PVS-Studio

Статический анализатор кода

Windows/Linux, C/C++/C#

ООО «СиПроВер»

Сайт: www.viva64.com

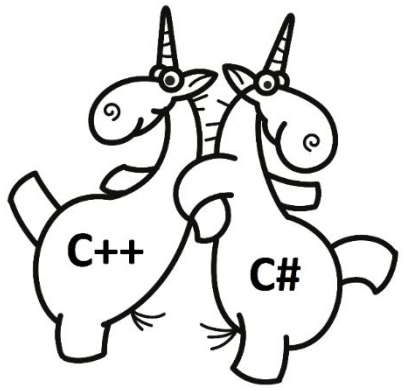
Контакты: support@viva64.com

2017

Статический анализатор кода PVS-Studio

- Выполняет анализ кода на языках: **C, C++, C++/CLI, C++/CX, C#**
- Поддерживаются проекты, разрабатываемые с помощью:
  - Windows: Visual C++, Clang, MinGW, Visual C#
  - Linux: Clang, GCC

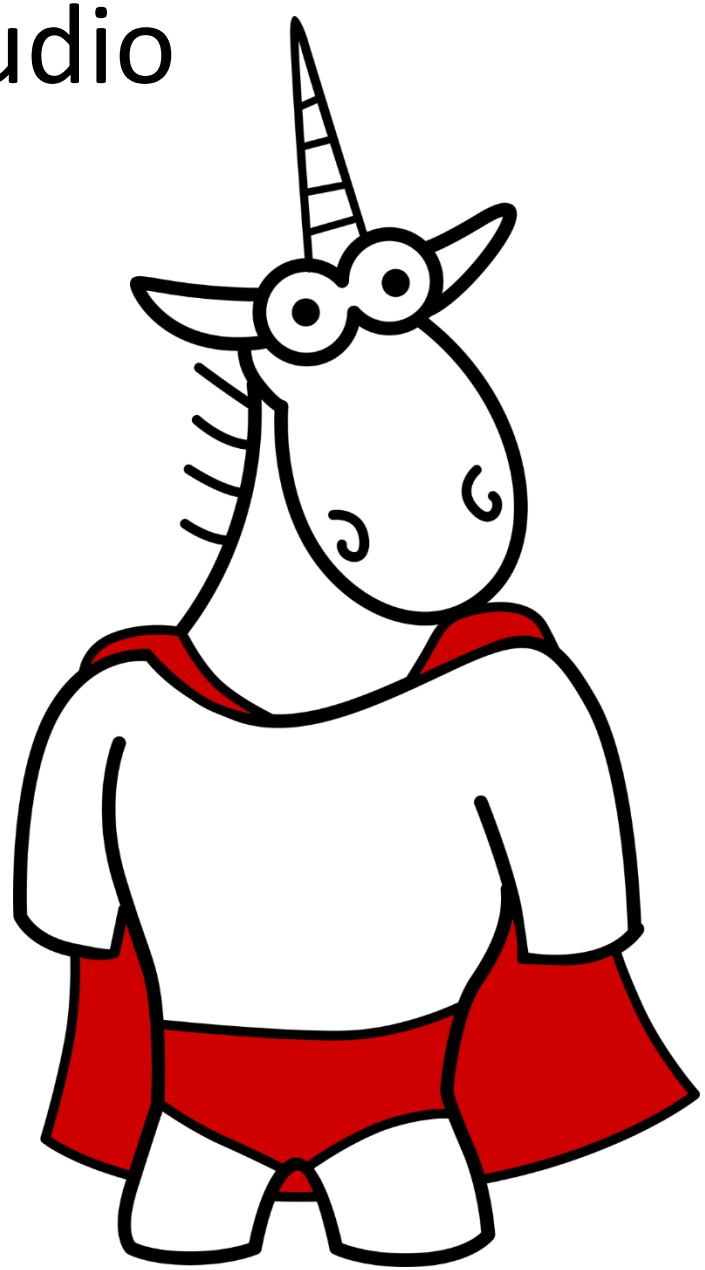
- Plugin для Visual Studio 2010-2015
- Интеграция с SonarQube, QtCreator, CLion, Eclipse CDT, Anjuta DevStudio и т.д.
- Утилита Standalone



На начало 2017 года в PVS-Studio реализовано

- C, C++ диагностик: **349**
- C# диагностик: **130**

- Подробная on-line документация на русском и английском языке
- PDF



Основные возможности

- **Быстрый старт** (мониторинг компиляции)
  - Windows утилита: CLMonitoring
  - Linux утилита: pvs-studio-analyzer
- Прямая интеграция анализатора в системы автоматизации сборки и утилита BlameNotifier (рассылка писем)
- Режим автоматического анализа изменённых файлов
- Отличная масштабируемость
- Работа с ложными срабатываниями

Почему нужны анализаторы кода?

Почему команда PVS-Studio выбрала C, C++ и C#?

Почему C и C++?

- Эффективные, но сложные языки, в которых легко допустить ошибку
- Причём, так дело обстоит десятилетиями и вряд ли изменится

- Давайте проверим с помощью PVS-Studio первую версию компилятора Cfront, вышедшую в свет в 1985.

  - “К тридцатилетию первого C++ компилятора: ищем ошибки в Cfront” http://www.viva64.com/ru/b/0355/

Ошибка в компиляторе Cfront (1985)

```
Pexpr expr::typ(Ptable tbl)
{
  ....
  Pclass cl;
  ....
  cl = (Pclass) nn->tp;
  cl->permanent=1;
  if (cl == 0) error('i',"%k %s'sT missing",CLASS,s);
```

В начале указатель разыменовывается. Далее идёт проверка, которая говорит нам, что указатель мог быть nullptr.

Прошло 30 лет

- Ничего не изменилось. Язык C++ всё так же сложен и опасен.
- Размер кодовой базы растёт, и все важнее использовать инструменты статического анализа

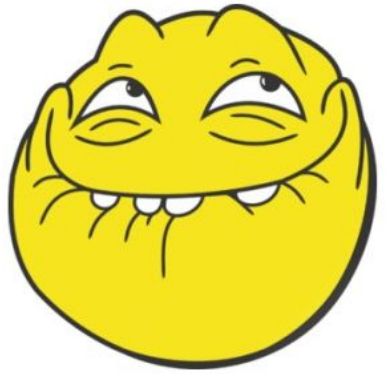
- Давайте проверим с помощью PVS-Studio код современного компилятора Clang

  - 2016 год. “Находим ошибки в коде проекта LLVM с помощью анализатора PVS-Studio” http://www.viva64.com/ru/b/0446/

Clang (ошибка найдена в октябре 2016)

```
bool PPCDarwinAsmPrinter::doFinalization(Module &M) {
  ....
  MachineModuleInfoMachO &MMIMacho =
      MMI->getObjFileInfo<MachineModuleInfoMachO>();

  if (MAI->doesSupportExceptionHandling() && MMI) {
```

Вначале указатель разыменовывается.
Далее идёт проверка, которая говорит нам, что указатель мог быть nullptr.



Почему C#?

- Быть может с C# ситуация лучше?
- Некоторые типы ошибок в C# невозможны
- Поэтому лучше, но не сильно
- На месте остаются опечатки, логические ошибки и т.д.
- Да и от того, что указатели назвали ссылками лучше не стало
  - Мы видим всё ту же ошибку с нулевой ссылкой
  - Давайте проверим, например, проект Microsoft PowerShell: http://www.viva64.com/ru/b/0447/

Такие ошибки актуальны и для C#

```
public CommandMetadata(CommandMetadata other)
{
  ....
  _parameters = new Dictionary<string, ParameterMetadata>(
     other.Parameters.Count, .....);

  if (other.Parameters != null)
```

Ошибка в проекте PowerShell: в начале используем ссылку, а затем проверяем.

Мы можем очень долго демонстрировать подобные примеры

- Анализатор PVS-Studio легко находит ошибки в известных проектах:
  - Linux kernel - http://www.viva64.com/ru/b/0460/
  - GCC - http://www.viva64.com/ru/b/0425/
  - MSBuild - http://www.viva64.com/ru/b/0424/
  - Qt - http://www.viva64.com/ru/b/0424/
  - И так далее - http://www.viva64.com/ru/inspections/
- Это говорит о востребованности статического анализа кода
- Давайте посмотрим какие ошибки умеет искать PVS-Studio

Диагностические возможности PVS-Studio

Ошибки при переносе кода на 64-битные платформы

Эту ошибку мы нашли с помощью PVS-Studio в проекте **TortoiseSVN**

```
DialogBoxParam(g_hmodThisDll,
                    MAKEINTRESOURCE(IDD_LOGIN),
                    g_hwndMain,
                    (DLGPROC)(LoginDialogProc),
                    (long)this);
```

V220 Suspicious sequence of types castings: memsize -> 32-bit integer -> memsize. The value being casted: 'this'. logindialog.cpp 105

Пояснение. Тип ***long*** в Win64 по прежнему 32-битный. В 64-битной программе объект может быть создан за пределами младших 4 Гигабайт памяти. В этом случае значение указателя будет испорчено. Неприятная ошибка, которая может проявляться очень редко после долгой работы программы. Правильно: (LPARAM)(this).

Адрес локальной переменной возвращается из функции по ссылке

Эту ошибку мы нашли с помощью PVS-Studio в проекте **LLVM**

```
SingleLinkedListIterator<T> &operator++(int) {
  SingleLinkedListIterator res = *this;
  ++*this;
  return res;
}
```

V558 Function returns the reference to temporary local object: res. LiveInterval.h 679

Арифметическое переполнение, потеря значимости

Эту ошибку мы нашли с помощью PVS-Studio в проекте **OpenXRay**

```
float CRenderTarget::im_noise_time;
....
param_noise_fps      = 25.f;
param_noise_scale    = 1.f;
im_noise_time        = 1/100;
....
```

V636 The '1 / 100' expression was implicitly cast from 'int' type to 'float' type. Consider utilizing an explicit type cast to avoid the loss of a fractional part. An example: double A = (double)(X) / Y;. gl_rendertarget.cpp 245

Выход за границу массива

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Notepad++**

```
int encodings[] = {1250, 1251, 1252, .... };

for (int i = 0; i <= sizeof(encodings)/sizeof(int); i++)
{
  int cmdID = em->getIndexFromEncoding(encodings[i]);
  ....
}
```

V557 Array overrun is possible. The value of 'i' index could reach 46. Notepad++ preferencedlg.cpp 984

Мёртвый код

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Unreal Engine 4**

```
int32 NumByteProperties = 0;
....
if (bIsByteProperty)
{
  NumByteProperties;
}
```

V607 Ownerless expression 'NumByteProperties'. codegenerator.cpp 633

Недостижимый код

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Linux Kernel**

```
if (val > 511)
  val = (val >> 1) | (1 << 9);
else if (val > 1022)
  val = (val >> 2) | (3 << 9);
```

V695 Range intersections are possible within conditional expressions. Example: if (A < 5) { ... } else if (A < 2) { ... }. Check lines: 439, 441. ad5933.c 441

Неинициализированные переменные

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Mono**

```
class ResXResourceWriter : IResourceWriter, IDisposable
{
  public static readonly string ResourceSchema = schema;
  ....
  static string schema = ....;
}
```

V3070 Uninitialized variable 'schema' is used when initializing the 'ResourceSchema' variable. ResXResourceWriter.cs 59

Пояснение. На момент инициализации *ResourceSchema* поле *schema* будет проинициализировано значением по умолчанию (*null* в данном случае)

Неиспользуемые переменные и аргументы

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Xenko**

```
public static Image New3D(int width, int height, int depth, ....)
{
    return new Image(
                CreateDescription(
                   TextureDimension.Texture3D,
                   width, width, depth,
                   mipMapCount, format, 1),
                dataPointer, 0, null, false);
}
```

V3065 Parameter 'height' is not utilized inside method's body. SiliconStudio.Xenko Image.cs 473

Некорректные операции сдвига

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Bitcoin**

```
static int64_t set_vch(....) {
  int64_t result = 0;
  ....
  return -(result & ~(0x80 << (8 * (vch.size() - 1))));
```

V629 Consider inspecting the '0x80 << (8 * (vch.size() - 1))' expression. Bit shifting of the 32-bit value with a subsequent expansion to the 64-bit type. script.h 169

Пояснение. Происходит переполнение при сдвиге 32-битного значения 0x80. Сейчас исправленный вариант кода выглядит так:

```
return -((int64_t)(result & ~(0x80ULL << (8 * (vch.size() - 1)))));
```

Неопределенное поведение

Эту ошибку мы нашли с помощью PVS-Studio в проекте
Network Security Services

```
waste[j & 0xf] = j++;
```

V567 Undefined behavior. The 'j' variable is modified while being used twice between sequence points. pk11slot.c 1926

Неправильная работа с типами

Эту ошибку мы нашли с помощью PVS-Studio в проекте **VirtualBox**

```
HRESULT EventClassID(BSTR bstrEventClassID);

static HRESULT VBoxCredentialProviderRegisterSENS(void)
{
  hr = pIEventSubscription->put_EventClassID(
         L"{d5978630-5b9f-11d1-8dd2-00aa004abd5e}");
```

V745 A 'wchar_t *' type string is incorrectly converted to 'BSTR' type string. Consider using 'SysAllocString' function. vboxcredentialprovider.cpp 231

Неправильное представление о работе функции/класса

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Unity3D**

```
private static readonly Regex UnsafeCharsWindows =
  new Regex("[^A-Za-z0-9\\_\\-\\.\\:\\,\\/\\@\\\\]");
```

V3057 Invalid regular expression patern in constructor. Inspect the first argument. AssetBundleDemo ExecuteInternalMono.cs 48

Пояснение. При попытке создания экземпляра класса Regex с данным паттерном мы получим исключение **System.ArgumentException** с сообщением:

```
parsing \"[^A-Za-z0-9\\_\\-\\.\\:\\,\\/\\@\\]\" -
Unrecognized escape sequence '\\_'.
```

Отсутствие виртуального деструктора

Все примеры длинные и их сложно поместить в презентацию. Поверьте, мы ищем такие проблемы.

А пока предлагаю заварить чашечку кофе.
Впереди у нас ещё долгий интере

Оформление кода не совпадает с логикой его работы

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Sony ATF**

```
public static QuatF Slerp(QuatF q1, QuatF q2, float t)
{
  double dot = q2.X * q1.X + q2.Y * q1.Y +
               q2.Z * q1.Z + q2.W * q1.W;
  if (dot < 0)
    q1.X = -q1.X; q1.Y = -q1.Y; q1.Z = -q1.Z; q1.W = -q1.W;
```

V3043 The code's operational logic does not correspond with its formatting. The statement is indented to the right, but it is always executed. It is possible that curly brackets are missing. Atf.Core.vs2010 QuatF.cs 282

Ошибки при работе с исключениями

Эту ошибку мы нашли с помощью PVS-Studio в проекте **OpenMW**

```
if (t1==t2)
  mOperands.push_back (t1);
else if (t1=='f' || t2=='f')
  mOperands.push_back ('f');
else
  std::logic_error ("failed to ....... ");
```

~~throw~~

V596 The object was created but it is not being used. The 'throw' keyword could be missing: throw logic_error(FOO); components exprparser.cpp 101

Переполнение буфера

Эту ошибку мы нашли с помощью PVS-Studio в проекте **FreeBSD**

```
#define   SID_VENDOR_SIZE    8
char    vendor[SID_VENDOR_SIZE];
....
strcpy(p->vendor,"Adaptec ");
```

V512 A call of the 'strcpy' function will lead to overflow of the buffer 'p->vendor'. aacraid_cam.c 571

Пояснение. Строка содержит 8 символов. Однако, следует учитывать, что функция *strcpy* добавит терминальный ноль. Он будет записан за пределами буфера.

Проблемы безопасности

Эту ошибку мы нашли с помощью PVS-Studio в проекте **PostgreSQL**

```
char *px_crypt_md5(....) {
  unsigned char final[MD5_SIZE];
  ....
  /* Don't leave anything around in vm they could use. */
  memset(final, 0, sizeof final);
}
```

Компилятор удаляет вызов функции *memset*: http://www.viva64.com/ru/w/V597/

V597 The compiler could delete the 'memset' function call, which is used to flush 'final' buffer. The RtlSecureZeroMemory() function should be used to erase the private data. pgcrypto crypt-md5.c 157

Путаница с приоритетом операций

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Linux Kernel**

```
static int nvme_pr_preempt(struct block_device *bdev,
  u64 old, u64 new, pr_type type, bool abort)
{
  u32 cdw10 = nvme_pr_type(type) << 8 | abort ? 2 : 1;
```

1

2

V502 Perhaps the '?:' operator works in a different way than it was expected. The '?:' operator has a lower priority than the '|' operator. core.c 1046

Разыменование нулевого указателя / нулевой ссылки

Эту ошибку мы нашли с помощью PVS-Studio в проекте **LibreOffice**

```
MenuBar *pMBar = pSysWin->GetMenuBar();

if ( pSysWin && pMBar )
{
  AddMenuBarIcon( pSysWin, true );
}
```

V595 The 'pSysWin' pointer was utilized before it was verified against nullptr. Check lines: 738, 739. updatecheckui.cxx 738

Ошибки синхронизации

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Unity3D**

```
internal void OnUnload()
{
  m_AssetBundle.Unload(false);
  if (unload != null)
    unload();
}
```

V3083 Unsafe invocation of event 'unload', NullReferenceException is possible. Consider assigning event to a local variable before invoking it. AssetBundleDemo AssetBundleManager.cs 47

Целочисленное деление на 0

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Inkscape**

```
} else if (type >= 3000 && type < 4000) {
    unsigned int chamferSubs = type-3000;
    double chamfer_stepsTime = 1.0/chamferSubs;
```

V609 Divide by zero. Denominator range [0..999]. lpe-fillet-chamfer.cpp 607

Опечатки и Copy-Paste

- Анализатор PVS-Studio эффективно выявляет опечатки и последствия неудачного Copy-Paste
- В анализаторе реализовано много диагностик для выявления ошибок этого рода
- Остановимся на них чуть подробнее и рассмотрим несколько примеров ошибок этого типа

- Дополнительно рекомендуем для чтения интересную статью “Эффект последней строки” - http://www.viva64.com/ru/b/0260/

Опечатки и Copy-Paste (пример N1)

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Clang**

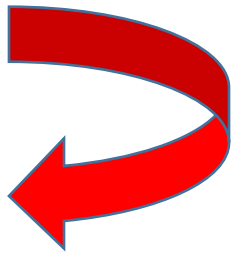
```
if ((OpcodeLHS == BO_EQ ||
     OpcodeLHS == BO_LE ||
     OpcodeLHS == BO_LE)
    &&
    (OpcodeRHS == BO_EQ ||
     OpcodeRHS == BO_GT ||
     OpcodeRHS == BO_GE))
```

V501 There are identical sub-expressions 'OpcodeLHS == BO_LE' to the left and to the right of the '||' operator. RedundantExpressionCheck.cpp 174

Опечатки и Copy-Paste (пример N2)

Эту ошибку мы нашли с помощью PVS-Studio в проекте **GCC**

```
return (!strcmp (a->v.val_vms_delta.lbl1,
                 b->v.val_vms_delta.lbl1)
        && !strcmp (a->v.val_vms_delta.lbl1,
                    b->v.val_vms_delta.lbl1));
```



V501 There are identical sub-expressions '!strcmp(a->v.val_vms_delta.lbl1, b->v.val_vms_delta.lbl1)' to the left and to the right of the '&&' operator. dwarf2out.c 1428

Опечатки и Copy-Paste (пример N3)

Эту ошибку мы нашли с помощью PVS-Studio в проекте MySQL

```
static int rr_cmp(uchar *a,uchar *b)
{
  if (a[0] != b[0])
    return (int) a[0] - (int) b[0];
  if (a[1] != b[1])
    return (int) a[1] - (int) b[1];
  if (a[2] != b[2])
    return (int) a[2] - (int) b[2];
  if (a[3] != b[3])
    return (int) a[3] - (int) b[3];
  if (a[4] != b[4])
    return (int) a[4] - (int) b[4];
  if (a[5] != b[5])
    return (int) a[1] - (int) b[5];
  if (a[6] != b[6])
    return (int) a[6] - (int) b[6];
  return (int) a[7] - (int) b[7];
}
```

5

V525 The code containing the collection of similar blocks. Check items '0', '1', '2', '3', '4', '1', '6' in lines 680, 682, 684, 689, 691, 693, 695. sql records.cc 680

Опечатки и Copy-Paste (пример N4)

Эту ошибку мы нашли с помощью PVS-Studio в проекте PowerShell

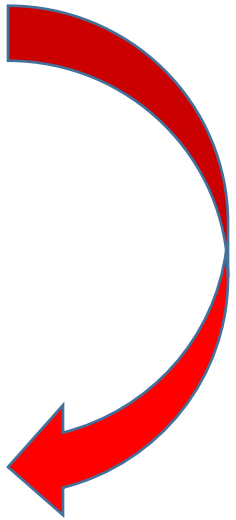
```
internal Version BaseMinimumVersion { get; set; }
internal Version BaseMaximumVersion { get; set; }

protected override void ProcessRecord()
{
  if (BaseMaximumVersion != null &&
      BaseMaximumVersion != null &&
      BaseMaximumVersion < BaseMinimumVersion)
```

V3001 There are identical sub-expressions 'BaseMaximumVersion != null' to the left and to the right of the '&&' operator. System.Management.Automation ImportModuleCommand.cs 1663

Опечатки и Copy-Paste (пример N5)

Эту ошибку мы нашли с помощью PVS-Studio в проекте **Roslyn**

```
if (i % 2 == 0)
{
  thread1.Start();
  thread2.Start();
}
else
{
  thread1.Start();
  thread2.Start();
}
```



V3004 The 'then' statement is equivalent to the 'else' statement. GetSemanticInfoTests.cs 2269

Опечатки и Copy-Paste (пример N6)

Эту ошибку мы нашли с помощью PVS-Studio в проекте **MonoDevelop**

```
xmlWriter.WriteAttributeString("name",
  suite.TestType == "Assembly" ?
    result.Test.FullName : result.Test.FullName);
```

V3012 The '?:' operator, regardless of its conditional expression, always returns one and the same value: result.Test.FullName. GuiUnit_NET_4_5 NUnit2XmlOutputWriter.cs 207

Мы показали вам малую часть того, что может находить анализатор PVS-Studio

- Подробная таблица диагностических возможностей: http://www.viva64.com/ru/w/

- Там же вы найдете подробное описание всех диагностик

Main PVS-Studio diagnostic abilities	C, C++ diagnostics	C# diagnostics
64-bit issues	V101-V128, V201-V207, V220, V221, V301-V303	-
Check that addresses to stack memory does not leave the function	V506, V507, V558, V758	-
Arithmetic over/underflow	V636, V658	V3040, V3041
Array index out of bounds	V557, V582, V643	V3106
Check for double-free	V586, V749	-
Dead code	V606, V607	-
Microoptimization	V801-V815	-
Unreachable code	V551, V695, V734	-
Uninitialized variables	V573, V614, V679, V730, V737	V3070
Unused variables	V603, V751, V763	V3061, V3065, V3077
Illegal bitwise/shift operations	V610, V629, V673, V684	-
Undefined/unspecified behavior	V567, V610, V611, V681, V704, V708, V726, V736	-
Incorrect handling of the types (HRESULT, BSTR, BOOL, VARIANT_BOOL)	V543, V544, V545, V716, V721, V724, V745, V750, V676, V767	-
Improper understanding of function/class operation logic	V518, V530, V540, V541, V554, V575, V597, V598, V618, V630, V632, V663, V668, V698, V701, V702, V717, V718, V720, V723, V725, V727, V738, V742, V743, V748, V762, V764	V3010, V3057, V3068, V3072, V3073, V3074, V3082, V3084, V3094, V3096, V3097, V3102, V3103, V3104, V3108
Misprints	V501, V503, V504, V508, V511, V516, V519, V520, V521, V525, V527, V528, V529, V532, V533, V534, V535, V536, V537, V539, V546, V549, V552, V556, V559, V560, V561, V564, V568, V570, V571, V575, V577, V578, V584, V587, V588, V589, V590, V592, V600, V602, V604, V606, V607, V616, V617, V620, V621, V622, V625, V626, V627, V633, V637, V638, V639, V644, V646, V650, V651, V653, V654, V655, V660, V661, V662, V666, V669, V671, V672, V678, V682, V683, V693, V715, V722, V735, V747, V754, V756, V765, V767	V3001, V3003, V3005, V3007, V3008, V3009, V3011, V3012, V3014, V3015, V3016, V3020, V3028, V3029, V3034, V3035, V3036, V3037, V3038, V3050, V3055, V3056, V3057, V3062, V3063, V3066, V3081, V3086, V3091, V3092, V3107, V3109
Missing Virtual destructor	V599, V689	-
Coding style not matching the operation logic of the source code	V563, V612, V628, V640, V646, V705	V3018, V3033, V3043, V3067, V3069
Copy-Paste	V501, V517, V519, V523, V524, V571, V581, V649, V656, V691, V760, V766	V3001, V3003, V3004, V3008, V3012, V3013, V3021, V3030, V3058
Incorrect usage of exceptions	V509, V565, V596, V667, V740, V741, V746, V759	V3006, V3052, V3100
Buffer overrun	V512, V514, V594, V635, V641, V645, V752, V755	-
Security issues	V505, V510, V511, V512, V518, V531, V541, V547, V559, V560, V569, V570, V575, V576, V579, V583, V597, V598, V618, V623, V642, V645, V675, V676, V724, V727, V729, V733, V743, V745, V750	V3022, V3023, V3025, V3027, V3053, V3063
Operation priority	V502, V562, V593, V634, V648	-
Null pointer pointer/null reference dereference	V522, V595, V664, V757	V3019, V3042, V3080, V3095, V3105
Unchecked parameter dereference	V595, V664	V3095

Демонстрация возможностей PVS-Studio

- Для демонстрации возможностей анализатора мы проверяем открытые проекты. На начало 2017 года нами проверено **280** проектов.

- Побочный результат: в этих проектах нашей командой было найдено **10700** ошибок

- Это именно 10700 ошибок, а не количество сообщений, выданных анализатором

Демонстрация возможностей PVS-Studio

- Благодаря нашей команде и анализатору PVS-Studio, в открытых проектах исправили более 10000 ошибок
- Вы можете увидеть все эти ошибки здесь: http://www.viva64.com/ru/examples/
- База ошибок постоянно пополнятся и её можно использовать при написании статей о качестве кода и составлении стандартов кодирования

Демонстрация возможностей PVS-Studio

- В среднем, в одном открытом проекте мы нашли 10700 / 280 = 38 ошибок
- 38 ошибок на проект - это мало
- Поэтому важно подчеркнуть, что это - **побочный эффект**
- У нас нет цели найти как можно больше ошибок. Часто мы останавливаемся, когда нашли достаточное количество дефектов в проекте для написания статьи.

Демонстрация возможностей PVS-Studio

- Мы добились колоссальных результатов в устранении ошибок в мире open-source проектов, не ставя такую цель

- В этом нам помогли:
  - мощные диагностические возможности PVS-Studio
  - возможность быстрого анализа даже незнакомых проектов

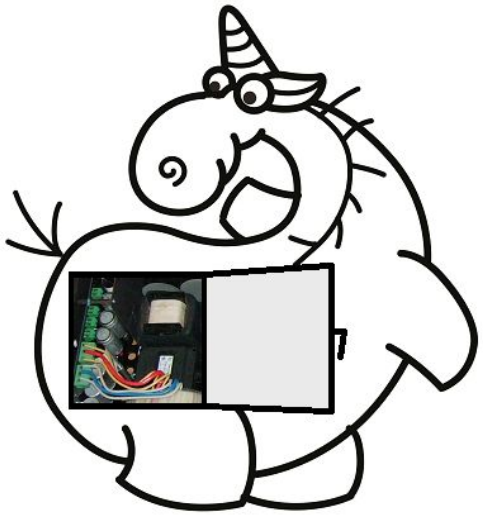
Правильный сценарий использования

- Конечно, интересно и полезно запустить анализатор PVS-Studio и найти ошибку, которую до этого безуспешно искали 50 часов
  - http://www.viva64.com/ru/b/0221/
- Хорошо проверять проекты и описывать найденные ошибки, как делаем это мы в рекламных целях
  - http://www.viva64.com/ru/inspections/
- **Но следует помнить, что разовые проверки - это неправильный способ использования анализатора кода!**

Правильный сценарий использования

- Статический анализатор приносит пользу, когда он используется регулярно
- Два основных варианта:
  - Автоматический анализ изменённого кода
  - Ночные проверки
- Подробнее эти режимы рассмотрены в документации

Кратко о внутреннем устройстве PVS-Studio



Используемые технологии

- На примерах было продемонстрировано, что PVS-Studio эффективно выявляет разнообразные типы ошибок

- Кратко перечислим технологии, которые положены в основу анализатора

- Подробнее эта тема раскрыта в статье “Как PVS-Studio ищет ошибки: методики и технологии” http://www.viva64.com/ru/b/0466/

Используемые технологии

- Сопоставление с шаблоном (pattern-based analysis) на основе абстрактного синтаксического дерева применяется для поиска мест в исходном коде, которые похожи на известные шаблоны кода с ошибкой.

Пример. Иногда не там прибавляют 1 при использовании функции *strlen*:

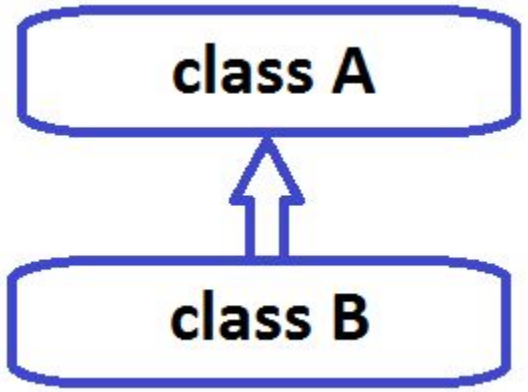
```
realloc(name, strlen(name+1))
```

На самом деле следовало написать:

```
realloc(name, strlen(name)+1)
```

Используемые технологии

- Вывод типов (type inference) на основе семантической модели программы позволяет анализатору иметь полную информацию о всех переменных и выражениях, встречающихся в коде.



Самый просто пример: чтобы знать, что функция *printf* используется неправильно, нужно знать типы фактических аргументов.

Используемые технологии

- Символьное выполнение (symbolic execution) позволяет вычислять значения переменных, которые могут приводить к ошибкам, производить проверку диапазонов (range checking) значений
- Анализ потока данных (data-flow analysis) используется для вычисления ограничений, накладываемых на значения переменных при обработке различных конструкций языка. Например, какие значения может принимать переменная внутри блоков if/else.

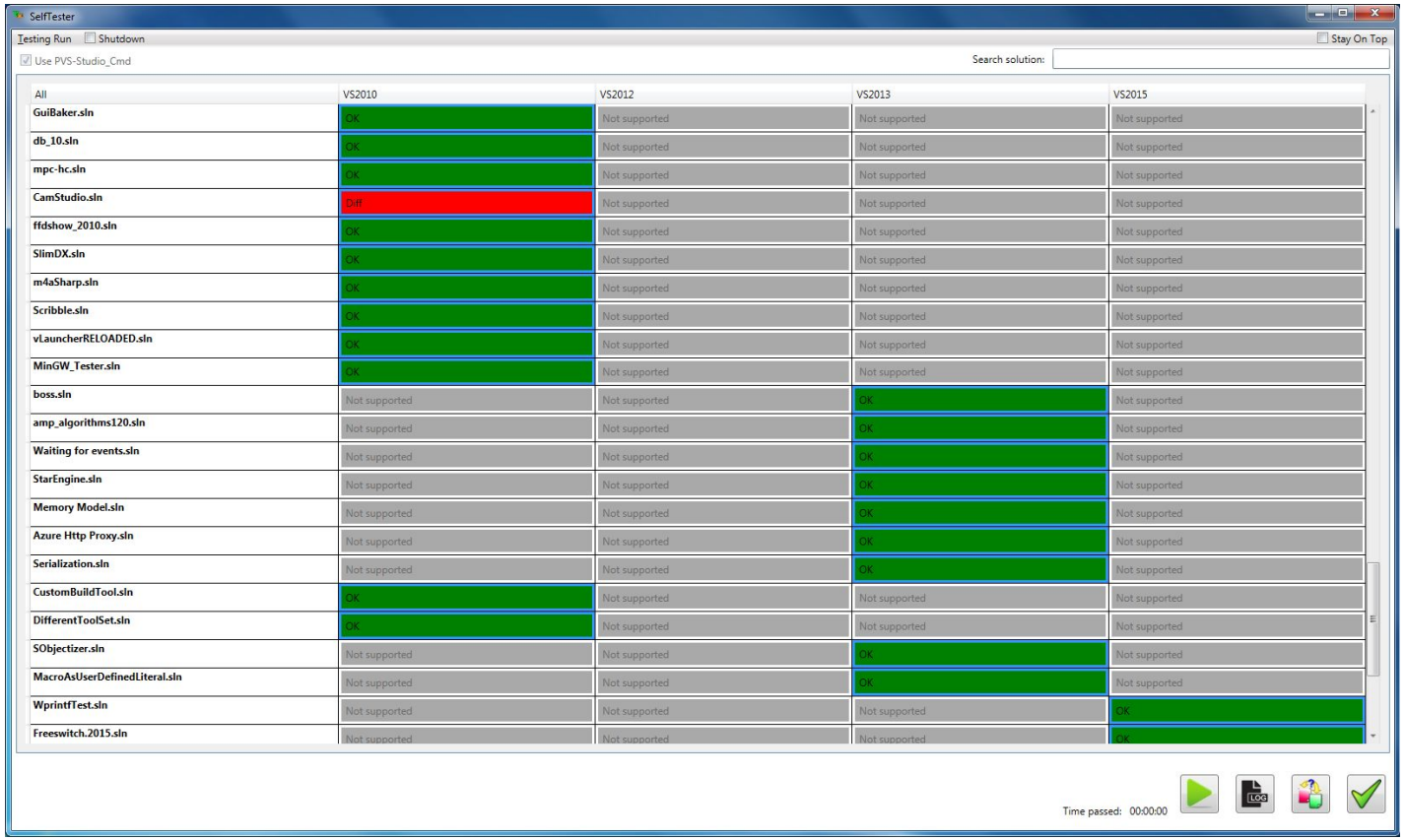
Используемые технологии

- Аннотированние методов (method annotations) предоставляет больше информации об используемых методах, чем может быть получено путём анализа только их сигнатуры.

- C/C++. На данный момент проаннотировано **6570** функций (стандартные библиотеки C и C++, POSIX, MFC, Qt, ZLib и так далее).
- C#. На данный момент проаннотировано **920** функций.

Используемые технологии

- Для разработки эффективных диагностик наша команда использует большой набор регрессионных тестов
- Написан специальный инструментарий для работы с тестовой базой открытых проектов



Используемые технологии

- Тестовая база:
  - **C++** Windows (Visual C++): **120** проектов
  - **C++** Linux (GCC): ещё **34** проекта
  - **C#** Windows: **54** проекта

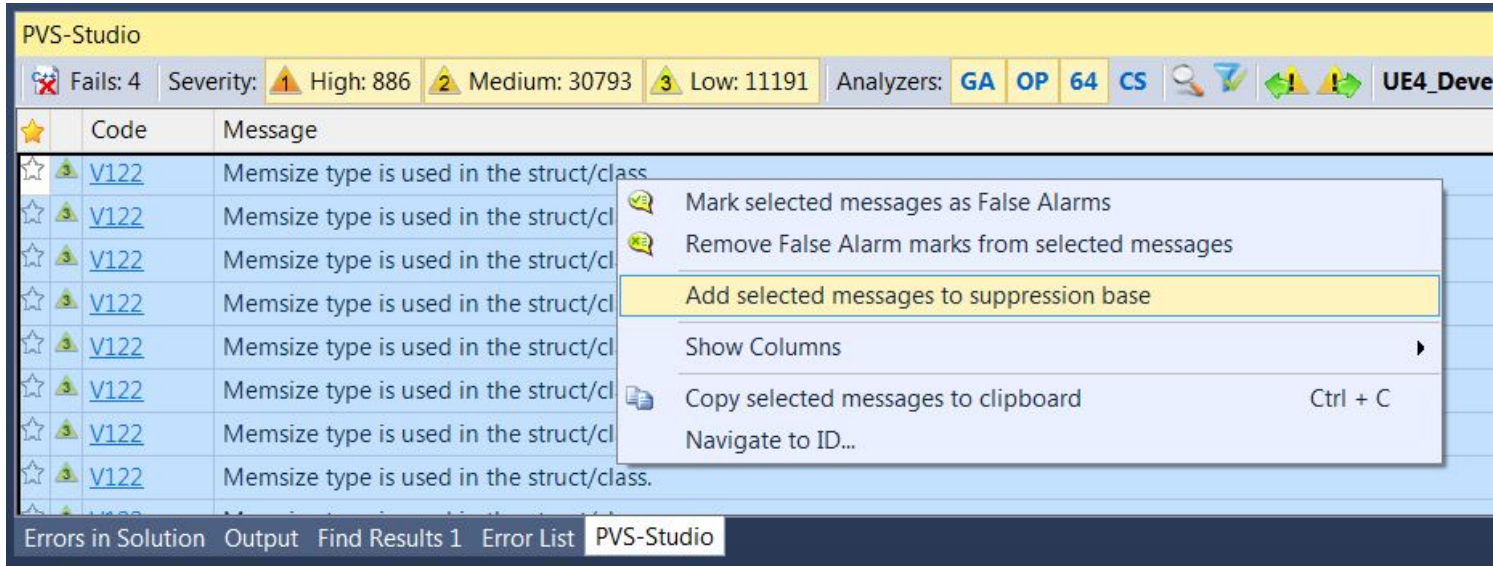
- Всего мы используем 7 методик тестирования нашего проекта
  - См. раздел “Тестирование PVS-Studio” http://www.viva64.com/ru/b/0466/

Использование PVS-Studio

- Думаю, вы уже устали, поэтому минутка юмора
- Кратко суть статического анализа кода сводится к следующему:

Использование PVS-Studio: внедрение

- Бывает непросто начать использовать статический анализ в большом проекте
- Непонятно, что делать с сообщениями в старом коде…
- Мы предлагаем решение: база разметки
- Подробнее: http://www.viva64.com/ru/b/0364/



Использование PVS-Studio: подавление ложных срабатываний

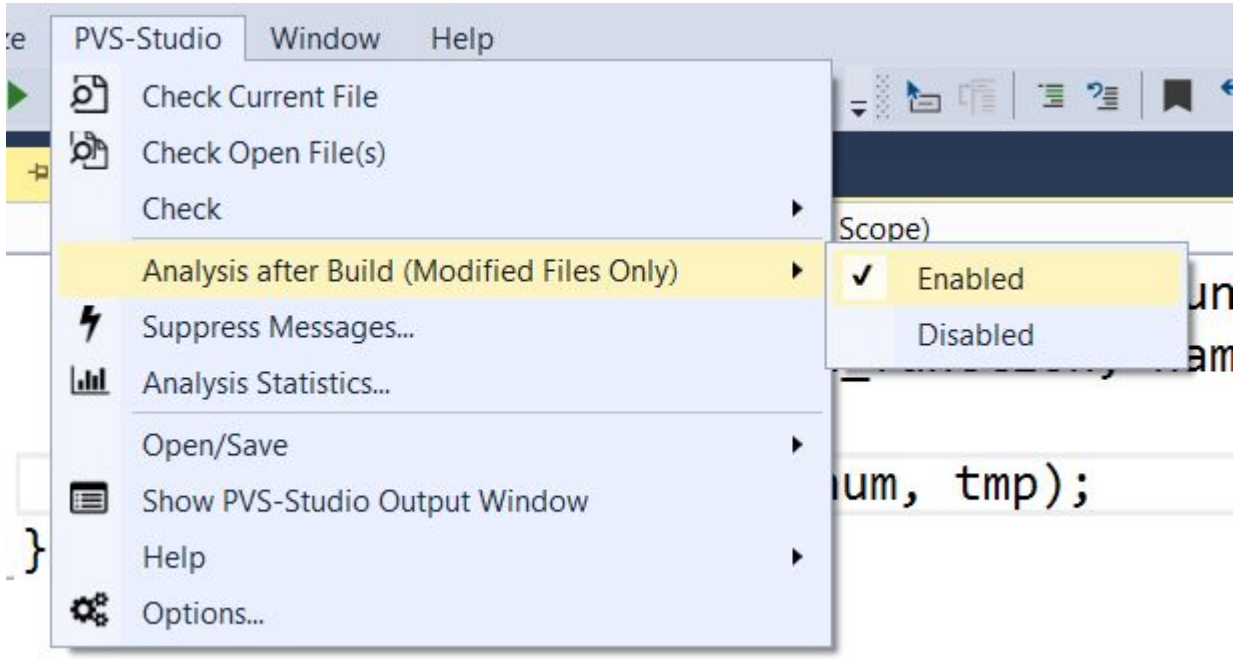
- Различные способы подавить ложные срабатывания в конкретных строках кода
- Подавление ложных срабатываний в макросах
- Подавление ложных предупреждений с помощью файлов конфигурации диагностик pvsconfig
- Подробнее: http://www.viva64.com/ru/m/0017/

Использование PVS-Studio: исключение из анализа

- Возможность исключить из анализа файлы по имени, папке или маске
- Интерактивная фильтрация результатов анализа (лога) в окне PVS-Studio:
  - по коду диагностики
  - по имени файла
  - по включению слова в текст диагностики

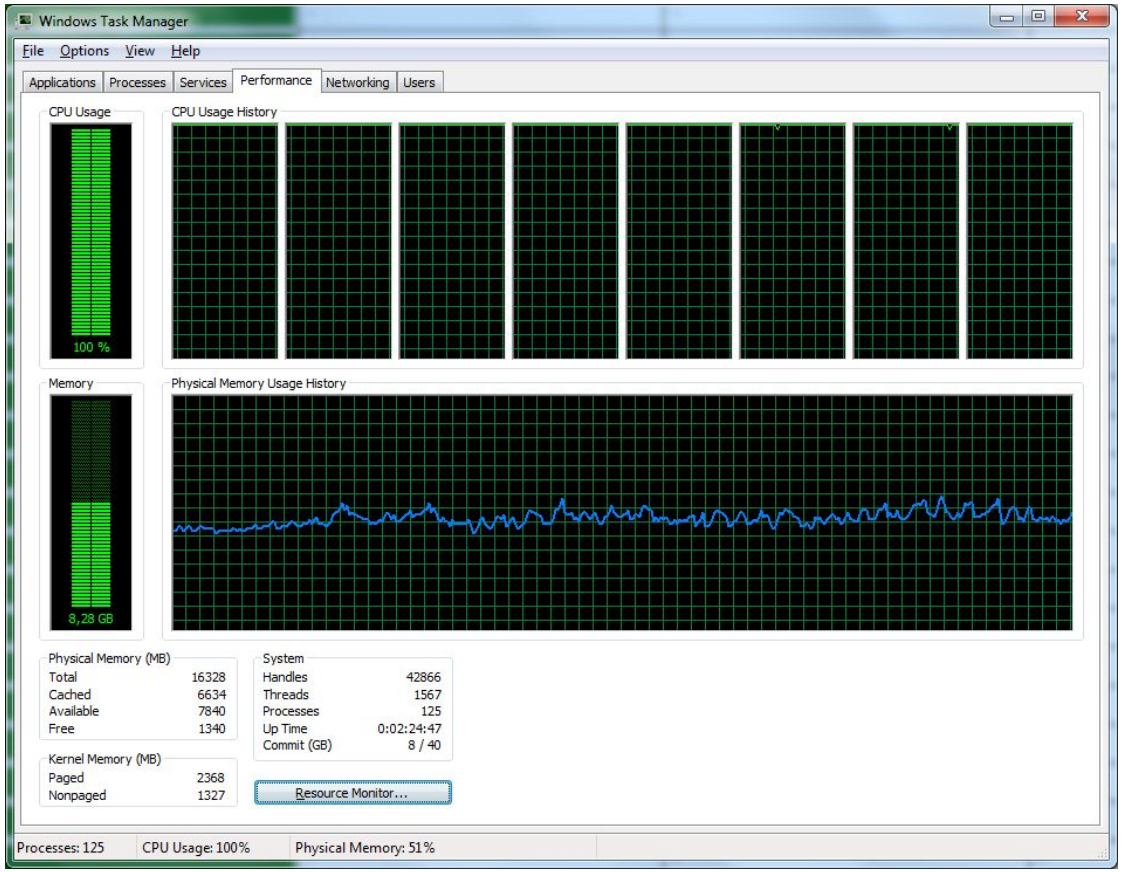
Использование PVS-Studio: автоматический анализ файлов после их перекомпиляции

- Наиболее эффективно исправить ошибку сразу после того, как она появилась в коде



Использование PVS-Studio: масштабируемость

- Поддержка многоядерных и многопроцессорных систем с настройкой количества используемых ядер
- Поддержка IncrediBuild



Использование PVS-Studio: непрерывная интеграция

- Запуск из командной строки для проверки всего решения: позволяет интегрировать PVS-Studio в ночные сборки, чтобы утром у всех был свежий лог
- Сохранение и загрузка результатов анализа: можно ночью проверить код, сохранить результаты, а утром загрузить их и смотреть
- Утилита BlameNotifier: инструмент позволяет рассылать письма разработчикам об ошибках, которые PVS-Studio нашел во время ночного прогона
- Использование относительных путей в файлах отчета

Использование PVS-Studio: прочее

- Удобная online-справка по всем диагностикам, которая доступна и из программы, и на сайте, а также документация в .pdf одним файлом
- Интерактивная фильтрация результатов анализа (лога) в окне PVS-Studio
- Статистика ошибок в Excel
- Автоматическая проверка на наличие новых версий PVS-Studio

Использование PVS-Studio: Linux

- С PVS-Studio легко работать в Linux
- Но, чтобы не гадать с настройками и ключами запуска, просим познакомиться с инструкцией
- Как запустить PVS-Studio в Linux: http://www.viva64.com/ru/m/0036/

- Я знаю, что мы все не любим читать инструкции. Но поверьте, этот тот случай, когда всё просто, кратко и экономит ваше время!

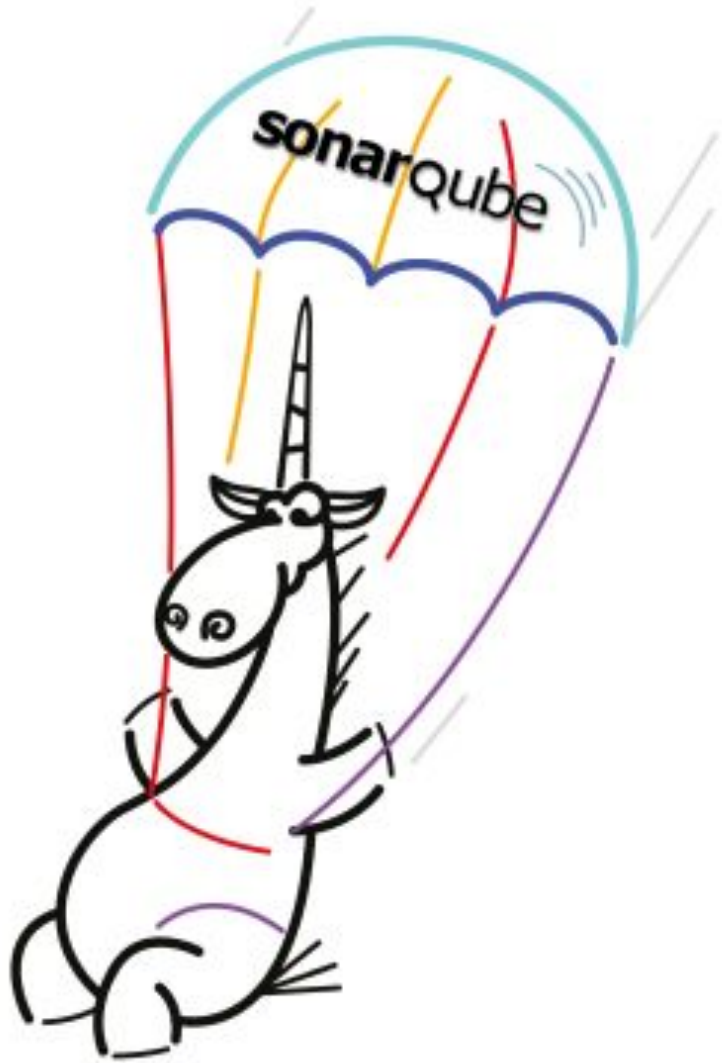
Использование PVS-Studio: быстрый старт

- Отдельного внимания заслуживает возможность быстро попробовать PVS-Studio на любом проекте
- Для этого можно отследить запуски компилятора и собрать всю необходимую для анализа информацию
- Windows:
  - Утилита Standalone
  - Инструкция: http://www.viva64.com/ru/m/0033/
- Linux
  - Утилита pvs-studio-analyzer
  - Инструкция: см. «Быстрый старт» в документе http://www.viva64.com/ru/m/0036/

Использование PVS-Studio: SonarQube

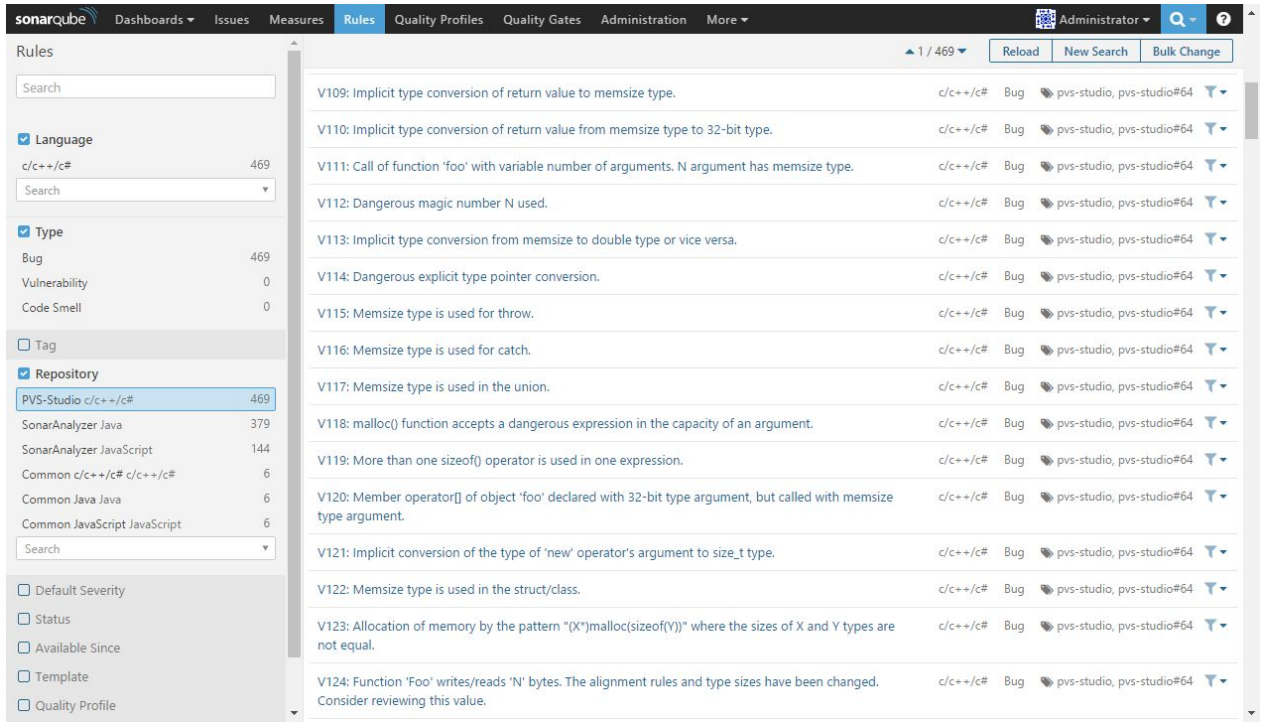
- Для импорта результатов анализа в SonarQube мы разработали плагин sonar-pvs-studio-plugin

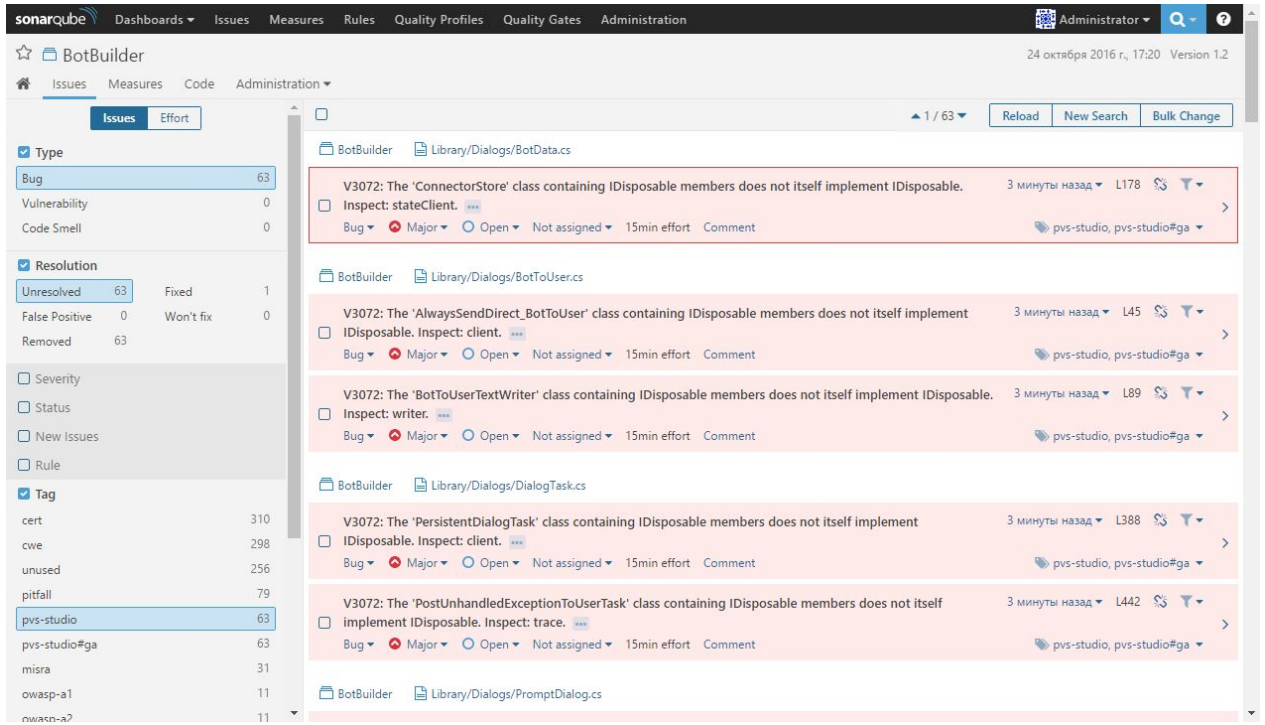
- Использование плагина позволяет добавлять сообщения, найденные анализатором PVS-Studio, в базу сообщений сервера SonarQube



Использование PVS-Studio: SonarQube

- Подробности изложены в статье «Контролируем качество кода с помощью платформы SonarQube» http://www.viva64.com/ru/b/0452/





Скачать и попробовать PVS-Studio

Скачать и попробовать PVS-Studio

- Можно скачать и попробовать демонстрационную версию
  - Windows: http://www.viva64.com/ru/pvs-studio-download/
  - Linux: http://www.viva64.com/ru/pvs-studio-download-linux/
- Про ограничения демонстрационной версии: http://www.viva64.com/ru/m/0009/
- Вы можете написать нам и получить на время полную версию: support@viva64.com

Клиенты

Клиенты:


HOTHEAD
MTC
IBM
VISPIRON
TERADYNE
Viber
KS
ePLAN
mpdv
The MES Experts!
NATIONAL INSTRUMENTS
MERGE
CSIR
innovmetric
Volkswagen
nanometrics
BRUKER
intel
EMERSON
Process Management
GEDIS
A Rohde&Schwarz Company
WARGAMING.NET
LET'S BATTLE
ALIGN
TECHNOLOGY
microOLAP
rocksteady
GlobalView
NG
Branch Technology
neople
CRÉDIT AGRICOLE
Adobe
INEOQUEST
DR
Systems
NEXON GT
exness
PROGNOZ
hp
EPIC
GAMES
NCSOFT
DAYBREAK
SIGNON
MEMBER OF TÜV SÜD GROUP
invisalign
Artec 3D
Hex-Rays
DALET
NEC
TEOCO
THE EMPLOYEE OWNED COMPANY
F-Secure
VEEAM
BAE SYSTEMS
INSPIRED WORK
KASPERSKY lab
SUPERCELL
RSI
create, evaluate, operate
a2ia
Cerner
WAVES
NAT GAMES
ESPRIT
Centrify
Prescient
Technologies
FUJITSU
ZeniMax
MEDIA INC.
CADMATIC
LOGOS

Клиенты:


JEPPESEN
GENTEX CORPORATION
SPORTVISION
INTER FORUM
UBM - LOHN
rovi
Bern University of Applied Sciences
MetaQuotes SOFTWARE
OpenConnect
VECTORWORKS
StruSoft Structural Design Software
COPADATA
Forsk
LeuTek USU
SIGMA3
CSoft development
MAXON
GT Gamma Technologies
SEK
TECH SOFT 3D
CINEGY
ZECURION
Gaijin entertainment
NVIDIA
HAUFE.Gruppe
informatica
Nobel Biocare
KNOWLEDGE BASE
NEXUS SOFTWARE
Microsoft
NEXSTREAMING
Schneider Electric
infotecs
FotoNation a TESSERA company
equipoise software
imcGAMES
GE Healthcare
eurocom
c-core
MAIN CONCEPT
MORNINGSTAR
HMGCC
LENEL
BOSCH
CAE
DB Mobility Networks Logistics
OctoNus
Johnson&Johnson
AVENTURA Situational Awareness
cockos INCORPORATED
BUCHI
id
Tradeweb
COHERENT SOLUTIONS
Symantec
Fairchild Imaging
McKESSON
APTEAN
EXFO
CR2 CHANNEL BANKING SOFTWARE
1С ФИРМА «1С»
PHILIPS
pitney bowes

Купить PVS-Studio

Типы лицензий

Team License	Enterprise License
Подходит небольшим отделам, обычно это первый опыт использования анализаторов	Подходит сразу для нескольких отделов в компании
Только Windows-версия	Версии для Linux и Windows
	Поддержка систем непрерывной интеграции (continuous integration)
	Интеграция с SonarQube
	Инструмент BlameNotifier
	Возможность предлагать для реализации собственные диагностики.
Лицензию возможно купить только на 1 год	Лицензию возможно купить на 1, 2 или 3 года
Ответ на письма в течение 48 часов	Ответ на письма в течение 24 часов
Обычно используется в командах до 9 человек	В основном используется при размере команд более 10 человек

Индивидуальные лицензии

- Мы позиционируем свой продукт как B2B решение и у нас нет индивидуальных лицензий
- Почему так получилось: http://www.viva64.com/ru/b/0320/

- Индивидуальные разработчики могут воспользоваться вариантом **бесплатного лицензирования**
- Как использовать PVS-Studio бесплатно: http://www.viva64.com/ru/b/0457/

Купить PVS-Studio

- Для заказа лицензии и получения информации о ценах, пожалуйста, напишите нам: support@viva64.com

Помимо приобретения лицензии на статический анализатор кода PVS-Studio возможны другие варианты сотрудничества

Сотрудничество: аудит

- Выполнение аудита кода и правка ошибок
- Примеры сотрудничества подобного типа:
  - Как команда PVS-Studio улучшила код Unreal Engine: http://www.viva64.com/ru/b/0330/
  - Как перенести проект размером в 9 млн строк кода на 64-битную платформу: http://www.viva64.com/ru/b/0342/

- Мы можем на регулярной основе контролировать качество кода и вносить в него правки
  - Имеем опыт работ в этом направлении, но эта информация попадает под NDA

Сотрудничество

- На базе нашего анализатора мы можем разработать на заказ специализированное решение
- Мы также готовы обсуждать сотрудничество, кто готов за процент с продаж привести к нам клиентов
- По этим и другим вопросам: support@viva64.com

Презентация подходит к концу
Спасибо всем, кто добрался сюда

Полезные ссылки

- Расскажем о некоторых интересных и полезных материалах, которые можно найти на сайте компании

- Электронная книга «Главный вопрос программирования, рефакторинга и всего такого» - http://www.viva64.com/ru/b/0391/

- Электронная книга «Разработки 64-битных приложений на языке Си/Си++» - http://www.viva64.com/ru/l/full/

Полезные ссылки

- Как 10 лет назад начинался проект PVS-Studio: http://www.viva64.com/ru/b/0465/
- Контролируем качество кода с помощью платформы SonarQube: http://www.viva64.com/ru/b/0452/
- Руководство по разработке модулей расширений на C# для Visual Studio 2005-2012 и Atmel Studio: http://www.viva64.com/ru/a/0082/

Всем спасибо! До свидания

- Написать письмо: support@viva64.com
- Подписаться на твиттер: @Code_Analysis

- Скачать PVS-Studio для Windows: http://www.viva64.com/ru/pvs-studio/

- Скачать PVS-Studio для Linux: http://www.viva64.com/ru/pvs-studio-download-linux/