



DDGF: Dynamic Directed Greybox Fuzzing with Path Profiling

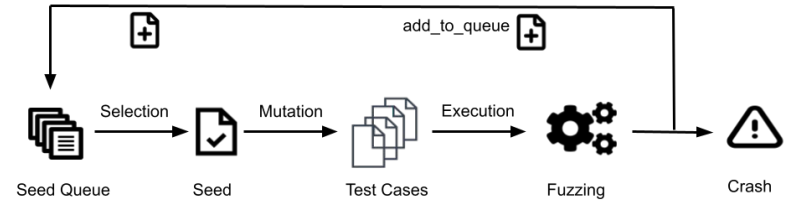
Haoran Fang, Kaikai Zhang, Donghui Yu, Yuanyuan Zhang*

Shanghai Jiao Tong University

International Symposium on Software Testing and Analysis (ISSTA 2024)

Coverage-Guided Fuzzing

- Evolutionary Algorithm
 - simple but effective
- AFL++/Syzkaller/Fuzzilli...
 - general & domain-specific
- Large-Scale Industrial Practice
 - OSS-Fuzz, OneFuzz ..
 - 24/7 continuous fuzzing
 - over 36,000 bugs across 1,000 projects

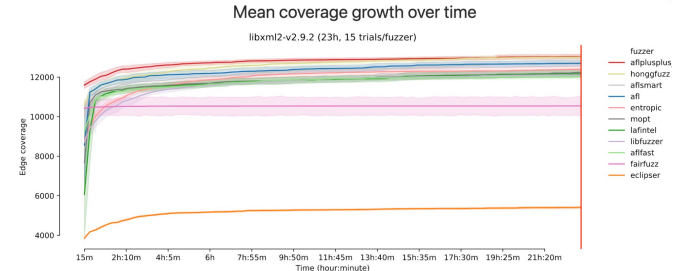


Fuzzing is NOT a Black Box

- Start fuzzing, wait for the results
 - crash -> replay
- Coverage Plateau
 - Is current test sufficient ?
 - Where is it stuck?
- More Intermediate Status Exposure
 - understand -> direct

```
american fuzzy lop ++2.65d (libpng_harness) [explore] (0)
process timing | overall results
run time : 0 days, 0 hrs, 0 min, 43 sec | cycles done : 15
last new path : 0 days, 0 hrs, 0 min, 1 sec | total paths : 703
last uniq crash : none seen yet | uniq crashes : 0
last uniq hang : none seen yet | uniq hangs : 0
cycle progress | map coverage
now processing : 261*1 (37.1%) | map density : 5.78% / 13.98%
paths timed out : 0 (0.00%) | counts coverage : 3.30 bits/tuple
stage progress | findings in depth
now trying : splice 14 | favored paths : 114 (16.22%)
stage execs : 31/32 (96.88%) | new edges on : 107 (23.76%)
total execs : 2.55M | total crashes : 0 (0 unique)
exec speed : 61.2k/sec | total tmouts : 0 (0 unique)
fuzzing strategy yields | path geometry
bit flips : n/a, n/a, n/a | levels : 11
byte flips : n/a, n/a, n/a | pending : 121
arithmetic : n/a, n/a, n/a | pend fav : 0
known ints : n/a, n/a, n/a | own finds : 699
dictionary : n/a, n/a, n/a | imported : n/a
havoc/splice : 506/1.05M, 193/1.44M | stability : 99.88%
py/custom : 0/0, 0/0
trim : 19.25%/53.2k, n/a | [cpu000: 12%]
```

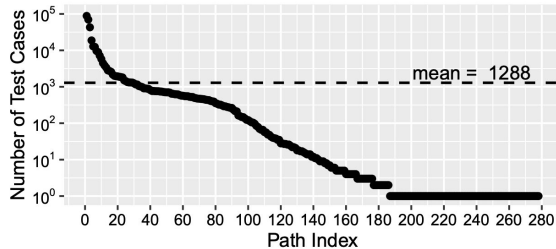
AFL status screen



coverage plateau for libxml2

Motivation

```
1 int wavlike_read_fmt_chunk (SF_PRIVATE *psf, int fmtsize) {
2     ...
3     switch (wav_fmt->format) {
4         case WAVE_FORMAT_IMA_ADPCM:
5             ...
6             break;
7         case WAVE_FORMAT_GSM610:
8             ...
9             break;
10        case WAVE_FORMAT_EXTENSIBLE:
11            // heap overflow (commit a8ab5b3)
12            // how to focus this case in fuzzing?
13        }
14    }
```



- Directed Greybox Fuzzing
 - distance-based
 - recompile to change targets
 - smooth and slow
 - distinguish different cases?
- Communication
 - bitmap in AFL
 - one-way mapping
 - AFLFast long tail map
 - cksum of tracebit
 - decode Index ?

Research Questions*

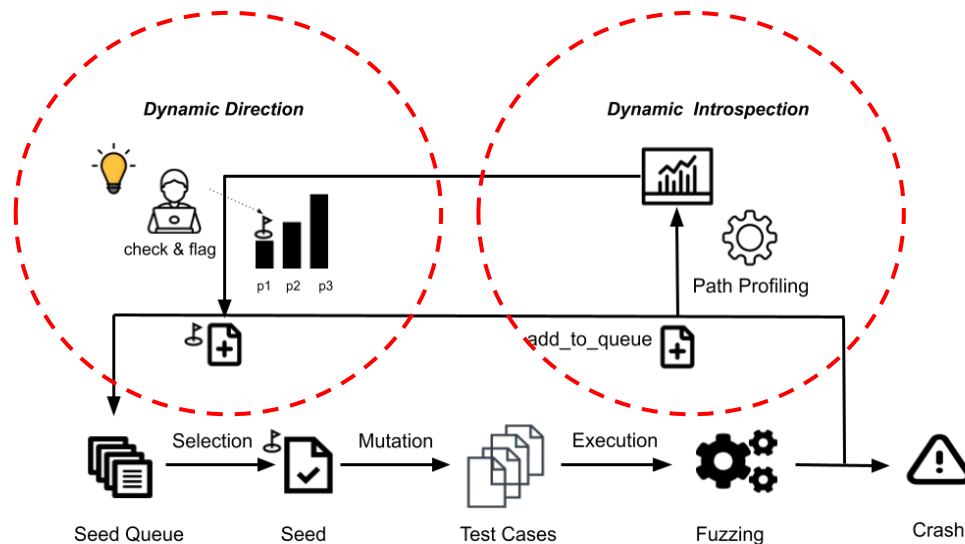


RQ1. How can the fuzzer explain what prevents it from progressing ?

RQ2. How can we facilitate a more effective communication between fuzzer and security auditor ?

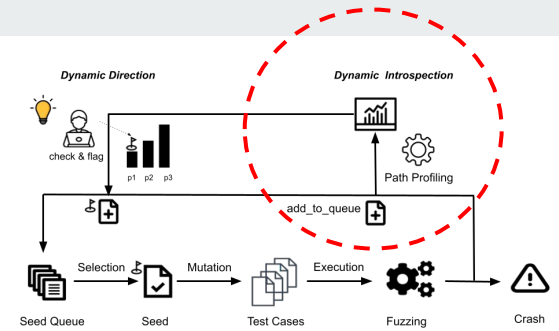
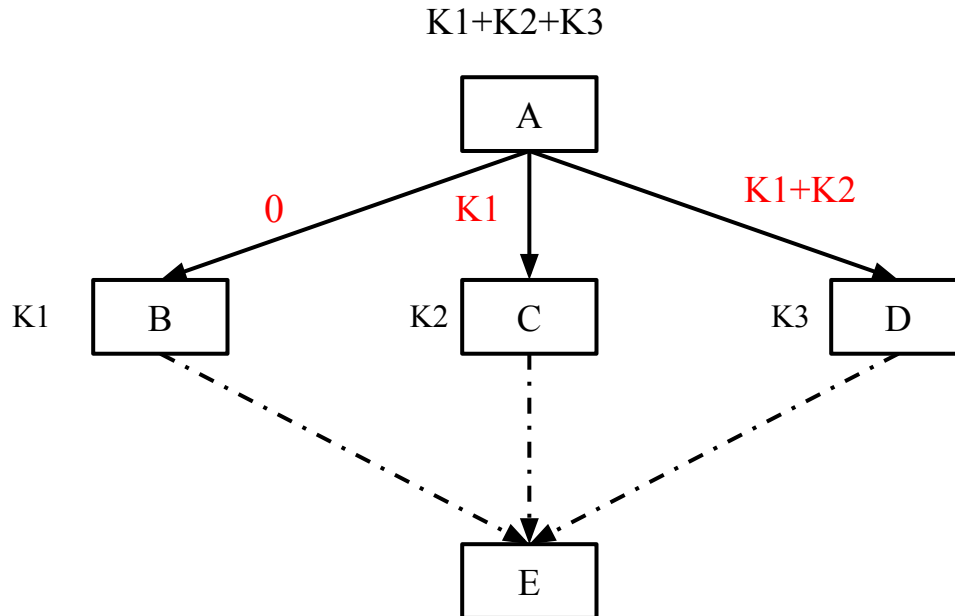
* "Fuzzing: Challenges and Reflections," in *IEEE Software*, vol. 38, no. 3, pp. 79-86, May-June 2021

Design Overview



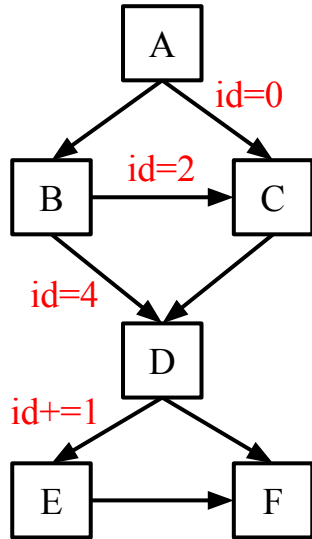
- Share the Same View of Code
 - how to describe a path?
 - Ball-Larus path encoding/profiling
- Interactive Way
 - decode & check paths
 - add flags for target path
- Performance Overhead
 - trade-off
 - profiling for seeds

Ball-Larus Path Profiling*



*Thomas Ball and James R. Larus. Efficient path profiling. (MICRO 1996).

Ball-Larus Path Profiling



Path	Encoding
ACDF	0
ACDEF	1
ABCDF	2
ABCDE F	3
ABDF	4
ABDEF	5

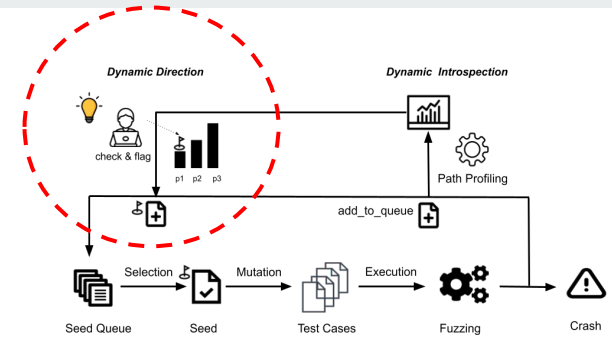
- Encoding
 - unique number(id)
 - reverse topological order
 - spanning tree & instrument
- Decoding
 - path \leftrightarrow id \Rightarrow hit_count
 - user check & flag
 - connection

count[id]++

flag[id] = True/False?

Path Flagging

- check & decode the path frequency
 - insights from introspection
- add/change the flag for target paths
 - at any time of fuzzing
 - 3 flagging strategies
- flags
 - connection between user and fuzzer
 - influence on seed selection



Seed Selection

- Intuition

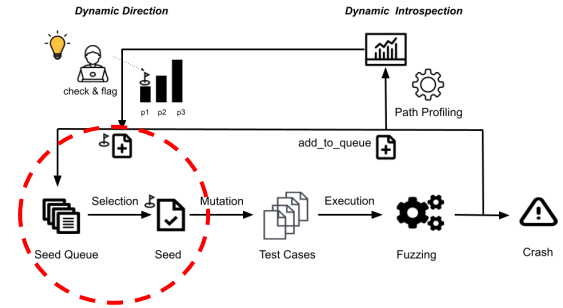
- seeds with flag tend to produce testcases with flag
- flags(feature) are saved in evolution

- Interactive Evolutionary Algorithm

- old research topic
- fitness function + human evaluation

- Back to Motivating Example

- target path/cases



```
1 int wavlike_read_fmt_chunk (SF_PRIVATE *psf, int fmsize) {
2     ...
3     switch (wav_fmt->format) {
4         case WAVE_FORMAT_IMA_ADPCM:
5             ...
6             break;
7         case WAVE_FORMAT_GSM610:
8             ...
9             break;
10        case WAVE_FORMAT_EXTENSIBLE:
11            // heap overflow (commit a8ab5b3)
12            // how to focus this case in fuzzing?
13        }
14    }
```

Implementation

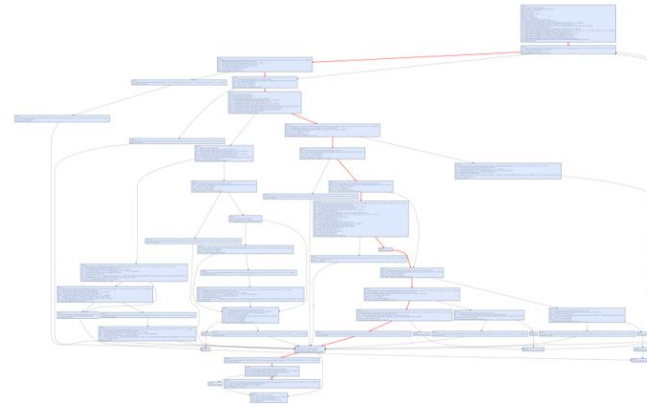


Dashboard

Implementation

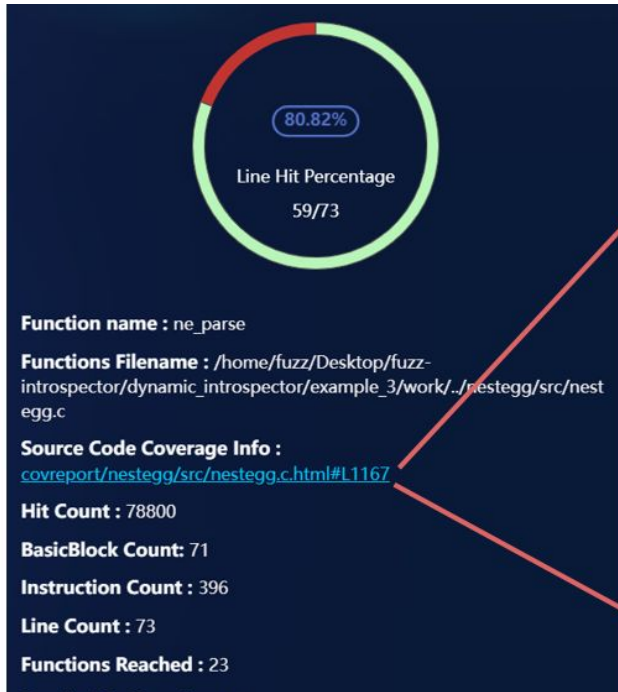


```
Function :  
ne_parse  
Total Path Num :  
347  
Current Path :  
block 1 -> block 2 -> block 3 -> block 4 -> block 8  
-> block 9 -> block 11 -> block 13 -> block 14 ->  
block 16 -> block 17 -> block 18 -> block 19 ->  
block 20 -> block 47 -> block 48 -> block 49 ->  
block 50
```



Path Frequency & Decoding in CFG

Implementation

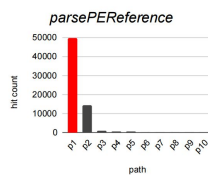


```
1169 78.8k int64_t * data_offset;
1170 78.8k uint64_t id, size, peeked_id;
1171 78.8k struct ebml_element_desc * element;
1172
1173 78.8k assert(ctx->ancestor);
1174
1175 2.72M for (..) {
1176 2.72M     if (max_offset > 0 && ne_io_tell(ctx->io) == max_offset) {
1177         /* Reached end of offset allowed for parsing - return gracefully */
1178         r = 1;
1179         break;
1180     }
1181 2.72M     r = ne_peek_element(ctx, &id, &size);
1182 2.72M     if (r != 1)
1183         break;
1184 2.72M     peeked_id = id;
1185
1186 2.72M     element = ne_find_element(id, ctx->ancestor->node);
1187 2.72M     if (element) {
1188         1.81M         if (element->flags & DESC_FLAG_SUSPEND) {
1189             24.0k         assert(element->id == ID_CLUSTER && element->type == TYPE_MASTER);
1190             24.0k         ctx->log(ctx, NESTEGG_LOG_ERROR, "suspend parse at %lix", id);
1191             24.0k         r = 1;
1192             24.0k         break;
1193             24.0k         }
1194
1195 1.79M         r = ne_read_element(ctx, &id, &size);
1196 1.79M         if (r != 1)
1197             break;
1198 1.79M         assert(id == peeked_id);
1199
1200 1.79M         if (element->flags & DESC_FLAG_OFFSET) {
1201             69.1k         data_offset = (int64_t *) (ctx->ancestor->data + element->data_offset);
1202             69.1k         *data_offset = ne_io_tell(ctx->io);
1203             69.1k         if (*data_offset < 0) {
1204                 r = -1;
1205             }
1206         }
1207     }
1208 }
```

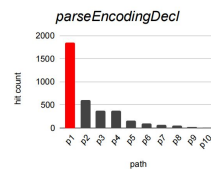
Line Coverage , Combined with Fuzz-Introspector(Illvm-cov)

Evaluation

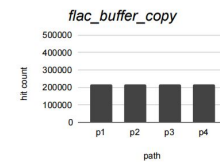
- Magma benchmark & real programs
- Dynamic Introspection
 - insight ?
 - typical long-tail shape
 - huge imbalance
 - 20% paths , 80% hits
 - decode top-2 paths
 - blocking path



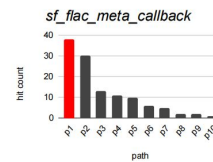
(a) XML003



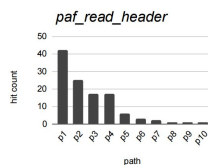
(b) XML009



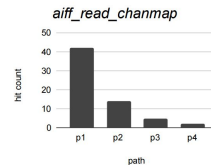
(c) SND006



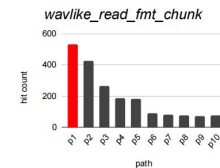
(d) SND007/024



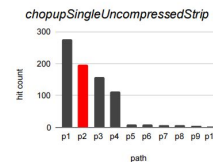
(e) SND001



(f) SND005



(g) SND017/020



(h) TIF007/014

Blocking paths are highlighted in red.

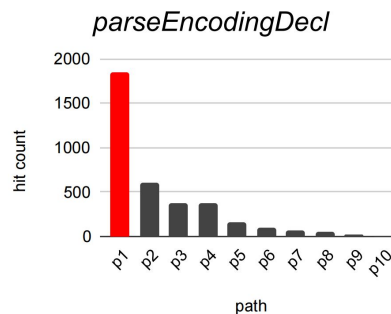
Fuzzing wasted too much resources on these paths.

Evaluation

Listing 2: Code Snippet for Illustrating Path Frequency

```
1 const xmlChar *
2 xmlParseEncodingDecl(xmlParserCtxtPtr ctxt) {
3     xmlChar *encoding = NULL;
4     if (CMP8(CUR_PTR,
5         'e','n','c','o','d','i','n','g')) {
6         // ....
7         // MAGMA(XML009)
8     }
9     return ;
10 }
```

String Checking



(b) XML009

Rejected by first *if* statement

Evaluation

- Dynamic Direction
 - up to 100x~ speedup
 - only 1 flag: TIF008-10x
- Flagging Strategies
 - #1 Exclude the Blocking Path
 - #2 Identify the Key Control-Flow
 - #3 Identify the Key Data-Flow

Bug ID	AFL++	DDGF	Speedup	#Flags	Strategy
SND001	7.41m	4s	>100x	13	
SND005	15.22m	2s	>400x	4	
SND006	39.30m	16s	>140x	4	
SND007	20.01m	13s	>90x	16	1
SND017	13.48m	2.21m	6x	62	
SND020	30.25m	3.19m	10x	62	
SND024	14.39m	6s	>140x	10	
TIF012	4.39m	0.98m	4.5x	1	2
TIF014	4.29m	2.11m	2x	11	
TIF014-cp	34.24m	6.89m	5x	11	1
XML003	12.71m	7.15m	1.7x	22	
XML009	11.8m	2.28m	5x	22	
PNG007	1.62h	0.49h	3.3x	2	
TIF002	13.39h	4.35h	3.0x	1	2
TIF008	19.88h	1.67h	11.9x	1	
SQL002	12.4m	10.6m	1.1x	3	
SQL014	2.07h	0.73h	2.8x	21	1
SQL018	1.31h	0.76h	1.7x	8	2
SQL020	10.0h	2.0h	5x	3	3
SSL020	17.6h	1.86h	9.5x	10	2
PDF003	1.86h	11.5m	9.7x	6	
PDF010	2.13h	2.29m	55x	36	1
PDF019	T.O	16.99h	>1.4x	110	
PDF018	19.67h	42s	>1600x	1	2
PDF021	T.O	T.O	-	6	

Strategy 1 : Exclude the Blocking Path



Listing 3: Code Snippet for Flagging Strategy 1

```
1 sf_flac_meta_callback () {  
2     switch (metadata->type){  
3         case FLAC_METADATA_TYPE_STREAM_INFO:  
4             ...  
5             // MAGMA_LOG(SND007)  
6             // MAGMA_LOG(SND024)  
7             break;  
8         case FLAC_METADATA_TYPE_UNDEFINED:  
9             break;  
10    }  
11 }
```

Exclude top blocking path(TYPE_UNDEFINED), and add flags for all remaining paths.

TTE acceleration: 20min ⇒ 13s

Strategy 2 : Identify the Key Control-Flow

Listing 4: Code Snippet for Flagging Strategy 2

```
1 static int NeXTDecode(TIFF* tif, ..) {
2     switch (n) {
3         case 1: ..break
4         case 2: ..break
5         ...
6         default: // MAGMA_LOG(TIF008)
7     }
8 }
9
10 int TIFFInitNeXT(TIFF* tif, int scheme) {
11     (void) scheme;
12     tif->tif_predecode = NeXTPreDecode;
13     tif->tif_decoderow = NeXTDecode;
14     tif->tif_decodestrip = NeXTDecode;
15     tif->tif_decodetile = NeXTDecode;
16     return (1);
17 }
```

6,000 Seeds, 0 Hits for the first 15h.

What hinders the triggering?

function pointer initialization

only 6 Hits , no more than 6/6000 seeds

Add only 1 flag for this key control-flow path .

TTE acceleration: 19.88h ⇒ 1.67h

Strategy 3 : Identify the Key Data-Flow

- data-flow constraint
- focus on the key def-use chain
- TTE : 10h ⇒ 2h

Listing 5: Code Snippet for Flagging Strategy 3

```
1 static ExprList *exprListAppendList (...) {
2     // MAGMA_LOG(SQL020);
3 }
4
5 int sqlite3WindowRewrite(Select *p, ...) {
6     // pWin use, data-flow constraint
7     if (p -> pWin && ...) {
8         pSort = exprListAppendList (...);
9     }
10 }
11
12 Window* sqlite3WindowAlloc(Parse *pParse, ...) {
13     // pWin def
14     pWin = (Window*)sqlite3DbMallocZero(pParse->db, sizeof(
15         Window));
16 }
17 YYACTIONTYPE yy_reduce(..) {
18     switch(yyruleno) {
19         case 315:
20         case 316:
21         case 317: sqlite3WindowAlloc(pParse, ...)
22         ...
23     }
24 }
```

use

def

Performance Overhead



Table 3: Comparison of Runtime Overhead between DDGF and AFL++.

Program	AFL++	DDGF	Overhead
sndfile_fuzzer	6.52M	6.11M	6%
tiff_read_rgb_fuzzer	101M	89.9M	11%
sqlite3_fuzzer	41.2M	37.5M	9%
xml_read_memory_fuzzer	22.3M	17.6M	21%
pdf_fuzzer	209k	173k	19%
libpng_read_fuzzer	312M	281M	10%

Total # executions for 2 Hours



Thanks