Санкт-Петербургский государственный университет

Кафедра системного программирования

# Устранение ложных срабатываний статических анализаторов кода символьным исполнением для платформы .NET

Седлярский Михаил Андреевич, 21.M07-мм группа

**Научный руководитель:** к.ф.-м.н. Д.А. Мордвинов, доцент кафедры системного программирования

Санкт-Петербург
2022

# Введение

- Современная разработка нуждается в автоматическом анализе кода
- Методы статического анализа предрасположены к ложноположительным срабатываниям
- Символьное исполнение лишено такого недостатка, но плохо масштабируется

# Существующие подходы

- Легковесные анализаторы, микрограмматики (Google Sanitizers)
- Межпроцедурный анализ, сепарационная логика (Infer)
- Большие данные, обработка крупных семантических графов (Graspan)

# Существующие подходы

В данной работе предлагается объединить методы статического анализа и символьного исполнения. Можно подкреплять и валидировать результаты статического анализатора таргетированным исполнением в символьной виртуальной машине (прим. TASMAN)

# Постановка задачи

**Целью** разработка приложения устраняющего ложноположительные срабатывания статических анализаторов кода символьным исполнением для платформы .NET

**Задачи (осень)**:

- провести сравнительный анализ статических анализаторов кода для платформы .NET;
- запустить статические анализаторы на настоящих проектах. На основании полученных результатов выбрать анализатор кода, с которым будут проводиться дальнейшие работы;
- разработать и реализовать алгоритм преобразования результатов статического анализатора кода в цели для движка символьного исполнения.

# Постановка задачи

**Задачи (весна)**:

- разработать и реализовать способ передачи целей в движок символьного исполнения;
- разработать и реализовать алгоритм ранжирования результатов статического анализа на основе результатов символьного исполнения
- провести эксперименты и оценить качество полученного инструмента.

# Сравнение анализаторов

Таблица: Сводная информация о статических анализаторах .NET

| Название | Поддерживаемые платформы | | | Поддержка SARIF | Open source |
|---|---|---|---|---|---|
| | win | linux | macos | | |
| Roslyn analysers | + | + | + | + | + |
| ReSharper | + | + | + | + | - (есть бесплатные лицензии) |
| PVS-Studio | + | + | + | + (через plog-converter) | - (есть бесплатные лицензии) |
| InferSharp | + (через wsl) | + | ± (работает через docker, но теоретически можно собрать нативно) | + | + |
| Sonar Scanner for .NET + sonar-dotnet | + | + | + | + | + |
| CHECKMARX SAST | web решение | | | | - (можно получить демо версию) |

# Сравнение анализаторов (практика)

Таблица: Результаты запуска анализаторов на 17 проектах

| Проект | Infer# | | | | sonar-scanner-msbuild | | | pvs studio | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | roslyn | sonar-dotnet | прочее | | | | | |
| | NULLPTR_DEREFERENCE | PULSE_RESOURCE_LEAK | THREAD_SAFETY_VIOLATION | STACK_VARIABLE_ADDRESS_ESCAPE | Всего | Всего | Всего | V3022 (Expression <exp> is always <value>) | V3080 (Possible null dereference) | V3146 (Possible null dereference. A method can return default null value) | V3106 (Possibly index is out of bound) | Всего |
| efcore | 107 | 2 | 8 | 0 | 264 | 2893 | 487 | анализ решения съедает все ресурсы на сервере | | | | |
| litedb | 1 | 7 | 6 | 0 | 0 | 470 | 9 | 15 | 19 | 1 | 0 | 110 |
| moq4 | 0 | 1 | 0 | 0 | 214 | 441 | 5 | 4 | 2 | 0 | 0 | 61 |
| NLog | 30 | 75 | 44 | 0 | 17 | 266 | 12 | 34 | 9 | 0 | 1 | 163 |
| nunit | 26 | 192 | 7 | 0 | 218 | 856 | 4 | 28 | 8 | 0 | 0 | 203 |
| xunit | 0 | 0 | 0 | 0 | 292 | 1092 | 0 | 19 | 11 | 0 | 0 | 127 |
| btcpayserver | 4 | 6 | 4 | 0 | 267 | 1062 | 1 | 37 | 99 | 13 | 0 | 374 |
| AutoMapper | 0 | 0 | 0 | 0 | 88 | 167 | 0 | 2 | 1 | 0 | 0 | 38 |
| spbu-homework | 0 | 1 | 4 | 0 | 33 | 105 | 0 | 0 | 0 | 0 | 2 | 5 |
| parallel-program | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| parallel-program | 0 | 0 | 0 | 0 | 11 | 3 | 6 | 0 | 0 | 0 | 0 | 0 |
| parallel-program | 0 | 0 | 0 | 0 | 8 | 10 | 0 | 0 | 0 | 0 | 0 | 2 |
| BenchmarkDotN | 2 | 11 | 0 | 0 | 18 | 19 | 0 | 14 | 0 | 0 | 0 | 82 |
| ILSpy (ILSpy.XP | 48 | 4 | 0 | 2 | 873 | 1829 | 0 | 98 | 128 | 0 | 17 | 792 |
| OpenRA | 34 | 49 | 1 | 0 | 0 | 2990 | 0 | 13 | 39 | 0 | 2 | 342 |
| Newtonsoft.Json | 14 | 271 | 5 | 0 | 1478 | 740 | 0 | 30 | 10 | 0 | 34 | 265 |
| RestSharp | 0 | 53 | 0 | 0 | 28 | 47 | 9 | 0 | 1 | 0 | 0 | 16 |

# Сравнение анализаторов

**Был выбран** PVS-studio т.к. выдаёт интересные с точки символьного исполнения результаты, обладает относительным разнообразием правил

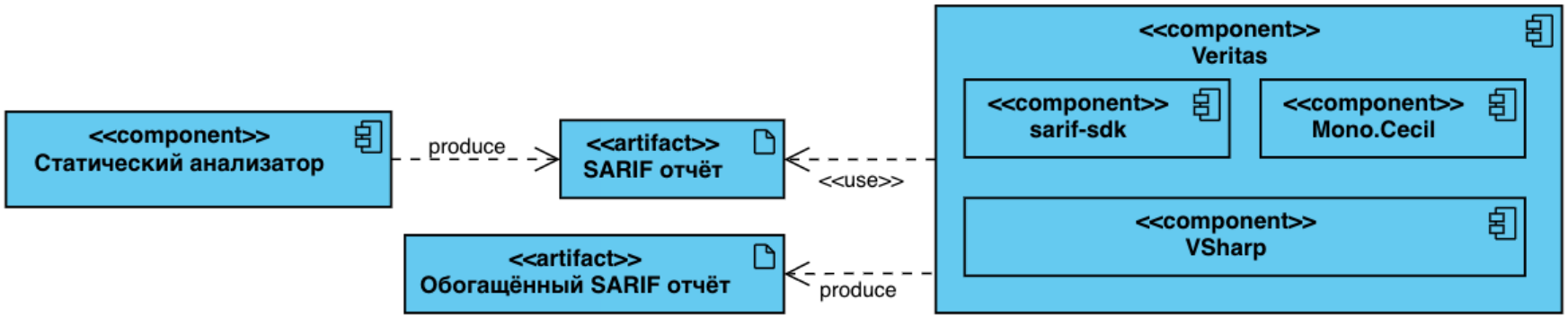
**Побочные результаты:**

- были найдены баги в Infer# и PVS-Studio; Собранная диагностика помогла их оперативно исправить

# Общая структура решения

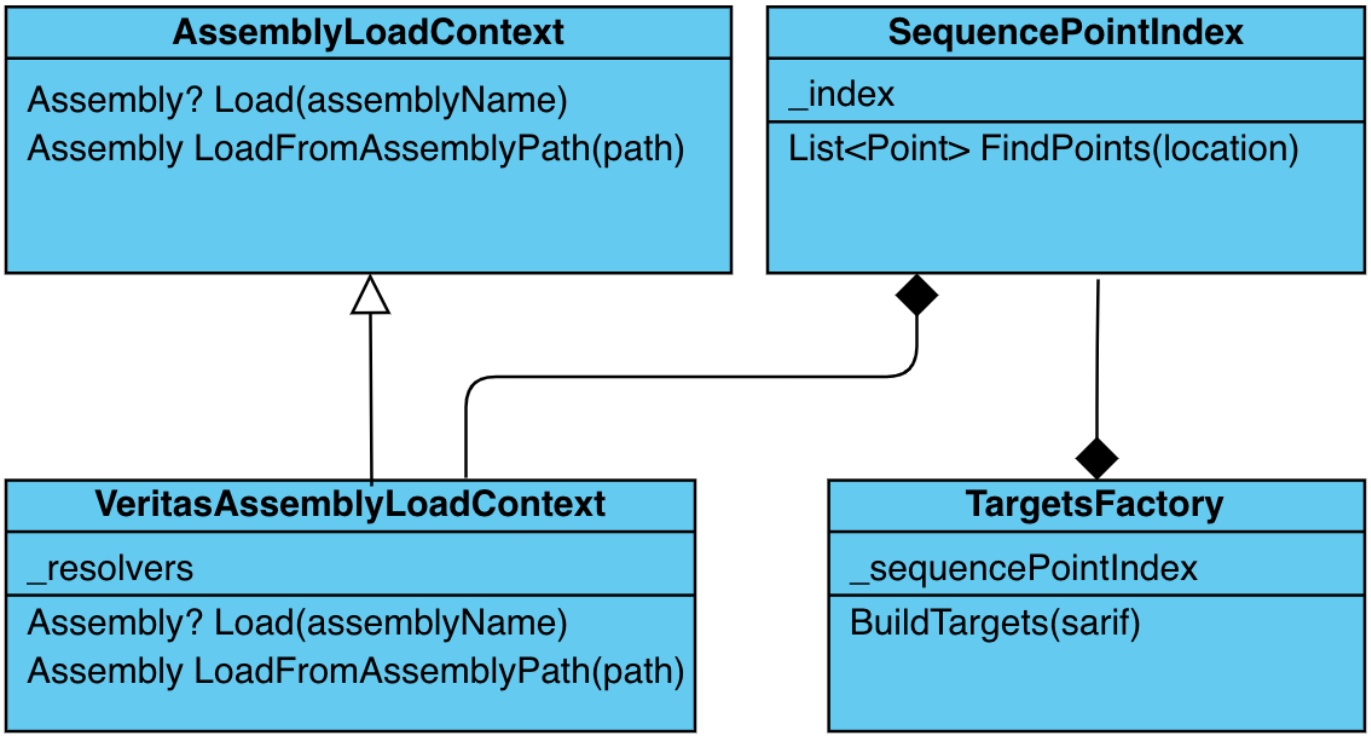
Обёртка вокруг V#, которая преобразует SARIF отчёты анализаторов в цели символьного исполнения, а затем ранжирует или обогащает изначальный отчёт основываясь на результатах символьного анализа

Рис.: Диаграмма развёртывания



# Общая структура решения

Рис.: Диаграмма классов



# Экспериментальное исследование

Постановка эксперимента

- Проверим насколько полно Veritas может покрыть целями (targets) ошибки из отчётов PVS-Studio
- Генератор целей Veritas будет запускаться на 3-ёх проектах: BTCPayserver, NLog и LiteDB
- Затем подсчитывается доля срабатываний, для которых получилось сгенерировать цели, от общего числа срабатываний подходящих типов: V3080 (Possible null dereference), V3146 (Possible null dereference. A method can return default null value), V3106 (Possibly index is out of bound).

# Экспериментальное исследование

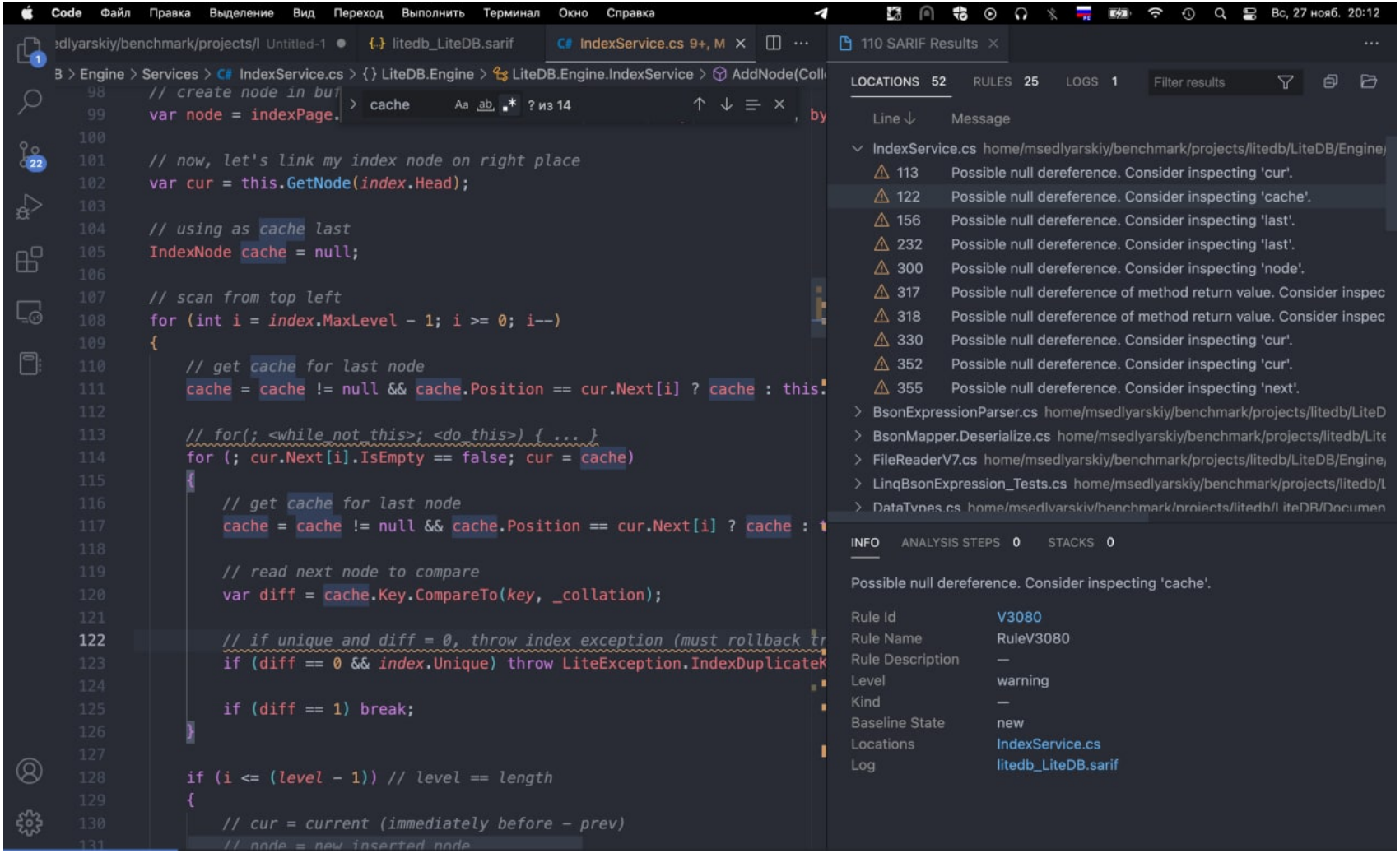
| Проект | Срабатывания с целями, шт | Поддерживаемые срабатывания, шт | Доля |
|---|---|---|---|
| BTCPayServer | 67 | 114 | 58,77% |
| NLog | 7 | 10 | 70,00% |
| LiteDB | 12 | 20 | 60,00% |

Таблица: Результаты построения целей для трёх проектов

- Чем можно объяснить, что от 30% до 40% результатов PVS-Studio не удалось отобразить на IL инструкцию?
- Результаты PVS-Studio, оставшиеся без целей, имеют некорректную привязку к местоположению в коде

# Экспериментальное исследование

Рис.: Примеры некорректной локализации ошибок анализатором PVS-Studio



# Результаты

- проведён сравнительный анализ статических анализаторов кода для платформы .NET
- были запущены статические анализаторы на настоящих проектах. На основании полученных результатов был выбран анализатор кода (PVS-Studio)
- разработан и реализован алгоритм преобразования результатов статического анализатора кода в цели для движка символьного исполнения V#
- разработан алгоритм загрузки сборок и зависимостей для платформы .net core
- были выявлены ошибки в анализаторах Infer# и PVS-Studio; оказано содействие разработчикам проектов в их устранении

# Результаты

Планы на следующий семестр:

- разработать и реализовать способ передачи целей в движок символьного исполнения
- разработать и реализовать алгоритм ранжирования результатов статического анализа на основе результатов символьного исполнения
- провести эксперименты и оценить качество полученного инструмента