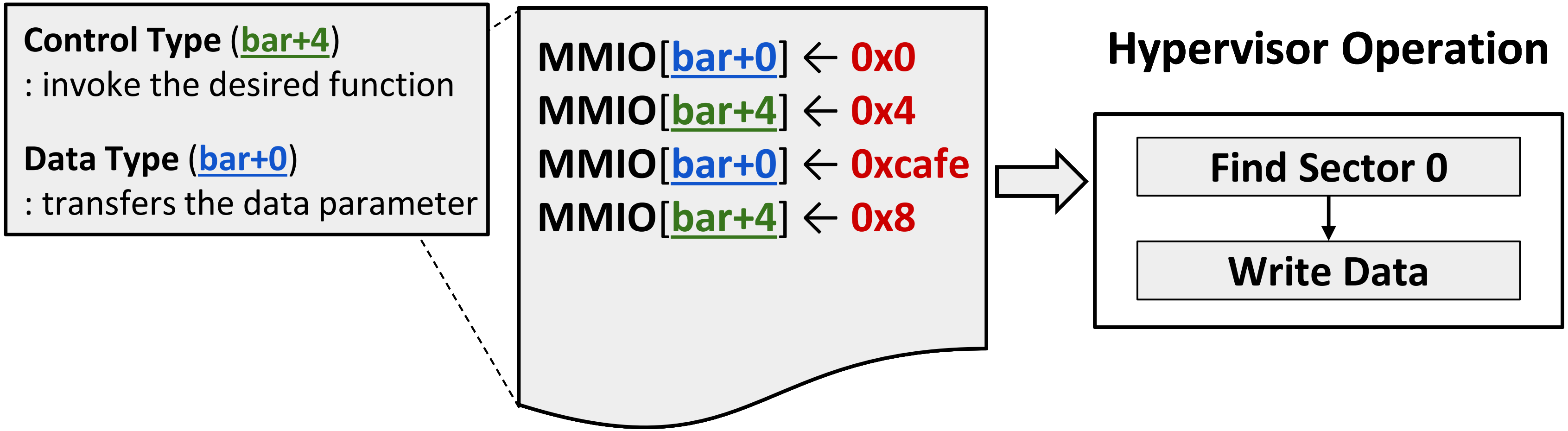
- Too difficult to infer **hidden input semantics** behind the hypervisor input
  - **IO address semantics:** correct semantic command should be given



Example: SCSI command input

Challenge 1: Hidden Input Semantics

- Too difficult to infer **hidden input semantics** behind the hypervisor input
  - **IO address semantics**: correct semantic command should be given

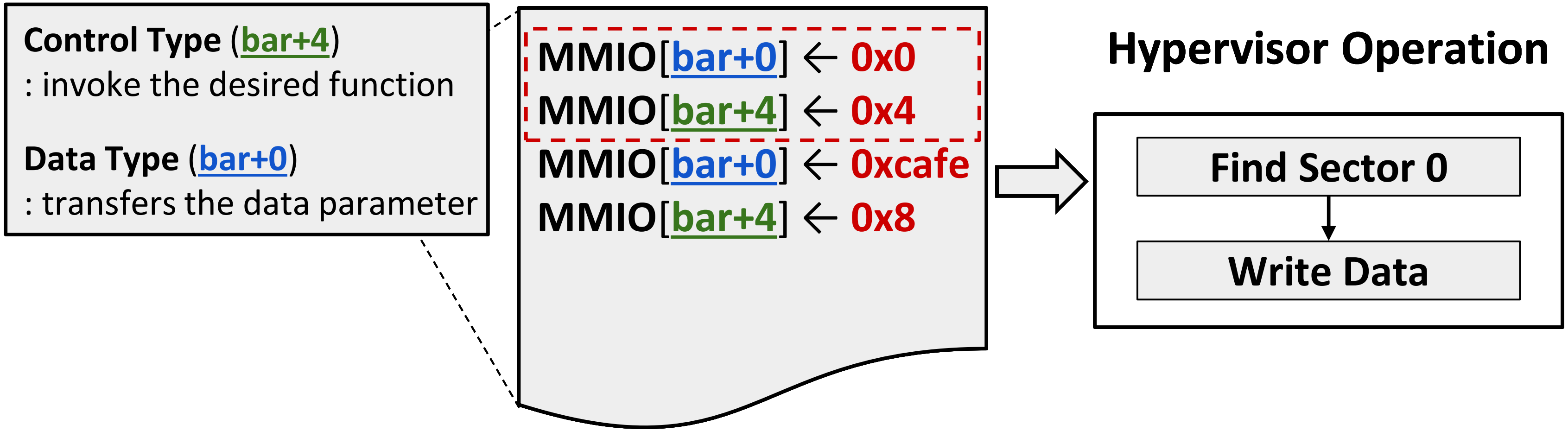


Example: SCSI command input

Challenge 1: Hidden Input Semantics

- Too difficult to infer **hidden input semantics** behind the hypervisor input
    - **IO address semantics:** correct semantic

Invoke the "Find Sector" func. (0x4) with the parameter (0x0)

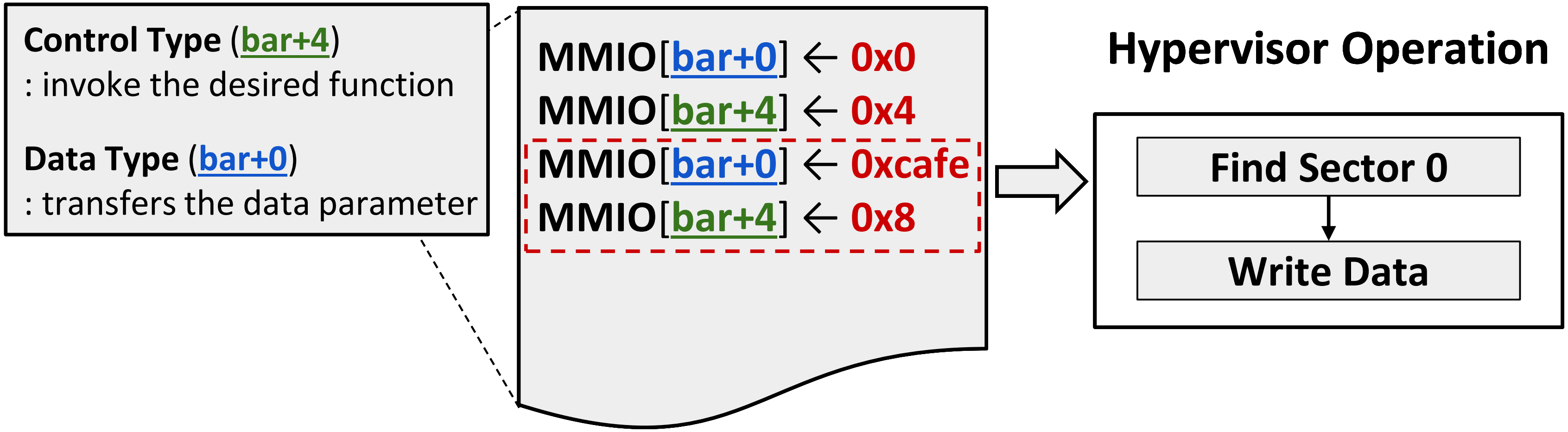


Example: SCSI command input

Challenge 1: Hidden Input Semantics

- Too difficult to infer **hidden input semantics** behind the hypervisor input
  - **IO address semantics**: correct semantic

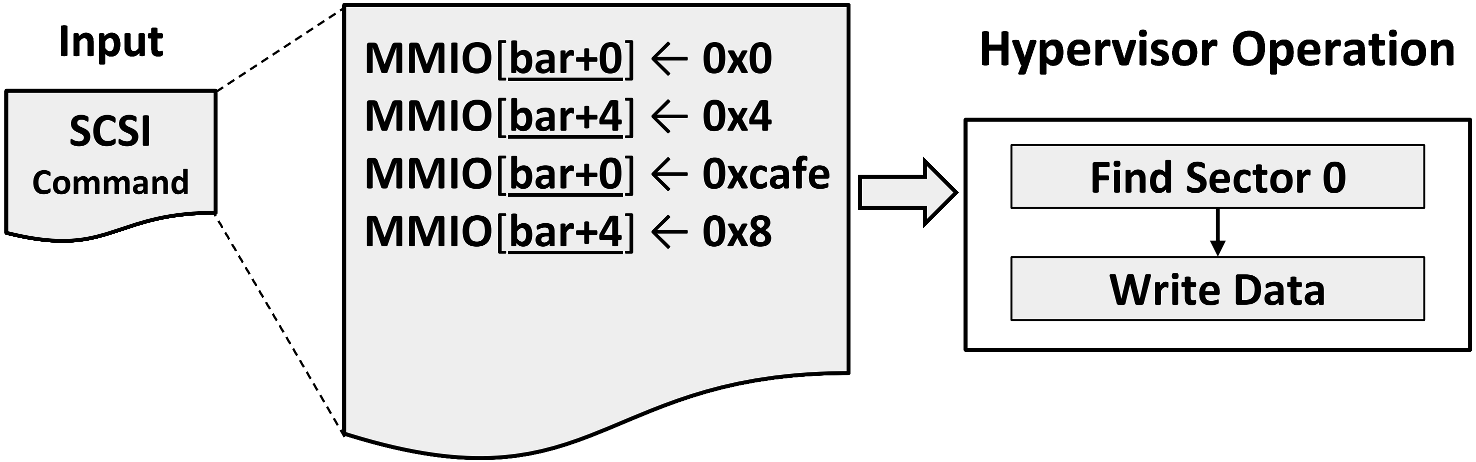
Invoke the "Write Data" func. (0x8) with the parameter (0xcafe)



Example: SCSI command input

Challenge 1: Hidden Input Semantics

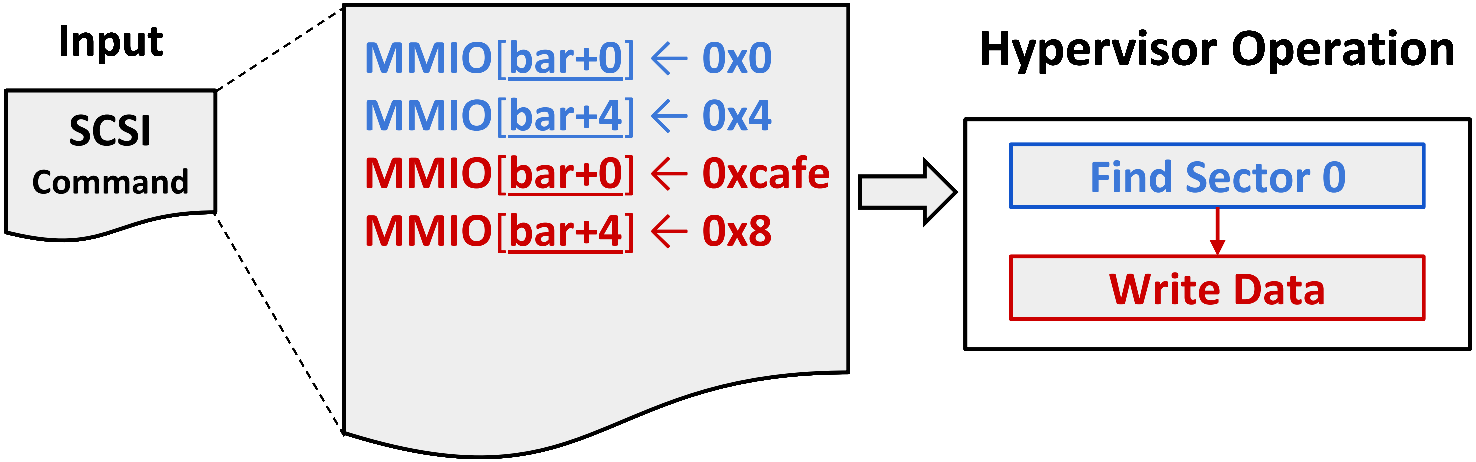
- Too difficult to infer **hidden input semantics** behind the hypervisor input
  - **IO address semantics:** correct semantic command should be given
  - **IO order semantics:** correct semantic order should be given



Example: SCSI command input

Challenge 1: Hidden Input Semantics

- Too difficult to infer **hidden input semantics** behind the hypervisor input
  - **IO address semantics:** correct semantic command should be given
  - **IO order semantics:** correct semantic order should be given

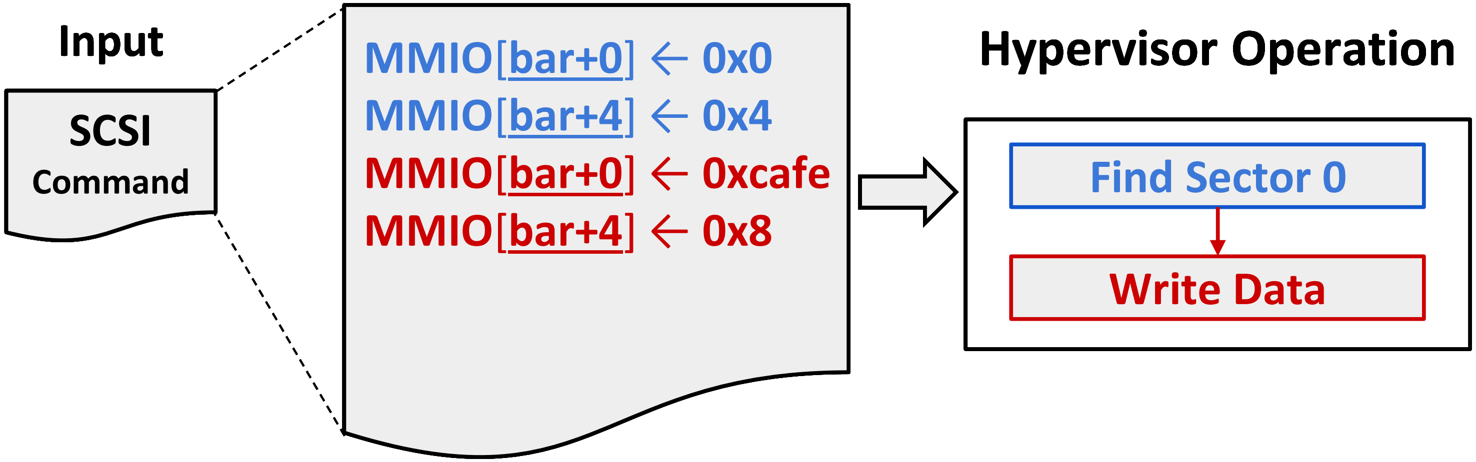


Example: SCSI command input

Challenge 1: Hidden Input Semantics

- Too difficult to infer **hidden input semantics** behind the hypervisor input
  - IO address semantics: correct semantic command should be given
  - IO

“Find Sector” should be performed before “Write Data”



Example: SCSI command input

Solution 1: Differential Learning on Input Semantics

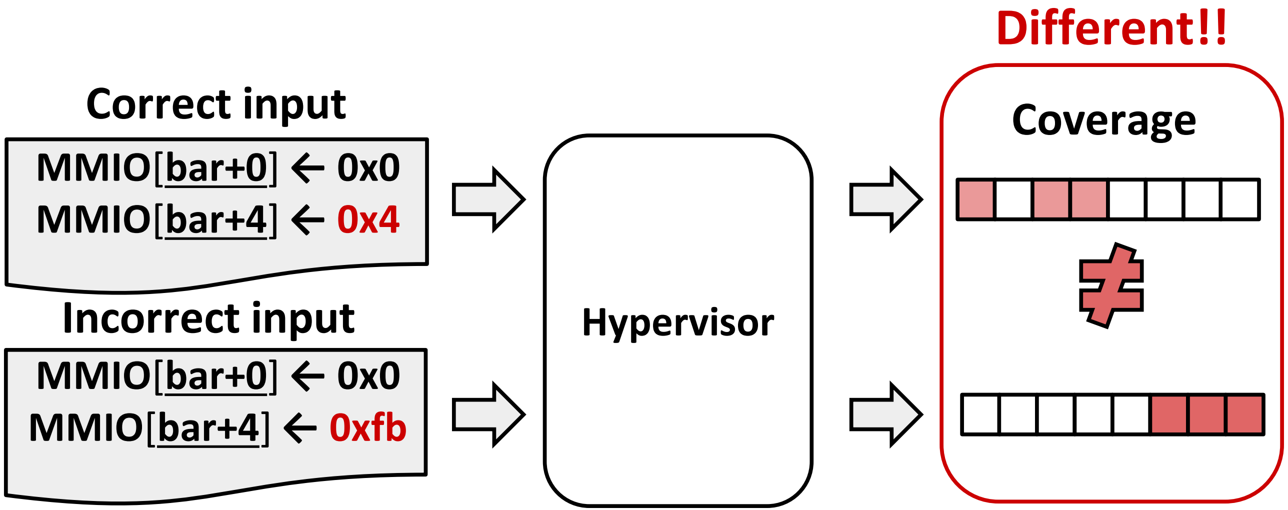
#1. IO address semantics

- **Different IO address types react to IO address values differently**

Solution 1: Differential Learning on Input Semantics

#1. IO address semantics

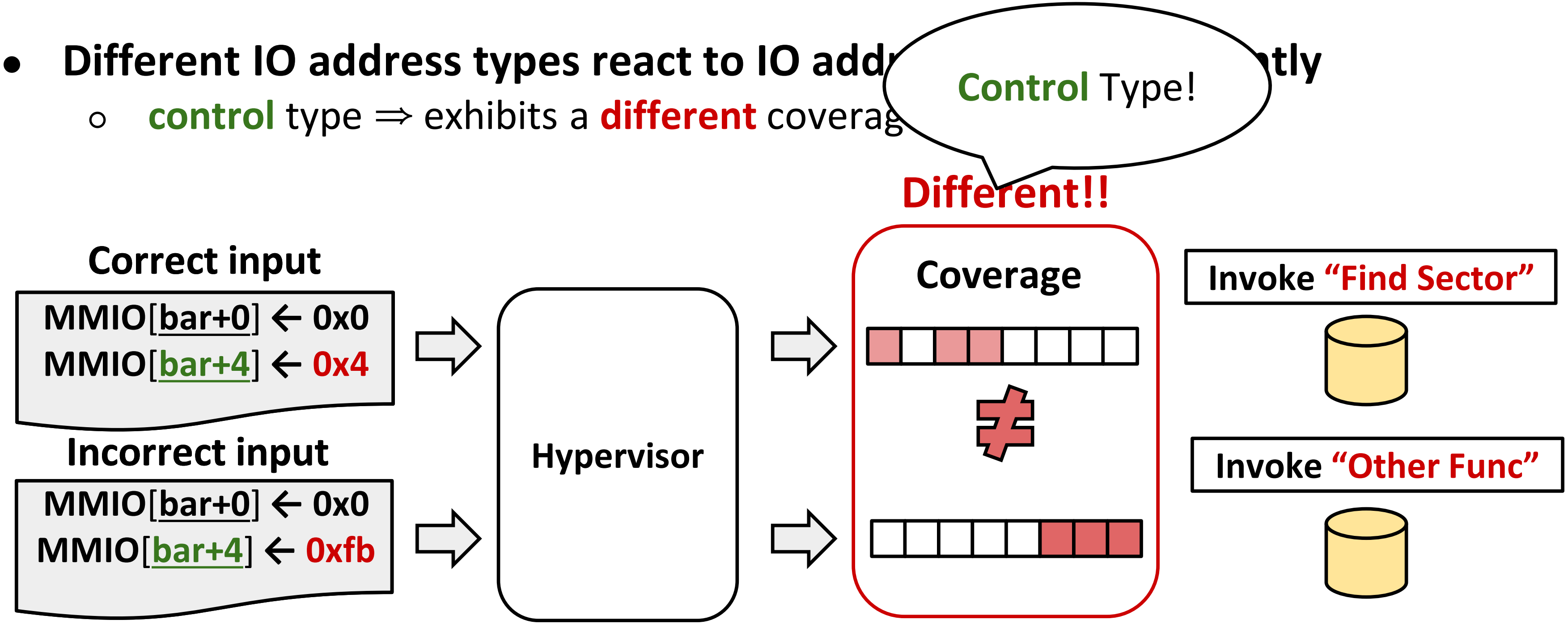
- **Different IO address types react to IO address values differently**
  - **control** type ⇒ exhibits a **different** coverage



Solution 1: Differential Learning on Input Semantics

#1. IO address semantics

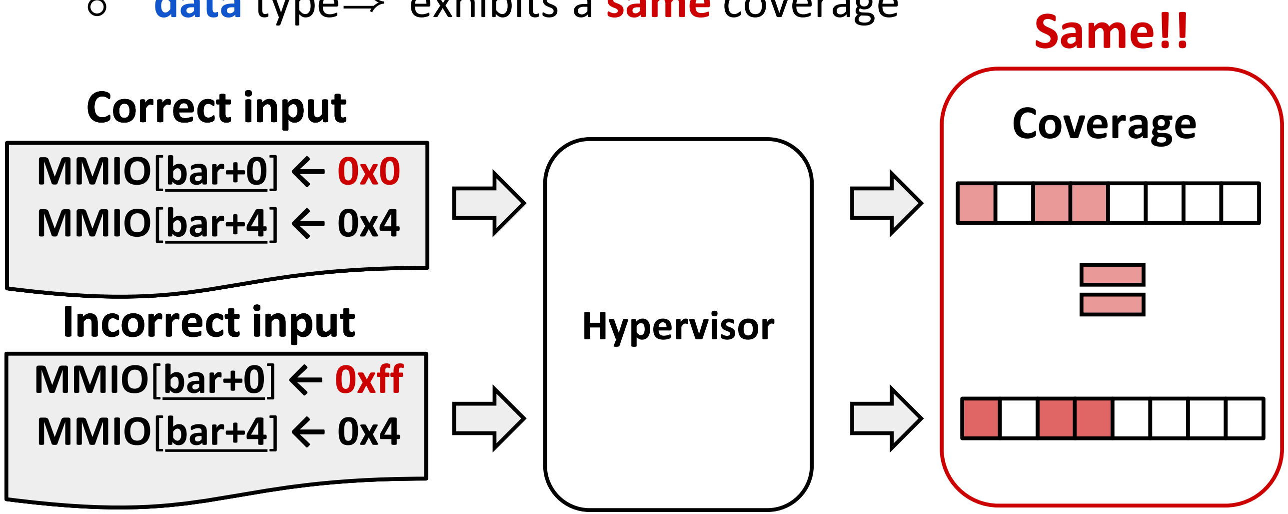
- **Different IO address types react to IO addr**[…]**tly**
  - control type ⇒ exhibits a **different** coverag[…]



Solution 1: Differential Learning on Input Semantics

#1. IO address semantics

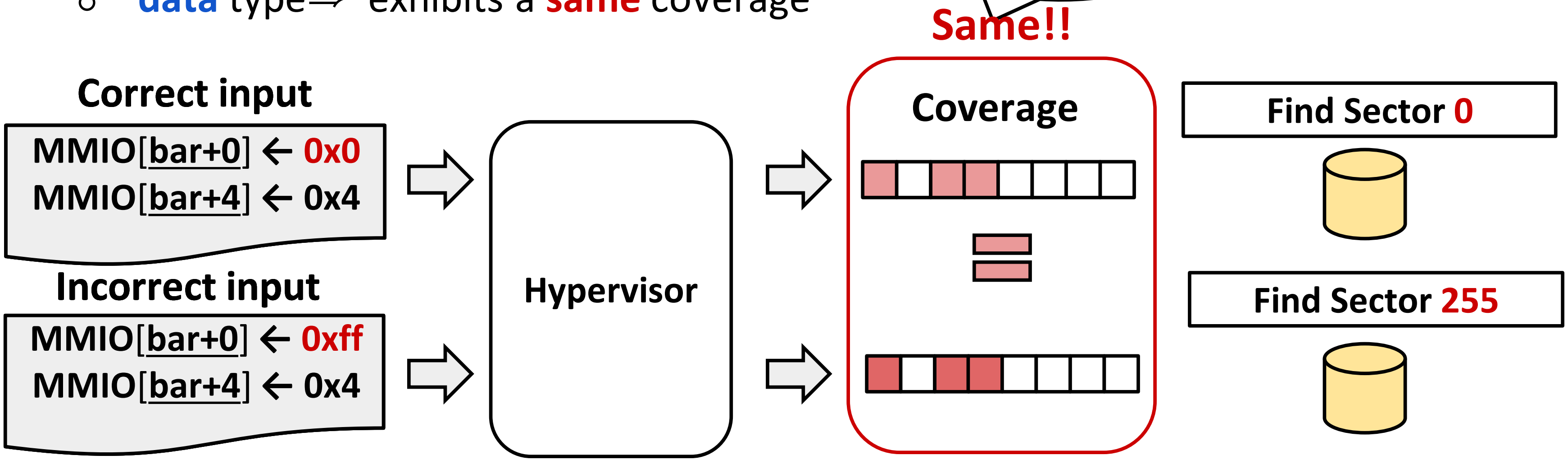
- **Different IO address types react to IO address values differently**
  - control type ⇒ exhibits a **different** coverage
  - data type⇒ exhibits a **same** coverage



Solution 1: Differential Learning on Input Semantics

#1. IO address semantics

- **Different IO address types react to IO addr ently**
  - control type ⇒ exhibits a **different** coverag
  - data type⇒ exhibits a **same** coverage



Solution 1: Differential Learning on Input Semantics

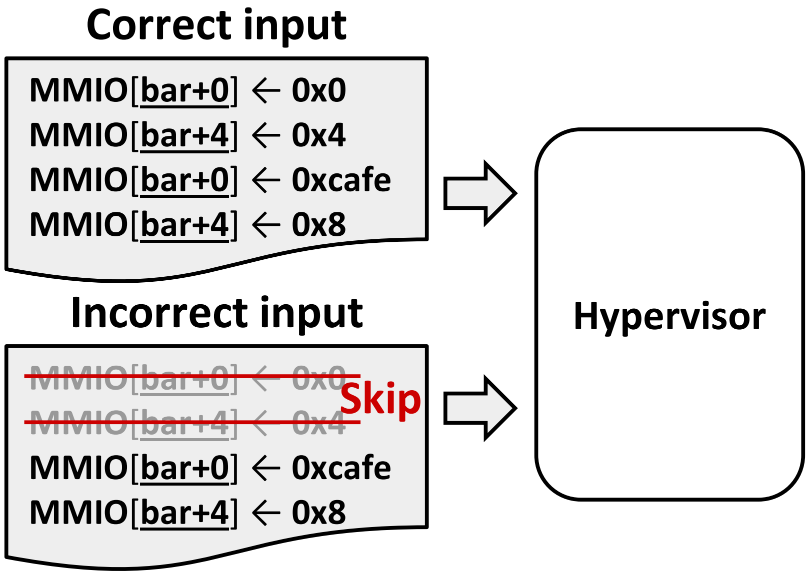
#2. IO order semantics

- **IO operations wouldn’t work correctly without prerequisite IO operations**
  - **absence of IO operations** ⇒ may distort **some following coverage**

Solution 1: Differential Learning on Input Semantics

#2. IO order semantics

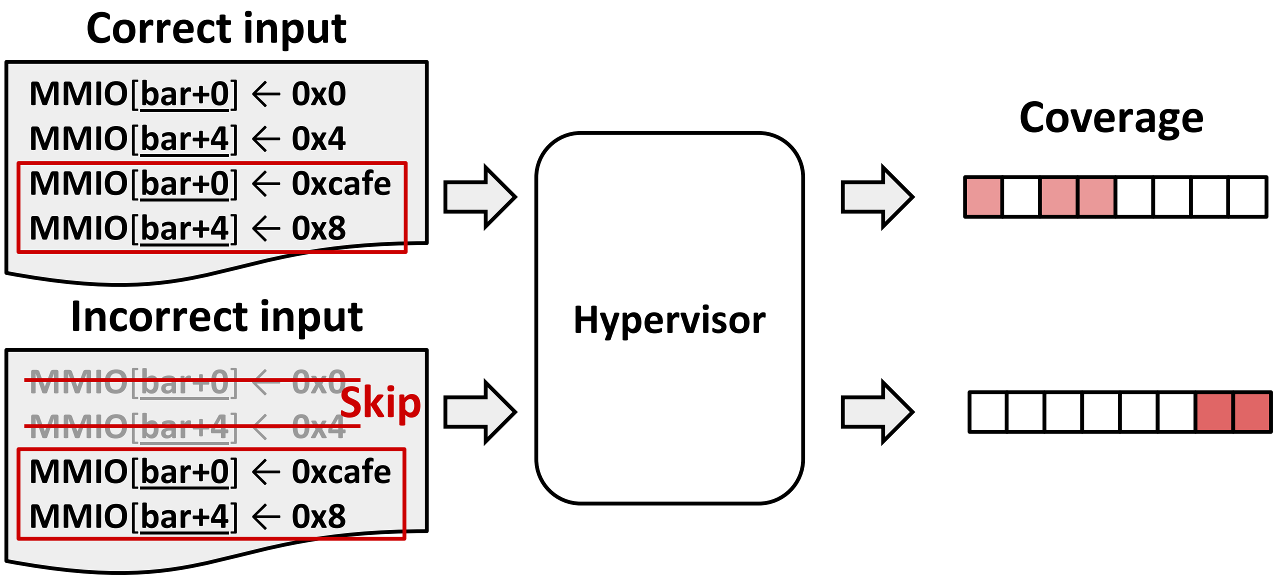
- **IO operations wouldn't work correctly without prerequisite IO operations**
  - **absence of IO operations ⇒** may distort **some following coverage**



Solution 1: Differential Learning on Input Semantics

#2. IO order semantics

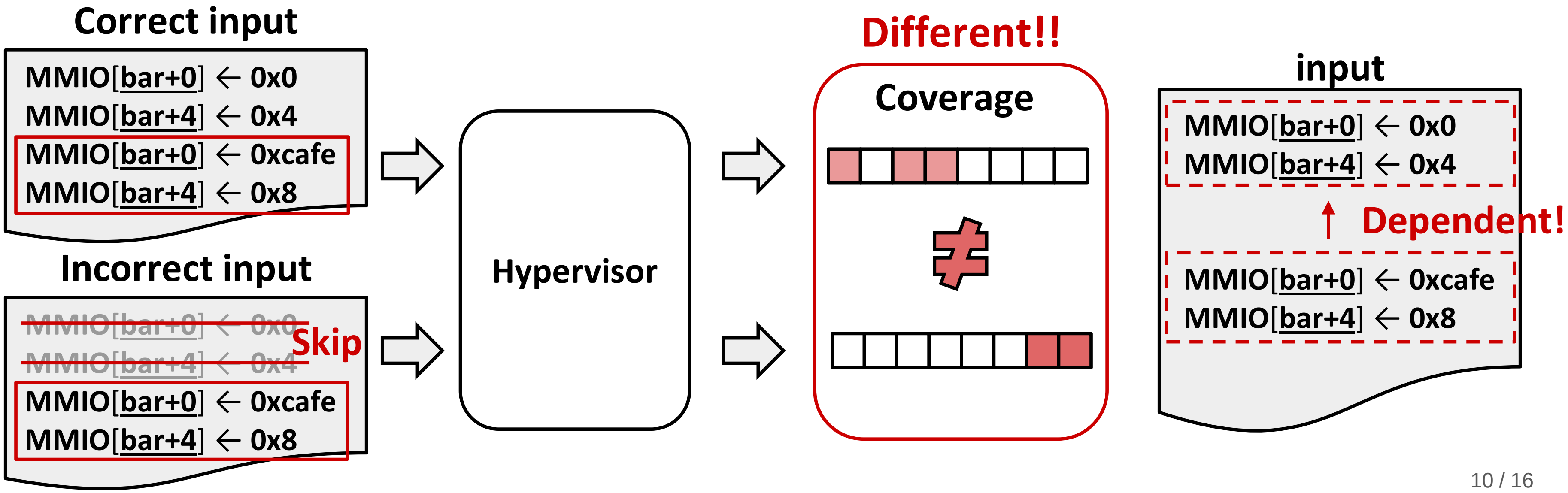
- **IO operations wouldn't work correctly without prerequisite IO operations**
  - **absence of IO operations** ⇒ may distort **some following coverage**



Solution 1: Differential Learning on Input Semantics

#2. IO order semantics

- **IO operations wouldn't work correctly without prerequisite IO operations**
  - **absence of IO operations ⇒** may distort **some following coverage**

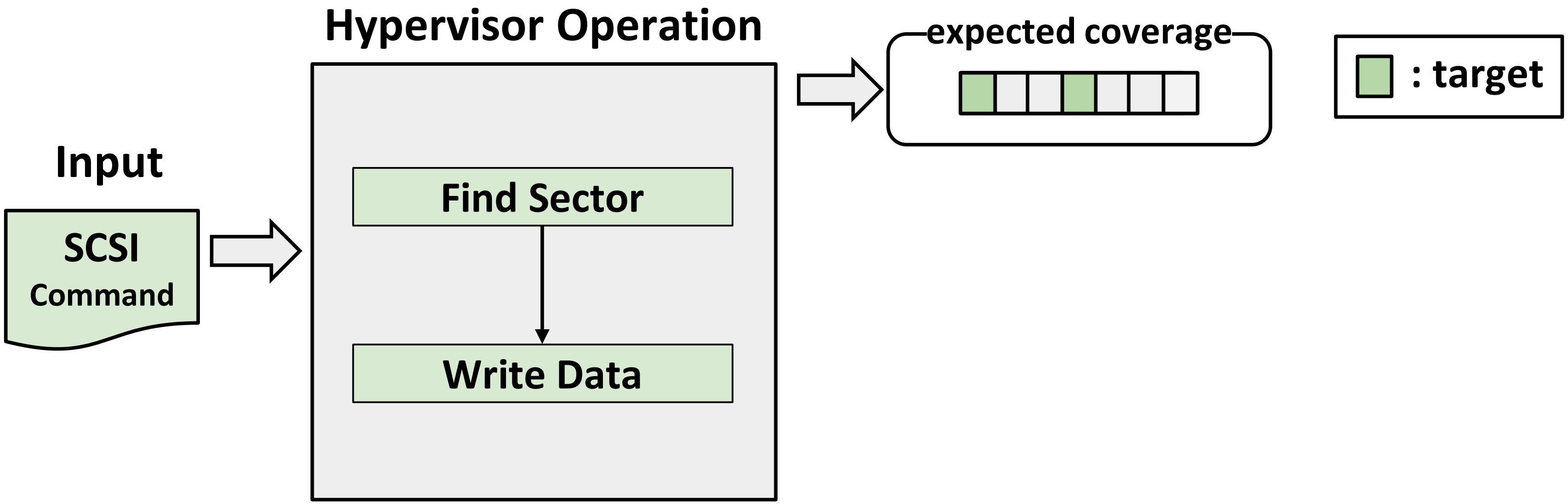


Challenge 2: Coverage Noises

- The measured input coverage includes **unwanted coverage**
  - due to the asynchronous event handling (e.g., timer, interrupt event)
  - asynchronous event introduces **non-deterministic (noise) coverage**

Challenge 2: Coverage Noises

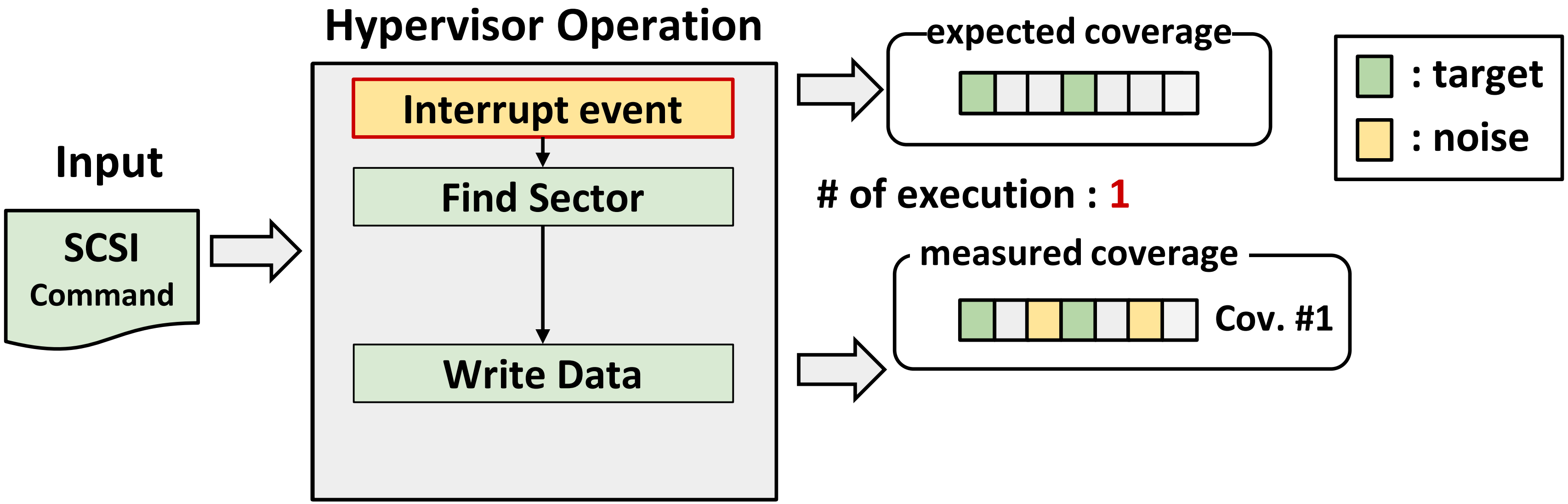
- The measured input coverage includes **unwanted coverage**
  - due to the asynchronous event handling (e.g., timer, interrupt event)
  - asynchronous event introduces **non-deterministic (noise) coverage**



Example: SCSI command input

Challenge 2: Coverage Noises

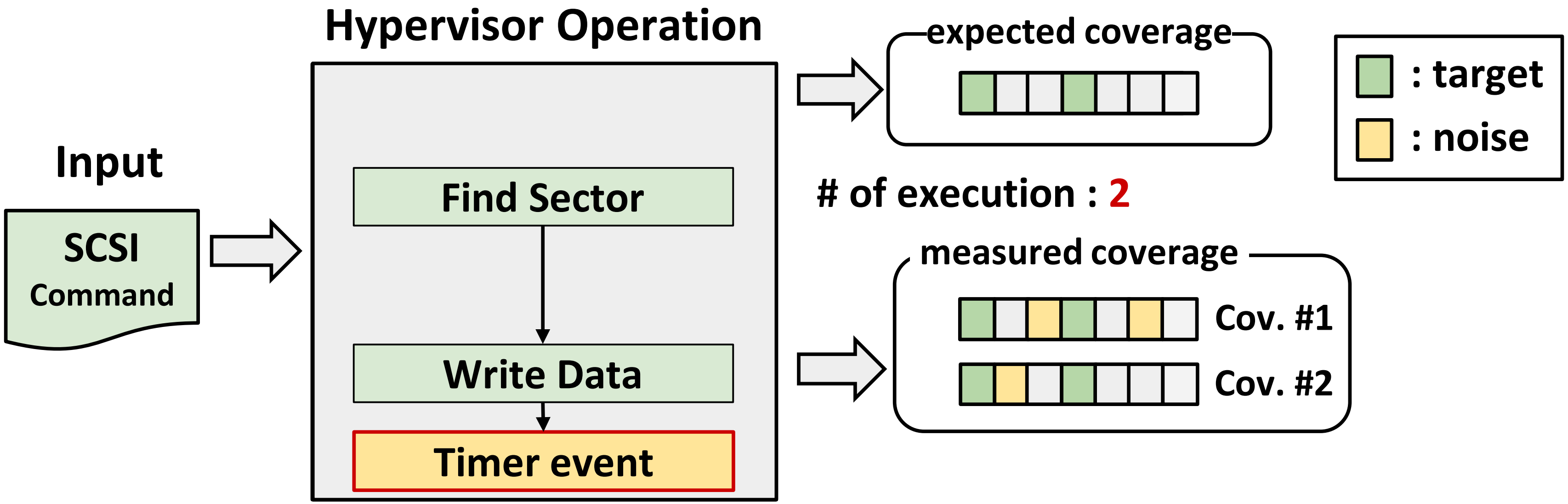
- The measured input coverage includes **unwanted coverage**
  - due to the asynchronous event handling (e.g., timer, interrupt event)
  - asynchronous event introduces **non-deterministic (noise) coverage**



Example: SCSI command input

Challenge 2: Coverage Noises

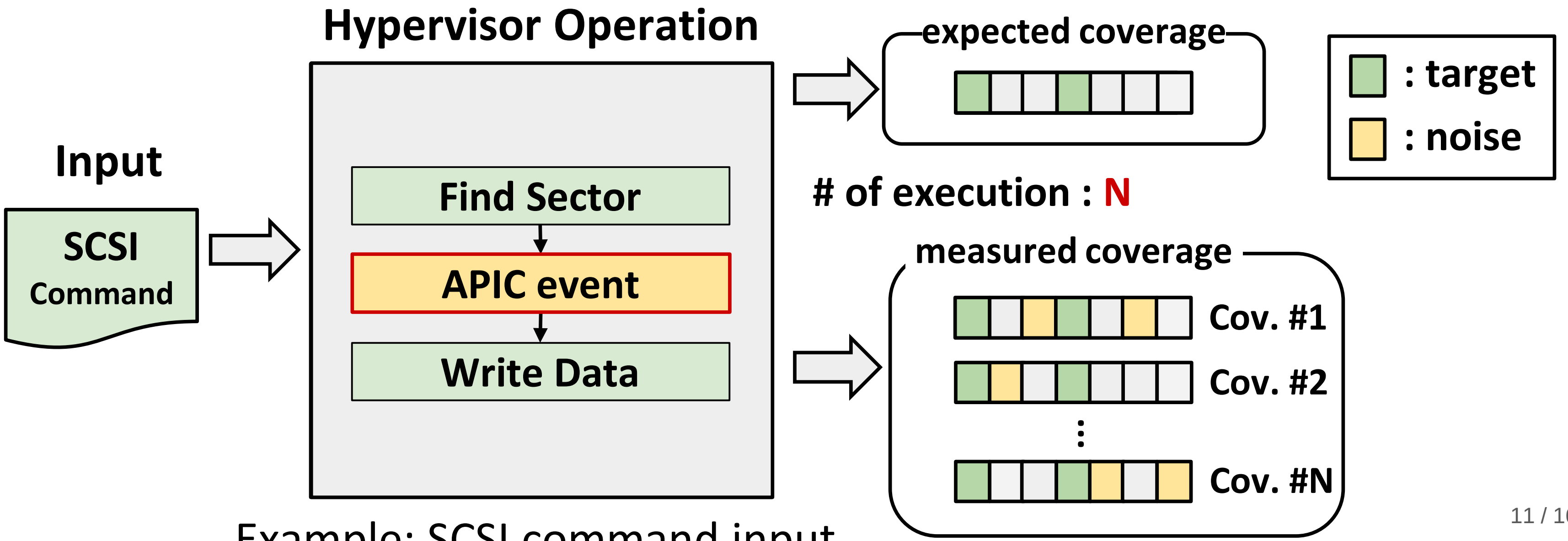
- The measured input coverage includes **unwanted coverage**
  - due to the asynchronous event handling (e.g., timer, interrupt event)
  - asynchronous event introduces **non-deterministic (noise) coverage**



Example: SCSI command input

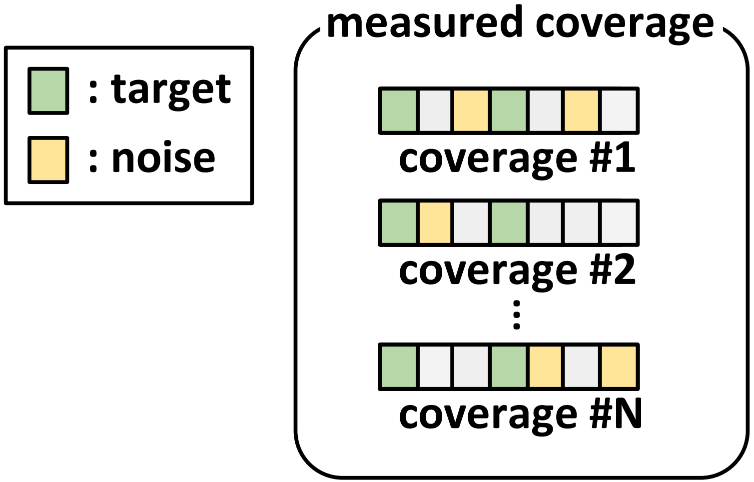
Challenge 2: Coverage Noises

- The measured input coverage includes **unwanted coverage**
  - due to the asynchronous event handling (e.g., timer, interrupt event)
  - asynchronous event introduces **non-deterministic (noise) coverage**

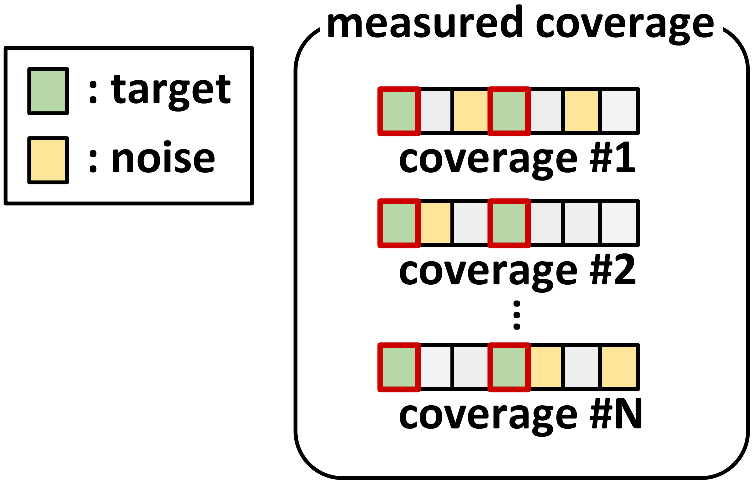


Example: SCSI command input

Solution 2: Statistical Differential Coverage Measurement

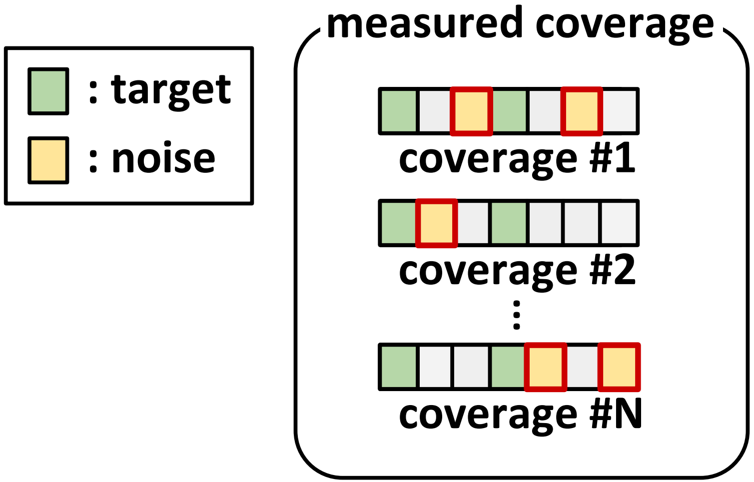


Solution 2: Statistical Differential Coverage Measurement



- **Target coverage** (■)
  - is always captured for all execution

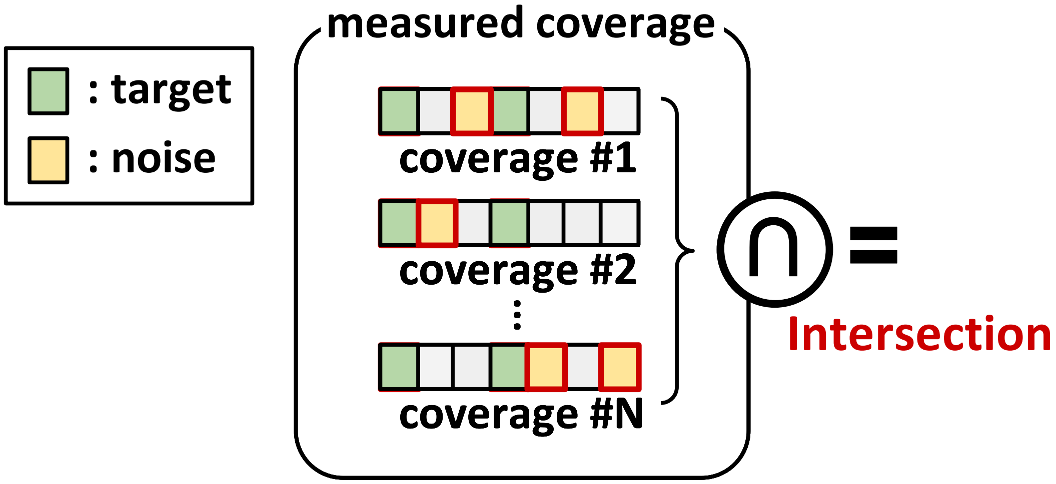
Solution 2: Statistical Differential Coverage Measurement



- Target coverage ( )
  - is always captured for all execution
- Noise coverage ( )
  - is captured differently for each execution

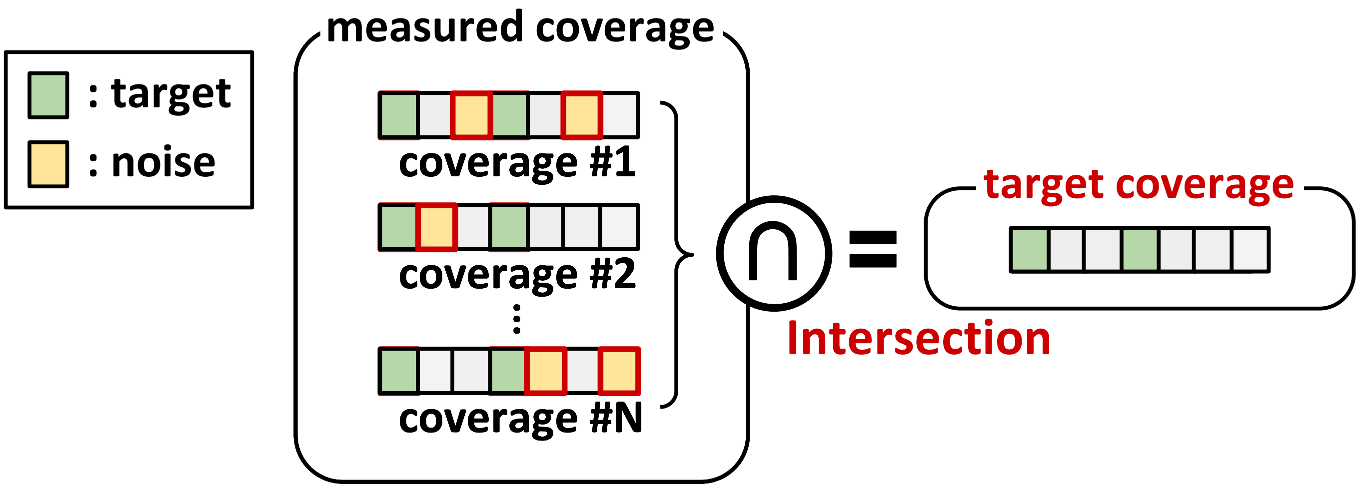
Solution 2: Statistical Differential Coverage Measurement

- Remove noise coverage **by intersecting all measured coverages**

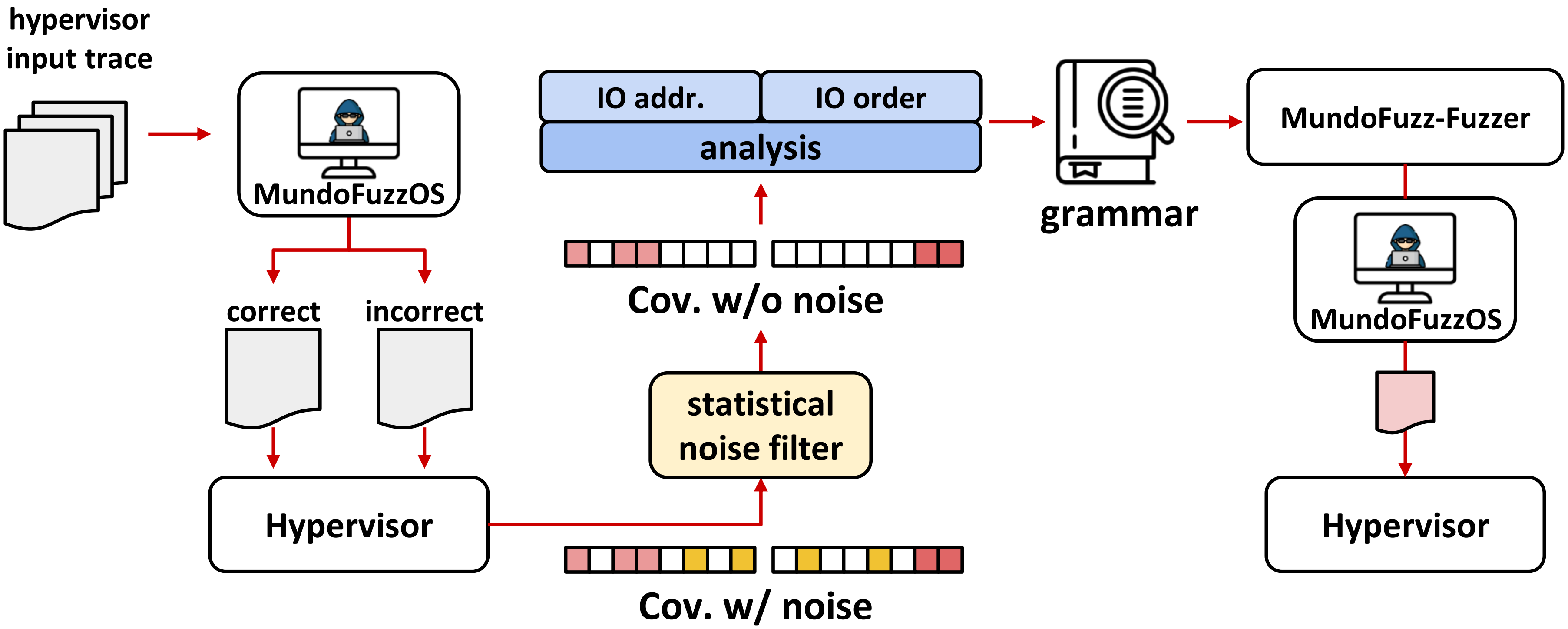


Solution 2: Statistical Differential Coverage Measurement

- Remove noise coverage **by intersecting all measured coverages**
  - the result only contains **target coverage**



Architecture of MundoFuzz



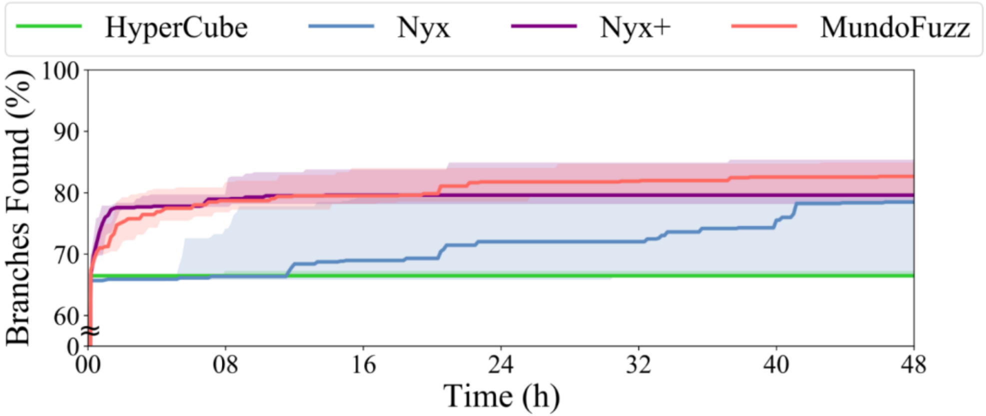
What MundoFuzz Found?

- MundoFuzz found new 40 bugs in QEMU and Bhyve
  - 23 bugs in QEMU
  - 17 bugs in Bhyve
  - 9 of these were acknowledged as CVEs

Hypervisor	Bug Types	Numbers
QEMU	Use-after-free	3
	Heap Overflow	2
	Segmentation Fault	3
	Infinite Loop	3
	Stack Overflow	1
	Assertion	11
Bhyve	Segmentation Fault	4
	Floating Point Exception	1
	Assertion	12

Our result

- Overall coverage: MundoFuzz outperforms state-of-art hypervisor fuzzer
  - HyperCube: **+4.91%**
  - Nyx: **+6.60%**
- MundoFuzz shows higher coverage than Nyx+ (with manual grammar rule)
  - for **USB-XHCI device** (48 hours)



Conclusion

- Proposed MundoFuzz, a hypervisor fuzzing technique
  - statistically removes noise coverage in raw coverage
  - automatically learns the grammar using two hidden semantics

- MundoFuzz discovered 40 new bugs (including 9 CVEs)

- MundoFuzz presented better coverage, compared to state of the arts.

Thank you!

Q & A

Contact Cheolwoo Myung
Ph.D. Student at Seoul National University (SNU)
cwmyung@snu.ac.kr