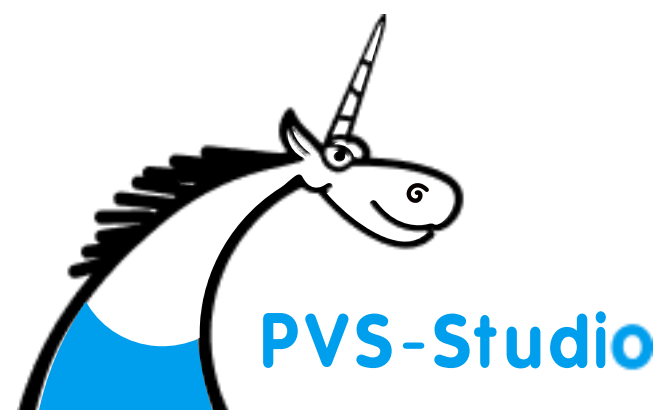


Испытания статических анализаторов кода

Домашнее задание – Послесловие



Андрей Карпов
Директор по развитию бизнеса



Андрей Карпов

ООО ПВС, Директор по развитию бизнеса

- Занимаюсь темой статического анализа более 18 лет
- Сооснователь проекта PVS-Studio
- Habr: [@Andrey2008](#)
- TG: [Бестиарий программирования](#)
- E-Mail: karpov (@) viva64.com



PVS-Studio

Сас Арустамян

АО «НПО «Эшелон», директор центра
оценки соответствия и тестирования

- Менеджер проекта АК-ВС 3
- Участник рабочей группы ТК 362,
разрабатывающей серии ГОСТ по РБПО
- Старший преподаватель МГТУ им. Баумана,
НИЯУ МИФИ
- Возглавляет команду, внедряющую РБПО
не в теории, а на практике для больших и
маленьких компаний



Эшелон
комплексная безопасность

Илья Антипов

АО «НПО «Эшелон», руководитель группы
центра оценки соответствия и тестирования

- Эксперт по статическому анализу
- Аспирант кафедры ИУ8
"Информационная безопасность" МГТУ
им. Баумана.



Эшелон
комплексная безопасность

Владимир Кочетков

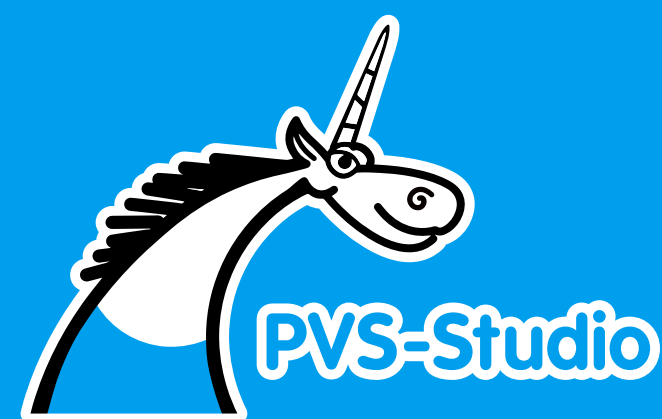
Руководитель отдела исследований и разработки анализаторов кода в Positive Technologies

- Эксперт по безопасности приложений, занимается этой областью уже свыше 15 лет
- Лидер сообщества безопасной разработки POSIdев
- Ведёт канал о прекрасном (и не очень) вокруг кода, ИИ и безопасности приложений:
t.me/art_code_ai



**positive
technologies**

Испытания статических анализаторов исходных кодов компилируемых и динамических языков программирования под руководством ФСТЭК России



- 1 апреля 2024 года введён в действие ГОСТ Р 71207-2024

Защита информации

Разработка безопасного программного обеспечения

Статический анализ программного обеспечения

Общие требования

- Нашу команду не вовлекли в разработку этого стандарта
- Однако понятно, что он важен для нас. Внимательно прочитав изложенные в нём требования, мы **начали вносить соответствующие доработки в PVS-Studio**

Говоря про ГОСТ, выделим один момент на тему проверки требований

8

- В ГОСТ Р 71207–2024 есть следующие числовые требования
- На синтетических тестах и на открытых проектах анализатор должен показать:
 - а) долю ошибок первого рода (ложноположительных срабатываний, FP) — не более 50 %;
 $FP \leq 50\%$
 - б) долю ошибок второго рода (ложноотрицательных срабатываний, т. е. пропусков заведомо известных ошибок, FN) — не более 50 %;
 $FN \leq 50\%$

Попутно я рассказывал про стандарт

9

- [PVS-Studio соответствует требованиям ГОСТ Р 71207–2024](#)
- Доклад на Certification Day 2024. [Статический анализ C++ кода по ГОСТ Р 71207-2024 на примере PVS-Studio](#)
- Доклад на GigaConf 2024. [Использование статического анализатора в разработке безопасного программного обеспечения \(ГОСТ Р 71207-2024\) на примере PVS-Studio](#)
- Цикл из 5 вебинаров, посвященных ГОСТ Р 71207–2024:
 - [Общее описание и актуальность](#)
 - [Терминология](#)
 - [Критические ошибки](#)
 - [Технологии анализа кода](#)
 - [Процессы](#)

Анализатор PVS-Studio доработан, статьи написаны, доклады сделаны, но...

10

- Странные общения с потенциальными заказчиками
- Есть «документ», подтверждающий, что PVS-Studio соответствует ГОСТ Р 71207–2024?
- Ээ... Его не существует...
- В ГОСТ Р 71207–2024 описан процесс проверки соответствия инструмента требованиям, но:
 - **Он трудоёмок.** Потенциальному заказчику нереально провести все необходимые испытания на высоком уровне
 - Если мы проведём, то к такому испытанию не будет доверия



Вопрос есть, а ответа нет

11

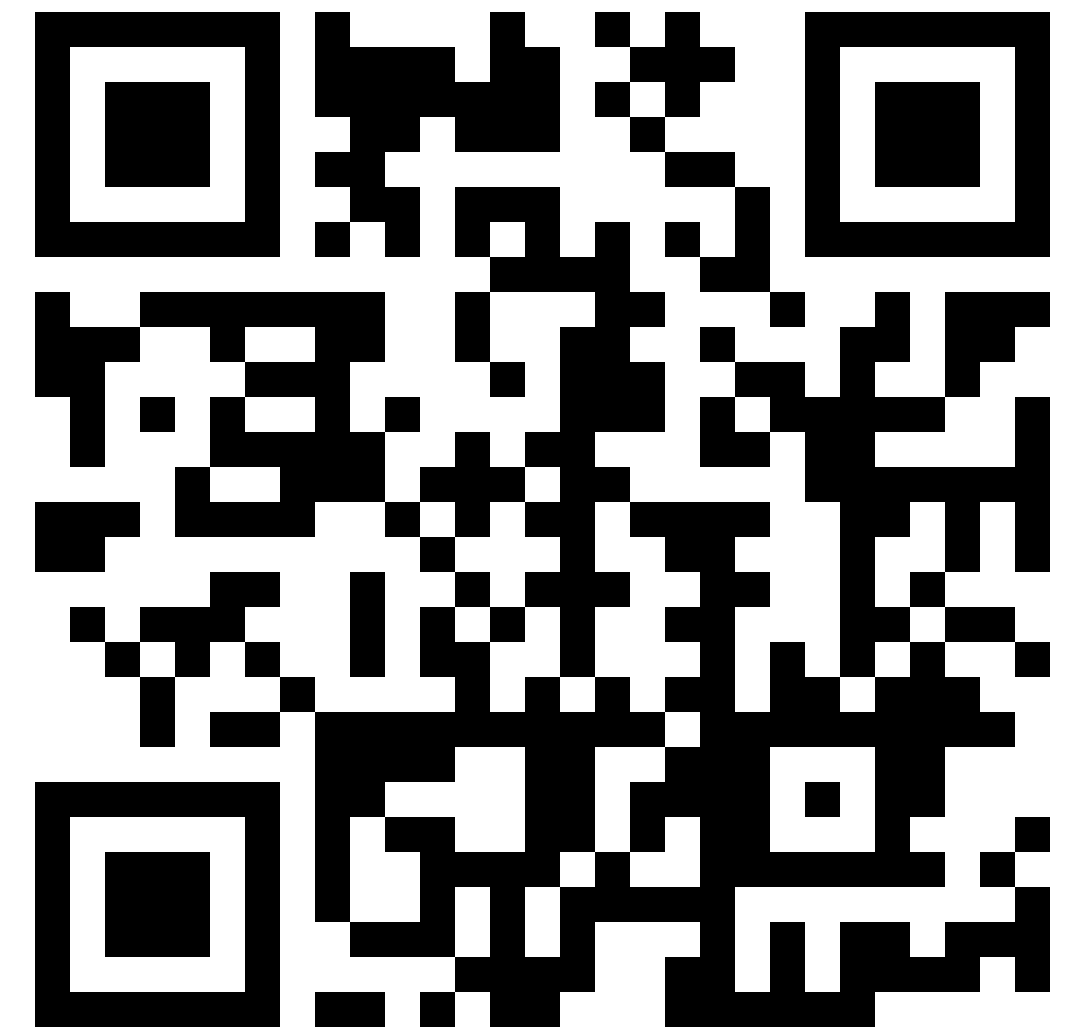
- Пользователи спрашивают нас и других разработчиков анализаторов
- Мы спрашиваем ФСТЭК в письме
- Компании спрашивают ФСТЭК
- Другие разработчики анализаторов спрашивают ФСТЭК
- **В общем, все друг друга спрашивают ☺**



Лёд тронулся в начале 2025 года

12

- [ФСТЭК России объявила о начале масштабных испытаний статических анализаторов](#)
- Из разработчиков анализаторов участие приняли:
 - АО «НПО «Эшелон»
 - АО «Позитив Текнолоджиз»
 - АО «СОЛАР СЕКЬЮРИТИ»
 - ИСП РАН
 - ООО «ПВС»
- Языки: C, C++, C#, Java, Go, Python, JavaScript



- [Испытания статических анализаторов](#)
(BIS Journal – «Информационная безопасность бизнеса», № 2(57) 2025, стр. 72–82)
- [Итоги этапа «Домашнее задание» испытаний статических анализаторов под патронажем ФСТЭК России](#)
- [Мой комментарий](#) касательно PVS-Studio в telegram-канале
- Сейчас идёт основной этап испытаний, но он выходит за рамки этого доклада

Что такое этап «Домашнего задания»

14

- Каждый предлагает свой набор тестов
- «Синхронизироваться» на синтетических тестах
- Понять, как нужно создавать тесты и делать замеры
- Насколько подход с тестами в целом работает
- Автоматизированный подсчёт процентов
- **Декларативный этап.** Каждый сам произвёл замеры и предоставил отчёт
- На этапе ДЗ верификации результатов нет

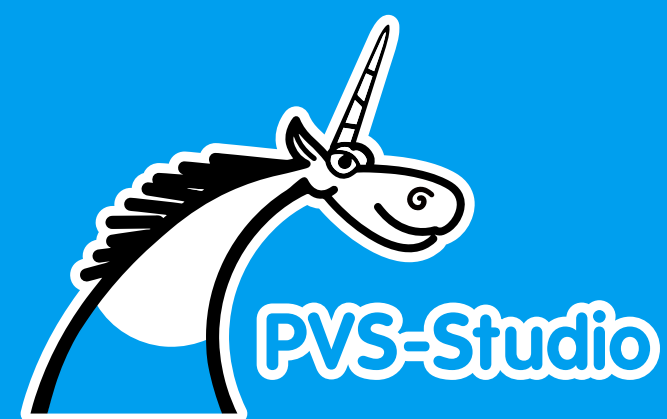
Тестируемые категории критических ошибок

15

- Основной набор (ГОСТ Р 71207-2024: п.6.3, 6.4, 6.5):
 - Ошибки непроверенного использования чувствительных данных
 - Ошибки целочисленного переполнения и некорректного совместного использования знаковых и беззнаковых чисел
 - И т.д.
- Дополнительный набор. Например, для Java, Go, C#, Python:
 - ошибки деления на ноль
- [Подробнее](#) про состав основного и дополнительного набора

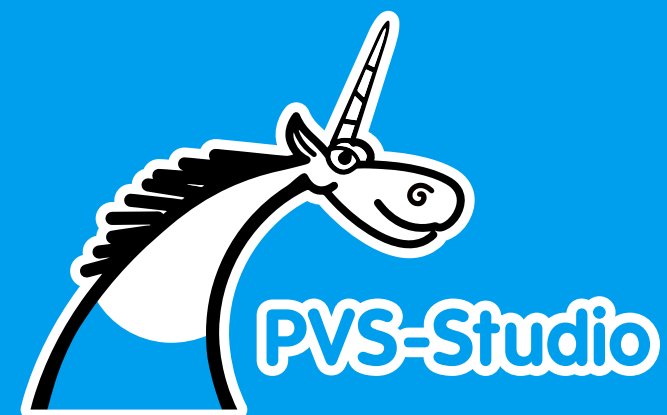


Что волнует и какие аспекты
хочется объяснить



- Результаты этапа «Домашнего задания» сложны для восприятия
- Я столкнулся с неправильной их интерпретацией и выводами
- Полезно внести ясность и объяснить некоторые моменты
- Например, следует разобрать понятие «Ложные срабатывания» (FP)
- Только тогда можно правильно толковать результаты и обсуждать показатели 50% FP, указанные в ГОСТ

Ложноположительные срабатывания (FP)



- Ложноположительное срабатывание
(False Positive, FP) - ложное срабатывание инструмента (проблема, о которой сообщил инструмент, на самом деле в коде отсутствует)
- Не путать с «Не актуально» (Won't fix) – истинное срабатывание, которое по тем или иным причинам исправлять нецелесообразно



- Без понимания природы FP не получится говорить о процентных показателях
- Кто-то в рекламных материалах говорит, что FP анализатора в районе 0%
- Мы спокойно говорим, что PVS-Studio выдаёт много ложных срабатываний
- В ГОСТ Р 71207-2024 говорится про 50% FP
- Как всё это понимать? Что хорошо, что плохо? Что означают проценты на практике?

Что не так с «процентом ложных срабатываний»

21

- Люди хотят простой ответ на вопрос: сколько ложных срабатываний у анализатора X?
- Но простого ответа не будет!
- Это не так работает
- Давайте разбираться



Давайте разбираться с FP

22

- Действительно, можно ориентировать статический анализатор на 0% ложных срабатываний
- 0% FP это очень просто!
- Есть виды предупреждений, у которых нет FP или почти нет



- Стилистические правила
- Sonar – Java: Literal suffixes should be upper case
 - Неправильно: `long long1 = 11;`
 - Правильно: `long long1 = 1L;`
- MISRA C, 2023: Rule 7.3 - The lowercase character “l” shall not be used in a literal suffix
 - Неправильно: `const uint64_t a = 0U11;`
 - Правильно: `const uint64_t a = 0ULL;`

- Do not use floating-point variables as loop counters
- Или используется, или не используется ☺ – FP 0%

```
void func(void) {  
    for (float x = 0.1f; x <= 1.0f; x += 0.1f) {  
        /* Loop may iterate 9 or 10 times */  
    }  
}
```


- Существует множество «профилактических» диагностик, которые не выдают ложные срабатывания
- Они полезны и занимают свою нишу
- Однако их недостаточно, когда мы говорим о глубоком статическом анализе для выявления сложных ошибок!

- Анализатор выдаёт 100500 «профилактических» предупреждений с FP 0%
- И выдаёт тысячи предупреждений про out-of-bound и т.д. с уровнем FP 99%
- Тогда получается, что в **среднем** анализатор очень «качественный»: FP, например, 1% 😊
- Мы получили абсолютно **достоверный**, но совершенно **бестолковый** показатель $FP = 1\%$



- Нужно «провести вычисления» (анализ потока данных).
Чтобы находить такие дефекты, как:
 - выход за границу массива;
 - разыменование нулевого указателя;
 - деление на 0.
- Соответствующим детекторам свойственен невысокий уровень FP, так как анализатор ругается, только если смог просчитать значения
- Но он может и ошибаться
- Для повышения точности нужна настройка

«Вычислительные» диагностики близки к критическим ошибкам ГОСТ Р 71207-2024

28

- ГОСТ Р 71207-2024 в основном про критические ошибки, которые можно найти с помощью анализа потока данных и его разновидностей:
 - ошибки переполнения буфера;
 - ошибки целочисленного переполнения;
 - ошибки разыменовывания нулевого указателя;
 - ошибки использования неинициализированных переменных;
 - ошибки утечек памяти;
 - и т.п.

Пример поиска нулевого указателя

29

```
int32_t ctgGetFetchName(...) {  
    STablesReq* pReq =  
        (STablesReq*)taosArrayGet(pNames, pFetch->dbIdx);  
    if (NULL == pReq) {  
        qError("fail to get the %dth tb in pTables, tbNum:%d",  
            pFetch->tbIdx,  
            (int32_t)taosArrayGetSize(pReq->pTables));  
        return TSDB_CODE_CTG_INTERNAL_ERROR;  
    }  
    ....  
}
```

PVS-Studio: V522 Dereferencing of the null pointer 'pReq' might take place. ctgUtil.c 1769

Пример, откуда берутся FP (связь переменных)

30

```
/* compute bytes to read -- error on overflow */
len = nitems * size;
if (size && len / size != nitems) {
    gz_error(state, Z_STREAM_ERROR, "request does not fit in a size_t");
    return 0;
}

#ifdef __clang_analyzer__
/* clang-analyzer does not see size==0 through len==0 below. */
if (!size)
    return 0;
#endif

/* read len or fewer bytes to buf, return the number of full items read */
return len ? gz_read(state, buf, len) / size : 0;
```

Ошибся и PVS-Studio: V609 Divide by zero. gzread.c 405

- Например, нередко функции не размечены как `[[noreturn]]`, хотя таковыми являются:
 - Можно отрефакторить код, добавив `[[noreturn]]`
 - Если код править нельзя, можно использовать механизм разметки функций в PVS-Studio
- В разных проектах по-разному считается, могут функции выделения памяти возвращать NULL или нет

```
struct tree_entry *te;  
te = calloc(1, sizeof(*te));  
te->next = t->stack; // Детектировать разыменование  
                    // нулевого указателя или нет?
```

Эвристические детекторы с высоким процентом FP

32

- **Then** и **else** ветка оператора **if** идентичны. Однозначно полезная диагностика! [Proofs](#). Но сколько исключений не делай, много FP



- Этот код подозрителен? Есть здесь ошибка?

```
cm::optional<EventLogger> const o1{ 1 };  
cm::optional<EventLogger> const o2{ 2 };  
cm::optional<EventLogger> const o3{ 2 };
```

Здесь FP, но сложно научить анализатор ПОНИМАТЬ ВЫСОКОУРОВНЕВЫЕ ЗАДУМКИ

33

```
static bool testComparison(std::vector<Event>& expected)
{
    cm::optional<EventLogger> const o1{ 1 };
    cm::optional<EventLogger> const o2{ 2 };
    cm::optional<EventLogger> const o3{ 2 };
    cm::optional<EventLogger> const o4{};
    cm::optional<EventLogger> const o5{};
    EventLogger const e1{ 2 };

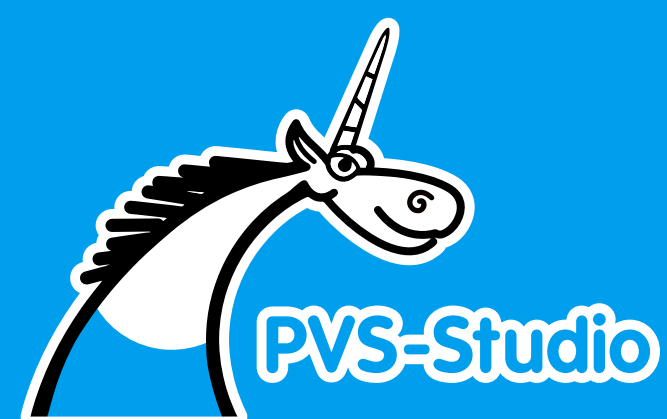
    ASSERT_TRUE(!(o1 == o2) && o1 != o2);
    ASSERT_TRUE(o1 < o2 && !(o1 >= o2));
    ASSERT_TRUE(!(o1 > o2) && o1 <= o2);

    ASSERT_TRUE(o2 == o3 && !(o2 != o3));
    ASSERT_TRUE(!(o2 < o3) && o2 >= o3);
    ASSERT_TRUE(!(o2 > o3) && o2 <= o3);
}
```

FP PVS-Studio: V525 The code contains the collection of similar blocks. Check items '1', '2', '2' in lines 522, 523, 524. testOptional.cxx 522

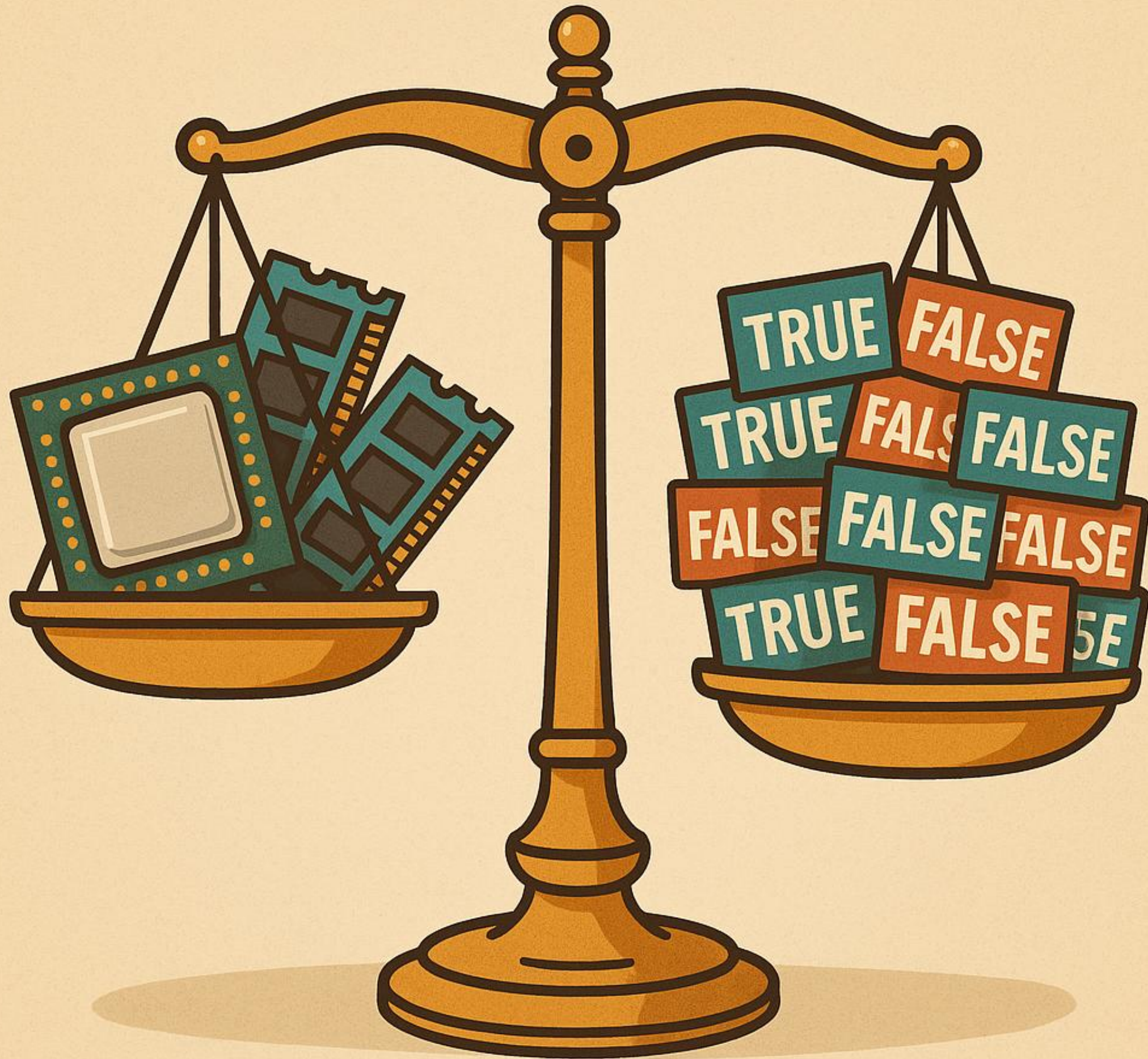


Комментарий Владимира Кочеткова



Откуда берутся фолзы

35



Любой подход к анализу — это компромисс между ресурсами и метриками качества

Множество состояний любого нетривиального приложения сравнимо с бесконечным

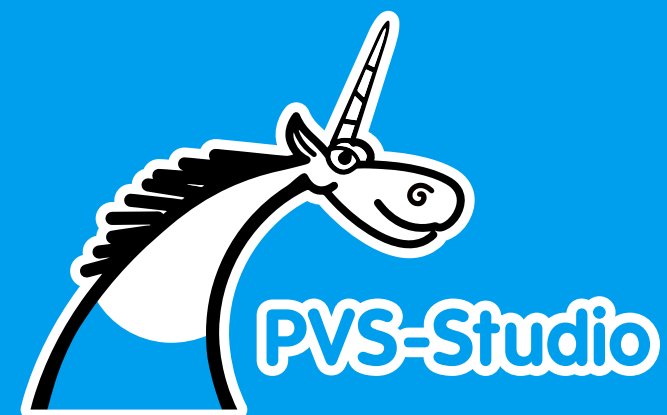
Задача анализатора — найти дефектные состояния или их признаки

Невозможно перечислить бесконечное множество состояний за конечное время

Какие-то состояния будут теряться из-за упрощения и приводить к ложным срабатываниям

Метрика	Формула	Назначение
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Не следует использовать для несбалансированных случаев
Recall	$\frac{TP}{TP + FN}$	Используется, когда FN дороже FP
FPR	$\frac{FP}{FP + TN}$	Используется, когда FP дороже FN
Precision	$\frac{TP}{TP + FP}$	Следует использовать, когда важна точность положительных сработок

Подытожим про ложные срабатывания (FP)



- Вне контекста типов детекторов бессмысленно рассматривать и обсуждать % FP
- ГОСТ Р 71207-2024 говорит про замер **критических ошибок**, которые дают средний % FP
- Замеры на синтетических тестах в целом возможны
- Есть сложности и проблемы, но % FP для критических ошибок имеет рамки, а не полностью абстрактен

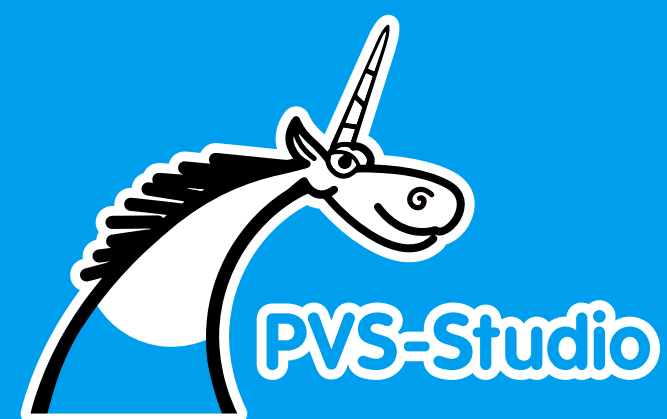
- В PVS-Studio реализовано много эмпирических диагностик для поиска опечаток
- Это делает его по умолчанию шумным (много FP)
- При этом, например, есть моя статья “[Характеристики анализатора PVS-Studio на примере EFL Core Libraries, 10-15% ложных срабатываний](#)”
- Здесь нет никакого противоречия!
- Как так?

- Перед использованием, анализатор нужно настроить:
 - Выключить неактуальные диагностики
 - Разметить функции, макросы
 - Настроить детекторы
 - Ослабить настройки для файлов с тестами
 - И т.д.
- Уровень FP станет приемлемым для работы



- Жаль, что ГОСТ Р 71207-2024 , описывая замеры при испытаниях **на открытых проектах, не говорит о какой-либо настройке** инструментов
- На мой взгляд, это делает замеры малополезными, несмотря на то, что они касаются только критических ошибок
- Я планирую предложить мысли по доработке стандарта в будущем
- Но это отдельная тема и история

Итоги этапа «Домашнее задание»



- Выложен под свободной лицензией GPLv2
- Специалисты «Базальт СПО» содействовали с размещением комплекта в открытом доступе на ресурсе altlinux.space



- У меня не осталось изначальной картинки, но на ней половина ячеек в таблицах была красной
 - Однако где-то в течение 2 месяцев всё сильно преобразилось
 - Для **основного** и **дополнительного** набора тестов достигнуты показатели, заданные стандартом
- (кроме одного пункта: FP для разыменования нулевого указателя в C/C++)

- **TP** – True Positive – Сколько ошибок выявлено
- **TN** – True Negative – Сколько обманок обнаружено
- **FP** – False Positive – Ложноположительные срабатывания (ошибки первого рода) (требование ГОСТ: **не более 50 %**)
- **FN** – False Negative – Ложноотрицательные срабатывания, т. е. пропуски заведомо известных ошибок (ошибки второго рода) (требование ГОСТ: **не более 50 %**)
- **TP** обратно пропорционально **FN**
- **TN** обратно пропорционально **FP**

PVS-Studio, замеры этапа ДЗ, С и С++

46

Проценты по группам ГОСТ					
C, C++					
Группа ГОСТ	TP (относ. файлов с FLAW) > 50%	TN (относ. файлов БЕЗ FLAW) > 50%	FP (относ. файлов БЕЗ FLAW) < 50%	FN (относ. файлов с FLAW) < 50%	Время анализа
SEC-STR-FORMAT	89 / 100 (89.0 %)	54 / 96 (56.3 %)	42 / 96 (43.8 %)	11 / 100 (11.0 %)	00:03
SEC-BUF-OVERFLOW	1248 / 1717 (72.7 %)	1099 / 1860 (59.1 %)	761 / 1860 (40.9 %)	469 / 1717 (27.3 %)	00:27
SEC- SYNCHRONIZATION	8 / 8 (100.0 %)	8 / 8 (100.0 %)	0 / 8 (0.0 %)	0 / 8 (0.0 %)	00:02
SEC-MEMORY	506 / 660 (76.7 %)	316 / 580 (54.5 %)	264 / 580 (45.5 %)	154 / 660 (23.3 %)	00:14
SEC-NULL	501 / 615 (81.5 %)	326 / 785 (41.5 %)	459 / 785 (58.5 %)	114 / 615 (18.5 %)	00:16
SEC-TAINT	620 / 915 (67.8 %)	381 / 676 (56.4 %)	295 / 676 (43.6 %)	295 / 915 (32.2 %)	00:12
SEC-OVERFLOW-OR- INT-UINT	573 / 833 (68.8 %)	447 / 778 (57.5 %)	331 / 778 (42.5 %)	260 / 833 (31.2 %)	00:15
SEC-UNINITIALIZED	237 / 392 (60.5 %)	616 / 728 (84.6 %)	112 / 728 (15.4 %)	155 / 392 (39.5 %)	00:13
SEC-SECURITY	12 / 19 (63.2 %)	10 / 10 (100.0 %)	0 / 10 (0.0 %)	7 / 19 (36.8 %)	00:01
SEC-DIV-0	547 / 992 (55.1 %)	1193 / 1541 (77.4 %)	348 / 1541 (22.6 %)	445 / 992 (44.9 %)	00:29
SEC-LEAKS	239 / 463 (51.6 %)	656 / 723 (90.7 %)	67 / 723 (9.3 %)	224 / 463 (48.4 %)	00:14
Всего	4580 / 6714 (68.2 %)	5106 / 7785 (65.6 %)	2679 / 7785 (34.4 %)	2134 / 6714 (31.8 %)	02:31

PVS-Studio, замеры этапа ДЗ, C#

47

Проценты по группам ГОСТ					
C#					
Группа ГОСТ	TP (относ. файлов С FLAW) > 50%	TN (относ. файлов БЕЗ FLAW) > 50%	FP (относ. файлов БЕЗ FLAW) < 50%	FN (относ. файлов С FLAW) < 50%	Время анализа
SEC-BUF-OVERFLOW	28 / 35 (80.0 %)	35 / 35 (100.0 %)	0 / 35 (0.0 %)	7 / 35 (20.0 %)	00:12
SEC- SYNCHRONIZATION	7 / 7 (100.0 %)	6 / 6 (100.0 %)	0 / 6 (0.0 %)	0 / 7 (0.0 %)	00:20
SEC-MEMORY	755 / 910 (83.0 %)	570 / 800 (71.3 %)	230 / 800 (28.8 %)	155 / 910 (17.0 %)	00:28
SEC-NULL	1616 / 2160 (74.8 %)	1787 / 2038 (87.7 %)	251 / 2038 (12.3 %)	544 / 2160 (25.2 %)	00:48
SEC-TAINT	854 / 971 (88.0 %)	540 / 549 (98.4 %)	9 / 549 (1.6 %)	117 / 971 (12.0 %)	00:51
SEC-OVERFLOW-OR- INT-UINT	1424 / 1837 (77.5 %)	1283 / 1637 (78.4 %)	354 / 1637 (21.6 %)	413 / 1837 (22.5 %)	00:35
SEC-SECURITY	10 / 11 (90.9 %)	11 / 11 (100.0 %)	0 / 11 (0.0 %)	1 / 11 (9.1 %)	00:11
SEC-DIV-0	1493 / 1853 (80.6 %)	1071 / 1386 (77.3 %)	315 / 1386 (22.7 %)	360 / 1853 (19.4 %)	00:34
SEC-LEAKS	427 / 600 (71.2 %)	548 / 558 (98.2 %)	10 / 558 (1.8 %)	173 / 600 (28.8 %)	00:21
Всего	6614 / 8384 (78.9 %)	5851 / 7020 (83.3 %)	1169 / 7020 (16.7 %)	1770 / 8384 (21.1 %)	04:23

PVS-Studio, замеры этапа ДЗ, Java

48

Проценты по группам ГОСТ					
Java					
Группа ГОСТ	TP (относ. файлов с FLAW) > 50%	TN (относ. файлов БЕЗ FLAW) > 50%	FP (относ. файлов БЕЗ FLAW) < 50%	FN (относ. файлов с FLAW) < 50%	Время анализа
SEC-BUF-OVERFLOW	699 / 1263 (55.3 %)	731 / 1122 (65.2 %)	391 / 1122 (34.8 %)	564 / 1263 (44.7 %)	00:16
SEC- SYNCHRONIZATION	3 / 3 (100.0 %)	0 / 0 (NaN %)	0 / 0 (NaN %)	0 / 3 (0.0 %)	00:03
SEC-MEMORY	343 / 547 (62.7 %)	276 / 489 (56.4 %)	213 / 489 (43.6 %)	204 / 547 (37.3 %)	00:14
SEC-NULL	1727 / 3073 (56.2 %)	1920 / 2857 (67.2 %)	937 / 2857 (32.8 %)	1346 / 3073 (43.8 %)	01:55
SEC-TAINT	1208 / 1267 (95.3 %)	753 / 1466 (51.4 %)	713 / 1466 (48.6 %)	59 / 1267 (4.7 %)	00:19
SEC-OVERFLOW-OR- INT-UINT	1233 / 2250 (54.8 %)	1528 / 2147 (71.2 %)	619 / 2147 (28.8 %)	1017 / 2250 (45.2 %)	00:21
SEC-SECURITY	12 / 18 (66.7 %)	8 / 9 (88.9 %)	1 / 9 (11.1 %)	6 / 18 (33.3 %)	00:07
SEC-DIV-0	1039 / 1869 (55.6 %)	1098 / 1632 (67.3 %)	534 / 1632 (32.7 %)	830 / 1869 (44.4 %)	00:18
SEC-LEAKS	576 / 726 (79.3 %)	598 / 765 (78.2 %)	167 / 765 (21.8 %)	150 / 726 (20.7 %)	00:14
Всего	6840 / 11016 (62.1 %)	6912 / 10487 (65.9 %)	3575 / 10487 (34.1 %)	4176 / 11016 (37.9 %)	03:52

- В статье: «Показатели PVS-Studio по TPR и TNR выше 50%...»
- Отзывы и комментарии: Ммм, жаль что у вас $TP > 50\%$ ☹
- Так ведь это наоборот хорошо! ☺ Видимо путаница с FP
- Итоговые средние показатели PVS-Studio на этапе ДЗ:
 - C, C++: FP = 34 % (требуется $\leq 50\%$)
 FN = 32 % (требуется $\leq 50\%$)
 - C#: FP = 17 % (требуется $\leq 50\%$)
 FN = 21 % (требуется $\leq 50\%$)
 - Java FP = 34 % (требуется $\leq 50\%$)
 FN = 38 % (требуется $\leq 50\%$)

За два месяца удалось так сильно доработать анализатор PVS-Studio?

50

- Действительно, было **много хороших доработок**
- Однако ощущения остались двойственные



- Улучшены механизмы анализа потока данных
- Новые исключения в детекторах
- Более точное расставление меток для фильтрации критических ошибок
- Несколько новых детекторов

- Дополнительная разметка функций
 - Как истоки/стоки (для поиска ошибок непроверенного использования чувствительных данных)
Например, `sethostname` был не размечен как сток данных
 - Оказывается, не умели вычислять результат работы функции `std::max`
 - И т.д.



- Разделение некоторых предупреждений на два вида
- Например, V575 – функция принимает странный аргумент
- Но не каждое такое предупреждение означает критическую ошибку разыменования нулевого указателя

```
int* p = NULL;
```

```
// Критическая ошибка разыменования нулевого указателя  
memset(p, 0, 10); // Есть V575, есть ГОСТ маркер
```

```
// Странный код, но не ошибка разыменования нулевого указателя  
free(p); // По-прежнему есть V575, но нет маркера
```

- Многие тесты не проходили по причине, что ожидался просто другой вариант маркировки предупреждений
- Например, этот код должен одновременно порождать предупреждения типа SEC-BUF-OVERFLOW и SEC-TAINT

```
int* p = new int[10];  
int n;  
scanf("%d", &n);  
p[n] = 1;
```

- Теперь диагностика V1010 имеет обе пометки, но больше от этого анализатор находить не стал

- Категория C++ тестов для 6.3.г – Ошибки некорректного использования системных процедур и интерфейсов, связанных с обеспечением информационной безопасности
- Мы подтянули анализатор на 30% в этой категории
- Круто?
- Ну как сказать...
- Там было 19 тестов с ошибками, из которых 6 (30%) про использование маски 0777

```
int file;  
void func(void) {  
    int mode = 0;  
    mode = 0777;  
    file = open("dummy.txt", O_WRONLY | O_CREAT, mode); // FLAW  
    close(file);  
}
```

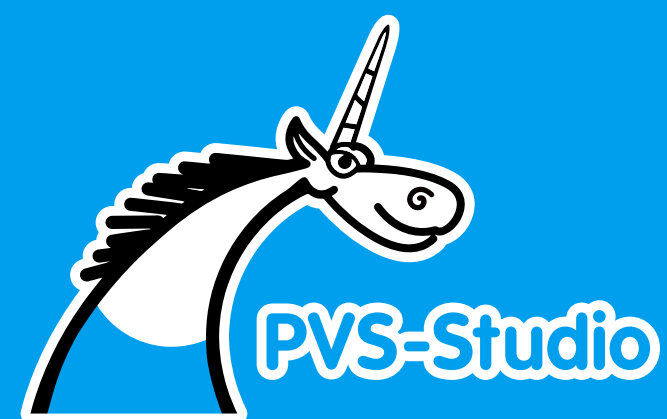
- Сделали новую диагностику [V1118](#). Полезно? Полезно
- Но после этого анализатор не стал находить на 30% больше реальных ошибок рассматриваемой категории 😊

- Дополнительный раздел тестов для C#: утечка ресурсов

```
public void func(string f)
{
    using (BufferedStream os =
        new BufferedStream(new FileStream(f, FileMode.Create))) // FLAW
    {
        os.Close();
    }
}
```

- Одна новая диагностика [V3222](#) дала прирост в +35%
- Ощущение подгонки для результата

Проблемные моменты и сложности ДЗ



- Участники являются первопроходцами
- Когда делаешь новое, в процессе совершаются ошибки и выявляются сложности
- Смысл первого испытания анализаторов – во многом как раз выявить такие моменты и учесть

Сложности маппинга диагностик на категории критических ошибок ГОСТ

60

- В 2024 году мы думали, что сделали «ГОСТ-выборку»
- Тесты показали, что это мы только думали 😊
- Переделки, т.к. оказалось, что некоторые диагностики PVS-Studio то должны быть с ГОСТ-маркером, то не должны. То вообще должны разделяться на 2-3 маркера
- Вопрос, как быть, например, с открытыми инструментами. Там потенциально те же сложности. Кто будет делать выборки, «разделять» диагностики на разные категории и т.д.?



- ИСП РАН создали генератор синтетических тестов, что позволило в буквальном смысле генерировать тысячи тестов
- Плюс. Это создало массив тестов. Без их вклада наборы явно были бы маловаты
- Минус. Оказалось невозможно проверить, насколько тесты корректны
- В процессе в них было выявлено и исправлено множество ошибок. Но неизвестно, сколько их осталось и насколько они влияют на результат

Проблематика генерации тестов

62

```
int* global;
void func(void)
{
    int number = 0;
    int var;
    scanf("%d", &number);
    var = (number > INT_MAX / sizeof(int) ? 0 : 1);
    if (var) {
        global = (int*)malloc(number * sizeof(int)); // no overflow
    }
    free(global);
}
```

Тип теста: Ошибки целочисленного переполнения и некорректного совместного использования знаковых и беззнаковых чисел.

Значение number ограничено проверкой, поэтому считается, что тест не содержит ошибок, но...

Ошибки целочисленного переполнения и некорректного совместного использования знаковых и беззнаковых чисел

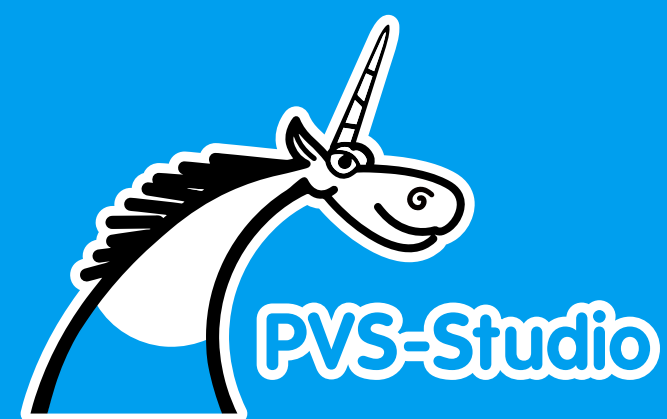
63

```
int number = 0;
scanf("%d", &number); // number может быть < 0
// Проверка действительно не пропускает некорректные
// значения дальше. Но при этом она сама попадает под
// определение:
// ошибка некорректного совместного использования
// знаковых и беззнаковых чисел
var = (number > INT_MAX / sizeof(int) ? 0 : 1);
```

- Тесты проверяют в огромном количестве вариаций достижимость/недостижимость кода
- Но при этом сами используемые конструкции однообразны
- Например, в **14099 файлах**, подготовленных ИСП РАН для C и C++ **нет ни одной операции сдвига <<, >>**
- Хотя это нередкие операции, которые могут быть связаны с ошибками переполнения
- Наш десяток тестов с << тонет в количестве и не влияет на оценку

- В сгенерированных тестах нет:
 - `std::array`
 - `std::string`
 - `realloc`
- Зато есть:
 - 940 операторов `goto` (используется в каждом 15-ом тесте)
 - Кажется, на практике `goto` всё же не настолько распространён ☺
- Это сделало тестирование на этапе ДЗ однобоким

Пара слов про основной этап



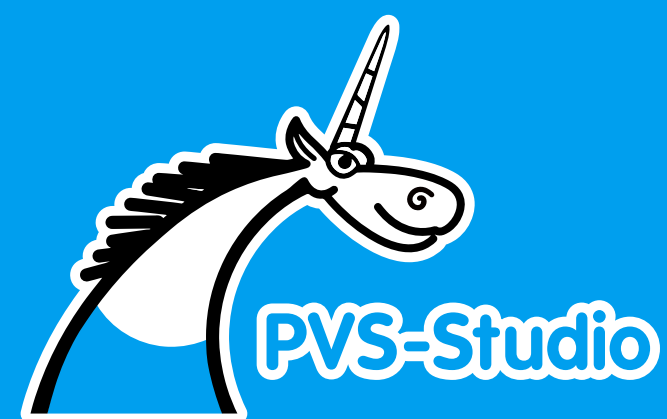
Дальше идут испытания:

67

- Сложные синтетические примеры с ручной валидацией для проверки используемых технологий
- Изучение результатов проверки выбранных открытых проектов с ручной валидацией
- Проверка скорости работы анализаторов на больших открытых проектах

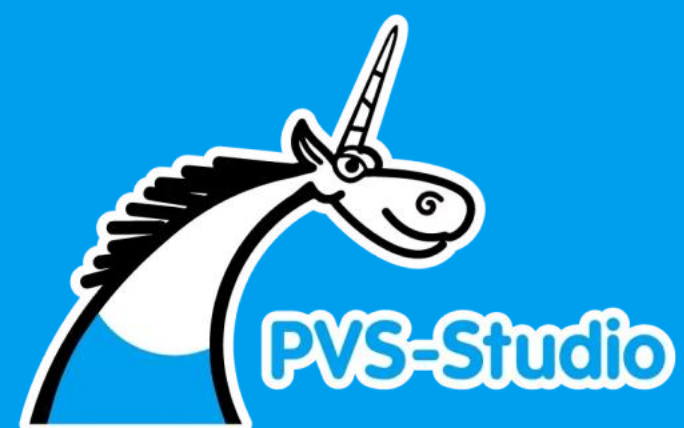
- В процессе уже выясняется множество сложностей и проблем
- Например, всплывает проблема пересечения разных технологий в рамках одного теста (а проверяется одна)
- **Но пока рассказывать про это и обсуждать рано**

Заключение и мысли

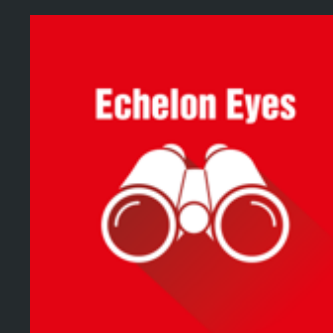


- По тому, как ресурсоёмко проходят испытания, теперь уже однозначно видно, что в одиночку компаниям заниматься подобным нереально/нерационально. Ощущение сложности теперь подтвердилось на практике 😊
- Считаю, что нужна модификация формулировки и подхода, описанного в ГОСТ Р 71207–2024 к оценке $FP \leq 50\%$ на открытых проектах. Требуется привязка к настройкам и качеству взятого проекта. Впрочем, это уже выходит за рамки обзора ДЗ

Авторский Telegram-канал
Андрея Карпова



Echelon Eyes
Новости ИБ: угрозы,
уязвимости, утечки,
инциденты,
аналитические обзоры,
изменения в нормативной
базе от экспертов ГК
«Эшелон».



Бесплатные SAST-плагины
для IDE

