

# Чем статический анализ кода полезен C# программисту



Константин Волоховский

Project Lead

# Константин Волоховский

- Руководжу одним из направлений разработки в PVS-Studio
- Раньше занимался веб-разработкой на .NET
- А ещё я пишу статьи
- И выступаю на конференциях

@kvolokhovskii



Релевантность

# Рост плотности ошибок

Размер проекта (число строк)	Типичная плотность ошибок
Менее 2К	0 – 25 ошибок на 1000 строк кода
2К-16К	0 – 40 ошибок на 1000 строк кода
16К-64К	0.5 – 50 ошибок на 1000 строк кода
64К-512К	2 – 70 ошибок на 1000 строк кода
512К и более	4 – 100 ошибок на 1000 строк кода

```

private IAssetPreviewService (bool isVisible)
{
    get(isVisible)
    { PreviewService.OnShowPreview();
    else (previewService != null)
        PreviewService.OnHidePreview();
    }
    previewService = ServiceProvider.TryGet<IAssetPreviewService>();
    if (previewService == null)
        return null;

    previewService.PreviewAssetUpdated += PreviewAssetUpdated;
    return previewService;
}
}

```

## Stride

Исправляя что-то в одном месте, можно сломать в другом

```
private IAssetPreviewService PreviewService
{
    get
    {
        if (previewService != null)
            return previewService;

        previewService = ServiceProvider.TryGet<IAssetPreviewService>();
        if (previewService == null)
            return null;

        previewService.PreviewAssetUpdated += PreviewAssetUpdated;
        return previewService;
    }
}
```

## Stride

Исправляя что-то в одном месте, можно сломать в другом

PVS-Studio: V3080. Possible null dereference. Consider inspecting 'PreviewService'.  
PreviewViewModel.cs 109

# Борьба за качество кода

Стандарты  
разработки



Автоматическое  
тестирование



Code review



**Статический  
анализ**



Что это такое



# Автоматизированное код-ревью

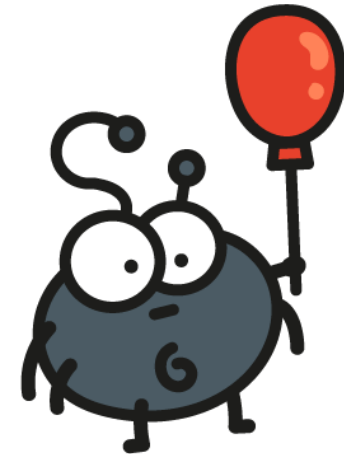
Неинтересно  
искать



Сложно найти



Не выглядят  
как ошибки



```
public static SqlDecimal operator /(SqlDecimal x, SqlDecimal y)
{
    ....
    bScaled = x._bScale;
    bPrecD = x._bPrec;
    ResScale = Math.Max(x._bScale + y._bPrec + 1,
                        s_cNumeDivScaleMin);
    ResInteger = x._bPrec - x._bScale + y._bScale;
    ResPrec = ResScale + x._bPrec + y._bPrec + 1;
    MinScale = Math.Min(ResScale, s_cNumeDivScaleMin);

    ResInteger = Math.Min(ResInteger, s_NUMERIC_MAX_PRECISION);
    ResPrec = ResInteger + ResScale;
    ....
}
```

## .NET 8

Ошибки, которые  
тяжело заметить

```
public static SqlDecimal operator /(SqlDecimal x, SqlDecimal y)
{
    ....
    bScaled = x._bScale;
    bPrecD = x._bPrec;
    ResScale = Math.Max(x._bScale + y._bPrec + 1,
                       s_cNumeDivScaleMin);
    ResInteger = x._bPrec - x._bScale + y._bScale;
    ResPrec = ResScale + x._bPrec + y._bPrec + 1;
    MinScale = Math.Min(ResScale, s_cNumeDivScaleMin);

    ResInteger = Math.Min(ResInteger, s_NUMERIC_MAX_PRECISION);
    ResPrec = ResInteger + ResScale;
    ....
}
```

## .NET 8

Ошибки, которые  
тяжело заметить

PVS-Studio: V3008 The 'ResPrec' variable is assigned values twice successively. Perhaps this is a mistake. Check lines: 1689, 1685. SQLDecimal.cs 1689

```
public override bool Equals(object obj)
{
    if (!(obj is ActorMaterializerSettings s)) return false;
    return
        s.InitialInputBufferSize == InitialInputBufferSize &&
        s.MaxInputBufferSize == MaxInputBufferSize &&
        s.Dispatcher == Dispatcher &&
        s.SupervisionDecider == SupervisionDecider &&
        s.SubscriptionTimeoutSettings == SubscriptionTimeoutSettings &&
        s.IsDebugLogging == IsDebugLogging &&
        s.OutputBurstLimit == OutputBurstLimit &&
        s.SyncProcessingLimit == SyncProcessingLimit &&
        s.IsFuzzingMode == IsFuzzingMode &&
        s.IsAutoFusing == IsAutoFusing &&
        s.SubscriptionTimeoutSettings == SubscriptionTimeoutSettings &&
        s.StreamRefSettings == StreamRefSettings;
}
```

**Akka.net**

Ошибки, которые  
скучно искать

```
public override bool Equals(object obj)
{
    if (!(obj is ActorMaterializerSettings s)) return false;
    return
        s.InitialInputBufferSize == InitialInputBufferSize &&
        s.MaxInputBufferSize == MaxInputBufferSize &&
        s.Dispatcher == Dispatcher &&
        s.SupervisionDecider == SupervisionDecider &&
s.SubscriptionTimeoutSettings == SubscriptionTimeoutSettings &&
        s.IsDebugLogging == IsDebugLogging &&
        s.OutputBurstLimit == OutputBurstLimit &&
        s.SyncProcessingLimit == SyncProcessingLimit &&
        s.IsFuzzingMode == IsFuzzingMode &&
        s.IsAutoFusing == IsAutoFusing &&
s.SubscriptionTimeoutSettings == SubscriptionTimeoutSettings &&
        s.StreamRefSettings == StreamRefSettings;
}
```

**Akka.net**

Ошибки, которые  
скучно искать

```
public IEnumerable<IMigration> GetCurrentMigrations()
{
    var currentDbVersion = new DbVersion(int.Parse(GrandVersion.MajorVersion),
                                          int.Parse(GrandVersion.MinorVersion));

    return GetAllMigrations()
        .Where(x => currentDbVersion.CompareTo(x.Version) >= 0)
        .OrderBy(mg => mg.Version.ToString())
        .OrderBy(mg => mg.Priority)
        .ToList();
}
```

## GrandNode

Ошибки, которые  
происходят из-за  
незнания

```
public IEnumerable<IMigration> GetCurrentMigrations()
{
    var currentDbVersion = new DbVersion(int.Parse(GrandVersion.MajorVersion),
                                          int.Parse(GrandVersion.MinorVersion));

    return GetAllMigrations()
        .Where(x => currentDbVersion.CompareTo(x.Version) >= 0)
        .OrderBy(mg => mg.Version.ToString())
        .OrderBy(mg => mg.Priority)
        .ToList();
}
```

## GrandNode

Ошибки, которые  
происходят из-за  
незнания

V3078 Original sorting order will be lost after repetitive call to 'OrderBy' method. Use 'ThenBy' method to preserve the original sorting. Grand.Infrastructure MigrationManager.cs 40

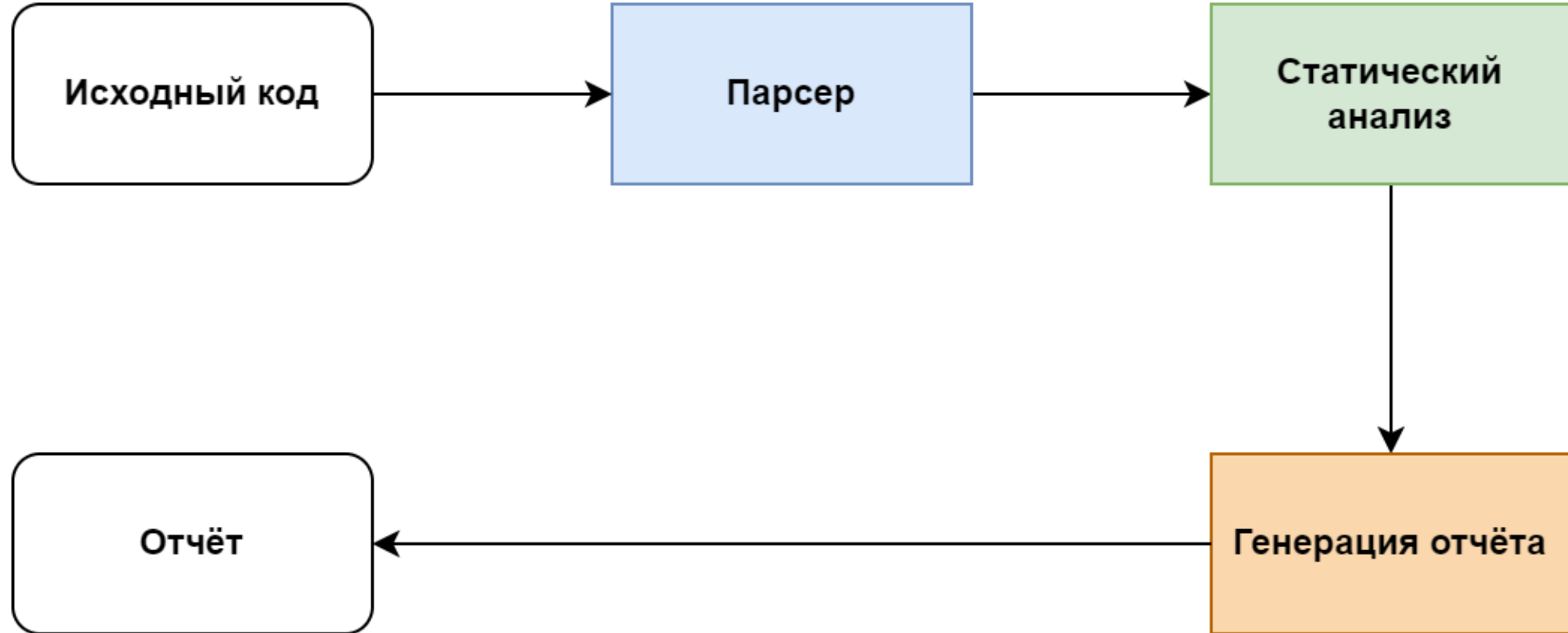
А как это работает?



# Анализатор

Scanning

Analyzing

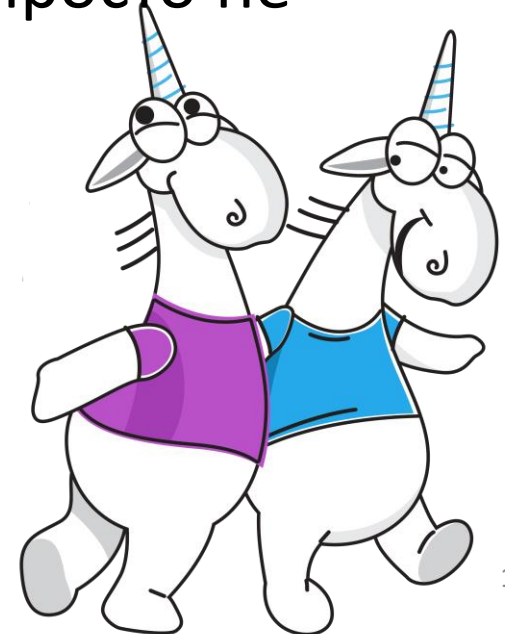


Building output

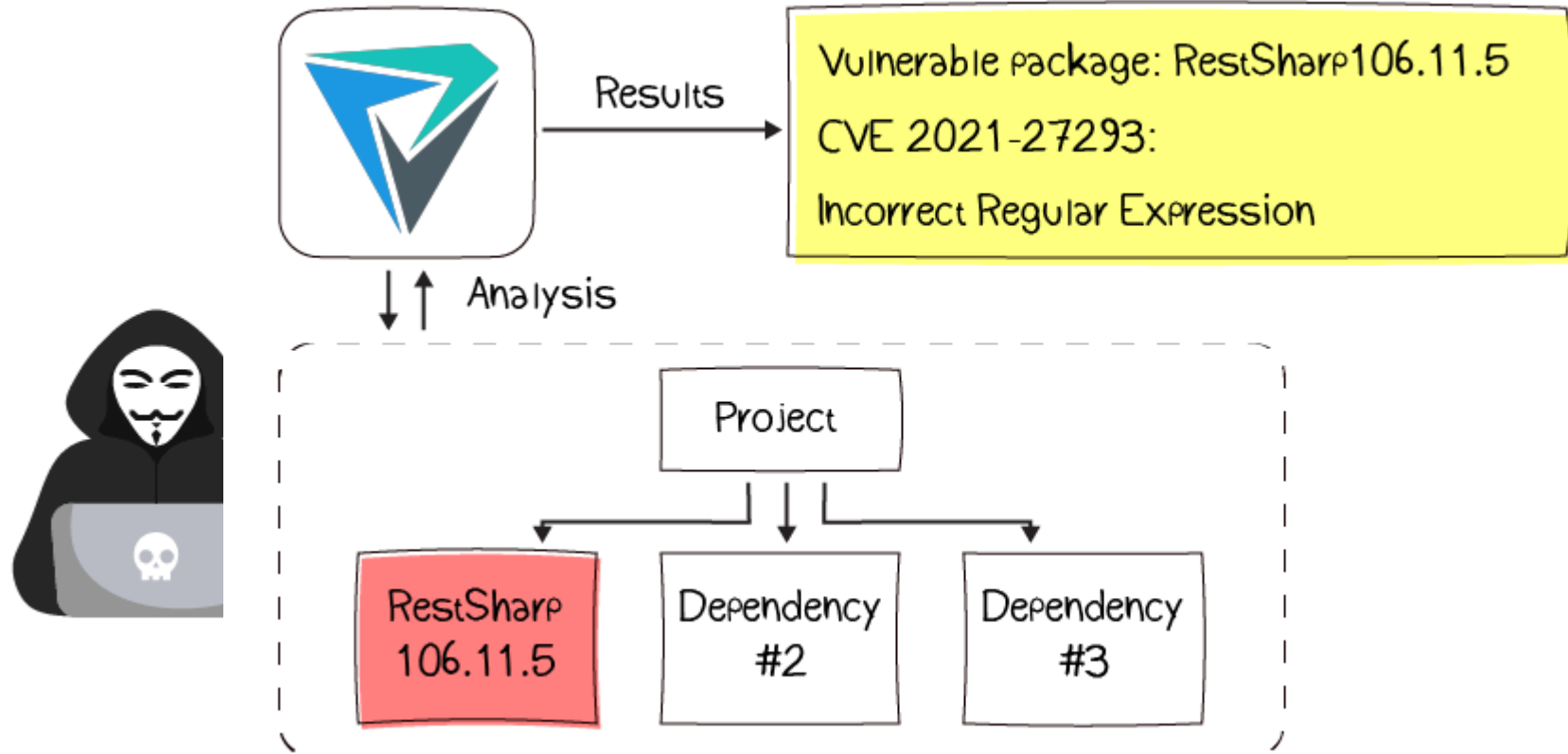
# Компилятор vs анализатор

**Главная цель анализатора – сделать код более качественным**

- Поэтому анализаторы не ограничены временем анализа
- А также имеют технологии, которые компиляторам просто не нужны:
  - Межмодульный и межпроцедурный анализ
  - SCA анализ
  - Анализ сторонних фреймворков
  - Taint анализ



# Software Composition Analysis (OWASP A06:2021)



# Анализ сторонних фреймворков (Unity)



# Daggerfall

```
public static string GetTerrainName(int mapPixelX, int mapPixelY)
{
    return string.Format("DaggerfallTerrain [{0},{1}]",
        mapPixelX,
        mapPixelY);
}
```

# Daggerfall

```
public static string GetTerrainName(int mapPixelX, int mapPixelY)
{
    return string.Format("DaggerfallTerrain [{0},{1}]",
        mapPixelX,
        mapPixelY);
}
```

**Упаковка**



# Daggerfall

```
public static string GetTerrainName(int mapPixelX, int mapPixelY)
{
    return string.Format("DaggerfallTerrain [{0},{1}]",
        mapPixelX.ToString(),
        mapPixelY.ToString());
}
```

# hogwarts

```
public BattleBlockData GetBlockData(int xindex, int yindex)
{
    return _battleBlocks.FirstOrDefault(x => x.BattlePos.X == xindex
                                            && x.BattlePos.Y == yindex);
}

public BattleBlockData GetRangelockData(int xindex, int yindex)
{
    return _rangeLayerBlocks.FirstOrDefault(x => x.BattlePos.X == xindex
                                              && x.BattlePos.Y == yindex);
}
```



# hogwarts

```
public BattleBlockData GetBlockData(int xindex, int yindex)
{
    return _battleBlocks.FirstOrDefault(x =>    x.BattlePos.X == xindex
                                             && x.BattlePos.Y == yindex);
}
```

**Захват значения**

```
public BattleBlockData GetRangeLockData(int xindex, int yindex)
{
    return _rangeLayerBlocks.FirstOrDefault(x =>    x.BattlePos.X == xindex
                                             && x.BattlePos.Y == yindex);
}
```

# hogwarts

```
public BattleBlockData GetBlockData(int xindex, int yindex)
{
    foreach (var block in _battleBlocks)
    {
        if ( block.BattlePos.X == xindex
            && block.BattlePos.Y == yindex)
        {
            return block;
        }
    }
    return null;
}
```

# Анализ помеченных данных (taint анализ)

- Taint-анализ – это технология, позволяющая отслеживать распространение непроверенных внешних данных по программе во время её работы.
- Примеры:
  - SQL-инъекции
  - Cross-site scripting (XSS)
  - Path traversal
  - XXE/XEE



# Welcome to BlogEngine.NET

Administrator © May 20, 2018 BlogEngine.NET

share



If you see this post it means that BlogEngine.NET is running and the hard part of creating your own blog is done. There is only a few things left to do.

[DOWNLOAD THEMES](#)

[OFFICIAL WEBSITE](#)

[DONATE](#)

```
private void LoadXmlRequest(string xml)
{
    var request = new XmlDocument();
    try
    {
        if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method")))
        {
            xml = xml.Substring(xml.IndexOf("<?xml"));
        }
        request.LoadXml(xml);
    }
    catch (Exception ex)
    {
        throw new MetaWeblogException("01", $"Invalid XMLRPC Request. ({ex.Message})");
    }
    ....
}
```

```
private void LoadXmlRequest(string xml)
{
    var request = new XmlDocument();
    try
    {
        if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method")))
        {
            xml = xml.Substring(xml.IndexOf("<?xml"));
        }
        request.LoadXml(xml);
    }
    catch (Exception ex)
    {
        throw new MetaWeblogException("01", $"Invalid XMLRPC Request. ({ex.Message})");
    }
    ...
}
```



# Кто этот ваш XML

В XML используется DTD (document type definition), который вместе с описанием структуры XML-документа, даёт возможность определять и использовать XML-сущности

```
<!ENTITY myExternalEntity SYSTEM  
"file:///C:/Windows/System32/drivers/etc/hosts">  
<order>&myExternalEntity;</order>
```



## Условия успешной ХХЕ

1. Нужен небезопасный обработчик пользовательского ввода
2. Нужен сам пользовательский ввод



```
internal class MetaWeblogHandler : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        try
        {
            var rootUrl = Utils.AbsoluteWebRoot.ToString();

            // context.Request.Url.ToString().Substring(0,
            // context.Request.Url.ToString().IndexOf("metaweblog.axd"));

            var input = new XMLRPCRequest(context);
            ....
        }
        ....
    }
}
```

```
internal class MetaWeblogHandler : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        try
        {
            var rootUrl = Utils.AbsoluteWebRoot.ToString();

            // context.Request.Url.ToString().Substring(0,
            // context.Request.Url.ToString().IndexOf("metaweblog.axd"));

            var input = new XMLRPCRequest(context);

            ....
        }
        ....
    }
}
```

```
public XMLHttpRequest(HttpContext input)
{
    var inputXml = ParseRequest(input);

    // LogMetaWeblogCall(inputXml);
    this.LoadXmlRequest(inputXml);
}

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

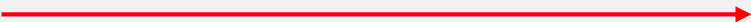
    context.Request.InputStream.Position = 0;
    context.Request.InputStream.Read(buffer, 0, buffer.Length);

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

```
public XMLHttpRequest(HttpContext input)
{
    var inputXml = ParseRequest(input);

    // LogMetaWeblogCall(inputXml);
    this.LoadXmlRequest(inputXml);
}

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);
}
```

```
private void LoadXmlRequest(string xml)
{
    var request = new XmlDocument();
    try
    {
        if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method")))
        {
            xml = xml.Substring(xml.IndexOf("<?xml"));
        }
        request.LoadXml(xml);
    }
    catch (Exception ex)
    {
        throw new MetaWeblogException("01", $"Invalid XMLRPC Request. ({ex.Message})");
    }
    ....
}
```

```
private void LoadXmlRequest(string xml)
{
    var request = new XmlDocument();
    try
    {
        if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method")))
        {
            xml = xml.Substring(xml.IndexOf("<?xml"));
        }
        request.LoadXml(xml);
    }
    catch (Exception ex)
    {
        throw new MetaWeblogException("01", $"Invalid XMLRPC Request. ({ex.Message})");
    }
    ....
}
```

```
private void LoadXmlRequest(string xml)
```

```
{
```

```
var request = new XmlDocument();
```

**.NET Framework 4.5**

```
try
```

```
{
```

```
if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method")))
```

```
{
```

```
xml = xml.Substring(xml.IndexOf("<?xml"));
```

```
}
```

```
request.LoadXml(xml);
```

```
}
```

```
catch (Exception ex)
```

```
{
```

```
throw new MetaWeblogException("01", $"Invalid XMLRPC Request. ({ex.Message})");
```

```
}
```

```
....
```

```
}
```

```
private void LoadXm
{
    var request = new
    try
    {
        if (!(xml.Start
        {
            xml = xml.Sub
        }
        request.LoadXm
    }
    catch (Exception
    {
        throw new Meta
    }
    ....
}
```

```
C:\Users\svasi\AppData\Roamir x + v - □ x
<?xml version="1.0" encoding="utf-8"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value>02</value>
        </member>
        <member>
          <name>faultString</name>
          <value>Unknown Method. (# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com       # source server
#       38.25.63.10      x.acme.com         # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1        localhost
#       ::1              localhost
#
# A special comment indicating that XXE attack was performed successfully.
#)</value>
      </member>
    </struct>
  </value>
</fault>
```

...Message}));



```
private void LoadXmlRequest(string xml)
{
    var request = new XmlDocument() { XmlResolver = null };
    try
    {
        if (!(xml.StartsWith("<?xml") || xml.StartsWith("<method")))
        {
            xml = xml.Substring(xml.IndexOf("<?xml"));
        }
        request.LoadXml(xml);
    }
    catch (Exception ex)
    {
        throw new MetaWeblogException("01", $"Invalid XMLRPC Request. ({ex.Message})");
    }
    ....
}
```

# XXE

- CWE-611:  
Improper Restriction of XML External Entity Reference
- CWE Top 25: 23
- OWASP Top 10:
  - 2017: A4:2017 – XML External Entities (XXE)
  - 2021: A05:2021 – Security Misconfiguration
- OWASP ASVS 4.0.3: 5.5.2

**Последствия:** утечки данных, SSRF

# ГОСТ Р 71207–2024

- Стандарт устанавливает требования к внедрению, методам работы и методике проверки требований к инструментам анализа.
- Входит в комплекс стандартов, направленных на предотвращение уязвимостей в программах и применяется совместно с ГОСТ Р 56939.
- Требуемые методы работы:
  - ...
  - Межпроцедурный и межмодульный контекстно-чувствительный анализ **помеченных** данных (**taint checking**);
  - ...



Статья



Видео

Больше примеров ошибок

## BTCPay Server Check

```
if ( request.PaymentTolerance < 0
    && request.PaymentTolerance > 100)
ModelState.AddModelError(nameof(request.PaymentTolerance),
    "PaymentTolerance can only be between 0 and 100 percent");
```

## BTC Pay Server Check

```
if ( request.PaymentTolerance < 0
    && request.PaymentTolerance > 100)
    ModelState.AddModelError(nameof(request.PaymentTolerance),
        "PaymentTolerance can only be between 0 and 100 percent");
```

PVS-Studio: V3022 Expression 'request.PaymentTolerance < 0 && request.PaymentTolerance > 100' is always false. Probably the '||' operator should be used here. GreenfieldStoresController.cs 241

# MudBlazor

```
var exp = new Expression(CalcExpression);  
var result = exp.Eval();  
if (result == double.NaN)  
{  
    Current = "ERROR";  
    return;  
}
```



# MudBlazor

```
var exp = new Expression(CalcExpression);  
var result = exp.Eval();  
if (result == double.NaN)  
{  
    Current = "ERROR";  
    return;  
}
```

PVS-Studio: V3076 Comparison of 'result' with 'double.NaN' is meaningless. Use 'double.IsNaN()' method instead.

# Entity Framework

```
for (var i = 0; i < result.Count; i++)
{
    for (var j = 0; j < expectedInnerNames.Count; j++)
    {
        Assert.True(
            result[i].OneToMany_Optional.Select(e => e.Name)
                .Contains(expectedInnerNames[j]));
    }
}
```

# Entity Framework

```
for (var i = 0; i < result.Count; i++)
{
    for (var j = 0; j < expectedInnerNames.Count; j++)
    {
        Assert.True(
            result[i].OneToMany_Optional.Select(e => e.Name)
                .Contains(expectedInnerNames[i]));
    }
}
```

PVS-Studio: V3081 The 'j' counter is not used inside a nested loop. Consider inspecting usage of 'i' counter. EFCore.Specification.Tests ComplexNavigationsQueryTestBase.cs 2393

# eShopOnContainers

....

```
var firstUrl = new Uri(urlHook, UriKind.Absolute);
```

```
var secondUrl = new Uri(url, UriKind.Absolute);
```

```
return firstUrl.Scheme == secondUrl.Scheme &&
```

```
    firstUrl.Port == secondUrl.Port &&
```

```
    firstUrl.Host == firstUrl.Host;
```

# eShopOnContainers

....

```
var firstUrl = new Uri(urlHook, UriKind.Absolute);
```

```
var secondUrl = new Uri(url, UriKind.Absolute);
```

```
return firstUrl.Scheme == secondUrl.Scheme &&
```

```
    firstUrl.Port == secondUrl.Port &&
```

```
    firstUrl.Host == firstUrl.Host;
```

PVS-Studio: V3001 There are identical sub-expressions 'firstUrl.Host' to the left and to the right of the '==' operator. GrantUrlTesterService.cs 48

# AWS SDK for .NET

```
var parts = path.Split('.');
for (int i = 0; i < parts.Length; i++)
{
    var part = parts[i];
    if (!(propertyValue is IList))
        throw new ArgumentException(
            "Object addressing by pathing segment '{part}' with indexer must be IList");

    if (!(propertyValue is IPropertyBag))
        throw new ArgumentException(
            "Object addressing by pathing segment '{part}' must be IPropertyBag");
    ....
}
```

# AWS SDK for .NET

```
var parts = path.Split('.');
for (int i = 0; i < parts.Length; i++)
{
    var part = parts[i];
    if (!(propertyValue is IList))
        throw new ArgumentException(
            "Object addressing by pathing segment '{part}' with indexer must be IList");

    if (!(propertyValue is IPropertyBag))
        throw new ArgumentException(
            "Object addressing by pathing segment '{part}' must be IPropertyBag");

    .... PVS-Studio: V3138 String literal contains potential interpolated expression. Consider
    inspecting: part. Fn.cs 82
}
```

# Интеграция в проект



# Проблемы с новыми инструментами



Внедрение



Использование

Как внедрить анализатор в  
существующий проект?

## Подход №0

### Рефакторинг

Выделяем время и исправляем все предупреждения

- Эффективно избавляет проект от предупреждений
- Неэффективно избавляет проект от критических проблем
- Больно

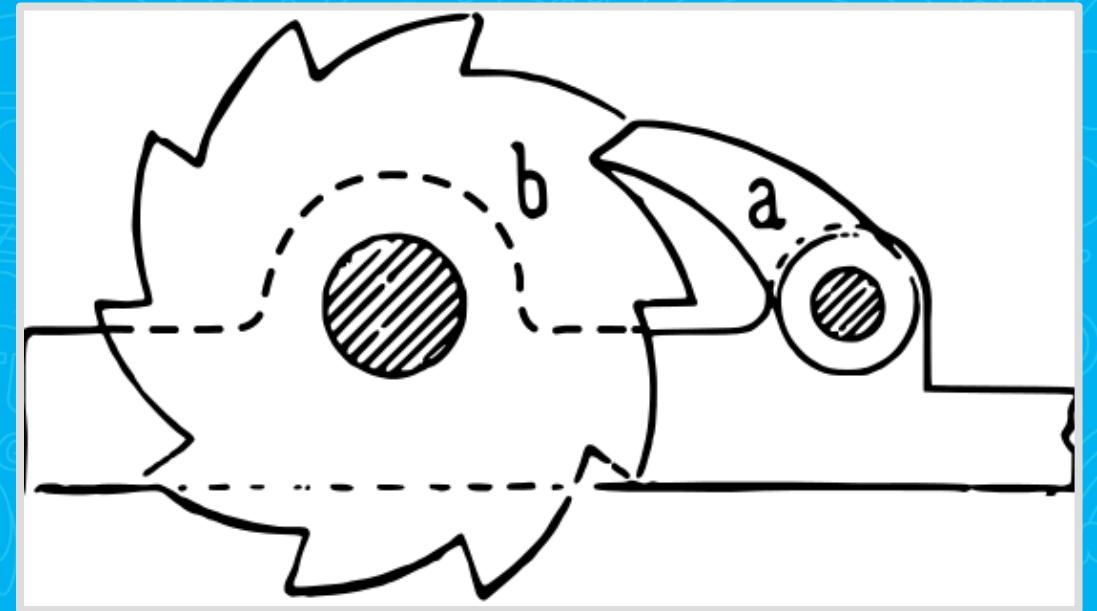


# Подход №1

## Храповик

Можно коммитить код, только если сохраняется заданный уровень предупреждений

- Гарантирует, что про предупреждения анализатора не забудут
- Даже если разработчик будет исправлять чужие ошибки вместо своих, общий тренд будет ПОЗИТИВНЫМ

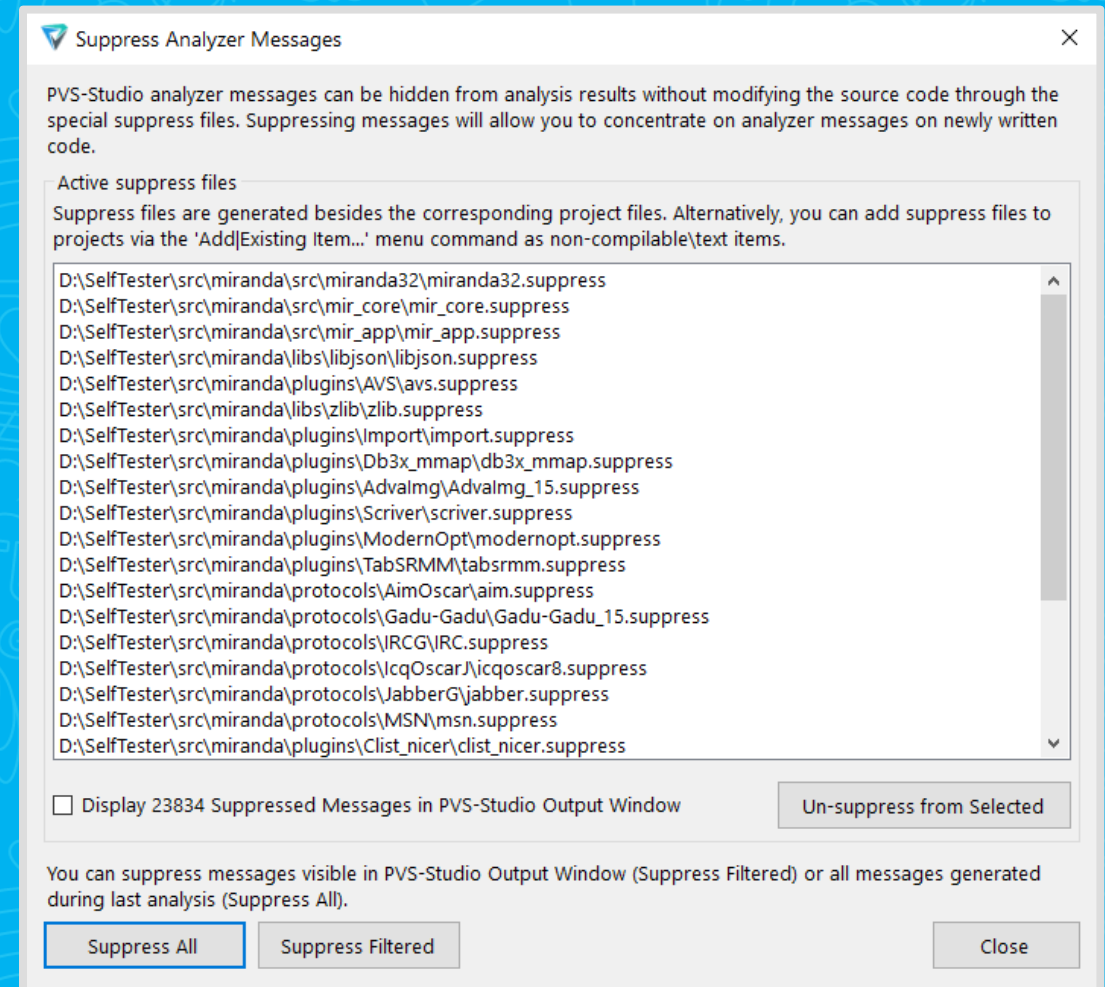


# Подход №2

## Base-line (база разметки)

Подавить все старые предупреждения

- Можно сразу начинать работать с новыми предупреждениями анализатора
- При желании, всегда можно вернуться к рефакторингу и расподавить старые предупреждения



Как его использовать?

# Интеграции

Анализаторы интегрируются с разными системами для более удобного использования:

- IDE
- CI/CD
- Docker



# Интеграции в PVS-Studio


## IDE

 [Visual Studio](#)

 [IntelliJ IDEA](#)

 [Rider](#)

 [CLion](#)

 [Visual Studio Code](#)

 [Qt Creator](#)

 [Eclipse](#)

## Distributed build

 [Incredibuild](#)

## Game engines

 [Unreal Engine](#)

 [Unity](#)

## Code quality

 [SonarQube](#)

 [DefectDojo](#)

## Build systems

 [MSBuild](#)

 [CMake](#)

 [Make](#)


 [Ninja](#)


 [Gradle](#)

 [Maven](#)

 [JSON Compilation Database](#)

## Embedded

 [Keil μVision, DS-MDK](#)

 [IAR Embedded Workbench](#)

 [Platform.io](#)

 [QNX Momentics](#)

 [TI ARM Code Generation](#)

## Virtualization

 [Docker](#)

 [WSL](#)

## CI

 [Jenkins](#)


 [TeamCity](#)

## Cloud CI

 [CircleCI](#)

 [Travis CI](#)

 [GitLab](#)

 [Azure DevOps](#)

 [GitHub Actions](#)





Вопросы!

Q&A

Константин Волоховский

@kvolokhovskii

```
for (int i = 0; i < 20; i++)
{
    ....
    if (i % 2 == 0)
    {
        thread1.Start();
        thread2.Start();
    }
    else
    {
        thread1.Start();
        thread2.Start();
    }
    ....
}
```

## .NET Compiler Platform (Roslyn)

```
for (int i = 0; i < 20; i++)
{
    ....
    if (i % 2 == 0)
    {
        thread1.Start();
        thread2.Start();
    }
    else
    {
        thread1.Start();
        thread2.Start();
    }
    ....
}
PVS-Studio: V3004 The 'then' statement is equivalent to the 'else'
statement. GetSemanticInfoTests.cs 2269
```

## .NET Compiler Platform (Roslyn)

```
public List<Result> Query(Query query)
{
    bool isGlobalQuery = string.IsNullOrEmpty(query.ActionKeyword);
    ....
    if (query == null)
    {
        throw new ArgumentNullException(paramName: nameof(query));
    }
}
```

**Microsoft Power  
Toys**

```
public List<Result> Query(Query query)
{
    bool isGlobalQuery = string.IsNullOrEmpty(query.ActionKeyword);
    ....
    if (query == null)
    {
        throw new ArgumentNullException(paramName: nameof(query));
    }
}
```

**Microsoft Power  
Toys**

PVS-Studio: V3095 The 'query' object was used before it was verified against null. Check lines: 56, 60. Main.cs 56

```
public void SetString(string key, string value)
{
    if (m_syncDictionary.TryGetValue(key,
                                     out string existingValue))
    {
        if (value != existingValue)
        {
            m_syncDictionary[key] = value;
        }
    }
    m_syncDictionary[key] = value;
}
```

**SanAndreasUnity**

```
public void SetString(string key, string value)
{
    if (m_syncDictionary.TryGetValue(key,
                                     out string existingValue))
    {
        if (value != existingValue)
        {
            m_syncDictionary[key] = value;
        }
    }
    m_syncDictionary[key] = value;
}
```

**SanAndreasUnity**

V3008 The 'm\_syncDictionary[key]' variable is assigned values twice successively. Perhaps this is a mistake. Check lines: 112, 108. SyncedBag.cs 112

```
public static bool IsTypeOf<T>(this TypeReference tr)
{
    var type = typeof(T);
    return tr.Name == type.Name
        && tr.Namespace == tr.Namespace;
}
```

**.NET 8**



```
public static bool IsTypeOf<T>(this TypeReference tr)
{
    var type = typeof(T);
    return tr.Name == type.Name
        && tr.Namespace == tr.Namespace;
}
```

**.NET 8**

PVS-Studio: V3001 There are identical sub-expressions 'tr.Namespace' to the left and to the right of the '==' operator. TypeReferenceExtensions.cs 365

```
public static int Compare(SourceLocation left,  
                          SourceLocation right) {  
    if (left < right) return -1;  
    if (right > left) return 1;  
    return 0;  
}
```

**IronPython**

```
public static int Compare(SourceLocation left,
                          SourceLocation right) {
    if (left < right) return -1;
    if (right > left) return 1;
    return 0;
}
```

**IronPython**

PVS-Studio: V3021 There are two 'if' statements with identical conditional expressions. The first 'if' statement contains method return. This means that the second 'if' statement is senseless. SourceLocation.cs 156