Разработка средств защиты информации

Лабораторная 2

Боязитов Вадим

932125

## 1. Провести эксперимент над  примерами с большим объемом данных.

### Доступный пул памяти значительно превышает фактически запрашиваемый объем

В рамках файла стресс теста задаем параметры

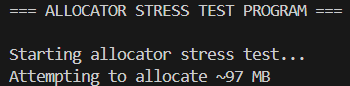
Allocator stress\_test\_allocator\_(1024, 1000000);

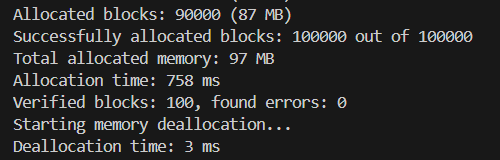
const int kNumBlocks = 100000;

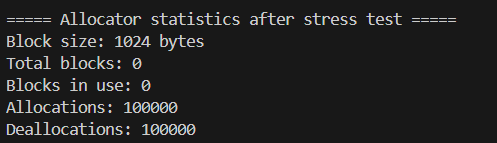
const int kBlockSize = sizeof(TestBlock);

const int kMegabytesToAllocate = kNumBlocks \* kBlockSize / (1024 \* 1024);

Вывод







Результат, все корректно отработало.

### Тестирование граничных условий

В рамках файла стресс теста задаем параметры

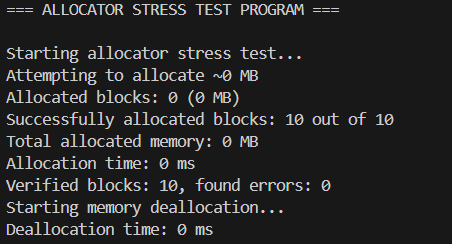
Allocator stress\_test\_allocator\_(1024, 10);

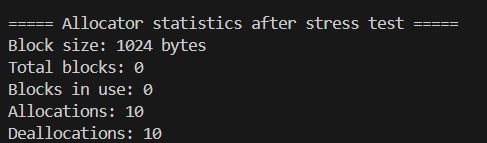
const int kNumBlocks = 10;

const int kBlockSize = sizeof(TestBlock);

const int kMegabytesToAllocate = kNumBlocks \* kBlockSize / (1024 \* 1024);

Вывод





Результат, все корректно отработало.

### Тест на переполнение

В рамках файла стресс теста задаем параметры

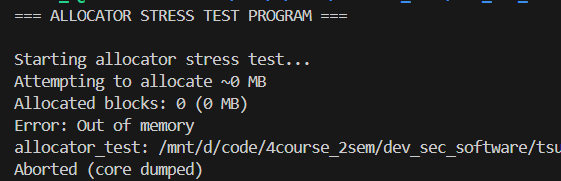
Allocator stress\_test\_allocator\_(1024, 100);

const int kNumBlocks = 150;

const int kBlockSize = sizeof(TestBlock);

const int kMegabytesToAllocate = kNumBlocks \* kBlockSize / (1024 \* 1024);

Вывод

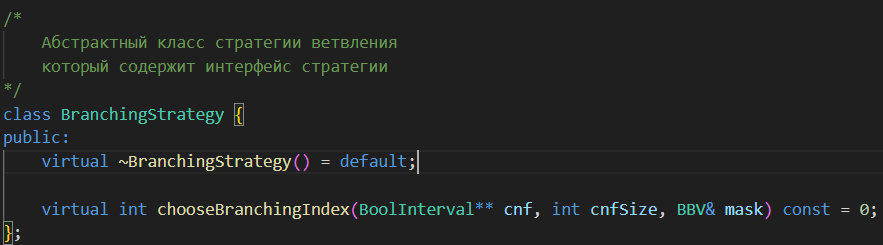


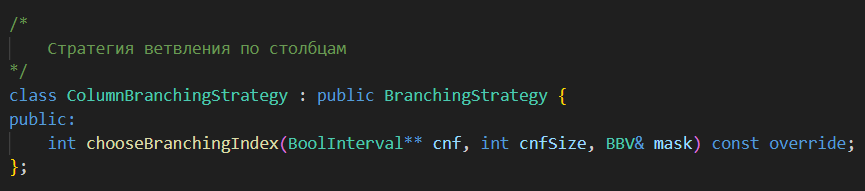
Результат, отловлена ошибка.

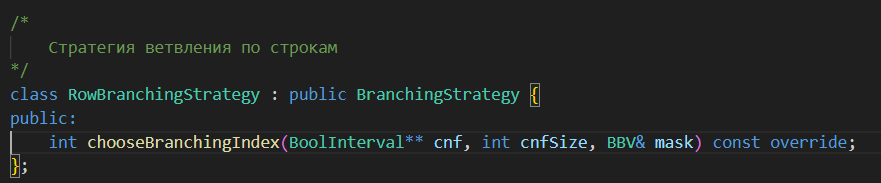
## 2. Обеспечить архитектурную возможность изменения правила выбора переменной ветвления (использовать паттерн "Стратегия") для SAT задачи.

### Шаги реализации паттерна

Создать интерфейс стратегий, описывающий этот алгоритм. Он должен быть общим для всех вариантов алгоритма.







### Описание реализаций стратегий

#### Стратегия ветвления по столбцам

Если в столбце мало '-'(отрицаний), значит, переменная чаще встречается в положительной форме, и её выбор может быстрее привести к упрощению формулы.

1. Собрать все индексы переменных, которые не замаскированы (mask[i] == 0)

2. Если таких переменных нет → вернуть -1

3. Для каждой строки (интервала) в CNF:

Если это первая строка:

Для каждого доступного столбца:

если значение равно '-', то записать 1, иначе 0

Иначе:

Для каждого доступного столбца:

если значение равно '-', увеличить счётчик для этого столбца на 1

5. Найти столбец с минимальным количеством '-' и вернуть его индекс

#### Стратегия ветвления по столбцам

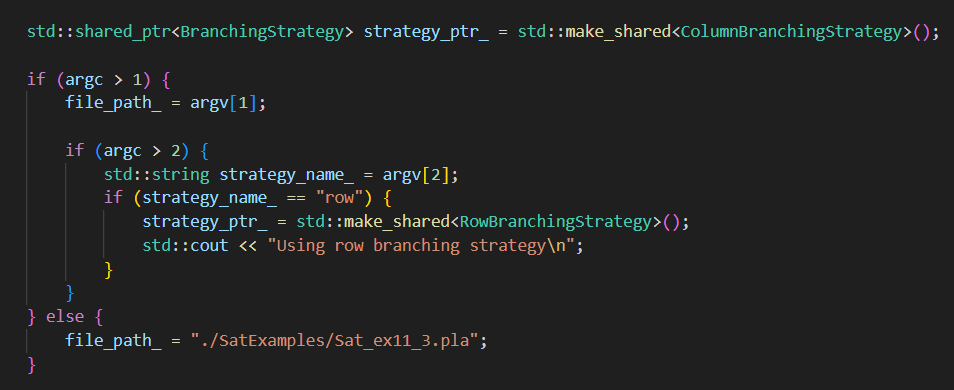
Ветвление по самой короткой строке (с минимальным числом активных переменных) может быстрее привести к упрощению КНФ.

1. Собрать все непустые строки из CNF и посчитать их вес:  
 Вес = количество незамаскированных переменных, которые не равны '-'  
  
2. Если нет подходящих строк → вернуть -1

3. Найди строку с минимальным ненулевым весом  
  
4. В этой строке найди первый незамаскированный столбец, значение которого не '-'  
 Если такой столбец найден → вернуть его индекс

5. Если метод не помог, то вызвать алгоритм по столбцам

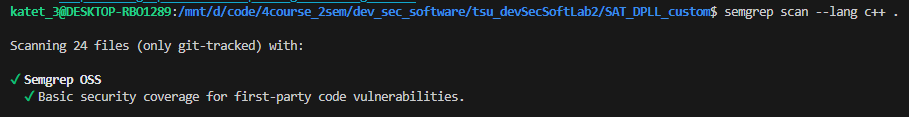
### Пример работы с выбором стратегии

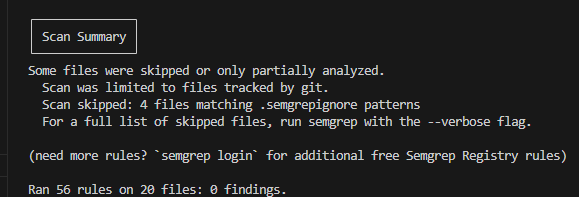


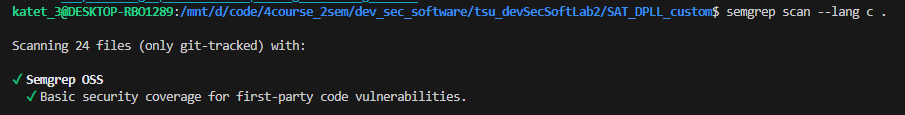
## 3. Исследовать пользовательский проект на уязвимости с помощью любого доступного статического анализатора, например pvs-studio , а также с помощью Valgrind динамического анализа.

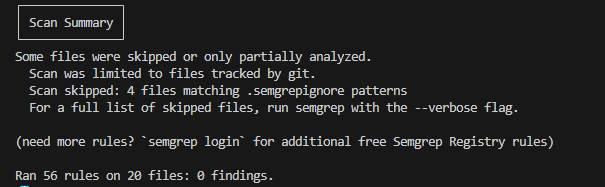
### Статический анализатор Semgrep

Используются стандартные правила.

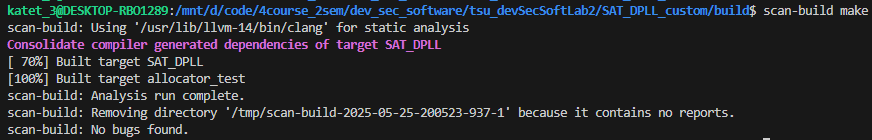


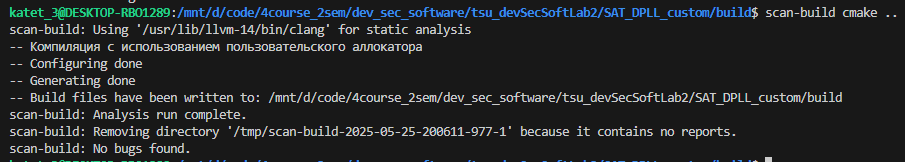


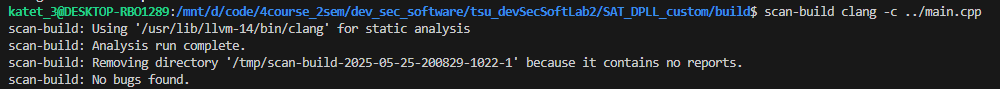


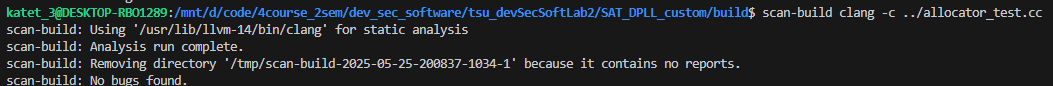


### Статический анализатор Clang

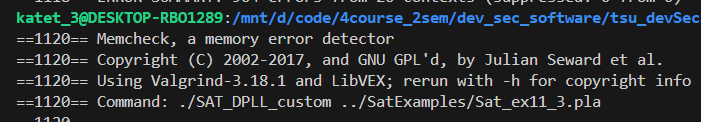




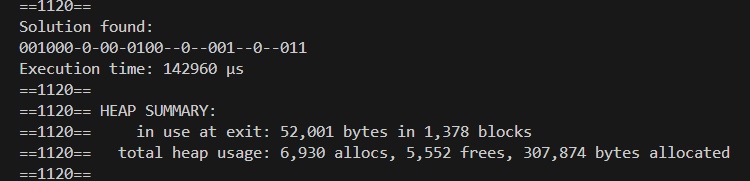


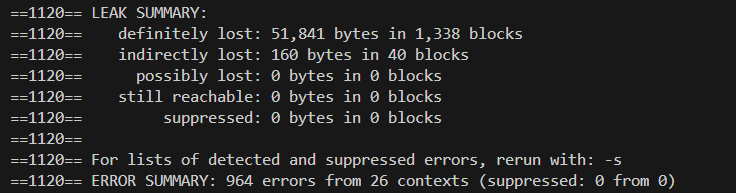


### Динамический анализатор Valgrind

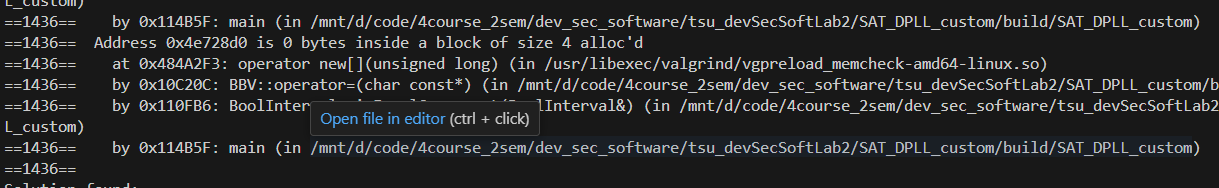


После выполнения смотрим отчет

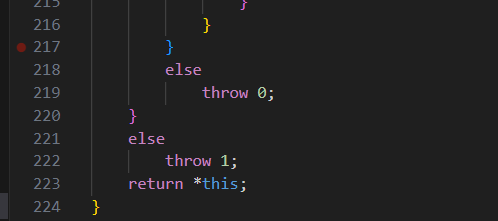
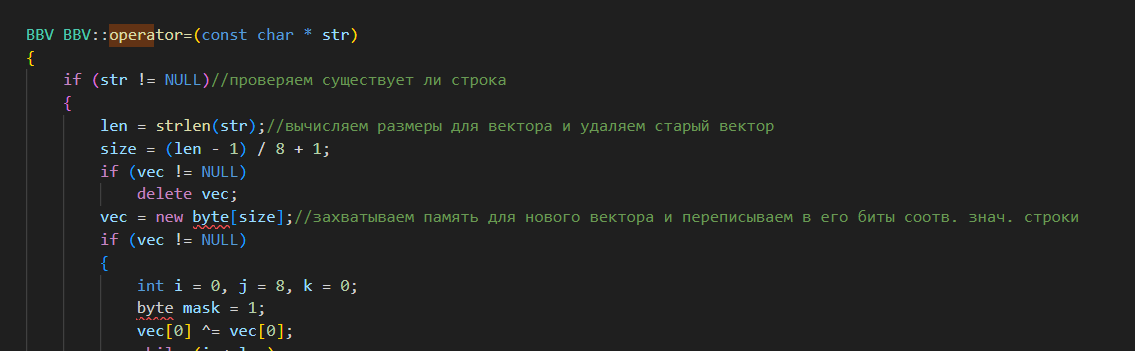




Разберем одну из заявленных ошибок



Проблема, обнаруженная Valgrind, связана с некорректным управлением памятью в операторе присваивания BBV::operator=(const char\*).



Проблемы этого кода:

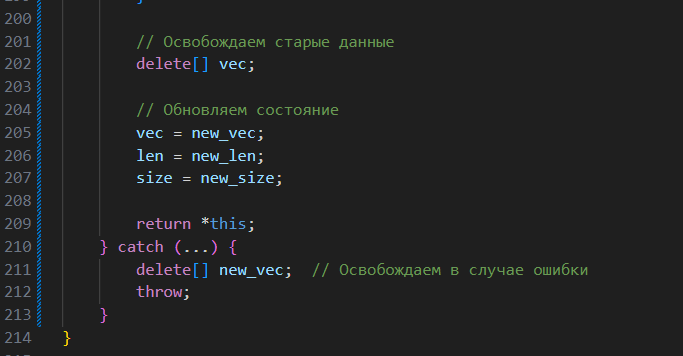
Используется new byte[size], но освобождается через delete вместо delete[]

Нет проверки на успешное выделение памяти после new

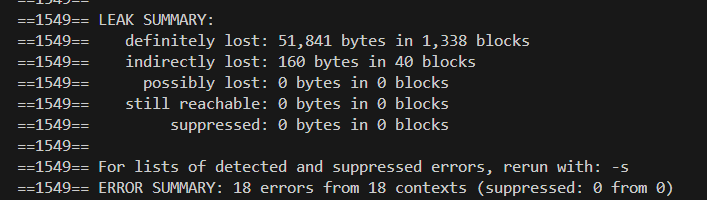
Если new выбрасывает исключение, старые данные уже удалены

Код после изменения будет иметь вид:





Тестируем



Результат на глаза, на сколько меньше стало ошибок.