



POSITIVE
TECHNOLOGIES

DevSecOps: внедрение в продуктовый конвейер и эксплуатация PT Application Inspector

Алексей Жуков

Эксперт отдела систем защиты приложений

Тимур Гильмуллин

Заместитель руководителя отдела технологий и процессов разработки

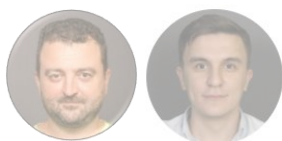
Дмитрий Рагулин

CI-инженер отдела технологий и процессов разработки

ptsecurity.com

PT Application Inspector: серия вебинаров

PT



22 октября

PT Application Inspector:
обзор новой версии и roadmap

- Проблематика уязвимостей в коде приложений
- Рассказ о продукте: возможности, демонстрация
- История развития и Roadmap

<https://www.youtube.com/watch?v=MSotiaSowmM>



Сегодня

**DevSecOps:
внедрение в продуктовый конвейер
и эксплуатация PT Application Inspector**

- Кейс внедрения PT Application Inspector в Positive Technologies (техническая часть)



Q1 2021

DevSecOps:
построение процесса устранения
уязвимостей в приложениях
на базе PT Application Inspector

- Кейс внедрения PT Application Inspector в Positive Technologies (организационная часть)

План вебинара



01

Введение

- От DevOps к DevSecOps
- PT Application Inspector: основные возможности и способы интеграции с CI-системами

02

PT Application Inspector в технологическом конвейере

- DevOps в Positive Technologies
- Релизная схема сборок с продвижениями
- Архитектура размещения и сборочный процесс
- Методика внедрения PT Application Inspector в сборочные CI-конвейеры

03

Компоненты решения

- Серверная часть
- Клиентская часть
- Интеграция с CI

04

Демонстрация

- Конвейер на GitLab CI
- Шаблоны в Open Source

05

Q&A



Введение

DevSecOps и PT Application Inspector



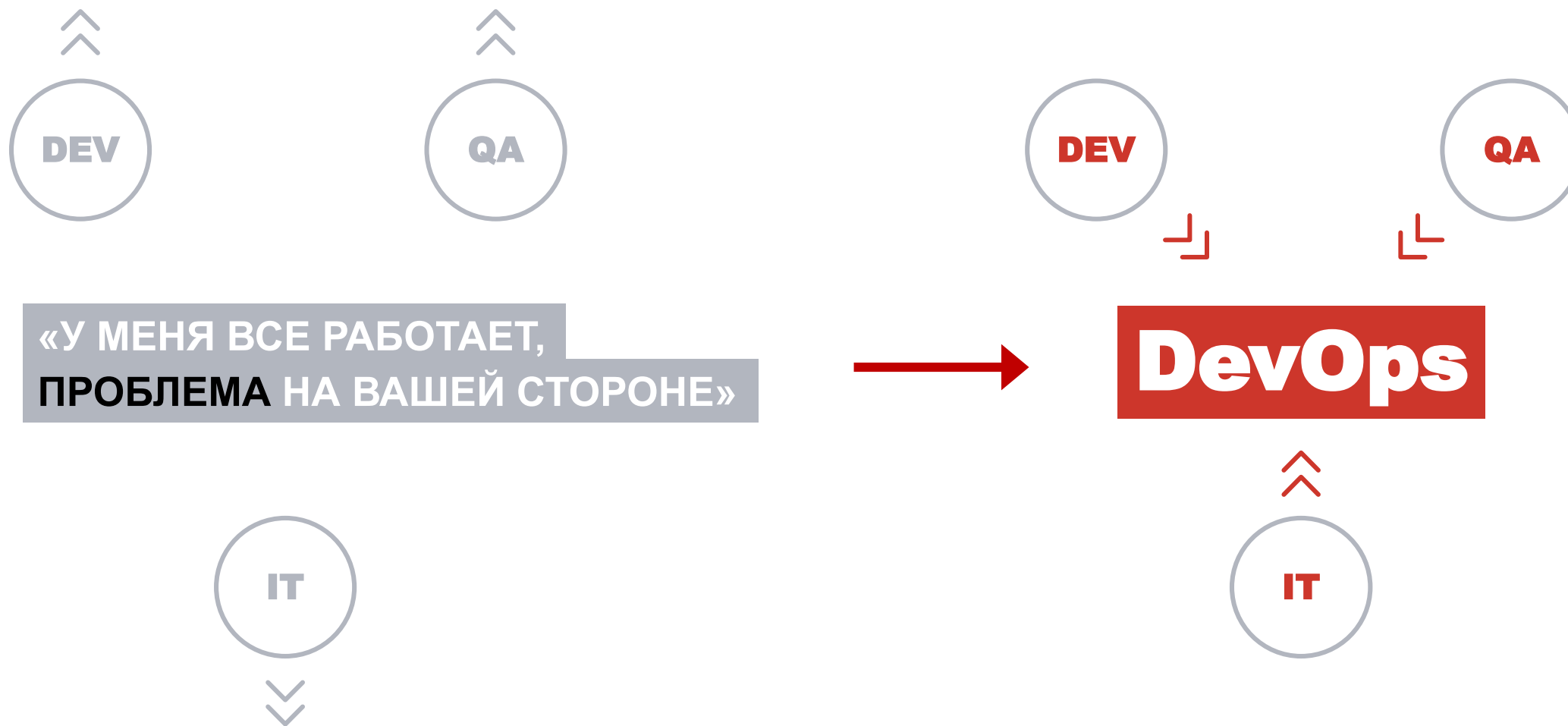
Алексей Жуков

Эксперт отдела систем защиты приложений

ptsecurity.com

Сначала о DevOps

PT



А как же безопасность?

50% веб-приложений имеют критически опасные уязвимости

Каждая пятая атака направлена на веб-приложения

82% уязвимостей содержатся в исходном коде приложения



Анализ кода как первый шаг



Что кроме кода

Анализ сторонних компонентов

Приложение — это не только код ваших разработчиков. Доля сторонних компонентов (third-party code) вполне сопоставима с объемом пользовательского кода.

Проверка конфигурационных файлов

Современные веб-приложения используют платформы веб-серверов и серверов приложений. Нюансы работы приложения определяются не только кодом, но и конфигурацией сервера.

Динамический анализ

DAST как способ анализа защищенности на этапе тестирования.

Встраивание в процессы

Интеграция с CI/CD

Для анализа нужен полный код приложения — независимо от того, в какой момент он формируется. Вместо плагинов для репозитория и запросов на доступ необходимы плагины для CI/CD-системы.

Возможности PT Application Inspector



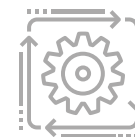
ТОЧНО НАХОДИТ И ПРИОРИТИЗИРУЕТ УЯЗВИМОСТИ

PT Application Inspector обнаруживает не только сами уязвимости, но и условия их эксплуатации — благодаря комбинации механизмов выявления уязвимостей и экспертизе Positive Technologies



ПРОВЕРЯЕТ ИСХОДНЫЙ КОД И ГОТОВЕ ПРИЛОЖЕНИЕ

PT Application Inspector использует статические, динамические и интерактивные методы (SAST, DAST и IAST), а по результатам анализа формирует безопасные эксплойты для ручной проверки наличия уязвимости



ИНТЕГРИРУЕТСЯ В ДЕЙСТВУЮЩИЕ ПРОЦЕССЫ РАЗРАБОТКИ

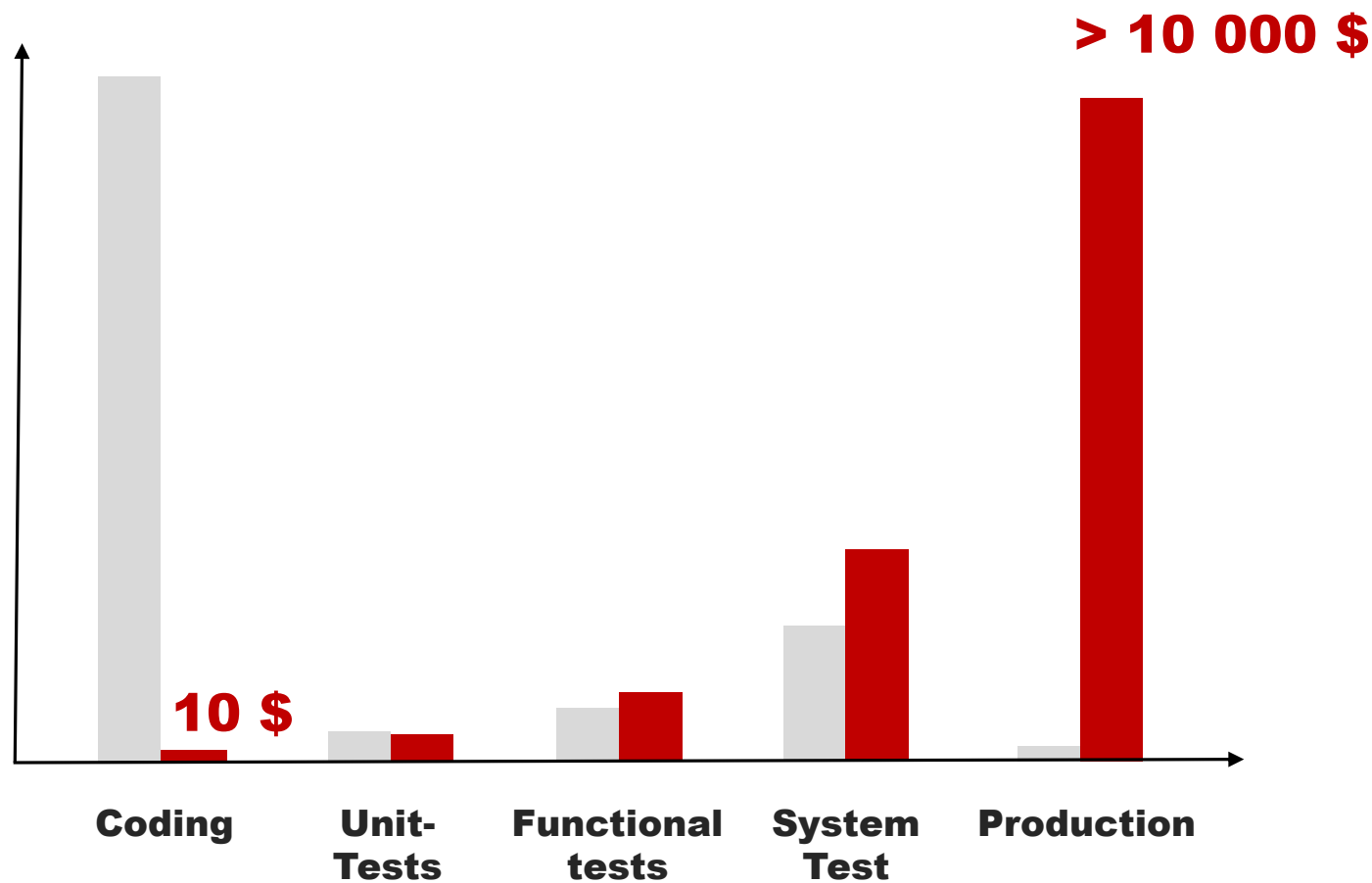
PT Application Inspector гибко встраивается в существующие процессы за счет готовых плагинов для подключения к системам сборки и доставки приложений, позволяет выстроить процесс безопасной разработки кода

Немного об экономике

PT

Сколько стоит баг, и зачем нужен shift-left

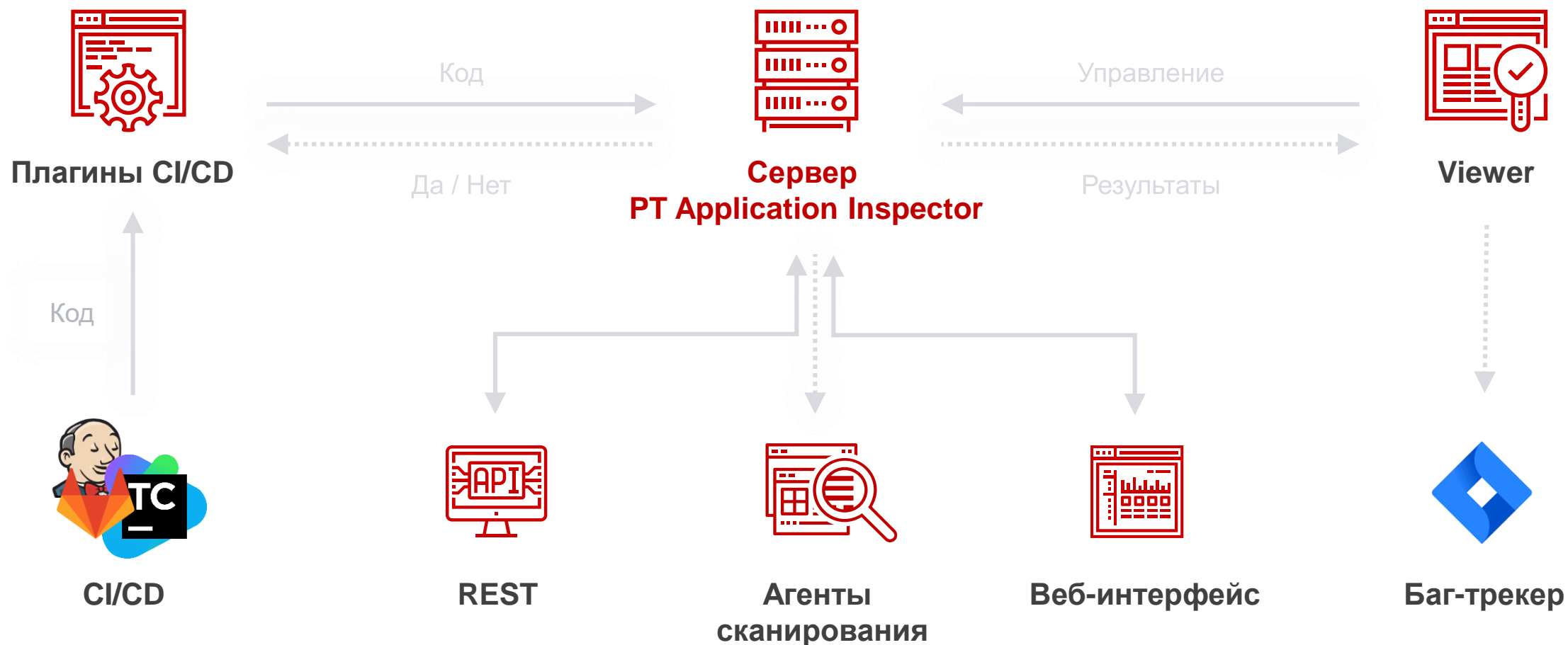
- Количество новых багов
- Стоимость исправления одного бага



Applied Software Measurement,
Capers Jones

Компоненты enterprise-решения

PT



PT Application Inspector в DevSecOps



ПРОХОЖДЕНИЕ СЕРТИФИКАЦИИ

Проведение анализа кода на самых ранних стадиях жизненного цикла снижает риск провалить сертификацию ПО на соответствие требованиям регуляторов



ОЦЕНКА КАЧЕСТВА КОДА

В составе решения реализован не только поиск уязвимостей в коде, но и анализ его качества, например поиск устаревших и неподдерживаемых функций или пустых обработчиков исключений



ПОИСК УЯЗВИМОСТЕЙ В ИСХОДНОМ КОДЕ ДЛЯ РАЗЛИЧНЫХ ЯЗЫКОВ

Анализатор поддерживает более десятка языков программирования и позволяет выявлять уязвимости с учетом используемых фреймворков и библиотек



АВТОМАТИЧЕСКОЕ БЛОКИРОВАНИЕ ВЫПУСКА УЯЗВИМЫХ КОМПОНЕНТОВ

Использование PT AI в режиме Security Gate позволяет задать правила автоматического анализа результатов для блокирования сборки при их несоответствии требованиям

Типы CI/CD-плагинов

PT Application Inspector

PT



Плагины для командной строки



Нативные плагины

- **Универсальность**
можно запускать из любой CI/CD-системы
- **Два варианта исполнения**
.NET или Java
- **Требуют развертывания**
на агентах сборки

- **Работа в парадигме CI/CD**
интеграция с Jenkins, TeamCity, Azure DevOps
- **Открытые решения**
github.com/PositiveTechnologies/ptaiPlugins
- **Не требуют развертывания**
на агентах сборки



PT Application Inspector в технологическом конвейере



Тимур Гильмуллин

Заместитель руководителя отдела технологий и процессов разработки

ptsecurity.com

Пара слов о DevOps в Positive Technologies



ПОДДЕРЖИВАЕМ CI-КОНВЕЙЕР

от коммита строки кода разработчиками
до публикации готовых продуктов и лицензий
на серверах обновлений

ДОСТИЖЕНИЯ

- Создали и с 2014 года развиваем CI-конвейер на основе типовых конфигураций и релизной схемы с продвижениями
- Внедрили конвейер в 10+ продуктов компании
- Делимся наработками в опенсорсе [DevOpsHQ](#) и на митапах Op!DevOps! ([2016](#), [2017](#))

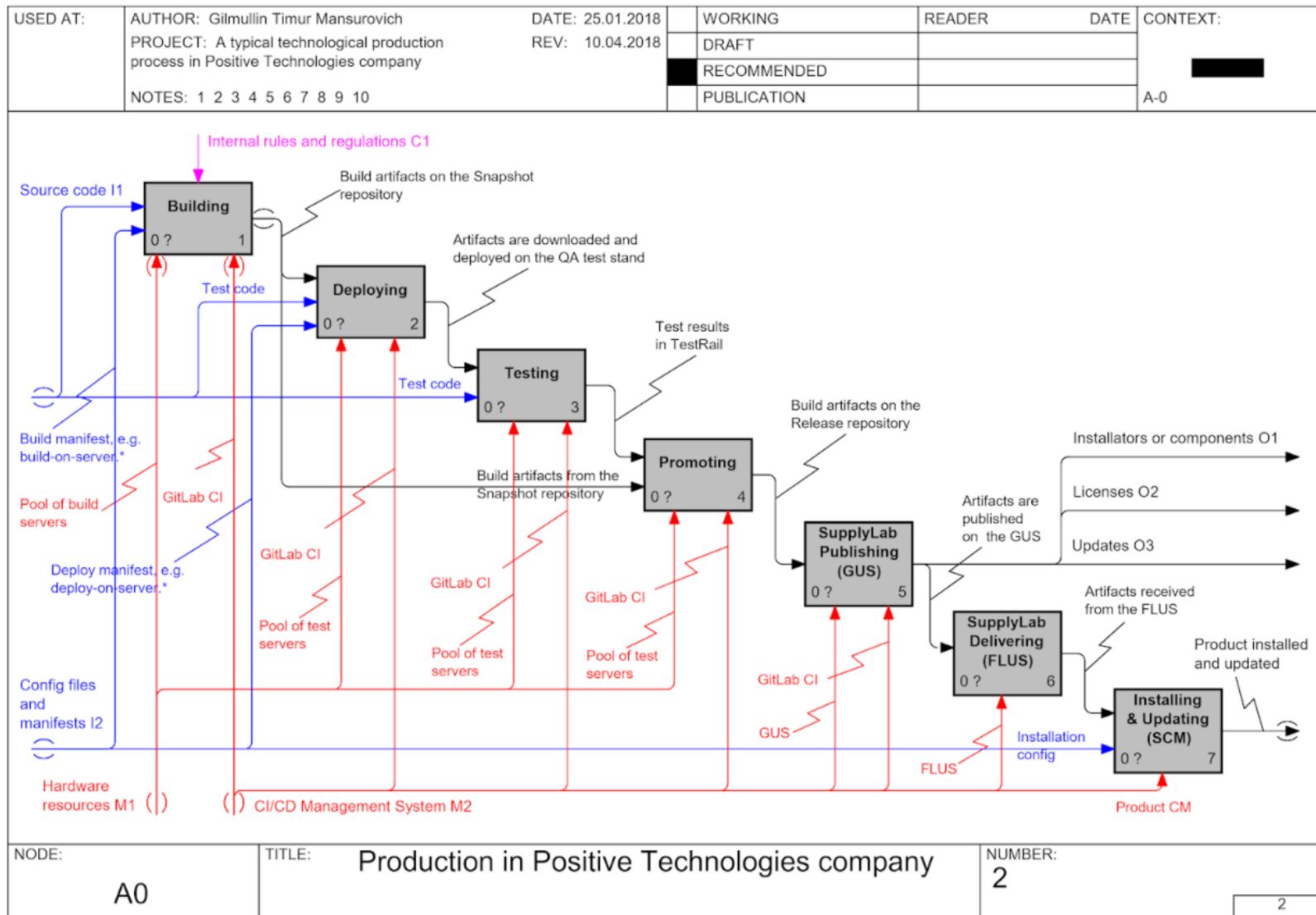
НЕМНОГО СТАТИСТИКИ

- > **9,5 К** сборочных конфигураций
- > **2,7 М** сборок в год
- > **8,2 ТБ** хранимых артефактов
- > **40+** билд-агентов в high, med, low сборочных пулах

ПИШЕМ СТАТЬИ

- [Личный опыт: как выглядит наша система Continuous Integration](#)
- [Автоматизация процессов разработки: как мы в Positive Technologies внедряли идеи DevOps](#)
- [Моделирование производственных процессов в ИТ-компаниях](#)
- [Управление хаосом: наводим порядок с помощью технологической карты](#)

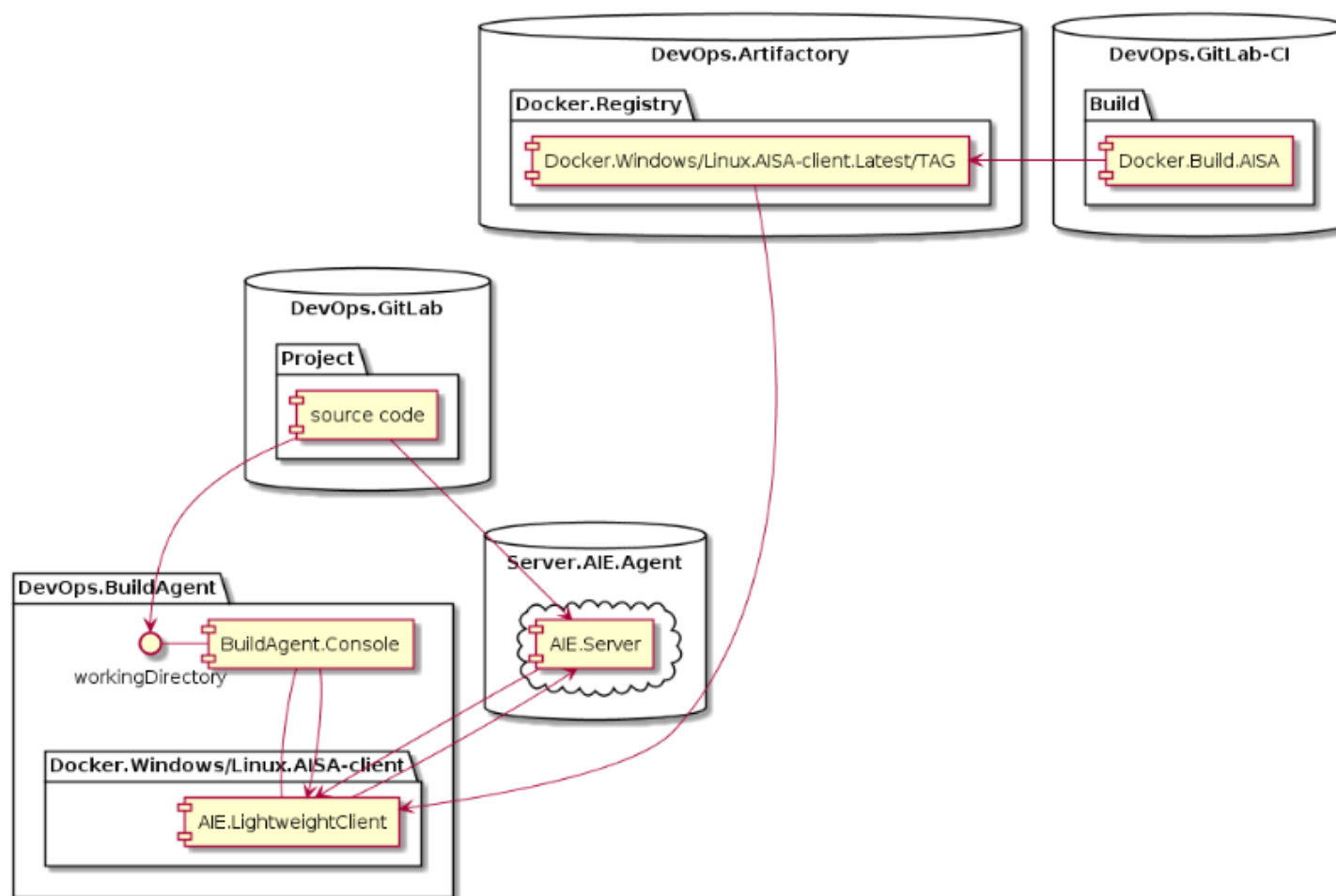
Релизная схема сборки с продвижениями



Легенда

- **Building**
Сборка компонентов и инсталлятора продукта, сохранение в snapshot-репозиторий на Artifactory
- **Deploying**
Деплой на тестовые стенды
- **Testing**
Автоматическое или ручное тестирование
- **Promoting**
Продвижение из snapshot-репозитория в release-репозиторий
- **Publishing**
Публикация на центральном сервере обновлений GUS
- **Delivering**
Тиражирование на серверы доставки обновлений заказчиком FLUS
- **Installing & Updating**
Установка или обновление продукта в инфраструктуре заказчика

Архитектура размещения PT Application Inspector

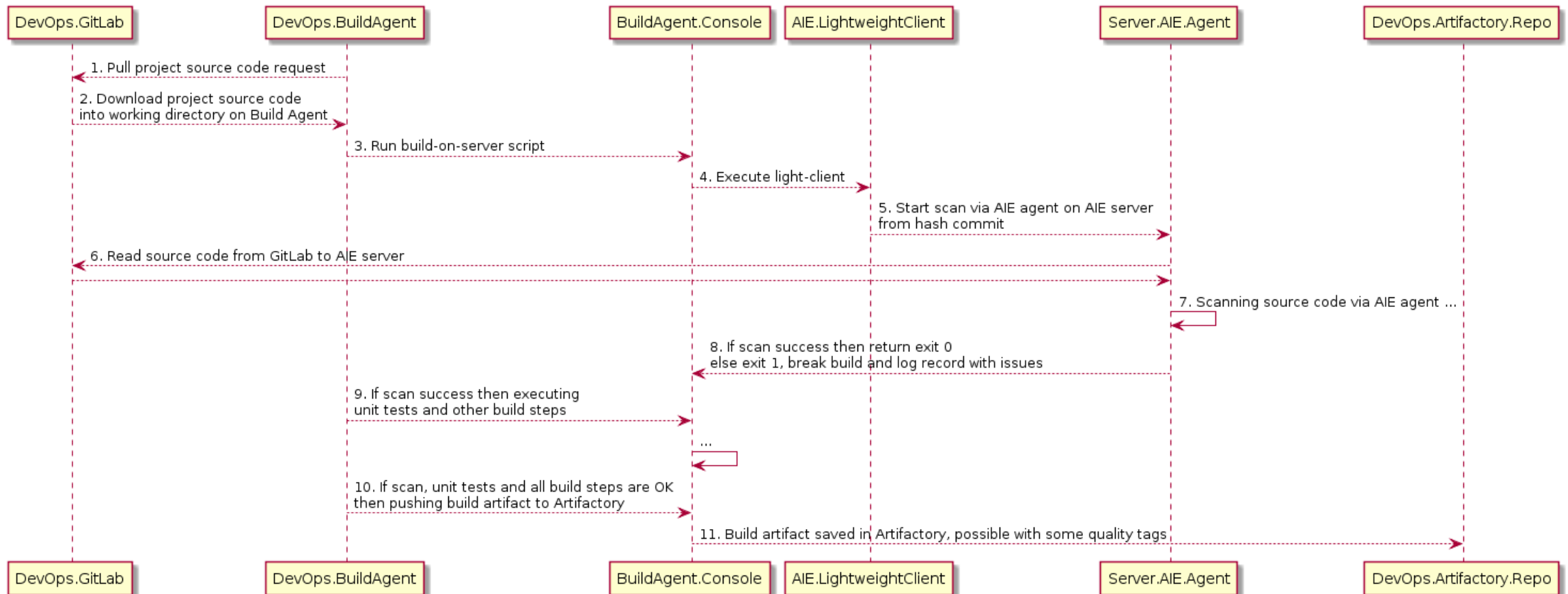


Легенда

- **DevOps.BuildAgent**
сборочный агент (Linux или Windows)
- **BuildAgent.Console**
системная консоль сборочного агента
- **Server.AIE.Agent**
агент сканирования (PT Application Inspector)
- **DevOps.GitLab**
корпоративное хранилище кода
- **DevOps.GitLab-CI**
корпоративная CI-система
- **DevOps.Artifactory**
корпоративное хранилище артефактов (артефакторий)
- **Docker.Registry**
хранилище докер-образов в артефактории
- **DevOps.Artifactory.Repo**
репозиторий в артефактории для хранения бинарных файлов, выгружаемых туда после сборки
- **Docker.Build.AISA**
артефакт сборки клиента AISA (докер-образ)
- **Docker.Windows/Linux.AISA-client.Latest/TAG**
все докер-образы клиента AISA (Linux и Windows)
- **Docker.Windows/Linux.AISA-client**
выбранный докер-образ, «обертка» над клиентом AISA
- **AIE.LightweightClient**
легковесный клиент AISA внутри докер-контейнера для работы с API сервера PT Application Inspector

Сборочный процесс с использованием PT Application Inspector

PT



Основные этапы методики

01

Подготовка серверной части

- Установка и настройка сервера на отдельной виртуальной машине
- Установка и настройка агентов сканирования

02

Подготовка клиентской части

- Организация релизного CI-цикла для клиента AISA (поставка докер-образом)

03

Подготовка проекта сканирования

- На стороне сервера
- Через клиент AISA

04

Работа с CI-системами

- Шаблоны сканирования в GitLab CI
- Метараннеры TeamCity
- Прочие средства



Компоненты решения

Серверная, клиентская и SI-части



Дмитрий Рагулин

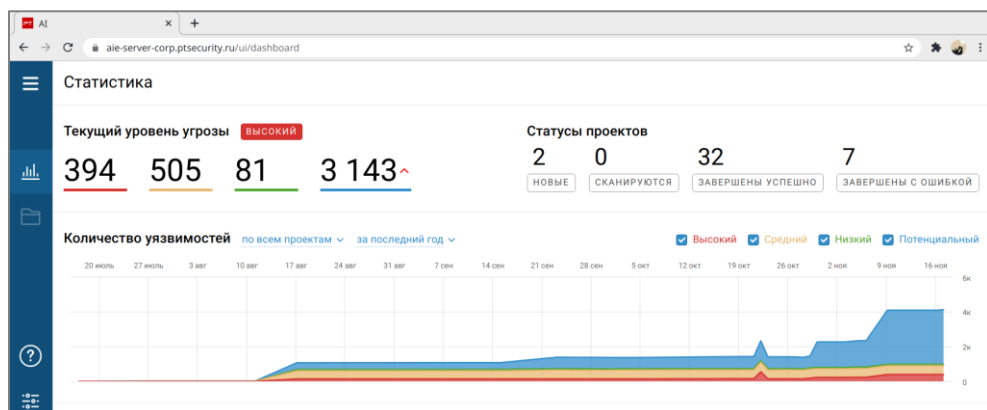
SI-инженер отдела технологий и процессов разработки

ptsecurity.com

Серверная часть PT Application Inspector

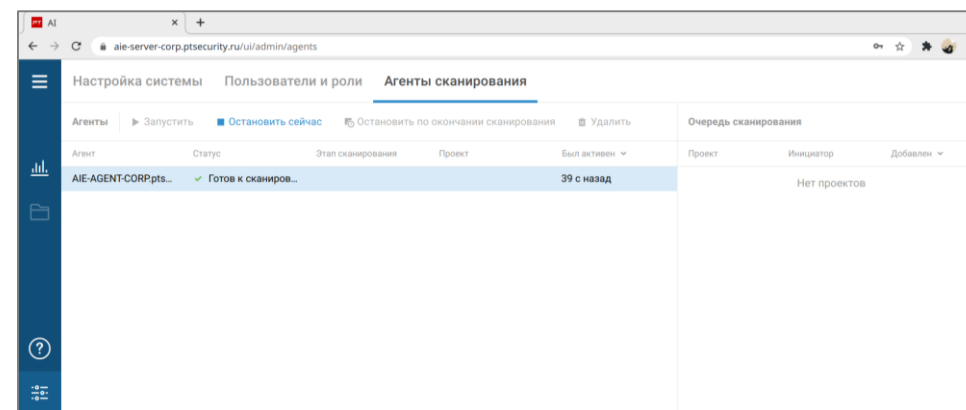
PT

PT Application Inspector Server



Процессор: Intel Core i7 с частотой 3,2 ГГц или аналоги
Оперативная память: От 8 ГБ
Жесткий диск: от 200 ГБ
Сетевой адаптер: от 10 Мбит/с
ОС: 64-разрядная версия Windows Server 2012 R2 и выше

PT Application Inspector Agent



Процессор: Intel Core i7 с частотой 3,2 ГГц или аналоги
Оперативная память: от 8 ГБ
Сетевой адаптер: от 10 Мбит/с
Браузер: Microsoft Edge, Mozilla Firefox 46 и выше,
Google Chrome 50 и выше

Скрипты для развертывания сервера PT Application Inspector

https://github.com/devopshq/dohq-ai-best-practices/tree/master/AIE_Server_installation

Релизный CI-цикл для AISA

I

III

II

if tests OK

DevOps AI.Shell.Docker Supply Project <Active branches>

Проект для сборки и сопровождения в DevOps инфраструктуре клиентской части AISA (AI.Shell проект). Admin: DevOps, подробнее: <https://wiki.ptsecurity.com/display/DevOps/Application+Inspector+Enterprise+as+PT+Service>

Overview Change Log Statistics Current Problems Investigations Muted Problems Build Chains Flaky Tests 0

Hide successful configurations

Builds

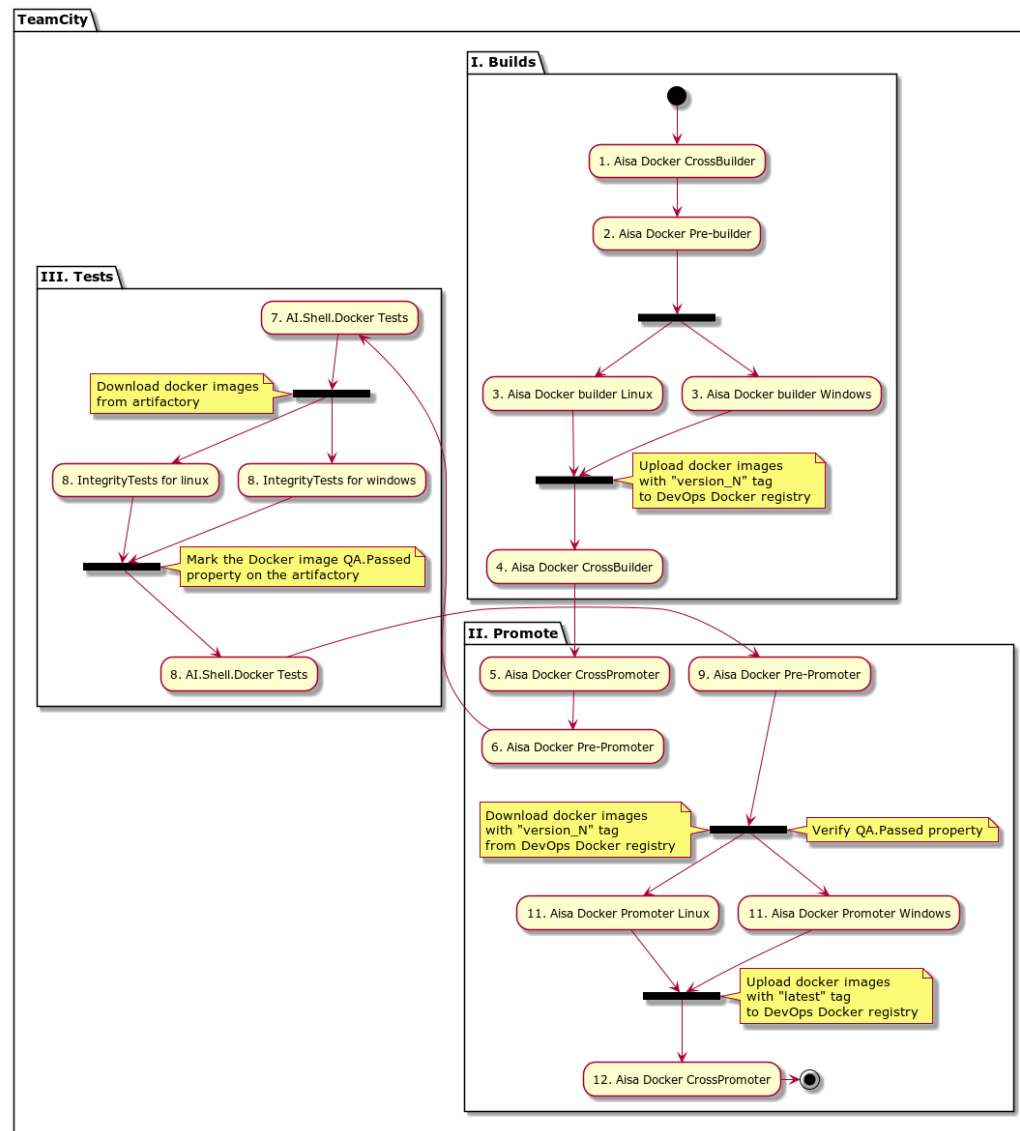
- Aisa Docker CrossBuilder
- Docker builds
 - Aisa Docker builder Linux
 - Aisa Docker builder Windows
 - Aisa Docker Pre-builder

Tests Automatic tests. Admin: DevOps

- AI.Shell.Docker Tests
 - Tests Linux
 - [IntegrityTest] Docker aisa-windows create project
 - [IntegrityTest] Docker aisa-windows create project with policy
 - [IntegrityTest] Docker aisa-windows create project and quick scan
 - [IntegrityTest] Docker aisa-windows create project and long scan
 - [IntegrityTest] Docker aisa-windows quick scan
 - [IntegrityTest] Docker aisa-windows long scan
 - [IntegrityTest] Docker aisa-windows get multi-report
 - Tests Windows

Tools

- Aisa Docker promoter
 - Aisa Docker Cross Promoter
 - Promoter builds
 - Aisa Docker Pre-Promoter
 - Aisa Docker Promoter Linux
 - Aisa Docker Promoter Windows



Клиентская часть

PT Application Inspector AISA client



```
$ aisa --version
3.6.1.4931

$ aisa --project-settings-file $CI_PROJECT_PATH_SLUG.aiproj --scan-target ./ --reports "HTML,JSON" --reports-folder ".report"
Work started
```

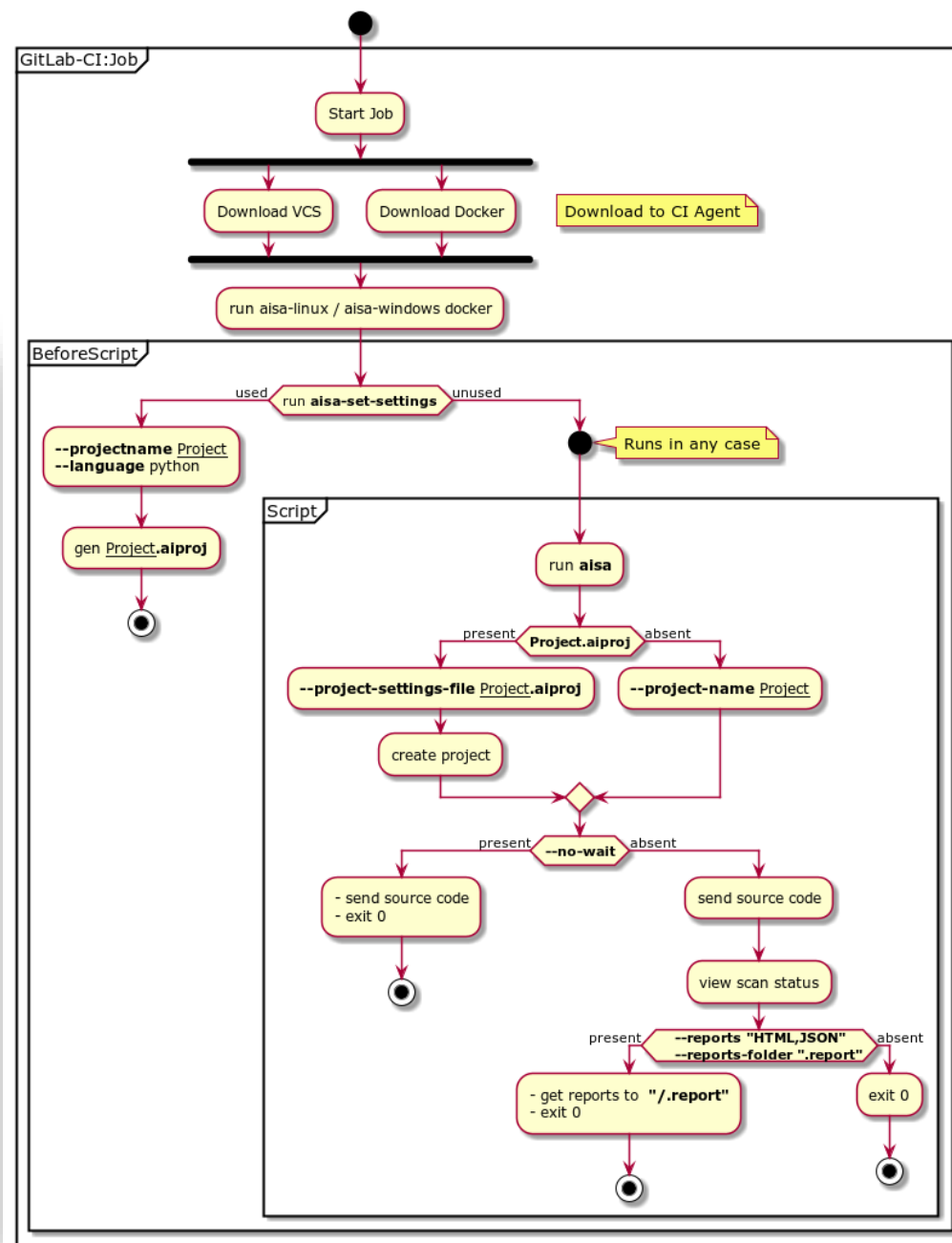
Параметр запуска	Пример значения	Описание	Обязательный параметр?
--project-name	DevOps_Sandbox	Название проекта (регистронезависимо), который надо просканировать. Проект должен быть создан на сервере AIE на момент запуска сканирования	Да, при отсутствии ключа --project-settings-file
--project-settings-file	Test.aiproj	Путь к файлу с параметрами проекта	Да, при отсутствии ключа --project-name
--policies-path	./policy.json	Путь к файлу с описанием политик	Нет
--scan-target	source/folder	Путь к каталогу или файлу приложения для сканирования	Да
--reports-folder	.ptai	Путь к каталогу, куда будут сохранены файлы отчетов	Нет
--reports	"HTML,JSON"	Типы отчетов, создаваемых по окончании сканирования. Может принимать несколько значений через запятую: HTML, PDF, JSON, WAF	Нет
--no-wait		Отключает ожидание результатов сканирования, значительно сокращает время работы	Нет
--scan-off		Отключает сканирование на сервере AIE, используется для создания проекта (в паре с аргументом --project-settings-file)	Нет

* Поддерживаемые языки сканирования: Java, PHP, C#, Objective-C, C++, Visual Basic, SQL, Swift, Python, JavaScript, Kotlin, Go.

Реализация

Пример для GitLab CI

```
1 # (c) DevOpsHQ, 2020
2
3 Start AIE scan:
4   stage: scan
5   image: [your-repo]/aisa-linux:latest
6   before_script:
7     - aisa-set-settings
8     --projectname $CI_PROJECT_PATH_SLUG
9     --language $PROJECT_LANG
10  script:
11    - aisa --version
12    - aisa
13      --project-settings-file $CI_PROJECT_PATH_SLUG.aiproj
14      --scan-target ./
15      --reports "HTML,JSON"
16      --reports-folder ".report"
17    - aisa-codequality -i .report -o codequality.json
18  allow_failure: true
19  only:
20    variables:
21      - $PROJECT_LANG
22  artifacts:
23    expire_in: 14 day
24    paths:
25      - .report/
26
```



Подготовка проекта к сканированию

Файл конфигурации ai-project-settings

```
{
  // Main Settings
  "ProjectName": "Test_Proj", // Имя проекта
  "ProgrammingLanguage": "Csharp", // Язык приложения.
  // Доступны: Java, Php, Csharp, Vb, ObjectiveC, Cplusplus, Sql, Swift, Python, JavaScript, Kotlin, Go
  "ScanAppType": "CSharp , Configuration, Fingerprint, PmTaint ", // Модули поиска уязвимостей.
  // Доступны: Php, Java, CSharp, JavaScript, Configuration, Fingerprint (включает в себя DependencyCheck), PmTaint, BlackBox

  "ThreadCount": 1, // Количество потоков
  "Site": "http://localhost", // Адрес сайта
  "IsDownloadDependencies": true, // Загрузить зависимости

  "IsUsePublicAnalysisMethod": false, // Искать от доступных public и protected методов
  "IsUseEntryAnalysisPoint": true, // Искать от точек входа

  "ScanUnitTimeout": 600, // Максимальное время сканирования файла в seconds
  "PreprocessingTimeout": 60, // Таймаут препроцессинга в minutes
  "CustomParameters": null, // Дополнительные параметры запуска

  // C# Parameters
  "ProjectType": "Solution", // Типа проекта. Доступны: Solution, Website
  "SolutionFile": "path_to_solution.sln", // Путь к файлу решения/проекта
  "WebSiteFolder": "path_to_website", // Директория сайта

  //JavaScript Parameters
  "JavaScriptProjectFile": "path_to_file", // Путь к файлу скрипта
  "JavaScriptProjectFolder": "path_to_dir", // Путь к корневой папке проекта javascript

  //PMTaint Parameters
  "UseTaintAnalysis": false, // Использовать ли движок AI.Taint для анализа
  "UsePmAnalysis": true, // Использовать ли движок PT.PM для анализа
  "DisableInterpretCores": false, // Игнорировать ли ядра интерпретации (C#, Java, PHP) при анализе

  // YARA Parameters
  "UseDefaultFingerprints": true, // Использовать предустановленные фингерпринты
  "UseCustomYaraRules": false, // Использовать пользовательские правила YARA (сами правила задаются только через базу в UI)
```

Тиражирование конфигураций сканирования:

- генерируем в момент сборки,
- сохраняем в Git-репозитории конкретного проекта,
- сохраняем в служебном Git-репозитории

Примеры типовых шаблонов

```
1 # (c) DevOpsHQ, 2020
2
3 # This template is used for projects that were generated in advance.
4
5 Start AIE scan:
6   image: [your-repo]/aisa-linux:latest
7   script:
8     - aisa --version
9     - aisa
10     --project-name $CI_PROJECT_PATH_SLUG
11     --scan-target ./
12     --reports "HTML,JSON"
13     --reports-folder ".report"
14   allow_failure: true
15   artifacts:
16     expire_in: 3 day
17     paths:
18     - .report/
```

existing_project.yml

yam... 100% 18/18 ln :1

```
1 # (c) DevOpsHQ, 2020
2
3 # This template is used for projects that were not generated in advance.
4
5 Start AIE scan:
6   image: [your-repo]/aisa-linux:latest
7   before_script:
8     - aisa-set-settings
9     --projectname $CI_PROJECT_PATH_SLUG
10    --language $PROJECT_LANG
11   script:
12     - aisa --version
13     - aisa
14     --project-settings-file $CI_PROJECT_PATH_SLUG.aiproj
15     --scan-target ./
16     --reports "HTML,JSON"
17     --reports-folder ".report"
18   allow_failure: true
19   only:
20     variables:
21     - $PROJECT_LANG
22   artifacts:
23     expire_in: 14 day
24     paths:
25     - .report/
```

non-existent_project.yml

Пример продуктового шаблона



```
1 Start AIE scan:
2   stage: test
3   image: docker.<repo>.ru/aisa-linux:latest
4   before_script:
5     - aisa-set-settings
6       --projectname $CI_PROJECT_PATH_SLUG
7       --language $PROJECT_LANG
8   script:
9     - aisa --version
10    - aisa
11      --project-settings-file $CI_PROJECT_PATH_SLUG.aiproj
12      --scan-target ./
13      --reports "HTML,JSON"
14      --reports-folder ".report"
15    - aisa-codequality -i .report -o codequality.json
16 allow_failure: true
17 only:
18   variables:
19     - $PROJECT_LANG
20 artifacts:
21   expire_in: 14 day
22   paths:
23     - .report/
24   reports:
25     codequality: codequality.json
26
```

Продуктовые шаблоны:

- Создаются под нужды конкретной команды
- Широкие возможности кастомизации
- Команда может вносить правки самостоятельно через MR

Примеры подключения проекта на сканирование



Журналы сканирования в GitLab CI

```
20 $ aisa-set-settings --projectname SCI_PROJECT_PATH_SLUG --language $PROJECT_LANG
21 [INFO]: Start generating project
22 [INFO]: Arguments is correct
23 -----
24 [INFO]: Project name: dragulin-sandbox
25 [INFO]: File name:    dragulin-sandbox.aiproj
26 [INFO]: Language:    python
27 [INFO]: SLN:         null
28 [INFO]: Source:      ./
29 [INFO]: Path:        ./
30 -----
31 [INFO]: File dragulin-sandbox.aiproj was generated successfully
32 $ aisa --version
33 3.6.1.4931
34 $ aisa --project-settings-file SCI_PROJECT_PATH_SLUG.aiproj --scan-target ./ --reports "HTML,JSON" --reports-folder ".report"
35 Work started
36 Loading project dragulin-sandbox
37 Upload sources 0%
38 Upload sources 1%
```

TeamCity-метараннеры

Build Step

Runner type:

Meta-Runner: Aisa scan (Linux) ▼
Meta-runner for run Application Inspector Shell Agent in linux Docker.
Note: The [Meta-Runner: Aisa scan \(Linux\)](#) meta-runner is defined in the [<Root project>](#) project.

Step name:

Optional, specify to distinguish this build step from other steps.

AISA Args:*

Please specify Arguments for Application Inspector Shell Agent

Aisa docker name:*

Please specify docker image name*

Docker Registry:*

Please specify Aisa docker registry*

Docker login:*

Please specify login for docker registry

Docker password:*

Please specify password for your docker registry account

TeamCity-метараннеры под Windows и Linux

https://github.com/devopshq/dohq-ai-best-practices/tree/master/TeamCity_meta-runners

Журналы сканирования в TeamCity

PT

Tree view | Tail

Download full build log (~12.65 KB) | .zip

View: All messages Console view Repeat block names

```
[19:22:08] The build is removed from the queue to be prepared for the start
[19:22:08] ▶ Collecting changes in 1 VCS root
[19:22:08] Starting the build on the agent linux-ci-1
[19:22:08] Clearing temporary directory: /home/teamcity/build_disk/data/temp/buildTmp
[19:22:08] ▶ Publishing internal artifacts (4s)
[19:22:08] Using vcs information from agent file: 1998c77e40927187.xml
[19:22:08] Checkout directory: /home/teamcity/build_disk/data/work/1998c77e40927187
[19:22:08] ▶ Updating sources: auto checkout (on agent) (1s)
[19:22:10] ▼ Step 1/1: Meta-Runner: Aisa scan (Linux) (11s)
[19:22:10]   ▼ [Step 1/1] Step 1/1: Application Inspector Shell Agent (Python) (11s)
[19:22:10]     [Step 1/1] Starting: /usr/bin/python3.5 .script.py
[19:22:10]     [Step 1/1] in directory: /home/teamcity/build_disk/data/work/1998c77e40927187
[19:22:21]     ▼ [Step 1/1] Aisa running
[19:22:21]       ▶ [Aisa running] Aisa: Prepare tasks
[19:22:21]       ▶ [Aisa running] Aisa: Pull docker image
[19:22:21]       ▶ [Aisa running] Aisa: Prepare container
[19:22:21]       ▼ [Aisa running] Aisa: Scan project
[19:22:21]         [Aisa: Scan project]
[19:22:21]         [Aisa: Scan project] Container 1605284530000 has started, reading the log ...
[19:22:21]         [Aisa: Scan project] -----
[19:22:21]         [Aisa: Scan project] Aisa version: 3.6.1.4931
[19:22:21]         [Aisa: Scan project] -----
[19:22:21]         [Aisa: Scan project] Work started
[19:22:21]         [Aisa: Scan project] Loading project DevOps_Sandbox
[19:22:21]         [Aisa: Scan project] Upload sources 0%
[19:22:21]         [Aisa: Scan project] Upload sources 1%
[19:22:21]         [Aisa: Scan project] Upload sources 2%
[19:22:21]         [Aisa: Scan project] Upload sources 4%
[19:22:21]         [Aisa: Scan project] Upload sources 6%
[19:22:21]         [Aisa: Scan project] Upload sources 8%
```



Демонстрация



Дмитрий Рагулин

СІ-инженер отдела технологий и процессов разработки

ptsecurity.com

Open source



README.md 21.7 KB

`</>` `📄` `Edit` `Web IDE` `Replace` `Delete` `🔍` `📁` `📄` `📄`

DevSecOps: внедрение в продуктовый конвейер и эксплуатация PT Application Inspector

Узнать подробнее о внедрении PT Application Inspector можно из вебинара: <https://www.ptsecurity.com/ru-ru/research/webinar/devsecops-vnedrenie-v-produktovyy-konvejer-i-ehkspluatatsiya-pt-application-inspector/>

Инструкция ниже, методика и рекомендации актуальны для AIE сервера v.3.6.1

Table of Contents

1. Для чего нужен PT Application Inspector?
2. Методика внедрения PT AI в CI
3. Описание сборочной CI-инфраструктуры
4. Краткая инструкция по развертыванию AIE сервера
 - Системные требования для установки сервера
 - Автоматизация установки средствами PowerShell
5. Цикл релизной сборки Application Inspector Shell Agent (AISA) в Docker
 - Инструкция по сборке Docker контейнеров
6. Инструкция по эксплуатации Docker образов с AISA
 - Параметризация проекта
7. Инструкция по использованию метараннеров в TeamCity
8. Инструкция по использованию шаблонов в GitLab CI
9. Содержимое репозитория

Для чего нужен PT Application Inspector?

PT Application Inspector — удобный инструмент для выявления уязвимостей и ошибок в приложениях, поддерживающий процесс безопасной разработки.

PT Application Inspector выделяется среди конкурентов исключительной точностью результатов благодаря сочетанию ключевых методов анализа с уникальной технологией абстрактной интерпретации. PT AI позволяет специалистам по ИБ выявлять и подтверждать уязвимости и признаки НДВ, например закладок, оставленных в исходном коде разработчиками или хакерами, а разработчикам — ускорить исправление кода на ранних стадиях разработки.

Список поддерживаемых языков: `java`, `php`, `c#`, `vb`, `objective-c`, `c++`, `sql`, `swift`, `python`, `javascript`, `go`

Подробнее о продукте: <https://www.ptsecurity.com/ru-ru/products/ai/>

Методика внедрения PT AI в CI

Основные этапы методики внедрения PT AI в CI:

1. Подготовка серверной части
2. Подготовка клиентской части в докере
3. Подготовка проекта сканирования на стороне сервера или через клиент AISA
4. Работа с шаблонами сканирования в GitLab-CI, метараннерами для сканирования в TeamCity или иными средствами CI-систем

Проект доступен на нашем [GitHub](https://github.com/devopshq/dohq-ai-best-practices):
github.com/devopshq/dohq-ai-best-practices

- Рекомендуемая методика внедрения
- Скрипты инсталляции и рекомендации по настройке серверной части PT Application Inspector
- Dockerfile для сборки AISA-клиента (Windows и Linux)
- Шаблоны шагов запуска сканирования через PT Application Inspector:
 - job для GitLab CI,
 - метараннеры для TeamCity



Заключение



Алексей Жуков

Эксперт отдела систем защиты приложений

ptsecurity.com

Как «пропилотировать» PT Application Inspector

PT



Заполните
заявку на странице
PT Application Inspector

[на сайте](#)

или свяжитесь
с вашим менеджером
в Positive Technologies
или [у партнеров](#) компании



Подписание NDA,
заполнение анкеты
о приложениях



Развертывание PT AI:
установка, настройка,
подключение приложений



Пилотный проект
при поддержке экспертов
Positive Technologies



Отчет
о найденных
уязвимостях

Презентация
результатов

≈ 4 недели



POSITIVE
TECHNOLOGIES

Подробнее

ptsecurity.com/ru-ru/products/ai/

github.com/devopshq/dohq-ai-best-practices

АЛЕКСЕЙ ЖУКОВ

alzhukov@ptsecurity.com

ТИМУР ГИЛЬМУЛЛИН

tgilmullin@ptsecurity.com

ДМИТРИЙ РАГУЛИН

dragulin@ptsecurity.com



@BigAppSec

ptsecurity.com