Avalanche: Обнаружение ошибок при помощи динамического анализа

ИСП РАН, Сидоров Денис

Январь 2012

Динамический и статический анализ кода

- Динамический анализ - анализ программы во время выполнения
- Статический анализ - анализ без выполнения программы

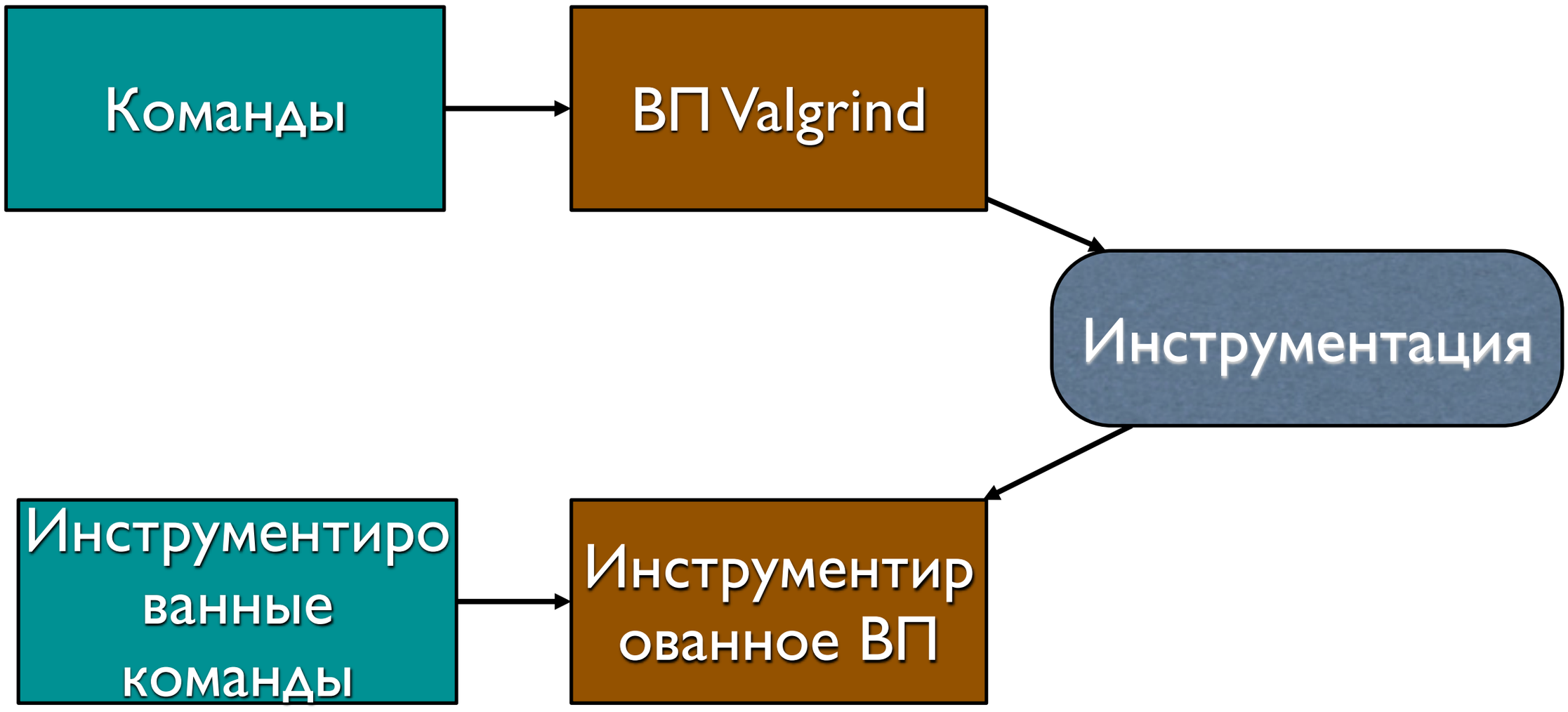
Обнаружение ошибок при помощи анализа программ

- Динамический анализ
  - Требуется набор входных данных и/или среда выполнения
  - Высокие требования к ресурсам
  - Высокая точность обнаружения
- Статический анализ
  - Работает на исходном или бинарном коде
  - Анализ абстрактной модели
  - Хорошая масштабируемость
  - Ложные срабатывания

Valgrind

- Фреймворк динамической инструментации
- Обнаруживаемые ошибки:
  - Утечки памяти
  - Ошибки работы с динамической памятью
  - Неинициализированные данные
  - Ошибки в многопоточных программах

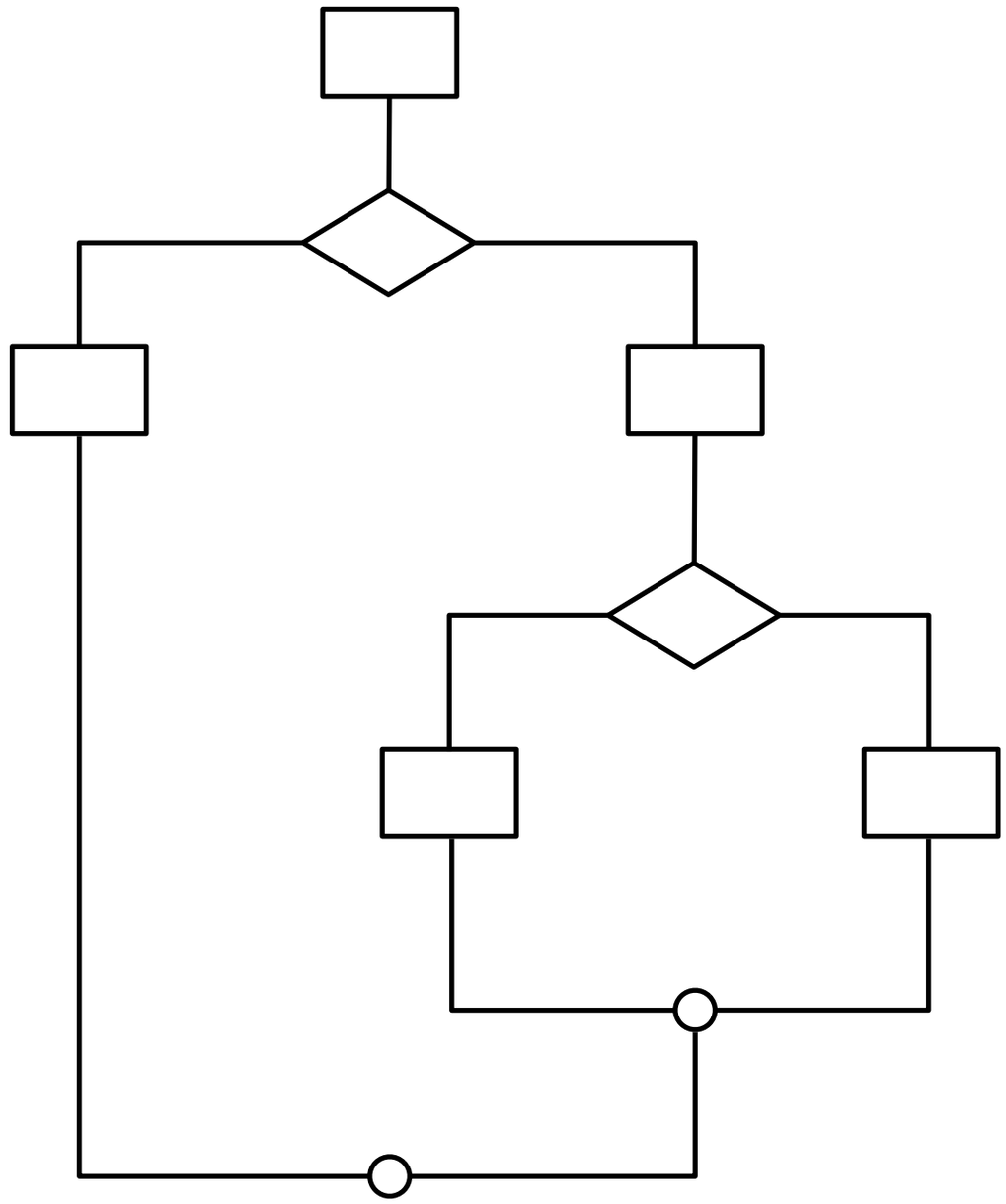
Valgrind: общая схема работы



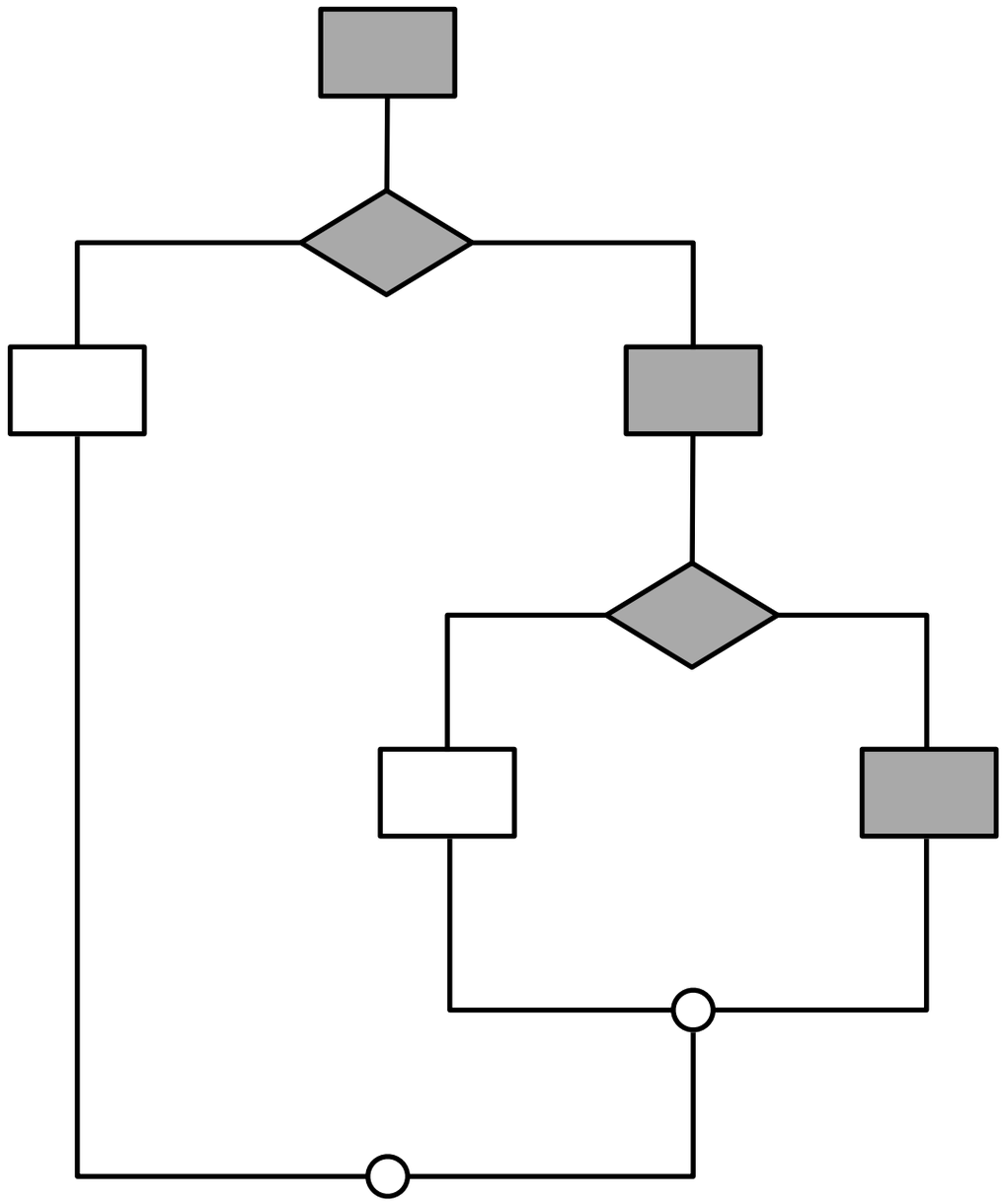
Внутреннее представление (ВП)

- SSA (Single Static Assignment form)
- ВП команд:
  - запись в регистр
  - запись по адресу
  - присваивание временным переменным
  - переходы
- ВП выражений:
  - константы
  - чтение из регистра
  - чтение по адресу
  - арифметические операции

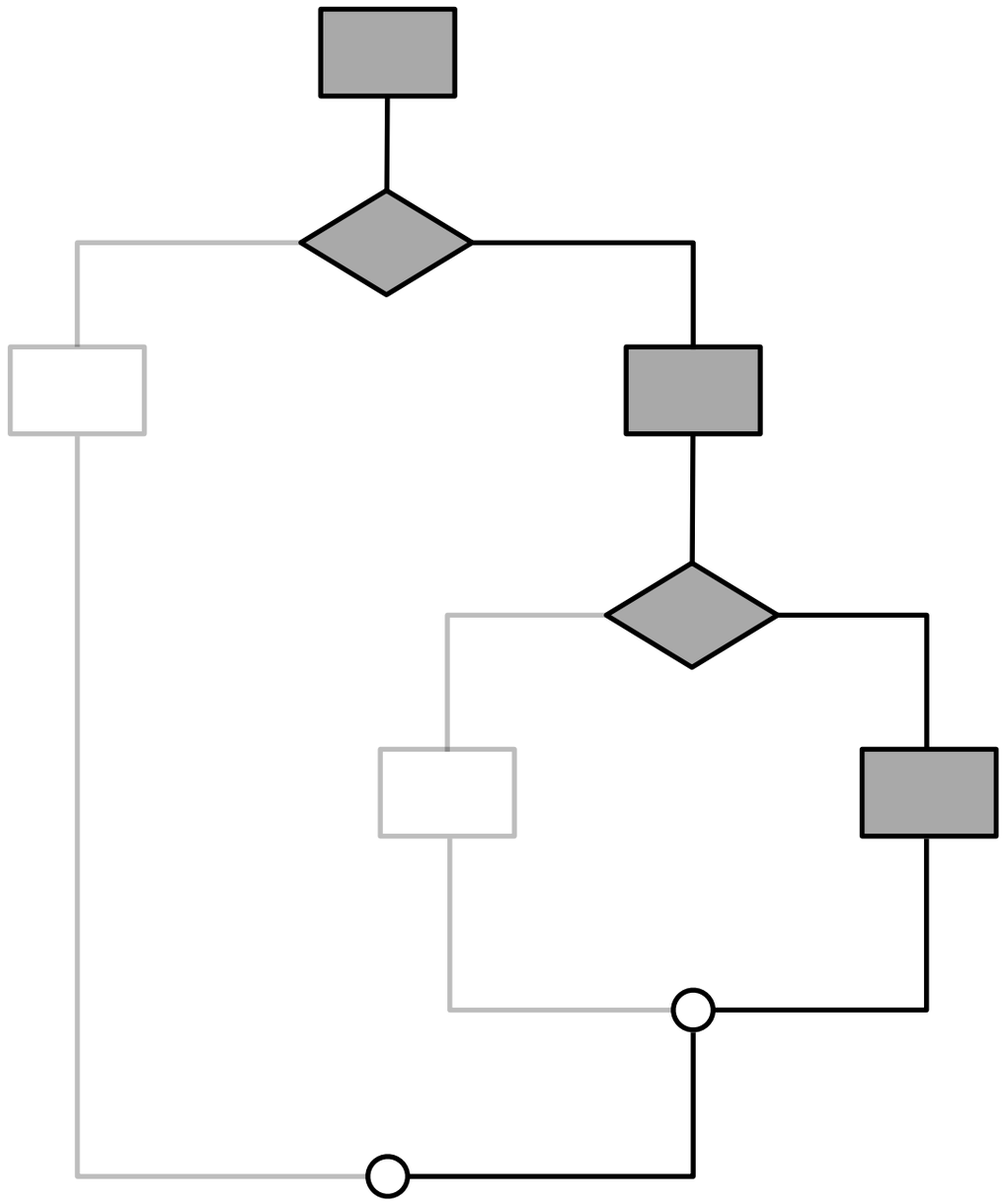
Трасса выполнения программы



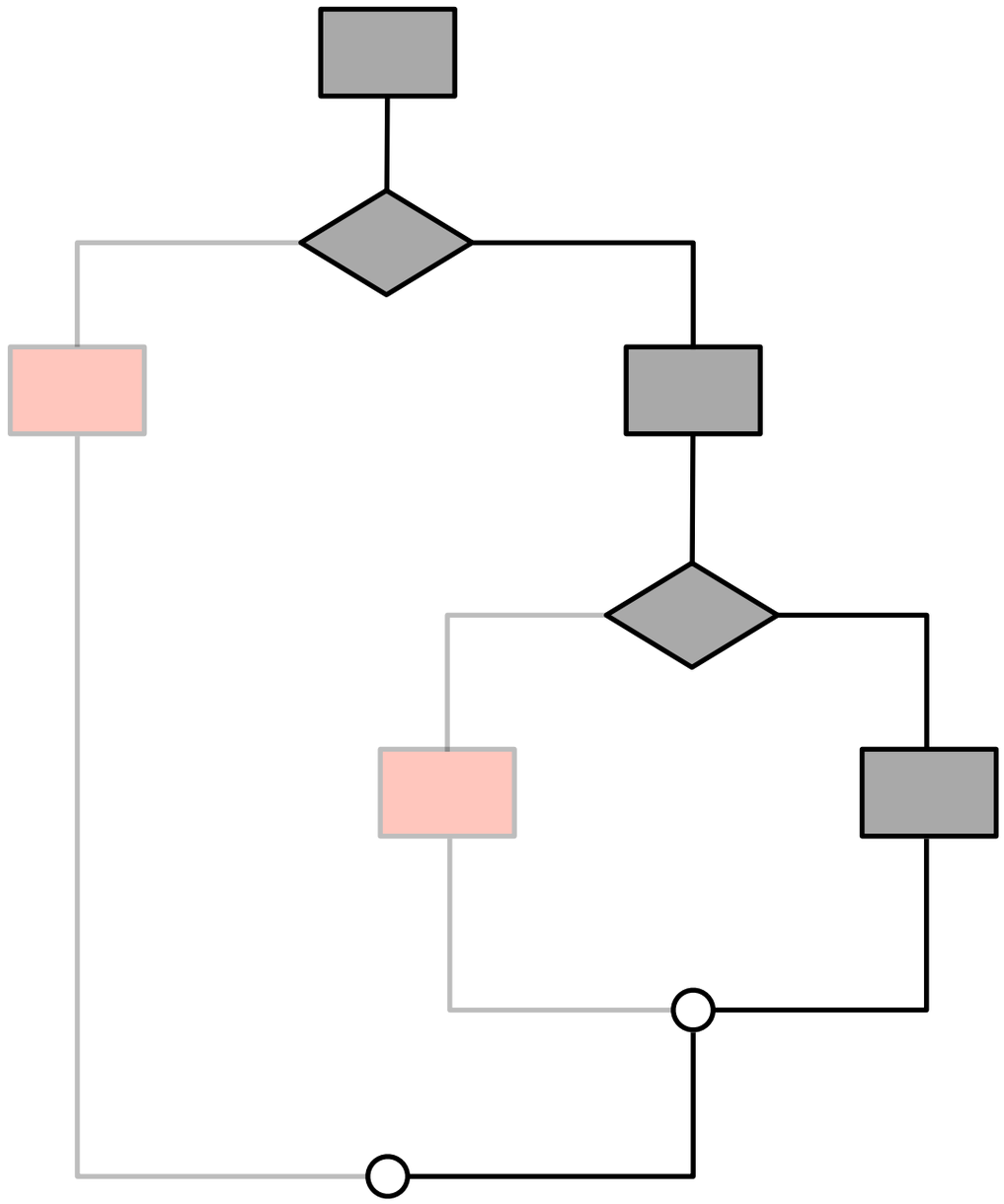
Трасса выполнения программы



Трасса выполнения программы



Трасса выполнения программы



Fuzz Testing

- Неправильные, случайные входные данные
- Flayer (Google) - обнаружены ошибки в OpenSSH и OpenSSL

Fuzz Testing: Контрпример

```
char *ptr = NULL;

FILE *f = fopen(filename, “r”);
char buf[3];

fread(buf, 3, 1, f);

if (buf[0] == ‘B’ && buf[1] == ‘A’ && buf[2] == ‘D’) {
    *ptr = ‘X’; /* CRASH! */
}
```

Работы в этой области

- EXE tool, Stanford University - символические вычисления
- SAGE framework, Microsoft Research - white-box fuzz testing
- KLEE, LLVM project

Неслучайные входные данные

Трасса выполнения	Система уравнений
Входные данные	Свободные переменные
Ветвления, потенциально опасные операции	Утверждения
Обход альтернативных путей	Инвертирование условий

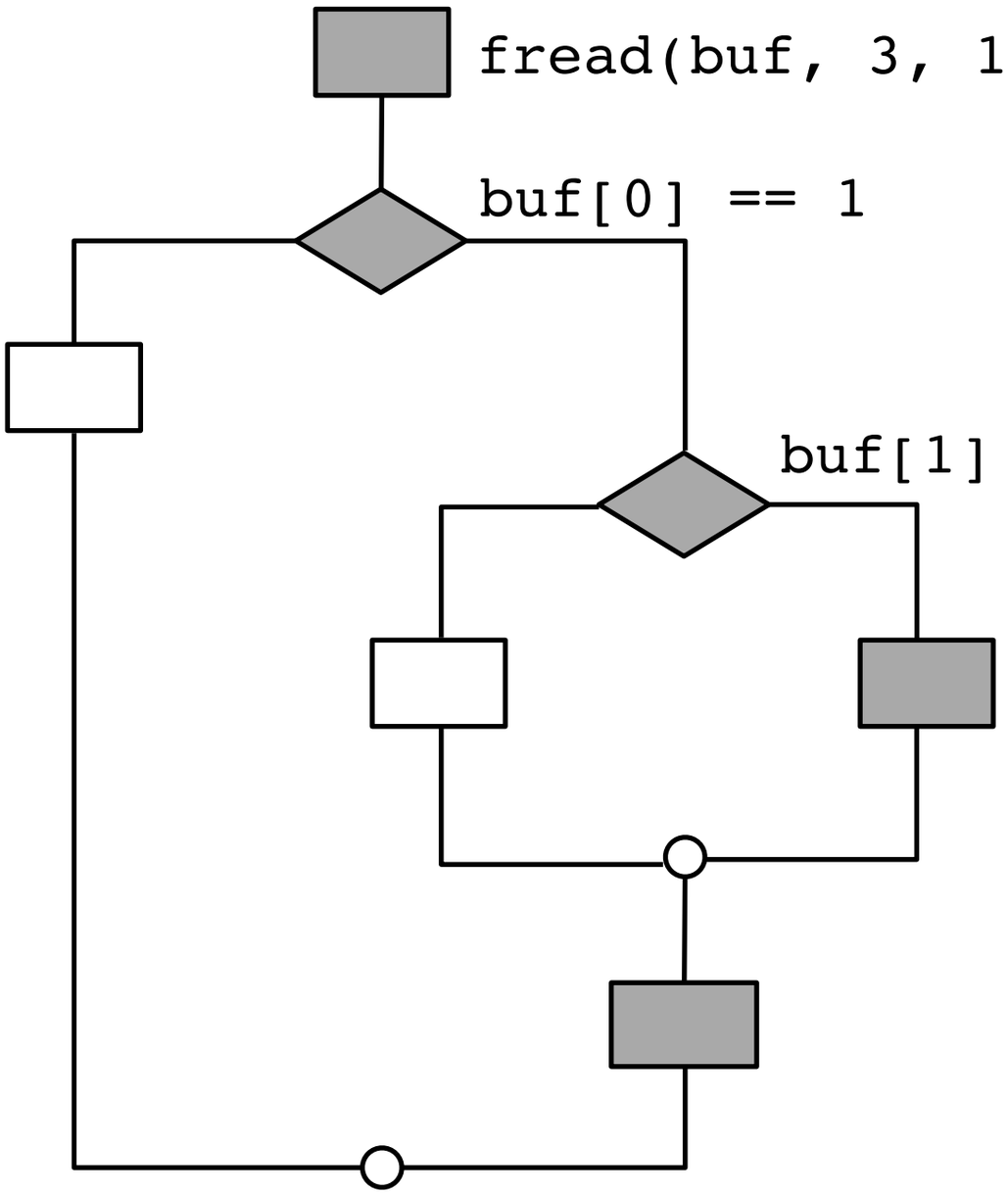
Пример

```
char *names[] = { “one”, “two”, ...};

char buf[3];
fread(buf, 3, 1, f);

if (buf[0] == 1) {
     int index;
     if (buf[1] + buf[2] > 0) {
          index = ...;
     }
     char *name = names[index];
     ...
} else {
     ...
}
```

Пример

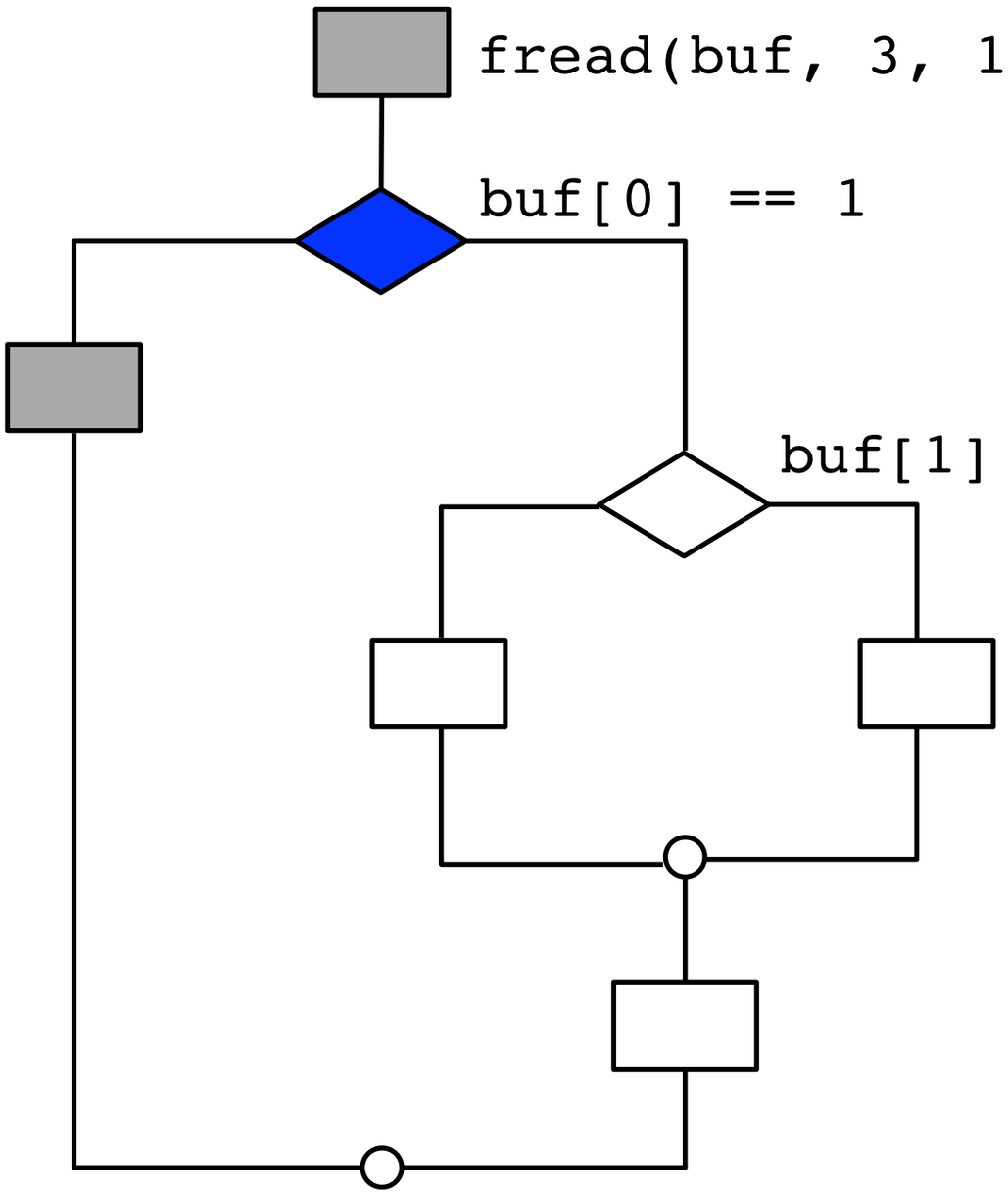


x1, x2, x3: byte

x1 = 1
x2 + x3 > 0

x1 = 1
x2 = 100
x2 = 200

Пример

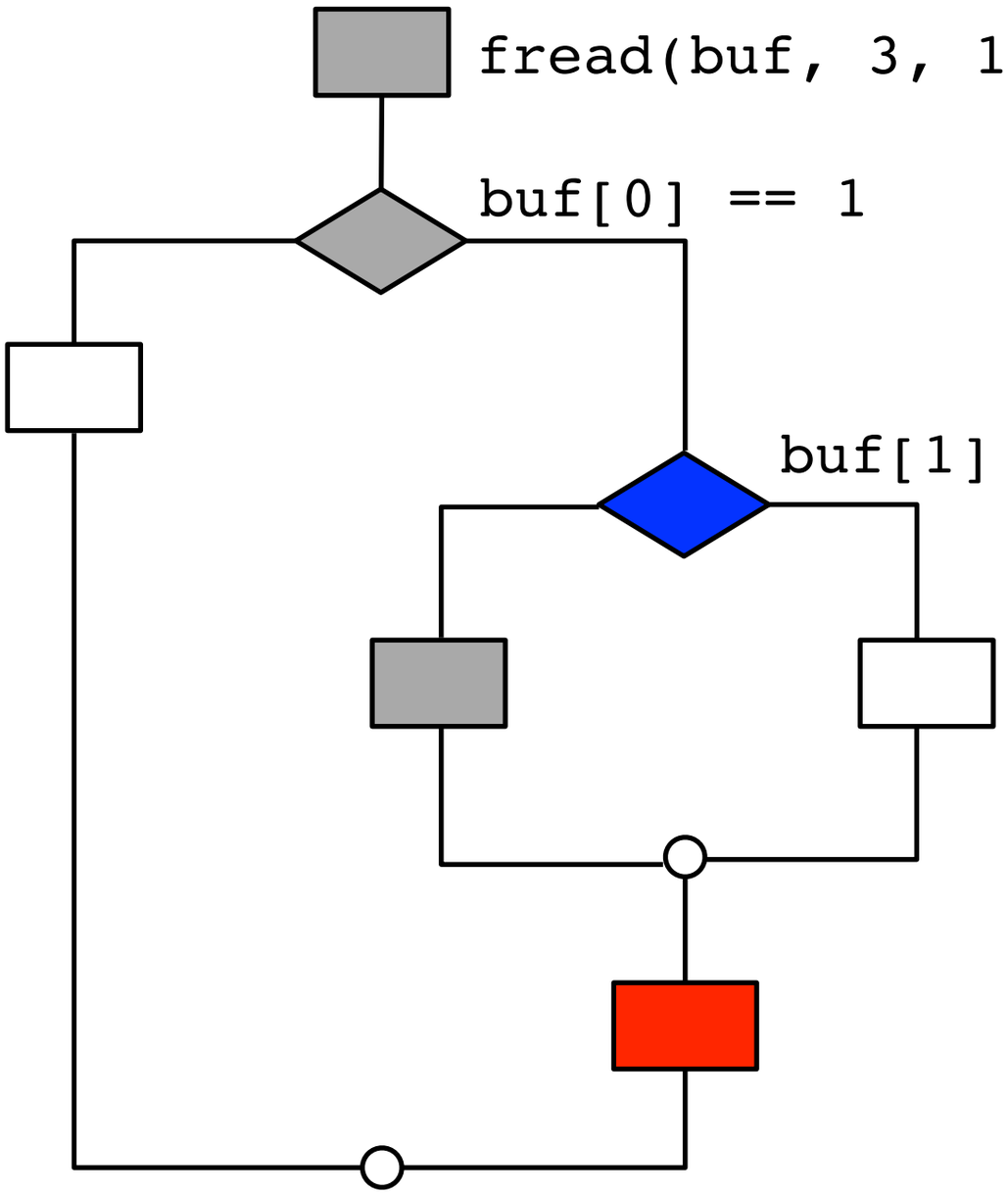


x1, x2, x3: byte

¬(x1 = 1)

x1 = 2
x2 = ...
x3 = ...

Пример



x1, x2, x3: byte

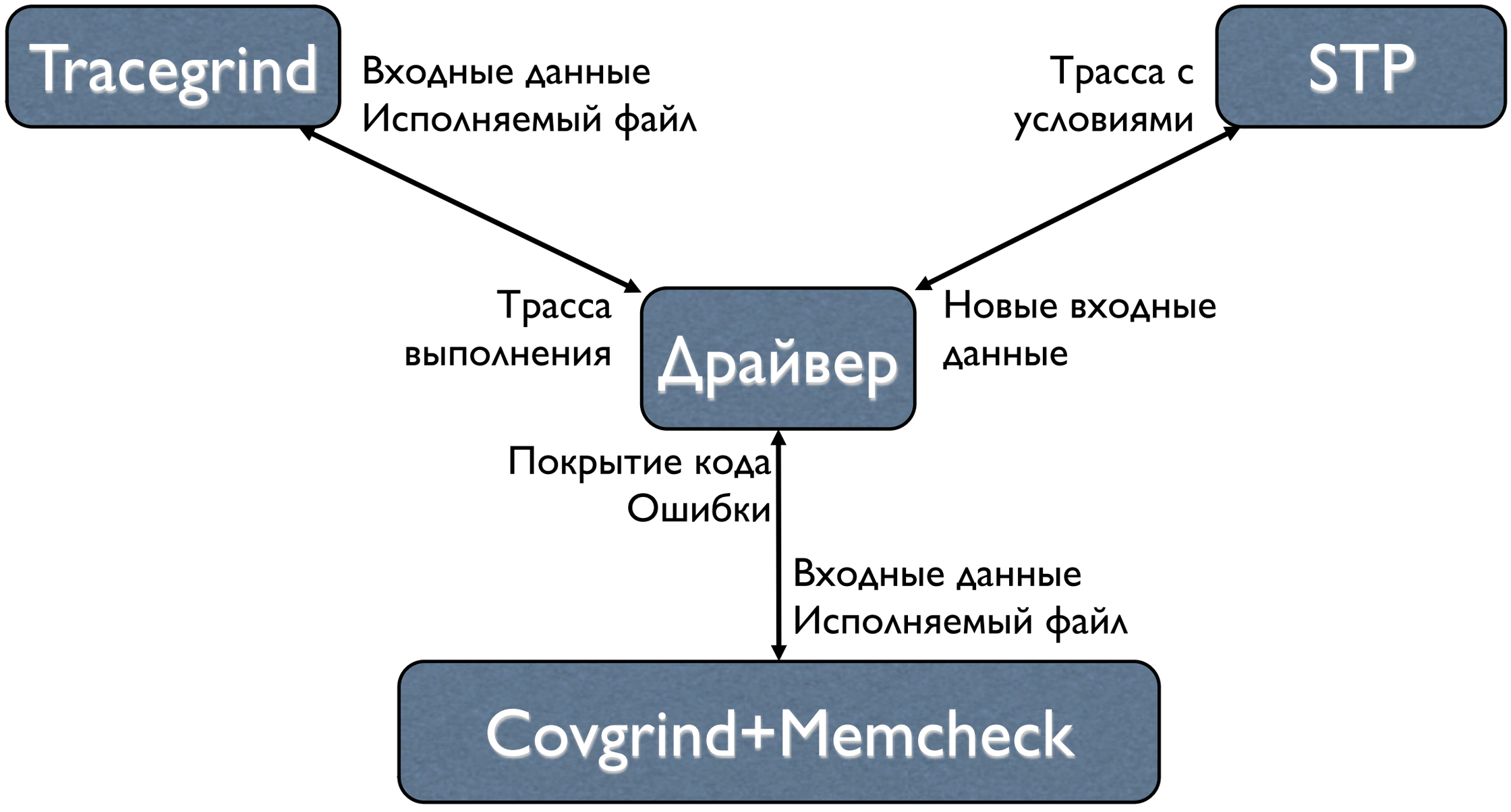
x1 = 1
¬(x2 + x3 > 0)

x1 = 1
x2 = 0
x3 = 0

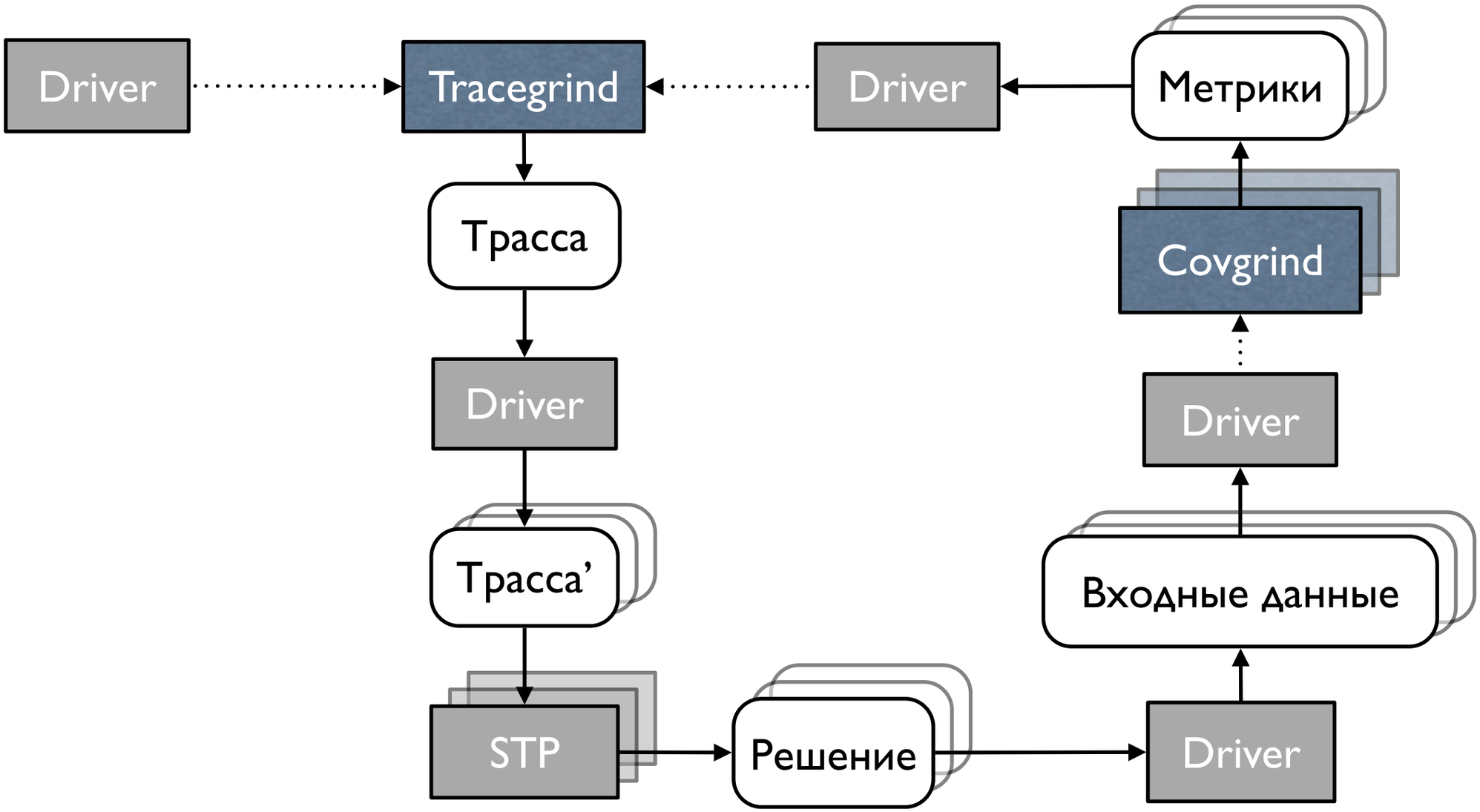
Avalanche

- Отслеживает поток помеченных (потенциально опасных) данных
- Изменяет входные данные, чтобы спровоцировать ошибку, или обойти новые части программы
- Обнаруживает критические ошибки - разыменование нулевого указателя, деление на ноль, неинициализированные данные, ошибки работы с памятью
- Генерирует набор входных данных для каждой найденной ошибки

Avalanche: Архитектура



Avalanche: итерация



Драйвер Avalanche

- Координация работы других компонентов
- Обход различных путей исполнения программы, инвертирование условий
- Поддержка параллельного и распределенного анализа

Tracegrind

- Отслеживает поток помеченных данных в программе
- Все данные прочитанные из внешних источников (файлы, сетевые сокеты, аргументы командной строки, переменные окружения)
- Переводит трассу выполнения в булевскую формулу (STP утверждения)

Tracegrind

- Моделирует оперативную память, регистры и временные переменные при помощи бит-векторов
- Моделирует команды при помощи операций и утверждений STP

Covgrind

- Вычисление эвристики для увеличения покрытия кода программы (количество новых ББ покрытых на текущей итерации)
- Перехват сигналов (обнаружение критических ошибок)
- Обнаружение ошибок работы с памятью при помощи Memcheck
- Обнаружение бесконечных циклов при помоши таймаутов

STP - Simple Theorem Prover

- SAT решатель (основан на MiniSat)
- Проект с открытым исходным кодом
- Поддерживает бит-векторы, широкий набор операций

Проект

- Опубликован на Google Code
  http://code.google.com/p/avalanche
- Лицензии:
  - Valgrind и Memcheck - GPL v2
  - STP - MIT license
  - Драйвер Avalanche, Tracegrind, Covgrind - Apache license

Avalanche: возможности

- Поддержка клиентских сетевых сокетов
- Поддержка переменных окружения и параметров командной строки
- Поддержка платформ x86/Linux и amd64/Linux, ARM/Linux, Android
- Поддержка фильтров по коду и входным данным
- Поддержка параллельного и распределенного анализа

Результаты

- Более 15-ти ошибок на проектах с открытым исходным кодом
- Ошибки подтверждены и/или исправлены разработчиками

Проект	Тип ошибок
mencoder	NPD
wget	NPD
swtool	NPD
libmpeg2	DBZ
gnash	UE
audiofile	IL
libsndfile	DBZ
libjpeg	DBZ
libmpeg3	NPD
libquicktime	NPD, IL
libwmf	NPD
mono	NPD
parrot	NPD, IL
llvm	NPD

Null Pointer Dereference (разуменование нулевого указателя) = NPD
Division By Zero (деление на ноль) = DBZ
Unhandled Exception (необработанная исключительная ситуация) = UE
Infinite Loop (бесконечный цикл) = IL

Планы на будущее

- Улучшение прозводительности
- Поддержка новых источников входных данных (серверные сетевые сокеты, и т. д.)
- Поддержка новых типов ошибок (многопоточные приложения)