ГОСТ Р 71207–2024 — Статический анализ программного обеспечения. Технологии анализа кода

Вебинар от команды PVS-Studio

Андрей Карпов

DevRel

Андрей Карпов

- Один из основателей проекта PVS-Studio — pvs-studio.ru
- Пишу и рассказываю про статический анализ и качество кода

Требования к методам анализа, реализованным в статическом анализаторе

- Методы анализа, реализованные в статическом анализаторе, должны обеспечивать поиск **критических ошибок**, типы которых определяются в соответствии с поддерживаемым языком программирования.

- Про критические ошибки был предыдущий вебинар: https://pvs-studio.ru/ru/blog/video/11084/

Статический анализатор должен реализовывать следующие методы анализа

- анализ программы на синтаксическом уровне;
- внутрипроцедурный анализ потоков данных и управления;
- межпроцедурный и межмодульный контекстно-чувствительный анализ потока данных;
- чувствительный к путям выполнения анализ потоков данных и управления;
- межпроцедурный и межмодульный контекстно-чувствительный анализ помеченных данных.

Вспомогательные виды анализов

- сигнатурный поиск;
- анализ псевдонимов;
- анализ косвенных вызовов;
- статистический анализ;
- анализ иерархии классов.

Если применяется анализ помеченных данных

- Должна быть предоставлена возможность конфигурации анализа: должны задаваться процедуры-источники и процедуры-стоки чувствительных данных.

Требуемые методы

Анализ программы на синтаксическом уровне

- Обрабатывается представление программы, полностью отражающее её синтаксическую структуру, например абстрактное синтаксическое дерево.
- Статический анализ и регулярные выражения
  https://pvs-studio.ru/ru/blog/posts/cpp/0087/

- Более того, даже для простых случаев важен не только синтаксис, но и семантика. См. следующий пример.

Анализ программы на синтаксическом уровне (плюс семантическом) (Overgrowth, C++)

```
typedef struct ovrHapticsClip_ {
  const void* Samples;
  ....
} ovrHapticsClip;

void ovr_ReleaseHapticsClip(ovrHapticsClip* hapticsClip) {
  ....
  delete[] hapticsClip->Samples;
}
```

PVS-Studio: V772 Calling a 'delete' operator for a void pointer will cause undefined behavior. OVR_CAPI_Util.cpp 380

Внутрипроцедурный анализ потоков данных и управления (Elasticsearch, Java)

```
private static byte char64(char x) {
  if ((int)x < 0 || (int)x > index_64.length)
    return -1;
  return index_64[(int)x];
}
```

PVS-Studio: V6025 Possibly index '(int) x' is out of bounds. BCrypt.java(431)

Внутрипроцедурный анализ потоков данных и управления (Protobuf, C++)

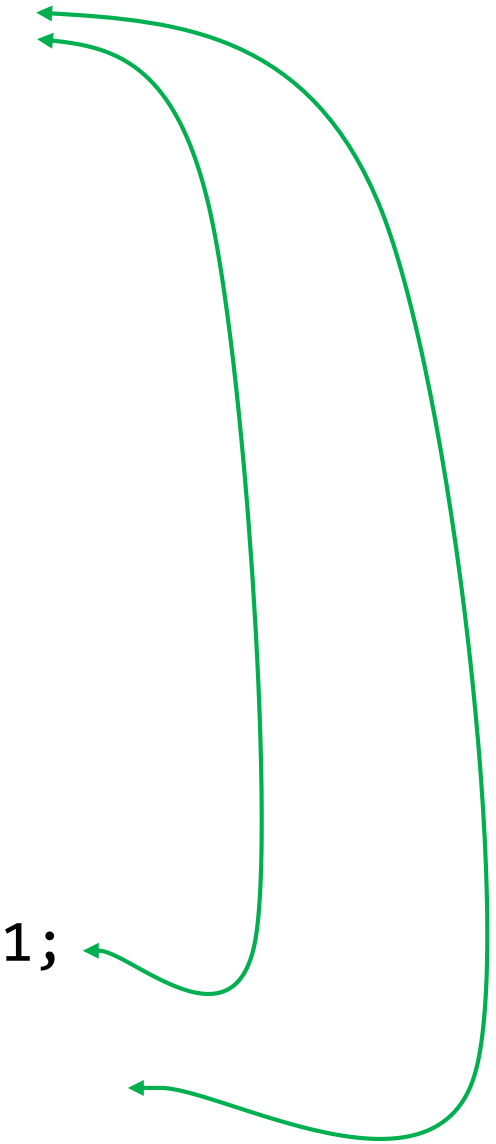
PVS-Studio:

- V547 Expression 'time.month <= kDaysInMonth[time.month] + 1' is always true. time.cc 83
- V547 Expression 'time.month <= kDaysInMonth[time.month]' is always true. time.cc 85

https://pvs-studio.ru/ru/blog/posts/cpp/0550/

```
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];
  }
}
```

```
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];
  }
}
```



```
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];
  }
}
```

Межпроцедурный контекстно-чувствительный анализ (AvalonStudio, C#)

```
// Если в функцию передали нулевую ссылку, то она вернёт false.

private static bool IsBuiltInType(ClangType cursor)
{
  var result = false;
  if (cursor != null && ....)
  {
    return true;
  }
  return result;
}
```

Межпроцедурный контекстно-чувствительный анализ (AvalonStudio, C#)

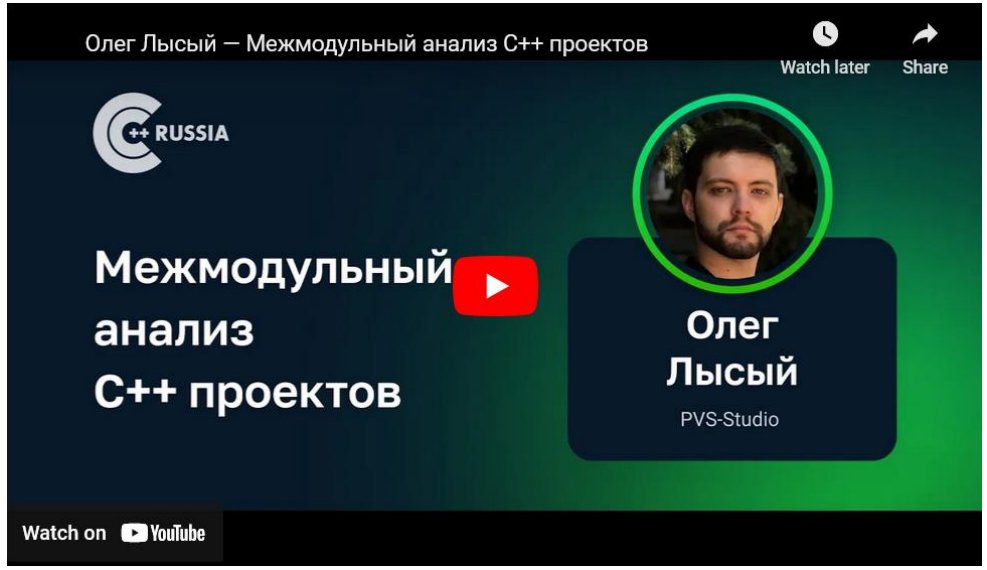
```
if (cursor.ResultType != null)          // <= Проверка
....
else if (cursor.CursorType != null)
{                                       // cursor.ResultType == null
  ....
  // Значит, здесь cursor.ResultType == null
  result.Append(cursor.CursorType.Spelling + " ",
                IsBuiltInType(cursor.ResultType) ? theme.Keyword
                                                 : theme.Type);
```

PVS-Studio: V3022 Expression 'IsBuiltInType(cursor.ResultType)' is always false. CPlusPlusLanguageService.cs 1105

Межмодульный контекстно-чувствительный анализ потока данных

- При выявлении свойств программы учитываются процедуры или переменные из различных модулей (единиц трансляции).

- Применительно к PVS-Studio: https://pvs-studio.ru/ru/blog/video/10834/



Межмодульный контекстно-чувствительный анализ потока данных (mc, C)

```
// Файл: widget-common.c
// Освобождение буфера памяти

Void widget_destroy (Widget * w)
{
  send_message (w, NULL, MSG_DESTROY, 0, NULL);
  g_free (w);
}
```

Межмодульный контекстно-чувствительный анализ потока данных (mc, C)

```
// Файл: editcmd.c
// Использование указателя после освобождения памяти
gboolean edit_close_cmd (WEdit * edit)
{
  Widget *w = WIDGET (edit);
  ....
  widget_destroy (w);      // <= Здесь освободили память
  if (....) .... else
  {
    edit = find_editor (DIALOG (g));
    if (edit != NULL)
      widget_select (w);  // <= Сейчас заглянем внутрь
  }
}
```

Межмодульный контекстно-чувствительный анализ потока данных (mc, C)

```
void
widget_select (Widget * w)
{
  WGroup *g;
  if (!widget_get_options (w, WOP_SELECTABLE))
    return;
  ....
}
```

Межмодульный контекстно-чувствительный анализ потока данных (mc, C)

```
// Добрались до использования уже разрушенной структуры

static inline gboolean
widget_get_options (const Widget * w, widget_options_t options)
{
  return ((w->options & options) == options);
}
```

PVS-Studio: V774 The 'w' pointer was used after the memory was released. editcmd.c 2258

Чувствительный к путям выполнения анализ потоков данных и управления

- Непонятно, почему это отделено от предыдущего: “Межпроцедурный и межмодульный контекстно-чувствительный анализ потока данных”.

- Возможно, нам это странно по причине, что в PVS-Studio всё чувствительно к путям выполнения.

Межпроцедурный и межмодульный контекстно-чувствительный анализ помеченных данных (C)

```
int getindex() {
  int index;
  scanf("%d", &index);
  return index;
}
void useindex(char *buf, int index) {
  buf[index] = 1;
}
void foo() {
  char buf[10];
  int i = getindex();
  useindex(buf, i);
}
```

PVS-Studio: V1010 Unchecked tainted data is used in the second argument: 'i'. Check lines: 14, 20, 9.

Рекомендуется также использование следующих технологий

Дополнительные методы анализа для поиска ошибок

- сигнатурный поиск;
- анализ псевдонимов;
- анализ косвенных вызовов;
- статистический анализ;
- анализ иерархии классов.

Сигнатурный анализ

- Определение наличия свойств программы при помощи поиска строк в исходном коде по некоторому образцу, в том числе заданному с помощью формального языка поиска.
- Например, при помощи регулярных выражений.
- Мы в статьях называли это «поиск по шаблону».

Сигнатурный анализ

- Но всё-таки сигнатурный анализ — это не регулярные выражения.
- Это более сложные техники.
- В чистом виде регулярные выражения применяются крайне редко.
- В ядре PVS-Studio для C и C++ реализовано около **700** диагностик. Регулярные выражения используются только в **5** из них.

Сигнатурный анализ (Blender, C++)

```
void BKE_gpencil_stroke_copy_settings(const bGPDstroke *gps_src,
                                      bGPDstroke *gps_dst)
{
  gps_dst->thickness = gps_src->thickness;
  gps_dst->flag = gps_src->flag;
  gps_dst->inittime = gps_src->inittime;
  gps_dst->mat_nr = gps_src->mat_nr;
  copy_v2_v2_short(gps_dst->caps, gps_src->caps);
  gps_dst->hardness = gps_src->hardness;
  copy_v2_v2(gps_dst->aspect_ratio, gps_src->aspect_ratio);
  gps_dst->fill_opacity_fac = gps_dst->fill_opacity_fac;
  copy_v3_v3(gps_dst->boundbox_min, gps_src->boundbox_min);
  copy_v3_v3(gps_dst->boundbox_max, gps_src->boundbox_max);
  gps_dst->uv_rotation = gps_src->uv_rotation;
  copy_v2_v2(gps_dst->uv_translation, gps_src->uv_translation);
  gps_dst->uv_scale = gps_src->uv_scale;
  gps_dst->select_index = gps_src->select_index;
  copy_v4_v4(gps_dst->vert_color_fill, gps_src->vert_color_fill);
}
```

Сигнатурный анализ (Blender, C++)

```
....
gps_dst->hardness = gps_src->hardness;
....
gps_dst->fill_opacity_fac = gps_dst->fill_opacity_fac;
....
```

PVS-Studio: V570 The 'gps_dst->fill_opacity_fac' variable is assigned to itself. gpencil_legacy.cc 1029

Базовая технология, но не бесполезная ☺

Сигнатурный анализ (Elasticsearch, Java)

```
public boolean equals(Object obj) {
  ....
  return index.equals(other.index)
      && type.equals(other.type)
      && Objects.equals(id, other.id)
      && docVersion == other.docVersion
      && found == other.found
      && tookInMillis == tookInMillis
      && Objects.equals(termVectorList, other.termVectorList);
}
```

PVS-Studio: V6001 There are identical sub-expressions 'tookInMillis' to the left and to the right of the '==' operator. TermVectorsResponse.java(152)

Анализ псевдонимов (C#)

```
object GetPotentialNull() {                    // Может вернуть null
    Random random = new();
    return random.NextDouble() > 0.5 ? new object() : null;
}

void Example_1() {
    object potentialNull = GetPotentialNull();
    ref object alias = ref potentialNull;

    if (alias != null) {
        // Диагностика V3080 “Possible null dereference” не срабатывает,
        // т. к. alias и potentialNull - одно и то же значение.
        _ = potentialNull.ToString();
    }
}
```

Анализ псевдонимов (C#)

```
void Example_2()
{
    var obj = new object();
    ref object alias = ref obj;
    alias = GetPotentialNull(); // Может вернуть null

    // Диагностика срабатывает, т. к. при
    // изменении значения alias меняется и obj.
    _ = obj.ToString(); // V3080: Possible null dereference.
}
```

Статистический анализ

- Интересно тем, что люди постоянно путаются в терминах:
  - **статистический** анализ;
  - **статический** анализ :)

- Статистический анализ — это часть/одна из технологий статического анализа.

Статистический анализ (Daggerfall Unity, C#)

```
case GuildServices.Identify:
  ....
  uiManager.PushWindow(UIWindowFactory.GetInstanceWithArgs(....));
  break;
case GuildServices.Repair:
  ....
  uiManager.PushWindow(UIWindowFactory.GetInstanceWithArgs(....));
  break;
case GuildServices.Training:
  ....
  UIWindowFactory.GetInstanceWithArgs(....);
  break;
case GuildServices.Donate:
  ....
  uiManager.PushWindow(UIWindowFactory.GetInstanceWithArgs(....));
  break;
```

PVS-Studio: V3201 Return value is not always used. Consider inspecting the 'GetInstanceWithArgs' method. PopupWindow.cs 359

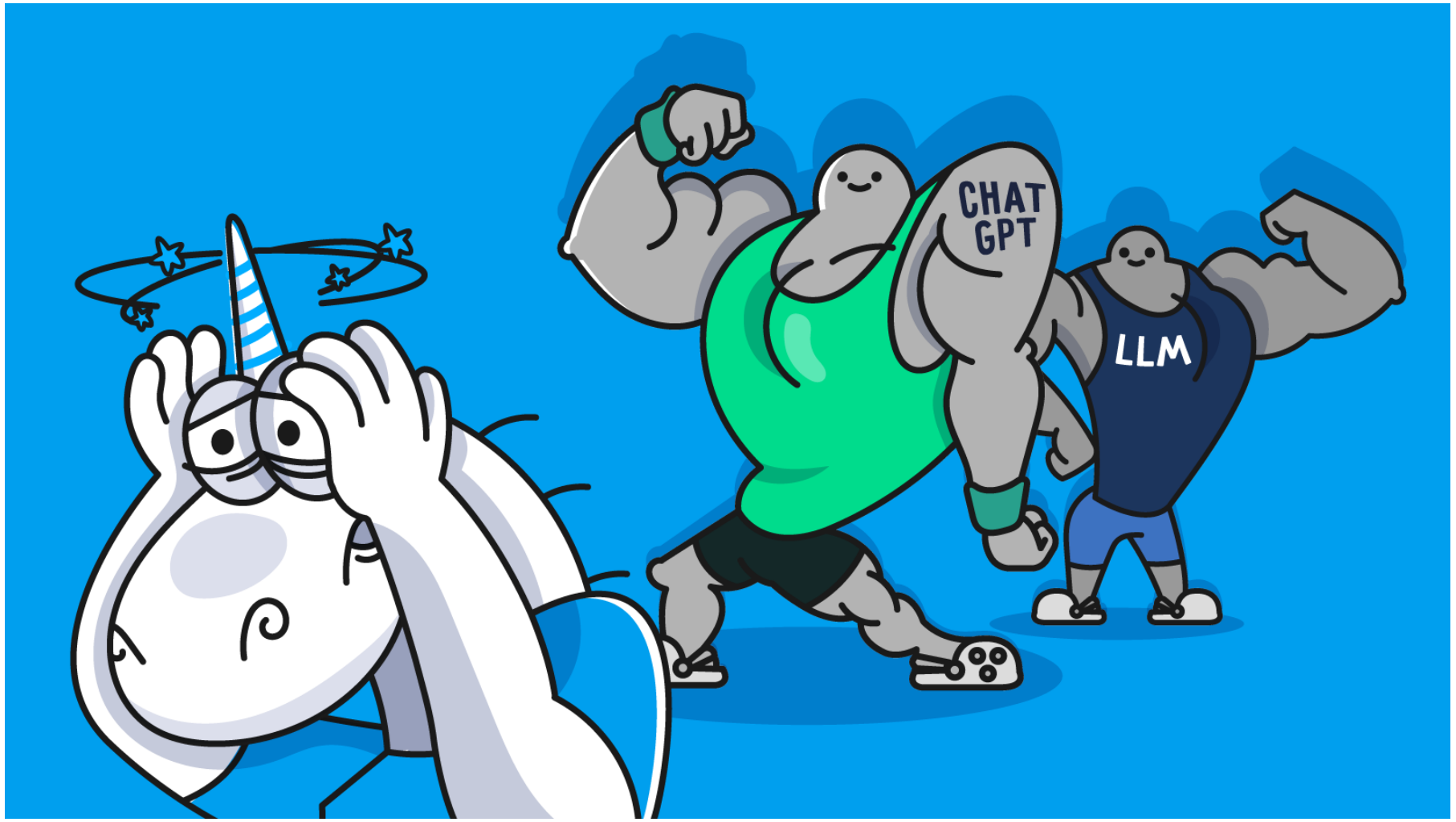
Анализ иерархии классов (Blender, C++)

```
typedef struct CurvesGeometry { .... };

namespace bke
{
  ....
  class CurvesGeometry : public ::CurvesGeometry { .... };

  class CurvesFieldInput : public fn::FieldInput
  {
    ....
    virtual std::optional<AttrDomain> preferred_domain(const CurvesGeometry &curves) const;
  };
  ....
}

namespace blender::nodes::node_geo_input_curve_handles_cc
{
  class HandlePositionFieldInput final : public bke::CurvesFieldInput
  {
    ....
    std::optional<AttrDomain> preferred_domain(const CurvesGeometry & /*curves*/) const;
  };
}
```

PVS-Studio: V762 It is possible a virtual function was overridden incorrectly. See first argument of function 'preferred_domain' in derived class 'HandlePositionFieldInput' and base class 'CurvesFieldInput'.

Что про ML/AI?

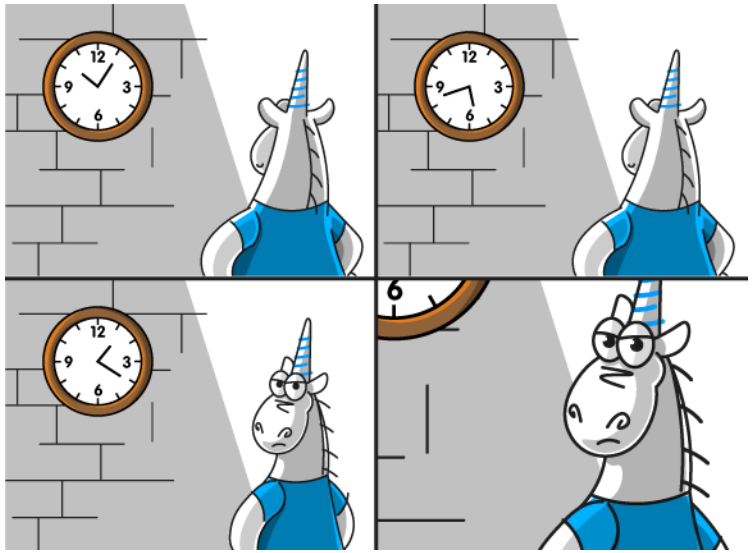
- Ничего



Некоторые требования к инструментам статического анализа

Требования

- Следует использовать такой статический анализатор (**набор анализаторов**), чтобы на технических средствах разработчика ПО для поиска критических ошибок
  - обеспечивался **полный анализ ПО с используемыми заимствованными компонентами**
  - за время, не превышающее **двое суток.**



Требования

- Требования по качеству выполняемого статического анализа предъявляются для критических типов ошибок и проверяются на квалификационном наборе тестов.
- Это отдельный раздел в стандарте «Методика проверки требований к статическому анализатору».

Статический анализатор должен обеспечивать достижение следующих показателей

- доля ошибок первого рода (ложноположительных срабатываний) — не более 50 %;
- доля ошибок второго рода (ложноотрицательных срабатываний, т. е. пропусков заведомо известных ошибок) — не более 50 %.

- Показатели проверяются на квалификационном наборе тестов.

Отчёты должны содержать

- перечень предупреждений о найденных ошибках;
- описание ошибок;
- тип ошибок;
- место в исходном коде программы, где найдены ошибки.

- Выдача результатов в том числе в SARIF формате.

Описание ошибок в документации

- описание ошибки;
- возможные причины возникновения;
- примеры ошибочного кода, для которого выдаётся предупреждение о данном типе ошибки;
- примеры или рекомендации по исправлению ошибки;
- соответствие типа ошибки в анализаторе одному или нескольким идентификаторам в системе классификации дефектов безопасности Common Weakness Enumeration (CWE).

V772. Calling a 'delete' operator for a void pointer will cause undefined behavior.

Анализатор обнаружил потенциально возможную ошибку в коде, связанную с тем, что оператор 'delete' или 'delete[]' применяется для нетипизированного указателя (void*). Согласно стандарту C++20 (п. п. §7.6.2.8/3) такое применение ведет к неопределенному поведению.

Рассмотрим пример такого кода:

```
class Example
{
  int *buf;

public:
  Example(size_t n = 1024) { buf = new int[n]; }
  ~Example() { delete[] buf; }
};

....
void *ptr = new Example();
....
delete ptr;
....
```

Подобный пример опасен тем, что компилятор в реальности не знает, к каком типу относится указатель 'ptr'. Поэтому, при удалении такого нетипизированного указателя могут произойти различные неприятности, например, может возникнуть утечка памяти: оператор 'delete' не вызовет деструктор объекта типа 'Example', на который ссылается указатель 'ptr'.

Если подразумевалась именно работа с нетипизированным указателем, то перед применением оператора 'delete' ('delete[]') его необходимо привести к изначальному типу, например так:

```
....
void *ptr = new Example();
....
delete (Example*)ptr;
....
```

Иначе, во избежание ошибок, рекомендуется использовать только типизированные указатели совместно с оператором 'delete' ('delete[]'):

```
....
Example *ptr = new Example();
....
delete ptr;
....
```

Данная диагностика классифицируется как:

- CWE-758
- CERT-MSC15-C

Документация PVS-Studio

Описание

Пример ошибочного кода

Примеры исправления

Сопоставление с CWE, SEI CERT, OWASP

Заключение

- Следующий вебинар как раз будет посвящён процессам.
- Пишите нам, если хотите узнать про какие-то темы подробнее. Мы собираем идеи для будущих вебинаров.
- Подписывайтесь и общайтесь с нами в VK:
  - https://vk.com/**pvsstudio_rus**

Q&A

Андрей Карпов

DevRel