RUB

CASA
Securing the Digital Society

Chair for
Software Security

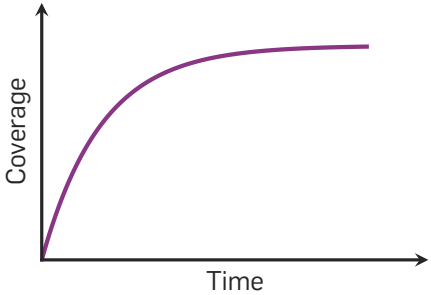
StorFuzz: Using Data-Diversity to Overcome Fuzzing Plateaus

Leon Weiß Tobias Holl Kevin Borgolte

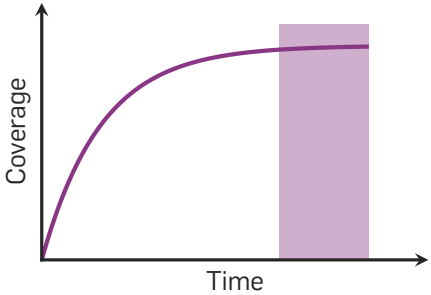
Ruhr University Bochum

April 16, 2026
48th International Conference on Software Engineering (ICSE)

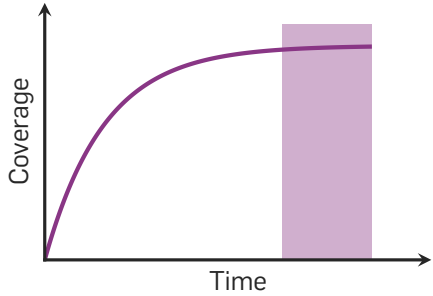
Fuzzing Plateaus



Fuzzing Plateaus

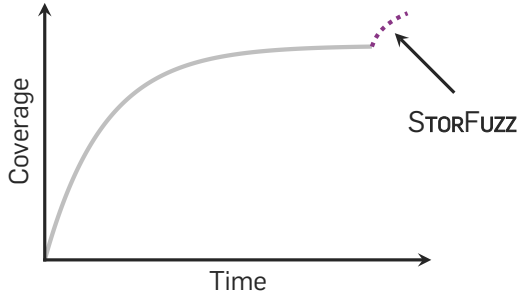


Fuzzing Plateaus



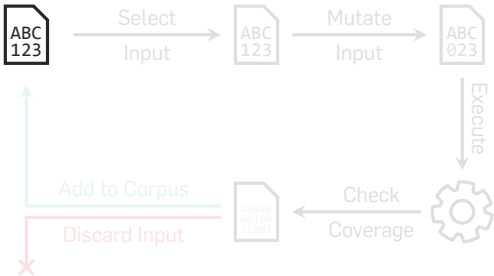
When we reach fuzzing plateaus we usually have **not** covered all code yet.

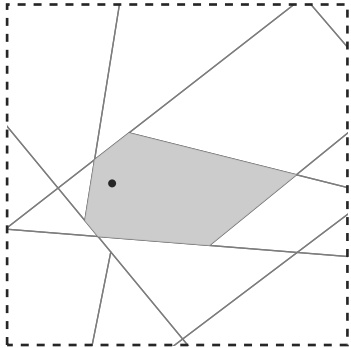
Fuzzing Plateaus



When we reach fuzzing plateaus we usually have **not** covered all code yet.

Fuzzing — Conceptually

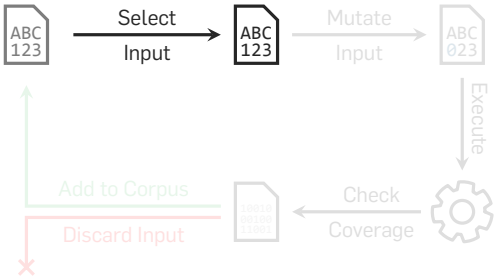


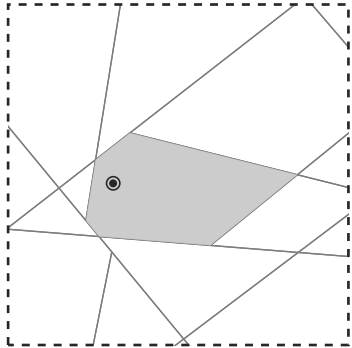


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

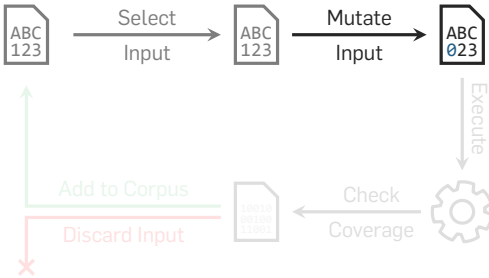


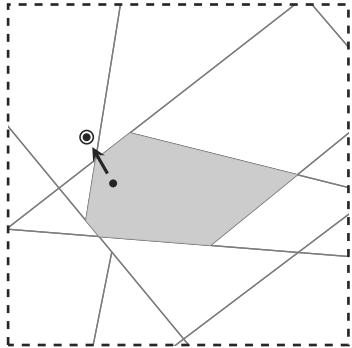


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

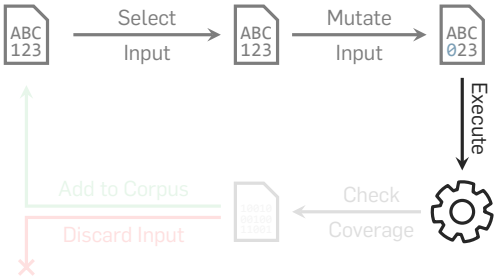


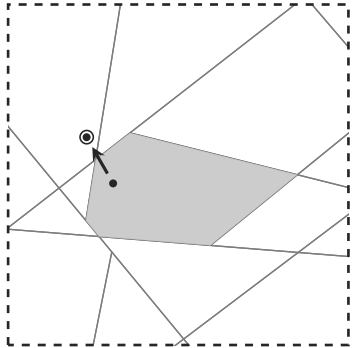


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

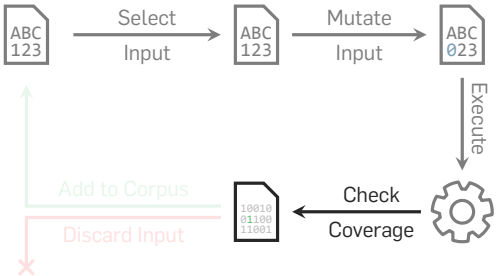


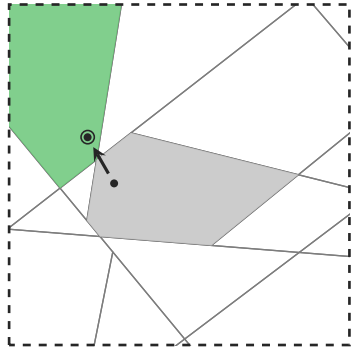


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

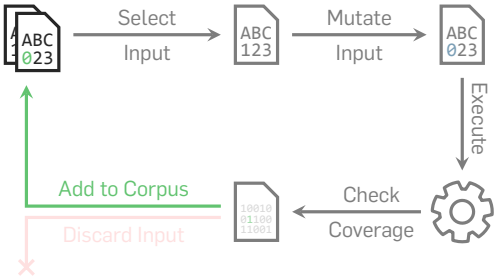


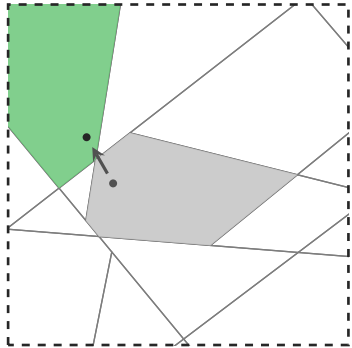


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

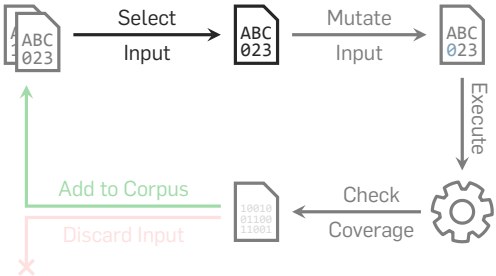


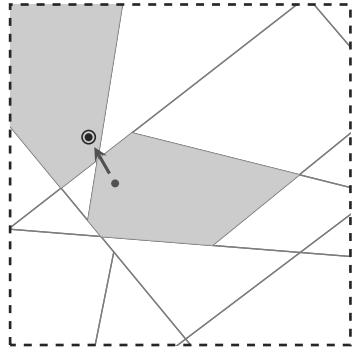


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

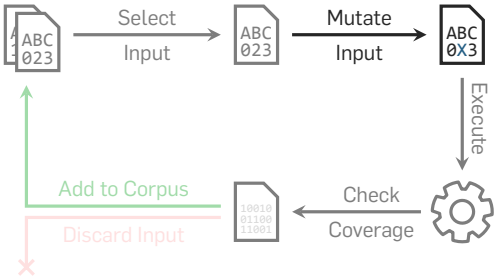


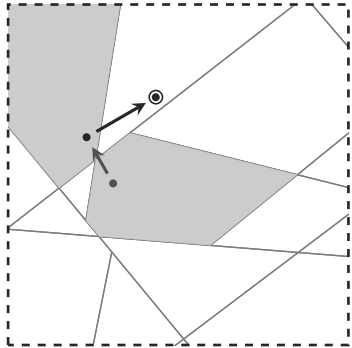


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

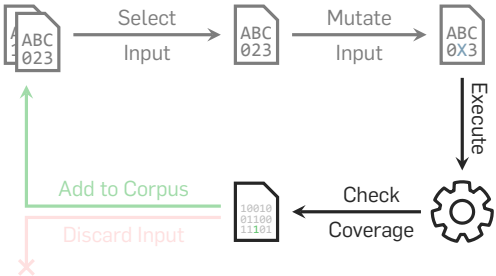


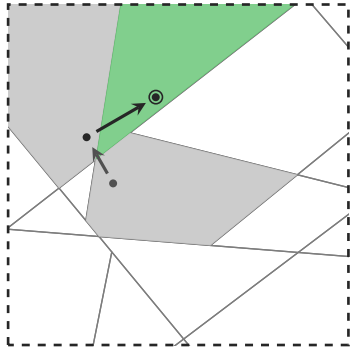


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

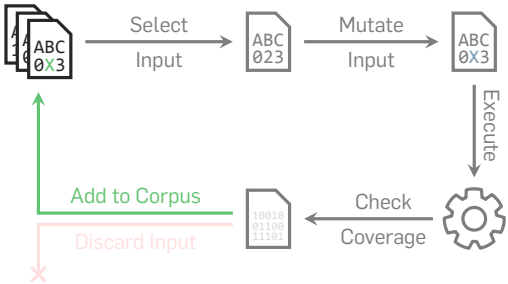


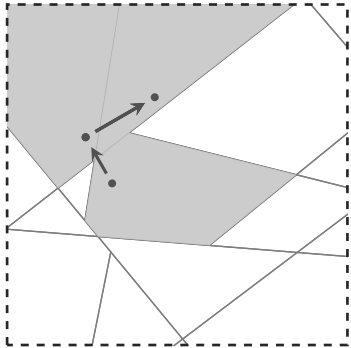


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

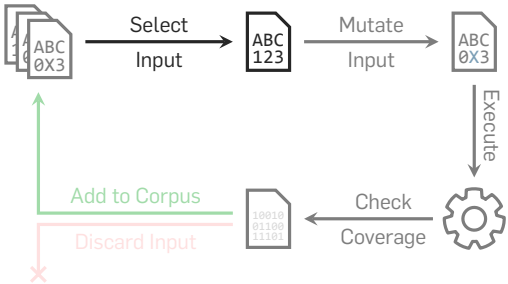


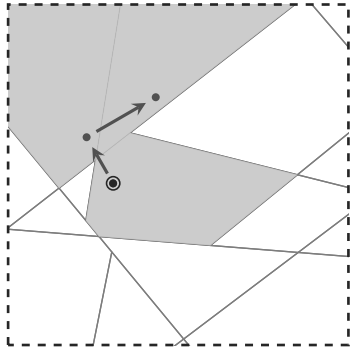


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

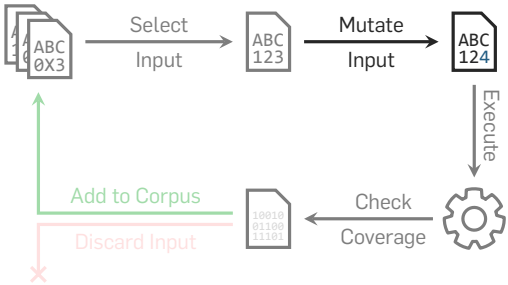


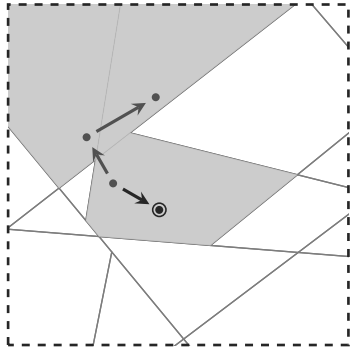


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

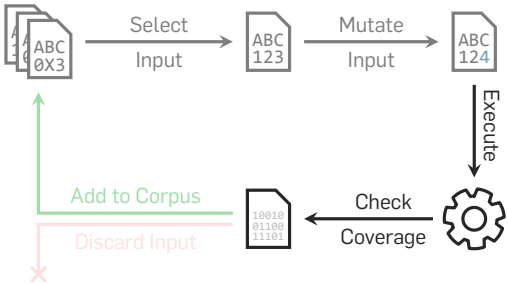


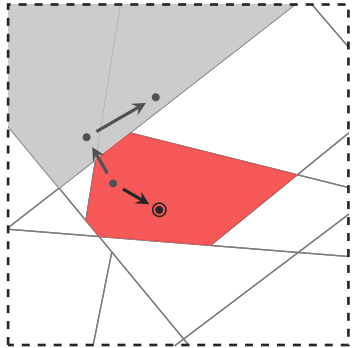


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

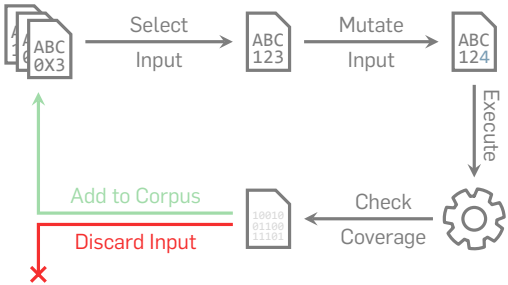


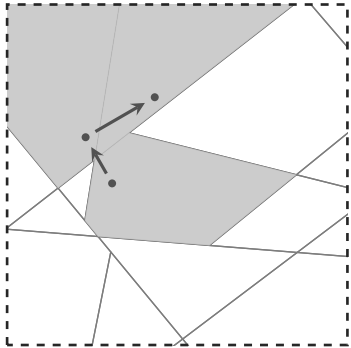


Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

Fuzzing — Conceptually

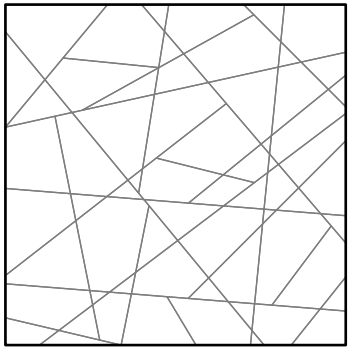




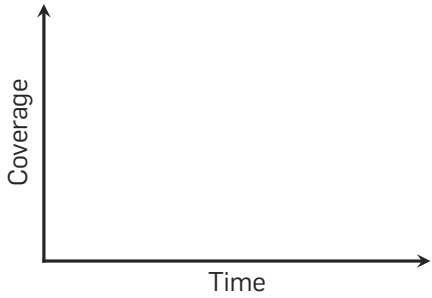
Input Space (magnified)

Fuzzers **randomly sample** from the input space in search of **new code coverage**.

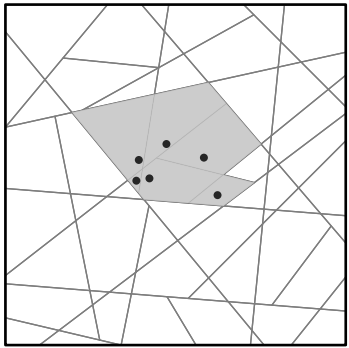
Reaching the Plateau



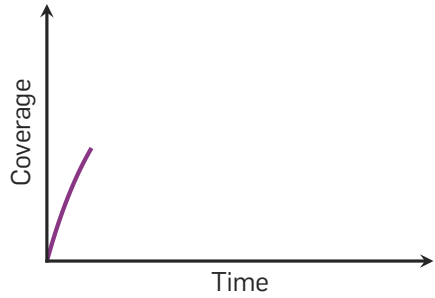
Input Space



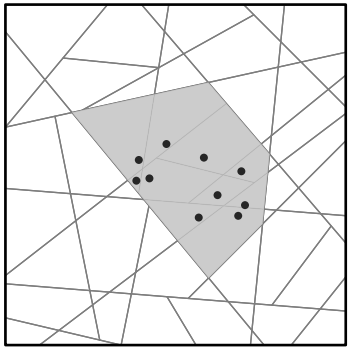
Reaching the Plateau



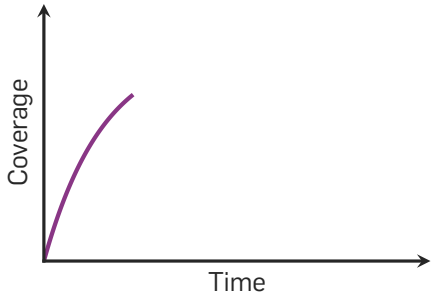
Input Space



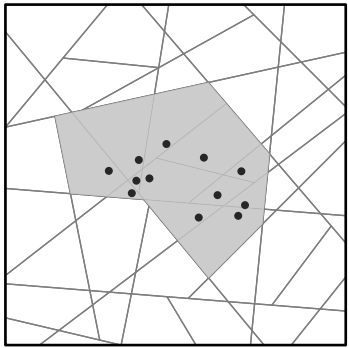
Reaching the Plateau



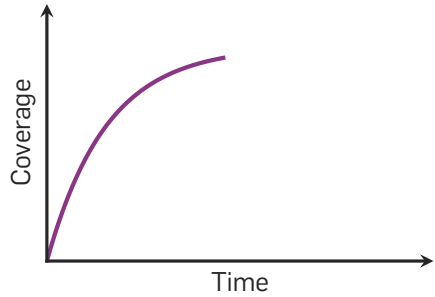
Input Space



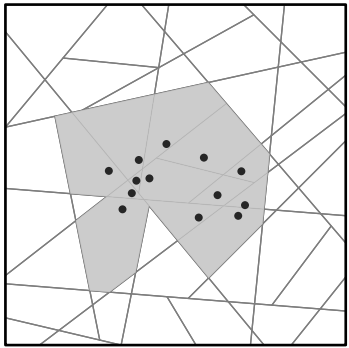
Reaching the Plateau



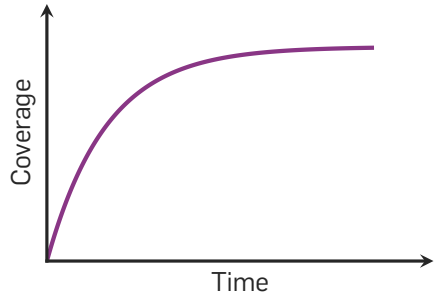
Input Space



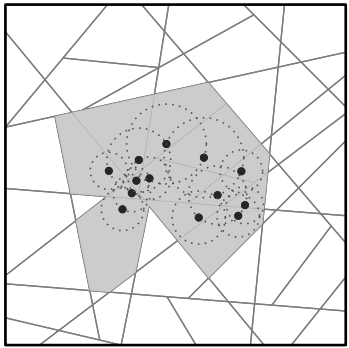
Reaching the Plateau



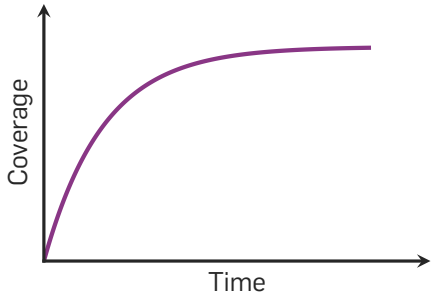
Input Space



Reaching the Plateau

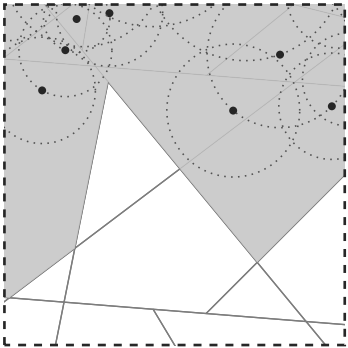


Input Space

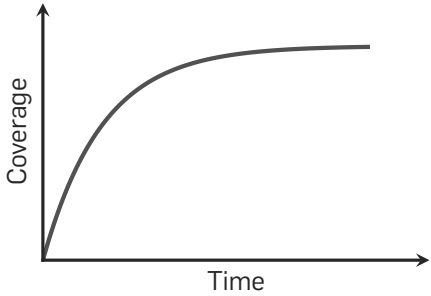


Plateaus are caused by **poorly placed** inputs so **mutations cannot reach new code**.

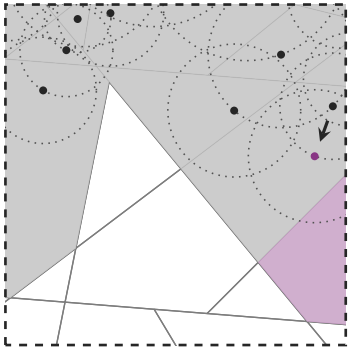
Overcoming the Plateau



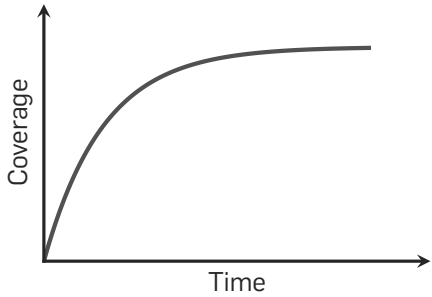
Input Space (magnified)



Overcoming the Plateau

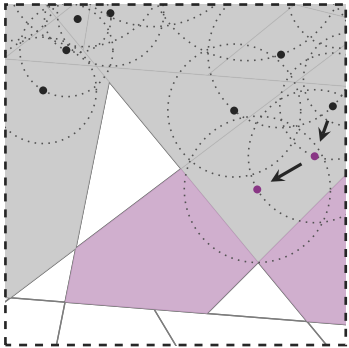


Input Space (magnified)

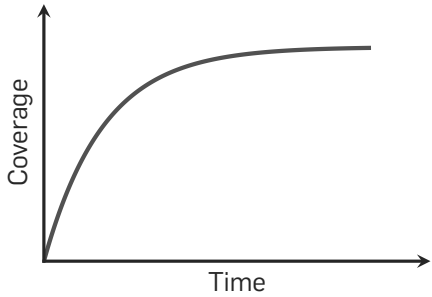


To overcome the plateau we can use **intermediate steps** to **diversify** the corpus.

Overcoming the Plateau

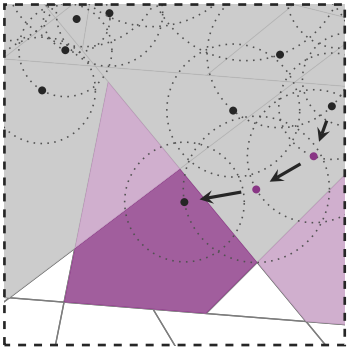


Input Space (magnified)

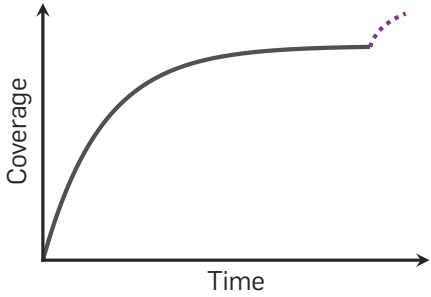


To overcome the plateau we can use **intermediate steps** to **diversify** the corpus.

Overcoming the Plateau



Input Space (magnified)



To overcome the plateau we can use **intermediate steps** to **diversify** the corpus.

StorFuzz — Core Idea

- Observation: More **diverse inputs** improve the chances of overcoming the plateau.

StorFuzz — Core Idea

- Observation: More **diverse inputs** improve the chances of overcoming the plateau.
- We want to reward **diversity in program behavior** during fuzzing.

StorFuzz — Core Idea

- Observation: More **diverse inputs** improve the chances of overcoming the plateau.
- We want to reward **diversity in program behavior** during fuzzing.
- Behavior at runtime is decided by the program itself, its **state**, and the given input.
- State is the combination of current location within the program and **stored values**.

StorFuzz uses **diversity** in the target's **stored values** as a measure for **diversity in state**.

StorFuzz — Optimizations

- Meaningful state is encoded in **long-lived** data.

StorFuzz — Optimizations

- Meaningful state is encoded in **long-lived** data.
- StorFuzz uses compile-time instrumentation to collect coverage on **stores to memory**.

StorFuzz — Optimizations

- Meaningful state is encoded in **long-lived** data.
- StorFuzz uses compile-time instrumentation to collect coverage on **stores to memory**.
- Some values are **not** related to state.

StorFuzz — Optimizations

- Meaningful state is encoded in **long-lived** data.
- StorFuzz uses compile-time instrumentation to collect coverage on **stores to memory**.
- Some values are **not** related to state.
- StorFuzz **excludes** loop counters, constants, copies, bulk operations, and pointers.

StorFuzz — Optimizations

- Meaningful state is encoded in **long-lived** data.
- StorFuzz uses compile-time instrumentation to collect coverage on **stores to memory**.
- Some values are **not** related to state.
- StorFuzz **excludes** loop counters, constants, copies, bulk operations, and pointers.
- State related variables only hold a **limited amount** of information.

StorFuzz — Optimizations

- Meaningful state is encoded in **long-lived** data.
- StorFuzz uses compile-time instrumentation to collect coverage on **stores to memory**.
- Some values are **not** related to state.
- StorFuzz **excludes** loop counters, constants, copies, bulk operations, and pointers.
- State related variables only hold a **limited amount** of information.
- StorFuzz **reduces values** to 8 bits and stores coverage in a separate map.

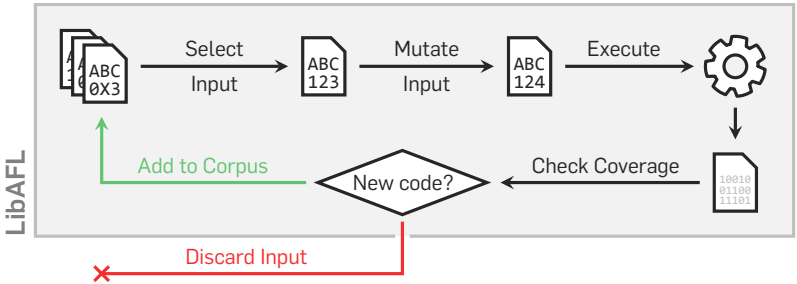
StorFuzz — Optimizations

- Meaningful state is encoded in **long-lived** data.
- StorFuzz uses compile-time instrumentation to collect coverage on **stores to memory**.
- Some values are **not** related to state.
- StorFuzz **excludes** loop counters, constants, copies, bulk operations, and pointers.
- State related variables only hold a **limited amount** of information.
- StorFuzz **reduces values** to 8 bits and stores coverage in a separate map.
- Each individual instruction stores to the **same variable** over time.

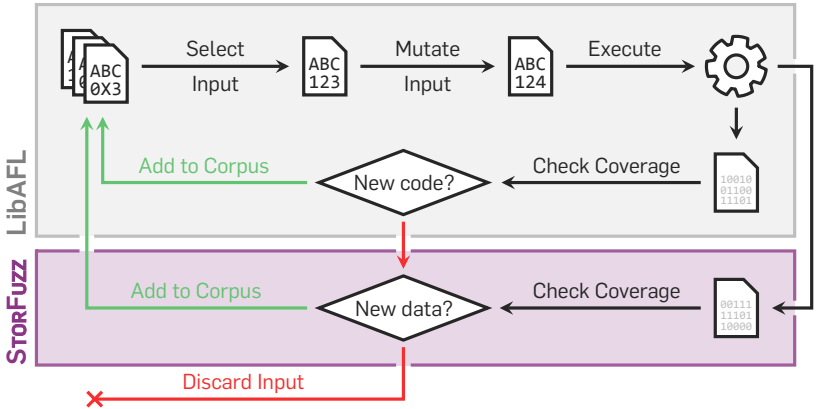
StorFuzz — Optimizations

- Meaningful state is encoded in **long-lived** data.
- StorFuzz uses compile-time instrumentation to collect coverage on **stores to memory**.
- Some values are **not** related to state.
- StorFuzz **excludes** loop counters, constants, copies, bulk operations, and pointers.
- State related variables only hold a **limited amount** of information.
- StorFuzz **reduces values** to 8 bits and stores coverage in a separate map.
- Each individual instruction stores to the **same variable** over time.
- StorFuzz combines the reduced value and the **ID of the store instruction** into coverage.

StorFuzz — Fuzzing



StorFuzz — Fuzzing



Data coverage is an **addition** to code coverage **for diversification**.

Evaluation

- We follow **best practices** (FuzzBench, 10 trials).

Evaluation

- We follow **best practices** (FuzzBench, 10 trials).
- We start evaluating from a corpus saturated with the **base fuzzer**.
- Starting from the OSS-Fuzz corpus we **saturate with 5×24h** of fuzzing.

Evaluation

- We follow **best practices** (FuzzBench, 10 trials).
- We start evaluating from a corpus saturated with the **base fuzzer**.
- Starting from the OSS-Fuzz corpus we **saturate with 5×24h** of fuzzing.
- We compare to **LibAFL** (baseline) and **DDFuzz** (based on LibAFL) (and DatAFLow).

Evaluation

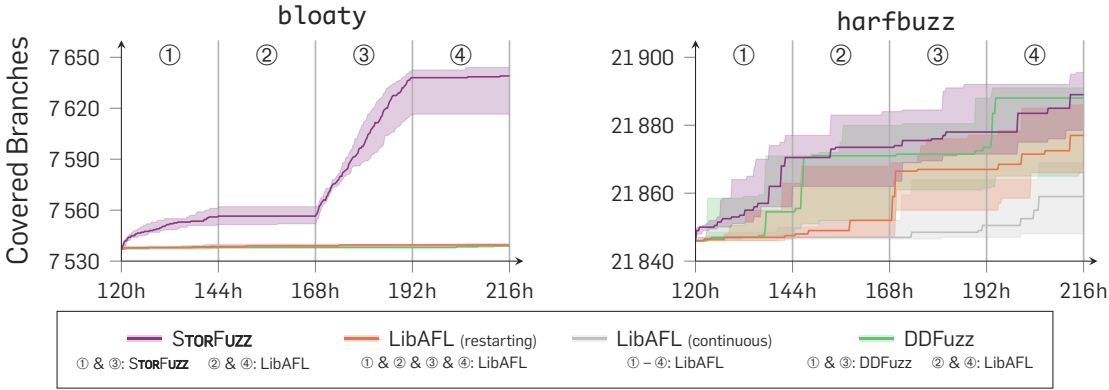
- We follow **best practices** (FuzzBench, 10 trials).
- We start evaluating from a corpus saturated with the **base fuzzer**.
- Starting from the OSS-Fuzz corpus we **saturate with 5×24h** of fuzzing.
- We compare to **LibAFL** (baseline) and **DDFuzz** (based on LibAFL) (and DatAFLow).
- In a separate evaluation we compare to **WingFuzz** (based on **libFuzzer**).

Evaluation — Escaping the Plateau

Benchmark	StorFuzz	LibAFL (baseline)	DDFuzz
bloaty	7 556.5 (+19.5)	7 538.5 (+1.5)	7 538 (+1)
curl	11 569.5 (+15.5)	11 557 (+3)	11 557 (+3)
freetype2	18 054.5 (+11.5)	18 044.5 (+1.5)	18 045 (+2)
harfbuzz	21 870.5 (+24.5)	21 847.5 (+1.5)	21 854.5 (+8.5)
libxml2	16 314 (+3)	16 311.5 (+0.5)	16 313 (+2)
libxslt	12 158.5 (+23.5)	12 143.5 (+8.5)	12 139 (+4)
mbedtls	4 419 (+1)	4 418 (+0)	4 418 (+0)
openh264	9 841 (+2)	9 839 (+0)	9 839 (+0)
openthread	4 651 (+70)	4 637.5 (+56.5)	4 583.5 (+2.5)
proj4	11 119 (+9)	11 112 (+2)	11 111 (+1)
sqlite3	21 592 (+47)	21 578.5 (+33.5)	21 582 (+37)
Best fuzzer after 24h	11/23 benchmarks	0/23 benchmarks	0/23 benchmarks

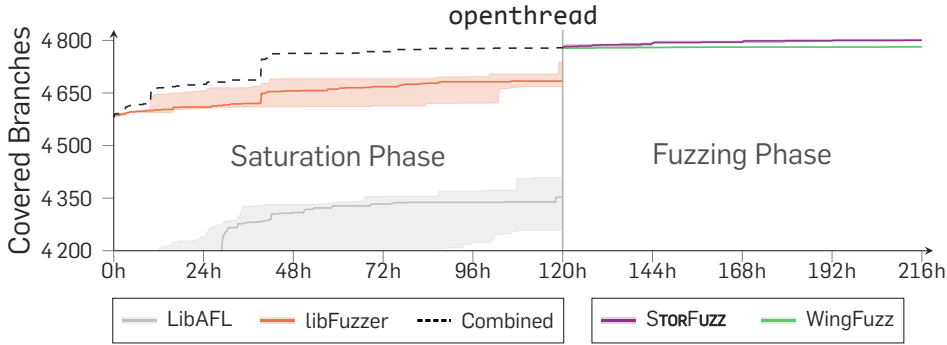
StorFuzz **escapes the plateau** better than restarting LibAFL or diversifying with DDFuzz.

Evaluation — Transferring the Diversity



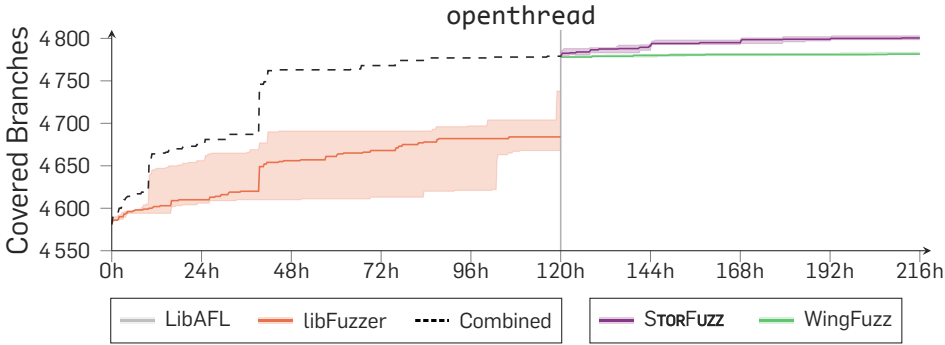
Subsequent fuzzers can effectively **pick up** on corpora diversified by StorFuzz.

Evaluation — Transferring the Diversity: WingFuzz



Both StorFuzz and WingFuzz **improve** over saturated corpora in **distinct ways**. Which of them performs better overall is **benchmark-specific**.

Evaluation — Transferring the Diversity: WingFuzz



Both StorFuzz and WingFuzz **improve** over saturated corpora in **distinct ways**. Which of them performs better overall is **benchmark-specific**.

Evaluation — Transferring the Diversity: WingFuzz

Covered in n trials	Only by StorFuzz			Only by WingFuzz		
	1+	3+	5+	1+	3+	5+
bloaty	24	6	2	58	61	66
curl	14	6	4	18	8	8
freetype2	8	4	1	21	13	9
harfbuzz	51	30	37	15	8	3
libxml2	13	5	3	26	6	5
libxslt	28	22	11	23	14	7
openssl				17	2	1
openthread	48	25	25	11	2	1
proj4	4	12	11	118	50	25
re2				7	4	4
sqlite3	88	31	30	128	31	28

Unique branches covered in at least 1, 3 and 5 trials (for all benchmarks see paper).

StorFuzz and WingFuzz **complement each** other by covering **unique program parts**.

Evaluation — Bug Finding Case Study

- We tested bug finding on the most popular projects in OSS-Fuzz.
- StorFuzz discovered **50** new bugs in 7 well-known projects (VLC, PHP, ImageMagick,...).
- We found bugs in areas **already covered** by OSS-Fuzz and previously untested code.
- Some bugs were introduced **13+** years ago and were not found by **4+** years of fuzzing.

StorFuzz finds bugs both by covering **new code** and **new states** in already covered code.

Takeaways

StorFuzz uses data-diversity to overcome fuzzing plateaus, cover new code, find new bugs and complement existing fuzzers.

Paper, code, data, and slides are available at
`softsec.link/icse26.storfuzz`

Funded by
DFG Deutsche Forschungsgemeinschaft
German Research Foundation

CASA
Securing the Digital Society

WWTF
Vienna Science and Technology Fund

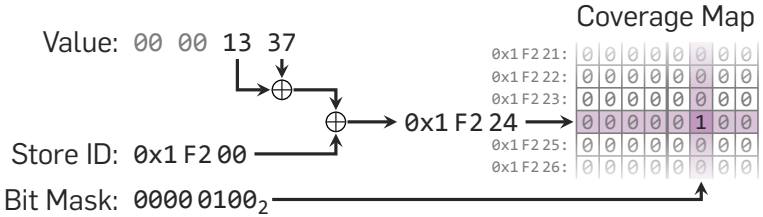
StorFuzz — Value Reduction

- StorFuzz uses a **simple** and **inexpensive** approach.
- Reduction is **directly injected** into the binary near the store.

```
uint8_t value_reduction(uint64_t v) {
    return (v & 0xFF) ^ ((v & 0xFF00) >> 8)
}
```

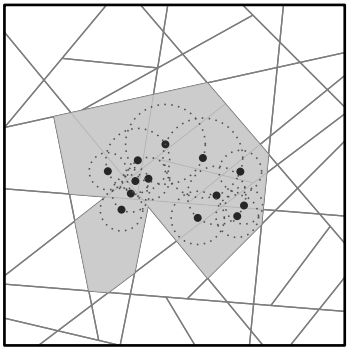
- Other reduction widths are possible (see ablation study).
- Experiments showed the default (8 Bits) performs best in the **general case**.

StorFuzz — Coverage Map



- StorFuzz optimizes the data coverage map by using **each bit** as its own entry.
- The store ID (fixed per store instruction) is split into **two parts**:
  1. Index into the coverage map.
  2. Bit mask to address single bit.
- Recording is done using a **single** `or` of the bit mask with the indexed byte.

Other Concepts of Escaping the Plateau

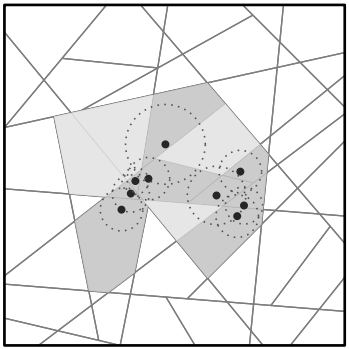


Restarts with Dropouts

(Schiller et al. [1])

Input Space

Other Concepts of Escaping the Plateau

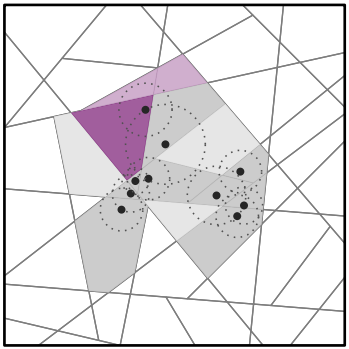


Restarts with Dropouts

(Schiller et al. [1])

Input Space

Other Concepts of Escaping the Plateau



Restarts with Dropouts

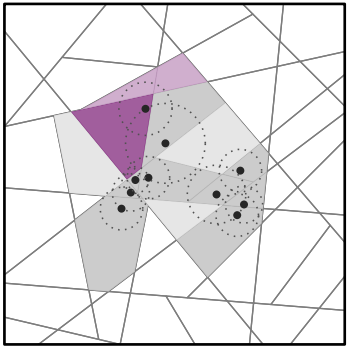
(Schiller et al. [1])

Input Space

Other Concepts of Escaping the Plateau

Restarts with Dropouts

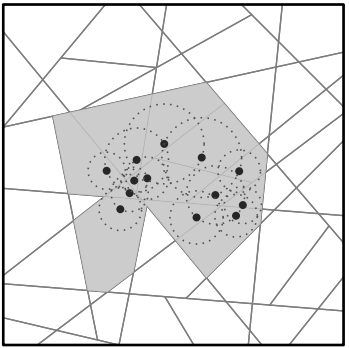
(Schiller et al. [1])



Input Space

Larger Mutation Steps

(e.g., Constraint Solving)

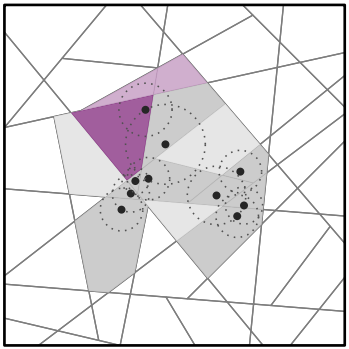


Input Space

Other Concepts of Escaping the Plateau

Restarts with Dropouts

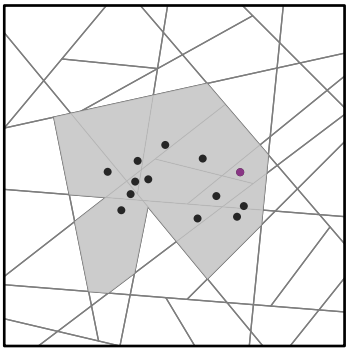
(Schiller et al. [1])



Input Space

Larger Mutation Steps

(e.g., Constraint Solving)

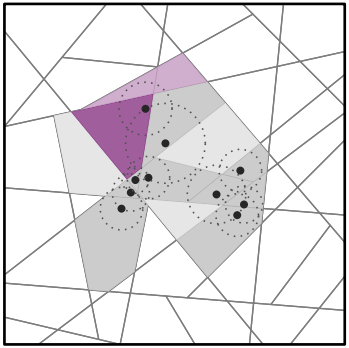


Input Space

Other Concepts of Escaping the Plateau

Restarts with Dropouts

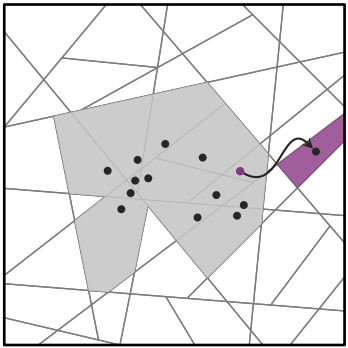
(Schiller et al. [1])



Input Space

Larger Mutation Steps

(e.g., Constraint Solving)

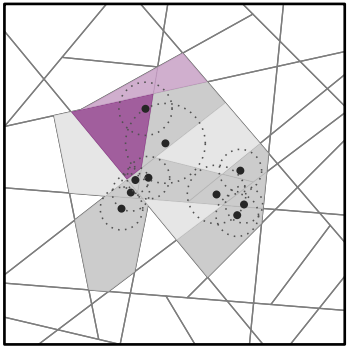


Input Space

Other Concepts of Escaping the Plateau

Restarts with Dropouts

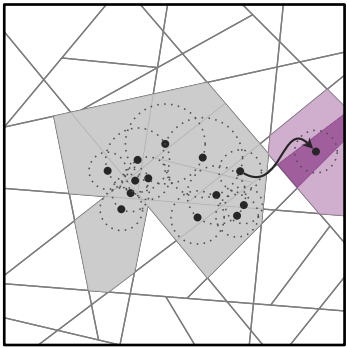
(Schiller et al. [1])



Input Space

Larger Mutation Steps

(e.g., Constraint Solving)



Input Space

Ablation Study

- We performed an ablation study for individual components:
  - Value reduction
  - Threshold for bulk operations
  - Exclusion of plain copies
- We include the data in the artifact.

The default choices in StorFuzz work best in the **general case**. Fine-tuning to an individual benchmark is possible and could improve results further.

References

[1] N. Schiller, X. Xu, L. Bernhard, N. Bars, M. Schloegel, and T. Holz. “Novelty Not Found: Exploring Input Shadowing in Fuzzing through Adaptive Fuzzer Restarts”. In: *ACM Transactions on Software Engineering and Methodology* 34.3 (Mar. 2025). DOI: 10.1145/3712186.