Software Engineering Conference Russia
November 14-15, 2019. Saint-Petersburg

Современное состояние исследований и разработок в области автоматического анализа программ

Александр Герасимов

ИСП РАН

Что такое ошибка в программе?

- IEEE 1044-2019 Standard Classification of Software Anomalies
- NIST. Software Error Analysis

Defect, anomaly, error, failure, fault… – много терминов, мало смысла

Что такое ошибка в программе?

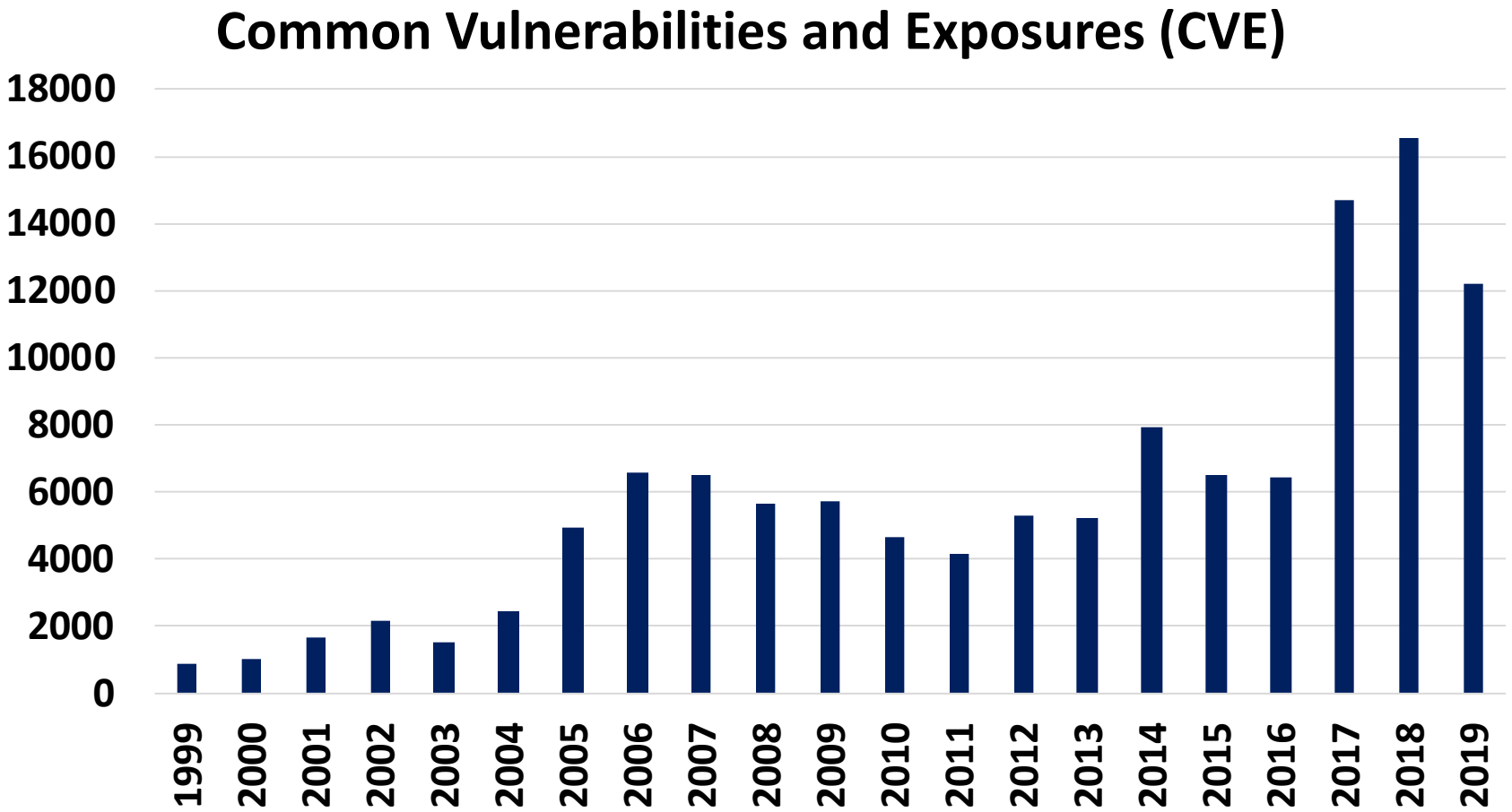
- IEEE 1044-2019 Standard Classification of Software Anomalies
- NIST. Software Error Analysis

Defect, anomaly, error, failure, fault… – много терминов, мало смысла

Программа должна работать корректно

в соответствии со спецификацией.

Зачем искать ошибки?



Критичность ошибки

Критичность ошибки определяется характером автоматизируемых функций.

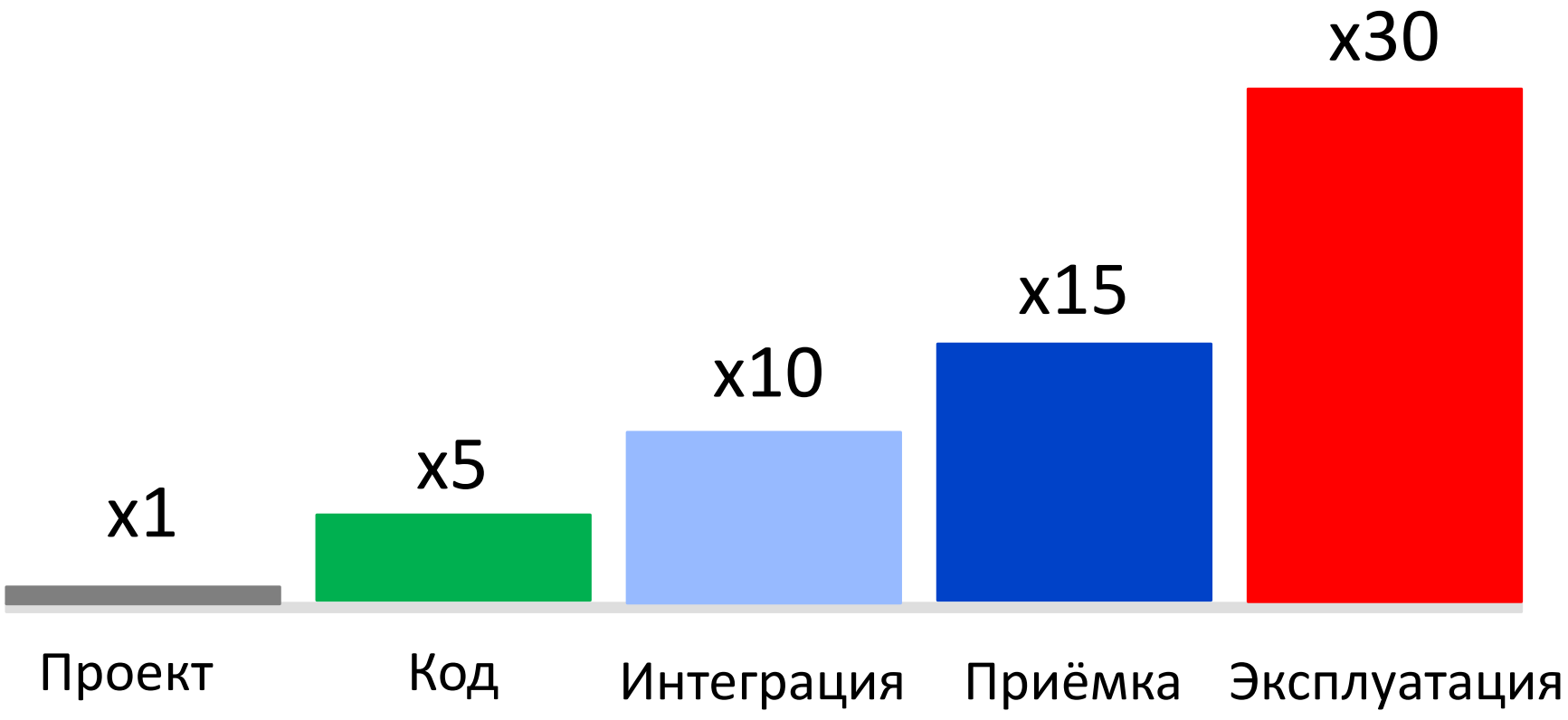
SECR

Цена ошибки

Ariane 5: переполнение при записи 64 бит в 16 битную ячейку

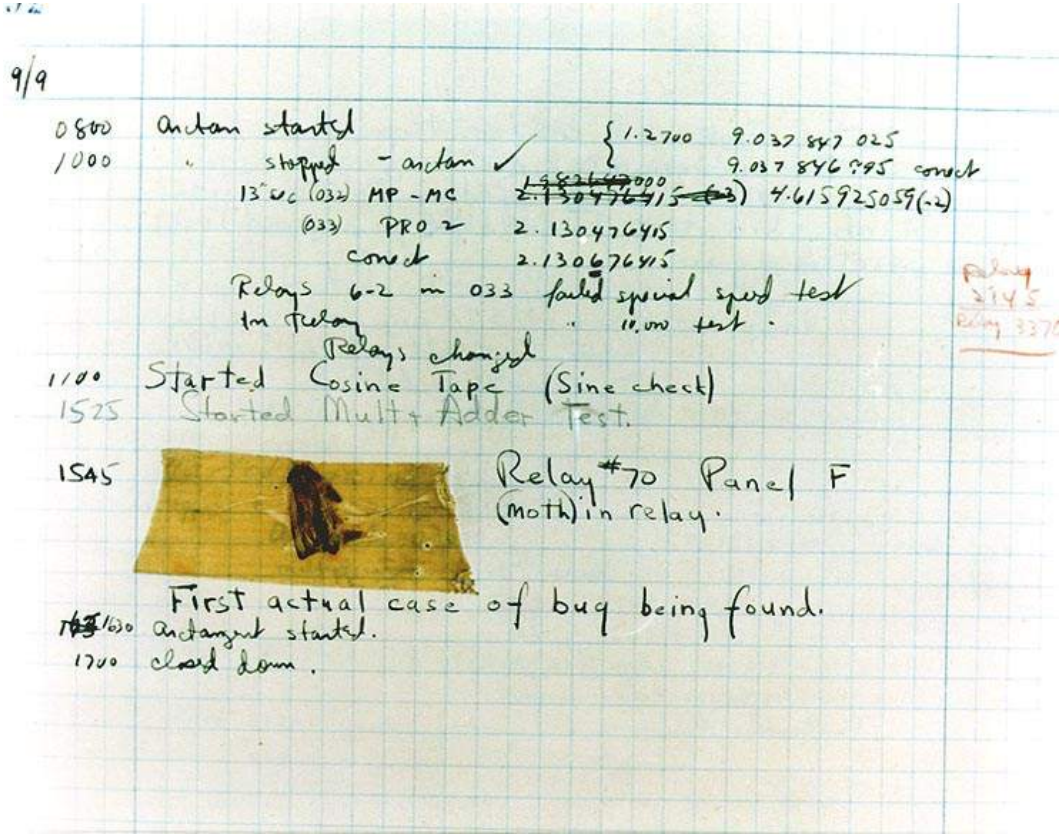
8 500 000 000$

Стоимость исправления ошибки



История

1. 1940-е: физическое удаление насекомых - debugging



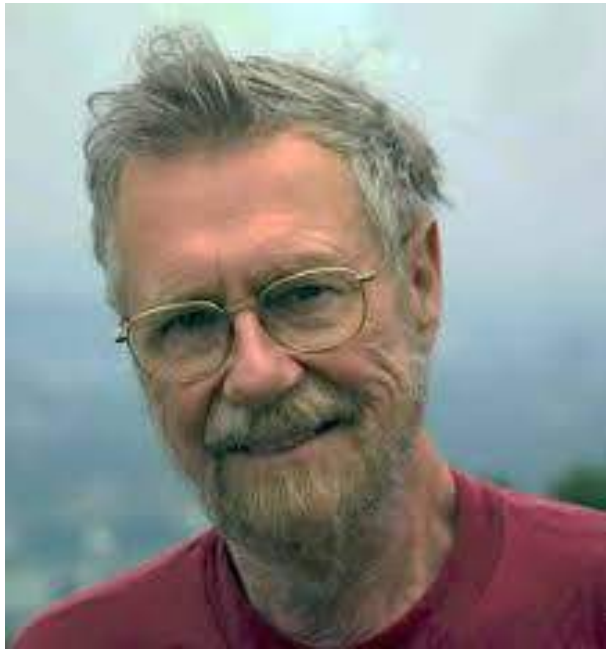
9 сентября 1947
Грэйс Хоппер

История

1. 1940-е: физическое удаление насекомых - debugging
2. 1950-е: Инспекция кода и тестирование

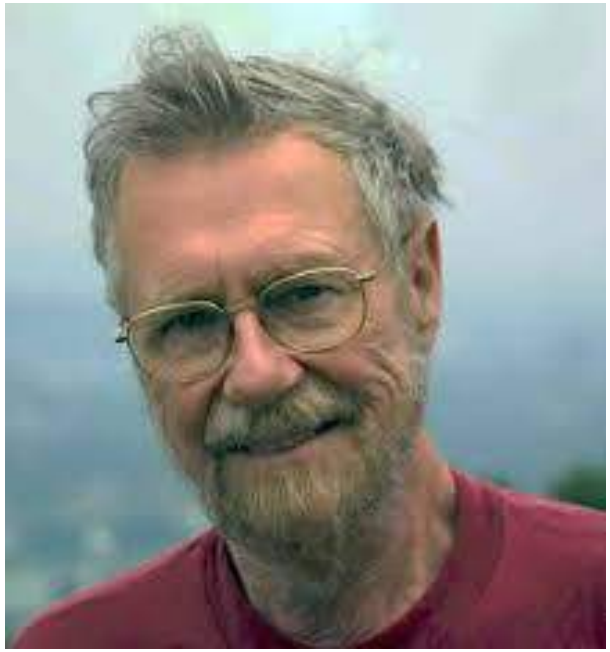
Тестирование

- Ручное
- Исчерпывающее



Тестирование

Тестирование программы может весьма эффективно продемонстрировать наличие ошибок, но безнадёжно неадекватно для демонстрации их отсутствия.



Эдсгер Вибе Дейкстра

Тестирование, что дальше?

Автоматическое построение покрытия

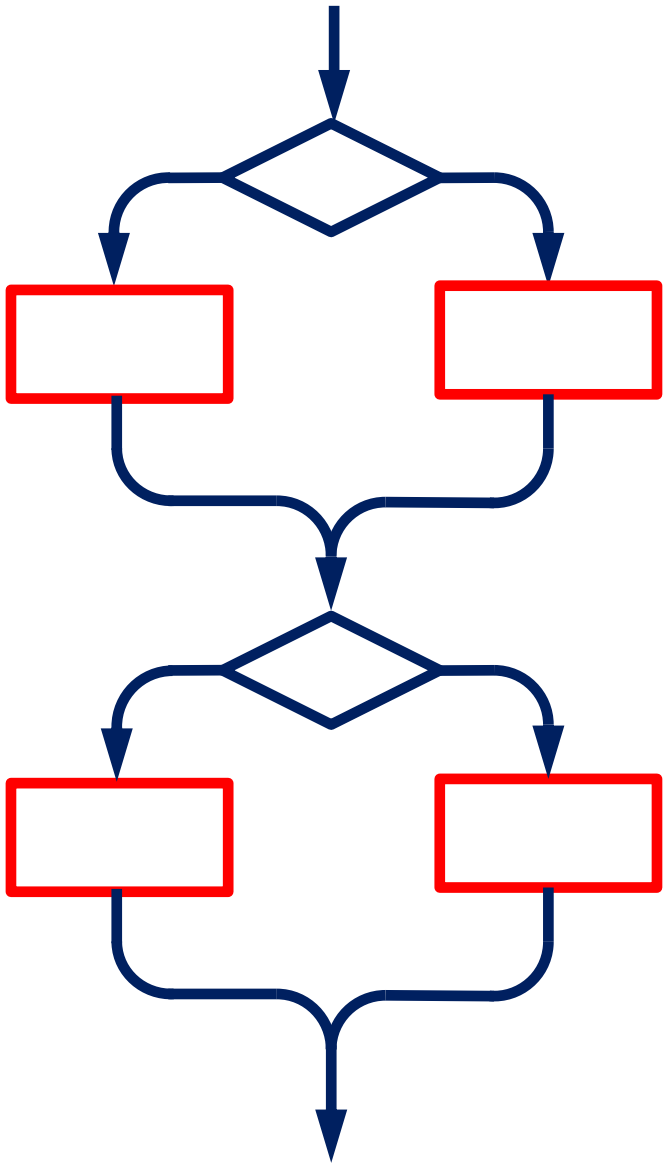
1975: Динамическое символьное исполнение

1980-е: Фаззинг

Покрытие кода

Автоматическое построение покрытия:

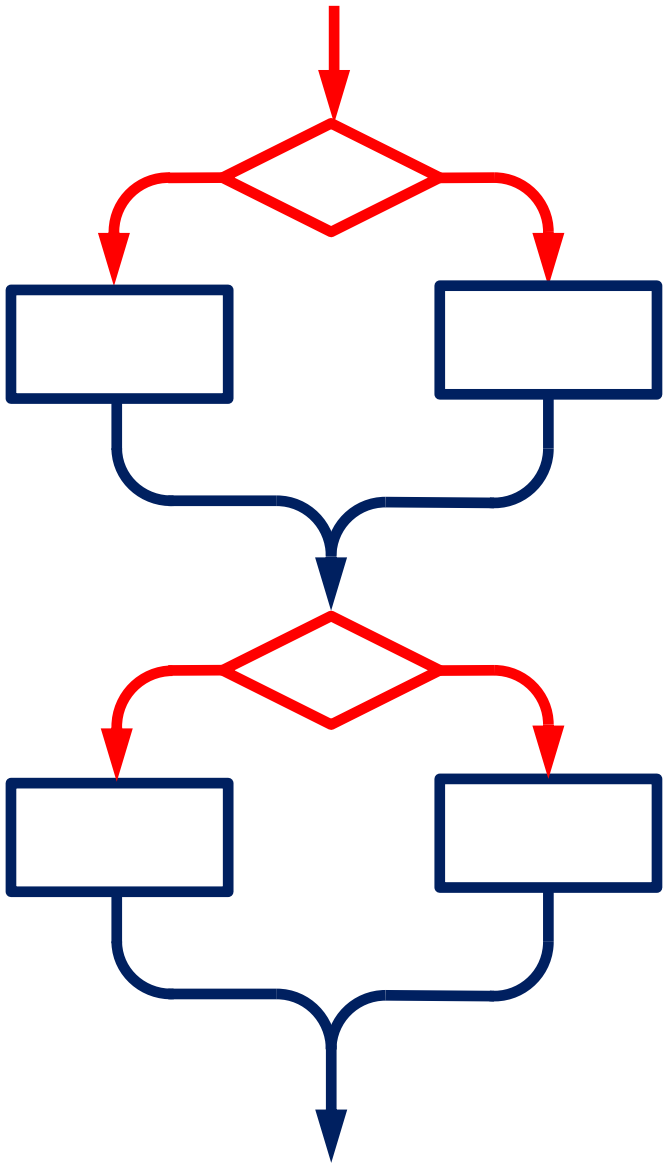
- По базовым блокам



Покрытие кода

Автоматическое построение покрытия:

- По базовым блокам
- По исходам условных выражений



Покрытие кода

Автоматическое построение покрытия:

- По базовым блокам
- По исходам условных выражений
- По исходам операндов условных выражений (MC/DC)

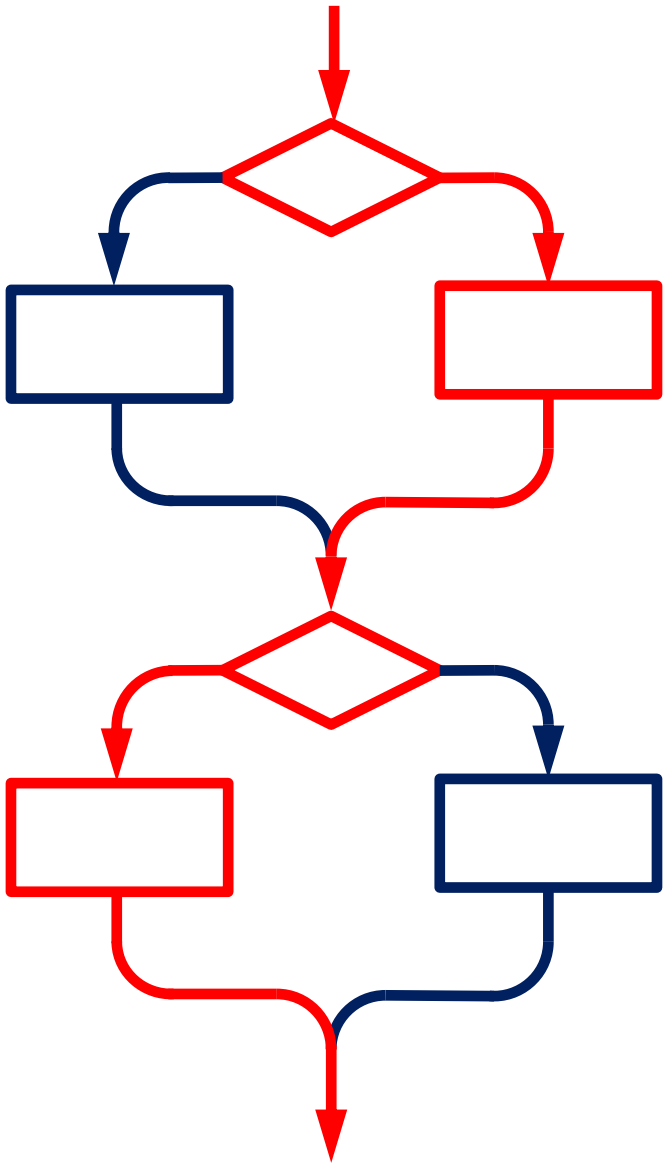
```
if(a || b){
…
}
```

a	T	T	F	F
b	T	F	T	F

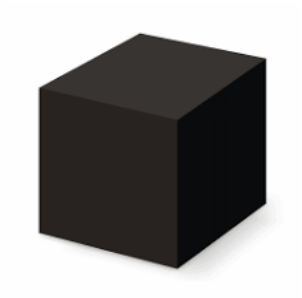
Покрытие кода

Автоматическое построение покрытия:

- По базовым блокам
- По исходам условных выражений
- По исходам операндов условных выражений (MC/DC)
- По путям



Фаззинг

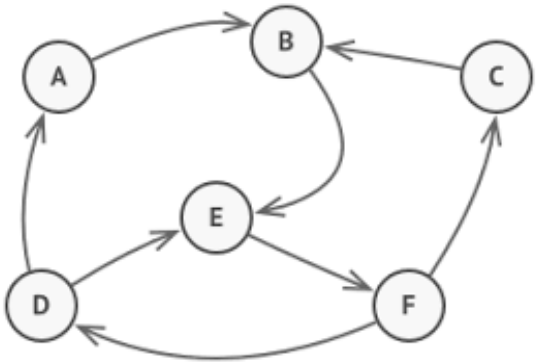


Черный ящик

Серый ящик

Прозрачный ящик

Псевдослучайные



По модели

Фаззинг: проблемы

```
void foo(long x)
{
    if ( x == 1234567)
    {
        abort();
    }
}
```

$$P(1234567) = \frac{1}{4294967295} = 0{,}000000000232831$$

Фаззинг: практика

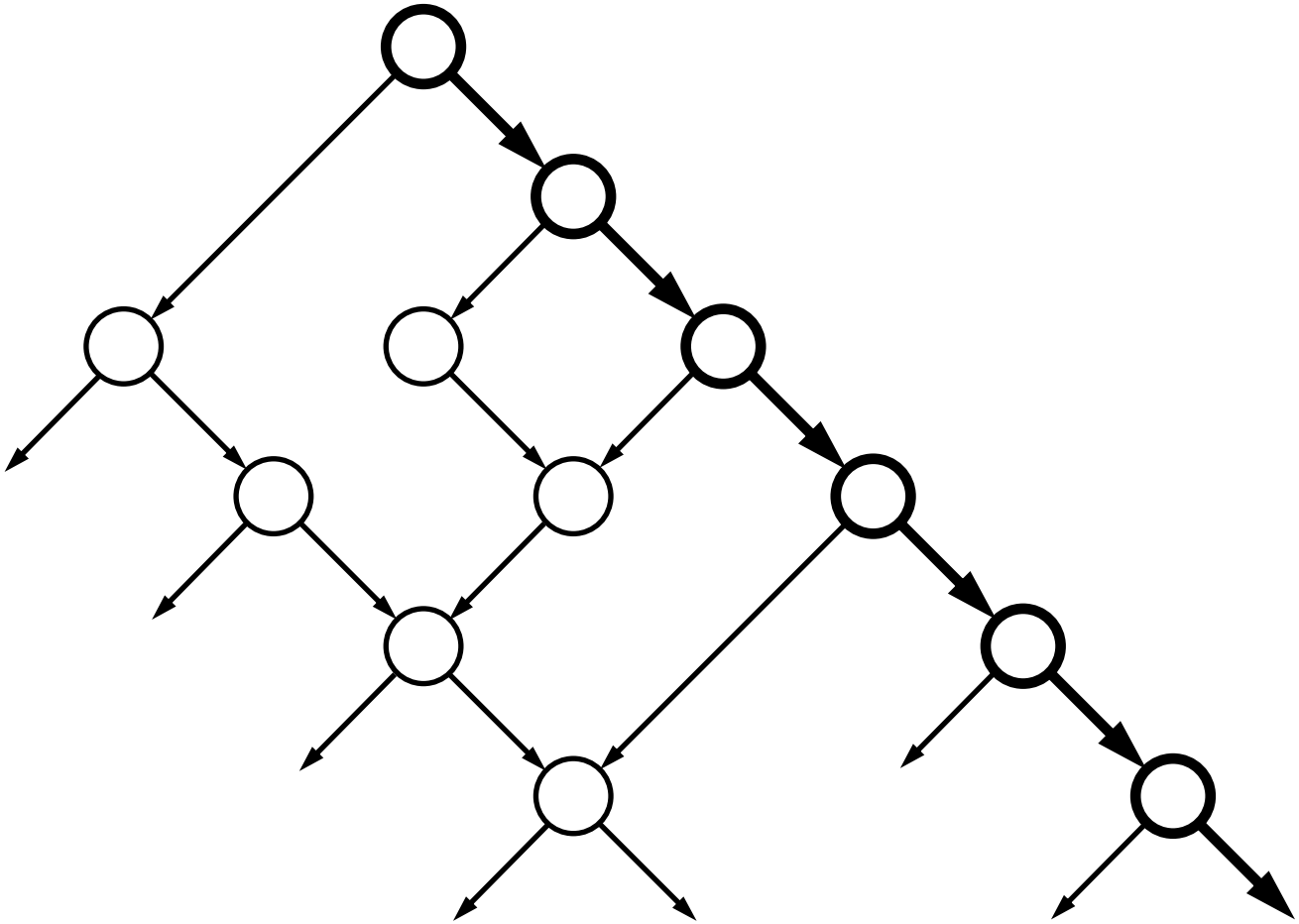
Применение санитизации кода позволяет найти «тихие» ошибки

- AddressSanitizer
- LeakSanitizer
- MemorySanitizer
- ThreadSanitizer
- UndefinedBehaviourSanitizer

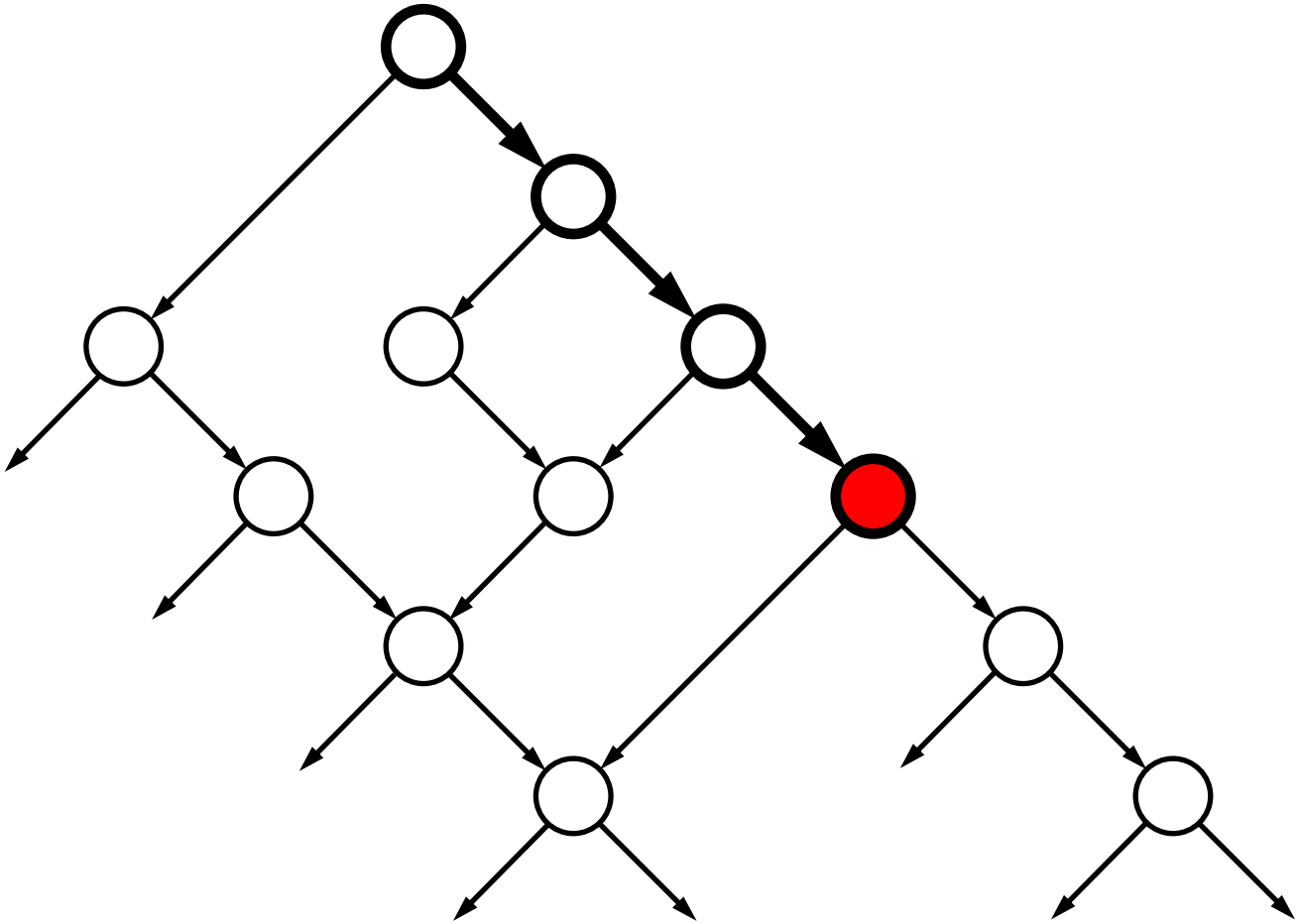
Фаззинг: инструменты

- Americal Fuzzy Lop
- Radamsa
- Peach Fuzzing Platform
- Defensics
- Syzkaller
- ISP Fuzzer

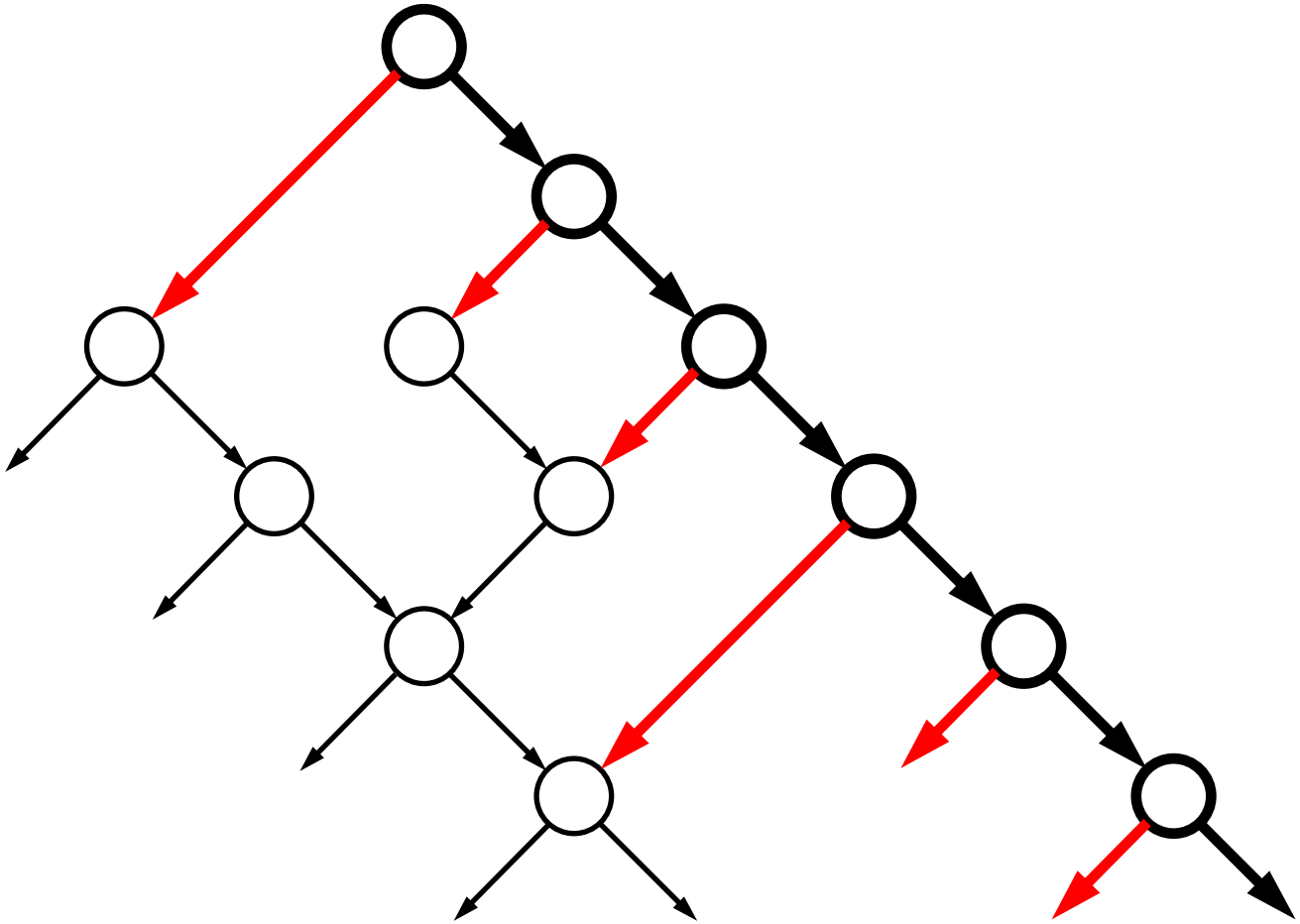
Динамическое символьное исполнение



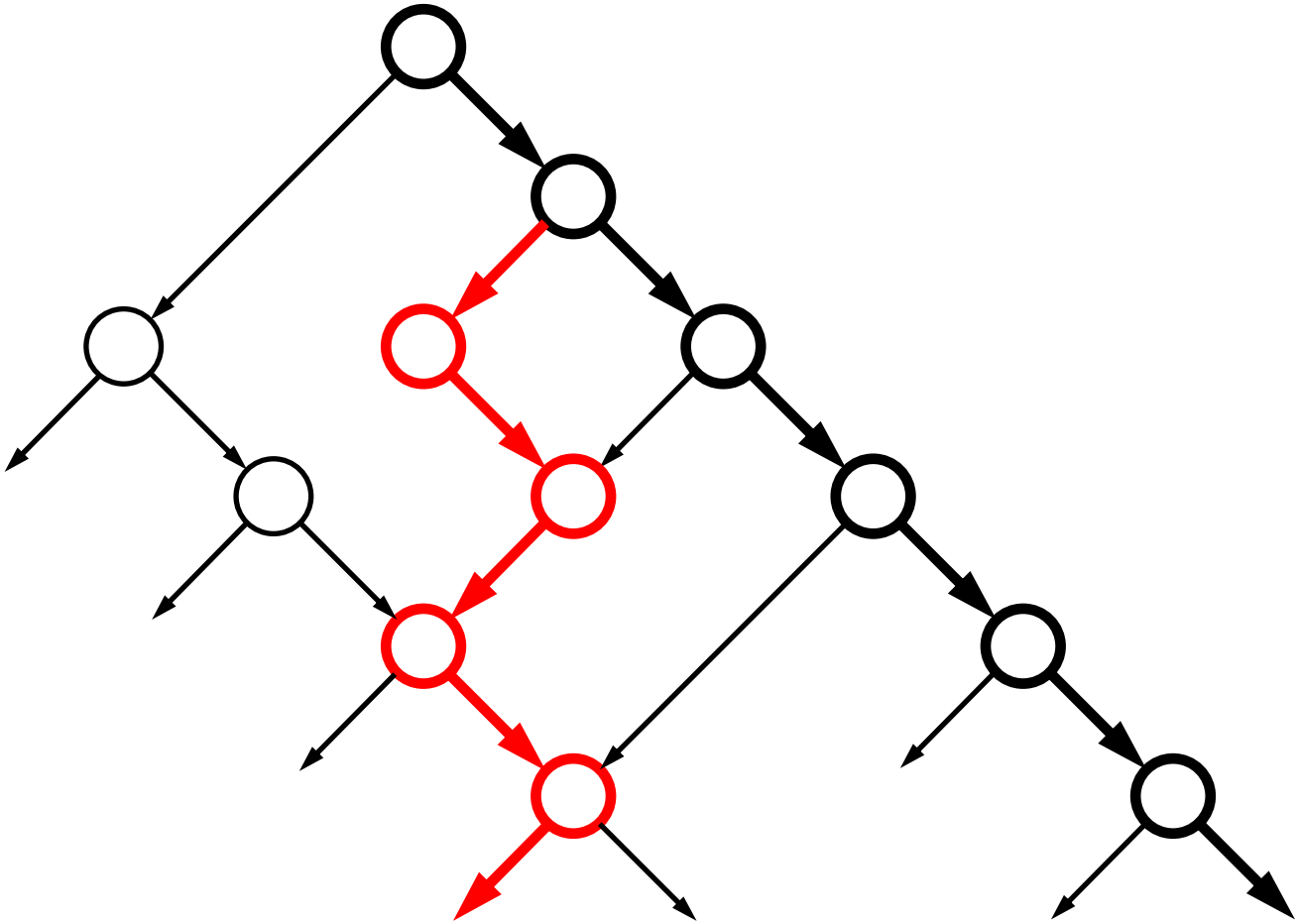
Динамическое символьное исполнение



Динамическое символьное исполнение



Динамическое символьное исполнение



Динамическое символьное исполнение

Динамическое Символьное исполнение

1975 – впервые дано описание метода

2005 – первые автоматические инструменты

Идея: целенаправленное построение тестового покрытия программы.

Проблемы: низкая производительность.

Причины:

- Замедление работы программы из-за инструментации
- Высокая вычислительная сложность решения задач SMT
- Большое количество путей для исследования, чаще - ∞

SECR

Динамическое Символьное исполнение

Известные среды анализа:

- KLEE
- S^2E
- angr
- Anxiety

Sage (Microsoft Research)

Impact: since 2007

- 200+ machine years (in largest fuzzing lab in the world)
- 1 Billion+ constraints (largest SMT solver usage ever!)
- 100s of apps, 100s of bugs (missed by everything else…)
- Ex: 1/3 of all Win7 WEX security bugs found by SAGE →
- Bug fixes shipped quietly (no MSRCs) to 1 Billion+ PCs
- Millions of dollars saved (for Microsoft and the world)
- SAGE is now used daily in Windows, Office, etc.

SECR

Статический анализ

1951 – Теорема Райса (Генри Гордон Райс) о невозможности определения нетривиальных свойств вычислимых функций.

Расходимся…

Статический анализ

1951 – Теорема Райса (Генри Гордон Райс) о невозможности определения нетривиальных свойств вычислимых функций.

1970-е: ситаксические анализаторы (lint)

2000-е:

- Klocwork inSight
- Coverity Prevent
- Svace

История

1. 1940-е: физическое удаление насекомых - debugging
2. 1950-е: Инспекция кода и тестирование
3. 1970-е: Стандарты кодирования и первые инструменты статического анализа (lint)

```
if (a == b) {
    …
}
```

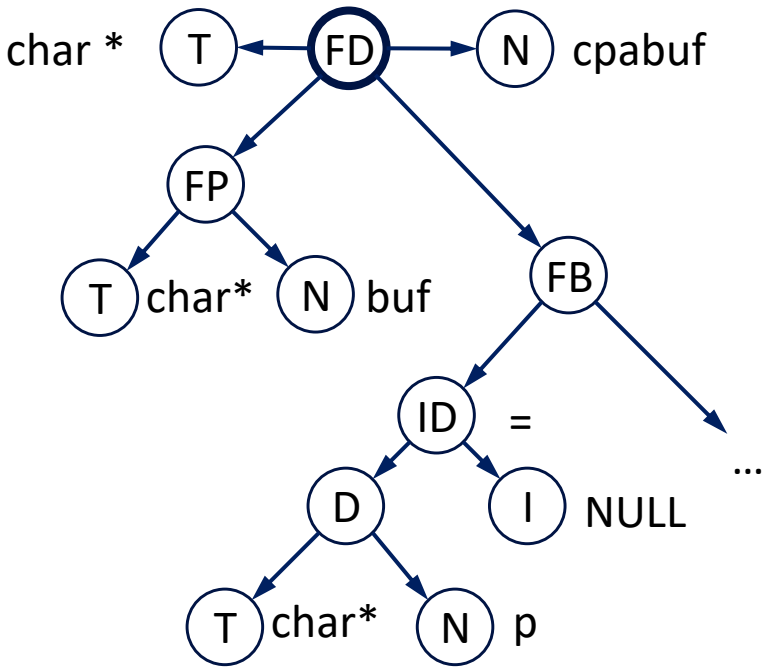
```
if (a = b) {
    …
}
```

Статический анализ

```
char * cpabuf(char * buf){
   char* p = NULL;

   if ( strlen( buf ) > 0 ) {
      if( buf [0] == 'a'  ) {
         p = malloc( strlen( buf ) + 1);
      } else {
         logerror('unexpected input');
      }
   }

   strcpy(p, buf);
   return p;
}
```

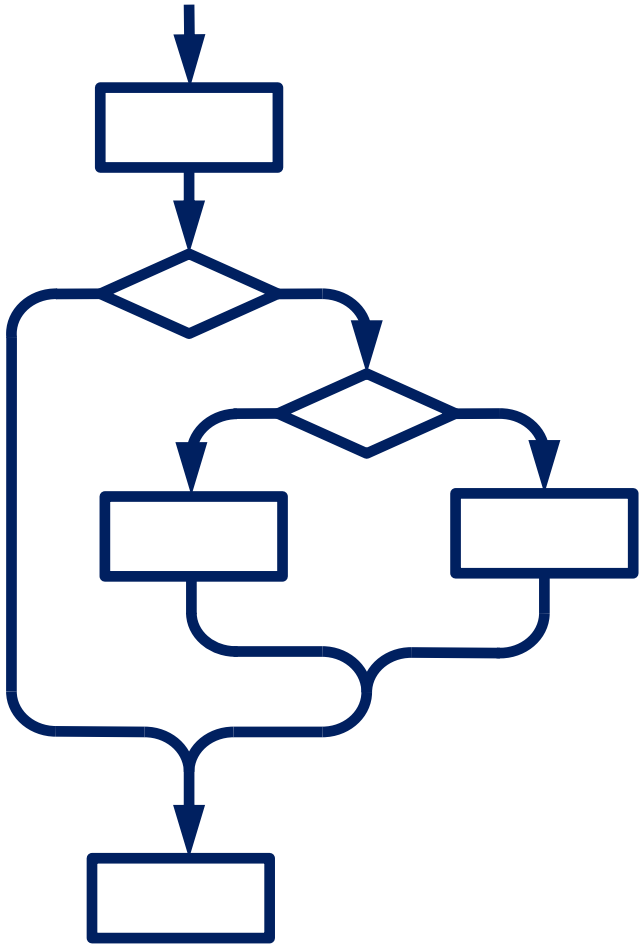


Статический анализ

```
char * cpabuf(char * buf){
   char* p = NULL;

   if ( strlen( buf ) > 0 ) {
      if( buf [0] == 'a'  ) {
         p = malloc( strlen( buf ) + 1);
      } else {
         logerror('unexpected input');
      }
   }

   strcpy(p, buf);
   return p;
}
```

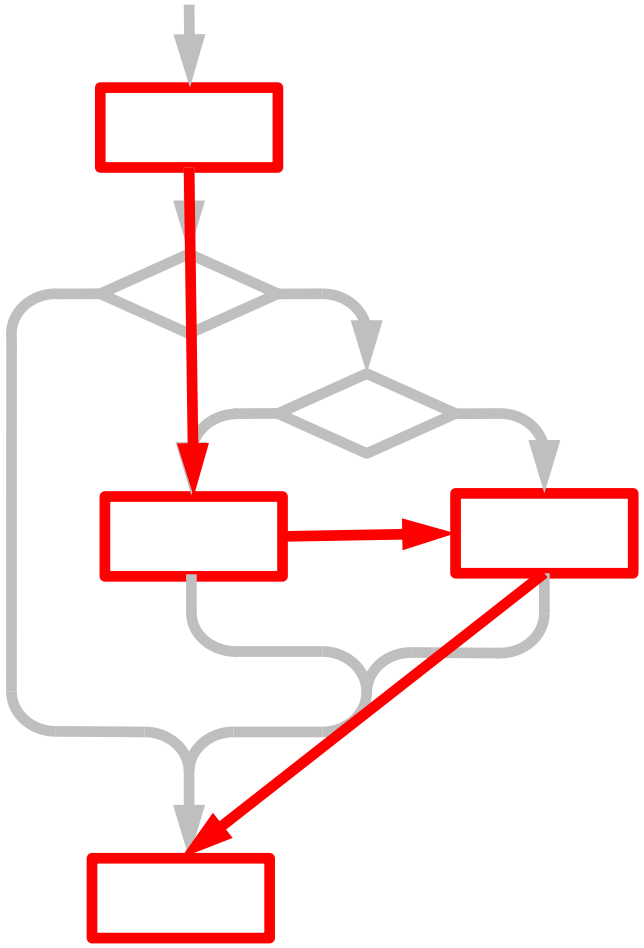


Статический анализ

```
char * cpabuf(char * buf){
   char* p = NULL;

   if ( strlen( buf ) > 0 ) {
      if( buf [0] == 'a'  ) {
         p = malloc( strlen( buf ) + 1);
      } else {
         logerror('unexpected input');
      }
   }

   strcpy(p, buf);
   return p;
}
```

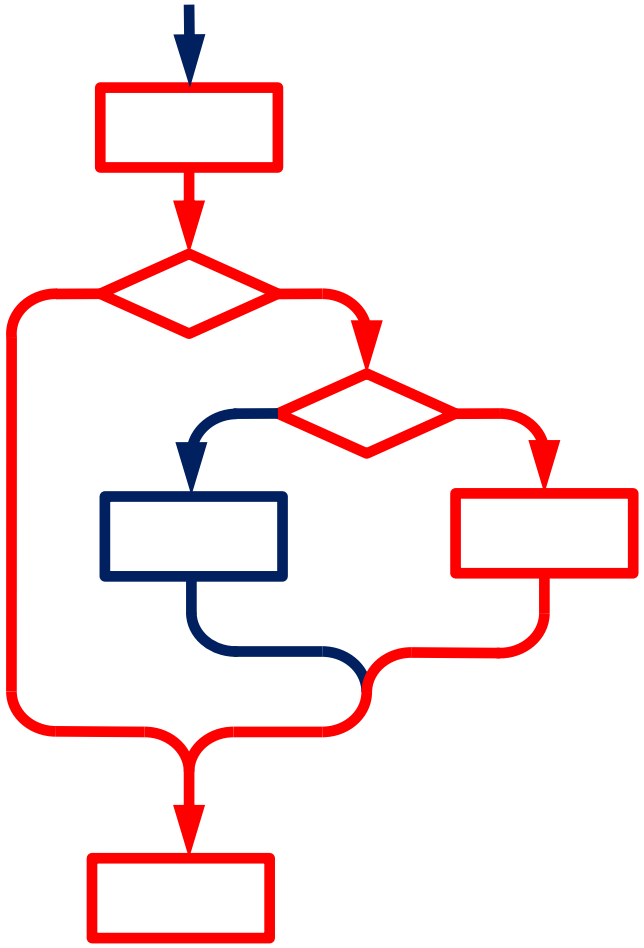


Статический анализ

```
char * cpabuf(char * buf){
   char* p = NULL; // Инициализация

   if ( strlen( buf ) > 0 ) {
      if( buf [0] == 'a'  ) {
         p = malloc( strlen( buf ) + 1);
      } else {
         logerror('unexpected input');
      }
   }

   strcpy(p, buf); // Реализация
   return p;
}
```



Статический анализ

```
char * cpabuf(char * buf){
   char* p = NULL; // Инициализация

   if ( strlen( buf ) > 0 ) {
      if( buf [0] == 'a'  ) {
         p = malloc( strlen( buf ) + 1);
      } else {
         logerror('unexpected input');
      }
   }

   strcpy(p, buf); // Реализация
   return p;
}
```

```
void foo(){

   char * a = cpabuf("allow"); // Ok

   char * b = cpabuf("deny"); // Error

}
```

Статический анализ

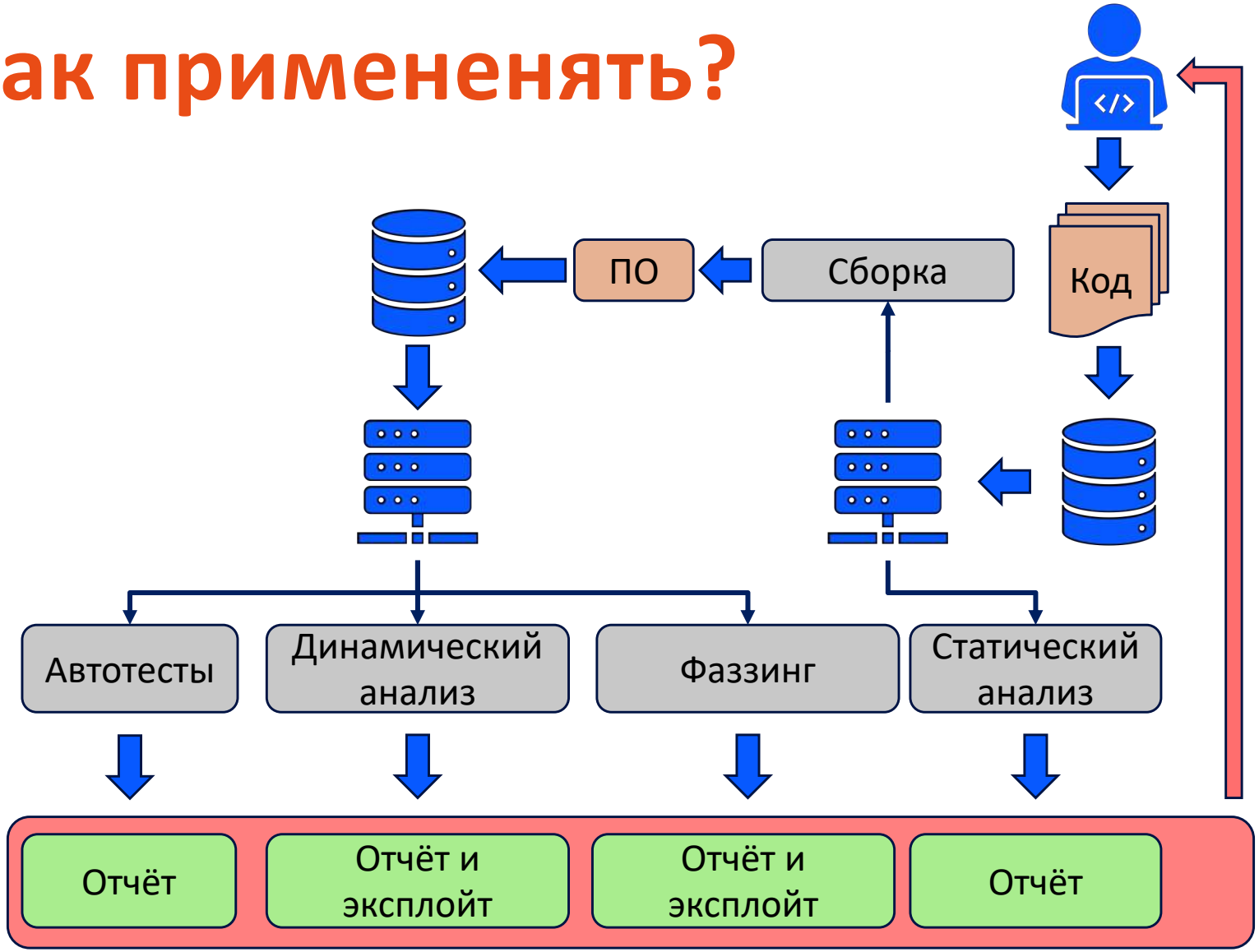
Проблемы:

- Ложные предупреждения об ошибках
- Пропуск ошибок

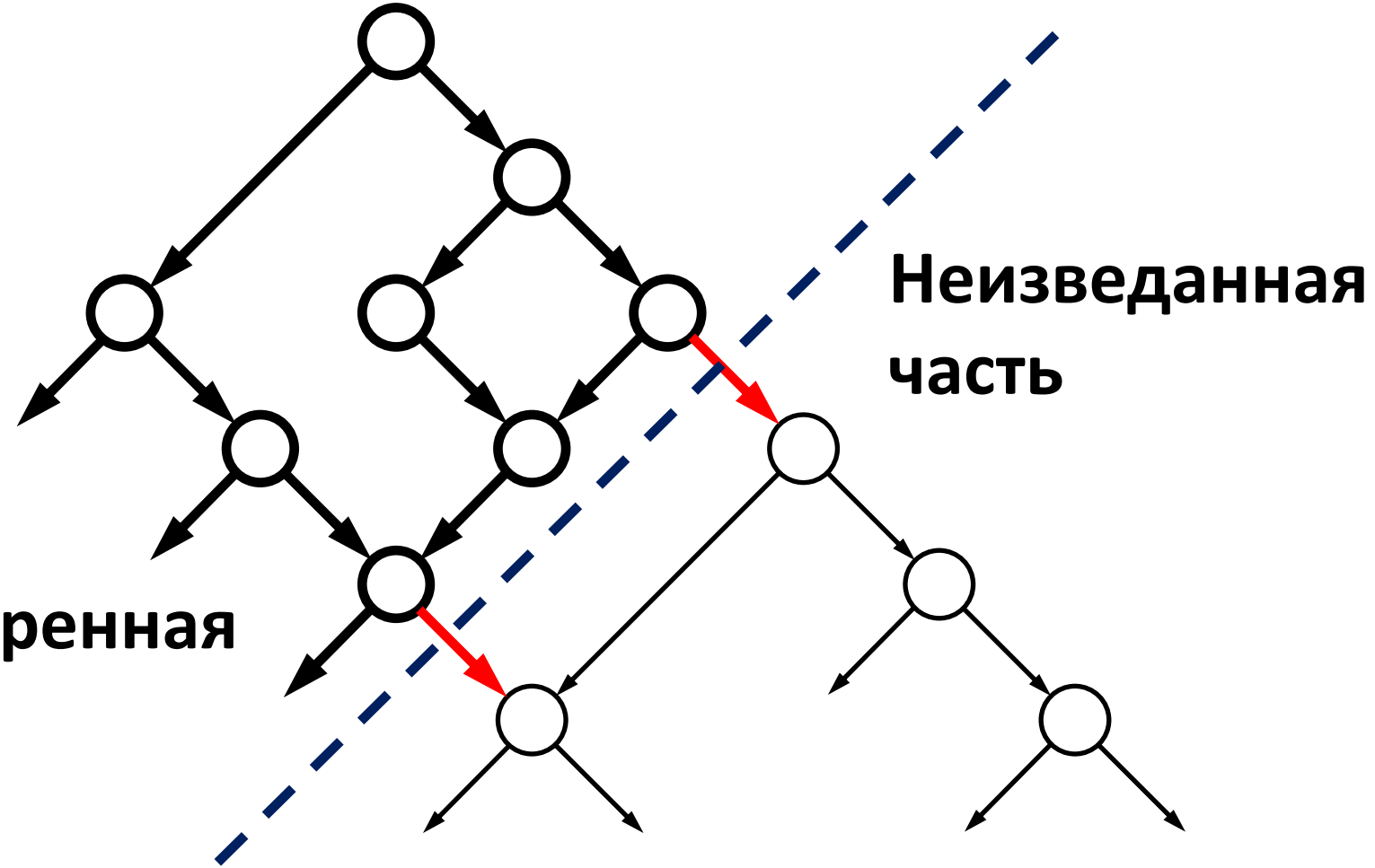
Преимущества:

- Производительность
- Масштабируемость

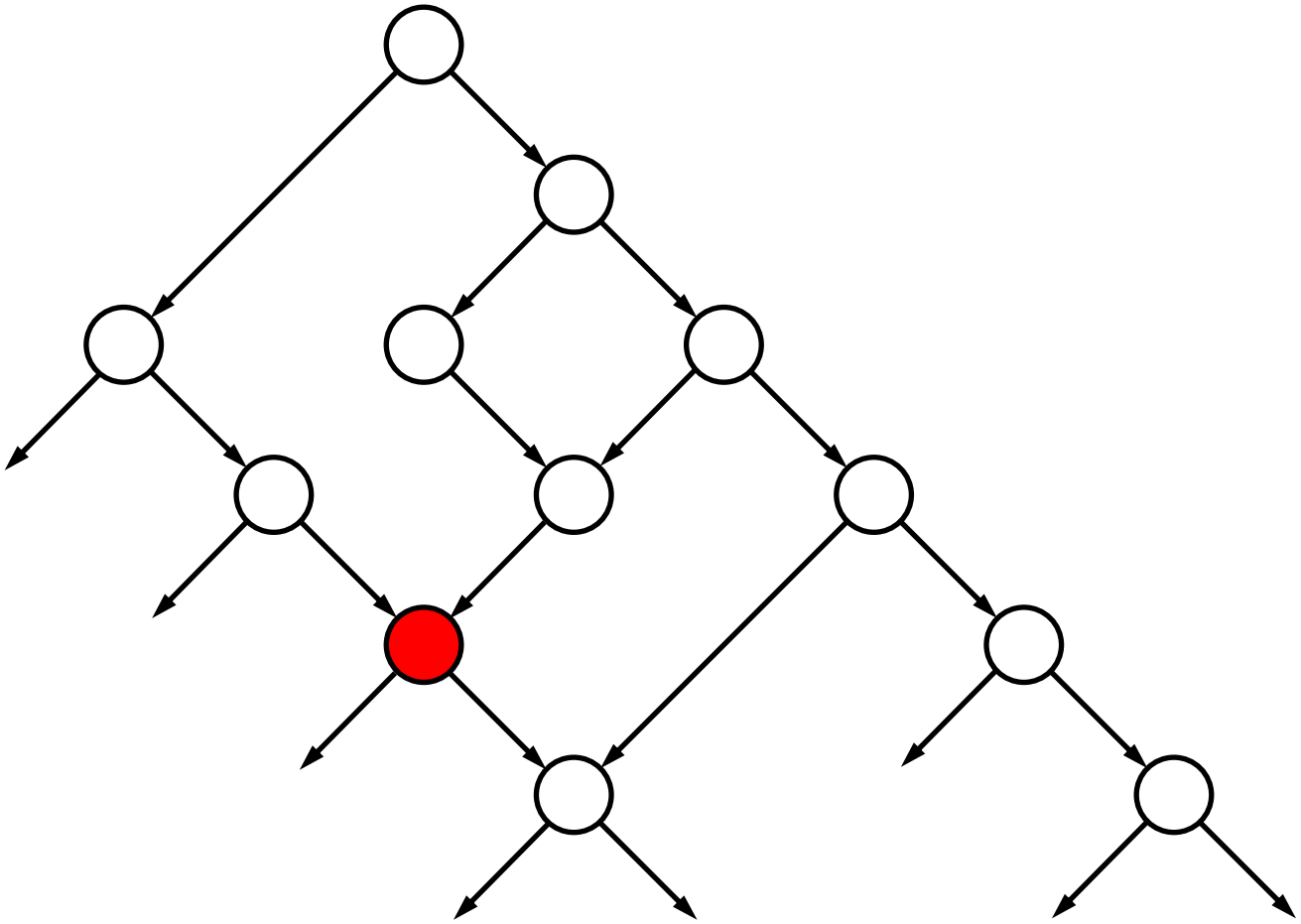
Как примененять?



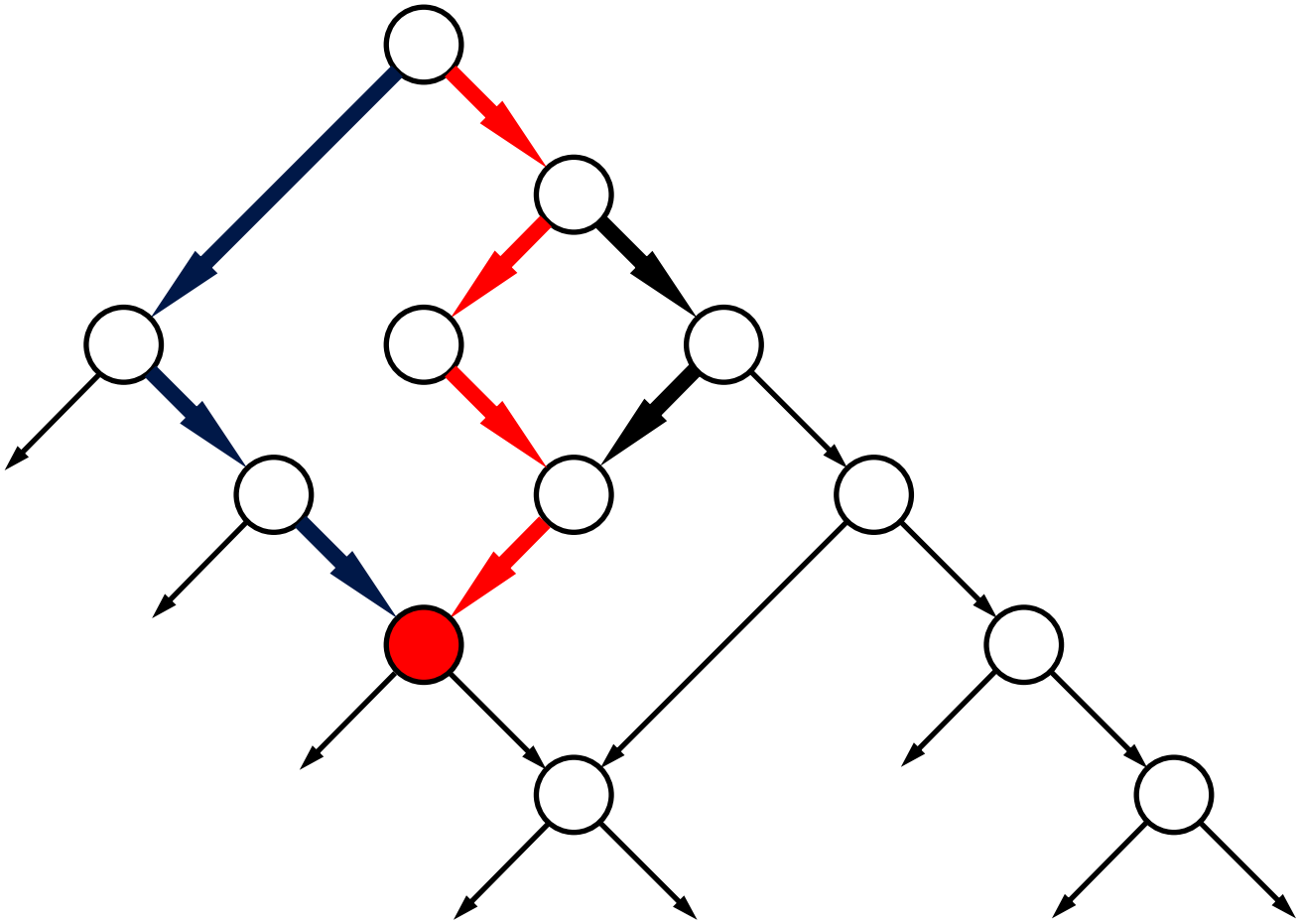
Комбинированные методы



Комбинированные методы



Комбинированные методы



Комбинированные методы

- Fuzzing + Динамическое символьное исполнение:
  - Driller
  - Crusher
- Статический анализ + динамическое символьное исполнение

Материалы

https://klee.github.io
https://s2e.systems
https://angr.io
https://github.com/google/AFL
https://github.com/shellphish/driller
https://www.synopsys.com/software-integrity/security-testing/static-analysis-sast.html
https://www.synopsys.com/software-integrity/security-testing/fuzz-testing.html
https://www.perforce.com/products/klocwork
https://www.ispras.ru/technologies

Контакты

Александр Герасимов

ИСП РАН

https://www.ispras.ru/groups/sp/staff.php

https://www.linkedin.com/in/alexander-gerasimov-9334767b/