DDGF: Dynamic Directed Greybox Fuzzing with Path Profiling

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

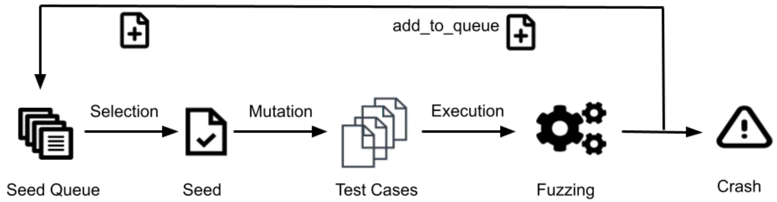
Shanghai Jiao Tong University

International Symposium on Software Testing and Analysis (ISSTA 2024)

ISSTA 24

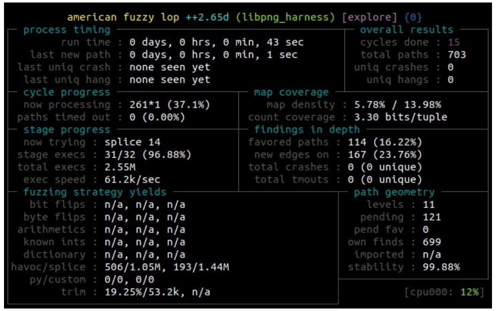
Coverage-Guided Fuzzing

- 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

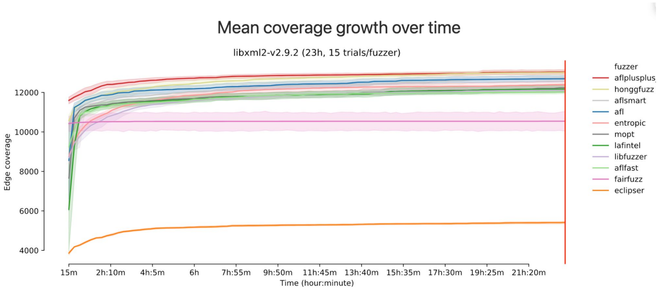


Fuzzing is NOT a Black Box

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



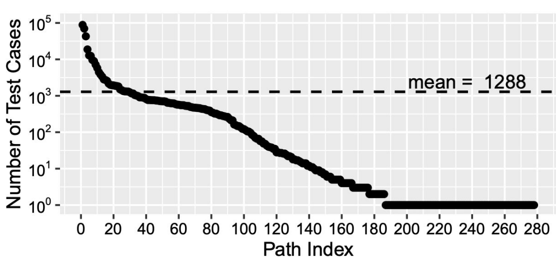
AFL status screen



coverage plateau for libxml2

Motivation

```
int wavlike_read_fmt_chunk (SF_PRIVATE *psf, int fmtsize) {
    ...
    switch (wav_fmt->format) {
        case WAVE_FORMAT_IMA_ADPCM:
            ...
            break;
        case WAVE_FORMAT_GSM610:
            ...
            break;
        case WAVE_FORMAT_EXTENSIBLE:
            // heap overflow (commit a8ab5b3)
            // how to focus this case in fuzzing?
    }
}
```



- 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 ?

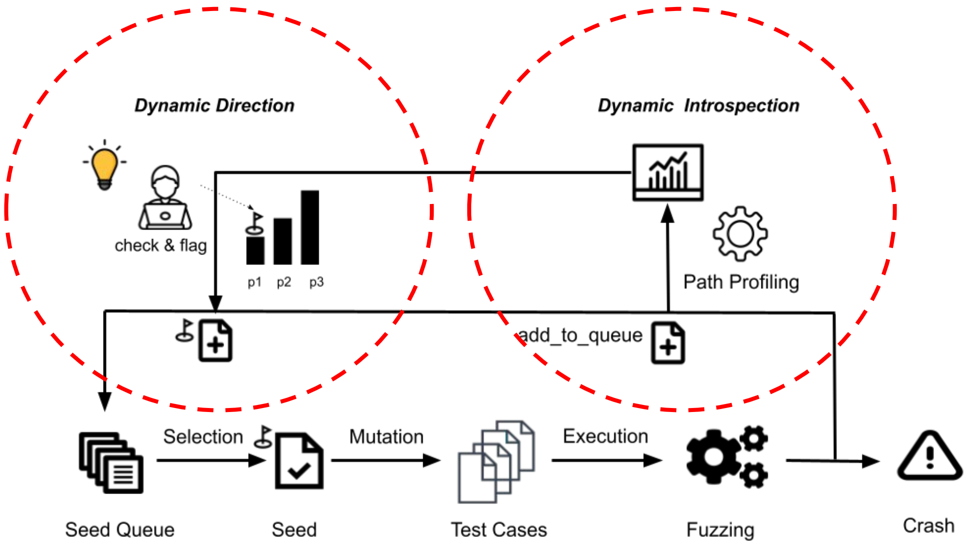
Research Questions*

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 ?

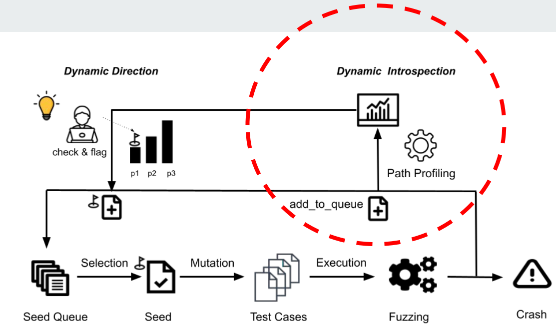
* "Fuzzing: Challenges and Reflections," in *IEEE Software*, vol. 38, no. 3, pp. 79-86, May-June 2021

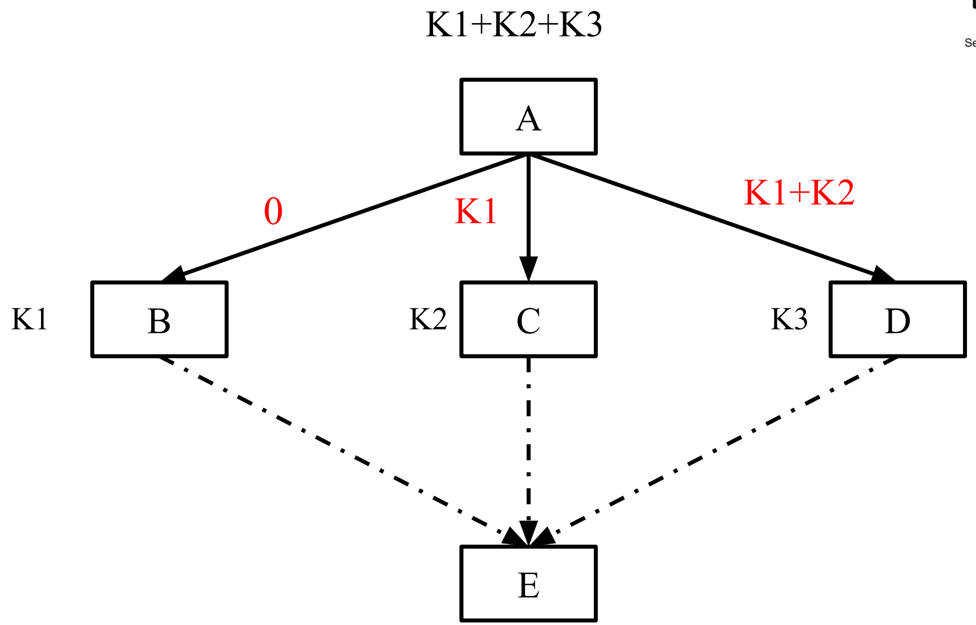
Design Overview



- 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

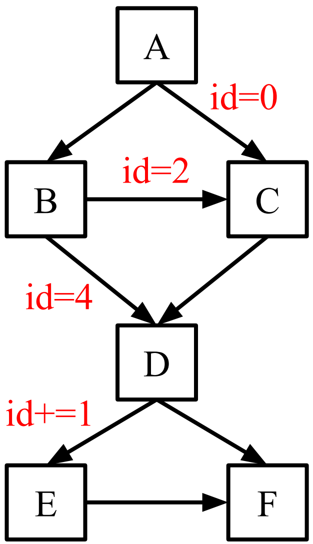
Ball-Larus Path Profiling*





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

Ball-Larus Path Profiling



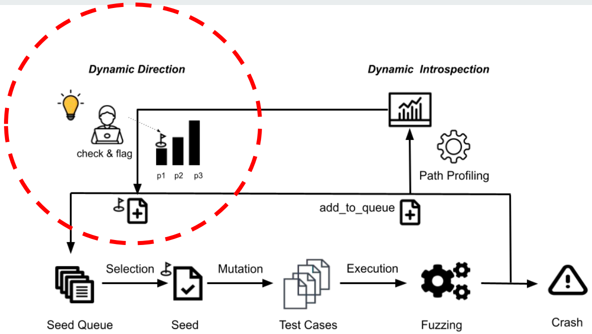
count[id]++

flag[id] = True/False?

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

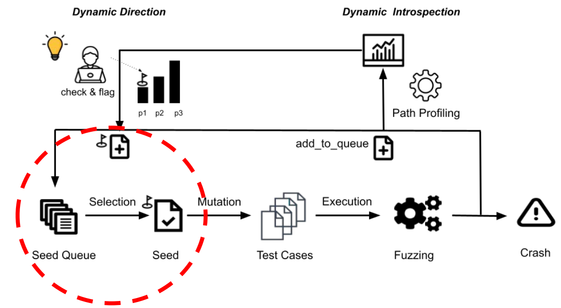
- Encoding
  - unique number(id)
  - reverse topological order
  - spanning tree & instrument
- Decoding
  - path ⇔ id => hit_count
  - user check & flag
  - connection

Path Flagging



- check & decode the path frequency
  - insights from instropection
- 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

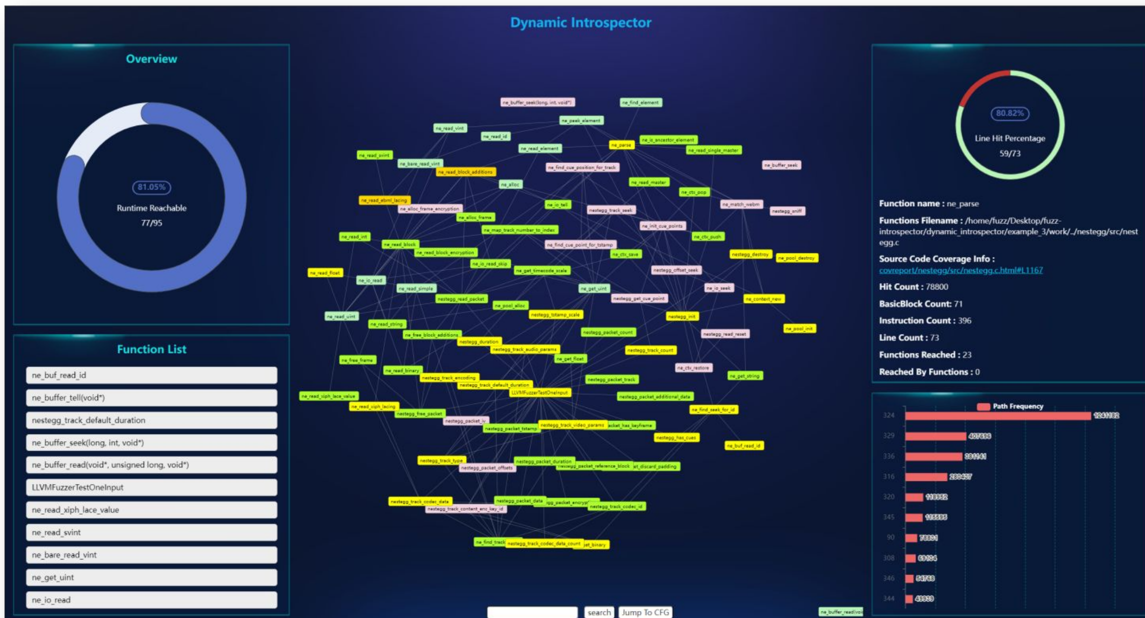
Seed Selection



- Intuition
  - seeds with flag tend to produce testcases with flag
  - flags(feature) are saved in evolution
- **Interactive Evolutionary Algorithm**
  - old research topic
  - fitness function + <u>human evaluation</u>
- Back to Motivating Example
  - target path/cases

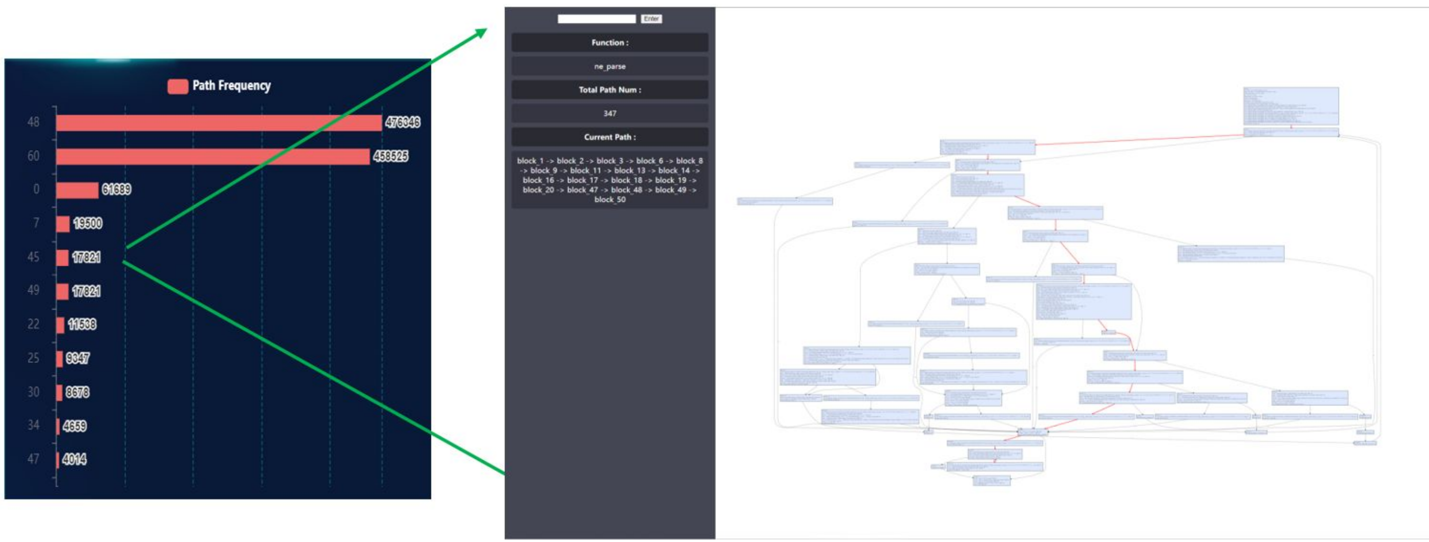
```
int wavlike_read_fmt_chunk (SF_PRIVATE *psf, int fmtsize) {
    ...
    switch (wav_fmt->format) {
        case WAVE_FORMAT_IMA_ADPCM:
            ...
            break;
        case WAVE_FORMAT_GSM610:
            ...
            break;
        case WAVE_FORMAT_EXTENSIBLE:
            // heap overflow (commit a8ab5b3)
            // how to focus this case in fuzzing?
    }
}
```

Implementation



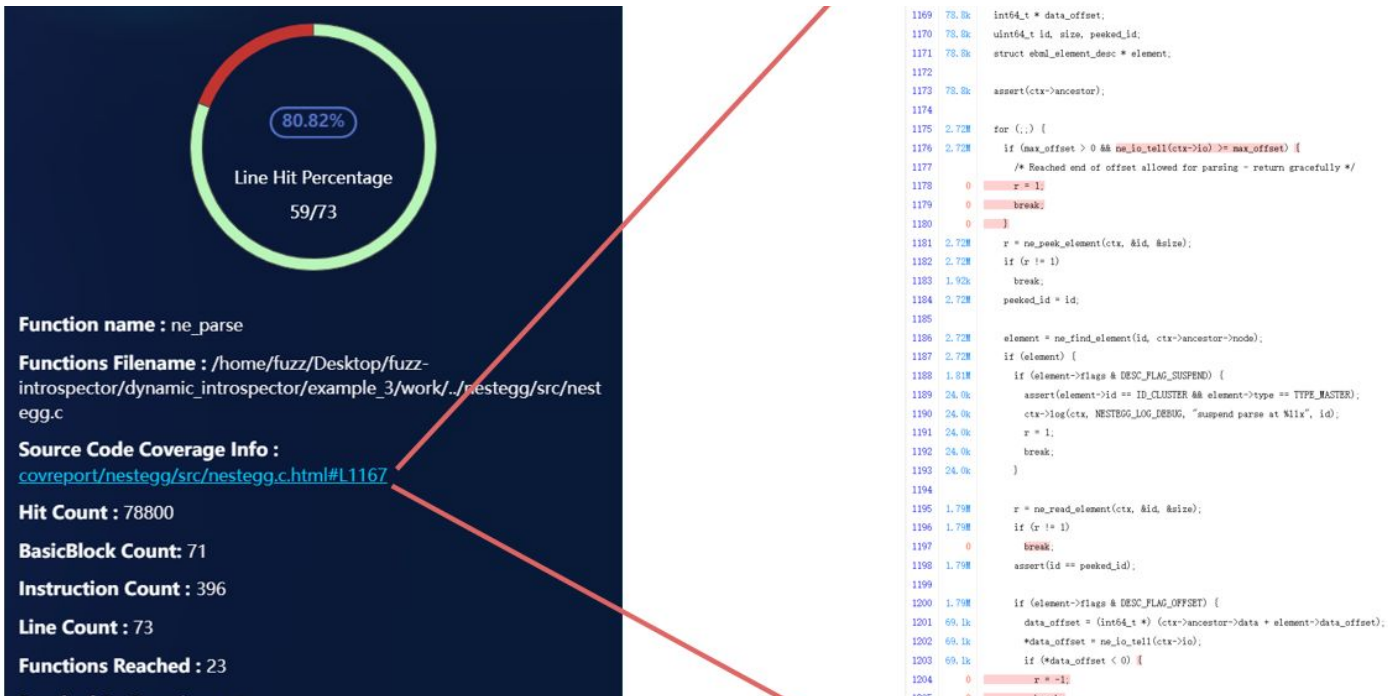
Dashboard

Implementation



Path Frequency & Decoding in CFG

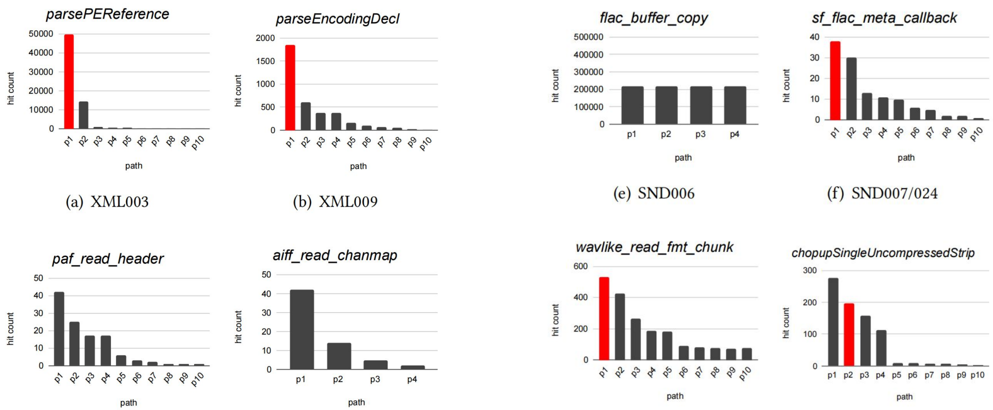
Implementation



Line Coverage , Combined with Fuzz-Introspector(llvm-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 (e) SND006 (f) SND007/024
(c) SND001 (d) SND005 (g) SND017/020 (h) TIF007/014

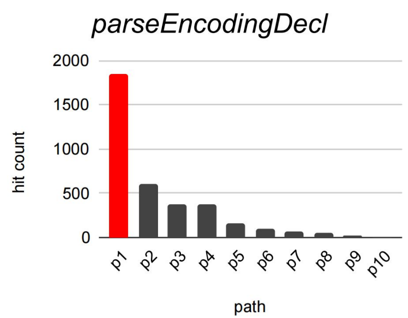
Blocking paths are highlighted in red.

Fuzzing wasted too much resources on these paths.

Evaluation

Listing 2: Code Snippet for Illustrating Path Frequency

```
const xmlChar *
xmlParseEncodingDecl(xmlParserCtxtPtr ctxt) {
    xmlChar *encoding = NULL;
    if (CMP8(CUR_PTR,
        'e','n','c','o','d','i','n','g')) {
            //   ....
            //  MAGMA(XML009)
    }
    return ;
}
```



(b) XML009

String Checking

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 Stragety 1

```
sf_flac_meta_callback () {
    switch (metadata->type){
        case FLAC_METADATA_TYPE_STREAM_INFO:
            ...
            // MAGMA_LOG(SND007)
            // MAGMA_LOG(SND024)
            break;
        case FLAC_METADATA_TYPE_UNDEFINED:
            break;
    }
}
```

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 Stragety 2

```
static int NeXTDecode(TIFF* tif, ..) {
    switch (n) {
        case 1: .. break
        case 2: .. break
          ...
        default:  // MAGMA_LOG(TIF008)
    }
}

int TIFFInitNeXT(TIFF* tif , int scheme) {
    (void) scheme;
    tif ->tif_predecode = NeXTPreDecode;
    tif ->tif_decoderow = NeXTDecode;
    tif -> tif_decodestrip = NeXTDecode;
    tif -> tif_decodetile = NeXTDecode;
    return (1) ;
}
```

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

use

def

Listing 5: Code Snippet for Flagging Stragety 3

```
static ExprList *exprListAppendList (...)  {
    // MAGMA_LOG(SQL020);
}

int sqlite3WindowRewrite(Select *p,  ...)  {
    // pWin use, data-flow  constraint
    if (p -> pWin && ...) {
        pSort = exprListAppendList (...) ;
    }
}

Window* sqlite3WindowAlloc(Parse *pParse,  ...)  {
    // pWin def
    pWin = (Window*)sqlite3DbMallocZero(pParse->db, sizeof(
          Window));
}

YYACTIONTYPE yy_reduce( ..) {
    switch(yyruleno) {
        case 315:
        case 316:
        case 317: sqlite3WindowAlloc(pParse,  ...)
        ...
    }
}
```

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