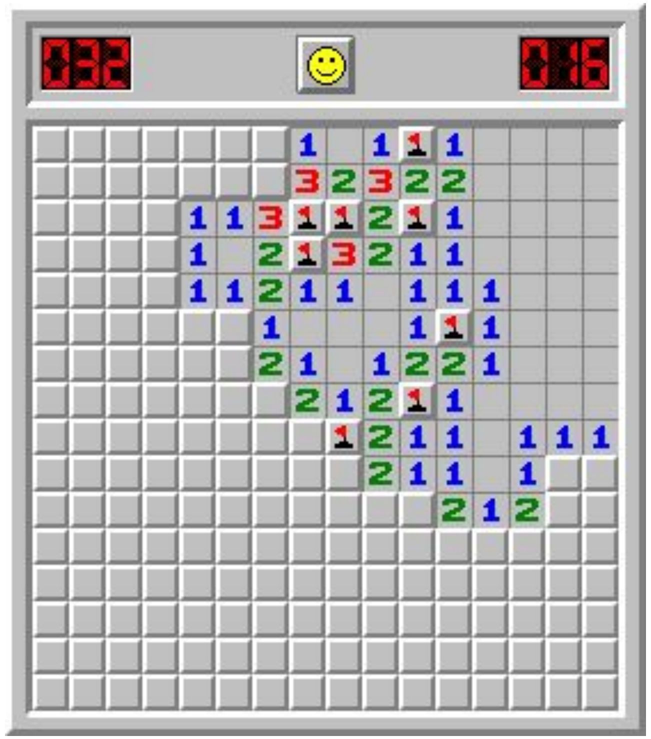
Evocatio: Conjuring Bug Capabilities from a Single PoC

Zhiyuan Jiang, Shuitao Gan, Adrian Herrera, Flavio Toffalini, Lucio Romerio, Chaojing Tang, Manuel Egele, Chao Zhang, Mathias Payer

Motivation

Fuzzing finds 1000s of crashes

- How severe are the crashes?
- Which bug should be fixed first?



So far, the user has to inspect each crash manually

Severity Assessment

- Scoring bug severity is subjective
- Highly dependent on threat model

We want:

- Determine bug severity across multiple dimensions
- Calculate severity based on user-defined threat model
- Fully automatic and objective

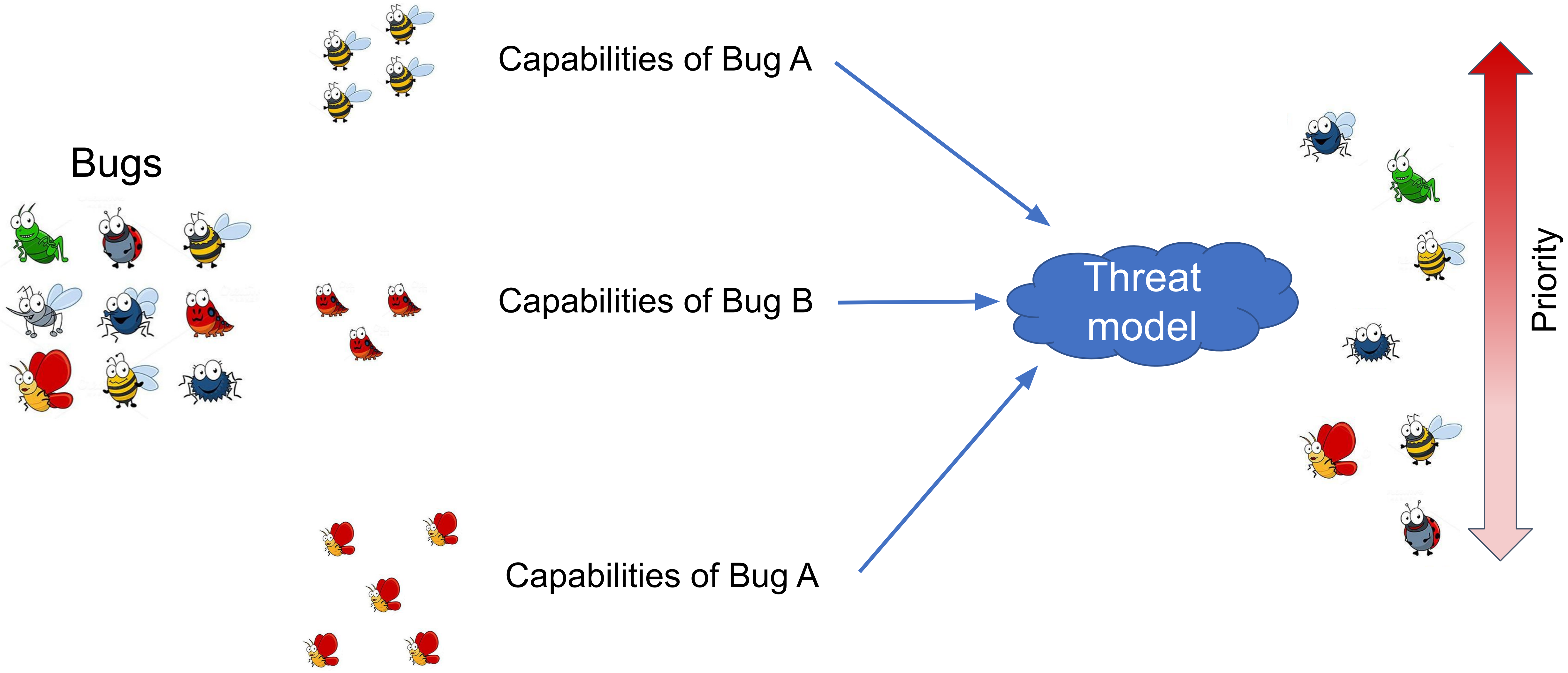
Bug Capability

Before assessing the bug severity

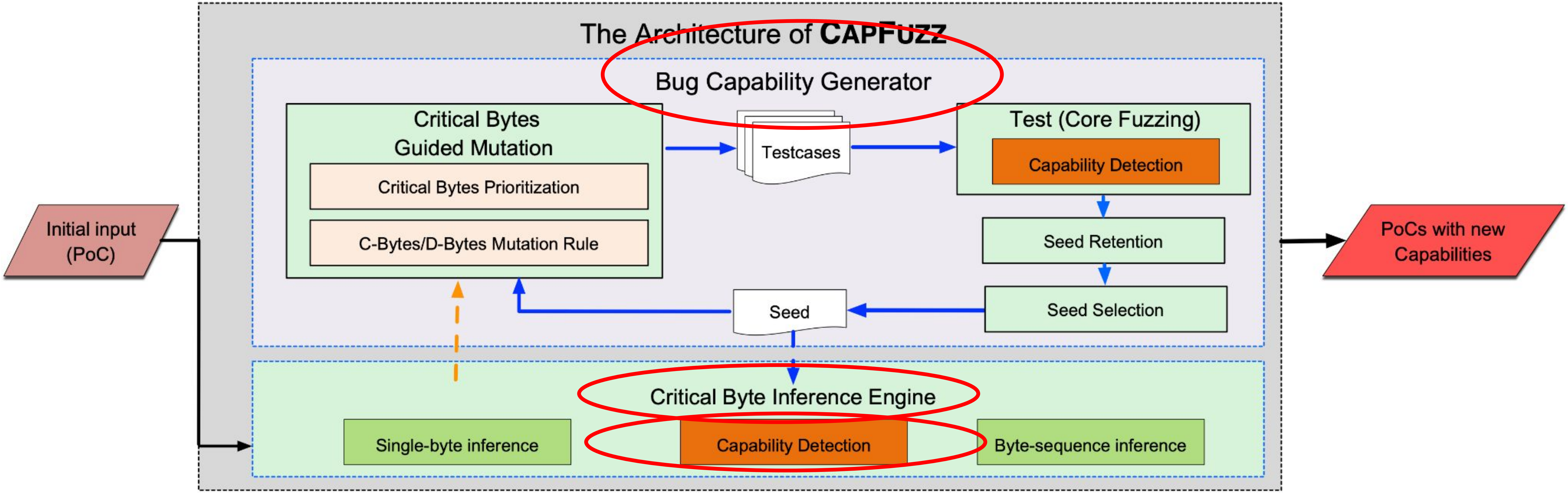
- What can the bug do?

$(\mathrm{Bug}_{type}, \mathrm{Acc}_{type}, \mathrm{Acc}_{len}, \mathrm{Buf}_{name}, \mathrm{Off}, \mathrm{Loc})$

- on the stack
- starts at 10th bytes into buffer
- buffer
- 5 bytes
- read
- out of bounds

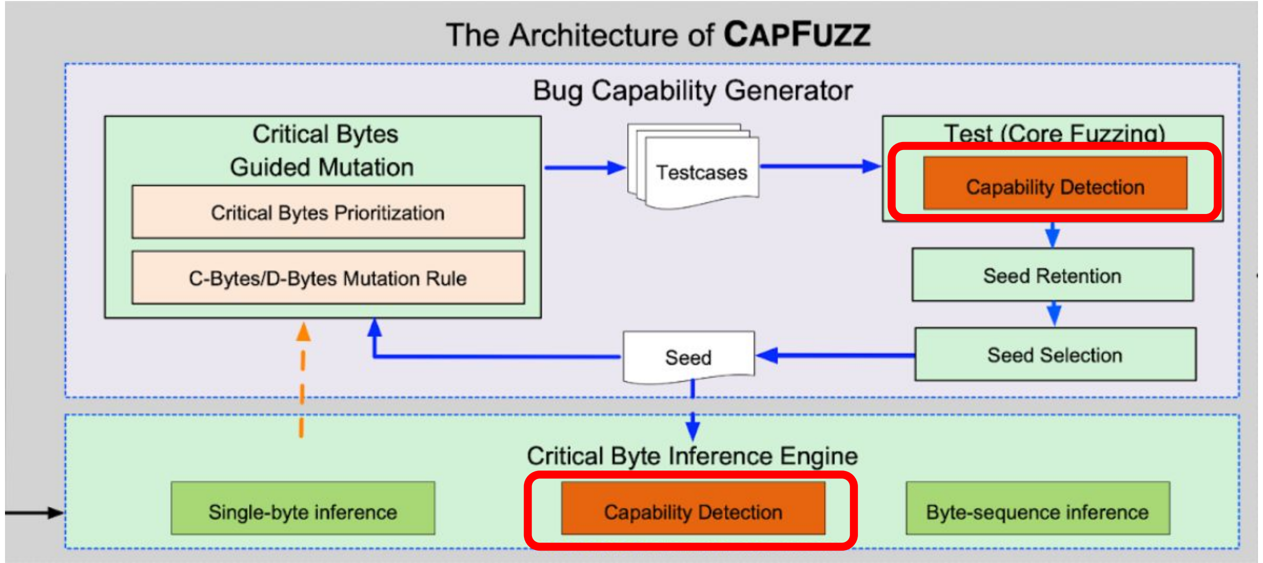

Bugs
Capabilities of Bug A
Capabilities of Bug B
Capabilities of Bug A
Threat model
Priority

Evocatio: Automatically Assessing Bug Capabilities



I) Capability Detection: CapSan

- Extract capability of a PoC automatically
- Sensitive to capability changing
- Configurable monitor items
- Convenient and light-weight



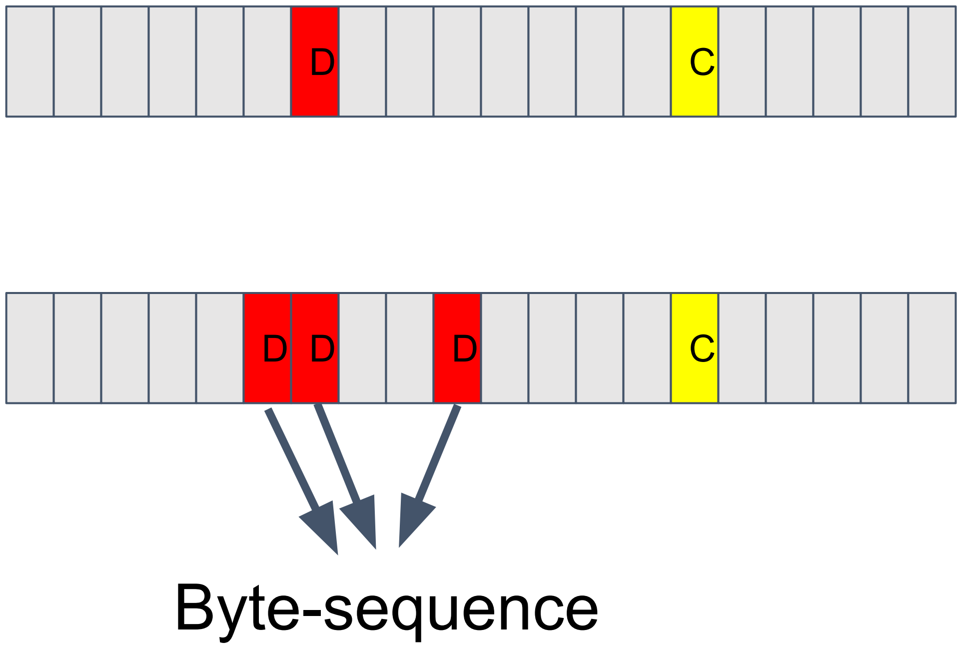
II) Capability Discovery: Critical Bytes Inference

Assess impact for each input byte

- C_{byte} : affecting control flow
- D_{byte} : affecting data flow

❖ Single-byte inference

❖ Byte-sequence inference



III) CapFuzz: Capability guided Fuzzing

Goal: find more capabilities of a bug

Input: single crashing seed

Output: seeds with different capabilities

- Prioritize Critical Bytes
- Mutation
- Seed Retention
- Seed Selection

Severity Assessment

Example threat model

- **Goal: achieve remote code execution**

- Bug type
- Max. length of OOB reads/writes
- Readable/writable address ranges
- Num. of OOB objects
- Max. OOB size objects
- Num. of different read/write offsets

Evaluation

- 38 bugs (34 CVEs + 4 issues)
- One PoC for each bug
- 8 real-world programs
- 6 bug types

Evaluation: Capabilities discovered by Evocatio

CVE	Bug Effect	Size		Origin		Origin Size		Origin Offset	
		Read	Write	Stack	Heap	Read	Write	Read	Write
CVE-2016-9532	HOF[SOF]	$[2^0 \ldots 2^3]2$	$[\mathbf{2^0}]1$	1	9	$[2^0 \ldots 2^{20}]65374$	$[\mathbf{2^0}]1$	$[2^0 \ldots 2^8]2$	$[\mathbf{2^0}]1$
CVE-2018-7871	HOF[W|UAF|N]	$[2^0 \ldots 2^3]4$	$[2^2 \ldots 2^3]6$	0	408	$[2^0 \ldots 2^{14}]216$	$[2^2 \ldots 2^{18}]13$	$[2^3 \ldots 2^{10}]54$	$[2^0]1$
CVE-2019-16705	HOF[W|UAF]	$[2^0 \ldots \mathbf{2^0} \ldots 2^{10}]3$	$[2^5 \ldots 2^5]1$	0	42	$[2^0 \ldots \mathbf{2^8} \ldots 2^{12}]81$	$[2^{15} \ldots 2^{18}]11$	$[\mathbf{2^3} \ldots 2^{10}]44$	$[2^0]1$
CVE-2021-3156	HOF[-]	-	$[\mathbf{2^0} \ldots 2^{10}]694$	0	2	-	$[2^2 \ldots \mathbf{2^4} \ldots 2^5]31$	-	$[2^0 \ldots 2^{10}]2$

- ~50% in the same risk level, quantitative estimate of severity
- Out of 16 patched CVEs, 7 patches were incomplete (and bypassable)

Key takeaways

Fuzzing detects bugs, assessing their severity is hard

- Programmers are overwhelmed by too many reports
- Bug severity assessment must be automatic and objective
- Completely fixing a bug is hard based on a single PoC

Our findings

- Bug capabilities give developers context
- Guided fuzzing detects underlying bug capabilities
- Evocatio detected 7 incomplete patches, generating new capabilities
- https://github.com/HexHive/Evocatio