

С++ линтеры — хорошо, но недостаточно



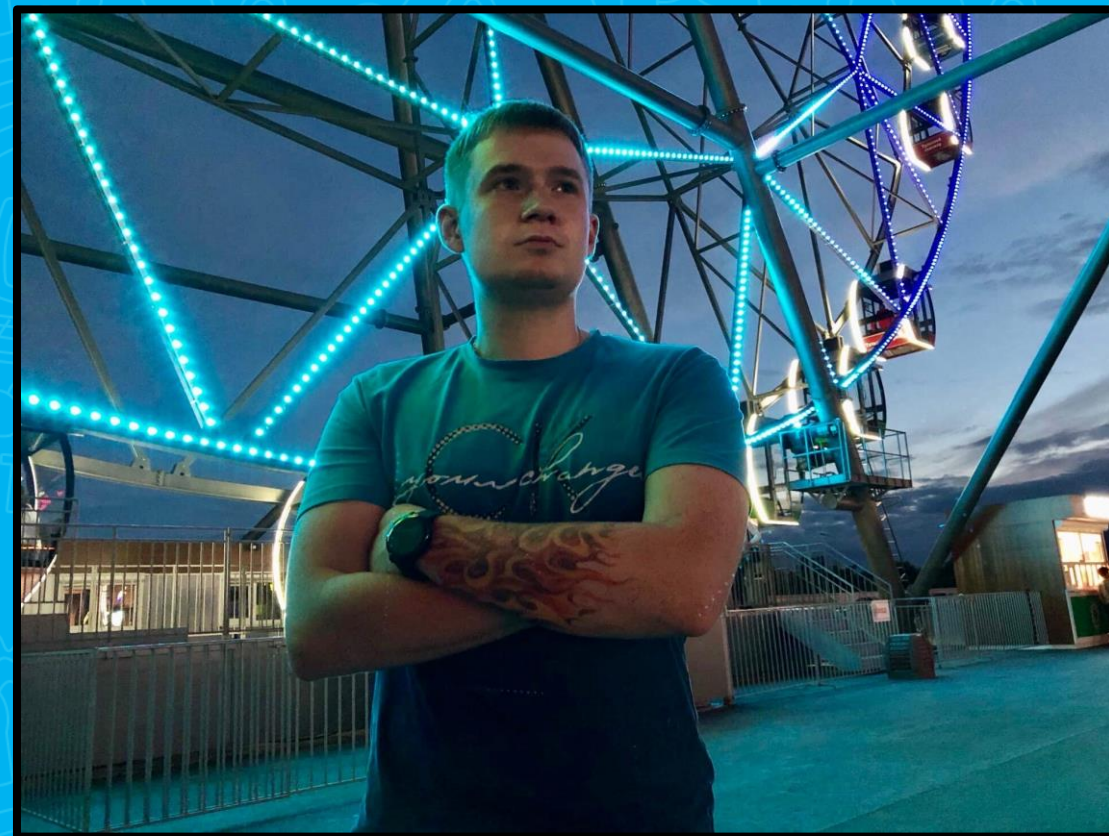
Евгений Фёклин
C++ developer

Докладчик

Евгений Фёклин

feklin@viva64.com

- Разработчик в C++ команде PVS-Studio
- Разработчик игр



Линтеры, или с чего начинался статический анализ кода

Что такое линтеры?

- Линтер – это программа, которая помогает программистам улучшить качество своего кода.
- Проверяет исходный код на соответствие стилевым правилам. Выявляет простые потенциальным ошибки и неэффективные практики.

Запрещённые функции, именованя и т.д.

- MISRA C: **exit, system**
- Стили: переменные и функции с большой буквы
- '**strcpy**': This function or variable may be unsafe. Consider using **strcpy_s** instead.

Ошибка в условии

```
int main() {  
    int x = 10;  
    if (x = 5) {  
        std::cout << "x is 5" << std::endl;  
    }  
  
    return 0;  
}
```

Нет ошибки в условии

```
int main() {  
    int x = 10;  
    if (x == 5) {  
        std::cout << "x is 5" << std::endl;  
    }  
  
    return 0;  
}
```

Неиспользуемая переменная

```
int main() {  
    int x = 10;  
    int y = 20;  
  
    std::cout << "x = " << x << std::endl;  
  
    return 0;  
}
```


Красота? Вот только чуть сложнее, и ...

- if (a + 1 < a + 1)
- if (a + 1**U** < a + 1**u**) // Но при этом надо различать A и a
- if (a + **0x1** < a + **1**)
- if (a + 1 **/*foo*/** < a + 1)
- if (**(a + 1)** < a + 1)
- if (**(a)** + 1 < a + 1)
- if (a + 1 < a + **((1)))**)

Проблематика подхода в лоб

- Линтеры могут применять жёсткие правила к коду, что может привести к недопониманию его смысла и намерений разработчика.

Способ решения: синтаксическое дерево

- С деревом проще решать подобные задачи
 - Легко пропускать скобки
 - В разном порядке сравнивать
 - Легко понять, что $1u$ и $1U$ одно и то же
 - В целом легче делать различные исключения

Ограничения регулярных выражений

- Не раскрываются макросы
- Не раскрываются `#include`

```
PUGI__FN bool set_value_convert(char_t*& dest, uintptr_t& header,  
                               uintptr_t header_mask, int value)  
{  
    char buf[128];  
    sprintf(buf, "%d", value);  
  
    return set_value_buffer(dest, header, header_mask, buf);  
}
```

```
#define sfstream std::fstream
#define schar char
#define suchar unsigned schar
#define sprintf std::printf
#define satof atof
#define satoi atoi
```

```
PUGI__FN bool set_value_convert(char_t*& dest, uintptr_t& header,
                               uintptr_t header_mask, int value)
{
    char buf[128];
    sprintf(buf, "%d", value);

    return set_value_buffer(dest, header, header_mask, buf);
}
```

PVS-Studio: V614 Uninitialized buffer 'buf' used. pugixml.cpp 3362

Решение: препроцессирование

- Процесс раскрытия в исходном коде директив компилятора и подстановка значений макросов
- Преобразованный код становится более пригодным для анализа и понимания

Важность сбора семантической информации

- Не зная тип x и y нельзя сказать, есть здесь ошибка или нет

```
unsigned x = ((rectP.right-rectP.left) - width) / 2 +  
            rectP.left;
```

```
unsigned y = ((rectP.bottom-rectP.top) - height) / 2 +  
            rectP.top;
```

```
if(x < 0) x = 0;
```

```
if(y < 0) y = 0;
```

PVS-Studio: V547 Expression 'y < 0' is always false. Unsigned type value is never < 0. fceux gui.cpp 33

А вот выход за границу массива?

```
float A[10];  
for (size_t i = 0; i < sizeof(A); i++)  
{  
    // Известно, что значение i лежит в пределах [0..39].  
    A[i] = 1.0; // Сообщение о выходе за границу массива.  
}
```

Добро пожаловать в мир анализа потока данных

- Диапазон
- Множество
- Точное значение
- Указатель на буфер размера N байт
- Нулевой указатель / Ненулевой указатель
- Валидный указатель / Освобождённый указатель
- И т.д.

```
static const int kDaysInMonth[13] = {  
    0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31  
};
```

```
bool ValidateDateTime(const DateTime& time) {  
    if (time.year < 1 || time.year > 9999 ||  
        time.month < 1 || time.month > 12 ||  
        time.day < 1 || time.day > 31 ||  
        time.hour < 0 || time.hour > 23 ||  
        time.minute < 0 || time.minute > 59 ||  
        time.second < 0 || time.second > 59) {  
        return false;  
    }  
    if (time.month == 2 && IsLeapYear(time.year)) {  
        return time.month <= kDaysInMonth[time.month] + 1;  
    } else {  
        return time.month <= kDaysInMonth[time.month];  
    }  
}
```

Protobuf, C++

```
static const int kDaysInMonth[13] = {  
    0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31  
};
```

```
bool ValidateDateTime(const DateTime& time) {  
    if (time.year < 1 || time.year > 9999 ||  
        time.month < 1 || time.month > 12 ||  
        time.day < 1 || time.day > 31 ||  
        time.hour < 0 || time.hour > 23 ||  
        time.minute < 0 || time.minute > 59 ||  
        time.second < 0 || time.second > 59) {  
        return false;  
    }  
    if (time.month == 2 && IsLeapYear(time.year)) {  
        return time.month <= kDaysInMonth[time.month] + 1;  
    } else {  
        return time.month <= kDaysInMonth[time.month];  
    }  
}
```

PVS-Studio: V547 / CWE-571 Expression 'time.month <= kDaysInMonth[time.month] + 1' is always true. time.cc 83 (+ аналогичное на проверку ниже)

А если посчитать нельзя?

```
void F(int X)
{
    int A = X;
    int B = X + 10;
    int Q[5];
    Q[B - A] = 1;
}
```

PVS-Studio: V557 CWE-787 Array overrun is possible. The 'B - A' index is pointing beyond array bound. test.cpp 126

Символьное выполнение

```
void F(int A, int B, int C)
{
    if (A < B)
        if (B < C)
            if (A < C)
                foo();
}
```

А вот такой код ошибочен или нет?

```
int main(int argc, char **argv)
{
    ....
    QByteArray arg(argv[a]);
    ....
    arg = arg.mid(1);
    arg.toLower();
    if (arg == "o")
        ....
}
```

QByteArray QByteArray::toLowerCase() const

PVS-Studio: V530 The return value of function 'toLowerCase' is required to be utilized. main.cpp
1522

А вот такой код ошибочен или нет?

```
int main(int argc, char **argv)
{
    ....
    QByteArray arg(argv[a]);
    ....
    arg = arg.mid(1);
    arg = arg.toLowerCase();
    if (arg == "o")
        ....
}
```

PVS-Studio: V530 The return value of function 'toLowerCase' is required to be utilized. main.cpp
1522

Аннотирование функций

```
int foo() {  
    return (rand() % 2 == 0) ? 10 : 20;  
}
```

```
int main(void) {  
    int A[15];  
    A[foo()] = 1;  
}
```

PVS-Studio: V557 Array overrun is possible. The value of 'foo()' index could reach 20.

А теперь соберём всё вместе

- Сопоставление с шаблоном (родственник регулярок)

- Но теперь мы знаем много информации
 - Как работают функции
 - Типы переменных
 - Значения
 - И т.д.

**Можно строить сложные
диагностики**

```
void appDebugger::CopyDump(const char * src, DWORD from,
                           const char * dst) {
    FILE * file = fopen(src, "rb");
    if (file) {
        ...
        if(size > from) {
            FILE * to = fopen(dst, "a+b");
            if(to) {
                char buf[128];
                while (from < size)
                {
                    ...
                    fread(buf, s, 1, file);
                    fwrite(buf, s, 1, file);
                    ...
                }
            }
        }
    }
}
```

Приключения
капитана Блада

PVS-Studio: V1075 The function expects the file to be opened for writing, but it was opened for reading. Debugger.cpp 172

```
void appDebugger::CopyDump(const char * src,
                           DWORD from, const char * dst) {
    FILE * file = fopen(src, "rb");
    if (file) {
        ...
        if(size > from) {
            FILE * to = fopen(dst, "a+b");
            if(to) {
                char buf[128];
                while (from < size)
                {
                    ...
                    fread(buf, s, 1, file);
                    fwrite(buf, s, 1, to);
                    ...
                }
            }
        }
    }
}
```

// <=

Приключения
капитана Блада

В PVS-Studio размечено около 8000 C и C++ функций

- стандартная библиотека C
- стандартная библиотека шаблонов (STL)
- UE4 / UE5
- WinAPI
- glibc (GNU C Library)
- Qt
- MFC
- и т.д.

Это всё?

- Нет!
- Межпроцедурный анализ
- Способность статического анализатора раскрывать точки вызова функций и учитывать влияние таких вызовов на состояние программы и переменных в локальном проверяемом контексте.

Это всё?

- Нет!
- Межмодульный анализ
- Позволяет дать информацию анализатору о полной структуре проекта, делая анализ более точным и качественным.

Это всё?

- Нет!
- Taint-анализ
- Методика, позволяющая отследить распространение по программе внешних непроверенных данных (например, XML)

Опять-таки не всё, но хватит

- Свои разметки

- CWE – потенциальные уязвимости

- Разные стандарты классификации:
 - MISRA
 - CERT Coding Standards

Ещё по этой теме



Андрей Карпов , Павел Еремеев

11 Янв 2022

Теги:
#StaticAnalysis

Технологии статического анализа кода PVS-Studio


- Абстрактное синтаксическое дерево (abstract syntax tree) и методика сопоставления с шаблоном (pattern-based analysis)
- Семантическая модель (semantic model) кода и вывод типов (type inference)
- Препроцессирование C и C++ исходного кода
- Отслеживание компиляции C и C++ исходного кода
- Анализ потоков данных (data-flow анализ) и символьное выполнение (symbolic execution)
- Межпроцедурный анализ
- Межмодульный анализ и аннотирование функций
- Taint-анализ (taint checking)
- Заключение
- Дополнительные ссылки





Межмодульный анализ C++ проектов


Олег Лысый – Межмодульный анализ C++ проектов

Смотреть ... Поделиться

 ++ RUSSIA

Межмодульный
анализ
C++ проектов 


**Олег
Лысый**
PVS-Studio

Посмотреть на  YouTube





Всем спасибо!



Q&A

feklin@viva64.com