ISCAS 中国科学院软件研究所学术年会'2022
暨计算机科学国家重点实验室开放周

学术论文

InstruGuard: Find and Fix Instrumentation Errors for Coverage-based Greybox Fuzzing（ASE 2021）

（发现并修复基于覆盖率的灰盒模糊测试中的插桩错误）

Yuwei Liu†, Yanhao Wang †, Purui Su, Yuanping Yu and Xiangkun Jia*

作者联系方式：xiangkun@iscas.ac.cn，purui@iscas.ac.cn

TCA/SKLCS, Institute of Software, Chinese Academy of Sciences
QiAnXin Technology Research Institute

Fig1

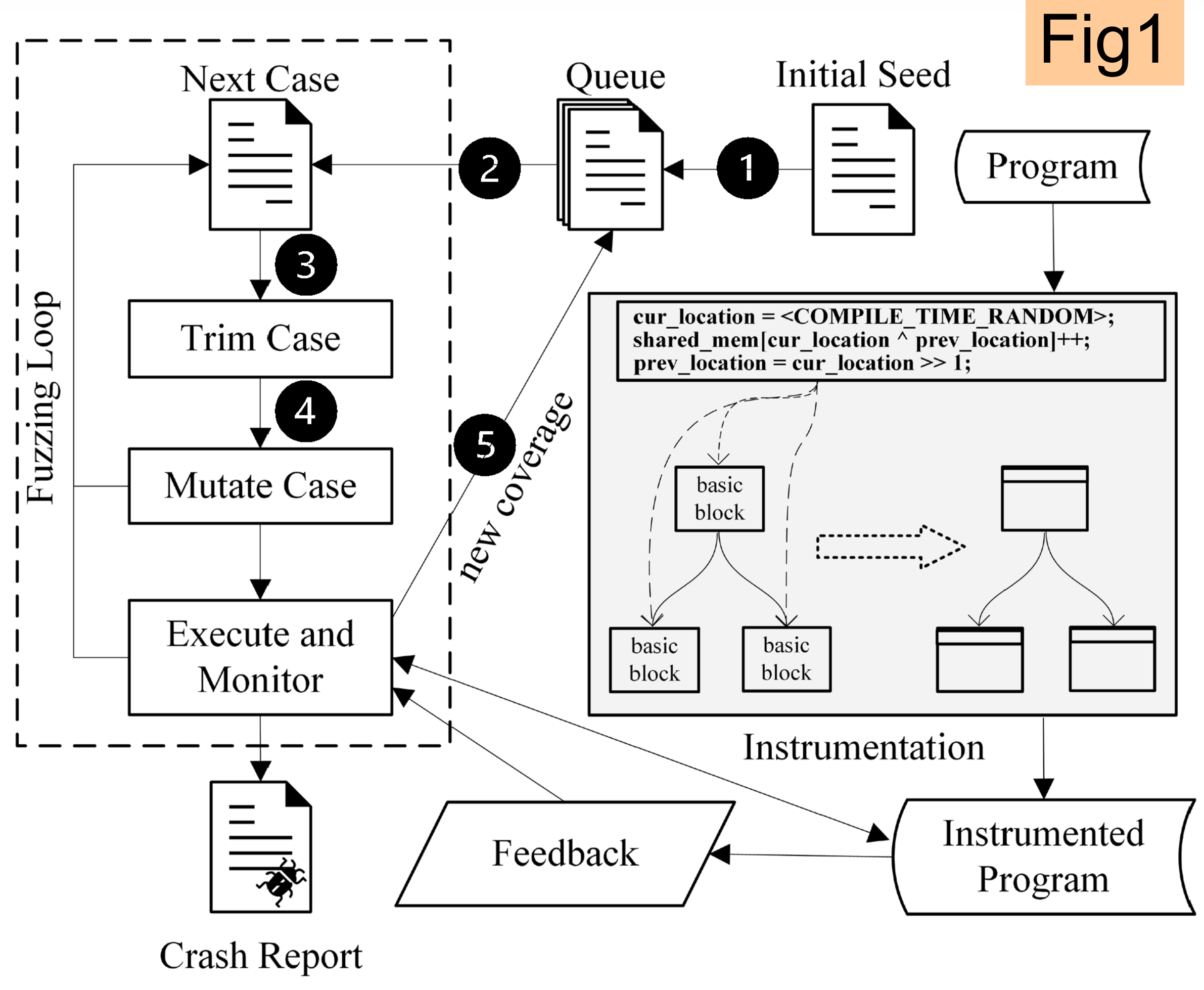
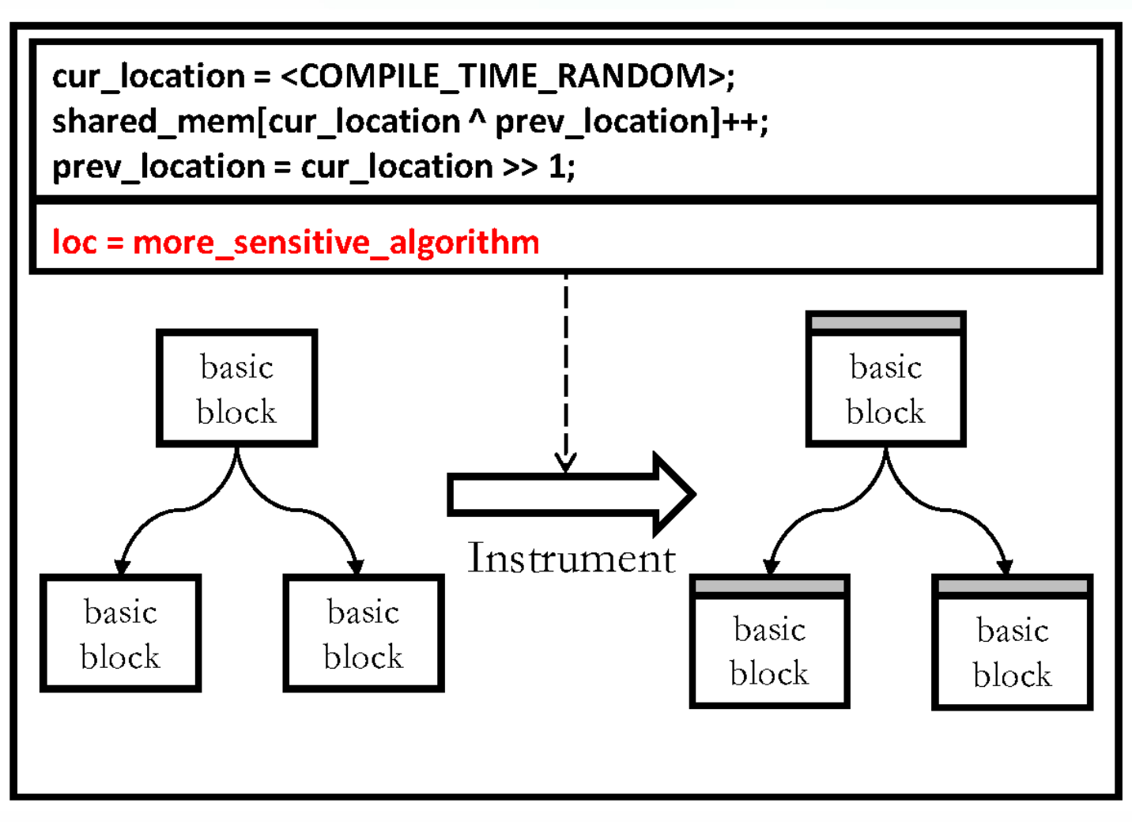


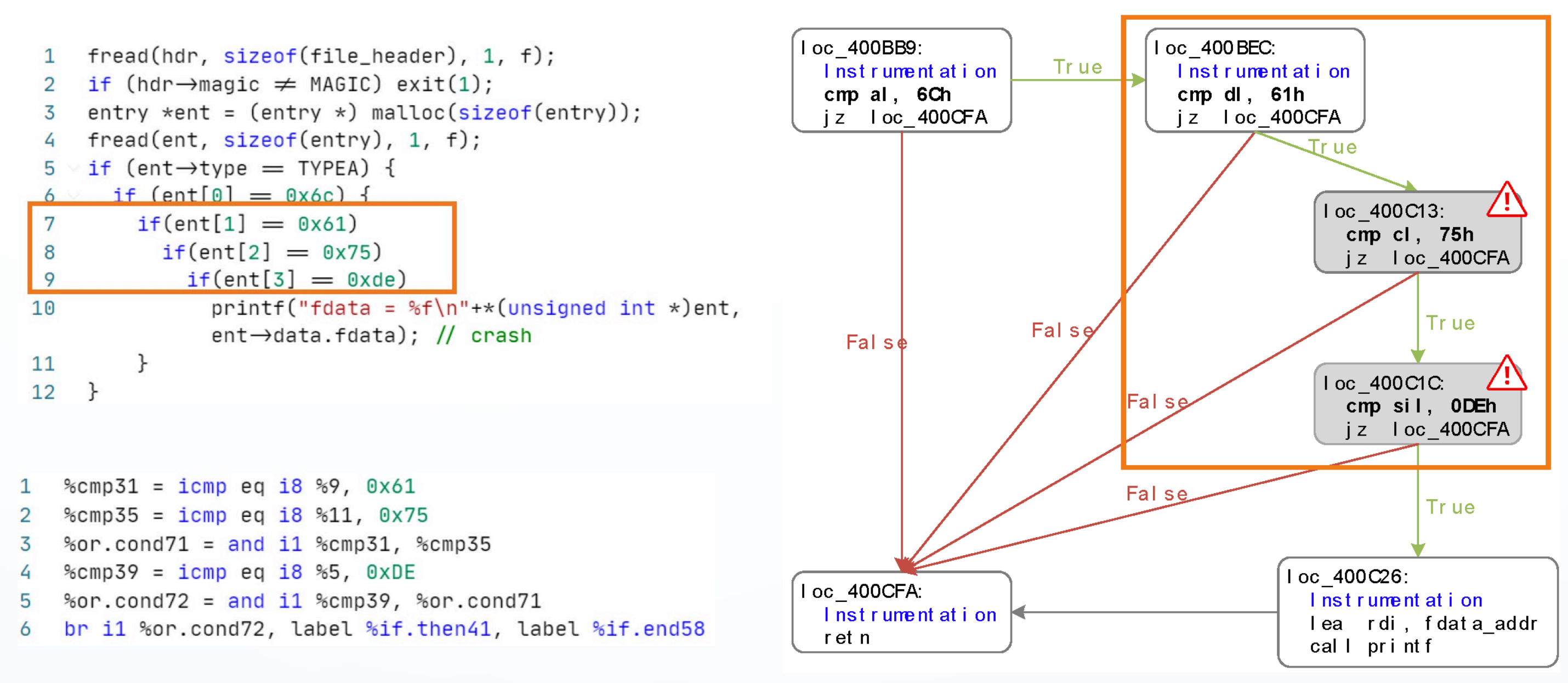
Fig2


cur_location = <COMPILE_TIME_RANDOM>;
shared_mem[cur_location ^ prev_location]++;
prev_location = cur_location >> 1;
loc = more_sensitive_algorithm
basic block
basic block
basic block
Instrument
basic block
basic block
basic block

- TortoiseFuzz
- MemLock
- MemFuzz
- IJON
- Angora
- Parmesan
- AFL-Sensitive
- ……

AFL等基于覆盖率的灰盒模糊测试工具依赖于代码插桩技术获取覆盖率反馈，进而指导测试循环中种子的选择和变异（Fig1）。同时，许多改进工作也利用了代码插桩获取更多的程序信息（Fig 2）。

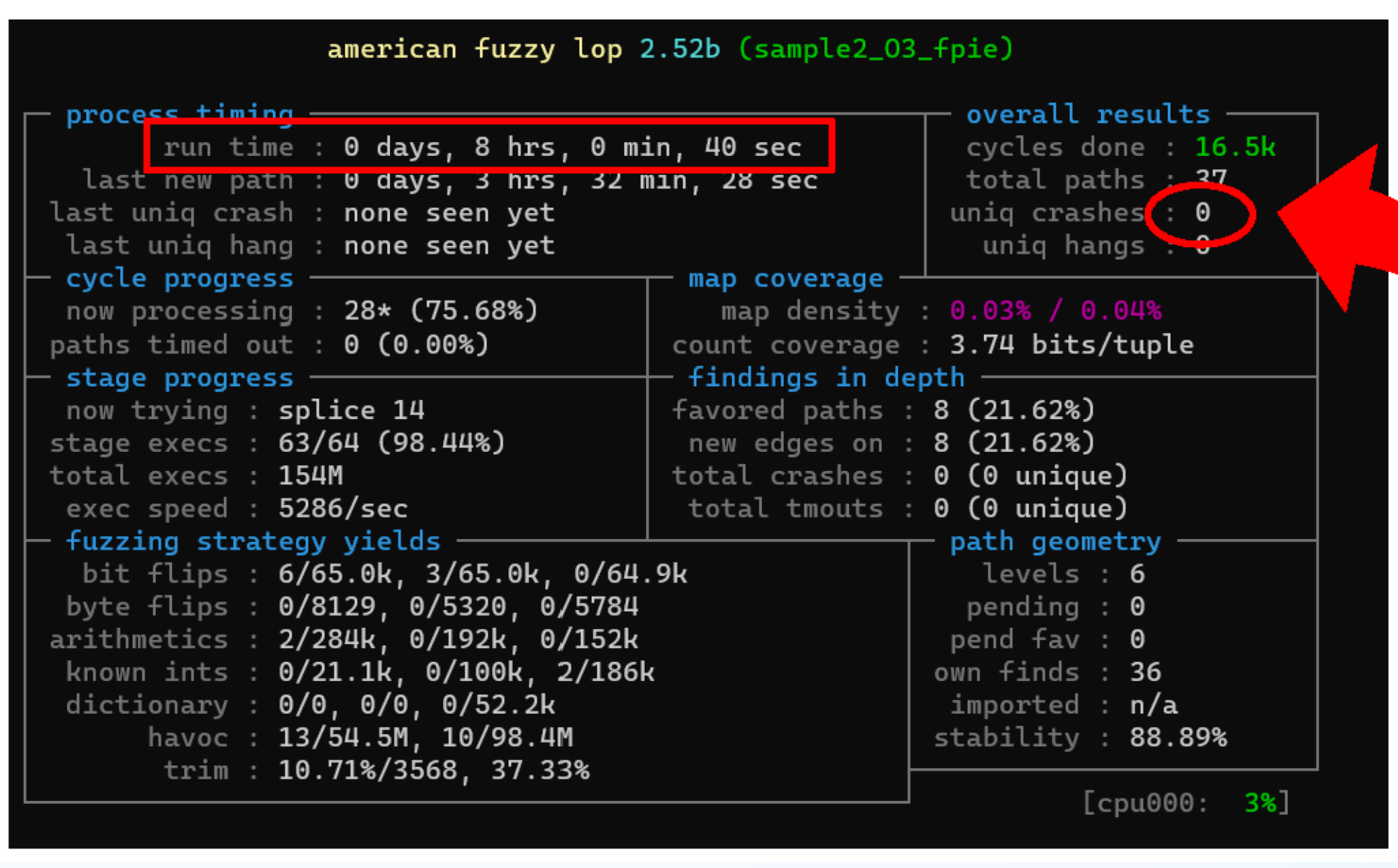
Fig3



```
fread(hdr, sizeof(file_header), 1, f);
if (hdr→magic ≠ MAGIC) exit(1);
entry *ent = (entry *) malloc(sizeof(entry));
fread(ent, sizeof(entry), 1, f);
if (ent→type = TYPEA) {
  if (ent[0] = 0x6c) {
    if(ent[1] = 0x61)
      if(ent[2] = 0x75)
        if(ent[3] = 0xde)
          printf("fdata = %f\n"+*(unsigned int *)ent,
          ent→data.fdata); // crash
    }
}
```

```
%cmp31 = icmp eq i8 %9, 0x61
%cmp35 = icmp eq i8 %11, 0x75
%or.cond71 = and i1 %cmp31, %cmp35
%cmp39 = icmp eq i8 %5, 0xDE
%or.cond72 = and i1 %cmp39, %or.cond71
br i1 %or.cond72, label %if.then41, label %if.end58
```

Fig4

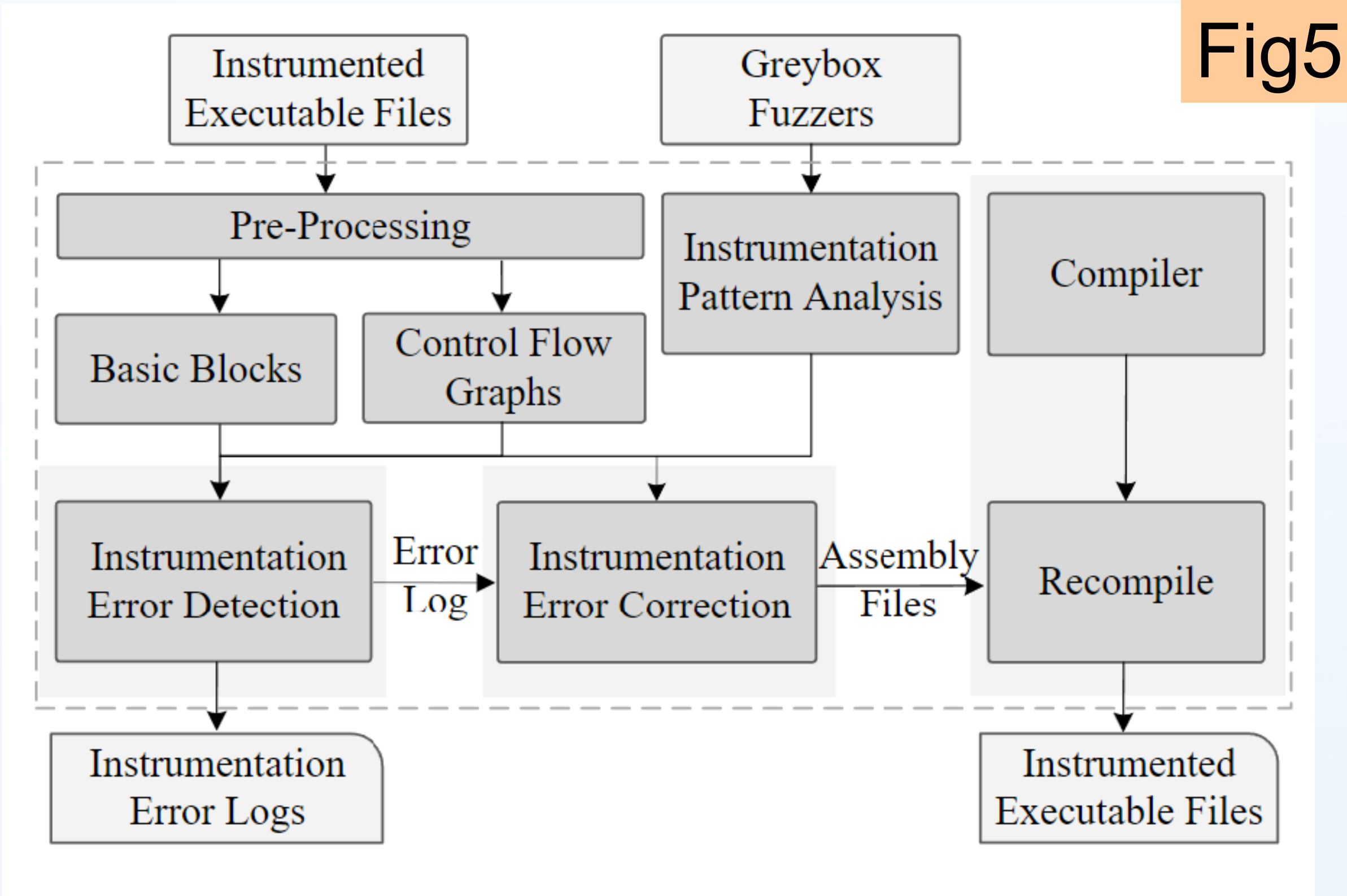


```
                american fuzzy lop 2.52b (sample2_03_fpie)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 8 hrs, 0 min, 40 sec       │  cycles done : 16.5k  │
│   last new path : 0 days, 3 hrs, 32 min, 28 sec      │  total paths : 37     │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : none seen yet                      │   uniq hangs : 0      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 28* (75.68%)      │    map density : 0.03% / 0.04%         │
│ paths timed out : 0 (0.00%)         │ count coverage : 3.74 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : splice 14             │ favored paths : 8 (21.62%)             │
│ stage execs : 63/64 (98.44%)        │  new edges on : 8 (21.62%)             │
│ total execs : 154M                  │ total crashes : 0 (0 unique)           │
│  exec speed : 5286/sec              │  total tmouts : 0 (0 unique)           │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 6/65.0k, 3/65.0k, 0/64.9k             │    levels : 6          │
│  byte flips : 0/8129, 0/5320, 0/5784                │   pending : 0          │
│ arithmetics : 2/284k, 0/192k, 0/152k                │  pend fav : 0          │
│  known ints : 0/21.1k, 0/100k, 2/186k               │ own finds : 36         │
│  dictionary : 0/0, 0/0, 0/52.2k                     │  imported : n/a        │
│       havoc : 13/54.5M, 10/98.4M                    │ stability : 88.89%     │
│        trim : 10.71%/3568, 37.33%                   ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu000:  3%]
```

然而，AFL等模糊测试工具的源代码插桩过程中，会因为基本块拆分等问题，导致本应该插在每一个基本块头部的分析代码，最终只插在了拆分后的第一个基本块（Fig 3）。进一步导致无法为模糊测试提供准确有效的反馈，而影响漏洞的发现（Fig4 ）。

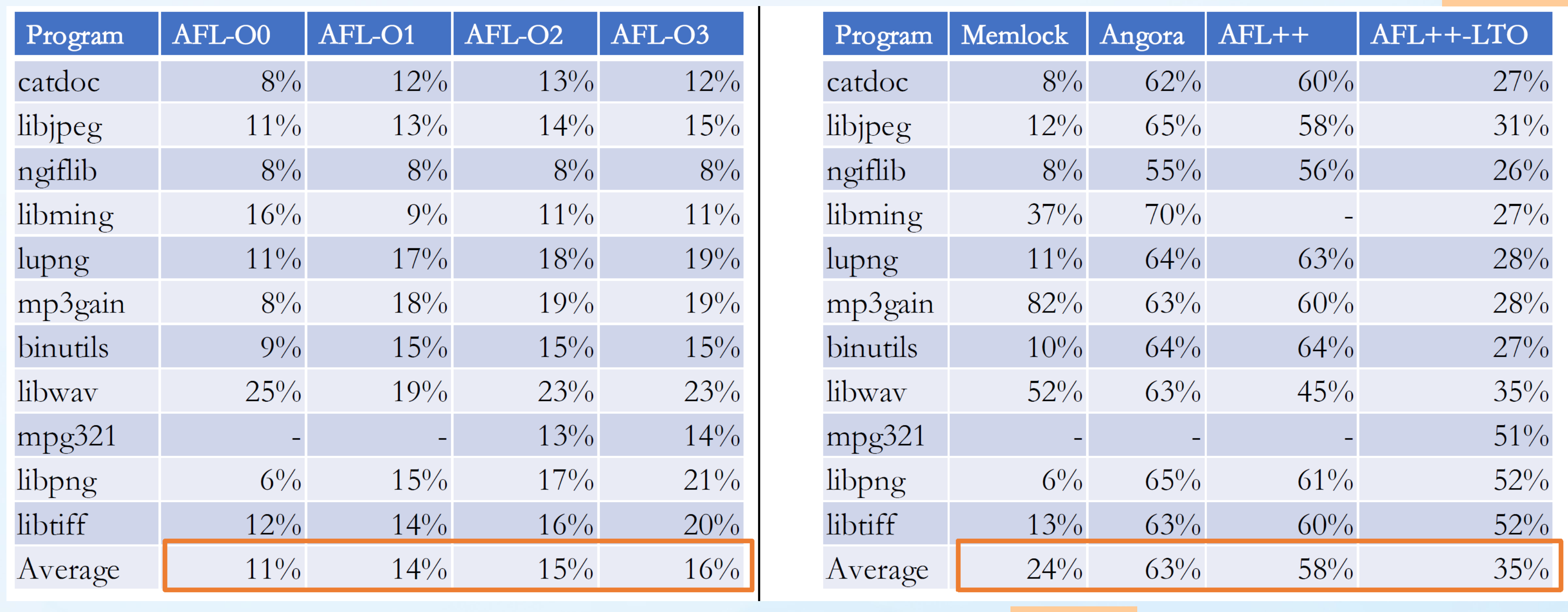
Fig5


Instrumented Executable Files
Greybox Fuzzers
Pre-Processing
Instrumentation Pattern Analysis
Compiler
Basic Blocks
Control Flow Graphs
Instrumentation Error Detection
Error Log
Instrumentation Error Correction
Assembly Files
Recompile
Instrumentation Error Logs
Instrumented Executable Files

因此，本文提出了**InstruGuard**，发现并修复插桩错误（Fig 5）。

发现：借助模糊测试插入的覆盖率反馈分析代码的二进制特征；
修复：借助二进制代码重写工具RetroWrite

Fig6



Program	AFL-O0	AFL-O1	AFL-O2	AFL-O3
catdoc	8%	12%	13%	12%
libjpeg	11%	13%	14%	15%
ngiflib	8%	8%	8%	8%
libming	16%	9%	11%	11%
lupng	11%	17%	18%	19%
mp3gain	8%	18%	19%	19%
binutils	9%	15%	15%	15%
libwav	25%	19%	23%	23%
mpg321	-	-	13%	14%
libpng	6%	15%	17%	21%
libtiff	12%	14%	16%	20%
Average	11%	14%	15%	16%

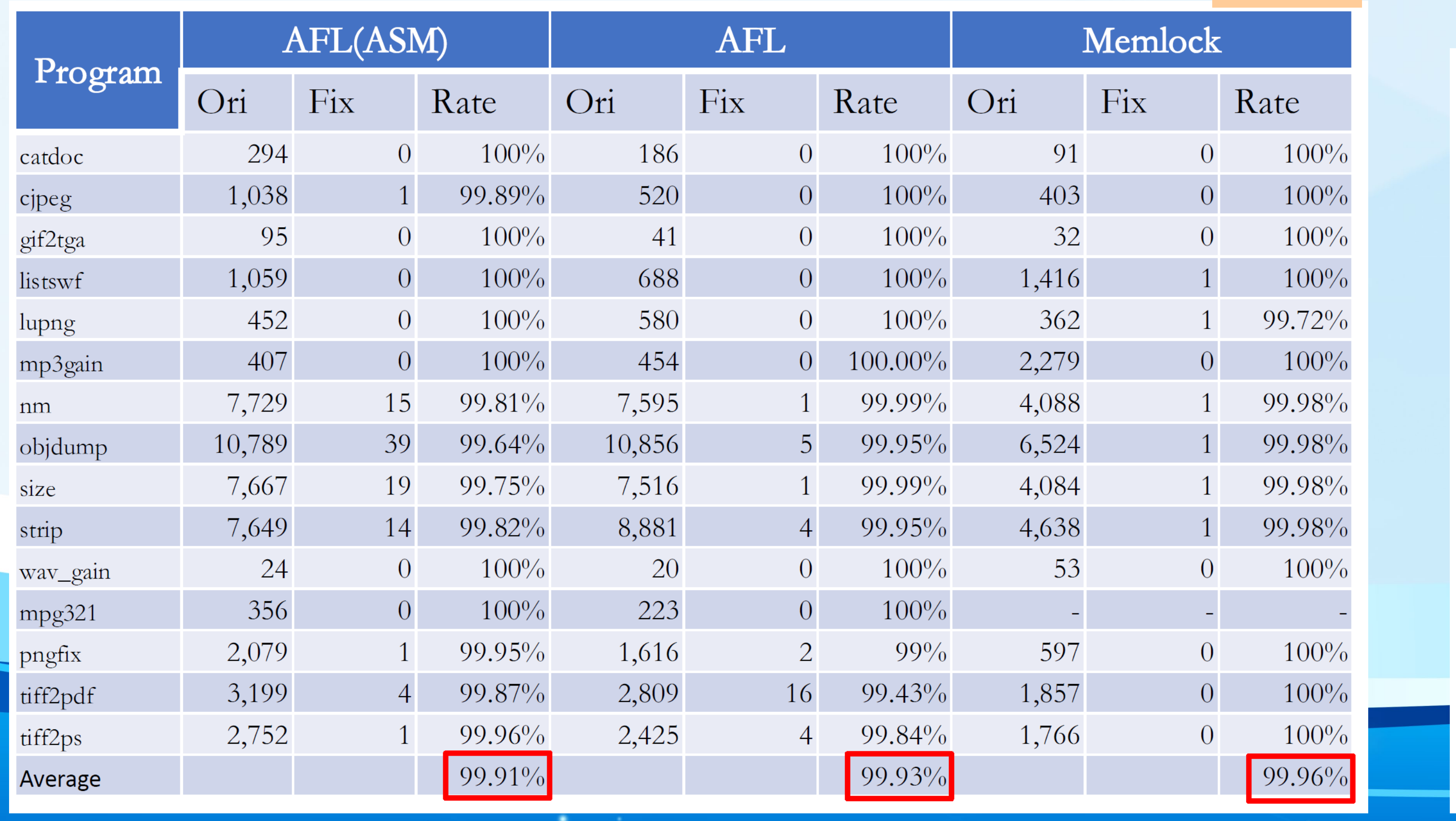
Program	Memlock	Angora	AFL++	AFL++-LTO
catdoc	8%	62%	60%	27%
libjpeg	12%	65%	58%	31%
ngiflib	8%	55%	56%	26%
libming	37%	70%	-	27%
lupng	11%	64%	63%	28%
mp3gain	82%	63%	60%	28%
binutils	10%	64%	64%	27%
libwav	52%	63%	45%	35%
mpg321	-	-	-	51%
libpng	6%	65%	61%	52%
libtiff	13%	63%	60%	52%
Average	24%	63%	58%	35%

实验发现：
插桩错误在不同的编译优化选项下和不同的灰盒模糊测试工具中是**普遍存在**的（Fig 6）。

InstruGuard能够修复错误的插桩，修复率达**99.9%**（Fig 7）。

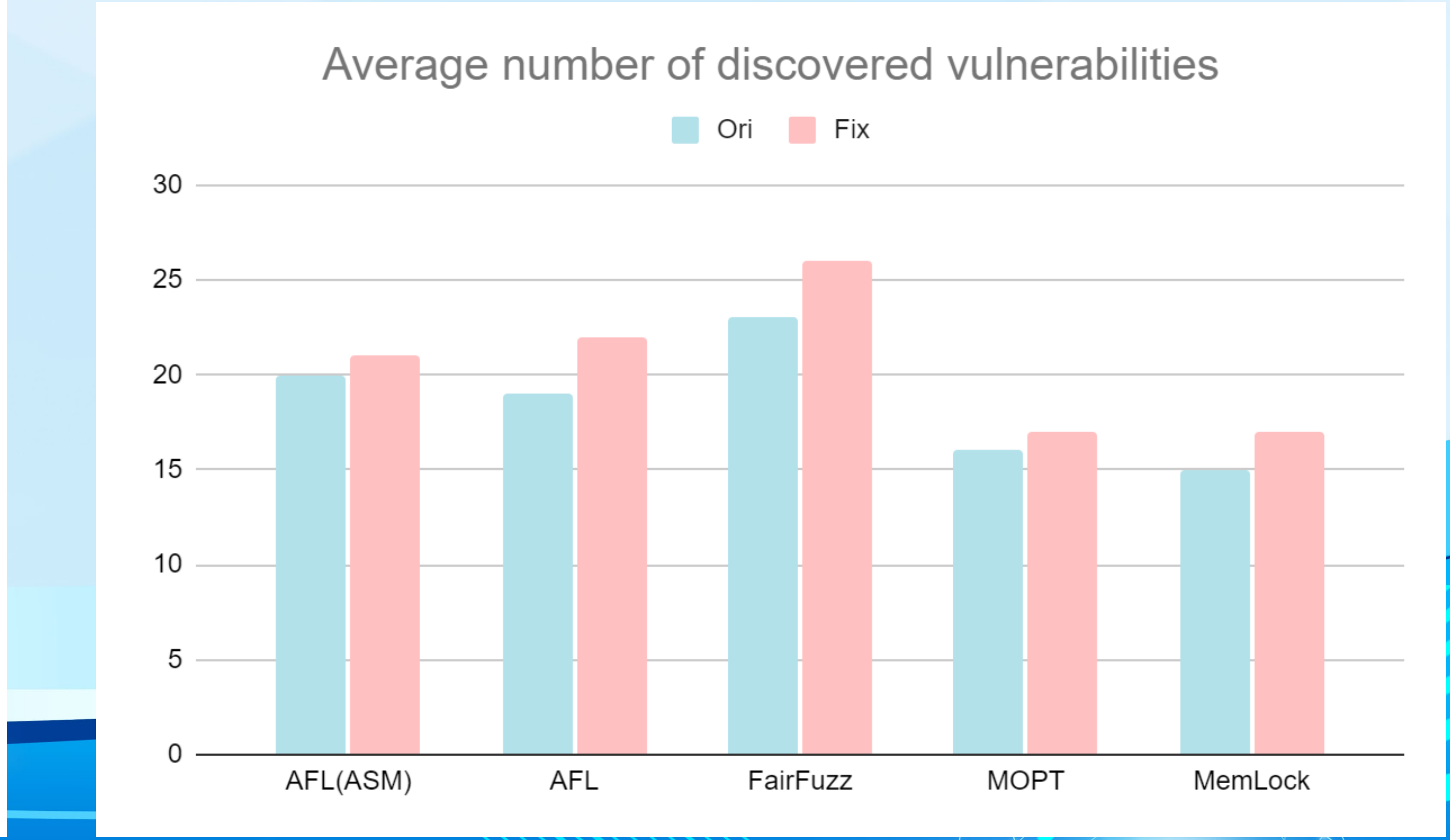
修复后的程序在漏洞挖掘的能力上表现更好（Fig 8）。

Fig7



Program	AFL(ASM)			AFL			Memlock		
	Ori	Fix	Rate	Ori	Fix	Rate	Ori	Fix	Rate
catdoc	294	0	100%	186	0	100%	91	0	100%
cjpeg	1,038	1	99.89%	520	0	100%	403	0	100%
gif2tga	95	0	100%	41	0	100%	32	0	100%
listswf	1,059	0	100%	688	0	100%	1,416	1	100%
lupng	452	0	100%	580	0	100%	362	1	99.72%
mp3gain	407	0	100%	454	0	100.00%	2,279	0	100%
nm	7,729	15	99.81%	7,595	1	99.99%	4,088	1	99.98%
objdump	10,789	39	99.64%	10,856	5	99.95%	6,524	1	99.98%
size	7,667	19	99.75%	7,516	1	99.99%	4,084	1	99.98%
strip	7,649	14	99.82%	8,881	4	99.95%	4,638	1	99.98%
wav_gain	24	0	100%	20	0	100%	53	0	100%
mpg321	356	0	100%	223	0	100%	-	-	-
pngfix	2,079	1	99.95%	1,616	2	99%	597	0	100%
tiff2pdf	3,199	4	99.87%	2,809	16	99.43%	1,857	0	100%
tiff2ps	2,752	1	99.96%	2,425	4	99.84%	1,766	0	100%
Average			99.91%			99.93%			99.96%

Fig8


Average number of discovered vulnerabilities
Ori
Fix
AFL(ASM)
AFL
FairFuzz
MOPT
MemLock