서울대학교[†]
SEOUL NATIONAL UNIVERSITY

Samsung Research[‡]

Constraint-guided Directed Greybox Fuzzing

Gwangmu Lee

SEOUL NATIONAL UNIVERSITY

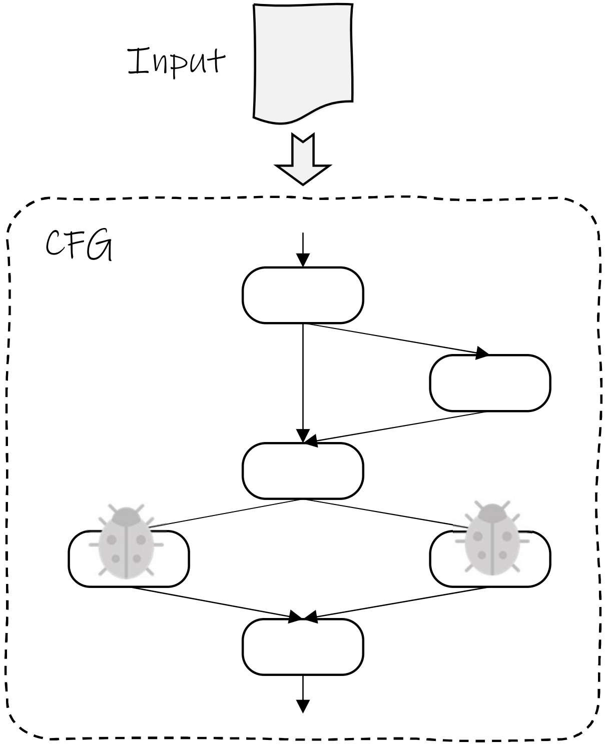
✉ gwangmu@snu.ac.kr
⌂ https://gwangmu.github.io

POSTDOC
JOB WANTED
POSITION

Authors

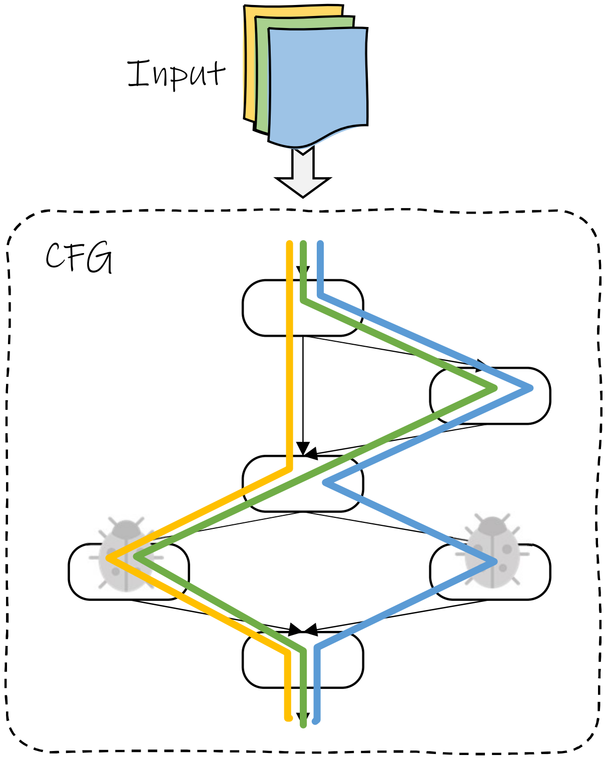
Gwangmu Lee[†] Woochul Shim[‡] Byoungyoung Lee[†]

Greybox Fuzzing [AFL], [libFuzzer]



Explore control flow w/ random inputs.

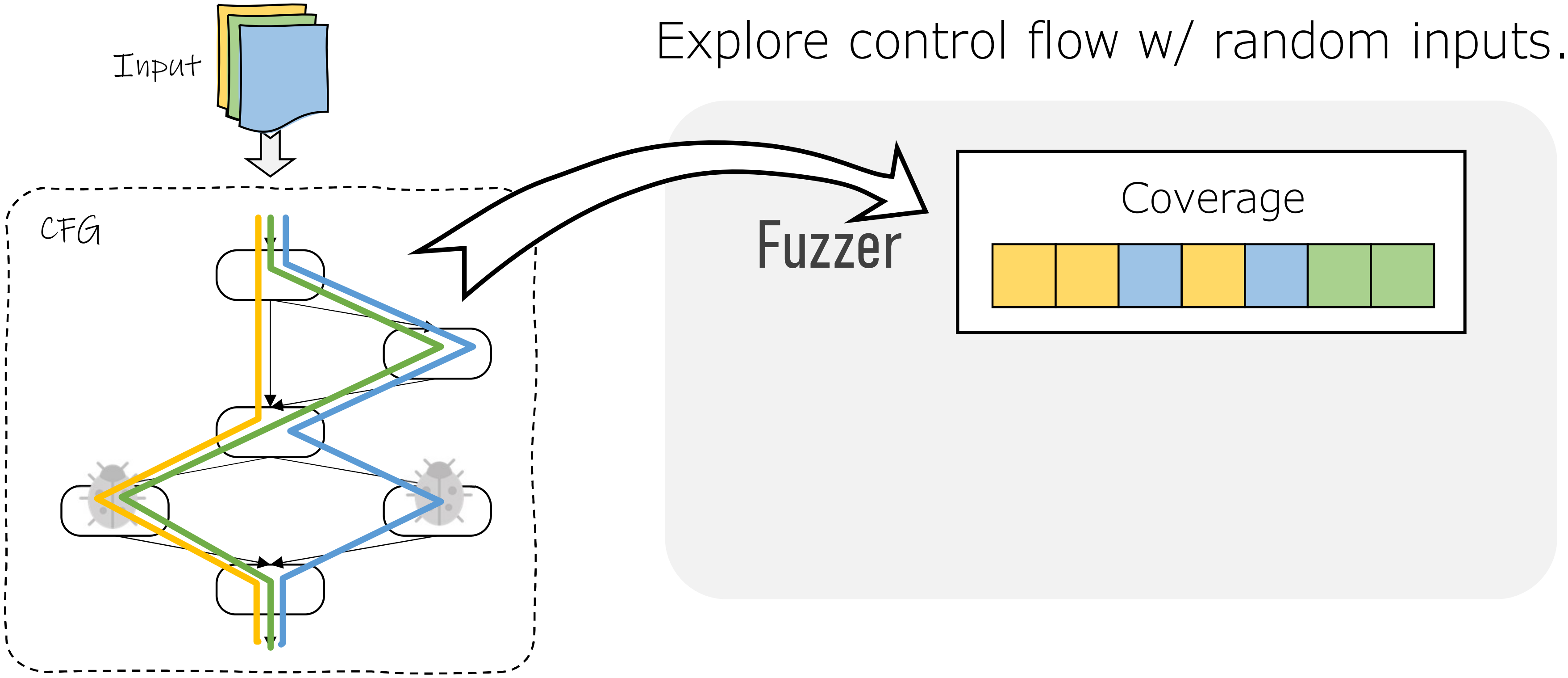
Greybox Fuzzing [AFL], [libFuzzer]



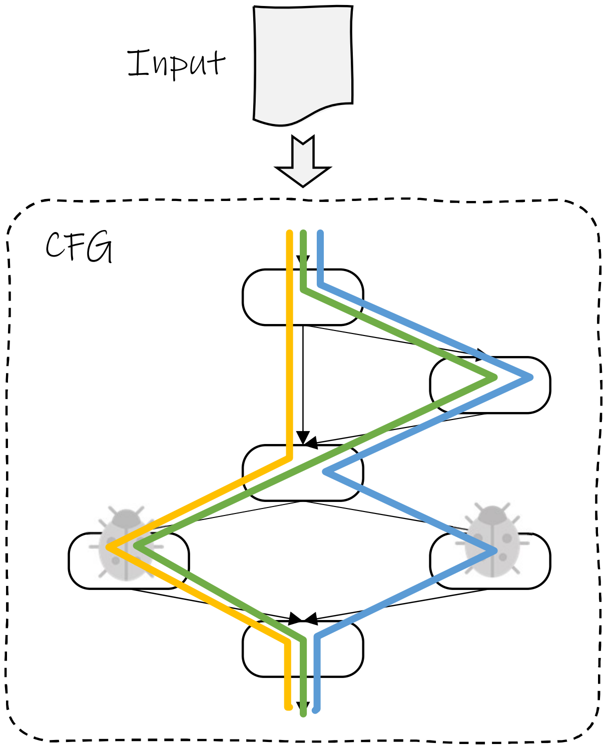
Explore control flow w/ random inputs.

Greybox Fuzzing [AFL], [libFuzzer]

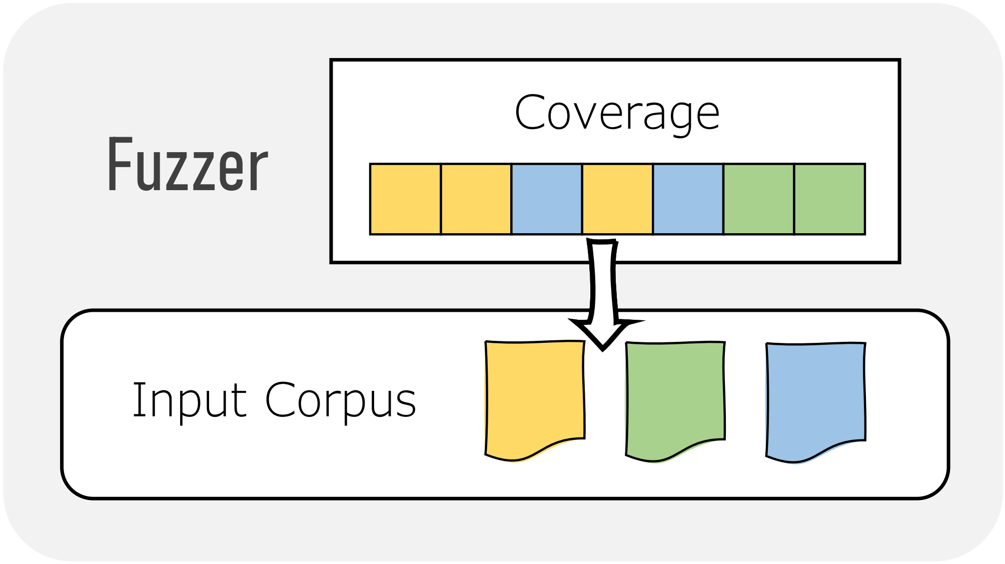
Explore control flow w/ random inputs.



Greybox Fuzzing [AFL], [libFuzzer]

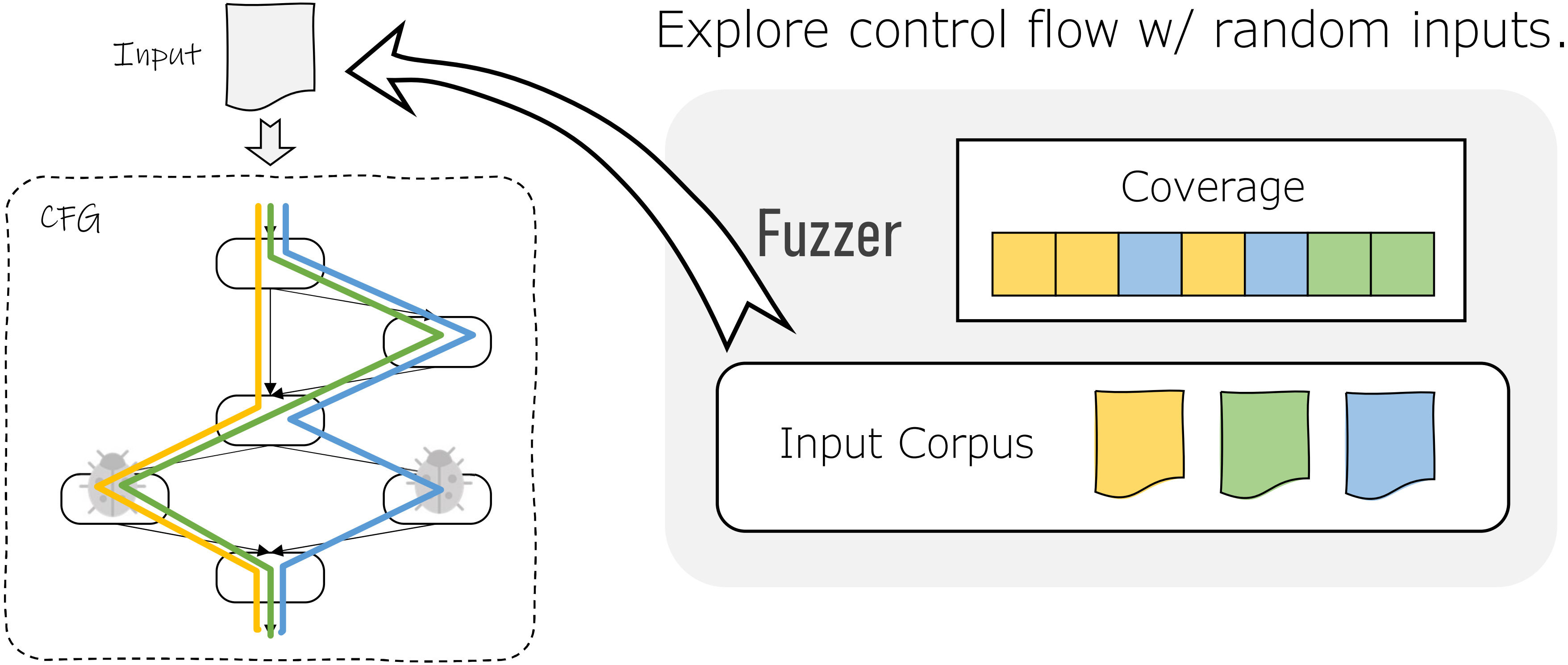


Explore control flow w/ random inputs.

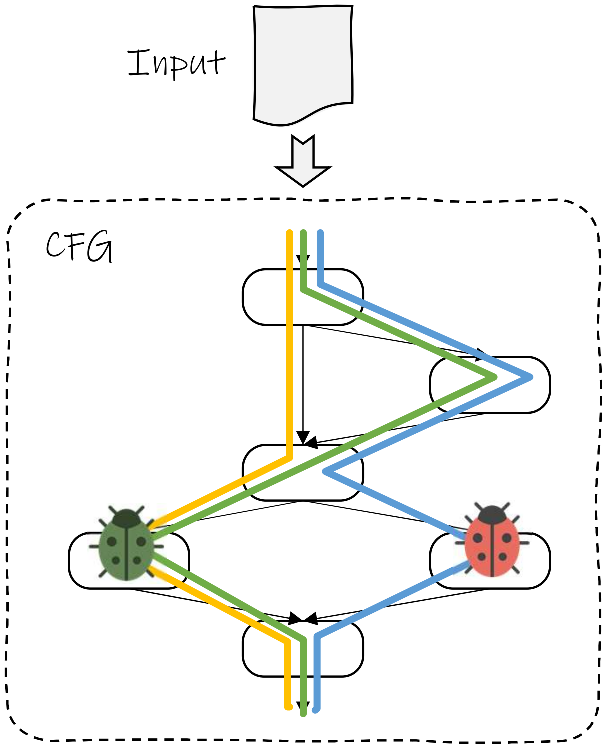


Greybox Fuzzing [AFL], [libFuzzer]

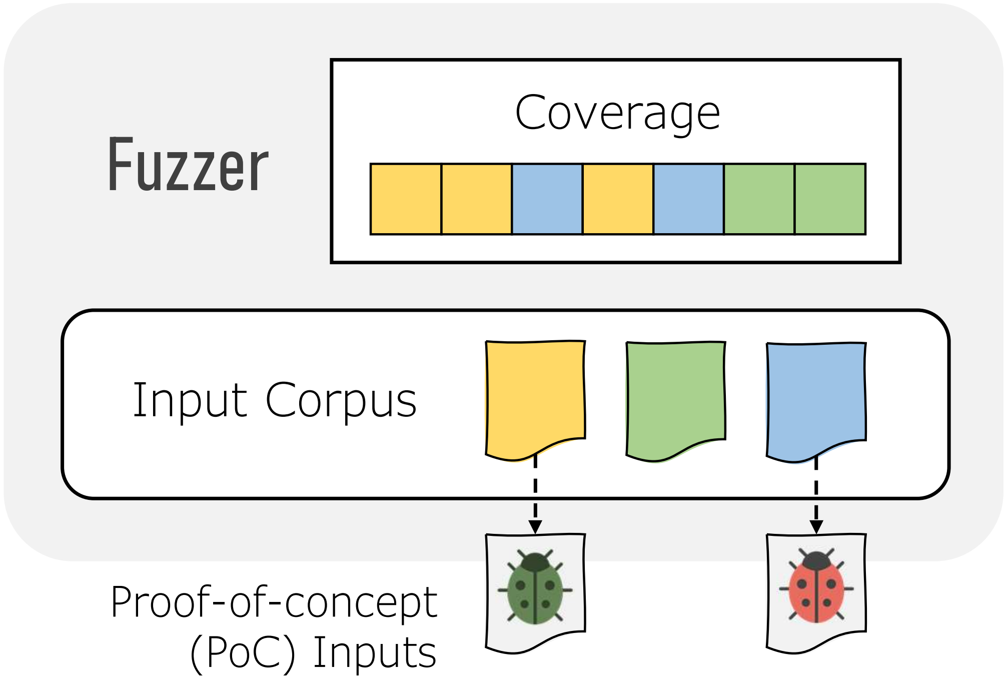
Explore control flow w/ random inputs.



Greybox Fuzzing [AFL], [libFuzzer]



Explore control flow w/ random inputs.



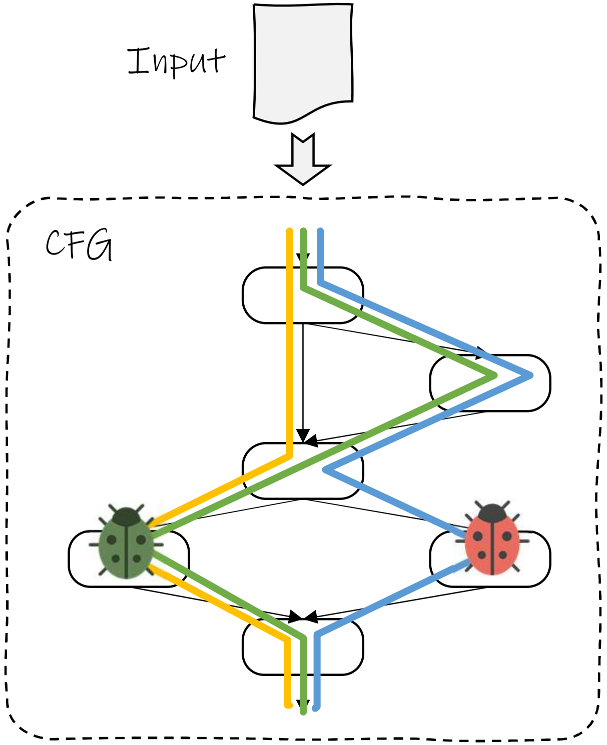
Fuzzer

Coverage

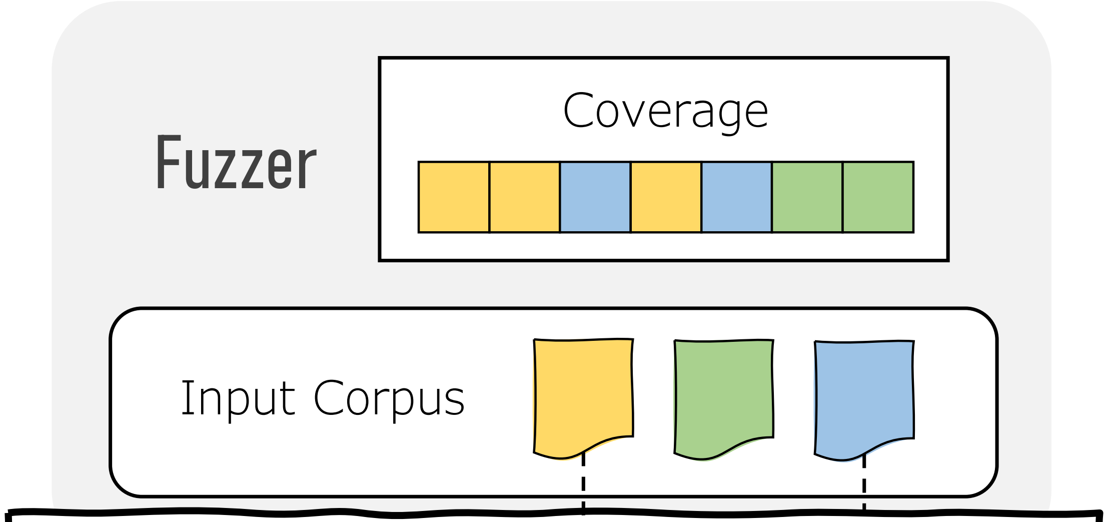
Input Corpus

Proof-of-concept
(PoC) Inputs

Greybox Fuzzing [AFL], [libFuzzer]

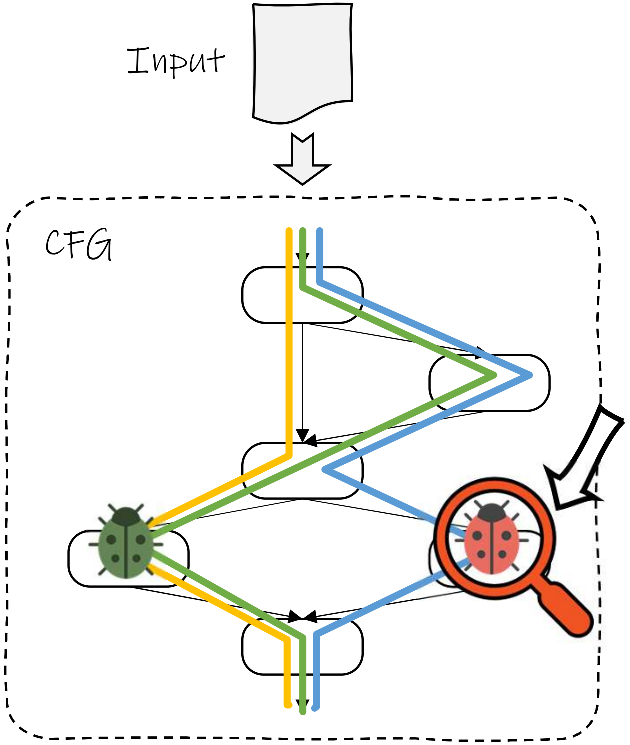


Explore control flow w/ random inputs.

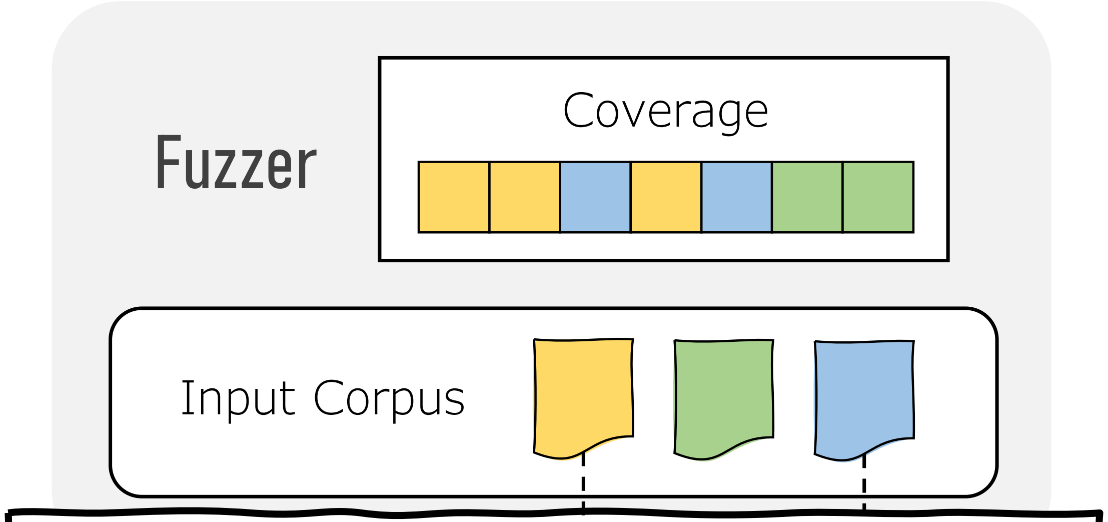


Q: What if there’s a site of interest?

Greybox Fuzzing [AFL], [libFuzzer]



Explore control flow w/ random inputs.



Q: What if there's a site of interest?

Applications of **Targeted** Fuzzing

Crash reproduction
for Debugging

Target: Crash site

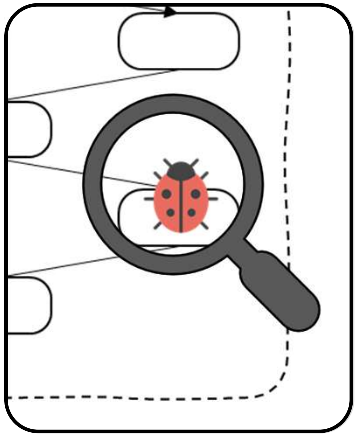
Static Analysis Verification
for False-positive Verification

Target: Reported site

1-day PoC Generation
for Exploitation

Target: Patched site

Applications of **Targeted** Fuzzing



Crash reproduction
for Debugging

Target: Crash site

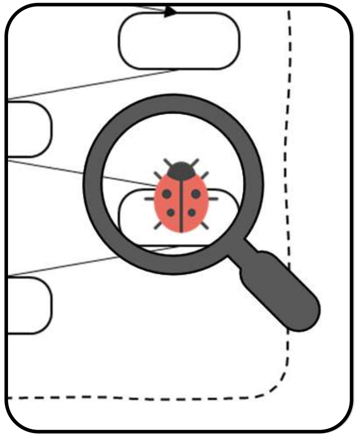
Static Analysis Verification
for False-positive Verification

Target: Reported site

1-day PoC Generation
for Exploitation

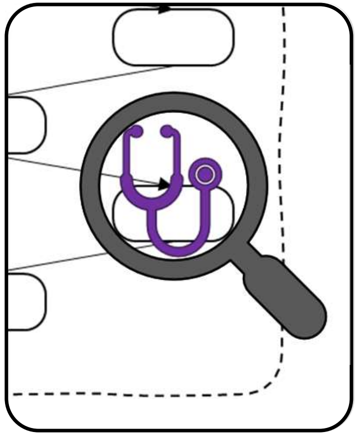
Target: Patched site

Applications of **Targeted** Fuzzing



Crash reproduction
for Debugging

Target: Crash site



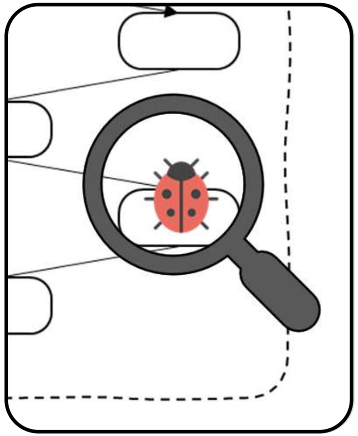
Static Analysis Verification
for False-positive Verification

Target: Reported site

1-day PoC Generation
for Exploitation

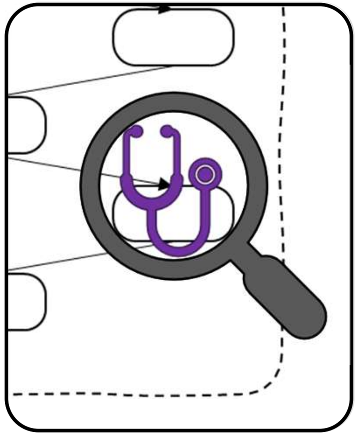
Target: Patched site

Applications of **Targeted** Fuzzing



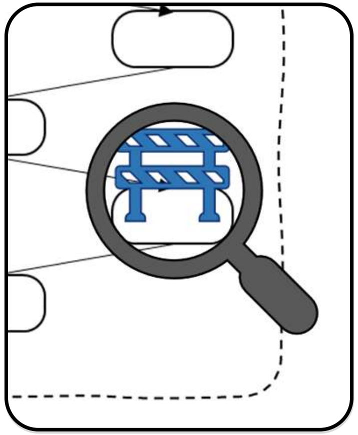
Crash reproduction
for Debugging

Target: Crash site



Static Analysis Verification
for False-positive Verification

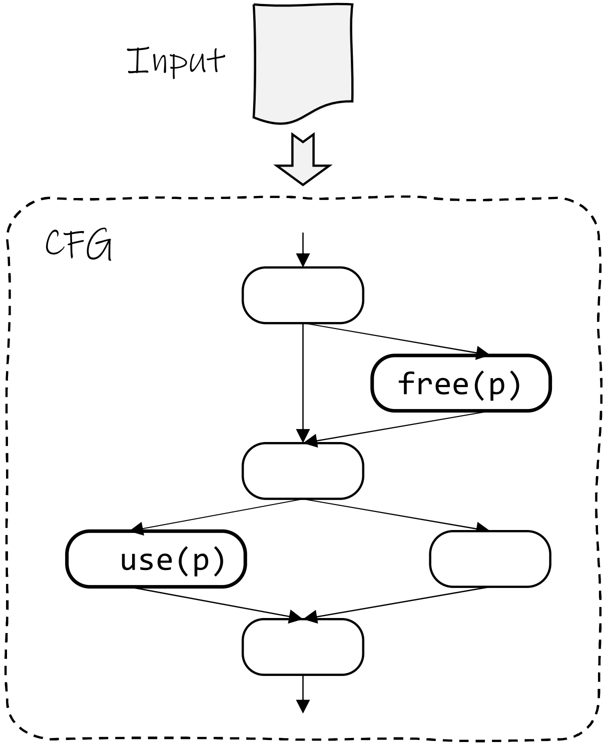
Target: Reported site



1-day PoC Generation
for Exploitation

Target: Patched site

Requirement 1: Prioritizing ordered target sites

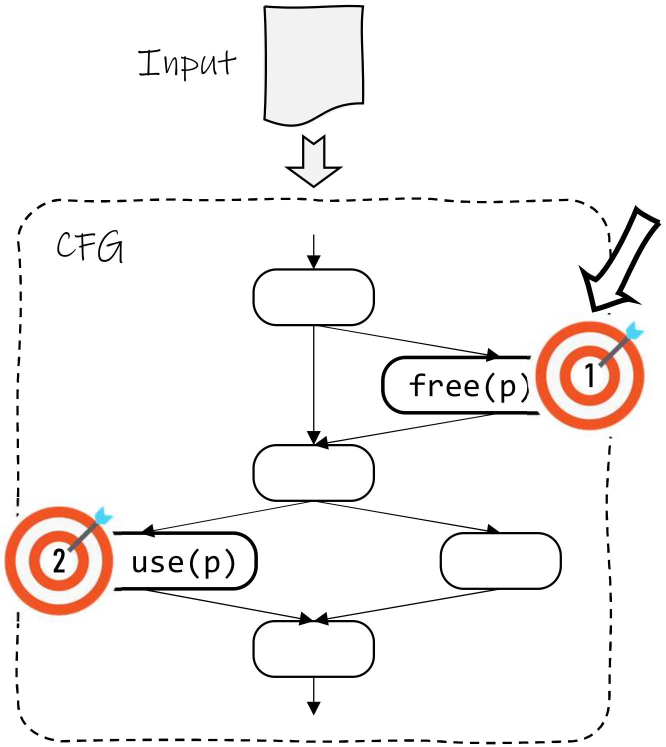


Case: Reproducing use-after-free

Input

free(p)
use(p)

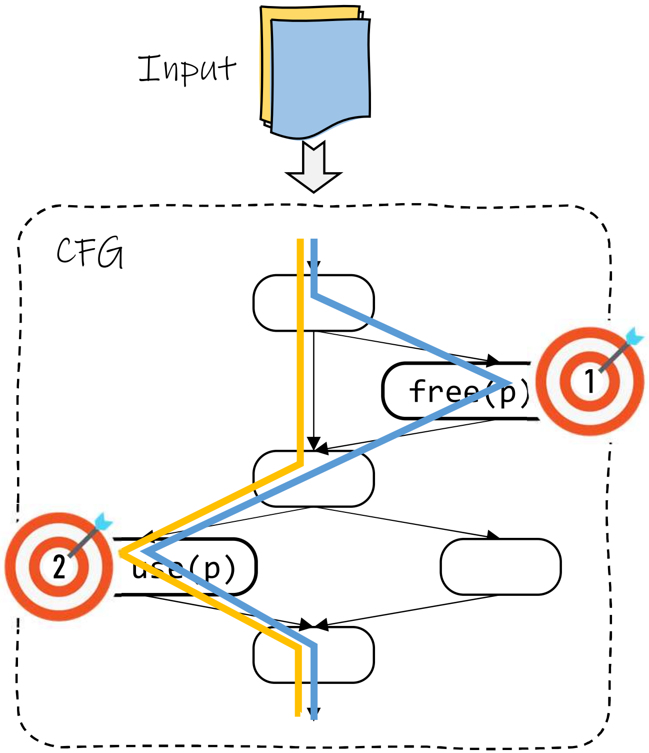
Requirement 1: Prioritizing ordered target sites



Case: Reproducing use-after-free

Input
`free(p)`
`use(p)`

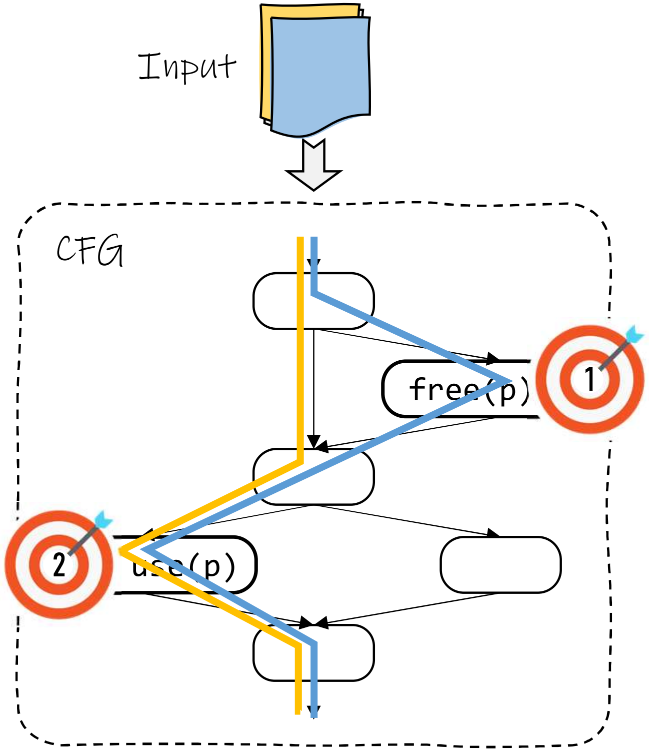
Requirement 1: Prioritizing ordered target sites



Case: Reproducing use-after-free

Input	🟨	🟦
free(p)	✗	✓
use(p)	✓	✓

Requirement 1: Prioritizing ordered target sites

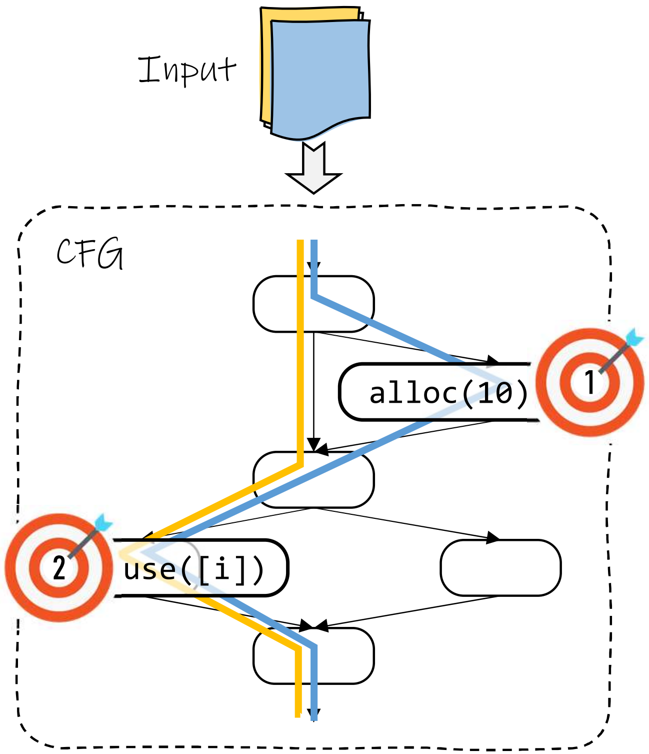


Case: Reproducing use-after-free

Input	🟨	🟦
free(p)	✗	✓
use(p)	✓	✓

Should be prioritize more

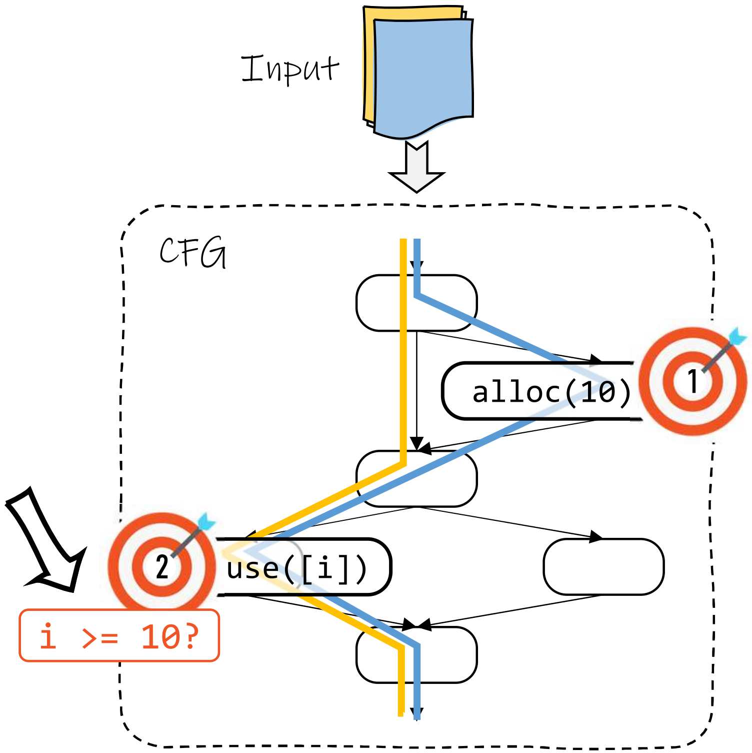
Requirement 2: Prioritizing data conditions



Case: Reproducing heap-buffer-overflow

Input	🟨	🟦
`alloc(10)`	✗	✓
`use([i])`	✓	✓

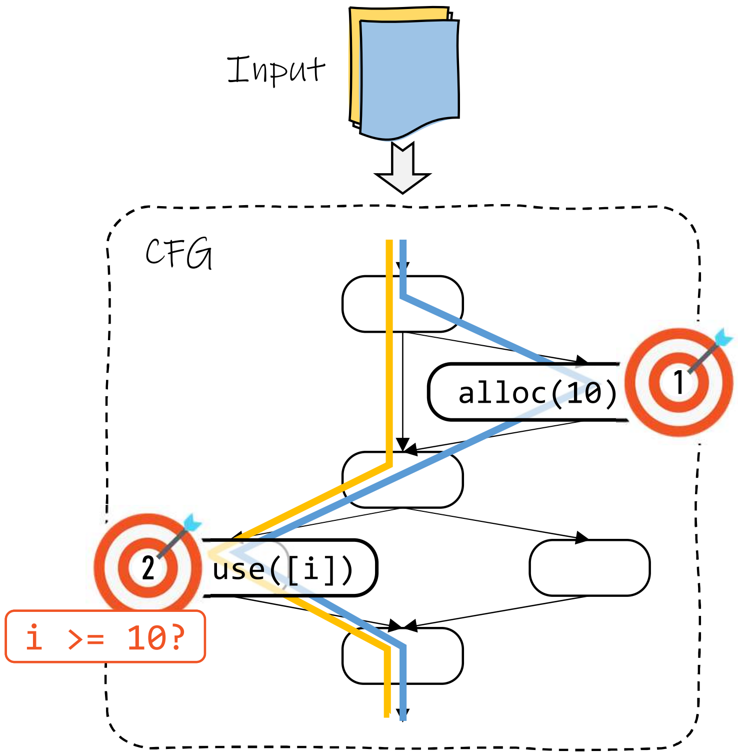
Requirement 2: Prioritizing data conditions



Case: Reproducing heap-buffer-overflow

Input	🟨	🟦
alloc(10)	✗	✓
use([i])	✓	✓
“i >= 10”		

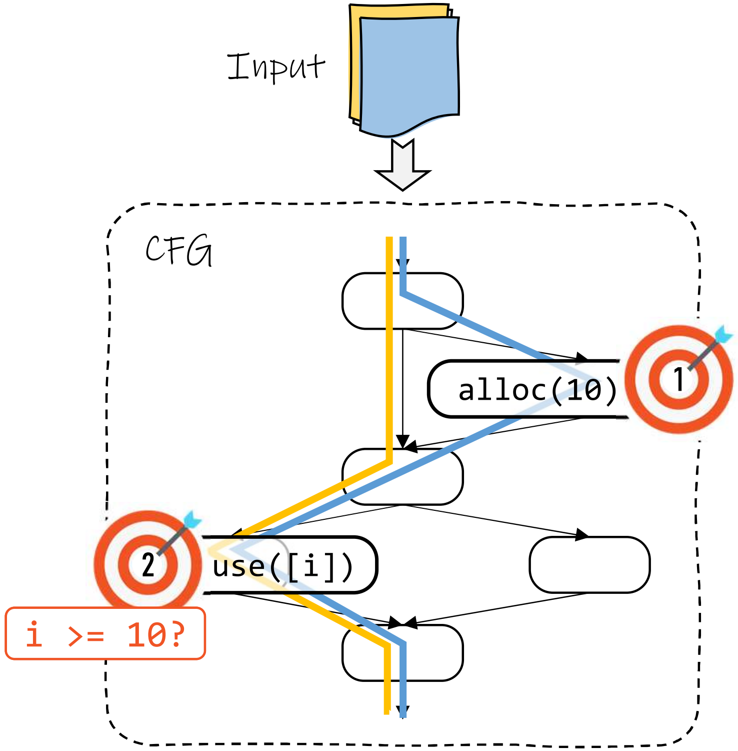
Requirement 2: Prioritizing data conditions



Case: Reproducing heap-buffer-overflow

Input	🟨	🟦 i=5
alloc(10)	✗	✓
use([i])	✓	✓
“i >= 10”		

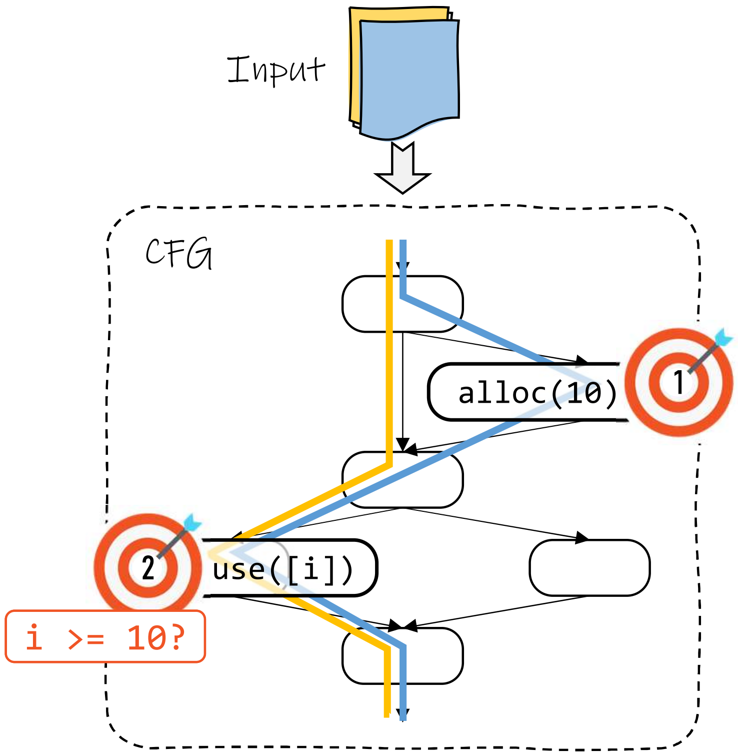
Requirement 2: Prioritizing data conditions



Case: Reproducing heap-buffer-overflow

Input	🟨	🟦 i=5
alloc(10)	✗	✓
use([i])	✓	✓
“i >= 10”	-	✗

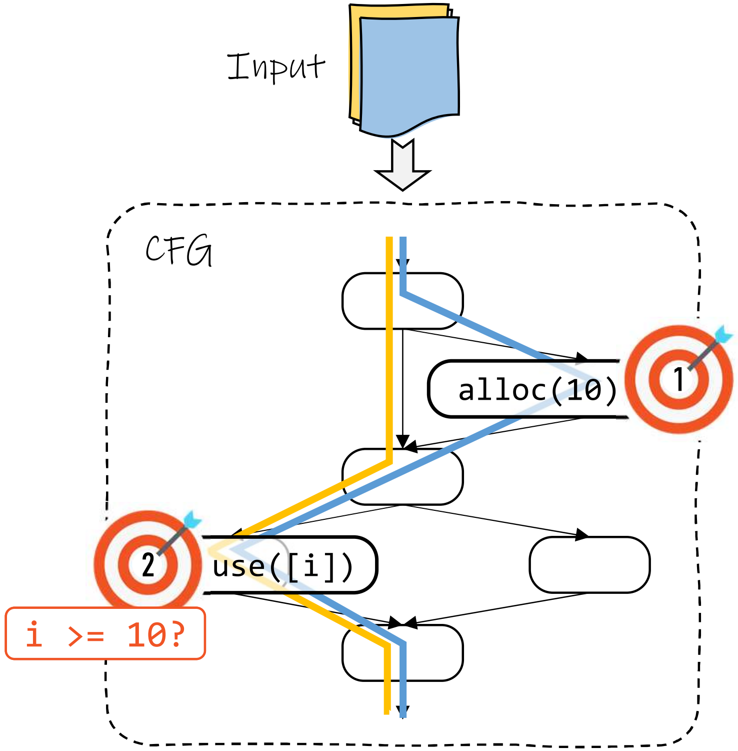
Requirement 2: Prioritizing data conditions



Case: Reproducing heap-buffer-overflow

Input	🟨	i=5	i=10
alloc(10)	✗	✓	
use([i])	✓	✓	
"i >= 10"	-	✗	

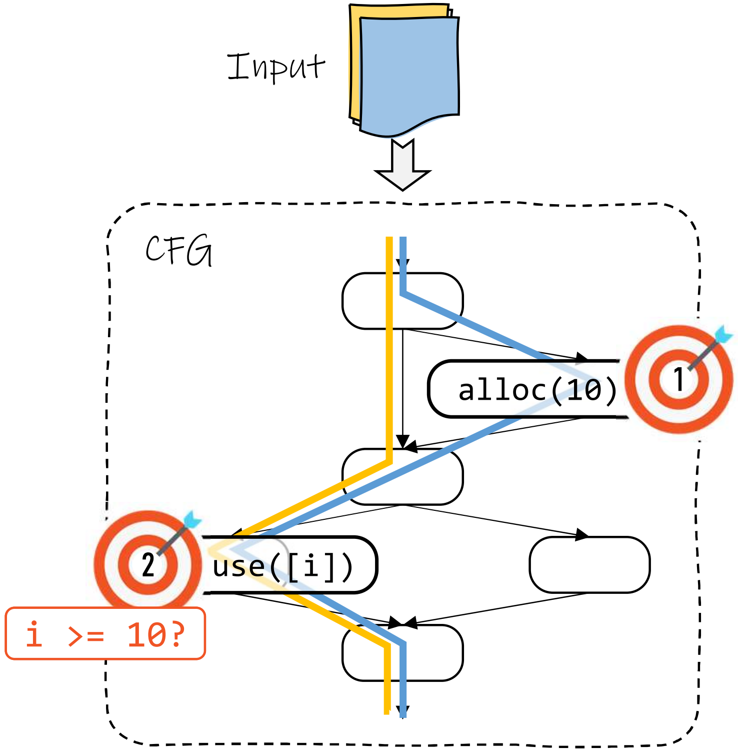
Requirement 2: Prioritizing data conditions



Case: Reproducing heap-buffer-overflow

Input	🟨	i=5	i=10
alloc(10)	✗	✓	✓
use([i])	✓	✓	✓
“i >= 10”	-	✗	✓

Requirement 2: Prioritizing data conditions



Case: Reproducing heap-buffer-overflow

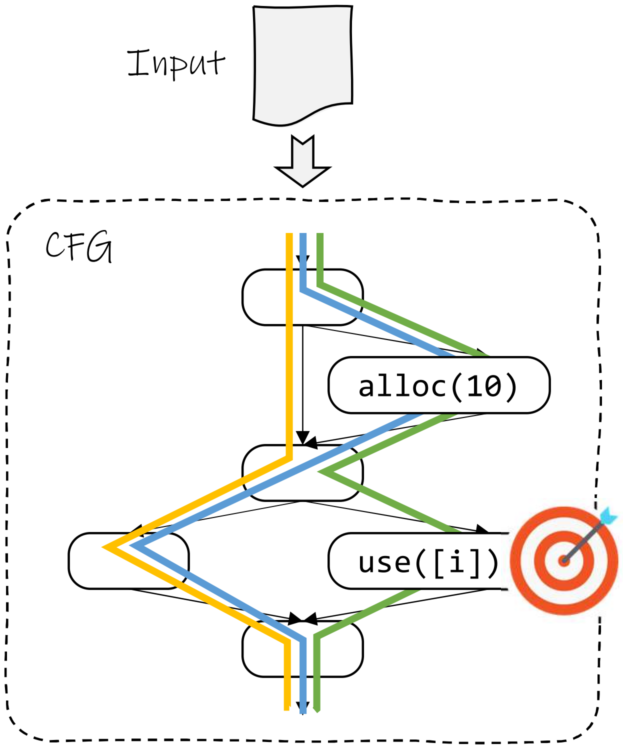
Input	🟨	i=5	i=10
alloc(10)	✗	✓	✓
use([i])	✓	✓	✓
“i >= 10”	-	✗	✓

Should be prioritize more

Constraint-guided
Directed Greybox Fuzzing

(CDGF)

CDGF: Directed Greybox Fuzzing (DGF) as a base. [AFLGo]

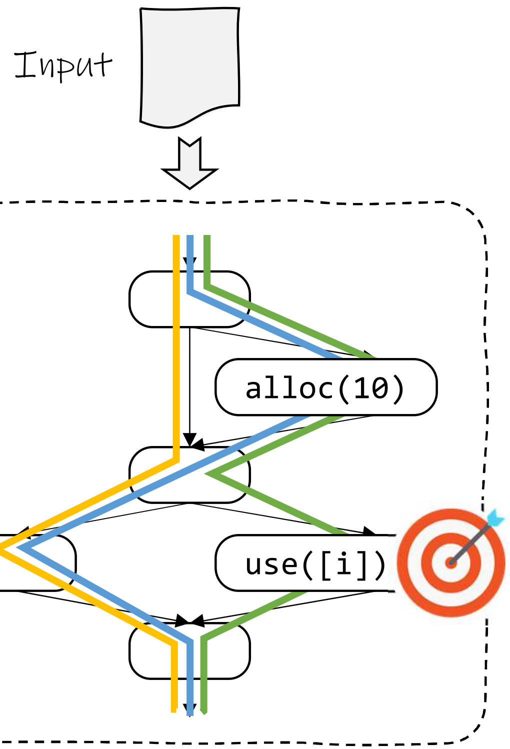


Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
		🎯	
🟨	Dist =	1	Low
🟦	Dist =	1	Low
i=5	Dist =	0	High
i=10	Dist =	0	High

➔ **DGF**: prioritize inputs by their minimum control-flow distance to 🎯.

CDGF: Directed Greybox Fuzzing (DGF) as a base. [AFLGo]

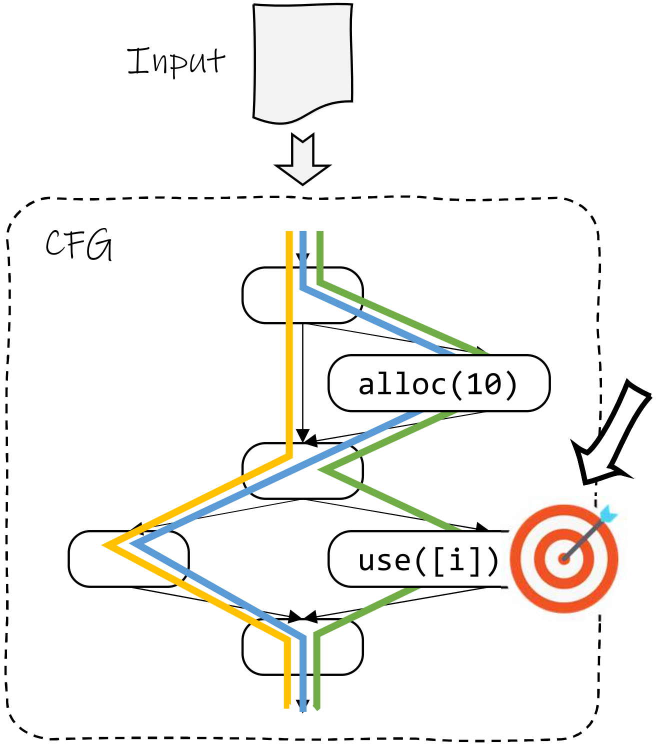


Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
		🎯	
🟨	Dist =	1	Low
🟦	Dist =	1	Low
i=5	Dist =	0	High
i=10	Dist =	0	High

➔ **DGF**: prioritize inputs by their minimum control-flow distance to 🎯.

CDGF: Directed Greybox Fuzzing (DGF) as a base. [AFLGo]

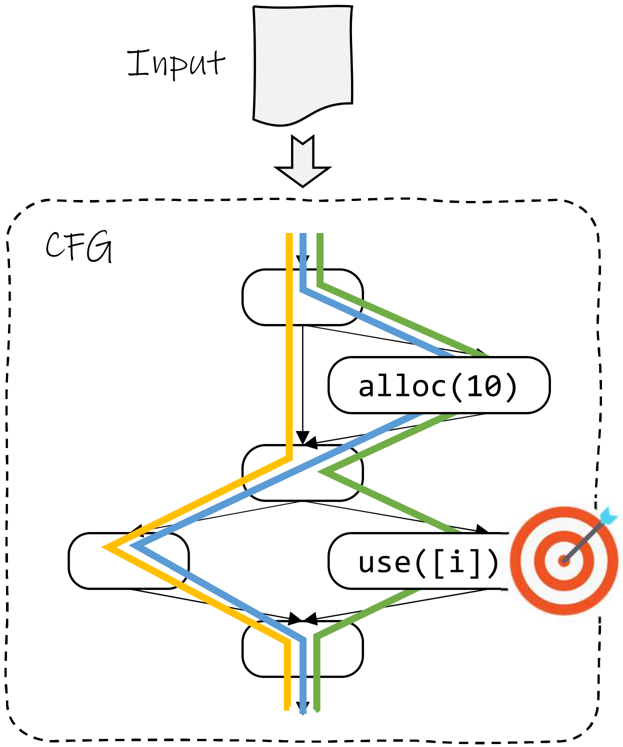


Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
		🎯	
🟨	Dist =	1	Low
🟦	Dist =	1	Low
i=5	Dist =	0	High
i=10	Dist =	0	High

➔ **DGF:** prioritize inputs by their minimum control-flow distance to 🎯.

CDGF: Directed Greybox Fuzzing (DGF) as a base. [AFLGo]

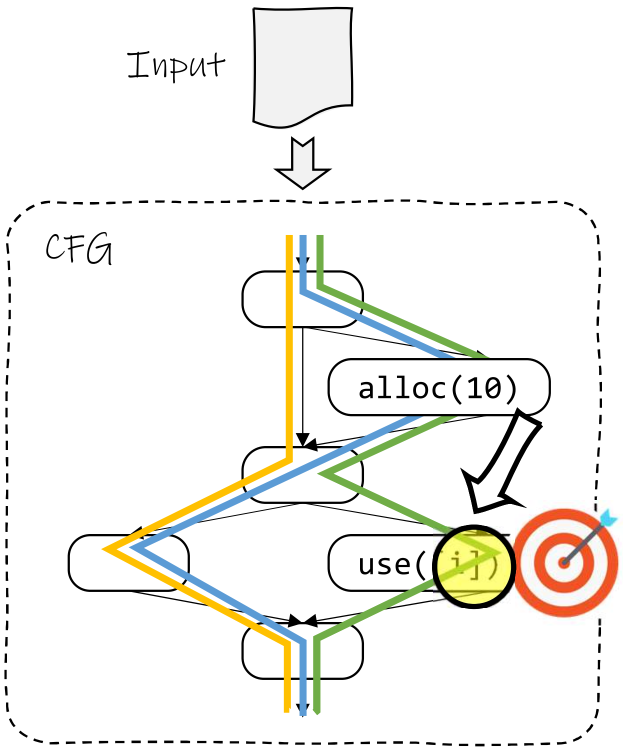


Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
		🎯	
🟨	Dist =	1	Low
🟦	Dist =	1	Low
i=5	Dist =	0	High
i=10	Dist =	0	High

➔ **DGF**: prioritize inputs by their minimum control-flow distance to 🎯.

CDGF: Directed Greybox Fuzzing (DGF) as a base. [AFLGo]

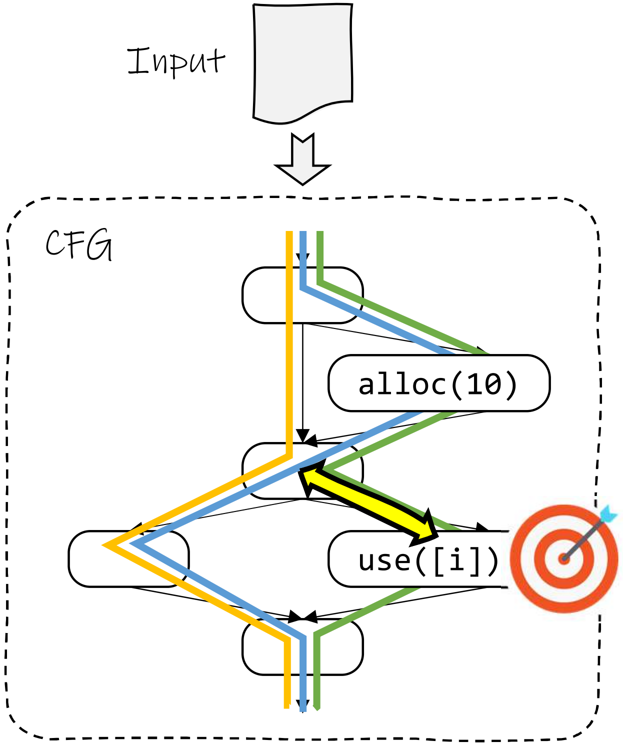


Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
		🎯	
🟨	Dist =	1	Low
🟦	Dist =	1	Low
i=5	Dist =	0	High
i=10	Dist =	0	High

➔ **DGF**: prioritize inputs by their minimum control-flow distance to 🎯.

CDGF: Directed Greybox Fuzzing (DGF) as a base. [AFLGo]

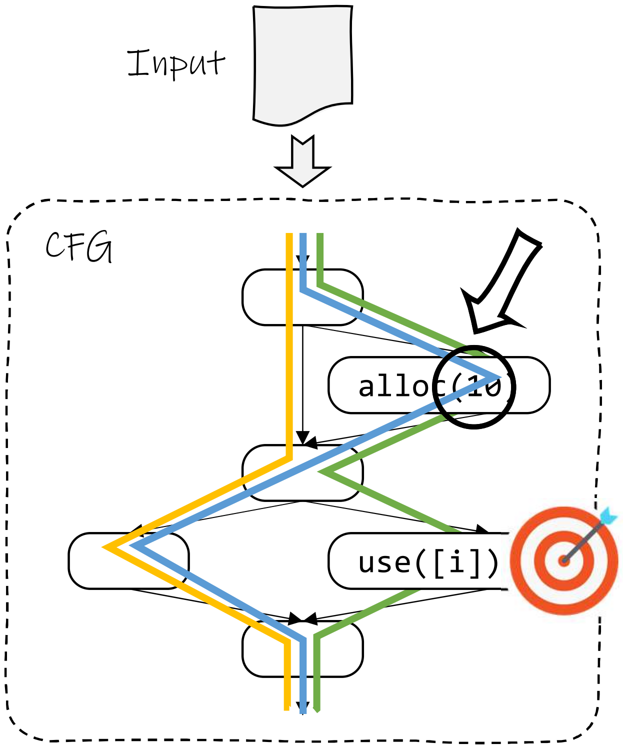


Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
		🎯	
🟨	Dist =	1	Low
🟦	Dist =	1	Low
i=5	Dist =	0	High
i=10	Dist =	0	High

➔ **DGF**: prioritize inputs by their minimum control-flow distance to 🎯.

CDGF: Directed Greybox Fuzzing (DGF) as a base. [AFLGo]

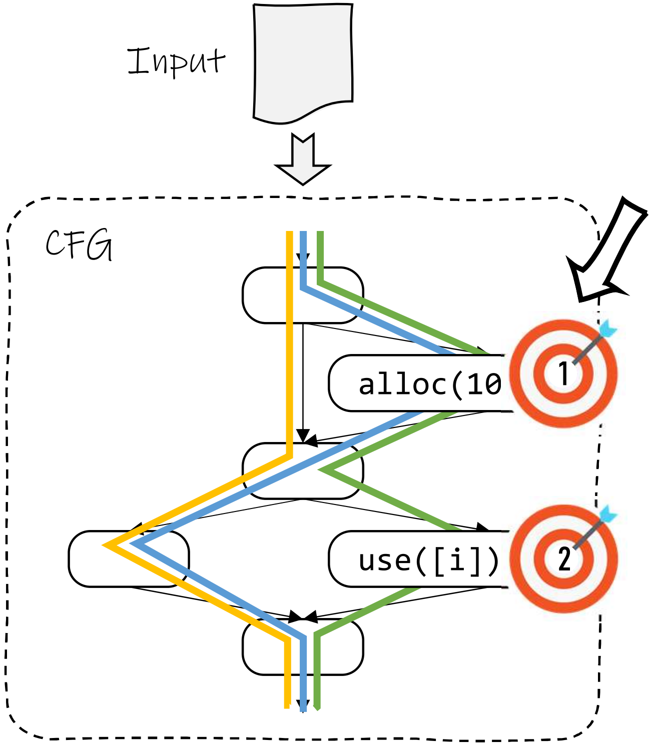


Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
		🎯	
🟨	Dist =	1	Low
🟦	Dist =	1	Low
i=5	Dist =	0	High
i=10	Dist =	0	High

➔ **DGF**: prioritize inputs by their minimum control-flow distance to 🎯.

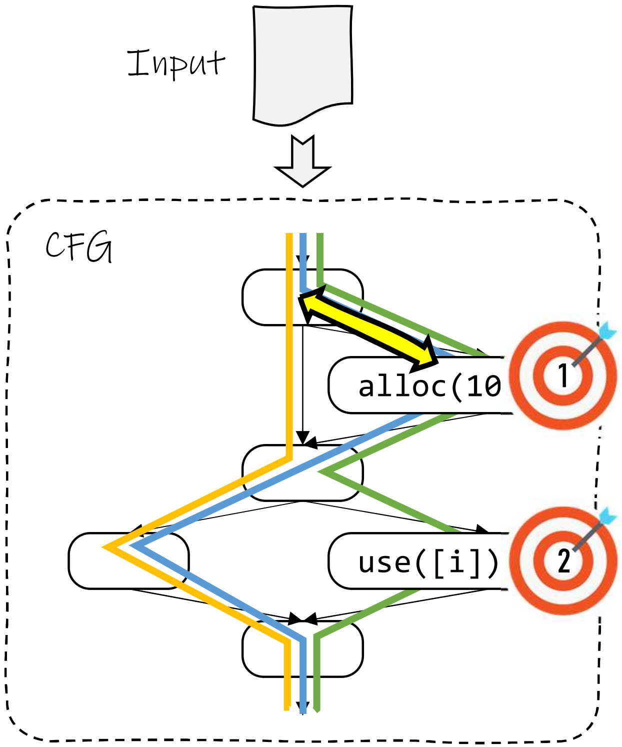
CDGF: Let’s introduce an order.



Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
	1	2	
🟨	Dist = 1,	1	Low
🟦	Dist = 0,	1	Low
i=5	Dist = 0,	0	High
i=10	Dist = 0,	0	High

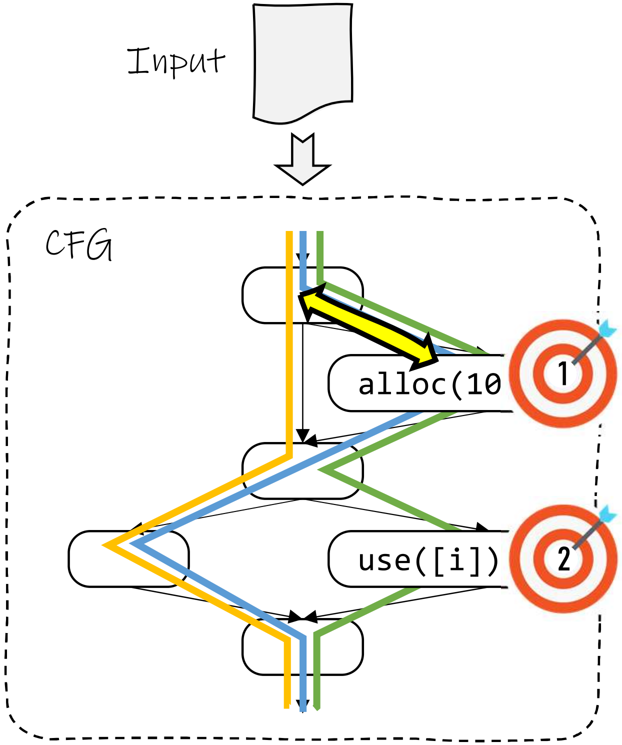
CDGF: Let’s introduce an order.



Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
	1	2	
	Dist = 1,	1	Low
	Dist = 0,	1	Low
i=5	Dist = 0,	0	High
i=10	Dist = 0,	0	High

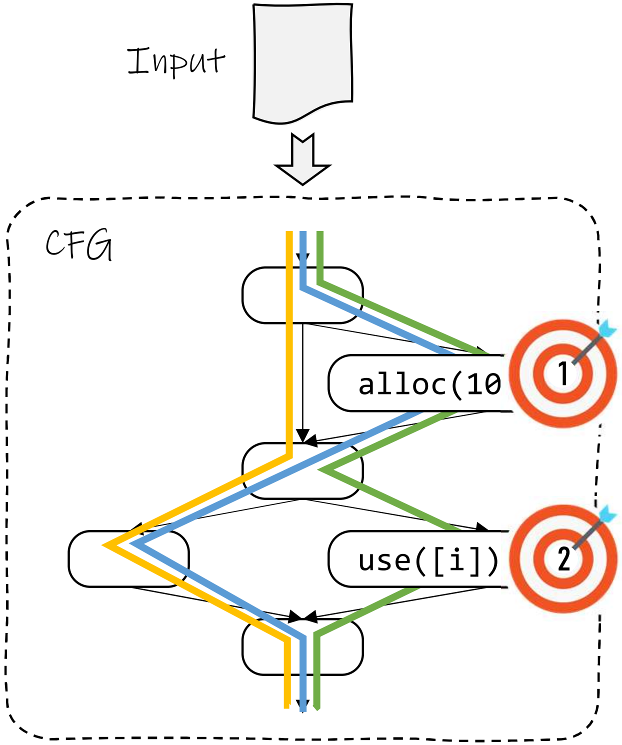
CDGF: Let's introduce an order.



Case: Reproducing heap-buffer-overflow

	Instrumentation	Priority
	1 2	
	Dist = 1, MAX	← Max-out as 1 is not reached yet
	Dist = 0, 1	
i=5	Dist = 0, 0	High
i=10	Dist = 0, 0	High

CDGF: Let's introduce an order.

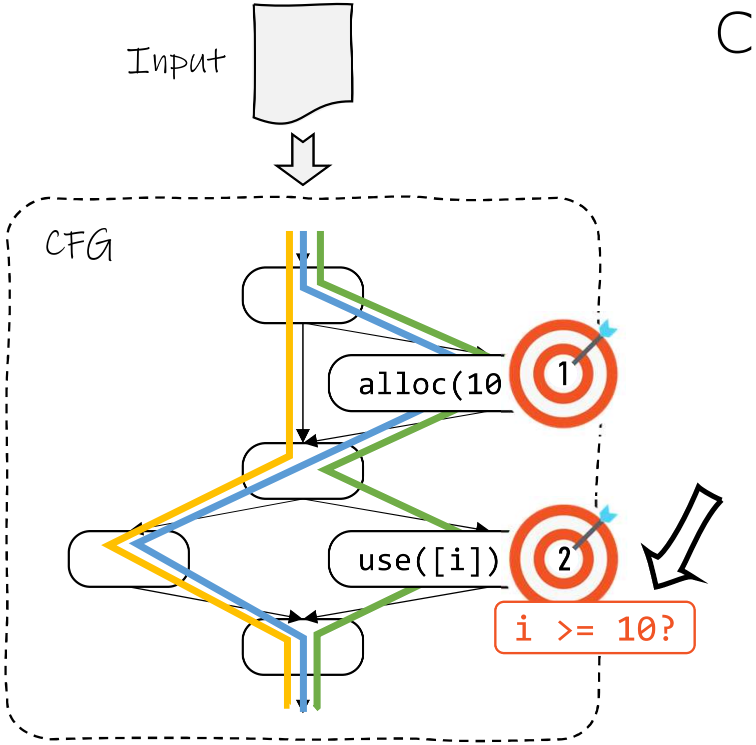


Case: Reproducing heap-buffer-overflow

	Instrumentation		Priority
	1	2	
🟨	Dist = 1,	$\boxed{\text{MAX}}$	Lowest
🟦	Dist = 0,	1	Low
i=5	Dist = 0,	0	High
i=10	Dist = 0,	0	High

➔ 🟦 is relatively prioritized than 🟨.

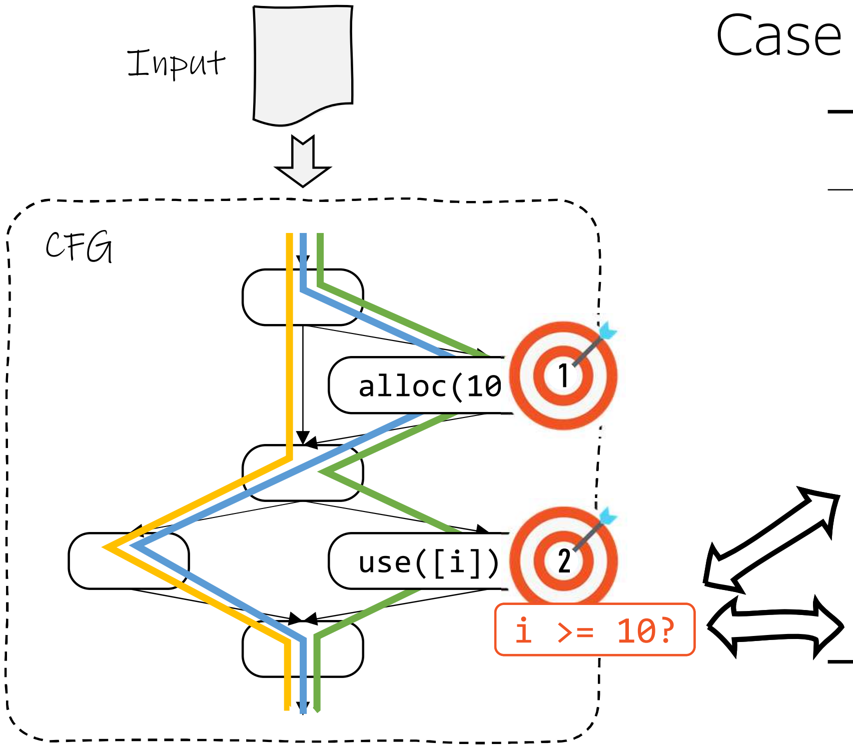
CDGF: …and the distance to data condition.



Case: Reproducing heap-buffer-overflow

	Instrumentation	Priority
	① ②	
🟨	Dist = 1, MAX	Lowest
🟦	Dist = 0, 1	Low
i=5	Dist = 0, 0	High
i=10	Dist = 0, 0	High

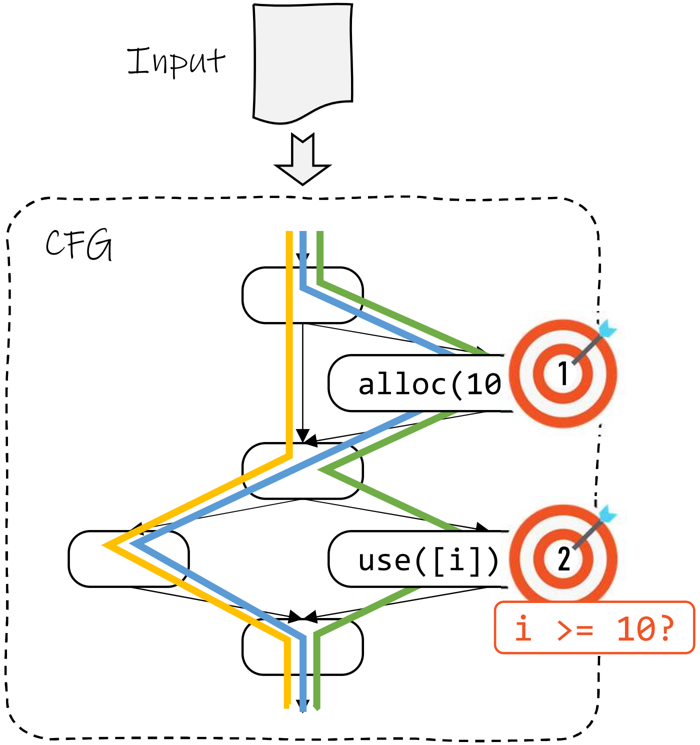
CDGF: …and the distance to data condition.



Case: Reproducing heap-buffer-overflow

	Instrumentation	Priority
	① ②	
	Dist = 1, $\boxed{\text{MAX}}$	Lowest
	Dist = 0, 1	Low
i=5	Dist = 0, 0	High
i=10	Dist = 0, 0	High

CDGF: …and the distance to data condition.

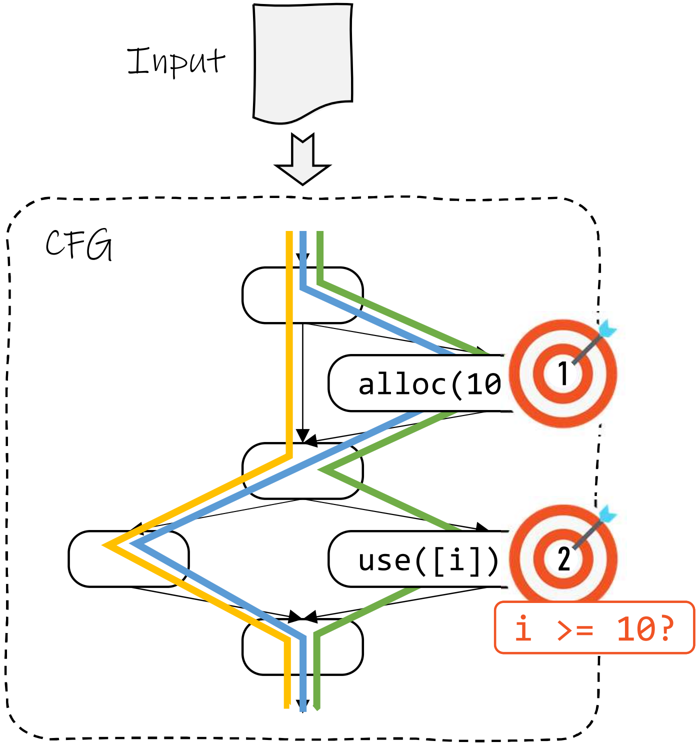


Case: Reproducing heap-buffer-overflow

	Instrumentation				Priority
		1	2	>=	
🟨	Dist =	1,	MAX	MAX	Lowest
🟦	Dist =	0,	1,	MAX	Low
i=5	Dist =	0,	0,	5	
i=10	Dist =	0,	0,	0	

← Integer distance to the solution.

CDGF: …and the distance to data condition.

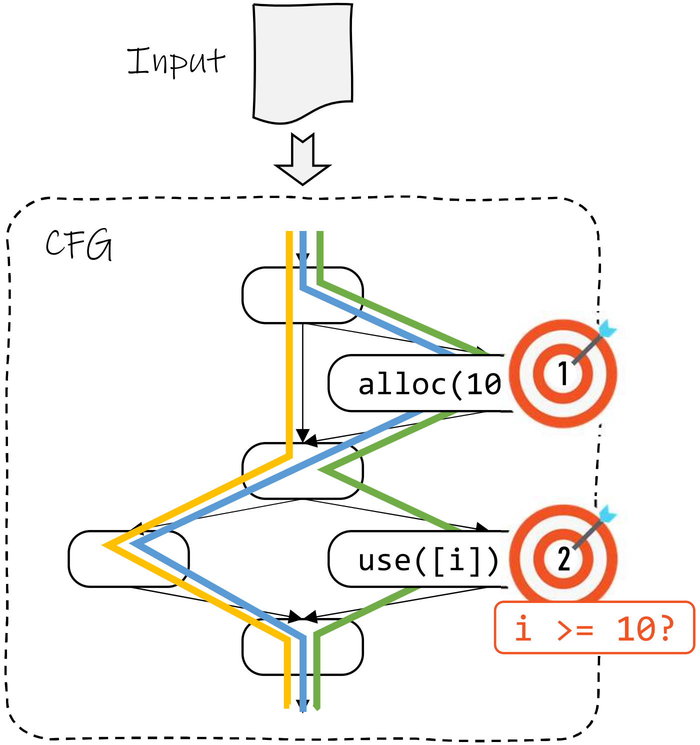


Case: Reproducing heap-buffer-overflow

	Instrumentation			Priority
	①	②	>=	
🟨	Dist = 1,	MAX	MAX	Lowest
🟦	Dist = 0,	1,	MAX	Low
i=5	Dist = 0,	0,	5	High
i=10	Dist = 0,	0,	0	Highest

➔ i=10 is prioritized the most.

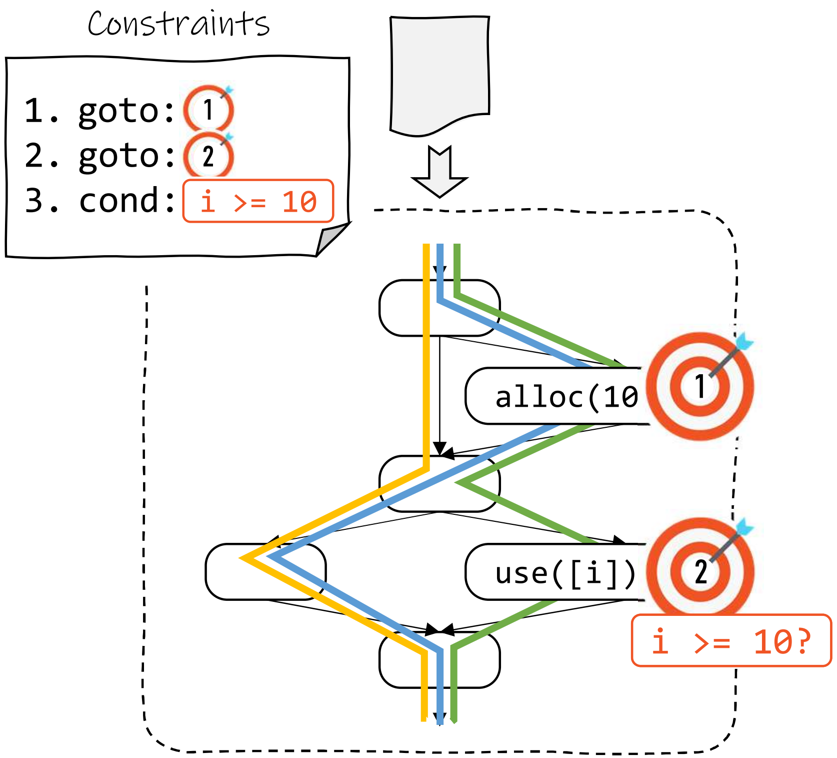
CDGF: Constraint Distance as a Generalized Metric.



Case: Reproducing heap-buffer-overflow

	Instrumentation				Priority
		①	②	>=	
🟨	Dist =	1,	MAX	MAX	Lowest
🟦	Dist =	0,	1,	MAX	Low
i=5	Dist =	0,	0,	5	High
i=10	Dist =	0,	0,	0	Highest

CDGF: Constraint Distance as a Generalized Metric.



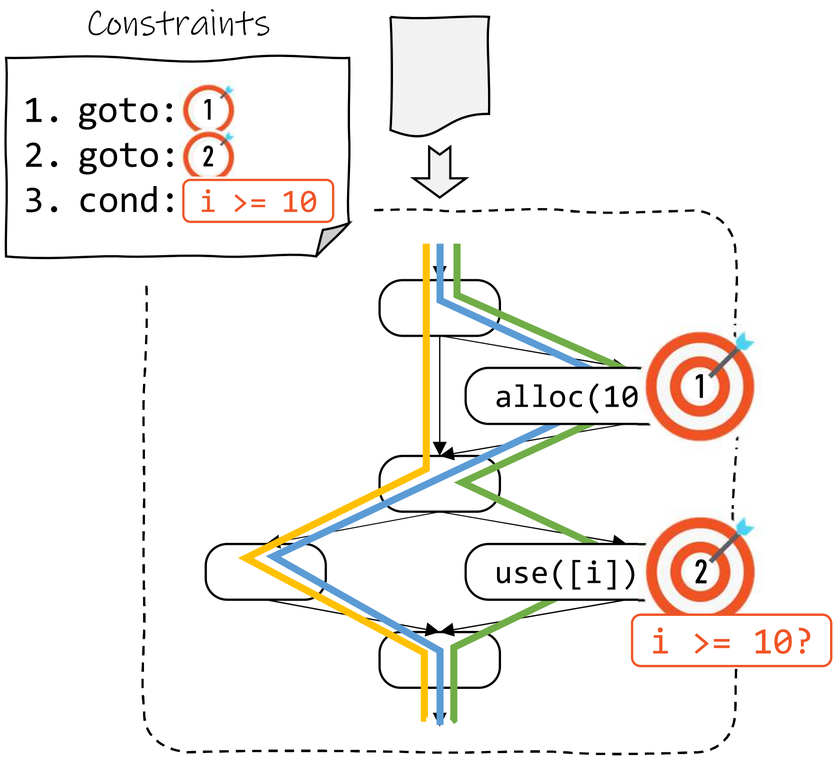
Constraints

1. goto: ①
2. goto: ②
3. cond: i >= 10

Case: Reproducing heap-buffer-overflow

	Instrumentation			Priority
	①	②	>=	
🟨	Dist = 1,	MAX	MAX	Lowest
🟦	Dist = 0,	1,	MAX	Low
i=5	Dist = 0,	0,	5	High
i=10	Dist = 0,	0,	0	Highest

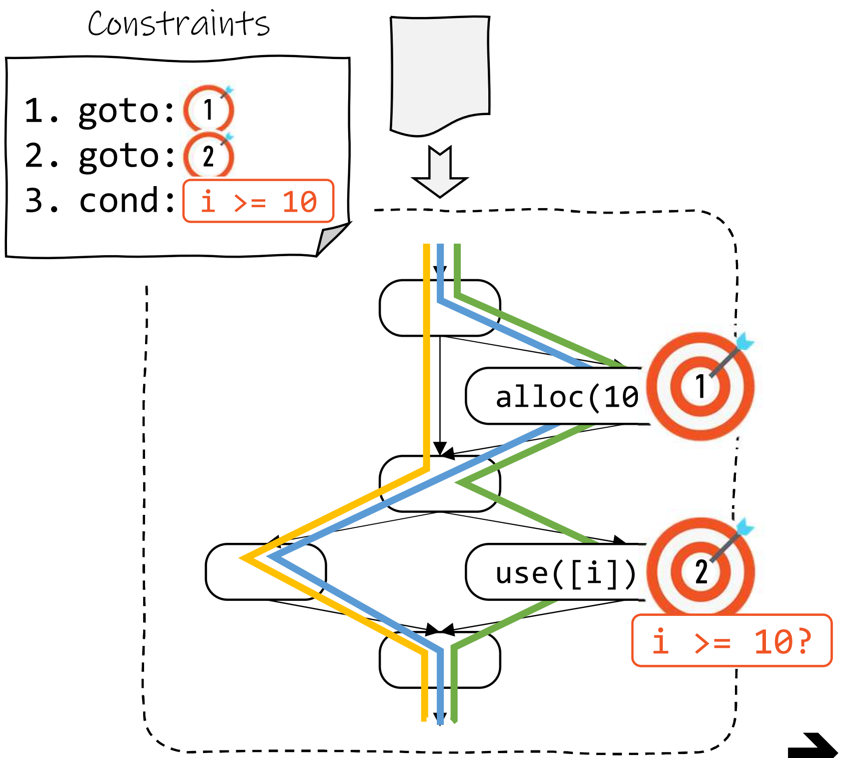
CDGF: Constraint Distance as a Generalized Metric.



Case: Reproducing heap-buffer-overflow

Instrumentation					
		①	②	>=	Constraints
🟨	Dist =	1	⊕MAX	⊕MAX	= 2Max+1
🟦	Dist =	0 ⊕	1	⊕MAX	= Max+1
i=5	Dist =	0 ⊕	0	⊕ 5	= 5
i=10	Dist =	0 ⊕	0	⊕ 0	= 0

CDGF: Constraint Distance as a Generalized Metric.

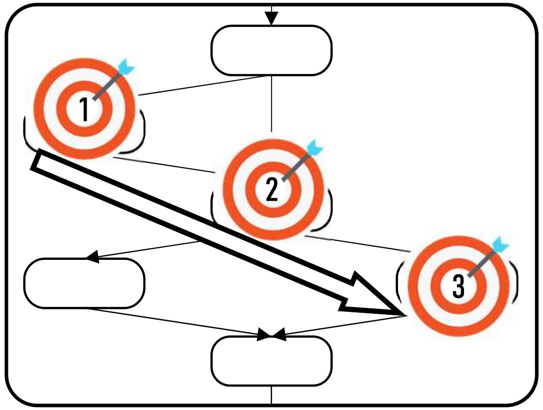


Case: Reproducing heap-buffer-overflow

Instrumentation					
		①	②	>=	Constraints
🟨	Dist =	1 ⊕	MAX ⊕	MAX =	2Max+1
🟦	Dist =	0 ⊕	1 ⊕	MAX =	Max+1
i=5	Dist =	0 ⊕	0 ⊕	5 =	5
i=10	Dist =	0 ⊕	0 ⊕	0 =	0

➔ Prioritization with a single distance metric.

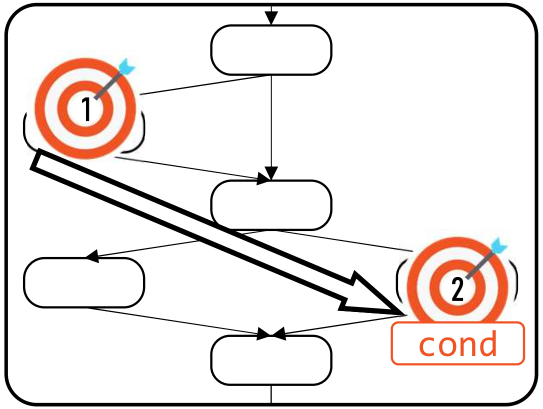
Template-based Constraint Generation



Multiple target sites

Use Cases

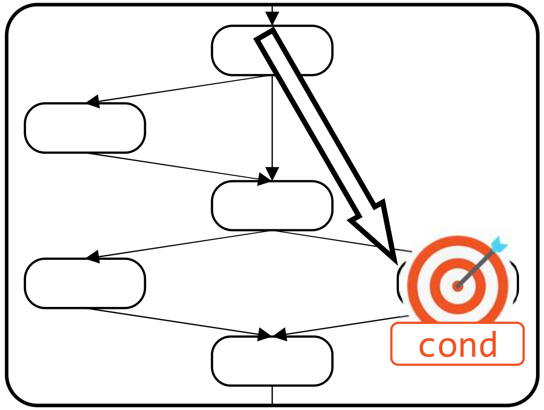
use-after-free (ASAN dump)
double-free (ASAN dump)
use-of-uninit-value (MSAN dump)



Two target sites
+ Data condition

Use Cases

heap-buffer-overf. (ASAN dump)
stack-buffer-overf. (ASAN dump)
Static anlys. verification (report)

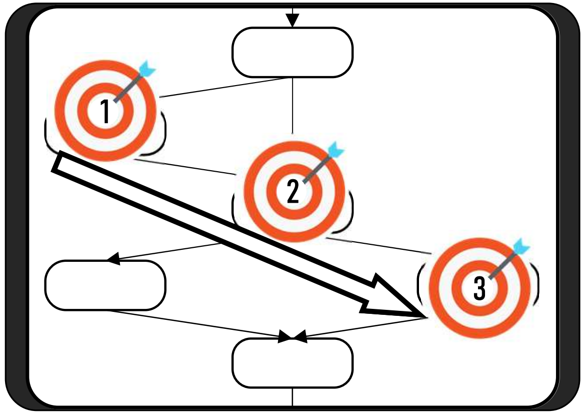


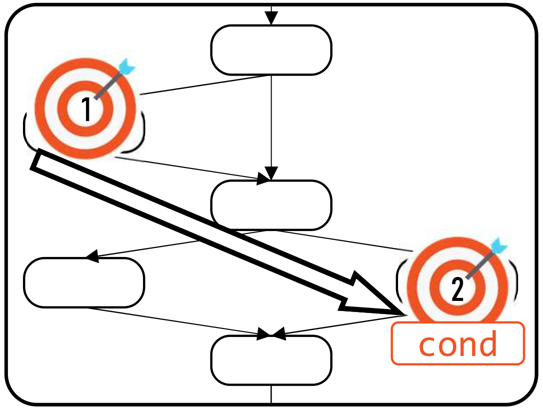
One target site
+ Data condition

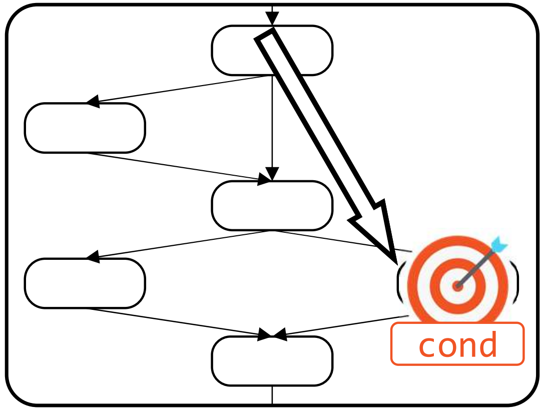
Use Cases

divide-by-zero (UBSAN dump)
assertion-failure (Debug dump)
1-day PoC generation (Fix commit)

Template-based Constraint Generation







Multiple target sites

Use Cases

use-after-free (ASAN dump)
double-free (ASAN dump)
use-of-uninit-value (MSAN dump)

Two target sites
+ Data condition

Use Cases

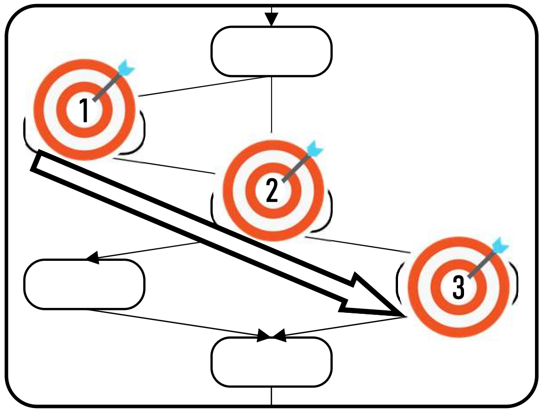
heap-buffer-overf. (ASAN dump)
stack-buffer-overf. (ASAN dump)
Static anlys. verification (report)

One target site
+ Data condition

Use Cases

divide-by-zero (UBSAN dump)
assertion-failure (Debug dump)
1-day PoC generation (Fix commit)

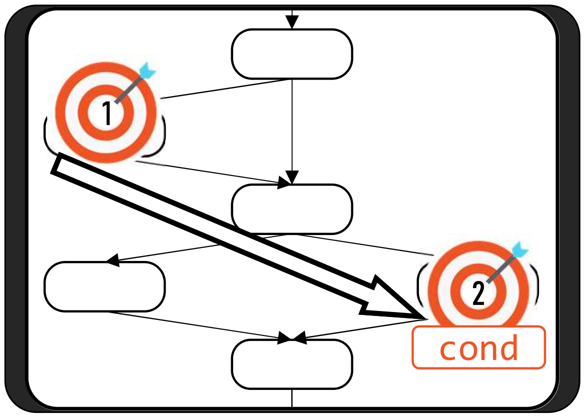
Template-based Constraint Generation



Multiple target sites

Use Cases

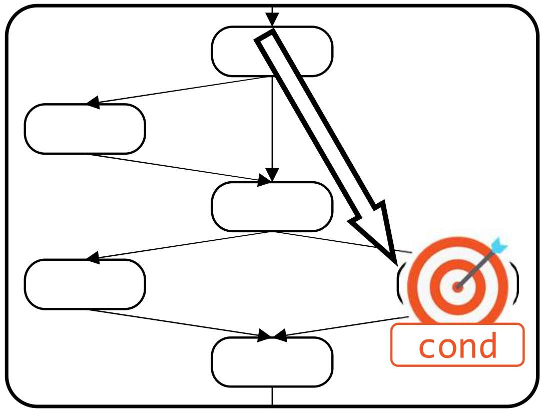
use-after-free (ASAN dump)
double-free (ASAN dump)
use-of-uninit-value (MSAN dump)



Two target sites
+ Data condition

Use Cases

heap-buffer-overf. (ASAN dump)
stack-buffer-overf. (ASAN dump)
Static anlys. verification (report)

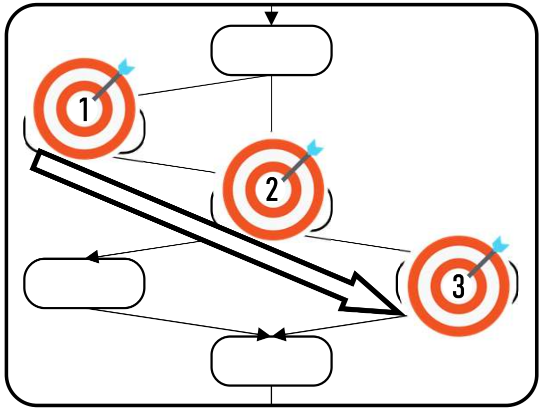


One target site
+ Data condition

Use Cases

divide-by-zero (UBSAN dump)
assertion-failure (Debug dump)
1-day PoC generation (Fix commit)

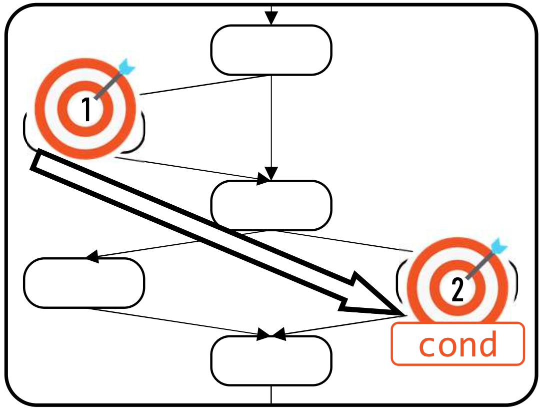
Template-based Constraint Generation



Multiple target sites

Use Cases

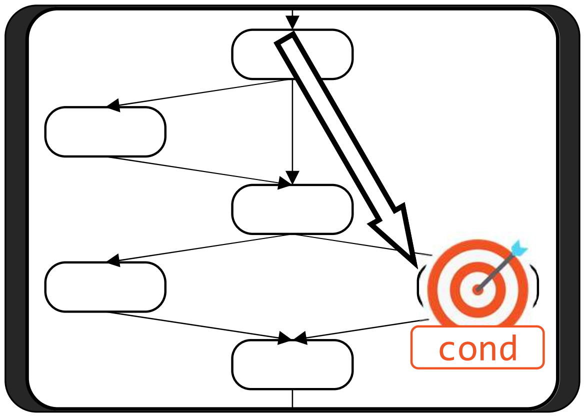
use-after-free (ASAN dump)
double-free (ASAN dump)
use-of-uninit-value (MSAN dump)



Two target sites
+ Data condition

Use Cases

heap-buffer-overf. (ASAN dump)
stack-buffer-overf. (ASAN dump)
Static anlys. verification (report)



One target site
+ Data condition

Use Cases

divide-by-zero (UBSAN dump)
assertion-failure (Debug dump)
1-day PoC generation (Fix commit)

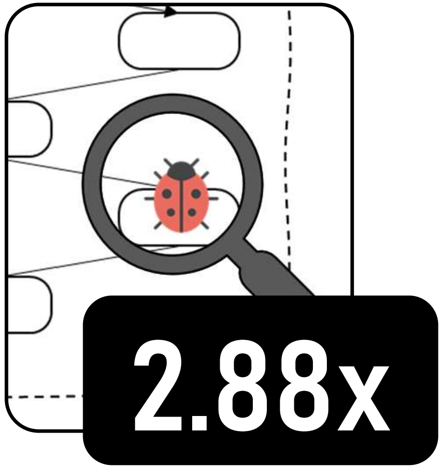
Implementation & Evaluation

Implementation

- Based on AFL 2.52b.
- Custom LLVM pass for distance instrumentation.

Evaluation

- CPU: 20-core Intel Xeon Gold 6209U @ 2.10GHz
- Memory: DDR4 502 GB



2.88x

Crash reproduction

with 47 real-world crashes

3.65x

1-day PoC Generation

with 12 real-world commits

Baseline: DGF (AFLGo)

Conclusion

- DGF lacks some of key mechanisms for targeted fuzzing.
    - Ordered target sites
    - Data conditions

- CDGF augments DGF with a new distance metric.
    - Ordered DGF-style distance + Angora-style data distance.

- The prototype implementation of CDGF outperforms DGF.
    - 2.88x speedup in crash reproduction.
    - 3.65x speedup in 1-day PoC generation.

Thank you for listening

Gwangmu Lee

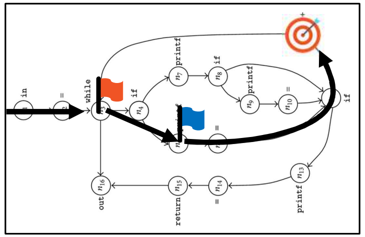
SEOUL NATIONAL UNIVERSITY
Currently looking for a **postdoc** position.

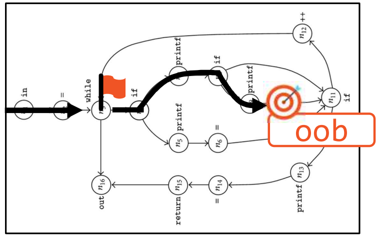
✉ gwangmu@snu.ac.kr
🏠 https://gwangmu.github.io

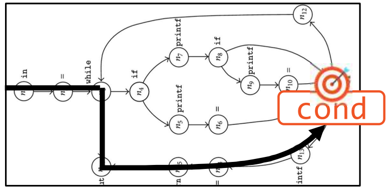
Backup

Template-based Constraint Generation

	Template	CGF	Applications

nT	`constraint %cause:`
	`  site <🚩>`
	`constraint %trans:`
	`  site <⚑>`
	`constraint %crash:`
	`  site <🎯>`
			ASAN: use-after-free
			double-free
			MSAN: use-of-uninit-value

2T+D	`constraint %alloc:`
	`  site <🚩>`
	`constraint %access:`
	`  site <🎯>`
	`  cond out-of-bound`
			ASAN: heap-buffer-overflow
			stack-buffer-overflow
			Static analysis verification

1T+D	`constraint %constr:`
	`  site <🎯>`
	`  cond data-cond`
			ASAN: assertion-failure
			divide-by-zero
			1-day PoC generation

Discussion

Some crash types are incompatible to current data distance.

- Global buffer overflow
  - Mostly used as a look-aside table.
  - Near-boundary access ≠ Near-overflow condition.

- Use-after-free
  - Data condition: “Given `free(p)` and `use(q)`, `p == q`”
  - Integer difference between pointers doesn’t make sense.