Poster: Beyond Reachability to Controllability via State Transition Entropy

Takuya Shimizu, Toshiki Takatera, Ryo Ozaki, Yudai Fujiwara, Yuichi Sugiyama
Ricerca Security, Inc.
Email: {takuyas, toshikit, ryoo, yudaif, yuichis}@ricsec.co.jp

***Abstract*—Fuzzing generates a large number of crashing inputs, and given a constant influx of new crashes, it is impractical to perform root-cause analysis and exploitability assessment for each one. Prior work assesses severity based on reachability (the range that can be read or written via memory-safety bugs), but cannot distinguish cases where severity differs greatly despite having similar reachability. We instead measure controllability: how freely an attacker can steer the next invalid access target within the reachable range. From diversified crashing inputs, we build a first-order Markov chain over ASan-reported access offsets and score bugs by the trajectory entropy of the chain. In a preliminary study on known CVEs, the entropy score is higher for bugs with higher controllability and separates vulnerabilities that reachability-based metrics fail to distinguish.**

I. Introduction

Fuzzing is widely used to discover software vulnerabilities, and continuous fuzzing platforms such as OSS-Fuzz and Syzbot report many crashing inputs daily. As shown in Figure 1, OSS-Fuzz has reported over 60,000 cumulative bugs, and the average time to fix grew from about 15 days in 2017 to about 280 days in 2025. This indicates that unfixed bugs are accumulating and fixes are not keeping pace. Root-cause analysis and exploit assessment for every crashing input are impractical. Therefore, a method to automatically assess the **fixing priority** (i.e., severity) of crashing inputs is needed.

Existing studies assess severity based on reachability [1] or controllable value set [2] but overlook the *sequential* freedom of accesses. Here, reachability denotes the range readable or writable via memory-safety violations; for example, Evocatio [1] mutates crashing inputs to explore reachable ranges and evaluates bugs based on their diversity. Yet vulnerabilities can exhibit different severity despite similar reachability, as illustrated with a concrete example in Section II.

We focus on **controllability** rather than reachability, defining it as how freely an attacker can choose the next access target within the reachable range. We treat each access offset from the overflowed object as a state and quantify controllability using the **entropy** of state transitions. We diversify the crashing input through fuzzing, extract offsets from ASan reports [3], build a Markov model over the resulting state-transition sequences, and compute trajectory entropy. In a preliminary evaluation on known CVEs, the proposed method distinguishes vulnerabilities with high and low controllability.

II. Motivating Example

Figure 2 shows two vulnerabilities with similar reachability but different severity. Example (a) writes sequentially to all columns in the specified row, whereas Example (b) writes only to mask-specified columns. Both have a bounds-check bug that allows access to rows outside the buffer. The reachable range (rows that can be accessed illegally) is identical for both.

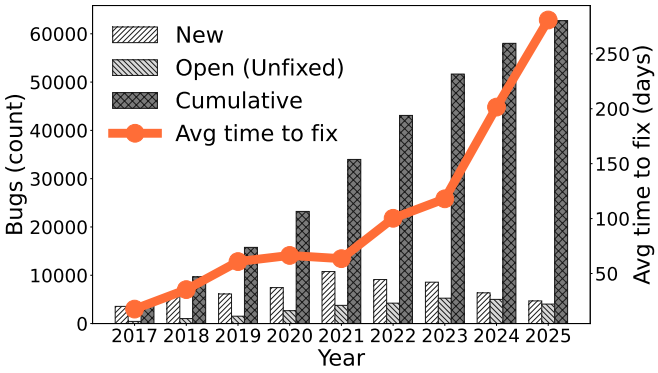


Fig. 1. Trends in the number of bug reports and the average time to fix in OSS-Fuzz (2017–2025). New: number of new reports per year, Open: unresolved reports at year end, Cumulative: cumulative reports at year end, Avg time to fix: average days to fix bugs resolved in that year (right axis).

However, their severity differs greatly because controllability differs, namely how freely an attacker can choose the next write target. In Example (a), once an invalid row is specified, the write offsets within that row are fixed consecutively as columns $0, 1, 2, \ldots$, whereas Example (b) allows arbitrary offset combinations within the invalid row, allowing the attacker to target specific memory locations. Existing reachability-based methods cannot capture this difference in controllability.

III. Approach

The proposed method is based on the observation that higher controllability yields more choices for the next state and thus higher trajectory entropy. To estimate this entropy, we build a first-order Markov model from ASan memory-violation reports. We define each state s as an offset from the overflowed object. Specifically, we diversify the crashing PoC through fuzzing to generate many inputs, execute each input, and extract **access offsets** from the overflowed object using ASan reports. We disable ASan's *halt_on_error* option to record consecutive memory violations from a single input and treat the access sequence as a state-transition sequence.

From the observed transition sequences, we estimate transition probabilities $P(s' \mid s)$ by maximum likelihood and define

```c
void tableCopyRow(
  table_t *t, size_t row, int *vals
) {
  if (row > t->n_row) return; // <- bug
  for (size_t i = 0; i < t->n_col; i++)
    t->cells[row*t->n_col+i] = vals[i];
}
```

(a) Sequential writes to all columns

```c
void tableCopyMaskedRow(
  table_t *t, size_t row, int *vals, bool *mask
) {
  if (row > t->n_row) return; // <- bug
  for (size_t i = 0; i < t->n_col; i++)
    if (mask[i])
      t->cells[row*t->n_col+i] = vals[i];
}
```

(b) Writes only to mask-specified columns

Fig. 2. Two bugs with similar reachability but different controllability. Both can access an invalid row due to a bounds-check bug (> should be >=).

the local entropy $H(s)$ for each state s as follows.

$$H(s) = -\sum_{s'} P(s' \mid s) \log P(s' \mid s)$$

Furthermore, letting $E[N_s]$ be the expected number of visits to state s from the initial state, the trajectory entropy of the Markov model [4] is computed by the following equation and used as the measure of controllability.

$$\mathcal{H} = \sum_{s} E[N_s] H(s)$$

We apply our method to the example in Section II with `n_col = 4`. In Example (a), write-offset transitions are always $0 \to 1 \to 2 \to 3$ regardless of input, so for all states $P(s+1 \mid s) = 1$ and $\mathcal{H} = 0$. In Example (b), transitions depend on the mask: mask$= [T,T,T,T]$ yields $0 \to 1 \to 2 \to 3$, mask$= [T,F,T,F]$ yields $0 \to 2$, and mask$= [T,F,F,T]$ yields $0 \to 3$. Thus, multiple next states exist from state 0, giving positive entropy, and the proposed method quantifies controllability differences even when reachability is identical.

IV. Preliminary Results

To validate the proposed method, we conducted a preliminary evaluation on known memory-corruption vulnerabilities. For each vulnerability, we ran 10 fuzzer instances for 12 hours, using the crashing input as the initial seed. The fuzzer is based on Evocatio [1], which has the strategy to increase the variety of ASan error types triggered by the bug and diversify crashing inputs. We then explored the reachable offset range from the overflowed object. We computed trajectory entropy for the resulting input set and compared it with severity judgments by human analysts.

Table I summarizes the results. CVE-2019-9114 and CVE-2020-11894 have similar offset-range widths (93 and 94), yet human severity assessments differ; the proposed method

TABLE I

Preliminary evaluation results. The offset range is the pair of minimum and maximum offsets explored by the fuzzer, and severity is judged by human analysts (☹: high, ☺: low).

CVE ID	Offset range	Entropy	Severity
CVE-2019-9114	$[-16, 77]$	4.13	☹
CVE-2020-11894	$[-15, 79]$	0.68	☺
CVE-2020-21827	$[-16, 4252]$	5.54	☹
CVE-2022-3598	$[-21, 9498]$	1.24	☺

reflects this with high entropy (4.13) for CVE-2019-9114 and low entropy (0.68) for CVE-2020-11894. Likewise, for CVE-2020-21827 and CVE-2022-3598, the latter has a wider offset range but lower severity, and the entropy aligns with that judgment. In addition, for CVE-2020-21827, human analysis produced an exploit that works under specific conditions, such as a no-ASLR (Address Space Layout Randomization) and no-PIE (Position-Independent Executable) environment. Overall, the method quantifies controllability differences that reachability (offset range) alone cannot capture.

V. Conclusion and Future Direction

In this work, we focus on controllability that reachability-based methods miss and quantify it via trajectory entropy to assess memory-corruption severity. The method builds a Markov model from ASan memory-violation reports and computes trajectory entropy to distinguish vulnerabilities with different severity even when reachability is similar. In the preliminary evaluation, we confirmed that vulnerabilities with higher controllability receive higher entropy.

The proposed method has limitations. The estimated entropy depends on the set of crashing inputs; insufficient coverage reduces estimation accuracy. Our evaluation is an initial result on a limited number of vulnerabilities; we plan to validate on large-scale datasets such as OSS-Fuzz and Syzbot and to conduct broad evaluations across multiple vulnerability types.

Acknowledgment

This paper is based on results obtained from two projects: JPJ004596, commissioned by the Acquisition, Technology & Logistics Agency (ATLA) Innovative Science and Technology Initiative for Security; and JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] Z. Jiang, S. Gan, A. Herrera, F. Toffalini, L. Romerio, C. Tang, M. Egele, C. Zhang, and M. Payer, "Evocatio: Conjuring bug capabilities from a single poc," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[2] G. Lacombe and S. Bardin, "Attacker control and bug prioritization," in *USENIX Security Symposium (USENIX Security)*, 2025.

[3] K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov, "AddressSanitizer: a fast address sanity checker," in *USENIX Conference on Annual Technical Conference (USENIX ATC)*, 2012.

[4] M. Kafsi, M. Grossglauser, and P. Thiran, "The entropy of conditional markov trajectories," *IEEE Trans. Inf. Theor.*, p. 5577 – 5583, 2013.

Poster: Beyond Reachability to Controllability via State Transition Entropy

RICERCA SECURITY

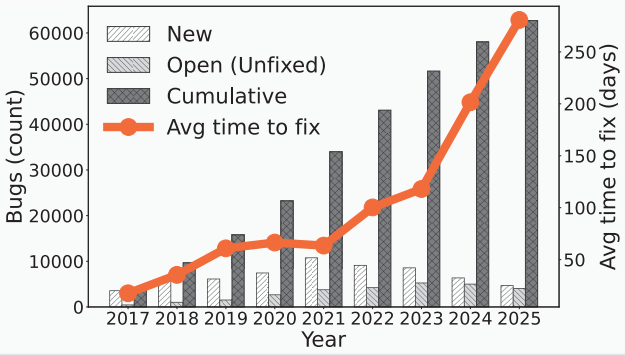
Takuya Shimizu, Toshiki Takatera, Ryo Ozaki, Yudai Fujiwara, Yuichi Sugiyama

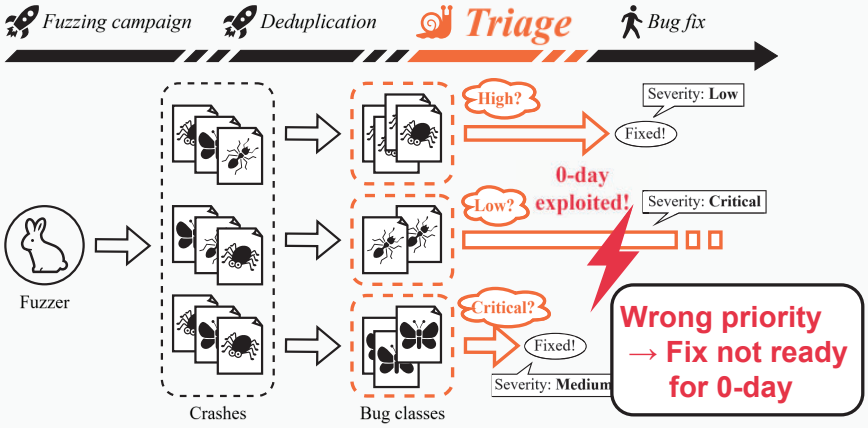
Ricerca Security, Inc.

1. Bug Triage Crisis: Which to Fix First?

Fuzzing success: 60,000+ bugs found

Time required for fix: **15 ➡ 280 days**





2. Missing Factor: Controllability, Not Just Reachability

Which bug do attackers prefer?

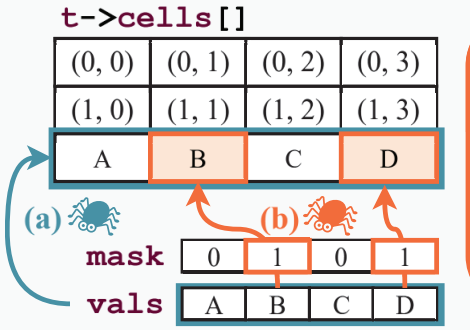
(a) Sequential overflow

```
1  void tableCopyRow(
2    table_t *t, size_t row, int *vals
3  ) {
4    if (row > t->n_row) return; // <- bug
5    for (size_t i = 0; i < t->n_col; i++)
6      t->cells[row*t->n_col+i] = vals[i];
7  }
```

(b) Selective overflow

```
1  void tableCopyMaskedRow(
2    table_t *t, size_t row, int *vals,
3    bool *mask
4  ) {
5    if (row > t->n_row) return; // <- bug
6    for (size_t i = 0; i < t->n_col; i++)
7      if (mask[i])
8        t->cells[row*t->n_col+i] = vals[i];
9  }
```

Both overflow into the same region
Same reachability — same risk?



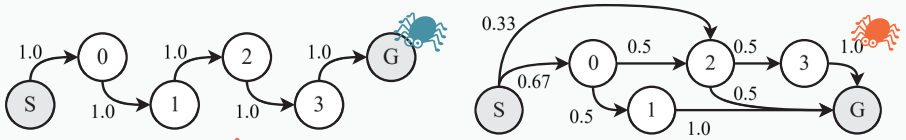
Core Insight

Reachability: same
Controllability: different
➡ Gap = Exploitability

Our Approach

1. Collect bug-triggering inputs
2. Observe state transitions
3. Build Markov chain model
4. Compute entropy :

$$\mathcal{H} = \sum_{s \in S} \underbrace{\mathbb{E}[N(s)]}_{\text{Expected visits}} \cdot \underbrace{H(s)}_{\text{Local entropy}}$$



$\mathcal{H} = 0$ **is more controllable (riskier)** $\mathcal{H} = 1.56$

3. Evaluation Results

Pair 1: Similar range, different risk

CVE ID	CVSS	PUT	Offset Range (Reachability)	Entropy (Controllability)	Risk (Oracle)
CVE-2019-9114	8.8	libming	**93** (-16, 77)	4.13	HIGH
CVE-2020-11894	9.1	libming	**94** (-15, 79)	0.68	LOW

Pair 2: Larger range ≠ higher risk

CVE-2020-21827	7.8	LibreDWG	**4268** (-16, 4252)	5.54	HIGH
CVE-2022-3598	6.5	LibTIFF	**9519** (-21, 9498)	1.24	LOW

❌ Offset range (reachability): cannot properly assess the risk
✅ Entropy (controllability): conforms to human experience

4. Summary & Future Work

- ✓ Same reachability ≠ same risk
- ✓ Entropy quantifies controllability (attacker's control freedom)
- ✓ Aligns with human analyst judgment

- ☐ Larger datasets (OSS-Fuzz, Syzbot)
- ☐ Extensible state definitions for threat models

References:
[1] Jiang et al., "Evocatio: Conjuring Bug Capabilities from a Single PoC," CCS '22
[2] Lacombe & Bardin, "Attacker Control and Bug Prioritization," USENIX Security '25
[3] Serebryany et al., "AddressSanitizer: A Fast Address Sanity Checker," USENIX ATC '12

K Program
ATLA Acquisition,Technology & Logistics Agency

Network and Distributed System Security (NDSS 2026)

This paper is based on results obtained from two projects: JPJ004596, commissioned by the Acquisition, Technology & Logistics Agency (ATLA) Innovative Science and Technology Initiative for Security; and JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).