MEMLOCK: Memory Usage Guided Fuzzing

[ICSE’20]

张灵毓

Outline

1. Background
   - Branch coverage guidance fuzzing (e.g. AFL)
   - Memory consumption bugs

2. Approach

3. Evaluation

Background

Branch coverage guidance

Suppose already know :
(En, B1), (B1, B2), (B2, B4), (B4, B5), (B5, Ex)

Initial:

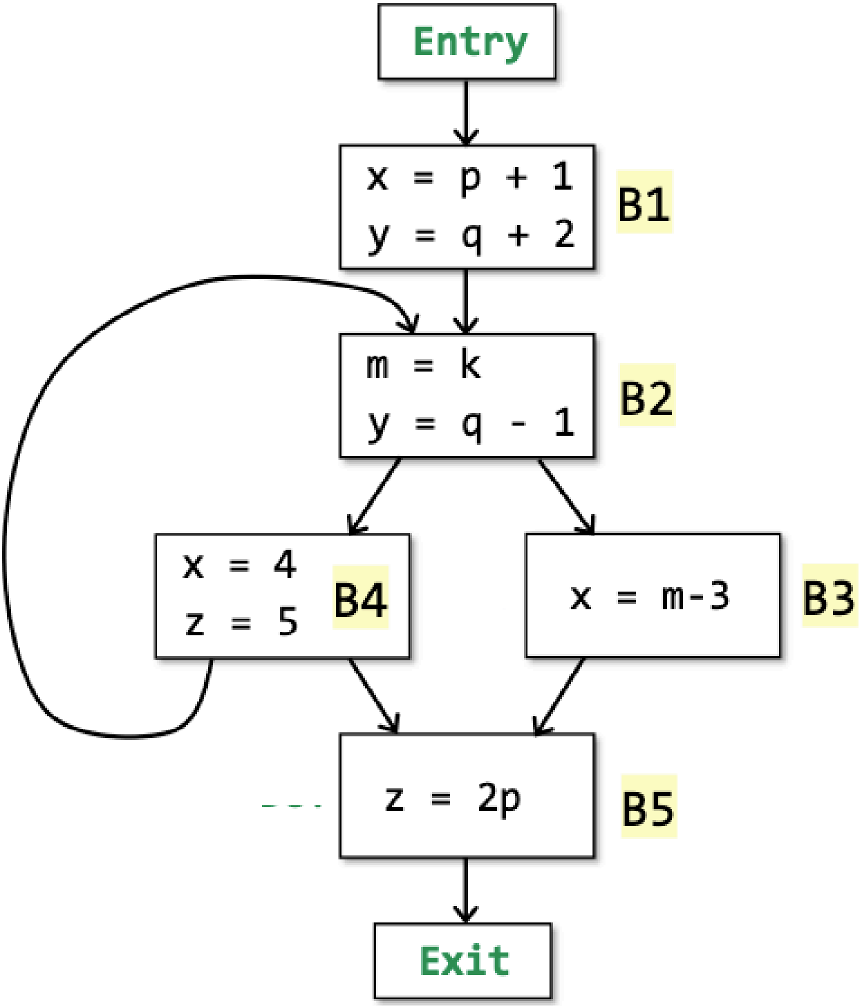
(En, B1):	00000000	(B1, B2):	00000000
(B2, B3):	00000000	(B2, B4):	00000000
(B4, B2):	00000000	(B3, B5):	00000000
(B4, B5):	00000000	(B5, Ex):	00000000

En → B1 → B2 → B3 → B5 → Ex

Final:

(En, B1):	00000001	(B1, B2):	00000001
(B2, B3):	00000001	(B2, B4):	00000000
(B4, B2):	00000000	(B3, B5):	00000001
(B4, B5):	00000000	(B5, Ex):	00000001

New branches



Branch coverage guidance

Suppose already know :

(En, B1), (B1, B2), (B2, B4), (B4, B2), (B4, B5), (B5, Ex)

Initial:

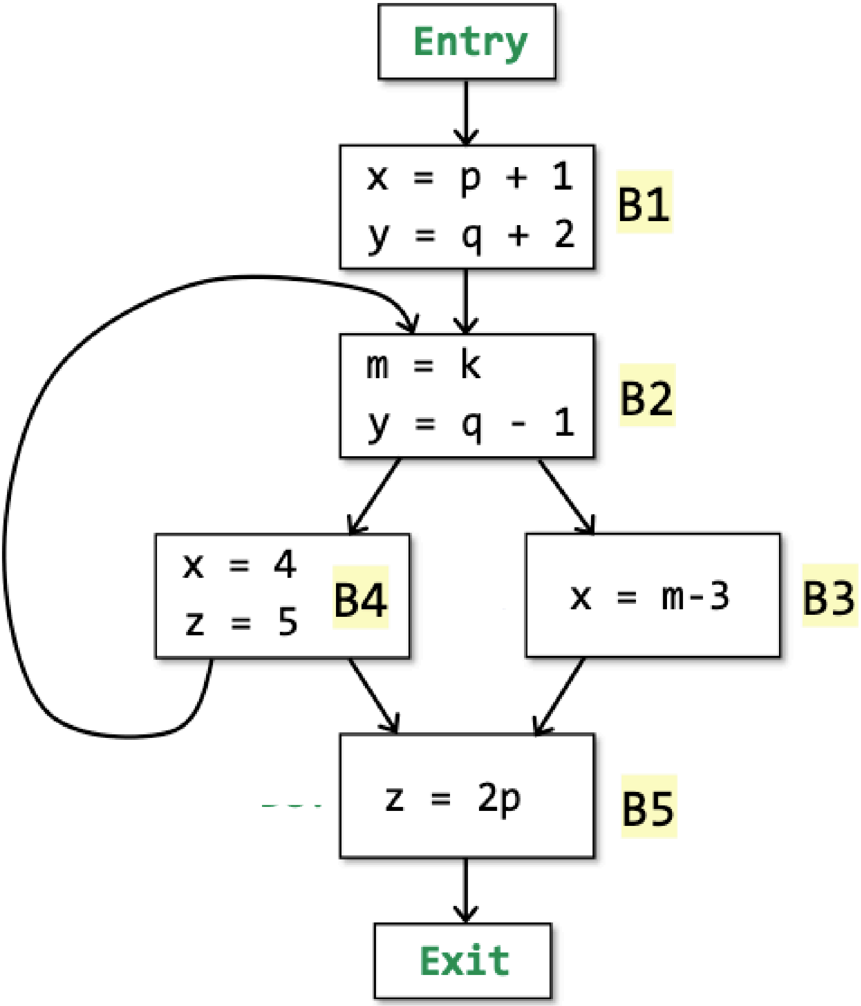
(En, B1):	00000000	(B1, B2):	00000000
(B2, B3):	00000000	(B2, B4):	00000000
(B4, B2):	00000000	(B3, B5):	00000000
(B4, B5):	00000000	(B5, Ex):	00000000

En → B1 → B2 → B4 →B2→ B4→ B5 → Ex

Final:

(En, B1):	00000001	(B1, B2):	00000001
(B2, B3):	00000000	(B2, B4):	00000010
(B4, B2):	00000001	(B3, B5):	00000000
(B4, B5):	00000001	(B5, Ex):	00000001

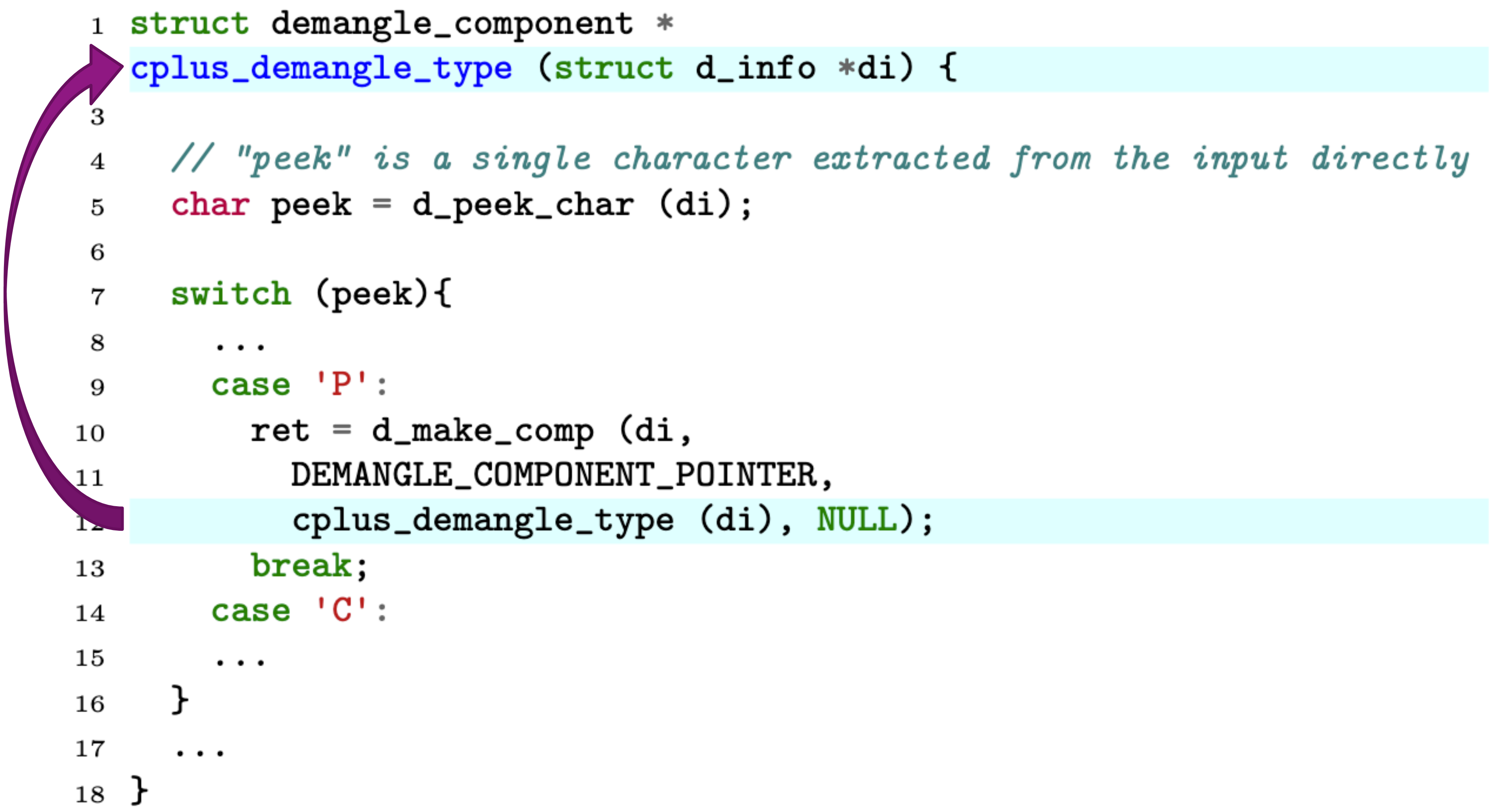
New bucket number



Bucket number calculation

```
static const u8 count_class_lookup8[256] = {
      [0]            = 0,              // 00000000
      [1]            = 1,              // 00000001
      [2]            = 2,              // 00000010
      [3]            = 4,              // 00000100
      [4 ... 7]      = 8,              // 00001000
      [8 ... 15]     = 16,             // 00010000
      [16 ... 31]    = 32,             // 00100000
      [32 ... 127]   = 64,             // 01000000
      [128 ... 255] = 128              // 10000000
};
```

Memory consumption bugs



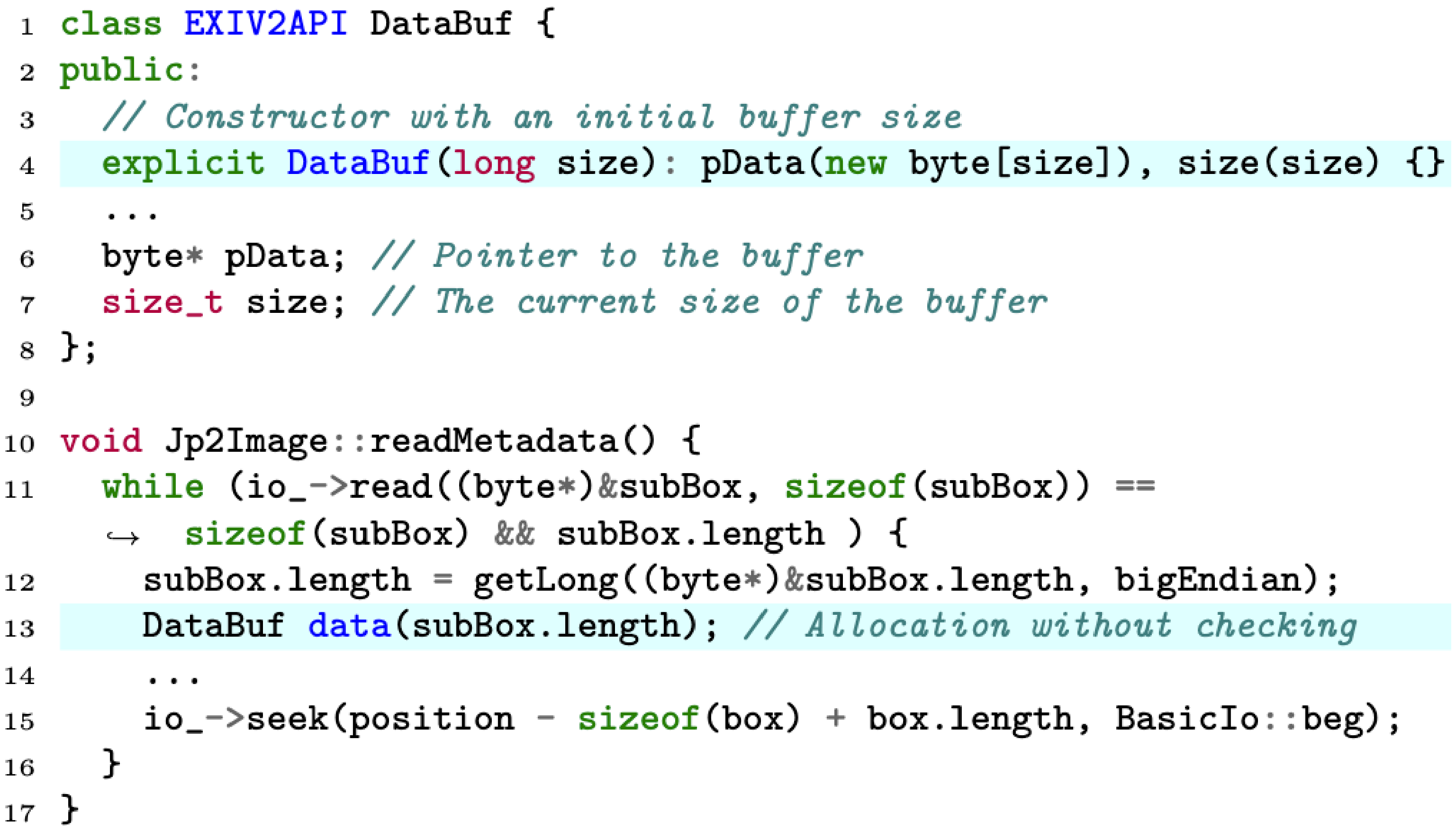
```
struct demangle_component *
cplus_demangle_type (struct d_info *di) {

  // "peek" is a single character extracted from the input directly
  char peek = d_peek_char (di);

  switch (peek){
    ...
    case 'P':
      ret = d_make_comp (di,
        DEMANGLE_COMPONENT_POINTER,
        cplus_demangle_type (di), NULL);
      break;
    case 'C':
    ...
  }
  ...
}
```

Condition for bug: tens of thousands of recursive depth.

Memory consumption bugs



```
class EXIV2API DataBuf {
public:
  // Constructor with an initial buffer size
  explicit DataBuf(long size): pData(new byte[size]), size(size) {}
  ...
  byte* pData; // Pointer to the buffer
  size_t size; // The current size of the buffer
};

void Jp2Image::readMetadata() {
  while (io_->read((byte*)&subBox, sizeof(subBox)) ==
  ↪  sizeof(subBox) && subBox.length ) {
    subBox.length = getLong((byte*)&subBox.length, bigEndian);
    DataBuf data(subBox.length); // Allocation without checking
    ...
    io_->seek(position - sizeof(box) + box.length, BasicIo::beg);
  }
}
```

Condition for bug: generate inputs with a large subBox.length.

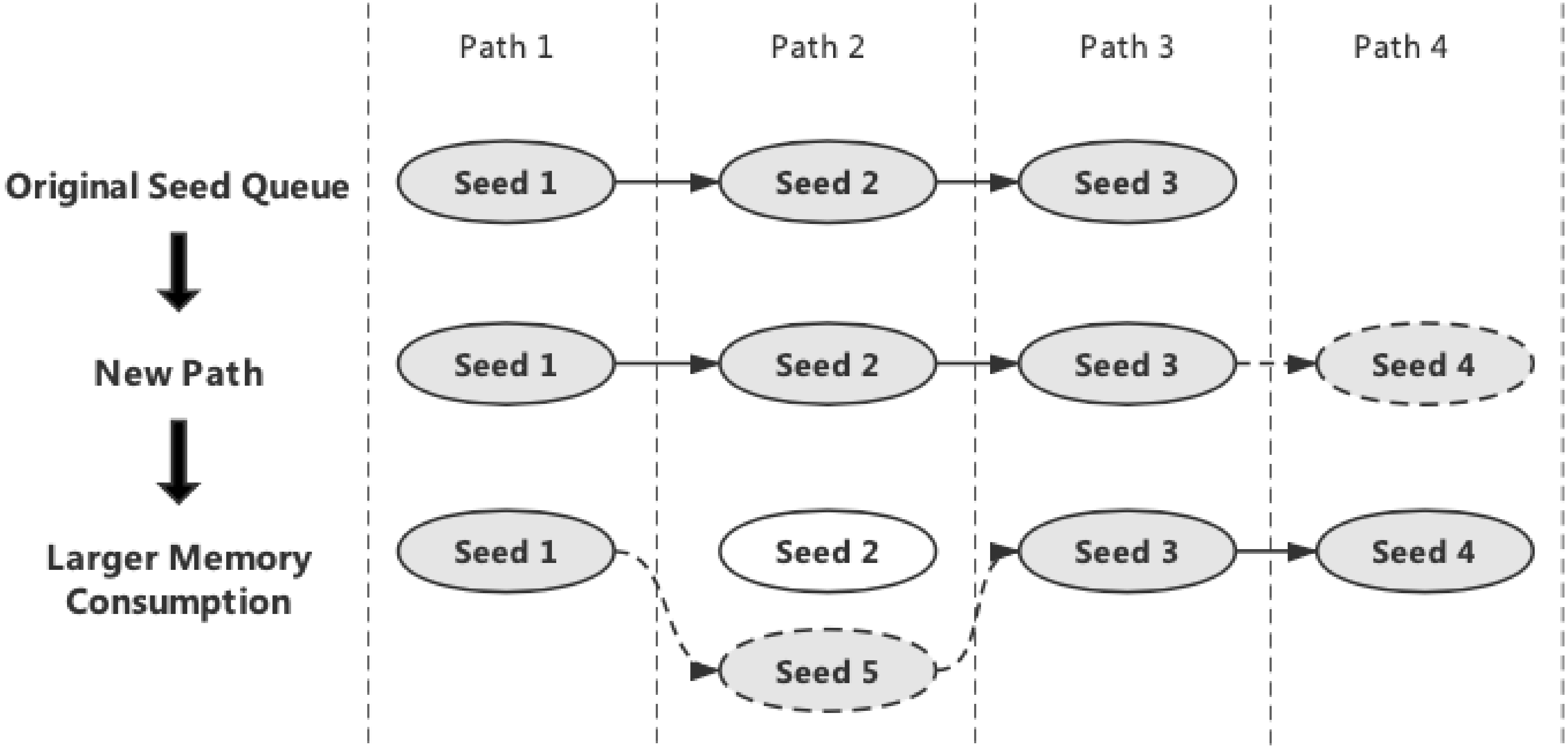
Approach

Memory information guidance

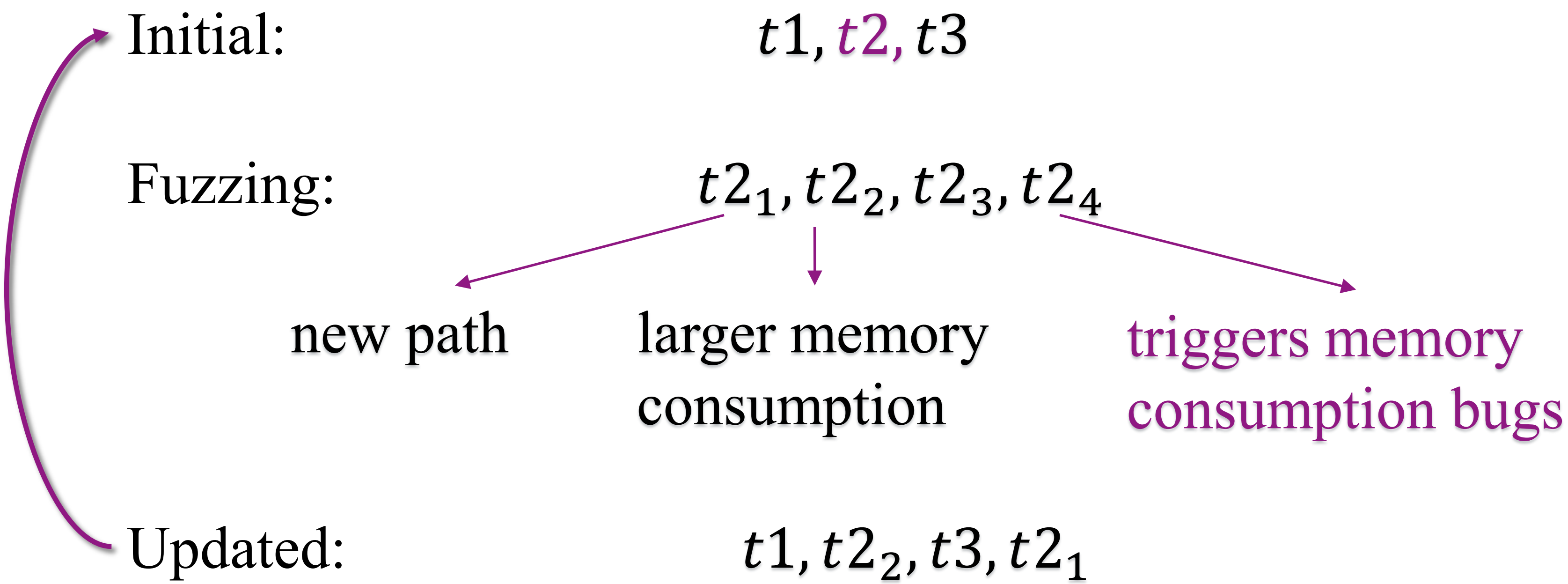
Key insight: lead to more memory consumption.

	Stack (Function) Memory	Heap (Operation) Memory
Instrumentation	Call/Return edge	Standard library
Max mem consumption for input t (execute path p)	fm_t	om_t
Max mem consumption	$fmmap[p] \leftarrow \max_{i \in I} fm_i$	$ommap[p] \leftarrow \max_{i \in I} om_i$
Update condition	$fm_t > fmmap[p] \parallel om_t > ommap[p]$	

Dynamic seed updating



Workflow



Initial: $t1, t2, t3$

Fuzzing: $t2_1, t2_2, t2_3, t2_4$

new path

larger memory consumption

triggers memory consumption bugs

Updated: $t1, t2_2, t3, t2_1$

Evaluation

Q1-Memory crashes

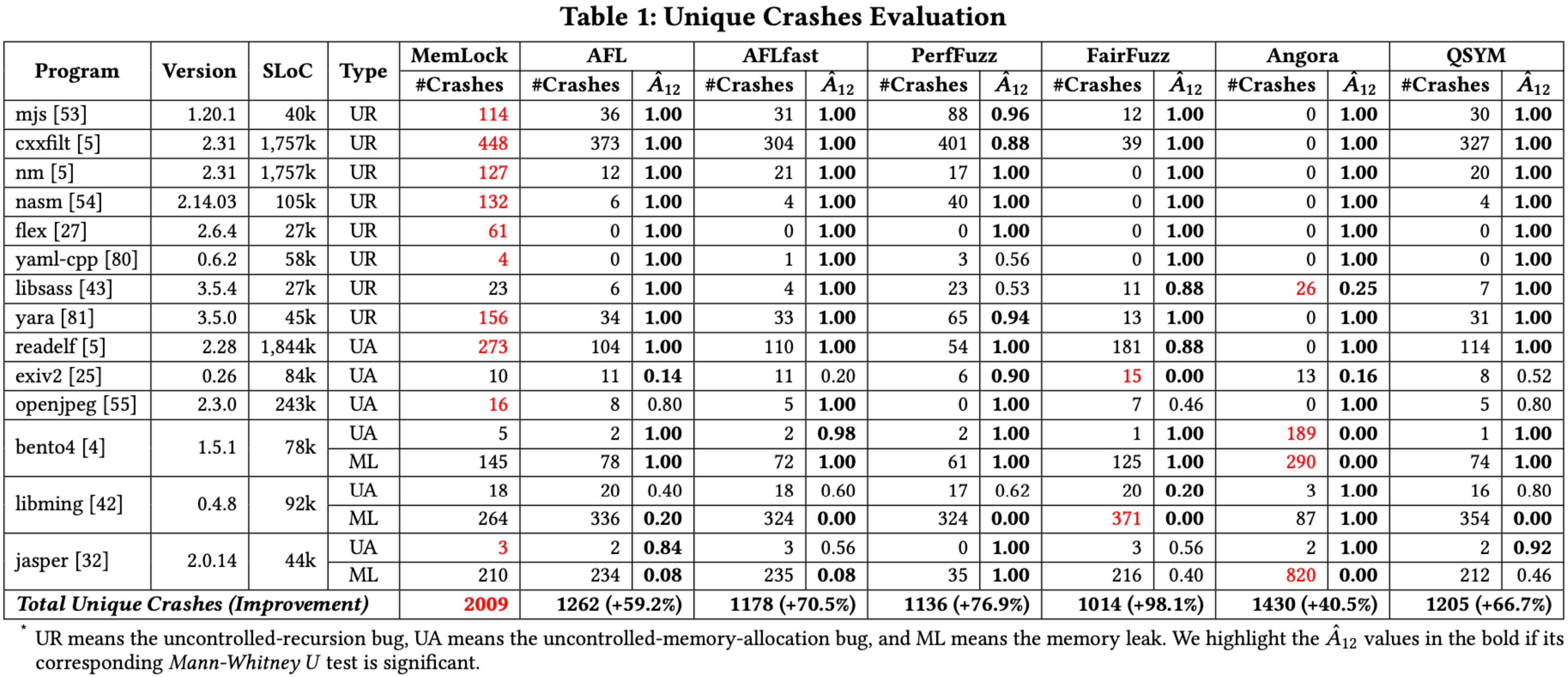


Table 1: Unique Crashes Evaluation

Program	Version	SLoC	Type	MemLock	AFL		AFLfast		PerfFuzz		FairFuzz		Angora		QSYM	
				#Crashes	#Crashes	$\hat{A}_{12}$	#Crashes	$\hat{A}_{12}$	#Crashes	$\hat{A}_{12}$	#Crashes	$\hat{A}_{12}$	#Crashes	$\hat{A}_{12}$	#Crashes	$\hat{A}_{12}$
mjs [53]	1.20.1	40k	UR	114	36	**1.00**	31	**1.00**	88	**0.96**	12	**1.00**	0	**1.00**	30	**1.00**
cxxfilt [5]	2.31	1,757k	UR	448	373	**1.00**	304	**1.00**	401	**0.88**	39	**1.00**	0	**1.00**	327	**1.00**
nm [5]	2.31	1,757k	UR	127	12	**1.00**	21	**1.00**	17	**1.00**	0	**1.00**	0	**1.00**	20	**1.00**
nasm [54]	2.14.03	105k	UR	132	6	**1.00**	4	**1.00**	40	**1.00**	0	**1.00**	0	**1.00**	4	**1.00**
flex [27]	2.6.4	27k	UR	61	0	**1.00**	0	**1.00**	0	**1.00**	0	**1.00**	0	**1.00**	0	**1.00**
yaml-cpp [80]	0.6.2	58k	UR	4	0	**1.00**	1	**1.00**	3	0.56	0	**1.00**	0	**1.00**	0	**1.00**
libsass [43]	3.5.4	27k	UR	23	6	**1.00**	4	**1.00**	23	0.53	11	**0.88**	26	**0.25**	7	**1.00**
yara [81]	3.5.0	45k	UR	156	34	**1.00**	33	**1.00**	65	**0.94**	13	**1.00**	0	**1.00**	31	**1.00**
readelf [5]	2.28	1,844k	UA	273	104	**1.00**	110	**1.00**	54	**1.00**	181	**0.88**	0	**1.00**	114	**1.00**
exiv2 [25]	0.26	84k	UA	10	11	**0.14**	11	0.20	6	**0.90**	15	**0.00**	13	**0.16**	8	0.52
openjpeg [55]	2.3.0	243k	UA	16	8	0.80	5	**1.00**	0	**1.00**	7	0.46	0	**1.00**	5	0.80
bento4 [4]	1.5.1	78k	UA	5	2	**1.00**	2	**0.98**	2	**1.00**	1	**1.00**	189	**0.00**	1	**1.00**
			ML	145	78	**1.00**	72	**1.00**	61	**1.00**	125	**1.00**	290	**0.00**	74	**1.00**
libming [42]	0.4.8	92k	UA	18	20	0.40	18	0.60	17	0.62	20	**0.20**	3	**1.00**	16	0.80
			ML	264	336	**0.20**	324	**0.00**	324	**0.00**	371	**0.00**	87	**1.00**	354	**0.00**
jasper [32]	2.0.14	44k	UA	3	2	**0.84**	3	0.56	0	**1.00**	3	0.56	2	**1.00**	2	**0.92**
			ML	210	234	**0.08**	235	**0.08**	35	**1.00**	216	0.40	820	**0.00**	212	0.46
Total Unique Crashes (Improvement)				**2009**	**1262 (+59.2%)**		**1178 (+70.5%)**		**1136 (+76.9%)**		**1014 (+98.1%)**		**1430 (+40.5%)**		**1205 (+66.7%)**	

* UR means the uncontrolled-recursion bug, UA means the uncontrolled-memory-allocation bug, and ML means the memory leak. We highlight the $\hat{A}_{12}$ values in the bold if its corresponding *Mann-Whitney U* test is significant.

Q2-Memory consumption vulnerability

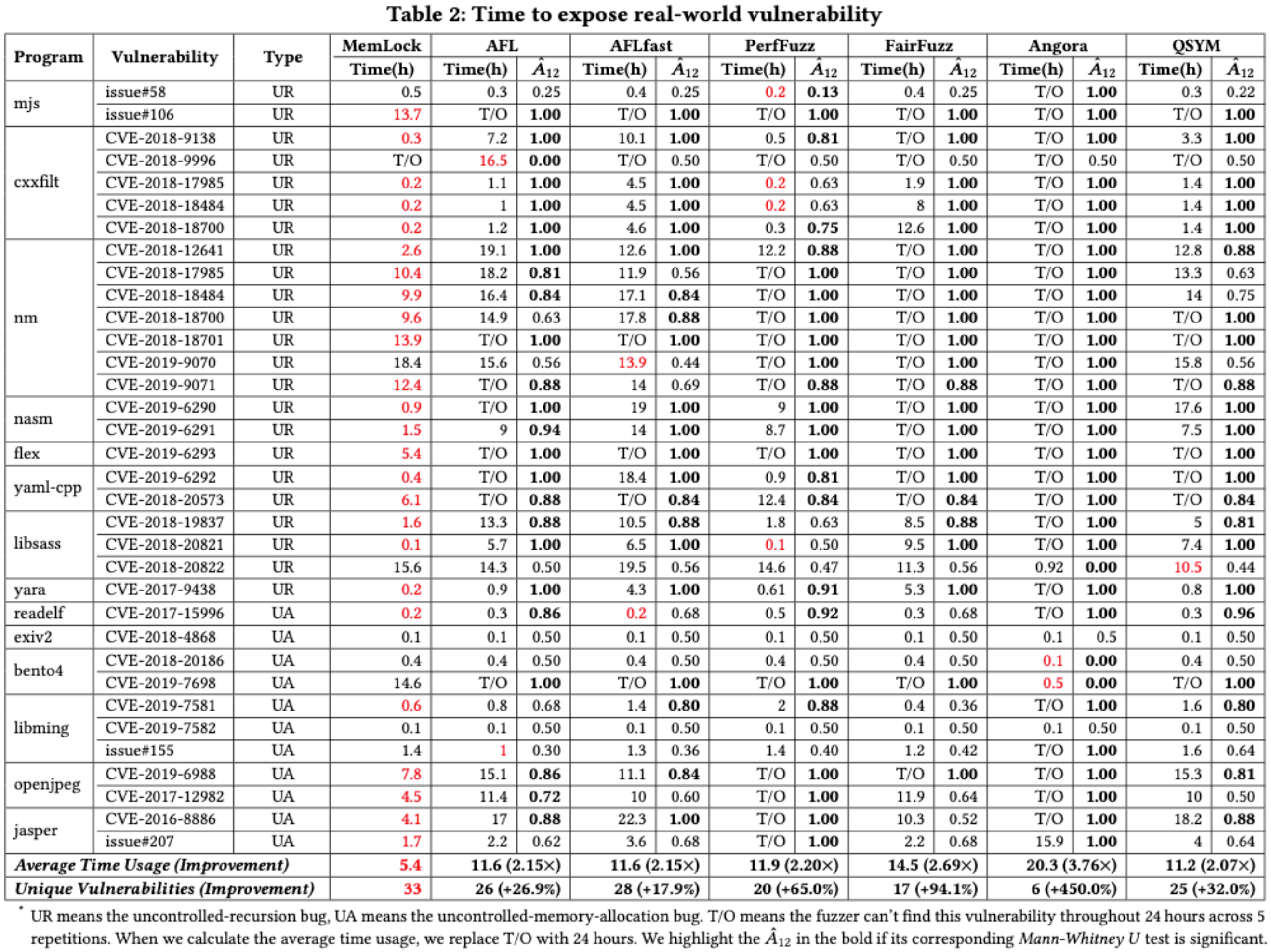


Table 2: Time to expose real-world vulnerability

Program	Vulnerability	Type	MemLock	AFL		AFLfast		PerfFuzz		FairFuzz		Angora		QSYM	
			Time(h)	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$	Time(h)	$\hat{A}_{12}$
mjs	issue#58	UR	0.5	0.3	0.25	0.4	0.25	0.2	**0.13**	0.4	0.25	T/O	**1.00**	0.3	0.22
	issue#106	UR	13.7	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
cxxfilt	CVE-2018-9138	UR	0.3	7.2	**1.00**	10.1	**1.00**	0.5	**0.81**	T/O	**1.00**	T/O	**1.00**	3.3	**1.00**
	CVE-2018-9996	UR	T/O	16.5	**0.00**	T/O	0.50	T/O	0.50	T/O	0.50	T/O	0.50	T/O	0.50
	CVE-2018-17985	UR	0.2	1.1	**1.00**	4.5	**1.00**	0.2	0.63	1.9	**1.00**	T/O	**1.00**	1.4	**1.00**
	CVE-2018-18484	UR	0.2	1	**1.00**	4.5	**1.00**	0.2	0.63	8	**1.00**	T/O	**1.00**	1.4	**1.00**
	CVE-2018-18700	UR	0.2	1.2	**1.00**	4.6	**1.00**	0.3	**0.75**	12.6	**1.00**	T/O	**1.00**	1.4	**1.00**
nm	CVE-2018-12641	UR	2.6	19.1	**1.00**	12.6	**1.00**	12.2	**0.88**	T/O	**1.00**	T/O	**1.00**	12.8	**0.88**
	CVE-2018-17985	UR	10.4	18.2	**0.81**	11.9	0.56	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	13.3	0.63
	CVE-2018-18484	UR	9.9	16.4	**0.84**	17.1	**0.84**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	14	0.75
	CVE-2018-18700	UR	9.6	14.9	0.63	17.8	**0.88**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
	CVE-2018-18701	UR	13.9	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
	CVE-2019-9070	UR	18.4	15.6	0.56	13.9	0.44	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	15.8	0.56
	CVE-2019-9071	UR	12.4	T/O	**0.88**	14	0.69	T/O	**0.88**	T/O	**0.88**	T/O	**1.00**	T/O	**0.88**
nasm	CVE-2019-6290	UR	0.9	T/O	**1.00**	19	**1.00**	9	**1.00**	T/O	**1.00**	T/O	**1.00**	17.6	**1.00**
	CVE-2019-6291	UR	1.5	9	**0.94**	14	**1.00**	8.7	**1.00**	T/O	**1.00**	T/O	**1.00**	7.5	**1.00**
flex	CVE-2019-6293	UR	5.4	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
yaml-cpp	CVE-2019-6292	UR	0.4	T/O	**1.00**	18.4	**1.00**	0.9	**0.81**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**
	CVE-2018-20573	UR	6.1	T/O	**0.88**	T/O	**0.84**	12.4	**0.84**	T/O	**0.84**	T/O	**1.00**	T/O	**0.84**
libsass	CVE-2018-19837	UR	1.6	13.3	**0.88**	10.5	**0.88**	1.8	0.63	8.5	**0.88**	T/O	**1.00**	5	**0.81**
	CVE-2018-20821	UR	0.1	5.7	**1.00**	6.5	**1.00**	0.1	0.50	9.5	**1.00**	T/O	**1.00**	7.4	**1.00**
	CVE-2018-20822	UR	15.6	14.3	0.50	19.5	0.56	14.6	0.47	11.3	0.56	0.92	**0.00**	10.5	0.44
yara	CVE-2017-9438	UR	0.2	0.9	**1.00**	4.3	**1.00**	0.61	**0.91**	5.3	**1.00**	T/O	**1.00**	0.8	**1.00**
readelf	CVE-2017-15996	UA	0.2	0.3	**0.86**	0.2	0.68	0.5	**0.92**	0.3	0.68	T/O	**1.00**	0.3	**0.96**
exiv2	CVE-2018-4868	UA	0.1	0.1	0.50	0.1	0.50	0.1	0.50	0.1	0.50	0.1	0.5	0.1	0.50
bento4	CVE-2018-20186	UA	0.4	0.4	0.50	0.4	0.50	0.4	0.50	0.4	0.50	0.1	**0.00**	0.4	0.50
	CVE-2019-7698	UA	14.6	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	0.5	**0.00**	T/O	**1.00**
libming	CVE-2019-7581	UA	0.6	0.8	0.68	1.4	**0.80**	2	**0.88**	0.4	0.36	T/O	**1.00**	1.6	**0.80**
	CVE-2019-7582	UA	0.1	0.1	0.50	0.1	0.50	0.1	0.50	0.1	0.50	0.1	0.50	0.1	0.50
	issue#155	UA	1.4	1	0.30	1.3	0.36	1.4	0.40	1.2	0.42	T/O	**1.00**	1.6	0.64
openjpeg	CVE-2019-6988	UA	7.8	15.1	**0.86**	11.1	**0.84**	T/O	**1.00**	T/O	**1.00**	T/O	**1.00**	15.3	**0.81**
	CVE-2017-12982	UA	4.5	11.4	**0.72**	10	0.60	T/O	**1.00**	11.9	0.64	T/O	**1.00**	10	0.50
jasper	CVE-2016-8886	UA	4.1	17	**0.88**	22.3	**1.00**	T/O	**1.00**	10.3	0.52	T/O	**1.00**	18.2	**0.88**
	issue#207	UA	1.7	2.2	0.62	3.6	0.68	T/O	**1.00**	2.2	0.68	15.9	**1.00**	4	0.64
Average Time Usage (Improvement)			**5.4**	**11.6 (2.15×)**		**11.6 (2.15×)**		**11.9 (2.20×)**		**14.5 (2.69×)**		**20.3 (3.76×)**		**11.2 (2.07×)**	
Unique Vulnerabilities (Improvement)			**33**	**26 (+26.9%)**		**28 (+17.9%)**		**20 (+65.0%)**		**17 (+94.1%)**		**6 (+450.0%)**		**25 (+32.0%)**	

* UR means the uncontrolled-recursion bug, UA means the uncontrolled-memory-allocation bug. T/O means the fuzzer can't find this vulnerability throughout 24 hours across 5 repetitions. When we calculate the average time usage, we replace T/O with 24 hours. We highlight the $\hat{A}_{12}$ in the bold if its corresponding *Mann-Whitney U* test is significant.

New vulnerabilities:

8 uncontrolled-recursion vulnerabilities

5 uncontrolled-memory-allocation

2 memory leak vulnerabilities

Q3-Memory leak evaluation

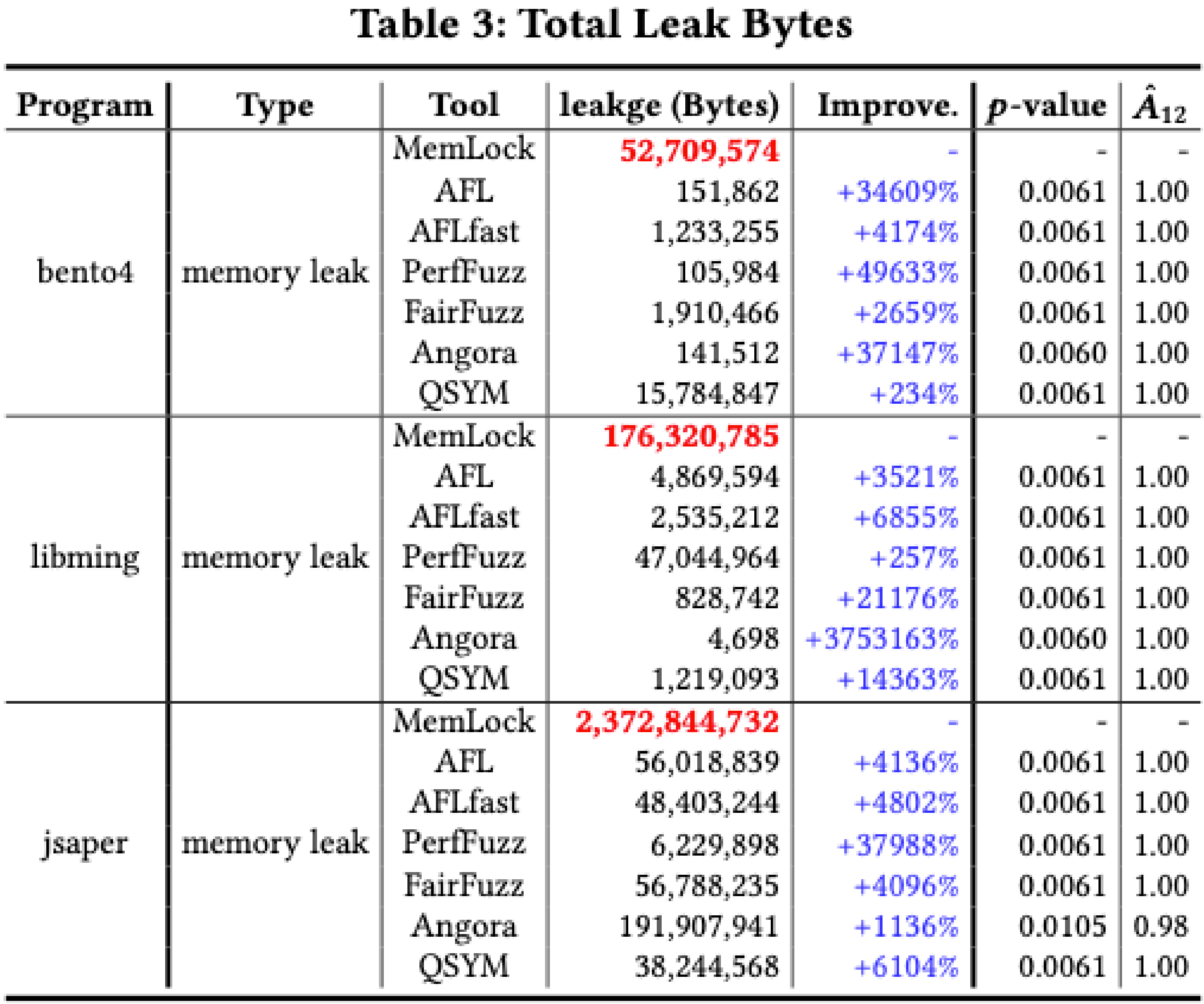


Table 3: Total Leak Bytes

Program	Type	Tool	leakge (Bytes)	Improve.	*p*-value	$\hat{A}_{12}$
bento4	memory leak	MemLock	**52,709,574**	-	-	-
		AFL	151,862	+34609%	0.0061	1.00
		AFLfast	1,233,255	+4174%	0.0061	1.00
		PerfFuzz	105,984	+49633%	0.0061	1.00
		FairFuzz	1,910,466	+2659%	0.0061	1.00
		Angora	141,512	+37147%	0.0060	1.00
		QSYM	15,784,847	+234%	0.0061	1.00
libming	memory leak	MemLock	**176,320,785**	-	-	-
		AFL	4,869,594	+3521%	0.0061	1.00
		AFLfast	2,535,212	+6855%	0.0061	1.00
		PerfFuzz	47,044,964	+257%	0.0061	1.00
		FairFuzz	828,742	+21176%	0.0061	1.00
		Angora	4,698	+3753163%	0.0060	1.00
		QSYM	1,219,093	+14363%	0.0061	1.00
jsaper	memory leak	MemLock	**2,372,844,732**	-	-	-
		AFL	56,018,839	+4136%	0.0061	1.00
		AFLfast	48,403,244	+4802%	0.0061	1.00
		PerfFuzz	6,229,898	+37988%	0.0061	1.00
		FairFuzz	56,788,235	+4096%	0.0061	1.00
		Angora	191,907,941	+1136%	0.0105	0.98
		QSYM	38,244,568	+6104%	0.0061	1.00

Q4-Memory consumption evaluation

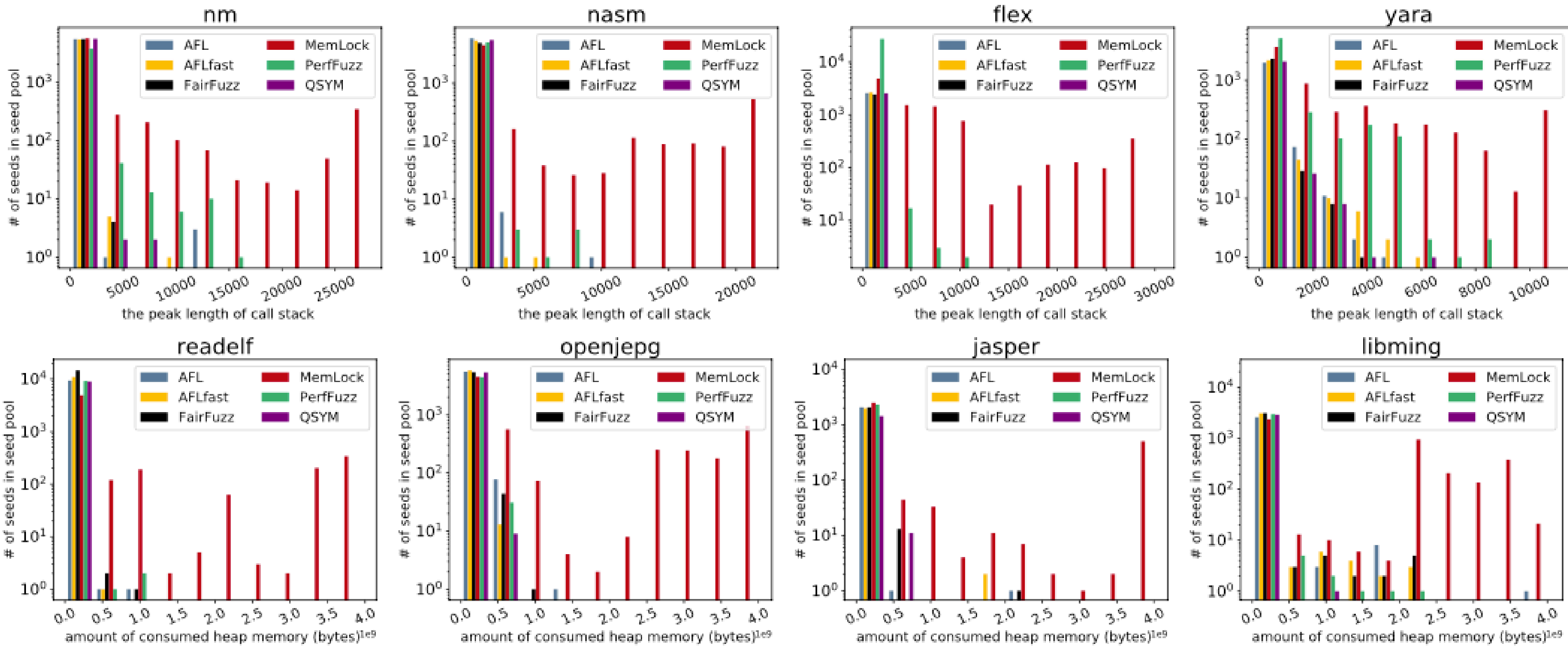


Figure 6: Seed distribution based on memory consumption. The larger the value on the right side is better.

Thanks