Использование статического анализатора в разработке безопасного ПО (ГОСТ Р 71207-2024)

на примере PVS-Studio

Андрей Карпов

DevRel

Андрей Карпов

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

ГОСТ Р 71207–2024

ГОСТ Р 71207–2024

Защита информации

РАЗРАБОТКА БЕЗОПАСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Статический анализ программного обеспечения

Общие требования

Введён в действие 01.04.2024

Введён впервые

Применяется совместно с ГОСТ Р 56939

- РАЗРАБОТКА БЕЗОПАСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ Общие требования
- Направлен на предотвращение и устранение уязвимостей программ

- Меры по разработке безопасного ПО:
  - …
  - Динамический анализ кода
  - **Статический анализ кода**
  - …

Актуальность
ГОСТ Р 71207–2024

Что такое «использование статических анализаторов кода»?

- Достаточно взять FindBugs для Java? Последний релиз был 9 лет назад.
- Ну хорошо, а SpotBugs? Последний стабильный релиз был 2 года назад.
- Cppcheck для C++?

- Анализатор регулярно запускается на сервере. Значит, внедрён?

Истории

- У нас статический анализ внедрён. DevOps отдел настроил регулярный запуск какого-то анализатора. Правда, отчёты никто не смотрит;
- Про KPI и правку имён переменных;
- Да-да, у нас есть инструмент для автоматического форматирования кода.

Актуальность ГОСТ для вашей команды

- Если вы разрабатываете небезопасное ПО — никак;
- В этом нет пренебрежения. Не везде безопасность нужна и важна;

- Доклад с практической стороны раскроет суть стандарта;
- На примерах посмотрим, что понимается под критическими ошибками и технологиями их поиска;
- Вы сможете понять, следует ли что-то менять в инструментарии и процессах.

Некоторые термины

Анализ потока данных

- Могут определяться возможные значения переменных и констант.
- Примеры артефактов:
  - переменная не инициализирована;
  - диапазон значений;
  - точное значение;
  - множество значений;
  - указатель:

    нулевой;

    неинициализированный;

    указывает на буфер размера N байт;

    память освобождена.

Анализ потока данных (RavenDB, C#)

```
public override void VisitMethod(MethodExpression expr)
{
  if (expr.Name.Value == "id" && expr.Arguments.Count == 0)
  {
    if (expr.Arguments.Count != 1)
    {
      throw new InvalidOperationException("....");
    }
```

PVS-Studio: V3022 Expression 'expr.Arguments.Count != 1' is always true. JavascriptCodeQueryVisitor.cs 188

Анализ потока управления (Far2l, C++)

```
BOOL WINAPI _export SEVENZ_OpenArchive(const char *Name,
                                       int *Type)
{
  Traverser *t = new Traverser(Name);
  if (!t->Valid())
  {
    return FALSE;
    delete t;
  }
  ...
```

PVS-Studio: V779 Unreachable code detected. 7z.cpp 203

Особенность PVS-Studio

- В PVS-Studio анализ потока данных и потока управления неразделимы.

- Нет повода рассматривать их по отдельности.
- Они работают вместе.

Пример (FreeCAD, C++)

```
QGVPage* QGIView::getQGVPage(TechDraw::DrawView* dView)
{
  ViewProviderPage* vpp = getViewProviderPage(dView);
  if (!vpp) {
    return vpp->getQGVPage();
  }
  return nullptr;
}
```

Анализ потока данных + анализ потока управления

PVS-Stidop: V522 Dereferencing of the null pointer 'vpp' might take place.
QGIView.cpp 592

Анализ помеченных данных

- Анализируется течение потока данных от источников до стоков;
- Основные цели:
  - Обнаружение попадания данных из источников в стоки, что может означать утечку конфиденциальных данных;
  - Небезопасное использование входных данных, которое может нарушить ход работы программы.

Модельный вариант (ошибки)

- Подтип некоторого типа ошибки в программе, отнесение к которому осуществляется исходя из особенностей реализации ошибки данного типа и особенностей языка программирования;
- Разбиение типа ошибки на модельные варианты применяется из-за технологических ограничений статического анализа;
- Примеры:
  - поиск неопределённого поведения;
  - поиск разыменований нулевого указателя.

Модельный вариант ошибки (MuseScore, C++)

```
Ms::Segment* NotationSelectionRange::rangeStartSegment() const {
  Ms::Segment* startSegment = score()->selection().startSegment();

  startSegment->measure()->firstEnabled();

  if (!startSegment) {
    return nullptr;
  }
```

PVS-Studio: V595 The 'startSegment' pointer was utilized before it was verified against nullptr. Check lines: 129, 131. notationselectionrange.cpp 129

Критическая ошибка (важнейшее понятие)

- Ошибка, которая может привести к нарушению безопасности обрабатываемой информации;
- Мы в статьях называли такие ошибки потенциальными уязвимостями;

- В настоящем стандарте в задачи статического анализа **не входит разграничение ошибок в части последствий**, необходимо найти потенциальные места ошибок;
- **Мы часто разбирали это в статьях. Приятно, что это звучит и в стандарте.**

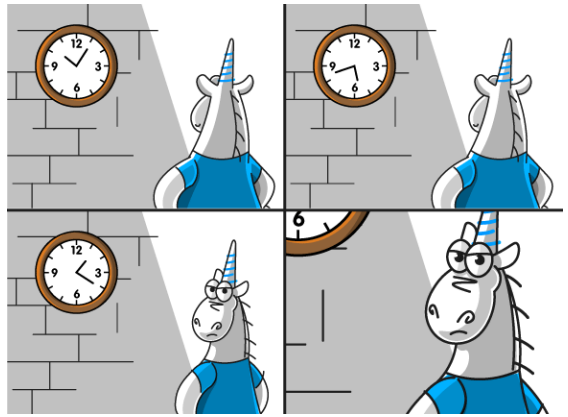
Чем полезно понятие «критическая ошибка»?

Классификация предупреждений

- Предупреждений от анализаторов всегда больше, чем хотелось бы;
- Проблематика, о которой слышал: сертификационная лаборатория присылает отчёт с большим количеством предупреждений;
- Как ответить, что не всё критично?
- Чем руководствоваться самой лаборатории?
- Теперь можно руководствоваться ГОСТ-ом и понятием «критическая ошибка»;
- Можно выделить соответствующее подмножество диагностических правил.

PVS-Studio и критические ошибки

- PVS-Studio умеет выявлять критические ошибки;
- Пока нет возможности выбрать подмножество критических ошибок;
- Сделаем.



Однако помним: «некритическая ошибка» тоже может быть опасной

- К классификации следует относиться вдумчиво;
- Всё условно и относительно;

- Формально верно найденная критическая ошибка может не представлять угрозы;
- Ошибка, не классифицированная как критическая, вполне может такой быть.

```
static bool samu_correct(struct samu *s1, struct samu *s2)
{
  ....
  } else if (s1_len != s1_len) {
    DEBUG(0, ("Password history not written correctly, "
              "lengths differ, want %d, got %d\n",
          s1_len, s2_len));
    ret = False;
```

Samba, C

- Ошибка найдена как **модельный вариант опечатки**.
  PVS-Studio: V501 There are identical sub-expressions to the left and to the right of the '!=' operator: s1_len != s1_len pdbtest.c 106
- Но фактически это может быть **ошибкой непроверенного использования чувствительных данных**.

Критические ошибки, которые должен находить анализатор для компилируемых языков

Ошибки непроверенного использования чувствительных данных (NcFTP, C)

```
static int NcFTPConfirmResumeDownloadProc(....)
{
  ....
  if (fgets(newname, sizeof(newname) - 1, stdin) == NULL)
    newname[0] = '\0';
  newname[strlen(newname) - 1] = '\0';
  ....
}
```

Воспроизведение: вводим строку с NUL в начале.

PVS-Studio: V1010 Unchecked tainted data is used in index.

https://pvs-studio.ru/ru/blog/posts/cpp/0599/

Ошибки непроверенного использования чувствительных данных (BlogEngine, C#)

Для презентации длинно, смотрите статью.
«Уязвимости из-за обработки XML-файлов: XXE в C# приложениях в теории и на практике»
https://pvs-studio.ru/ru/blog/posts/csharp/0918/

PVS-Studio: V5614 [OWASP-5.5.2] Potential XXE vulnerability inside method. Insecure XML parser is used to process potentially tainted data from the first argument: 'inputXml'. BlogEngine.Core XMLRPCRequest.cs 41

```
private static string ParseRequest(HttpContext context)
{
    var buffer = new byte[context.Request.InputStream.Length];

    context.Request.InputStream.Position = 0;
    context.Request.InputStream.Read(buffer, 0, buffer.Length);

    return Encoding.UTF8.GetString(buffer);
}
```

Целочисленное переполнение (libtorrent, C++)

```
void torrent::get_download_queue(std::vector<partial_piece_info>* queue) const
{
  ....
  const int blocks_per_piece = m_picker->blocks_in_piece(piece_index_t(0));
  ....
  int counter = 0;
  for (auto i = q.begin(); i != q.end(); ++i, ++counter)
  {
    ....
    pi.blocks = &blk[std::size_t(counter * blocks_per_piece)];
  }
```

PVS-Studio: V1028 Possible overflow. Consider casting operands of the 'counter * blocks_per_piece' operator to the 'size_t' type, not the result. torrent.cpp 7092

Некорректное совместное использование знаковых и беззнаковых чисел (Hypertext Preprocessor, C)

```
PHP_CLI_API size_t sapi_cli_single_write(....)
{
  ....
  size_t shell_wrote;
  shell_wrote = cli_shell_callbacks.cli_shell_write(....);
  if (shell_wrote > -1) {
    return shell_wrote;
  }
  ....
}
```

PVS-Studio: V605 Consider verifying the expression: shell_wrote > - 1. An unsigned value is compared to the number -1. php_cli.c 266

Ошибки переполнения буфера (ICU, C)

```
static char plugin_file[2048] = "";
U_CAPI void U_EXPORT2 uplug_init(UErrorCode *status) {
  ....
  // uprv_strncat раскрывается в strncat
  uprv_strncpy(plugin_file, plugin_dir,          2047);
  uprv_strncat(plugin_file, U_FILE_SEP_STRING,   2047);
  uprv_strncat(plugin_file, "icuplugins",        2047);
  uprv_strncat(plugin_file, U_ICU_VERSION_SHORT, 2047);
  uprv_strncat(plugin_file, ".txt",              2047);
  ....
```

Свободное место в буфере, а не размер буфера!

PVS-Studio: V645 The 'strncat' function call could lead to the 'plugin_file' buffer overflow. The bounds should not contain the size of the buffer, but a number of characters it can hold. icuplug.c

Ошибки некорректного использования системных процедур и интерфейсов, связанных с обеспечением ИБ (Crypto++, C++)

```
MicrosoftCryptoProvider::MicrosoftCryptoProvider()
{
  if(!CryptAcquireContext(&m_hProvider, 0, 0, PROV_RSA_FULL,
                          CRYPT_VERIFYCONTEXT))
    throw OS_RNG_Err("CryptAcquireContext");
}
```

PVS-Studio: V1109 The 'CryptAcquireContextA' function is deprecated. Consider switching to an equivalent newer function. osrng.cpp 50

> **Important** This API is deprecated. New and existing software should start using Cryptography Next Generation APIs. Microsoft may remove this API in future releases.

Ошибки при работе с многопоточными примитивами (Zephyr, C)

```
static int nvs_startup(struct nvs_fs *fs) {
  ....
  k_mutex_lock(&fs->nvs_lock, K_FOREVER);
  ....
  if (fs->ate_wra == fs->data_wra && last_ate.len) {
    return -ESPIPE;
  }
  ....
end:
  k_mutex_unlock(&fs->nvs_lock);
  return rc;
}
```

PVS-Studio: V1020 The function exited without calling the 'k_mutex_unlock' function. Check lines: 620, 549. nvs.c 620

Ошибки при работе с многопоточными примитивами (WildFly, Java)

```
private volatile ExpressionFactory factory;
....
@Override
public ExpressionFactory getExpressionFactory() {
  if (factory == null) {
    synchronized (this) {
      if (factory == null) {
        factory = delegate.getExpressionFactory();
        for (ExpressionFactoryWrapper wrapper : wrapperList) {
          factory = wrapper.wrap(factory, servletContext);
        }
      }
    }
  }
  return factory;
}
```

PVS-Studio: V6082 Unsafe double-checked locking. A previously assigned object may be replaced by another object. JspApplicationContextWrapper.java(74), JspApplicationContextWrapper.java(72)

Особенность поиска ошибок, связанных с многопоточностью

- Пример, когда ищутся не «вообще ошибки», а модельные ошибки;
- Причина: очень высокая сложность задачи с точки зрения статического анализа кода;
- Хотя ошибки относятся к критическим, ГОСТ делает для них исключение и говорит, что их поиск опционален;
- Разумное послабление.

C и C++ особенные: ещё критические ошибки

Ошибки разыменования нулевого указателя (MuditaOS, C++)

```
void CallLogDetailsWindow::onBeforeShow(...., SwitchData *data)
{
  ....
  if (auto switchData = dynamic_cast
                        <calllog::CallLogSwitchData *>(data);
      data != nullptr)
  {
    record = switchData->getRecord();
```

PVS-Studio: V757 It is possible that an incorrect variable is compared with nullptr after type conversion using 'dynamic_cast'. Check lines: 214, 214. CallLogDetailsWindow.cpp 214

Ошибки деления на ноль (LLVM, C)

```
COMPILER_RT_ABI du_int __udivmoddi4(...., du_int* rem) {
  ....
  if (d.s.low == 0) {
    if (d.s.high == 0) {
      if (rem)
        *rem = n.s.high % d.s.low;
      return n.s.high / d.s.low;
    }
```

PVS-Studio:

- V609 Mod by zero. Denominator 'd.s.low' == 0. udivmoddi4.c 61
- V609 Divide by zero. Denominator 'd.s.low' == 0. udivmoddi4.c 62

Ошибки управления динамической памятью (OpenToonz, C++)

```
template <class T> void doDirectionalBlur(....) {
  T *row, *buffer;
  ....
  row = new T[lx + 2 * brad + 2];
  ....
  free(row);
  r->unlock();
}
```

PVS-Studio: V611 The memory was allocated using 'new' operator but was released using the 'free' function. Consider inspecting operation logics behind the 'row' variable. motionblurfx.cpp 288

Ошибки управления динамической памятью (MuseScore, C++)

```
void GuitarPro6::readGpif(QByteArray* data)
{
  ....
  } else {
    delete slur;
    legatos[slur->track()] = 0;
  }
```

PVS-Studio: V774 The 'slur' pointer was used after the memory was released. importgtp-gp6.cpp 2592

Ошибки использования форматной строки (OpenCOLLADA, C++)

```
struct vector
{
  double x;
  double y;
  double z;
  void write(FILE* file) const
  {
    fprintf(file, "%f %f %f %f", x, y, z);
  }
```

PVS-Studio: V576 Incorrect format. A different number of actual arguments is expected while calling 'fprintf' function. Expected: 6. Present: 5. mayadmtypes.h 657

Использование неинициализированных переменных (CovidSim Model, C++)

```
void CalcLikelihood(....) {
    ....
    double ModelValue;
    // loop over all days of infection up to day of sample
    for (int k = offset; k < day; k++)
    {
        // add P1 to P2 to prevent degeneracy
        double prob_seroconvert = P.SeroConvMaxSens * ....;
        ModelValue += c * TimeSeries[k - offset].incI * prob_seroconvert;
    }
```

PVS-Studio: V614 Uninitialized variable 'ModelValue' used. CovidSim.cpp 5412

Ошибки утечек памяти (PMDK, C)

```
static enum pocli_ret
pocli_args_obj_root(...., char* in, PMEMoid** oidp)
{
    char* input = strdup(in);
    if (!input)
        return POCLI_ERR_MALLOC;

    if (!oidp)
        return POCLI_ERR_PARS;
```

PVS-Studio: V773 The function was exited without releasing the 'input' pointer. A memory leak is possible. pmemobjcli.c 238

И других ресурсов (CMake, C)

```
RHASH_API int rhash_file(....)
{
    FILE* fd;
    ....
    if ((fd = fopen(filepath, "rb")) == NULL) return -1;
    if ((ctx = rhash_init(hash_id)) == NULL) return -1;
    res = rhash_file_update(ctx, fd);
    fclose(fd);
```

PVS-Studio: V773 The function was exited without closing the file referenced by the 'fd' handle. A resource leak is possible. rhash.c 450

Обнаружение критических ошибок

- Чтобы обеспечить поиск критических ошибок на высоком уровне, стандарт описывает методы анализа, которые должны быть реализованы в статических анализаторах.

Требования к методам анализа, реализованным в статическом анализаторе

Внутрипроцедурный анализ потоков данных и управления (Unity3D, C#)

```
public NetworkConnection Get(int connId)
{
  if (connId < 0)
  {
    return m_LocalConnections[Mathf.Abs(connId) - 1];
  }

  if (connId < 0 || connId > m_Connections.Count)
  ....
}
```

PVS-Studio: V3063 A part of conditional expression is always false: connId < 0. UnityEngine.Networking ConnectionArray.cs 59

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

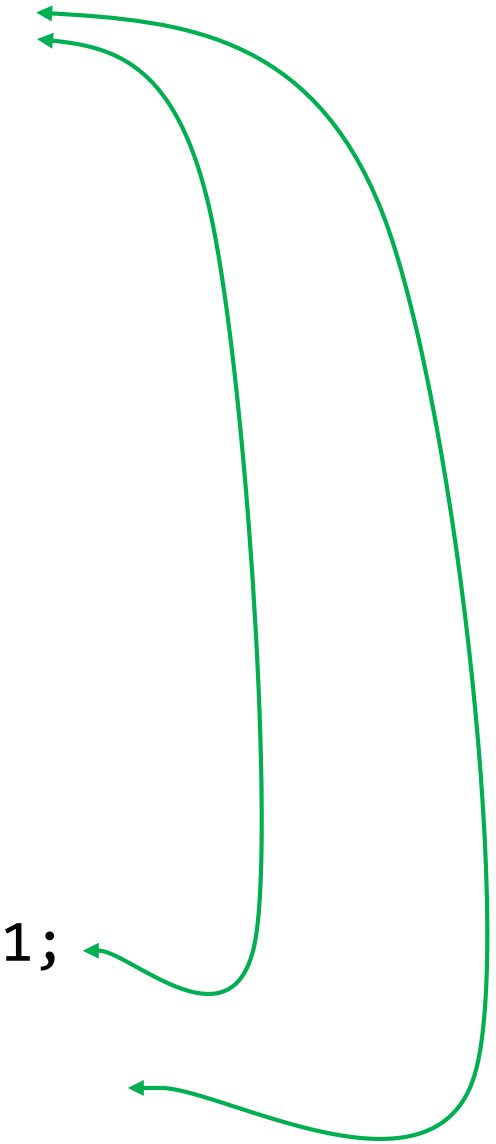
PVS-Studio:

- 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#)

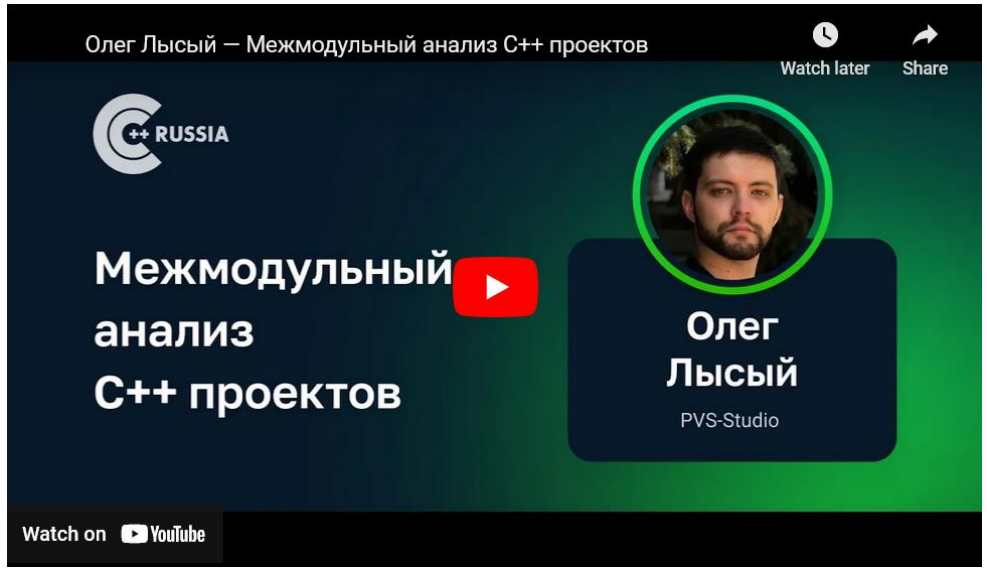
```
if (cursor.ResultType != null)        // <= Проверка


....
else if (cursor.CursorType != null)
{
  ....
  // 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, непонятно 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.

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

Рекомендуется также использование следующих технологий

Методы анализа для поиска дополнительных типов ошибок

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

- Рассмотрим некоторые, интересные на мой взгляд.

Сигнатурный анализ

- Мы в статьях называли это «поиск по шаблону»;
- 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.

Статический анализ — это что-то типа регулярок?

- **Нет!**

- Это очень маленькая его часть;
- Скорее всего, это не будут регулярные выражения в чистом виде (используется AST).

Сигнатурный анализ (Qemu, C)

```
target_ulong helper_mftc0_cause(CPUMIPSState *env)
{
  ....
  if (other_tc == other->current_tc) {
    tccause = other->CP0_Cause;
  } else {
    tccause = other->CP0_Cause;
  }
  ....
}
```

PVS-Studio: V523 The 'then' statement is equivalent to the 'else' statement. cp0_helper.c 383

Сигнатурный анализ (NetBeans, Java)

```
private RevisionInterval
getResortedRevisionInterval(SVNRevision revision1,
                            SVNRevision revision2) {
  RevisionInterval ret;
  if(revision1.equals(SVNRevision.HEAD) &&
     revision1.equals(SVNRevision.HEAD)) {
    ....
  }
```

PVS-Studio: V6001 There are identical sub-expressions 'revision1.equals(SVNRevision.HEAD)' to the left and to the right of the '&&' operator. RevertModifications.java(387)

Статистический анализ

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

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

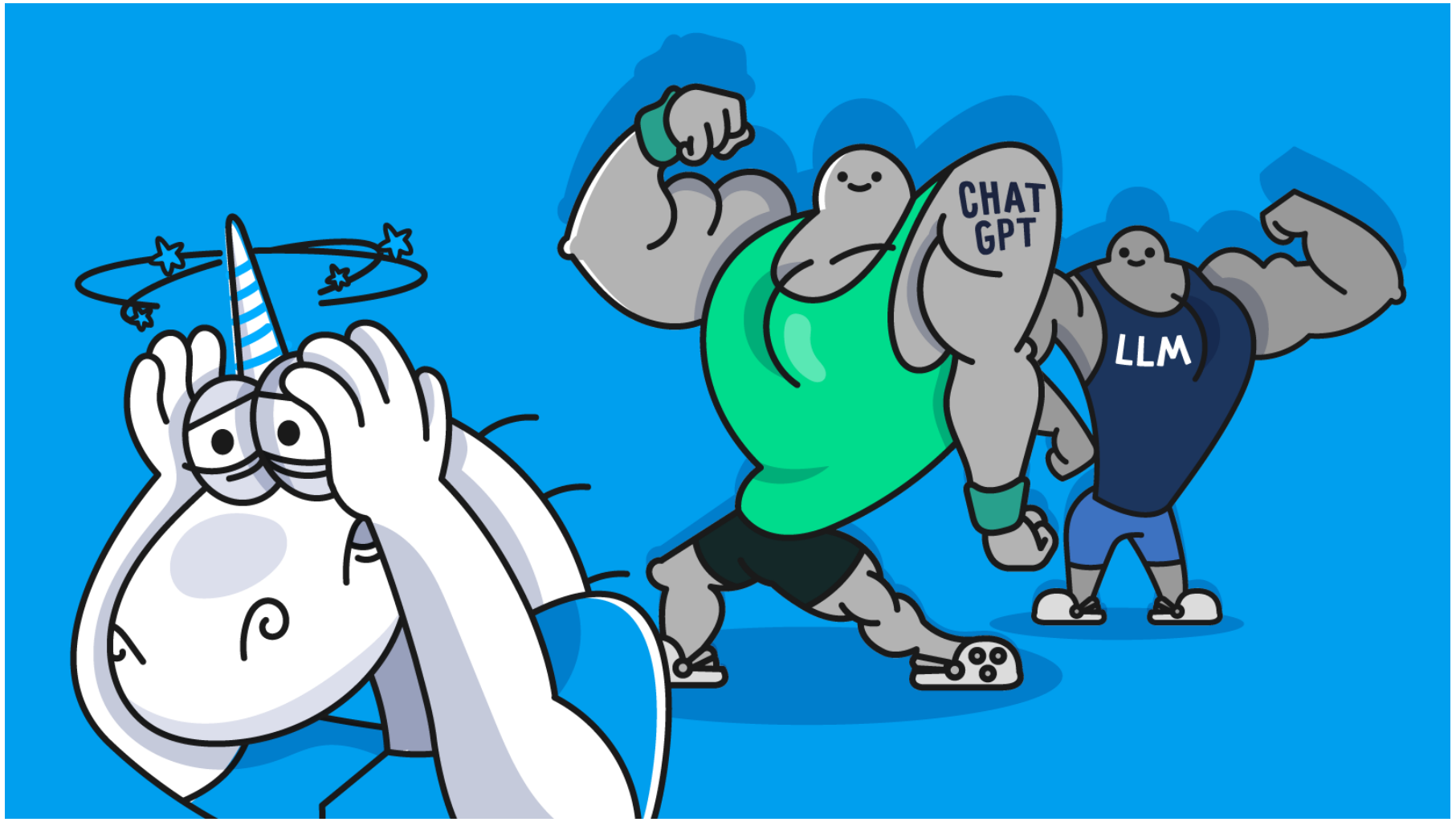
Статистический анализ (Linux Kernel, C)

```
static const
struct XGI330_LCDDataDesStruct2  XGI_LVDSNoScalingDesData[] = {
  {0,  648,  448,  405,  96, 2}, /* 00 (320x200,320x400,
                                    640x200,640x400) */
  {0,  648,  448,  355,  96, 2}, /* 01 (320x350,640x350) */
  {0,  648,  448,  405,  96, 2}, /* 02 (360x400,720x400) */
  {0,  648,  448,  355,  96, 2}, /* 03 (720x350) */
  {0,  648,    1,  483,  96, 2}, /* 04 (640x480x60Hz) */
  {0,  840,  627,  600, 128, 4}, /* 05 (800x600x60Hz) */
  {0, 1048,  805,  770, 136, 6}, /* 06 (1024x768x60Hz) */
  {0, 1328,    0, 1025, 112, 3}, /* 07 (1280x1024x60Hz) */
  {0, 1438,    0, 1051, 112, 3}, /* 08 (1400x1050x60Hz)*/
  {0, 1664,    0, 1201, 192, 3}, /* 09 (1600x1200x60Hz) */
  {0, 1328,    0, 0771, 112, 6}  /* 0A (1280x768x60Hz) */
};
```

PVS-Studio: V536 Be advised that the utilized constant value is represented by an octal form. Oct: 0771, Dec: 505. vb_table.h 1379

Что про ML/AI?

- Ничего



Связь с безопасной разработкой ПО и различные требования

Про выбор анализаторов

- Для соответствия требованиям настоящего стандарта разработчик ПО **должен** использовать в ходе разработки статический анализатор или **набор статических анализаторов** для поиска ошибок;
- Обратите внимание на «**набор анализаторов**». Не требуется выбирать один универсальный. Можно использовать несколько, выбирая наиболее подходящие и мощные решения.

Про выбор анализаторов

- При отсутствии применимых инструментов статического анализа, отвечающих всем требованиям, следует использовать в жизненном цикле ПО инструменты, наиболее полно выполняющие данные требования.
- **Приоритет следует отдавать инструментам, демонстрирующим лучшие ключевые показатели**.

Требования к инструментам статического анализа

- Следует использовать такой статический анализатор (набор анализаторов), чтобы **полный анализ ПО** с используемыми заимствованными компонентами выполнялся не более **двух суток**.

- PVS-Studio:
  - Xeon Gold 5220R, 24 ядра, 2.20 GHz,128 Gb;
  - 122 C и C++ проекта (Visual C++);
  - 1 час 34 минуты.

Требования к порядку выполнения статического анализа, внедрение

- Всё хорошо расписано. Понадобится — обращаетесь к ГОСТ-у.
  - …
  - Настройка инструмента статического анализа;
  - Разметка полученных результатов и формирование начальной базы предупреждений о потенциальных ошибках;
  - Предупреждения о потенциальных ошибках, полученные в ходе статического анализа ПО, должны быть размечены: просмотрены для экспертного определения их истинности или ложности.
  - …

Требования к обновлениям

- Следует использовать **регулярно обновляющиеся статические анализаторы**, поддерживающие современные стандарты языков программирования и соответствующие системы сборки.

- Вспоминаем FindBugs, SpotBugs.

Требования к регулярности

- **Для своевременного выявления и исправления ошибок статический анализ должен выполняться регулярно;**
- Накопление непроанализированных изменений ухудшает качество проводимой экспертизы;
  - Постоянно про это рассказываем;
  - Боремся с подходом «запустим анализатор перед релизом».
- Рассказ про: релиз два раза в год,
  скачиваем на неделю PVS-Studio перед этим.

«Анализатор запускается» не равно «анализатор внедрён»

- Статический анализ следует проводить регулярно на этапе конструирования;
- Статический анализ **всего** разрабатываемого ПО следует выполнять **не реже одного раза в 10 рабочих дней**, если за данный период времени исходный код был изменён.

Нет подходу «анализатор запускается, но не смотрим»!

- Для своевременной и эффективной работы с отчётом **просмотр предупреждений** следует выполнять:
  - при анализе **измененных** частей ПО — не позже, **чем через три рабочих дня** после выполнения анализа;
  - при анализе ПО целиком — не позже, **чем через 10 рабочих дней** после выполнения анализа.

Правка ошибок

- Потенциальные ошибки, которые были отнесены к истинным, не позже, чем через **10 рабочих дней** после выполнения разметки, следует **исправить**;
- Либо запланировать сроки их устранения в соответствии с принятыми процессами разработки ПО;
- Если применяется гибкая методология разработки, то план по устранению ошибок включается в ближайший цикл разработки.

Контроль

- Контроль над ходом устранения выявленных потенциальных ошибок следует выполнять **не реже одного раза в месяц**.
- Для контроля следует привлекать специалистов, которые не участвовали в процессе анализа, просмотра предупреждений и устранения ошибок.

Настройка анализа чувствительных данных

- Если статический анализатор для поиска ошибок применяет анализ помеченных данных, должна быть предоставлена возможность конфигурации анализа: **должны задаваться процедуры-источники и процедуры-стоки чувствительных данных**.

- В PVS-Studio есть механизм аннотирования функций;
- Но нет возможности задавать источники/стоки;
- Реализуем в одном из ближайших релизов.

Контролируемая сборка

- Проще говоря, не доверяйте анализаторам, которые умеют анализировать просто любой набор файлов :)

- При анализе ПО, которое требует сборки, должен быть применён вспомогательный анализ для уточнения межмодульных связей, использующий результаты контролируемой сборки.
- Анализ должен учитывать параметры программных инструментальных средств систем программирования, применяемых в ходе контролируемой сборки ПО.

Статический анализатор должен содержать в документации описание всех типов ошибок

- Для каждого типа ошибки должны быть приведены:
  - описания;
  - возможная причина возникновения;
  - примеры ошибочного кода, для которого выдаётся предупреждение о данном типе ошибки;
  - примеры или рекомендации исправления данного типа ошибки.
- Очевидное требования, но на практике далеко не все анализаторы справляются с документацией.

Cppcheck (описания просто нет)

Auto Variables

A pointer to a variable is only valid as long as the variable is in scope.
Check:
- returning a pointer to auto or temporary variable
- assigning address of an variable to an effective parameter of a function
- returning reference to local/temporary variable
- returning address of function parameter
- suspicious assignment of pointer argument
- useless assignment of function argument

Boolean

Boolean type checks
- using increment on boolean
- comparison of a boolean expression with an integer other than 0 or 1
- comparison of a function returning boolean value using relational operator

SpotBugs (чуть лучше)

FI: Finalizer only nulls fields (FI_FINALIZER_ONLY_NULLS_FIELDS)

This finalizer does nothing except null out fields. This is completely pointless, and requires that the object be garbage collected, finalized, and then garbage collected again. You should just remove the finalize method.

FI: Finalizer nulls fields (FI_FINALIZER_NULLS_FIELDS)

This finalizer nulls out fields. This is usually an error, as it does not aid garbage collection, and the object is going to be garbage collected anyway.

UI: Usage of GetResource may be unsafe if class is extended (UI_INHERITANCE_UNSAFE_GETRESOURCE)

Calling `this.getClass().getResource(...)` could give results other than expected if this class is extended by a class in another package.

Ещё рекомендация по документированию диагностик

- Применяемая в статических анализаторах типизация разнится, что затрудняет соотнесение предупреждений, полученных от разных анализаторов;
- Для облегчения работы с предупреждениями в диагностическую информацию добавляют соотнесение ошибки с одной из популярных систем классификации дефектов безопасности, например, с MITRE CWE;
- В описании ошибок также следует указывать их соответствие идентификаторам в системе классификации дефектов безопасности MITRE 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 здорового человека (PVS-Studio)

Описание

Пример ошибочного кода

Примеры исправления

Сопоставление с CWE, SEI CERT, OWASP

Пользовательские расширения

- Статический анализатор может поддерживать возможность доработки реализованных алгоритмов (правил) поиска конкретных типов ошибок;
- Может предоставлять добавление алгоритмов поиска новых типов ошибок пользователем анализатора;
- Опциональная возможность;
- В PVS-Studio на данный момент нет. Реализуем диагностики на заказ.

Почему подсветка кода в IDE это ещё не анализатор кода

- Практические соображения;
- Человек может просмотреть или отключить;
- Анализ должен быть регулярным процессом;
- Контроль;
- История и статистика.

Почему подсветка кода в IDE это ещё не анализатор кода

- В ГОСТ про это другими словами;
- Статический анализатор должен обеспечивать возможность автоматизации действий для использования в системах непрерывной интеграции;
- Должно быть обеспечено хранение результатов запуска статического анализатора и разметки этих результатов;
- Возможность сравнения результатов запусков анализатора.

Требования к специалистам, проводящим статический анализ

Требования к специалистам

- В рамках доклада разбирать смысла нет;
- Суть: специалисты должны быть специалистами :)
- Описание поможет в составлении должностных инструкций, вакансий, матриц компетенций и т.д.
- Возможность сопротивляться подходу: «DevOps-ники анализатор прикрутили и гоняют, вот пусть и смотрят логи заодно».

Методика проверки требований к статическому анализатору

Хорошо описано с точки зрения сертификации

- В состав набора квалификационных тестов должны входить ….
- Для ошибок на наборе тестов статический анализатор должен обеспечивать достижение следующих показателей:
  - Ложноположительных срабатываний — не более 50 %;
  - Ложноотрицательных срабатываний — не более 50 %;
- Состав набора квалификационных тестов следует регулярно пересматривать;
- Примером открытого набора тестов, частично соответствующего требованиям, является Juliet Test Suite.

Но разработчикам анализатора непонятно, от чего отталкиваться

- На разных тестах можно получить разные характеристики;
- **Получается: “Хорошо делай — хорошо будет”;**
- Честно;
- Но всё-таки чем руководствоваться и в какую сторону развивать анализатор?

Итоги

Для пользователей анализаторов кода

- Можно понять, используются ли качественные инструменты анализа или нужно поискать что-то помощнее;
- Можно понять, правильно ли в компании построен процесс использования анализаторов;
- Хорошо описан процесс внедрения;
- Понятно, как проводить аттестацию инструментов.

Для разработчиков анализаторов кода

- Появился хороший и полезный ориентир, помогающий создавать мощные инструменты анализа кода;

- PVS-Studio:
  - После чтения ГОСТ открыта большая эпик-задача на различные усовершенствования анализатора;
  - Изменилась дорожная карта развития PVS-Studio.

В целом приятные ощущения

- Возможно, у меня профессиональная деформация, но мне понравилось;
- ГОСТ написан понятно;
- Полезная информация и хорошие рекомендации;
- Не оторван от реальности! Всё по делу;
- Спасибо всем, кто работал над его составлением.

Q&A

Андрей Карпов

DevRel