

# Метод совмещения результатов статического и динамического анализа для цикла разработки безопасного программного обеспечения

Мишечкин Максим Владимирович

[mishmax@ispras.ru](mailto:mishmax@ispras.ru)

Курмангалеев Шамиль Фаимович

[kursh@ispras.ru](mailto:kursh@ispras.ru)

# Сложности для статического анализа

## Ошибки первого и второго рода из-за:

- использование виртуальных методов;
- использование вызовов функций по указателям на вычисляемый адрес;
- недостаточной точности построения карты памяти;
- ...

## Необходимость ручной разметки предупреждений:

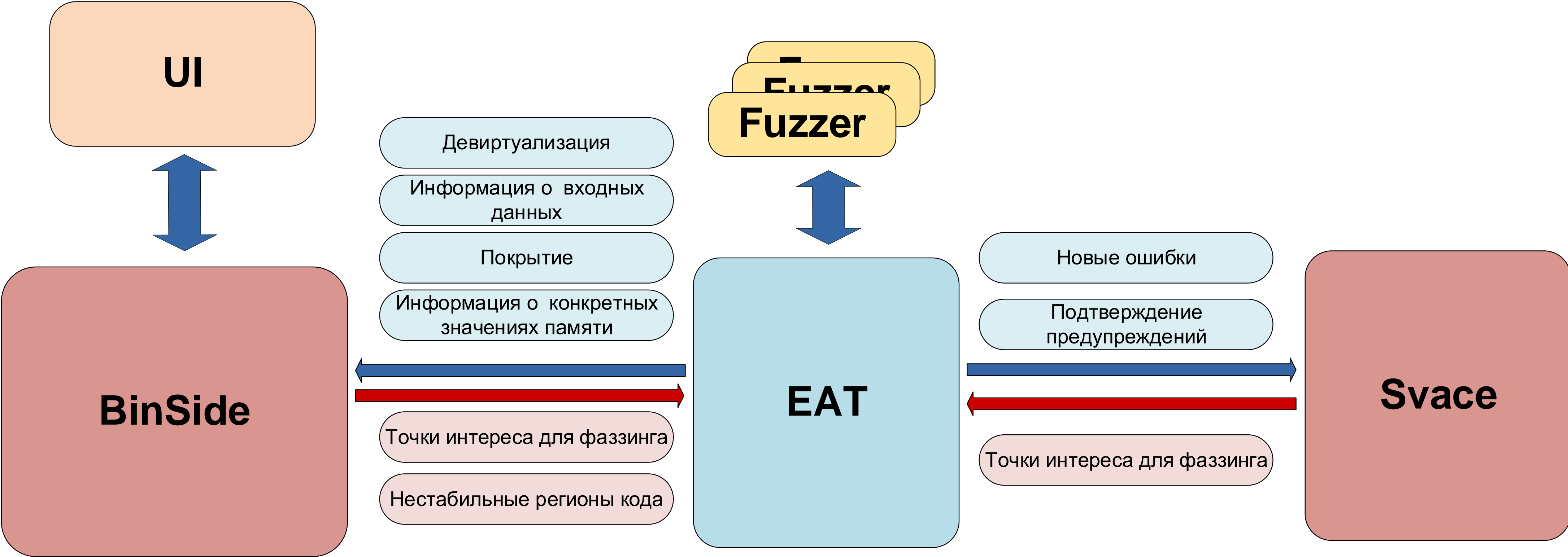
- разметка каждого предупреждения — это временные затраты;
- не полное понимание приоритета исправления найденных ошибок;
- нет гарантии, что человек правильно классифицирует ошибки;
- нет контекста на котором проявляется ошибка.

# Сложности для динамического анализа

- ограниченная эффективность выбора входных данных для мутации;
- нестабильные регионы кода, которые влияют на качество проводимых мутаций;
- ограниченное понимание влияния байт на покрытие;

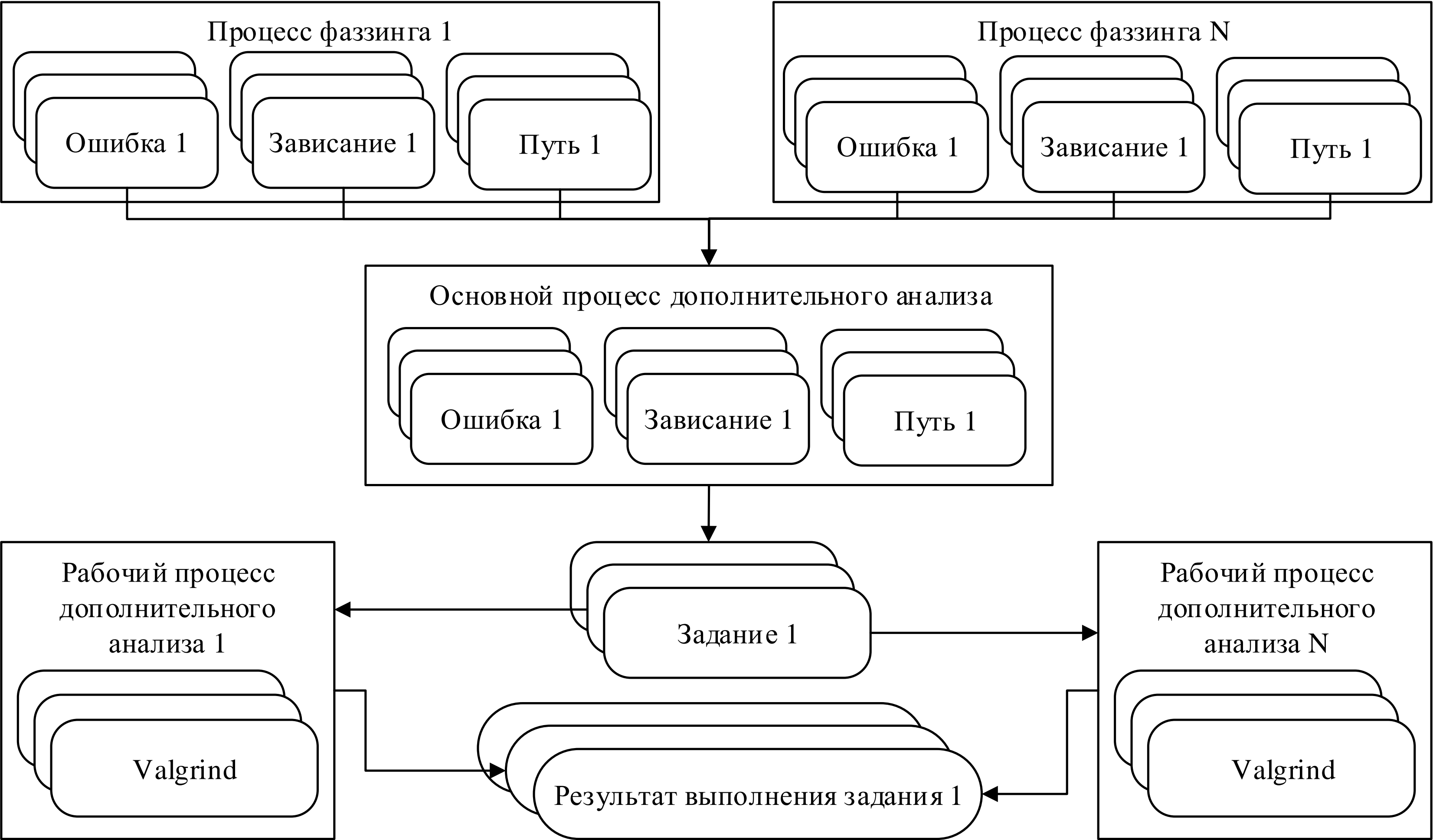
# Crusher

## Объединение статического и динамического анализа



# Crusher

## Модуль дополнительного анализа



# Crusher

## Интегрированные средства динамического анализа:

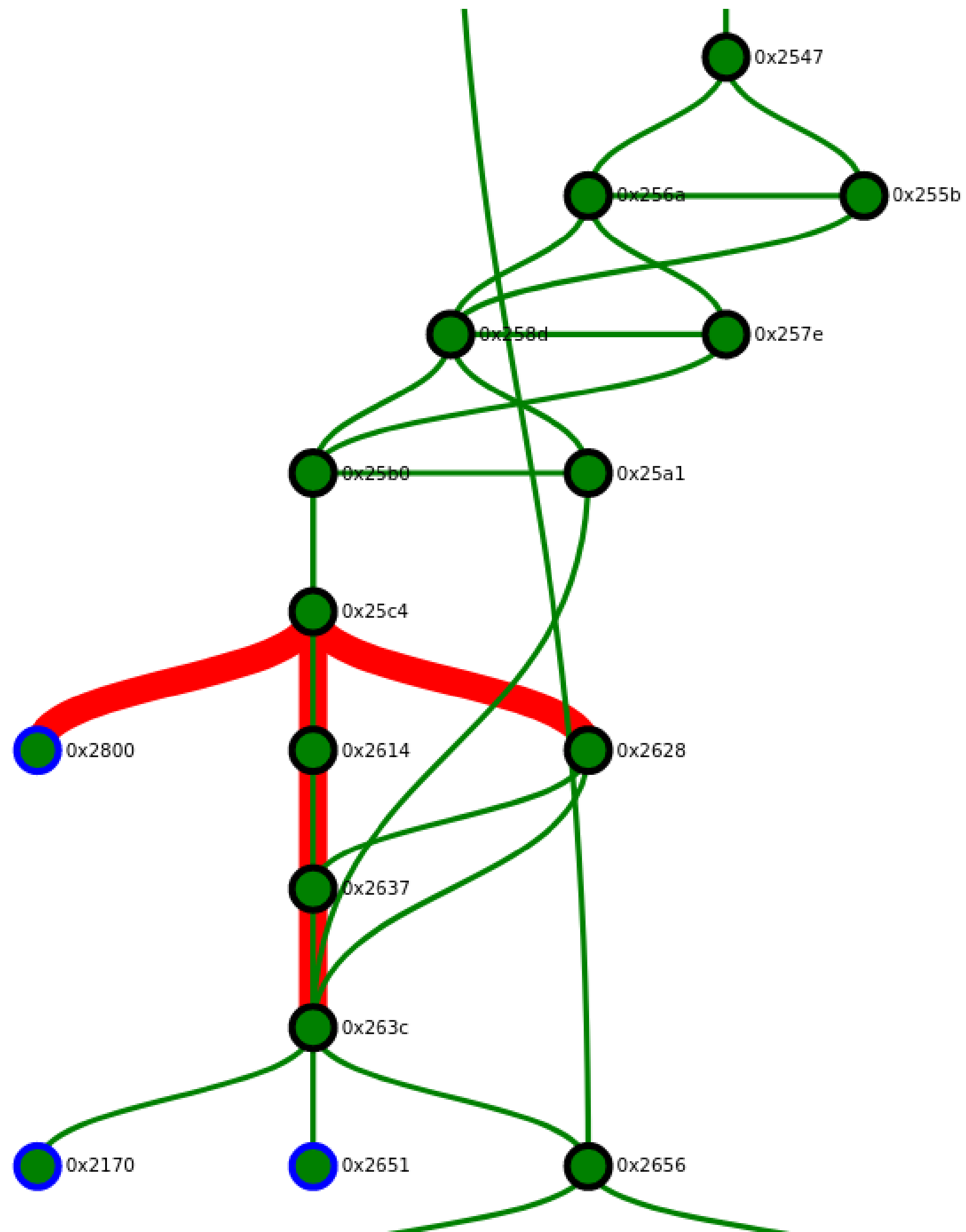
- Valgrind;
- Средства оценки критичности (exploitable, casr);
- DrMemory;
- QASan.

## Результаты расширенного анализа:

- Отчеты аварийных завершений для Svace;
- Подтверждение Svace Warnings;
- Lighthouse покрытие;
- Трассы для инструмента BinSide.

# Crusher

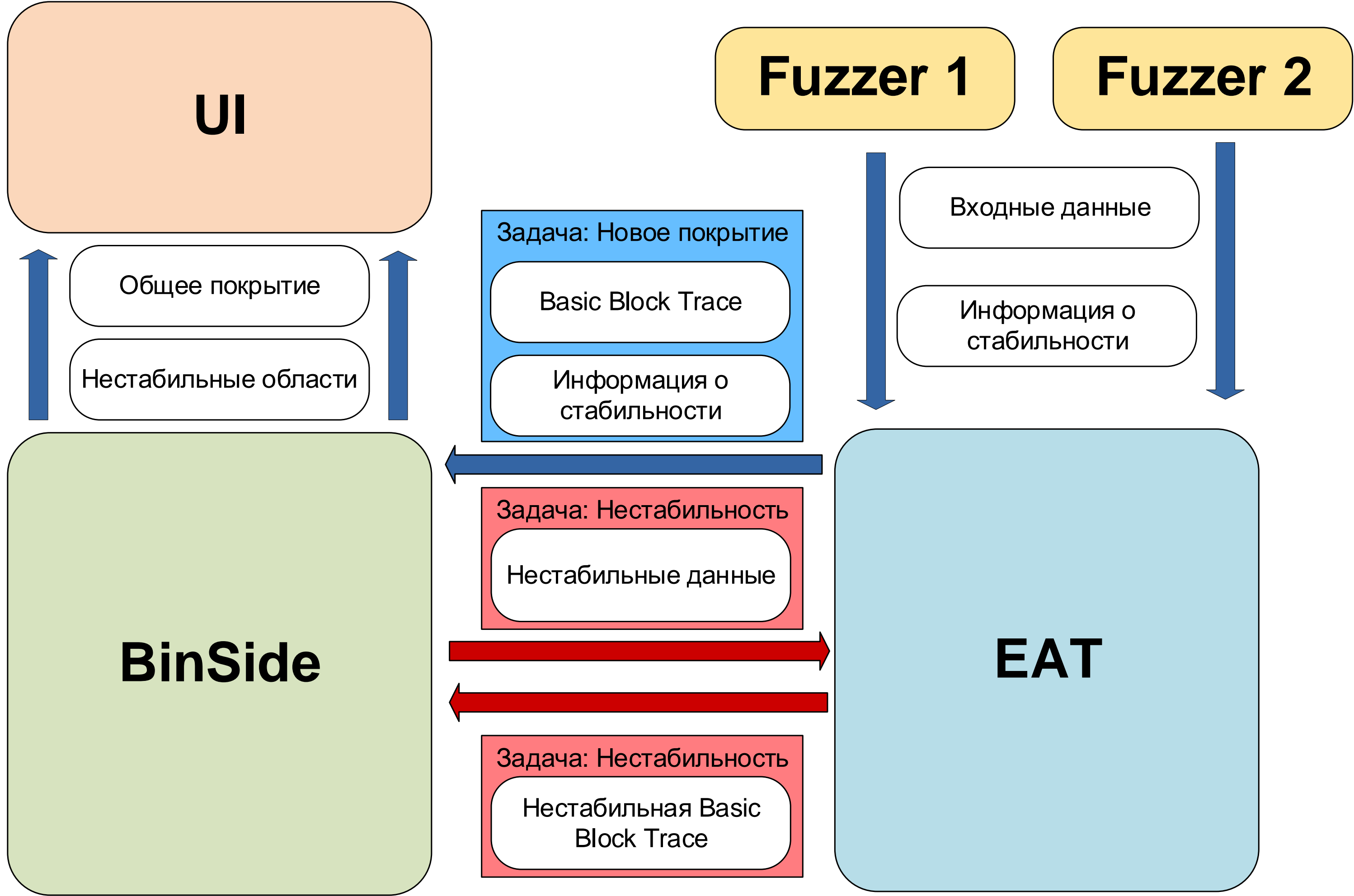
## Локализация нестабильного поведения в ПО



```
37     if (data[4] == 'f') {  
38         i++;  
39         int rand_number = dist(e);  
40         if (rand_number % 2 == 0) {  
41             i++;  
42         } else {  
43             i--;  
44         }  
45     }
```

# Crusher

## Локализация нестабильного поведения в ПО





# Crusher

## Подтверждение предупреждений Svace



# Crusher

## Подтверждение предупреждений Svace (результаты NGINX)

SvaceWarningsResults													
1004-BAD_COPY_PASTE-6	1004-BAD_COPY_PASTE-7	1004-BAD_COPY_PASTE-8	1005-BAD_COPY_PASTE-9	1005-BAD_COPY_PASTE-10	1005-BAD_COPY_PASTE-11	1008-BAD_COPY_PASTE-16	1009-BAD_COPY_PASTE-17	1018-DEREF_AFTER_F...	1018-DEREF_AFTER_F...	1018-DEREF_AFTER_F...	1018-DEREF_AFTER_F...	1018-DEREF_AFTER_F...	1018-DEREF_AFTER_F...
1024-DEREF_OF_NULL.CO...	1024-DEREF_OF_NULL.CO...	1025-DEREF_OF_NULL.CO...	1025-DEREF_OF_NULL.CO...	1026-DEREF_OF_NULL.CO...	1026-DEREF_OF_NULL.CO...	1027-DEREF_OF_NULL.CO...	1027-DEREF_OF_NULL.CO...	1030-DEREF_OF_NULL.EX...	1030-DEREF_OF_NULL.EX...	1030-DEREF_OF_NULL.EX...	1030-DEREF_OF_NULL.EX...	1031-DEREF_OF_NULL.EX...	1031-DEREF_OF_NULL.EX...
1046-DEREF_OF_NULL.ST...	1046-DEREF_OF_NULL.ST...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1073-INTEGER_OVERFLOW...	1074-INTEGER_OVERFLOW...	1074-INTEGER_OVERFLOW...
1111-NO_CAST_INTEGER...	1111-NO_CAST_INTEGER...	1111-NO_CAST_INTEGER...	1111-NO_CAST_INTEGER...	1111-NO_CAST_INTEGER...	1112-NO_CAST_INTEGER...	1112-NO_CAST_INTEGER...	1112-NO_CAST_INTEGER...	1112-NO_CAST_INTEGER...	1112-NO_CAST_INTEGER...	1113-NO_CAST_INTEGER...	1113-NO_CAST_INTEGER...	1113-NO_CAST_INTEGER...	1113-NO_CAST_INTEGER...
1121-NO_CAST_INTEGER...	1121-NO_CAST_INTEGER...	1122-NO_CAST_INTEGER...	1122-NO_CAST_INTEGER...	1122-NO_CAST_INTEGER...	1122-NO_CAST_INTEGER...	1122-NO_CAST_INTEGER...	1122-NO_CAST_INTEGER...	1123-NO_CAST_INTEGER...	1123-NO_CAST_INTEGER...	1123-NO_CAST_INTEGER...	1123-NO_CAST_INTEGER...	1123-NO_CAST_INTEGER...	1123-NO_CAST_INTEGER...
1133-NO_CAST_INTEGER...	1133-NO_CAST_INTEGER...	1134-NO_CAST_INTEGER...	1134-NO_CAST_INTEGER...	1134-NO_CAST_INTEGER...	1134-NO_CAST_INTEGER...	1134-NO_CAST_INTEGER...	1134-NO_CAST_INTEGER...	1143-NO_CAST_INTEGER...	1143-NO_CAST_INTEGER...	1143-NO_CAST_INTEGER...	1143-NO_CAST_INTEGER...	1143-NO_CAST_INTEGER...	1143-NO_CAST_INTEGER...
1145-NO_CAST_INTEGER...	1145-NO_CAST_INTEGER...	1146-NO_CAST_INTEGER...	1146-NO_CAST_INTEGER...	1146-NO_CAST_INTEGER...	1146-NO_CAST_INTEGER...	1146-NO_CAST_INTEGER...	1146-NO_CAST_INTEGER...	1151-NO_CAST_INTEGER...	1151-NO_CAST_INTEGER...	1151-NO_CAST_INTEGER...	1151-NO_CAST_INTEGER...	1151-NO_CAST_INTEGER...	1151-NO_CAST_INTEGER...
1152-NO_CAST_INTEGER...	1152-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1153-NO_CAST_INTEGER...	1154-NO_CAST_INTEGER...	1154-NO_CAST_INTEGER...

# Crusher

## Свace отчеты аварийных завершений

The screenshot displays the Svace web interface. At the top, there's a browser window with the address bar showing `localhost:8060/history/svacelight/index.html#show/test1/master/43138056b03d188970cbe1e458e3fd93e5882dd8/dd1a3fd3d5d7`. Below the browser, there are navigation elements: a dropdown menu with 'test1', another with 'master', a date/time string 'Fri Dec 28 16:51:09 MSK 2018', a 'Show' button, and a link '.svres:export import'.

The main content area is split into two columns. The left column, titled 'WARNING\_BY\_FUZZER (6)', lists several crash reports. Each report includes the text 'WARNING\_BY\_FUZZER (crashed-*N*) Program can crash here with signal 11' and a 'new' button. The fourth report is highlighted with a blue border and contains a backtrace:

- [backtrace] crash1 at prog.c:5
- fun at prog.c:25
- main at prog.c:34

Below the backtrace, there are 'Add comment', 'History', and two dropdown menus (one set to 'Undecided', the other to 'Unspecified').

The right column shows the source code for `/home/fedor/fuzzer/svace-test/test1/prog.c`. The code is as follows:

```
1 #include <stdio.h>
2
3 void crash1(void)
4 {
5     (crashed-1) Program can crash here with signal 11 [backtrace] crash1
6     *(int*)0 = 1;
7 }
8
9 void crash2(void)
10 {
11     *(int*)0 = 2;
12 }
13
14 void fun(int c)
15 {
16     int s = 0;
17     int i;
18     for (i = 0; i < c % 5 + 4; i++) {
19         s += i;
20     }
21     if (c % 2 == 0)
22         s += 1;
23     if (c % 3 == 0)
```

# Crusher

## Свace покрытие

proj master Tue Jan 31 12:51:16 MSK 2023 Show .svres:export import

**WARNING\_BY\_FUZZER (1)**

WARNING\_BY\_FUZZER (coverage\_queue) General previously fixed  
report

GENERAL\_WARNING:0

UD TP WF FP UC Severity Action History

- [backtrace] calc() at 1.cpp:9
- main at 1.cpp:17
- calc() at 1.cpp:10
- calc() at 1.cpp:7
- calc() at 1.cpp:8
- main at 1.cpp:12
- calc() at 1.cpp:5
- main at 1.cpp:13
- calc() at 1.cpp:6
- \_\_static\_initialization\_and\_destruction\_0(int, int) at iostream:74
- \_GLOBAL\_\_sub\_I\_Z4calcv at 1.cpp:19
- main at 1.cpp:18

Add comment

/fuzzing-tool/proj/1.cpp

```
1
2 #include <fstream>
3 #include <iostream>
4
5 void calc() {
6     int a = 2;
7     std::string b = "123";
8     int c = 3;
9     a += 3;
10 }
11
12 int main(int argc, char* argv[]) {
13     if (argc > 2) {
14         std::ofstream stream;
15         stream.open(argv[1]);
16     }
17     std::cout << "That`s ok" << std::endl;
18     calc();
19 }
20
```



# Crusher

## Valgrind отчеты Clion

The screenshot displays a code editor with the following C code snippet:

```
13 ngx_uint_t ngx_pagesize_shift;  
14 ngx_uint_t ngx_cacheline_size;  
15  
16  
17 void *  
18 ngx_alloc(size_t size, ngx_log_t *log)  
19 {  
20     void *p;  
21  
22     p = malloc(size);  
23     if (p == NULL) {  
24         ngx_log_error(NGX_LOG_EMERG, log, ngx_errno,  
25                     "malloc(%uz) failed", size);  
26     }  
27  
28     ngx_log_debug2(NGX_LOG_DEBUG_ALLOC, log, 0, "malloc: %p:%uz", p, size);  
29  
30     return p;  
31 }  
32  
33  
34 void *  
35 ngx_calloc(size_t size, ngx_log_t *log)  
36 {  
37     void *p;  
38  
39     p = ngx_alloc(size, log);  
40 }
```

The Valgrind Memcheck output in the bottom pane shows the following warnings:

- Leak\_PossiblyLost 14 warnings
  - ngx\_alloc.c 14 warnings
    - 128 bytes in 1 blocks are possibly lost in loss record 4 of 23 1 warning
      - 0x483B7F3 malloc
      - 0x145832 ngx\_alloc ngx\_alloc.c:22
      - 0x12D409 ngx\_crc32\_table\_init ngx\_crc32.c:117
      - 0x1228E2 main ngx.c:274
    - > 16,384 bytes in 1 blocks are possibly lost in loss record 16 of 23 1 warning
    - > 16,384 bytes in 1 blocks are possibly lost in loss record 17 of 23 1 warning
    - > 16,384 bytes in 1 blocks are possibly lost in loss record 18 of 23 1 warning
    - > 16,384 bytes in 1 blocks are possibly lost in loss record 19 of 23 1 warning
    - > 16,384 bytes in 1 blocks are possibly lost in loss record 20 of 23 1 warning
    - > 4,096 bytes in 1 blocks are possibly lost in loss record 8 of 23 1 warning
    - > 4,096 bytes in 1 blocks are possibly lost in loss record 9 of 23 1 warning
    - > 4,280 bytes in 1 blocks are possibly lost in loss record 10 of 23 1 warning
    - > 4,280 bytes in 1 blocks are possibly lost in loss record 11 of 23 1 warning
    - > 4,280 bytes in 1 blocks are possibly lost in loss record 12 of 23 1 warning
    - > 4,544 bytes in 1 blocks are possibly lost in loss record 13 of 23 1 warning
    - > 4,672 bytes in 1 blocks are possibly lost in loss record 14 of 23 1 warning
    - > 8 bytes in 1 blocks are possibly lost in loss record 1 of 23 1 warning

# Crusher

## Valgrind отчеты Svace

The screenshot displays the Svace web interface for a project named 'v1.21.4-loc...'. The main content area shows a Valgrind report for a memory leak in the file `ngx_cycle.c` at line 284. The report is titled 'Leak\_PossiblyLost null (14)'. The code editor shows the following code snippet:

```
1  /*
2  * Copyright (C) Igor Sysoev
3  * Copyright (C) Nginx, Inc.
4  */
5
6
7
8  #include <ngx_config.h>
9  #include <ngx_core.h>
10
11
12  ngx_uint_t ngx_pagesize;
13  ngx_uint_t ngx_pagesize_shift;
14  ngx_uint_t ngx_cacheline_size;
15
16
17  void *
18  ngx_alloc(size_t size, ngx_log_t *log)
19  {
20      void *p;
21
22      p = malloc(size);
23      if (p == NULL) {
24          ngx_log_error(NGX_LOG_EMERG, log, ngx_errno,
25                      "malloc(%uz) failed", size);
26      }
27
28      ngx_log_debug2(NGX_LOG_DEBUG_ALLOC, log, 0, "malloc: %p:%uz", p, size);
29
30      return p;
31  }
32
33
34  void *
35  ngx_calloc(size_t size, ngx_log_t *log)
36  {
37      void *p;
38
39      p = ngx_alloc(size, log);
40
41      if (p) {
42          ngx_memzero(p, size);
43      }
44
45      return p;
46  }
47
48
49  #if (NGX_HAVE_POSIX_MEMALIGN)
50
51  void *
52  ngx_memalign(size_t alignment, size_t size, ngx_log_t *log)
53  {
54      void *p;
55      int err;
56
57      err = posix_memalign(&p, alignment, size);
58
59      if (err) {
60          ngx_log_error(NGX_LOG_EMERG, log, err,
61                      "posix_memalign(%uz, %uz) failed", alignment, size);
62          p = NULL;
63      }
64
65      ngx_log_debug3(NGX_LOG_DEBUG_ALLOC, log, 0,
```

The right sidebar shows a message titled 'Leak\_PossiblyLost null ngx\_cycle.c:284' with the message content 'EMPTY BINSIDE COMMENT'. Below the message is a list of binside trace elements:

1. Role: defect
1. Binside trace element | 0x47337EF
2. Binside trace element | 0x3D752
3. Binside trace element | 0x1AFB7
4. Binside trace element | 0x42C96
5. Binside trace element | 0x1AAC2

**Спасибо за просмотр**