Статический анализ кода в методическом документе ЦБ РФ «Профиль защиты»

Профиль защиты прикладного ПО автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций

Андрей Карпов

ООО «ПВС»

Андрей Карпов

Директор по развитию бизнеса (CBDO)

- Один из основателей проекта PVS-Studio https://pvs-studio.ru
- 18 лет в сфере качества и анализа кода
- Хабр: @Andrey2008

- Telegram канал «Бестиарий программирования» t.me/programming_tales

PVS-Studio

Методический документ 2021 года

ПРОФИЛЬ ЗАЩИТЫ
прикладного программного обеспечения
автоматизированных систем и приложений кредитных
организаций и некредитных финансовых организаций

- Разработан Банком России (ЦБ РФ)
- PDF на сайте ЦБ РФ

- Буду использовать сокращение **ПЗ**

Новая версия конца 2025 года

- Информационное письмо Банка России № ИН-017-56/118 **от 26.12.2025**
  - Операторам по переводу денежных средств
  - Некредитным финансовым организациям
  - Операторам услуг платёжной инфраструктуры
- PDF на сайте ЦБ РФ
- Опубликована новая редакция методического документа «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций»

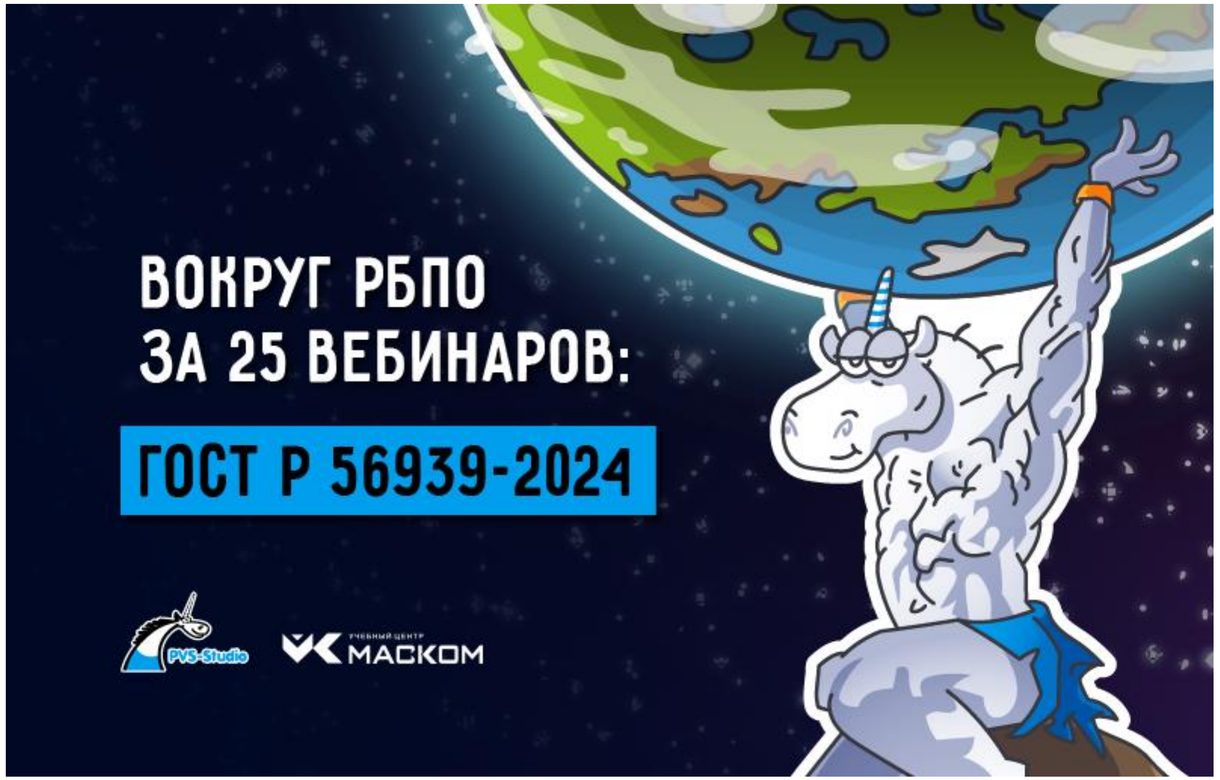
ПЗ вобрал в себя практики ГОСТ Р 56939-2024

- Документ обновлён с учётом практики применения ГОСТ Р 56939-2024: **Разработка безопасного программного обеспечения** (РБПО)

- Изменения существенные
- Было 215 страниц
- Стало 382 страницы

ГОСТ Р 56939-2024: РБПО

- Для тех, кто хочет познакомиться с этим стандартом



- pvs-studio.ru/ru/webinar/rbpo/

Например, в новый ПЗ добавлен раздел AGD_OPE_EXT.1 – Правила кодирования

- В ГОСТ Р 56939-2024 это раздел:
  Процесс 8 — Формирование и поддержание в актуальном состоянии правил кодирования
- Соответственно, описание похоже. Пример:
  - ГОСТ: Учитывать при разработке регламента оформления исходного кода и безопасного кодирования примеры опасных и безопасных конструкций для используемых в ПО языков программирования.
  - ПЗ: Разработчик должен учитывать при разработке регламента оформления исходного кода и безопасного кодирования примеры опасных и безопасных конструкций для используемых в ПО языков программирования.

Таблица (стр. 245)

- В методическом документе есть подробная таблица 7.4.3.1:

Связь задач безопасного жизненного цикла в соответствии с ГОСТ Р 56939-2024 и шагов процесса безопасного жизненного цикла ОО раздела 7.4 ПЗ

Задача безопасного жизненного цикла	**Процесс безопасного жизненного цикла в соответствии с ГОСТ Р 56939-2024**	**Шаг процесса (Приложение В)**
Задача «Организационная подготовка»	- Планирование и определение области применения процессов РБПО (в том числе, анализ текущего статуса реализации процессов РБПО, анализ потребностей в ресурсах для реализации РБПО, планирование развития процессов РПБО); - Обучение сотрудников Разработчика в соответствии с планом обучения по актуальной программе обучения; - Повышение осведомленности сотрудников Разработчика в части обеспечения информационной безопасности при реализации процессов РБПО;	- Планирование процессов РБПО; Определение потребностей, в т.ч. в ресурсах; - Определение ролевого состава команды; - Организация обучения и повышения осведомленности сотрудников Разработчика; - Повышение квалификации и осведомленности членов команды Разработчика; - Разработка руководств по процессам РБПО; - Формирование и поддержание в актуальном состоянии правил

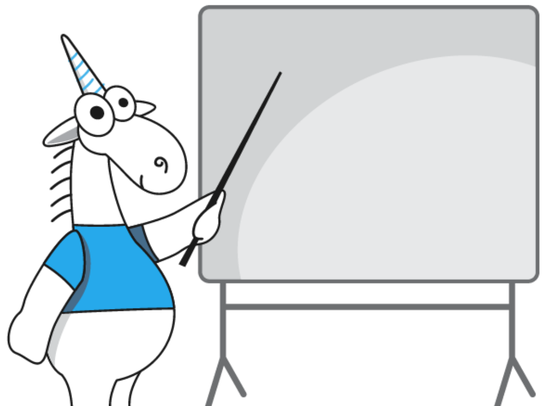
Почему стоит изучить и сам ГОСТ Р 56939-2024?

- Есть полезные детали, отсутствующие в ПЗ
- Например, в ГОСТ Р 56939-2024 второй процесс посвящён обучению сотрудников

- **ГОСТ Р 56939-2024. 5.2 Обучение сотрудников**
  В данном подразделе под обучением понимается совокупность методов и подходов, направленных на постоянное повышение квалификации, развитие профессиональных навыков, … и т.д.

Аналогичный раздел есть в ПЗ (стр. 255)

- **В рамках организации обучения и повышения осведомлённости работников ФО Разработчик должен:**
  - проводить анализ (не менее 1 раза в квартал) существующих (доступных для анализа) практик, документов, обучающих курсов и тренингов по РБПО;
  - проводить обучение (не менее 1 раза в пять лет) сотрудников типовым практикам РБПО с учётом актуальных потребностей;
  - осуществлять определение критериев пересмотра программ обучения (курсов, тренингов и т.п.);
  - … и т.д.



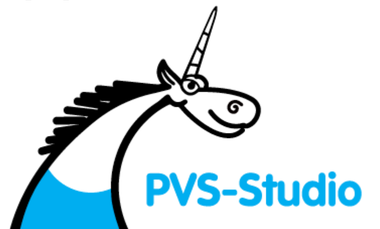
Но в ПЗ не сказано про артефакты

- Может быть полезно формализовать процесс и иметь артефакты реализации требований, перечисленные в ГОСТ:
  - 5.2.3.2 План обучения, включающий:
    - **список сотрудников, направляемых на обучение**;
    - сроки прохождения обучения;
    - наименование программы (курса, тренинга) обучения;
    - **ожидаемый результат обучения**.
  - 5.2.3.3 Артефакты реализации требований, **подтверждающие прохождение обучения**, включают (в зависимости от учебной программы, курса) свидетельства, дипломы, отчёты обучающих платформ и иные документы и материалы, подтверждающие прохождение сотрудником обучения.
  - И т.д.

Подробный разбор методического документа ПЗ ждёт своего героя

- Рассмотрим хорошо знакомую мне тему – статический анализ кода и PVS-Studio в частности
- Продолжаем разбираться в регламентах, актуальных для наших пользователей из ФИНТЕХ отрасли

- PVS-Studio – статический анализатор для поиска критических ошибок, опечаток и других дефектов в коде
- Сайт: pvs-studio.ru



Статический анализ кода

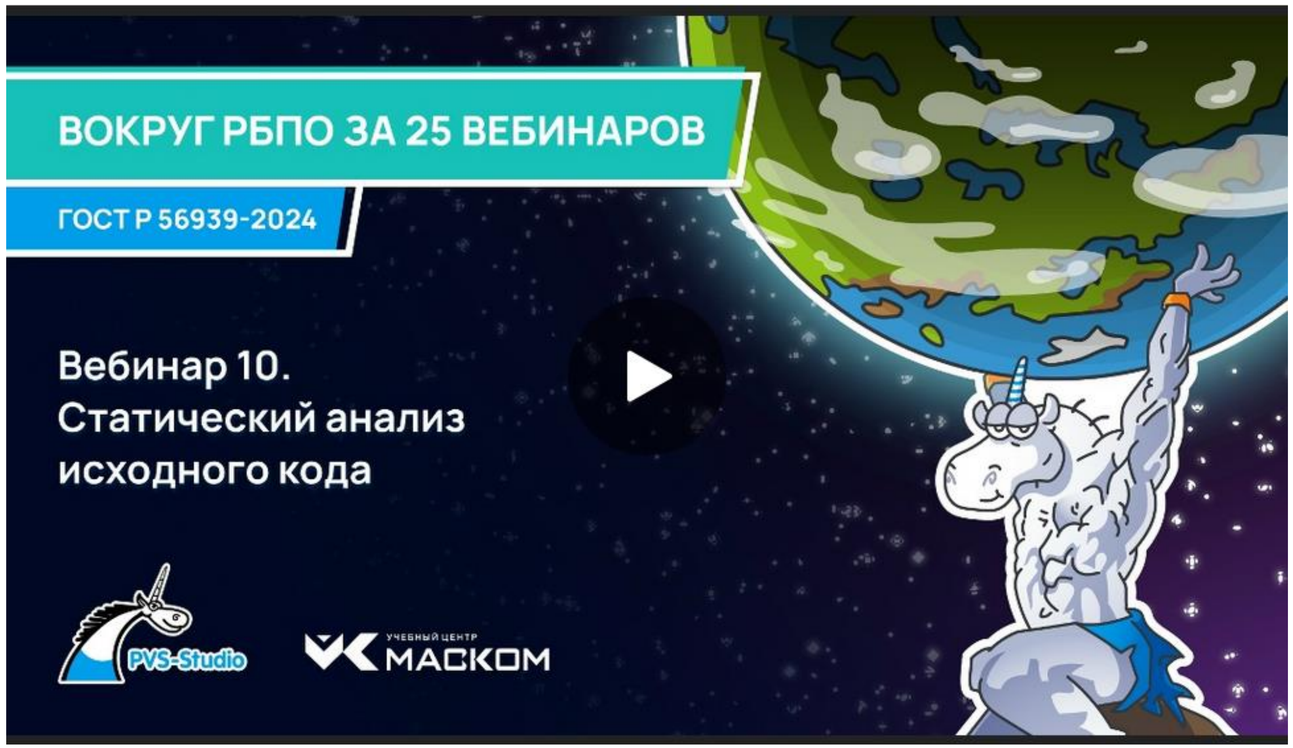
Статический анализ в ГОСТ Р 56939-2024

- **Процесс №10**

- 5.10 Статический анализ исходного кода
- 5.10.1 Цели
- 5.10.1.1 Предотвращение внесения потенциально опасных конструкций и ошибок в ПО, а также использования опасных конструкций и уязвимостей из заимствованного кода
- ...

Статический анализ в ГОСТ Р 56939-2024

- Вебинар 10. Статический анализ исходного кода



- pvs-studio.ru/ru/blog/video/11446/

Однако, внедряя статический анализ, следует обратиться к ГОСТ 71207-2024

ГОСТ 71207-2024
РАЗРАБОТКА БЕЗОПАСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.
Статический анализ программного обеспечения.
Общие требования.

- Впервые введён в действие 01.04.2024
- Доступен на сайте ФГБУ «Институт стандартизации»

Профиль защиты и ГОСТ 71207-2024

- Напрямую в ПЗ нет отсылки к ГОСТ 71207-2024
- Но видно, что в ПЗ учитывались описанные в стандарте практики и требования

- Например, перечисленные в документе методы анализа (см. ALC_TAT_EXT.1.2D), которые должны быть реализованы в инструментальном средстве, совпадают с перечисленными в пункте 7.4 ГОСТ Р 71207-2024

ALC_TAT_EXT.1.2D (стр. 158)

- Инструменты статического анализа должны реализовывать следующие методы анализа:
  - 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;
  - анализ программы на синтаксическом уровне.
- P.S. В документе 2021 года этого списка нет

ГОСТ Р 71207-2024, п.7.4

- Статический анализатор должен реализовывать следующие методы анализа:
  - 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;
  - анализ программы на синтаксическом уровне.

То, что требования ГОСТ и ПЗ совпадают, является логичным

- Это дополнительный аргументом обратиться к ГОСТ 71207-2024 при внедрении процесса статического анализа
- Стандарт расширит понимание методологии статического анализа:
  - Описано, как выбирать инструментальные средства анализа кода
  - Говорится о регулярности использования и времени исправления дефектов
  - Введено понятие «критические ошибки»
  - И т.д.

Что означает перечисление необходимых технологий?

- Можно использовать не любой инструмент, а те, которые выявляют дефекты безопасности, а не только, например, стиль именования переменных и функций

- В ГОСТ 71207-2024 для таких дефектов введён термин: 3.1.13 **критическая ошибка в программе:** Ошибка, которая может привести к нарушению безопасности обрабатываемой информации

Критические ошибки в программе

Для интерпретируемых языков это

- Ошибки непроверенного использования чувствительных данных
- Ошибки некорректного использования системных процедур и интерфейсов, связанных с обеспечением информационной безопасности
- Ошибки при работе с многопоточными примитивами

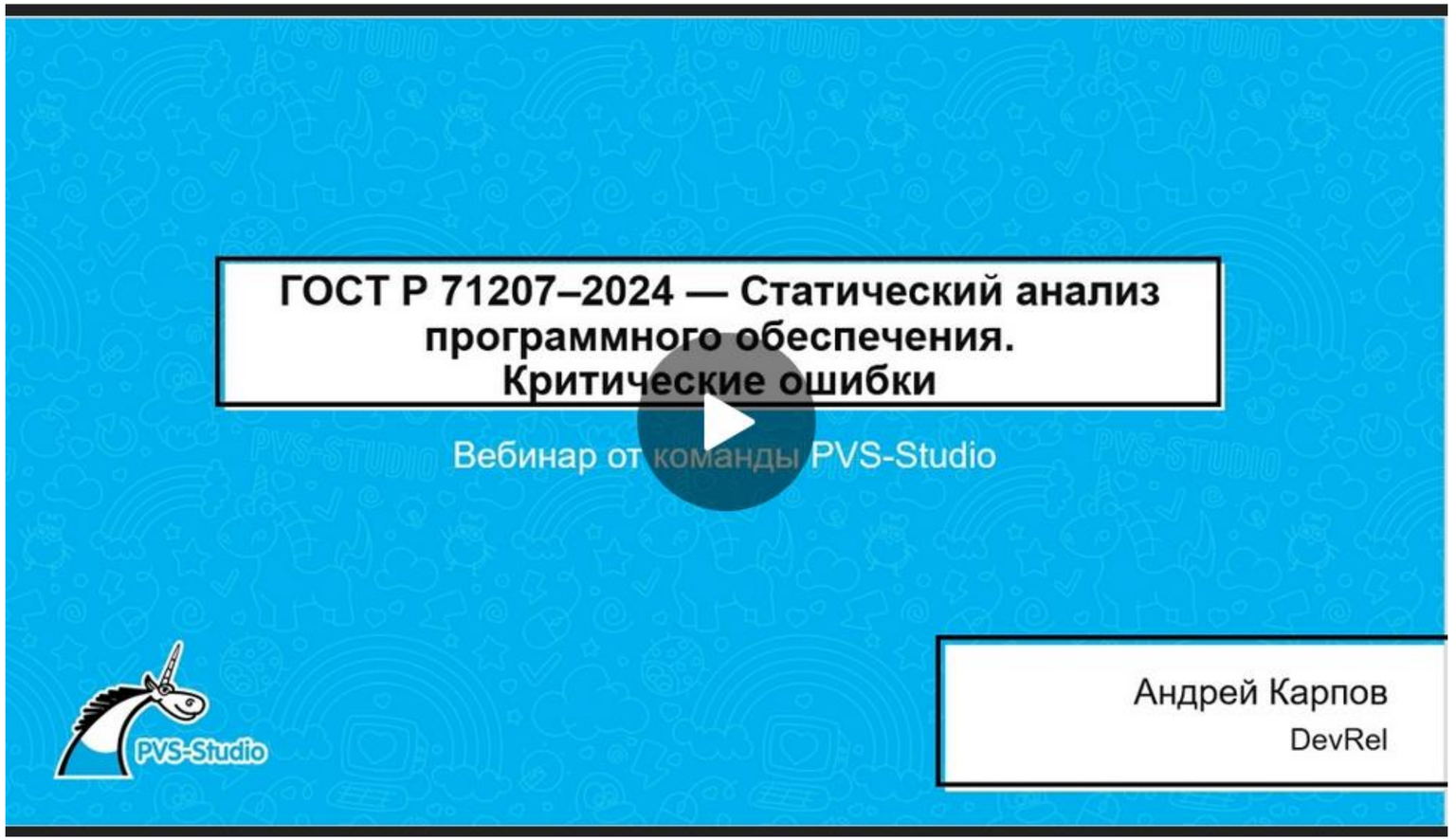
Для компилируемых языков это

- Ошибки непроверенного использования чувствительных данных
- Ошибки целочисленного переполнения и некорректного совместного использования знаковых и беззнаковых чисел
- Ошибки переполнения буфера
- Ошибки некорректного использования системных процедур и интерфейсов, связанных с обеспечением информационной безопасности
- Ошибки при работе с многопоточными примитивами

Дополнительные типы критических ошибок для C и C++

- Ошибки разыменования нулевого указателя
- Ошибки деления на ноль
- Ошибки управления динамической памятью
- Ошибки использования форматной строки
- Ошибки использования неинициализированных переменных
- Ошибки утечек памяти, незакрытых файловых дескрипторов и дескрипторов сетевых соединений

Подробнее про критические ошибки



- pvs-studio.ru/ru/blog/video/11084/

ГОСТ Р 71207

- **Вся серия докладов:**
- Общее описание и актуальность
- Терминология
- Критические ошибки
- Технологии анализа кода
- Процессы

Критические ошибки в программе

- Рассмотренные типы ошибок возможно выявлять только при наличии продвинутых методов (технологий) анализа

- Их нельзя найти с помощью сигнатурного анализа
  - 3.1.28 **сигнатурный анализ:** Статический анализ, определяющий наличие свойства программы при помощи поиска строк в исходном коде программы по некоторому образцу, в том числе заданному с помощью формального языка поиска, **например при помощи регулярных выражений**.

Методы анализа

Внутрипроцедурный анализ потоков данных и управления

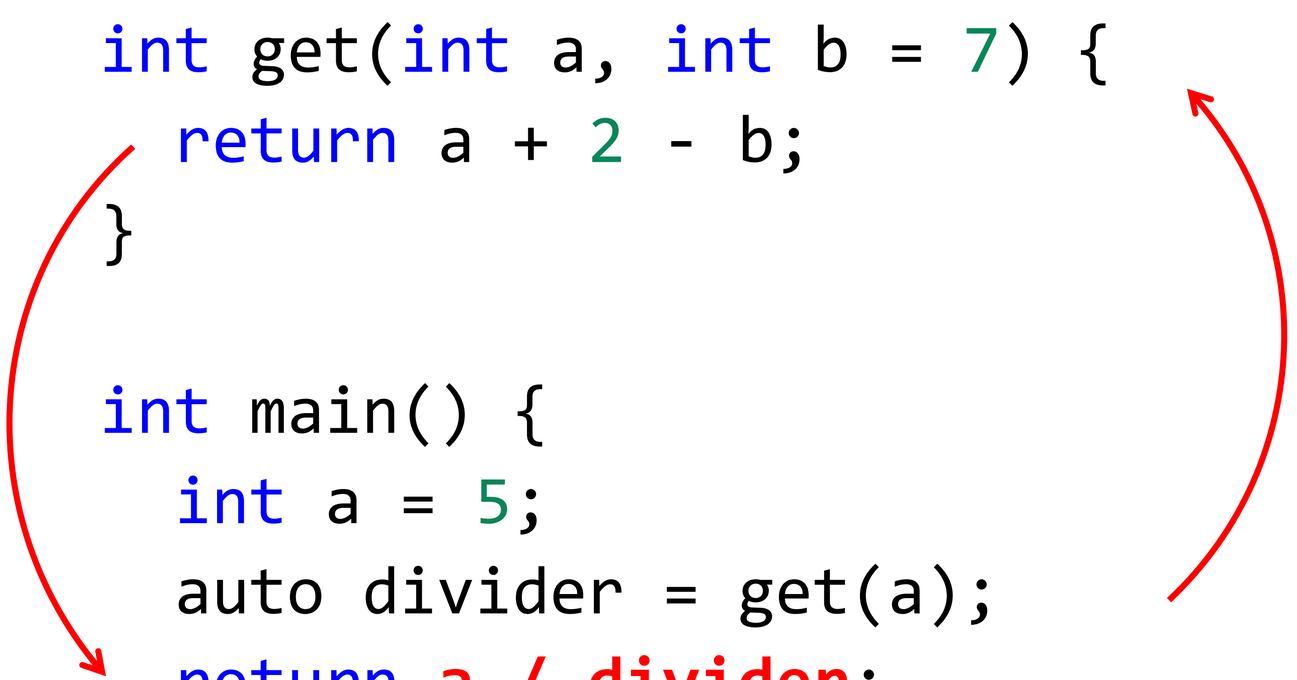
Java

```
private void updateOrdering(GeoElement geo, ObjectMovement movement) {
  ....
  if (index == firstIndex) {
    if (index != 0) {
      geo.setOrdering(orderingDepthMidpoint(index));
    }
    else {
      geo.setOrdering(drawingOrder.get(index - 1).getOrdering() - 1);
    }
  }
```

PVS-Studio: V6025 Index 'index - 1' is out of bounds.

Межпроцедурный и межмодульный контекстно-чувствительный анализ потока данных

C++



```
int get(int a, int b = 7) {
  return a + 2 - b;
}

int main() {
  int a = 5;
  auto divider = get(a);
  return a / divider;
}
```

PVS-Studio: V609 Divide by zero. Denominator 'divider' == 0.

Чувствительный к путям выполнения анализ потоков данных и управления

Java

```
private void checkRevocationUsingOCSP(X509Certificate[] certs)
                                        throws GeneralSecurityException {
  ....
  if (rs == null) {                                          // (A)
    if (_ocspFailOpen)
      logger.warnf(....);                                    // (Б)
    else
      throw new GeneralSecurityException(....);              // (В)
  }
  if (rs.getRevocationStatus() ==                            // (Г)
    OCSPProvider.RevocationStatus.UNKNOWN) {
```

Если rs хранит нулевую ссылку (точка А), то нормальный поток выполнения может быть прерван генерацией исключения (точка В). Но может и не прерываться (точка Б), что приведёт к доступу по нулевой ссылке (точка Г).

PVS-Studio: V6008 **Potential** null dereference of 'rs'.
CertificateValidator.java **701** , CertificateValidator.java **708**

Межпроцедурный и межмодульный контекстно-чувствительный анализ помеченных данных

C

```
static const char *basic_gets(int *cnt) {
  ....
  int c = getchar();
  if (c < 0) {
    if (fgets(command_buf, sizeof(command_buf) - 1, stdin)
          != command_buf) {
      break;
    }
    command_buf[strlen(command_buf)-1] = '\0'; /* remove endline */
    break;
  }
```

PVS-Studio: V1010 Unchecked tainted data is used in index: 'strlen(command_buf)'.

Анализ программы на синтаксическом уровне

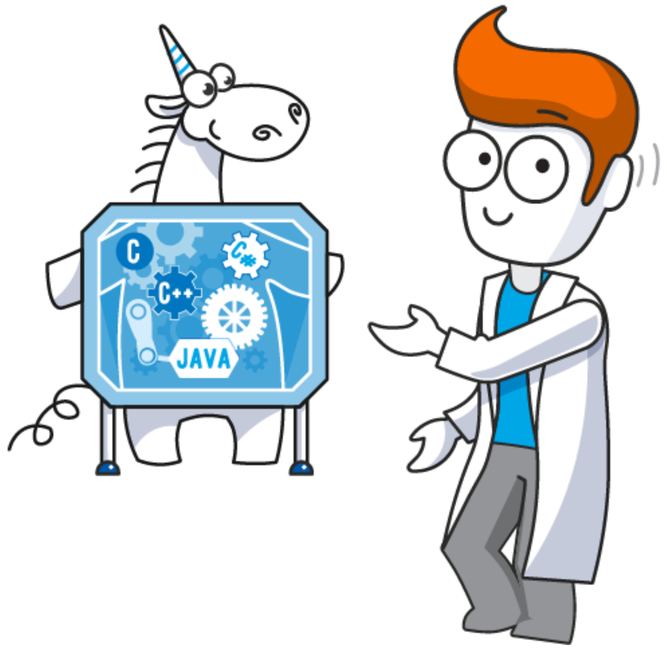
C

```
static void props_get_all_cb(....) {
  ....
  if (!item->menu_path)
    ERR("Notifier item %s dont have menu path.",
        item->menu_path);
```

PVS-Studio: V576 Incorrect format. Consider checking the third actual argument of the 'fprintf' function. A null pointer is used.

Подробнее про методы анализа на примере PVS-Studio

- Технологии, используемые в PVS-Studio



Выбор инструментов

- ALC_TAT_EXT.1.2D Разработчик должен определить инструменты статического анализа для каждого используемого в ПО языка программирования. Инструменты статического анализа должны реализовывать следующие методы анализа: … *(мы их только что рассмотрели)*

- Нет необходимости выбирать один инструмент
- Следует учитывать удобство для внедрения в ваши процессы

Дополнение. Взгляд ФСТЭК на выбор

- Методическая рекомендация ФСТЭК № 2025-07-011

  Тип недостатка: Необоснованный выбор инструментов, в том числе инструментов статического анализа исходного кода, для выстраивания и выполнения процессов РБПО

- Также смотрите выдержку из эфира AM Live "Разработка безопасного программного обеспечения (РБПО)"

Комментарии к некоторым разделам ПЗ, относящиеся к статическому анализу

ALC_TAT_EXT.1.4D (стр. 159)

- Разработчик должен проводить статический анализ с использованием инструментов статического анализа с регистрацией всех предупреждений о потенциальных ошибках, полученных по результатам работы инструментов статического анализа

- В ГОСТ Р 71207-2024 можно найти рекомендацию по временным рамкам. См. п. 5.5 и 5.8

ГОСТ Р 71207-2024 п 5.8

- Выданные статическим анализатором предупреждения должны быть размечены согласно 5.5. Для своевременной и эффективной работы с полученными результатами статического анализа просмотр выданных анализатором предупреждений следует выполнять:
  - **при анализе изменённых 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**

ALC_TAT_EXT.1.1C (стр. 159)

- Регламент проведения статического анализа исходного кода ПО должен содержать следующие сведения:
  - …
  - периодичность проведения статического анализа или события, при наступлении которых необходимо выполнять повторный статический анализ;
  - …
- И здесь можно оттолкнуться от ГОСТ

ГОСТ Р 71207-2024 п. 5.6

- …
- Статический анализ ПО в рамках жизненного цикла ПО следует проводить регулярно на этапе конструирования и комплексирования ПО. **Статический анализ всего разрабатываемого ПО следует выполнять не реже одного раза в 10 рабочих дней, если за данный период времени исходный код был изменён. Статический анализ добавленных или изменённых частей ПО следует выполнять после каждого внесённого изменения.**
- …

ALC_TAT_EXT.1 (стр. 158)

- Предотвращение внесения потенциально опасных конструкций и ошибок в ПО, а также использования опасных конструкций и уязвимостей **из заимствованного кода**.

- Следует проверять не только свой, но и используемый сторонний код
- ГОСТ Р 71207-2024 и ГОСТ Р 56939-2024 смотрят на этот вопрос аналогично

ADV_IMP.2.3D (стр. 96)

- Статический анализ кода проводится с использованием автоматизированных средств (программных инструментов) и **направлен на идентификацию потенциально опасных фрагментов кода**, в том числе:

- *Сейчас рассмотрим некоторые подпункты*

ADV_IMP.2.3D – A (стр. 98)

- Вызовов функциональных объектов (функций, методов, процедур) с передачей им в качестве аргументов данных, вводимых пользователем или принимаемых из внешних источников;

- Кажется, это другими словами сформулирован «анализ помеченных данных» (ГОСТ Р 71207-2024: п. 3.1.3).
- В ГОСТ (п. 7.6) говорится, инструмент должен предоставлять конфигурирование процедуры-источники и процедуры-стоки чувствительных данных

ADV_IMP.2.3D – Ж (стр. 99)

- Текстов функциональных объектов обработчиков ошибок и исключений

- Интересный пункт. Мы очень часто находим ошибки в обработчиках ошибок

ADV_IMP.2.3D (стр. 96)

- В ходе статического анализа кода необходимо проводить поиск **типовых ошибок программирования** (недостаточная проверка входных параметров функций, включение аутентификационных данных непосредственно в текст программ, некорректное преобразование типов, недостаточная обработка ошибок и исключений), а также определять статические пути исполнения программы

- Отсюда следует …

… поиск типовых ошибок …

- ПЗ не ограничивается при статическом анализе выявлением критических ошибок
- Не ограничивается ими и ГОСТ Р 71207-2024
- Критические ошибки – это про необходимый минимум, приоритет в их выявлении и исправлении

- Суть: в процессе статического анализатора должны выявляться критические ошибки и другие виды ошибок (при разумном вложении времени и сил)

PVS-Studio

PVS-Studio

- Разрабатывается с учётом требований, изложенных в ГОСТ Р 71207-2024
- Выявляет все виды критических ошибок, перечисленных в ГОСТ
- Реализует методы анализа, перечисленные в ПЗ / ГОСТ
- Соответствует требованиям, перечисленным в ПЗ

Поддерживаемые языки

- Сейчас: C, C++, C#, Java
- В 2026 году добавятся: Go, JavaScript, TypeScript
- Если смотрите доклад в записи, возможно, новые языки уже поддержаны. Актуальная информация на странице продукта: https://pvs-studio.ru/ru/pvs-studio/

Широкий спектр выявляемых дефектов

- ADV_IMP.2.3D: В ходе статического анализа кода необходимо проводить поиск типовых ошибок программирования…
- В PVS-Studio реализованы детекторы для всех типовых ошибок:
  - Неправильная работа с типами (HRESULT, BSTR, VARIANT_BOOL, float, double)
  - Арифметическое переполнение
  - Выход за границу массива
  - Мёртвый код
  - Опечатки
  - И т.д.
- Список всех предупреждений

PVS-Studio

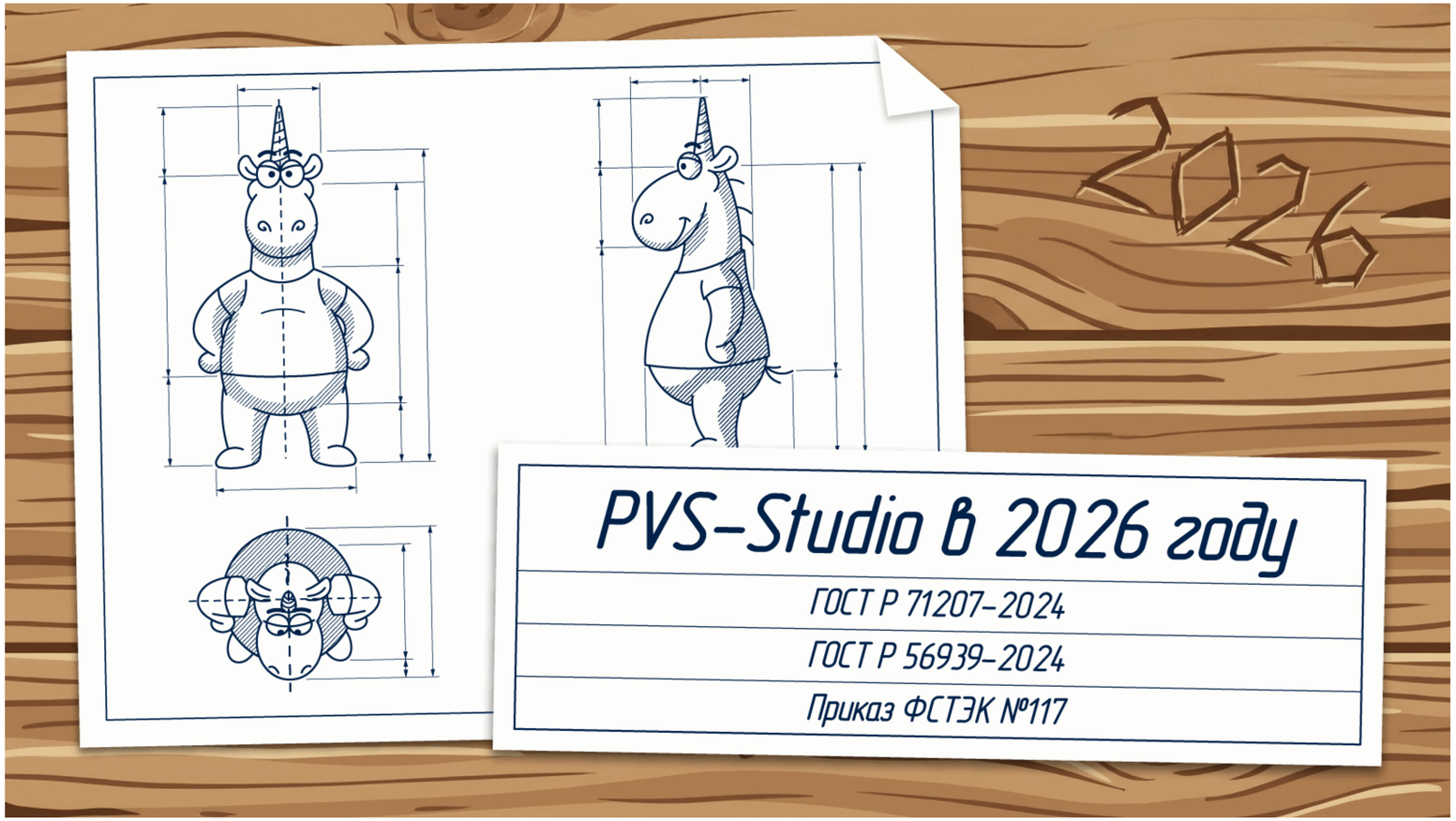
- Инструмент разрабатывается в России с 2008 года
- Запись в Едином Реестре российского ПО № 9837
- Может работать в закрытом контуре
- Запускается на большом количестве ОС, в том числе отечественных: Windows, macOS, Arch Linux, Astra Linux, CentOS, Debian GNU/Linux, Fedora, Linux Mint, openSUSE, Ubuntu, РЕД ОС и т.д.
- Подтверждена техническая совместимость анализатора PVS-Studio с Astra Linux. Сертификат № 31190/2025.

PVS-Studio

- В 2025 году мы приняли участие в мероприятии «**Испытания статических анализаторов исходных кодов** компилируемых и динамических языков программирования под руководством ФСТЭК России»
- Итоги испытаний были подведены в декабре 2025 года на открытой конференции ИСП РАН
- Кратко об итогах испытаний, а также ссылки на доклады и презентации pvs-studio.ru/ru/blog/posts/1332/

Подробнее о PVS-Studio

- Статический анализатор кода PVS-Studio в 2026: ГОСТ Р 71207, ГОСТ Р 56939, приказ ФСТЭК №117



Telegram-канал
PVS-Studio для бизнеса

Дополнительные материалы

Дополнительные материалы

- Компетенция: теория разработки безопасного ПО (РБПО)
  pvs-studio.ru/ru/blog/posts/1291/

- Ошибки Java по ГОСТу: обзор и примеры
  pvs-studio.ru/ru/blog/posts/java/1268/

- Статический анализ и ASOC: нулевая терпимость к ошибкам в проекте
  pvs-studio.ru/ru/blog/video/11504/

Отзывы клиентов

- Как крупнейший в РФ производитель сетевого оборудования Eltex сделал разработку безопаснее благодаря PVS-Studio
  pvs-studio.ru/ru/blog/posts/1315/

- Как СВД ВС использует PVS-Studio в сертифицируемой разработке защищённой ОС
  pvs-studio.ru/ru/blog/posts/1320/

Карпов Андрей Николаевич

- Карпов Андрей Николаевич, 1981
- ООО «ПВС», директор по развитию бизнеса
- Более 18 лет занимаюсь темой статического анализа кода и качества программного обеспечения. Автор большого количества статей, посвящённых написанию качественного кода на языке C++. Один из основателей проекта PVS-Studio. Долгое время являлся CTO компании и занимался разработкой C++ ядра анализатора. Основная деятельность на данный момент — развитие компании, обучение сотрудников и DevRel деятельность
- Другая информация и контакты