2 ИЮНЯ 2026, МОСКВА, ЛОФТ ГОЭЛРО

БЕКОН LUNTRY '26

ЕДИНСТВЕННАЯ КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

ФСТЭК и контейнеры: от заявки до сертификата

Андрей Слепых | Фобос-НТ, Анатолий Карпенко | LUNTRY

БЕКОН'26

ФСТЭК и контейнеры: от заявки до сертификата

Андрей Слепых (Фобос-НТ)

Анатолий Карпенко (Luntry)

whoami

- Автоматизатор автоматизации в Luntry
- Любитель митапошных форматов
- ITGM, TechTrain, DevOops, DEFCON'ы, SafeCode, БЕКОН, ТБ Форум, UDW, DUMP, CodeFest
- Веду канал «Технологический Болт Генона»
- Рисую несмешные мемы
- tg: @rusdacent

whoami

- Ведущий эксперт по сертификации в НТЦ "Фобос-НТ"
- Занимаюсь автоматизацией практик РБПО
- БЕКОН, ТБ Форум, DevOops
- Мастерю в D&D
- tg: @SlepoyPew

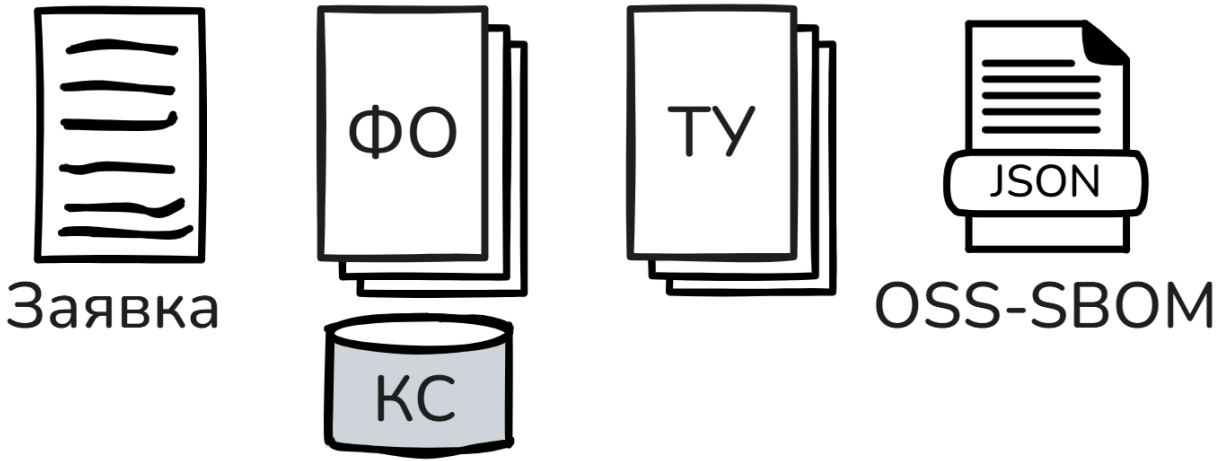
План

- Подача заявки: Ожидание/Реальность
- Подготовка ПиМ: Ожидание/Реальность
- Составление Протоколов: Ожидание/Реальность
- Заключение
- Бонус

БЕКОН'26

Подача заявки: Ожидание

Подача заявки. Ожидание.
Состав заявки на конец 2024 года



Источник

Подача заявки. Ожидание.
Состав заявки на середину 2026 года

от 20 января 2026 г. № 9

ИЗМЕНЕНИЯ,
которые вносятся в Положение о системе сертификации средств защиты информации, утвержденное приказом ФСТЭК России от 3 апреля 2018 г. № 55

1. Пункт 21 изложить в следующей редакции:

«21. К заявке на сертификацию прилагаются следующие документы:

технические условия в двух экземплярах;

техническое задание в двух экземплярах (в случае, если планируется проведение сертификации средства защиты информации на соответствие требованиям по безопасности информации, изложенным в техническом задании);

задание по безопасности в двух экземплярах (в случае необходимости его разработки в соответствии с требованиями по безопасности информации);

формуляр (паспорт) на средство защиты информации;

перечень заимствованных программных компонентов с открытым исходным кодом, входящих в состав средства защиты информации (в случае наличия в средстве защиты информации заимствованных программных компонентов с открытым исходным кодом);

перечень образов контейнеров, входящих в состав средства защиты информации (в случае наличия в средстве защиты информации образов контейнеров);

Источник

Подача заявки. Ожидание.

БЕКОН'26

OSS-SBOM (ИС от 26 сентября 2024 г. № 240/24/4436)

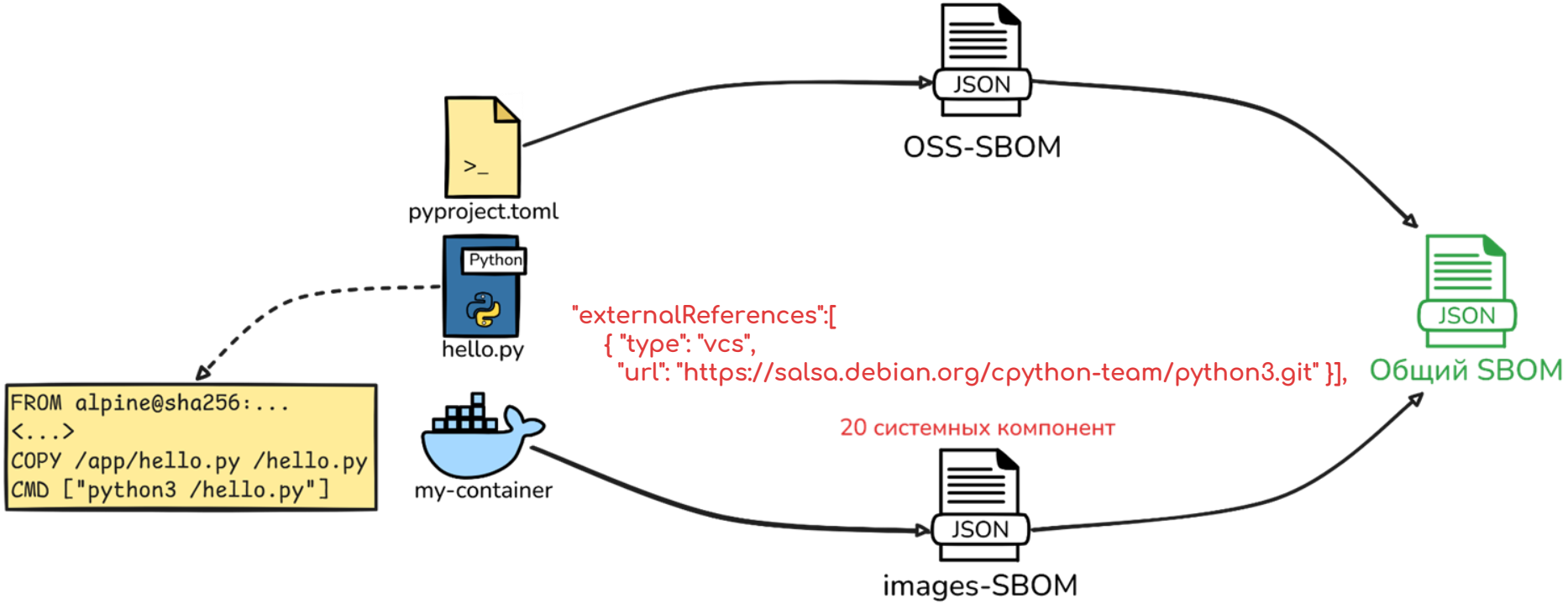
```
{  "bomFormat": "CycloneDX",
   "specVersion": "1.6",
   "version": 1,
   "metadata": {
      "timestamp": "2026-06-02T00:00:00",
      "component": {
         "type": "application",
         "name": "САМЫЙ ЛУЧШИЙ СКАНЕР", "version": "v.2.0.26",
         "manufacturer": { "name": "БЕКОН" } } },
   "components": [ {
      "type": "library",
      "name": "python3",
      "version": "3.12.1",
      "externalReferences": [
         { "type": "vcs",
         "url": "https://salsa.debian.org/cpython-team/python3.git" }],
      "properties":
         [ { "name": "GOST:attack_surface", "value": "yes" },
           { "name": "GOST:secutity_function", "value": "yes" } ]
   } ] }
```

images-SBOM (ИС от 13 января 2025 г. № 240/24/38)

```
{  "bomFormat": "CycloneDX",
   "specVersion": "1.6",
   "version": 1,
   "metadata": {
      "timestamp": "2026-06-02T00:00:00",
      "component": {
         "type": "application",
         "name": "САМЫЙ ЛУЧШИЙ СКАНЕР", "version": "v.2.0.26",
         "manufacturer": { "name": "БЕКОН" } } },
   "components": [ {
      "type": "container",
      "name": "my-container",
      "version": "1.0",
      "properties":
         [ { "name": "GOST:attack_surface", "value": "yes" },
           { "name": "GOST:secutity_function", "value": "yes" } ],
      "components": [ {
         "type": "container",
         "name": "python3",
         "version": "3.12.1",
         "properties":
            [ { "name": "GOST:attack_surface", "value": "yes" },
              { "name": "GOST:secutity_function", "value": "yes" } ]
   } ] }
```

Источник требования (п.2)

Подача заявки. Ожидание.
Проблема объединения разных SBOM-ов в один общий



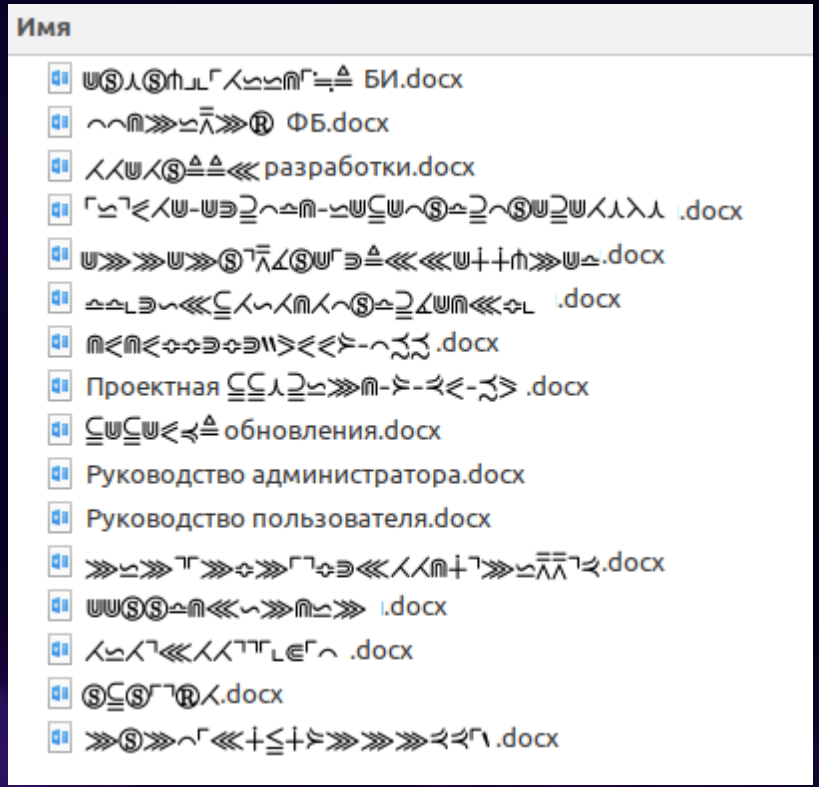
БЕКОН'26

Подача заявки: Реальность

Документация

БЕКОН'26

- Сначала ты ожидаешь, что придётся писать много документации
- …но потом понимаешь, что её ещё больше надо написать
- А ещё ничего не понятно, что требуется конкретно, не понятна терминология, не понятно вообще с каких документов начинать сначала
- … а ещё *.docx :)



Источник

БЕКОН'26

DocOps

КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

LaTeX

БЕКОН'26

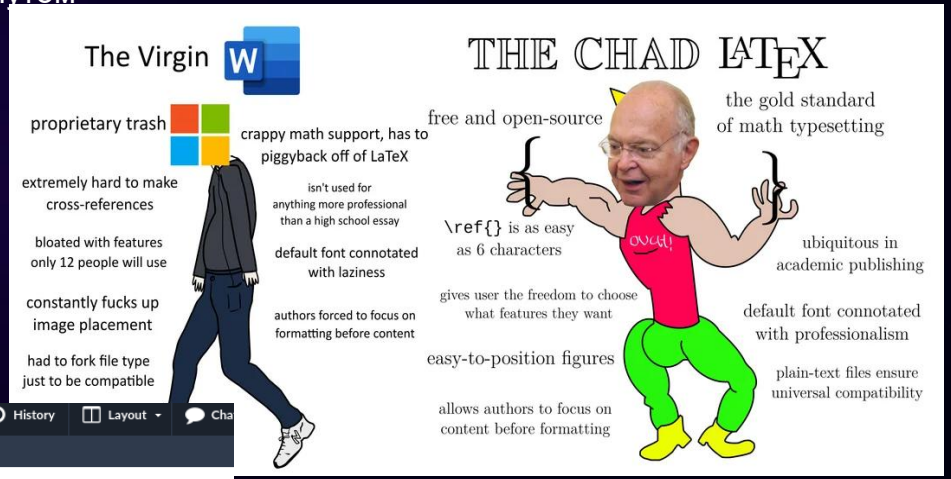
- Система вёрстки TeX создана в 1977 году Дональдом Кнутом
- LaTeX - макропакет для TeX
- TeX Live - дистрибутив LaTeX

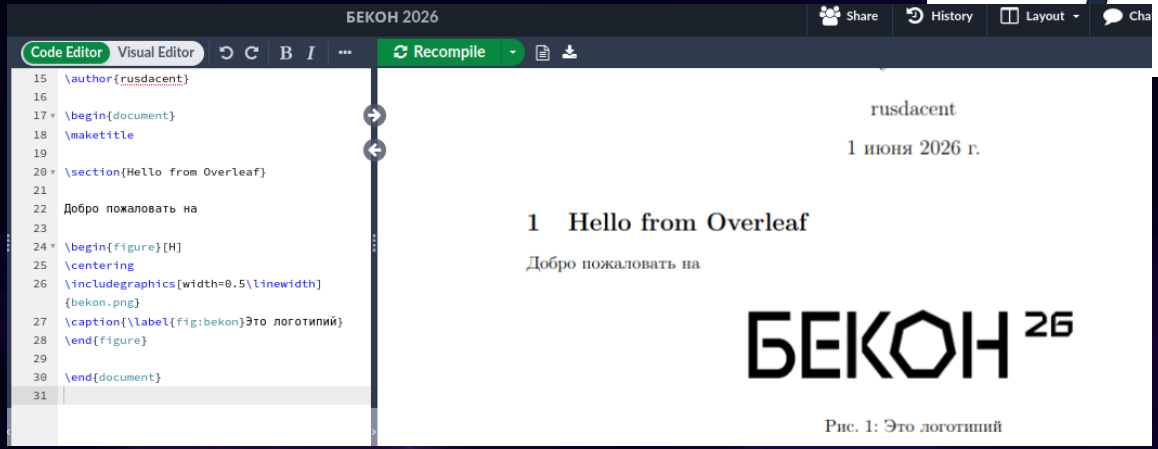
Сплошные плюсы

- Лежит в GitLab
- Нормальные diff-ы
- Готовые пакеты для всего
- Готовые шаблоны для всего

Сплошной минус

- Высокий порог входа





Оцениваем существующие сервисы

БЕКОН'26

Стек “ДО”

- JS/TS
- Python
- Go
- Rust

Стек “ПОСЛЕ”

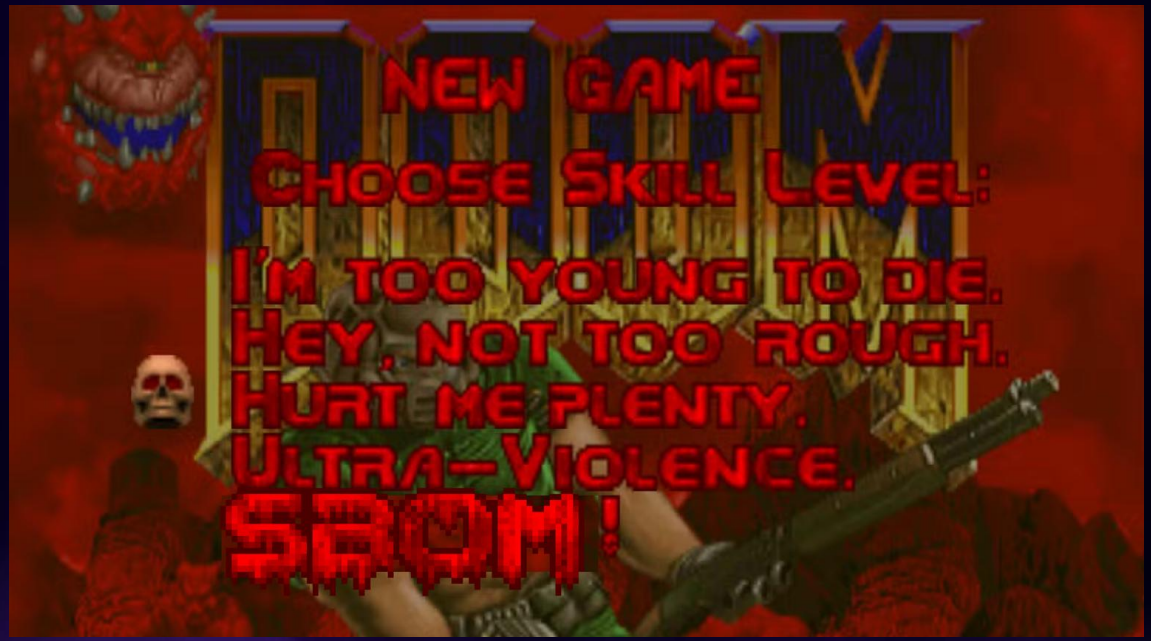
- JS/TS
- ~~Python~~
- Go
- Rust

Два интерпретатора, которые надо исследовать, это “весело”, поэтому избавляемся от Python…

…, но NodeJS остался

Читаем про SBOM (ППК) на код

- Первичная оценка ПА показала нам, что наша архитектура была слишком “развесистой”
- Проанализировали архитектуру и минимизировали компонентную базу



Читаем про SBOM (ППК) на контейнеры

БЕКОН'26

- Необходим всем участникам процесса сертификации для понимания архитектуры продукта
- Позволяет увидеть где можно и нужно провести оптимизацию окружения (образа)
- Этот SBOM проще, чем на зависимости

[Источник требования (пп. 2-3)]

Генерируем SBOM-ы

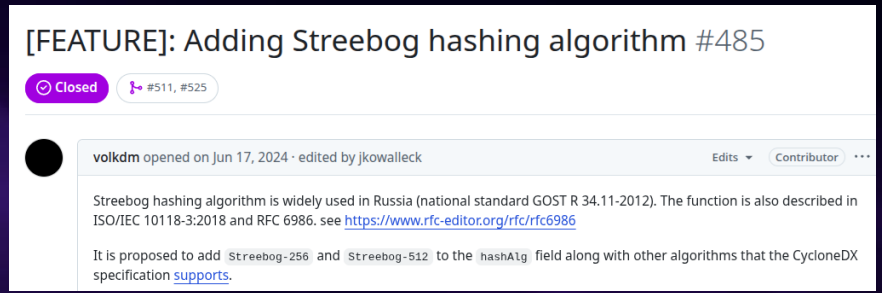
Начинали с Trivy для всего, но стало понятно, что получаем неполные SBOM-ы

- Для Go перешли на cyclonedx-gomod
- Для Rust оставили Trivy (далее перейдём на cargo sbom)
- JS остался на Trivy
- Для контейнеров используем Luntry

BONUS

21 октября 2025 года релизнулся CycloneDX 1.7

Завезли поддержку Стрибога



Результат (получили Решение)

БЕКОН'26

Федеральная служба по техническому и экспортному контролю

РЕШЕНИЕ
о проведении сертификации средства защиты информации

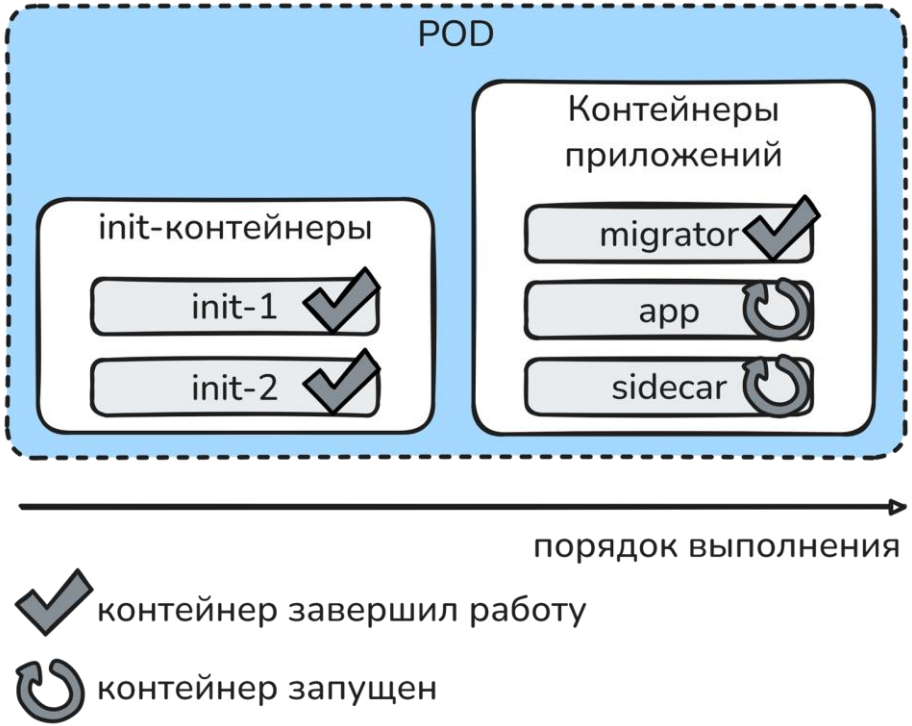
№ [redacted]

Наименование средства защиты информации:	программное обеспечение Luntry (Лантри) Certified
Назначение средства защиты информации:	для защиты информации, не содержащей сведения, составляющие государственную тайну, в ГИС 1 класса защищенности, в АСУ ТП 1 класса защищенности, для обеспечения безопасности персональных данных 1 уровня защищенности, на значимых объектах КИИ 1 категории

БЕКОН'26

Подготовка ПиМ: Ожидание

Подготовка ПиМ: Ожидание. Документация. Описание архитектуры безопасности



Описание архитектуры безопасности

- процесс инициализации
- описание временных сервисов

Источник требования (пп. 5-6)

Подготовка ПиМ: Ожидание. Документация. Описание архитектуры безопасности

БЕКОН'26

Обеспечение собственной защиты средства от несанкционированного доступа

```
#include <tunables/global>

profile k8s-apparmor-example-deny-write flags=(attach_disconnected) {
  #include <abstractions/base>

  file,

  # Deny all file writes.
  deny /** w,
}
```

AppArmor профиль

```
{
    "defaultAction": "SCMP_ACT_ERRNO",
    "architectures": [
        "SCMP_ARCH_X86_64",
        "SCMP_ARCH_X86",
        "SCMP_ARCH_X32"
    ],
    "syscalls": [
        {
            "names": [
                "accept4",
                "epoll_wait",
                "pselect6",
                "futex",
                "madvise",
                "epoll_ctl",
                "getsockname",
                "setsockopt",
                "vfork",
                "mmap",
```

seccomp профиль

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
```

NetworkPolicy

Источник требования (пп. 5-6)

Подготовка ПиМ: Ожидание.
Запрос Плана испытаний заимствованных компонент

Требования по безопасности (Приказ №76)

16.1. Формуляр средства должен содержать **контрольные суммы дистрибутива и исполняемых файлов программного обеспечения**, которые должны уточняться при обновлении средства в соответствии с настоящими Требованиями, а также состав аппаратной платформы средства.

- Контроль за исходным кодом -> составление SBOM по исходному коду
- Контроль за исходным кодом -> составление SBOM по образам
- Расчет КС **всех исполняемых файлов** (ELF-файлы, .py-скрипты, .jar-архивы, .js-скрипты)

Методические рекомендации ФСТЭК и ИС от 13 января 2025 г. № 240/24/38

- Объединение 2 типов SBOM в **общий SBOM на СЗИ**
- Обогащение GOST-овыми полями и ссылками для externalReferences
- Формальная верификация с помощью sbom-checker

В случае расхождения перечня исполняемых файлов и компонент из общего SBOM могут быть выданы замечания на доработку SBOM с комментарием:

Не обеспечена полнота перечня программных компонентов

Доклад про ИС от 13 января 2025 г. № 240/24/38

Подготовка ПиМ: Ожидание.
Какой базовый образ использовать?

БЕКОН'26

Lawful Good	Neutral Good	Chaotic Good
Lawful Neutral	True Neutral	Chaotic Neutral
Lawful Evil	Neutral Evil	Chaotic Evil

Подготовка ПиМ: Ожидание.
Какой базовый образ использовать?

БЕКОН'26

Lawful Good **FROM scratch**	Neutral Good **СертOS** **distroless**	Chaotic Good  **distroless**
Lawful Neutral **СертOS** **minimal**	True Neutral  **minimal**  **minimal**	Chaotic Neutral
Lawful Evil **СертOS** **FULL distro**	Neutral Evil  **FULL distro**	Chaotic Evil 

Подготовка ПиМ: Ожидание.
Подбор инструментов для проведения анализа контейнеров

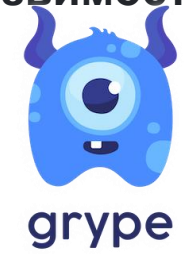
Архитектурный анализ [1]

checkov

SECCOMPARE

capsh

Компонентный анализ (SCA) и анализ уязвимостей



Статический анализ (SAST) [2]

Динамический анализ (unit и фаззинг) [3]

[1] Три кита РБПО. Кит №3. Архитектурный анализ
[2] Три кита РБПО. Кит №1. Статический анализ
[3] Многоликий динамический анализ

БЕКОН'26

Подготовка ПиМ: Реальность

Центр исследований безопасности системного ПО

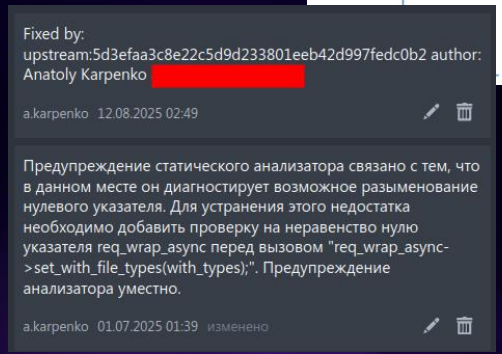
БЕКОН'26

Требуется для NodeJS:

- SAST
- Fuzzing
- Исследование уязвимостей
- …

Зачем это делать одному, если можно не делать одному?

- Сформирован в конце 2023 года на базе ИСП РАН под эгидой ФСТЭК и при поддержке РАН
- Ядро Linux + СПО
- https://portal.linuxtesting.ru/



Результаты деятельности Центра исследований безопасности системного программного обеспечения (Алексей Хорошилов)

Допиливаем SBOM-ы

БЕКОН'26

- Добавление ссылок на заимствованные компоненты (extrenalReferences)
  - sbom-updater

- Разметка ПА и ФБ (тут только “руками”)
  - Написали автоматизацию

- Добавляем информацию о компонентах, которых не было в SBOM
  - Написали автоматизацию

- Добавляем информацию о сервисах
  - Написали автоматизацию

- “Склейка” отдельных SBOM-ов в общий
  - sbom-unifier

- Проверка результата
  - sbom-checker

Считаем КС по ГОСТ для исполняемых файлов

БЕКОН'26

SBOM + КС = ♥


```
{
  "image": "[redacted]/luntry/3dparty/redis:7.2.11",
  "timestamp": "2025-11-10T01:04:37.838964+03:00",
  "image_layer_hash": "GOST R 34.11-2012 (/tmp/tmp.BtHS7XnulU) = s256:2ddcd9eaf3043b03af58ea65d5a
  "components": [
    {
      "path": "/usr/lib/x86_64-linux-gnu/libselinux.so.1",
      "hash": "s256:8035b733ab52ba530c3c330848646711de6d6bf72e70160e4de362f5b2715f03",
      "timestamp": "2025-11-10T01:04:37.933122+03:00"
    }
    ,
    {
      "path": "/usr/lib/x86_64-linux-gnu/libpam_misc.so.0.82.1",
      "hash": "s256:697946344ca2e0757bc9c11aaaddeca7028a2702cd2962ec3d18cc517cca5b67",
      "timestamp": "2025-11-10T01:04:37.937343+03:00"
    }
    ,
    {
      "path": "/usr/lib/x86_64-linux-gnu/libcom_err.so.2.1",
      "hash": "s256:f34a3de72282cbf359b23eb7b8587e080d039fba1243b8ebb5f4ba3d8884f072",
      "timestamp": "2025-11-10T01:04:37.940955+03:00"
    }
    ,
    {
      "path": "/usr/lib/x86_64-linux-gnu/libext2fs.so.2.4",
      "hash": "s256:04fc5204a9f572b5457bf9b0f15665117d272876e66ea96495814977dae2c620",
      "timestamp": "2025-11-10T01:04:37.949130+03:00"
    }
    ,
    {
      "path": "/usr/lib/x86_64-linux-gnu/libudev.so.1.7.5",
      "hash": "s256:f1636b5734525a4fa8185bb669064698a1a82b16be1481cf03368371d71ab8f9",
      "timestamp": "2025-11-10T01:04:37.954711+03:00"
    }
    ,
    {
      "path": "/usr/lib/x86_64-linux-gnu/libuuid.so.1.3.0",
      "hash": "s256:15f7a44b460097be4af0474d949db2e321026be398296e8bc37326d2415b5a8a",
      "timestamp": "2025-11-10T01:04:37.958895+03:00"
    }
    ,
    {
```


[Источник требования (пп. 2-3)]

План испытаний заимствованных компонент с открытым исходным кодом

БЕКОН'26

Первый план (было в ноябре 2025)

- Поверхность атаки - **100**
- Функции безопасности - **27**
- Не обеспечена полнота перечня заимствованных компонент - **МНОГО**
- Не хватало разметки ФБ и ПА сервисов
- Статический анализ - **107**
- Динамический анализ - **89**

Обновлённый план (стало в марте 2026)

- Поверхность атаки - **30**
- Функции безопасности - **27**
- Не обеспечена полнота перечня заимствованных компонент - **МАЛО**
- Статический анализ - **51**
- Динамический анализ - **30**

Статический анализ (SAST)

БЕКОН'26

- Межмодульный анализ
- **Анализируются все модули и open source компоненты реализующие ФБ и лежащие на ПА**
- Если под ЯП нет требуемого анализатора, то используем те, которые доступны (SonarQube, Clippy)
- Подтверждённые проблемы **репортим в апстрим или фиксим самостоятельно**

```go
func (n numTOCEntries) check(t TestingT, r *Reader) {
    if r.toc == nil {
        t.Fatal("nil TOC")
    }
```

Undecided | Unspecified | Undecided | Показать трассу

DEREF_AFTER_NULL After having been compared to a nil value at testutil.go:1665, pointer 'r.toc' is dereferenced at testutil.go:1668.

```go
    if got, want := len(r.toc.Entries), int(n); got != want {
        t.Errorf("got %d TOC entries; want %d", got, want)
    }
    t.Logf("got TOC entries:")
    for i, ent := range r.toc.Entries {
        entj, _ := json.Marshal(ent)
        t.Logf("  [%d]: %s\n", i, entj)
    }
    if t.Failed() {
        t.FailNow()
    }
}

func checks(s ...stargzCheck) []stargzCheck { return s }
```

SonarQube Issues Report

Generated: 2026-02-18 19:45:08 UTC
Project: EDR_src_visualizer | Statuses: OPEN,CONFIRMED

Summary (1219 issues)

Severity	Count
BLOCKER	0
CRITICAL	0
MAJOR	318
MINOR	880
INFO	21

MAJOR	src/components/blocks/imageDetailsComponents/sbomReport/view.tsx	58	typescript:S6479

```
55 |             /&gt;
56 |         )}
57 |         {props.sbomReport.map((report, key) =&gt; (
58 |             &lt;div key={key}&gt;

59 |                 &lt;KeyValue label={tCommon('Type')} value={'SBOM Report'} /&gt;
60 |                 &lt;KeyValue label={tCommon('Name')} value={report.name} /&gt;
61 |                 &lt;KeyValue label={tCommon('Digest')} value={report.digest} /&gt;
```

```
- cargo clippy --message-format=json -- -D warnings
                                        -D clippy::cast_ptr_alignment
                                        -D clippy::dbg_macro
                                        -D clippy::derive_partial_eq_without_eq
                                        -D clippy::elidable_lifetime_names
                                        -D clippy::ignore_without_reason
                                        -D clippy::literal_string_with_formatting_args
                                        -D clippy::path_buf_push_overwrite
                                        -D clippy::precedence_bits > clippy-raw.log 2>&1
- cat clippy-raw.log | clippy-sarif > clippy-report.sarif
```

Борьба с избыточностью/минимизация

БЕКОН'26

У нас уже были **оптимизации и distroless-образы**

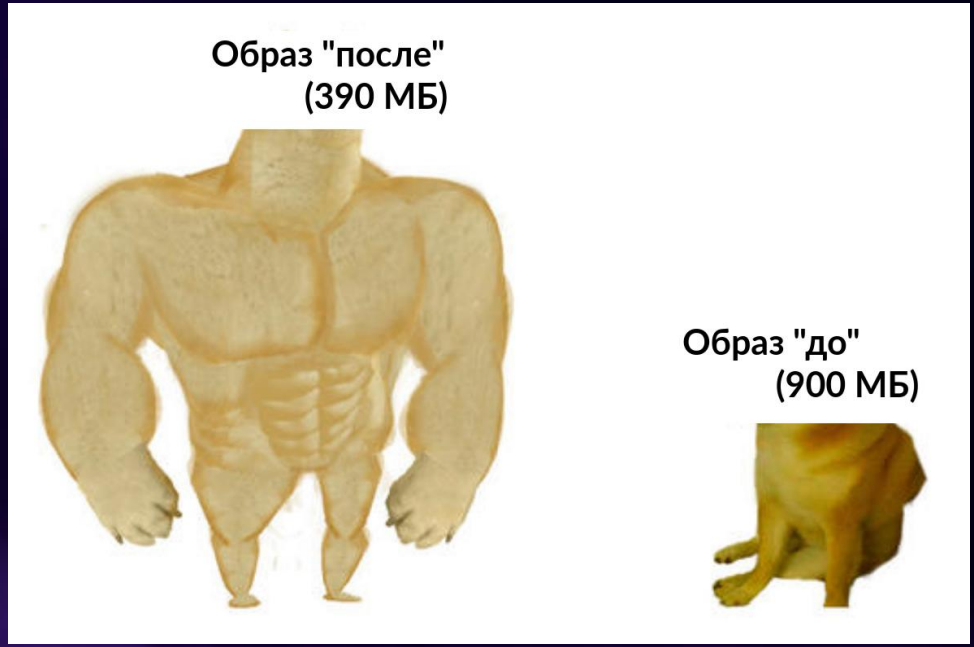
НО нет предела совершенству

- Отрефакторили и выкинули ненужные куски кода
- Допилили и оптимизировали некоторые образы
- Перенесли сборку полностью к себе

```
ARG CI_REGISTRY
FROM ${CI_REGISTRY}/proxy-gcr/distroless/static-debian13:nonroot

COPY --chown=nonroot:nonroot main.bin /charon

USER nonroot
CMD ["/charon"]
```



[Источник требования (п.5)]

Динамический анализ (фаззинг и unit-тесты)

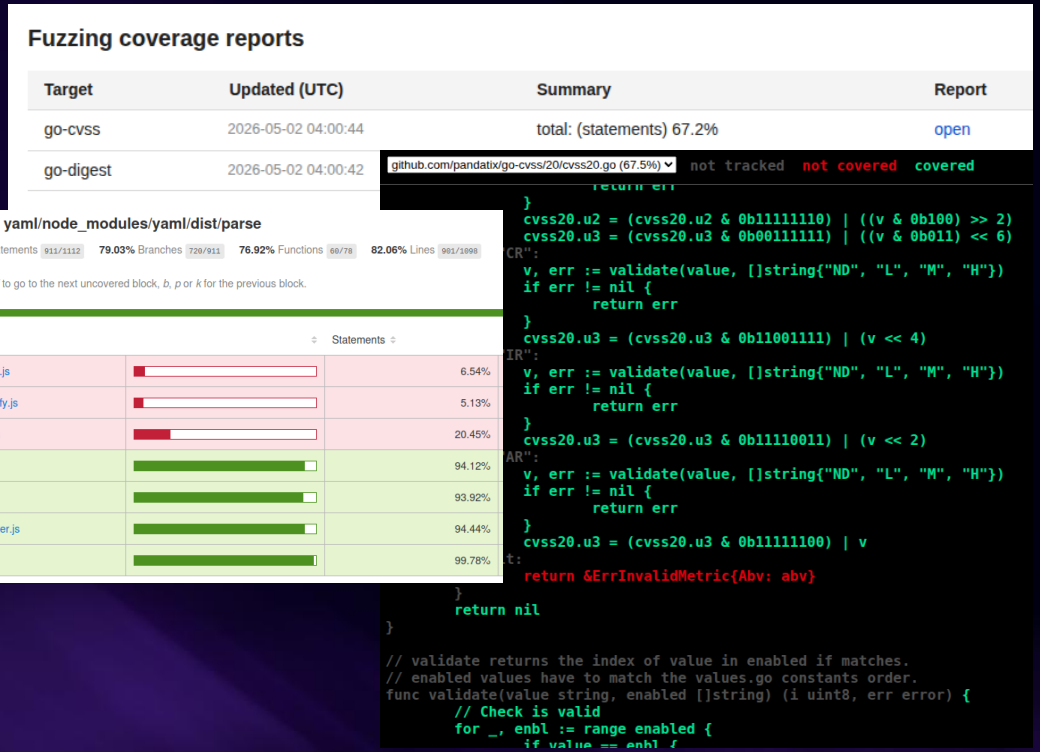
БЕКОН'26

Фаззинг (только для компонентов на ПА)

- Для Go “встроенный” go test fuzz
- Для JS использовал Jsfuzz
- Можно и нужно переиспользовать фаззинг, который уже есть в open source компонентах
- Считаем покрытие

unit-тесты

- Можно и нужно переиспользовать тесты, которые уже есть в open source компонентах
- Считаем покрытие



Fuzzing coverage reports

Target	Updated (UTC)	Summary	Report
go-cvss	2026-05-02 04:00:44	total: (statements) 67.2%	open
go-digest	2026-05-02 04:00:42		

All files yaml/node_modules/yaml/dist/parse

81.92% Statements 911/1112 79.03% Branches 720/911 76.92% Functions 60/78 82.06% Lines 901/1098

Press n or j to go to the next uncovered block, b, p or k for the previous block.

File	Statements
cst-scalar.js	6.54%
cst-stringify.js	5.13%
cst-visit.js	20.45%
cst.js	94.12%
lexer.js	93.92%
line-counter.js	94.44%
parser.js	99.78%

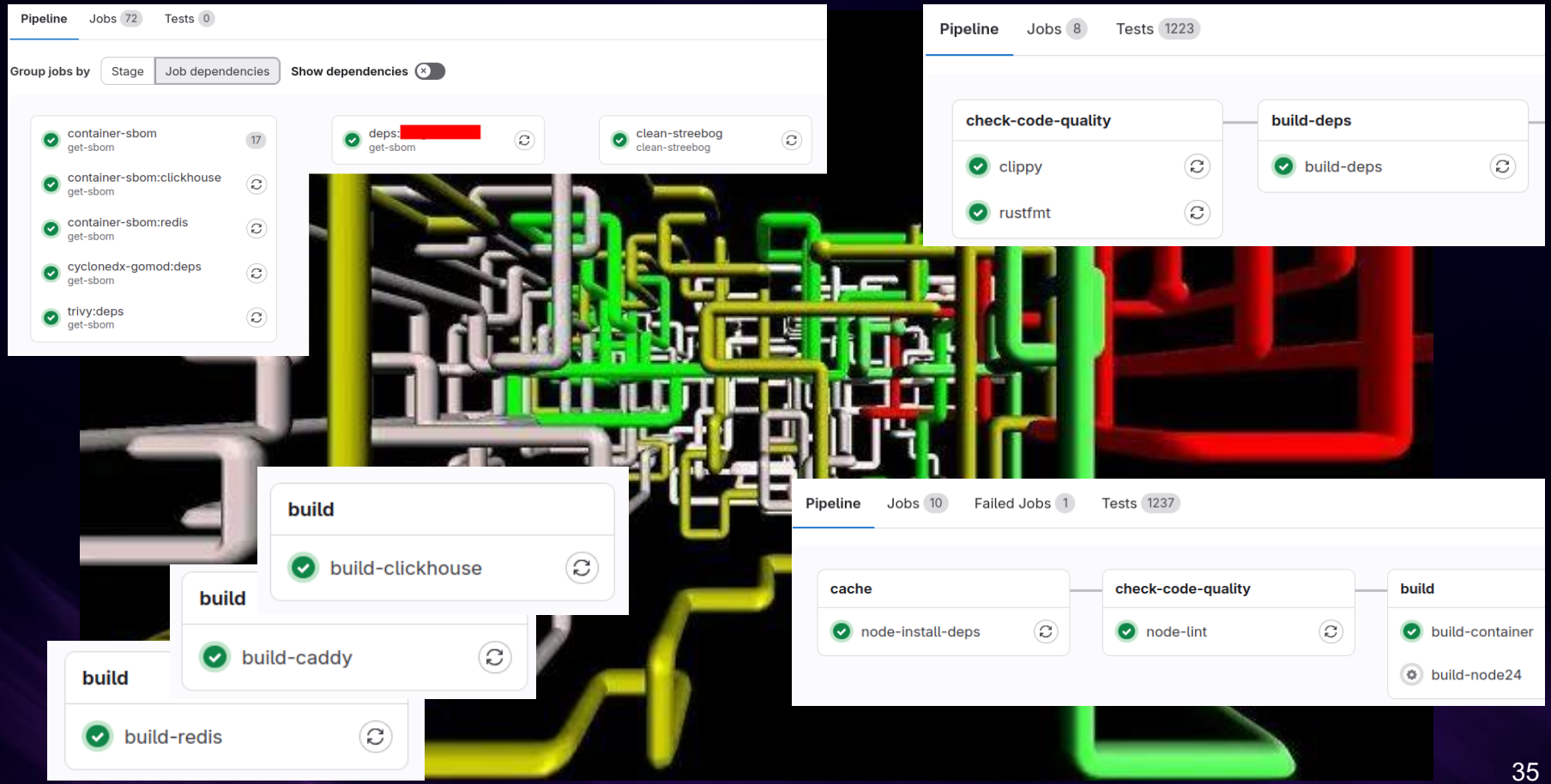
github.com/pandatix/go-cvss/20/cvss20.go (67.5%) not tracked not covered covered

```
        }
        cvss20.u2 = (cvss20.u2 & 0b11111110) | ((v & 0b100) >> 2)
        cvss20.u3 = (cvss20.u3 & 0b00111111) | ((v & 0b011) << 6)
    CR":
        v, err := validate(value, []string{"ND", "L", "M", "H"})
        if err != nil {
            return err
        }
        cvss20.u3 = (cvss20.u3 & 0b11001111) | (v << 4)
    IR":
        v, err := validate(value, []string{"ND", "L", "M", "H"})
        if err != nil {
            return err
        }
        cvss20.u3 = (cvss20.u3 & 0b11110011) | (v << 2)
    AR":
        v, err := validate(value, []string{"ND", "L", "M", "H"})
        if err != nil {
            return err
        }
        cvss20.u3 = (cvss20.u3 & 0b11111100) | v
    t:
        return &ErrInvalidMetric{Abv: abv}
    }
    return nil
}

// validate returns the index of value in enabled if matches.
// enabled values have to match the values.go constants order.
func validate(value string, enabled []string) (i uint8, err error) {
    // Check is valid
    for _, enbl := range enabled {
        if value == enbl {
```

Пайплайны (их будет много)

БЕКОН'26



Результат (согласовали ПиМ)

БЕКОН'26

Составление Протоколов: Ожидание

Составление Протоколов: Ожидание. Артефакты

БЕКОН'26

Архитектурный анализ	Анализ уязвимостей	Статический анализ (SAST)	Динамический анализ (unit и фаззинг)
- команды запуска и логи - отчеты сканирования - артефакты устранения Confirmed дефектов - результаты повторного сканирования	- команды запуска и логи - отчеты сканирования - обоснование неприменимости обнаруженных CVE - артефакты устранения Confirmed CVE - результаты повторного сканирования	- команды запуска и логи - SARIF-файл результатов анализа - отчет о выполнении разметки предупреждений CRITICAL и HIGH - артефакты устранения Confirmed дефектов - результаты повторного сканирования	- описание выбранной цели, входных корпусов - описание словарей - команда и лог сборки цели с санитайзерами (если поддерживаются) - команды запуска и логи фаззера - артефакты устранения дефектов, приводящих к падениям или зависаниям - отчеты о покрытии кода

Составление Протоколов: Ожидание. Артефакты

БЕКОН'26

Архитектурный анализ	Анализ уязвимостей	Статический анализ (SAST)	Динамический анализ (unit и фаззинг)
- команды запуска и логи - **отчеты** сканирования - **артефакты устранения** Confirmed дефектов - **результаты повторного сканирования**	- команды запуска и логи - **отчеты** сканирования - обоснование неприменимости обнаруженных CVE - **артефакты устранения** Confirmed CVE - **результаты повторного сканирования**	- команды запуска и логи - SARIF-файл результатов анализа - **отчет** о выполнении разметки предупреждений CRITICAL и HIGH - **артефакты устранения** Confirmed дефектов - **результаты повторного сканирования**	- описание выбранной цели, входных корпусов - описание словарей - команда и лог сборки цели с санитайзерами (если поддерживаются) - команды запуска и логи фаззера - **артефакты устранения** дефектов, приводящих к падениям или зависаниям - **отчеты** о покрытии кода

БЕКОН'26

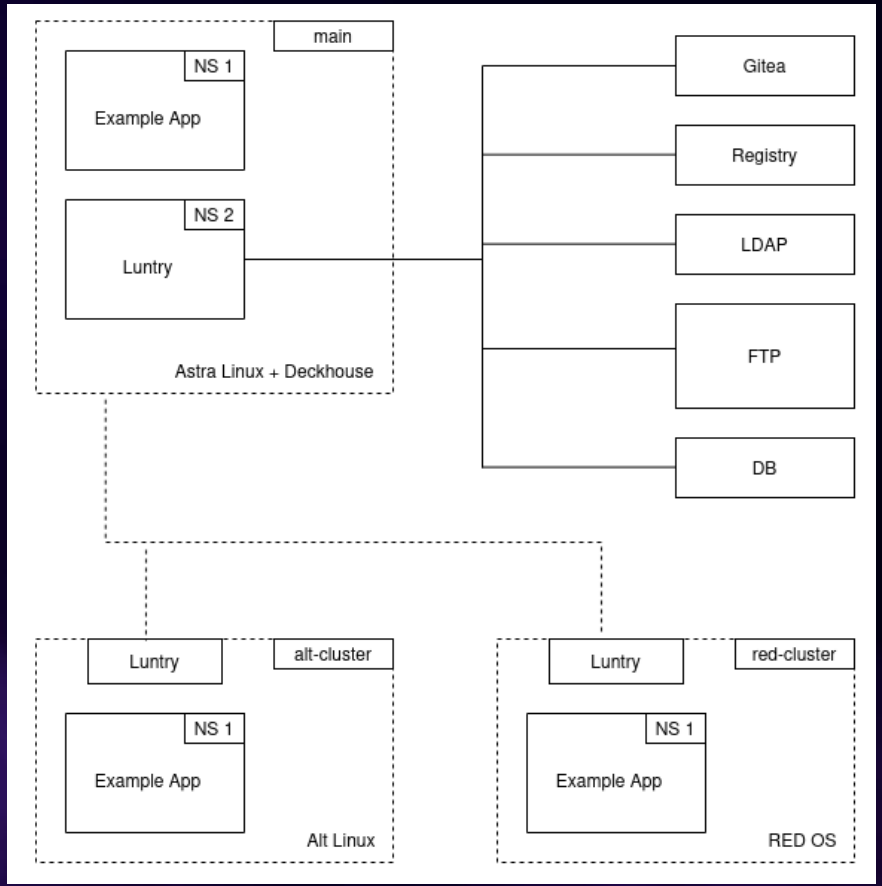
Составление Протоколов: Реальность

Стенд для испытаний

БЕКОН'26

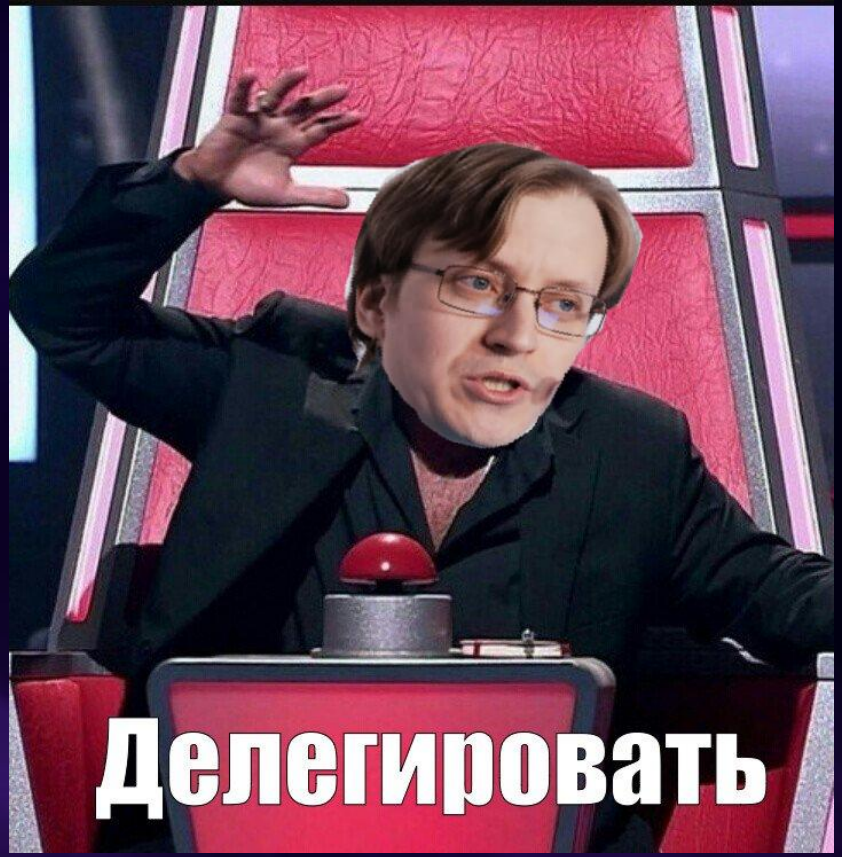
Должны обеспечить:

- Закрытый периметр
- Покрытие конфигураций всех заявленных вычислительных сред
- Возможность проверки всех ФБ, которые заявляем в Формуляре
- Обновление стенда по мере выявления и исправления проблем во время испытаний



Архитектурный анализ

- Делегировали лаборатории
- Были найдены дефекты в результате анализа KICS’ом, которые были успешно исправлены
- А с привилегиями контейнеров было всё норм 😎



Источник требования (п.5)

Анализ уязвимостей (Борьба с CVE)

БЕКОН'26

Регулярно обновляли зависимости, но не было процесса:

- Выстроили процесс обновлений
- Встроили в pipeline проверки

Если CVE невозможно закрыть через обновление, тогда нужно:

- Либо описать почему уязвимость не касается вашего кода
- Либо форкнуть к себе и внести исправление
- Либо “закрыть” своим кодом в продукте
- Послать патчи в апстрим

"ДО"

```
$ grype sbom:visualizer.json -o table --sort-by severity | grep -v "Negligible"
 ✔ Scanned for vulnerabilities     [8 vulnerability matches]
   ├── by severity: 0 critical, 2 high, 5 medium, 1 low, 0 negligible
   └── by status:   8 fixed, 0 not-fixed, 0 ignored

NAME    INSTALLED  FIXED IN  TYPE  VULNERABILITY        SEVERITY  EPSS           RISK
lodash  4.17.21    4.18.0    npm   GHSA-r5fr-rjxr-66jc  High      < 0.1% (11th)  < 0.1
lodash  4.17.23    4.18.0    npm   GHSA-r5fr-rjxr-66jc  High      < 0.1% (11th)  < 0.1
axios   1.13.5     1.15.0    npm   GHSA-fvcv-3m26-pcqx  Medium    < 0.1% (13th)  < 0.1
axios   1.13.5     1.15.0    npm   GHSA-3p68-rc4w-qgx5  Medium    < 0.1% (12th)  < 0.1
lodash  4.17.21    4.17.23   npm   GHSA-xxjr-mmjv-4gpg  Medium    < 0.1% (8th)   < 0.1
lodash  4.17.21    4.18.0    npm   GHSA-f23m-r3pf-42rh  Medium    < 0.1% (6th)   < 0.1
lodash  4.17.23    4.18.0    npm   GHSA-f23m-r3pf-42rh  Medium    < 0.1% (6th)   < 0.1
qs      6.14.1     6.14.2    npm   GHSA-w7fw-mjwx-w883  Low       < 0.1% (15th)  < 0.1
```

"ПОСЛЕ"

```
$ grype sbom:visualizer.json -o table --sort-by severity | grep -v "Negligible"
 ✔ Scanned for vulnerabilities     [0 vulnerability matches]
   ├── by severity: 0 critical, 0 high, 0 medium, 0 low, 0 negligible
   └── by status:   0 fixed, 0 not-fixed, 0 ignored
```

Финализация процесса

Сборка дистрибутива из исходников

- Собираемся в закрытом периметре
- Контролируем, что нет обращений на внешние ресурсы
- Вендоринг + хранилище артефактов

Фиксируем и собираем

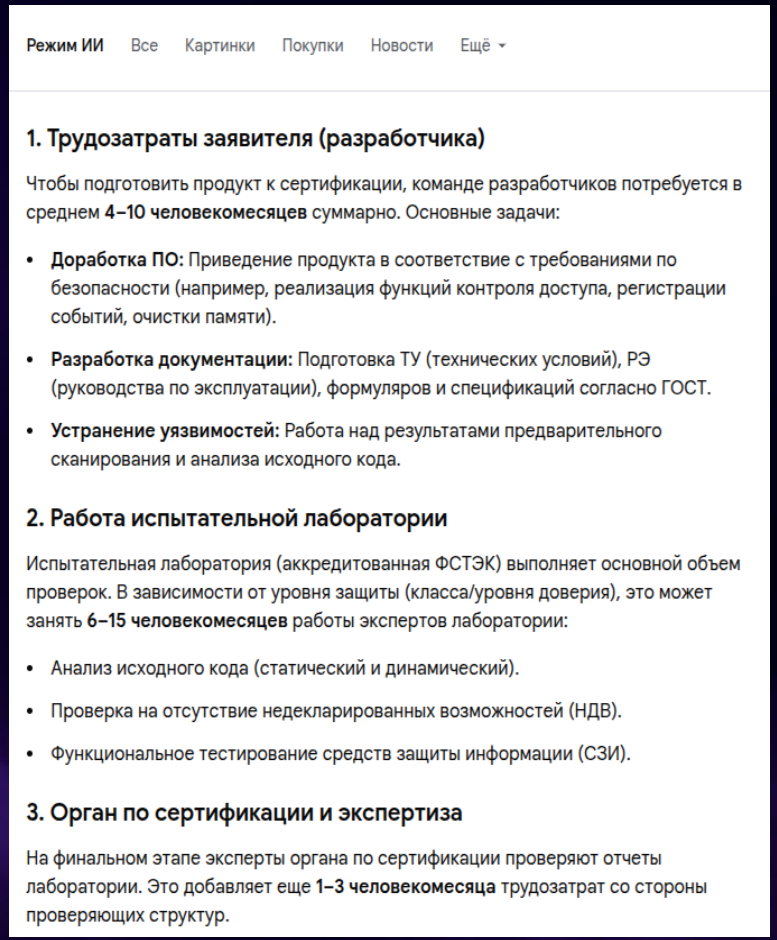
- Артефакты процесса сборки
  - параметры сборки
  - логи сборки
  - сетевой трафик
- Артефакты проверок
  - SAST
  - unit-тесты, fuzzing
  - SBOM-ы и CVE
  - protestware, “секреты” и т.д.
  - контрольные суммы для исполняемых файлов
- Дистрибутив

Команда и ресурсы (единственный слайд с ИИ)

БЕКОН'26

Если вы проходите **сертификацию в первый раз**, то очень сложно **самостоятельно оценить** реальные возможности и сроки сразу

Тесное взаимодействие Заявителя (Разработчика) и Испытательной лаборатории, поскольку Испытательная лаборатория не в состоянии провести весь анализ продукта самостоятельно без инициативы Заявителя



Режим ИИ Все Картинки Покупки Новости Ещё

1. Трудозатраты заявителя (разработчика)

Чтобы подготовить продукт к сертификации, команде разработчиков потребуется в среднем **4–10 человекомесяцев** суммарно. Основные задачи:

- **Доработка ПО:** Приведение продукта в соответствие с требованиями по безопасности (например, реализация функций контроля доступа, регистрации событий, очистки памяти).
- **Разработка документации:** Подготовка ТУ (технических условий), РЭ (руководства по эксплуатации), формуляров и спецификаций согласно ГОСТ.
- **Устранение уязвимостей:** Работа над результатами предварительного сканирования и анализа исходного кода.

2. Работа испытательной лаборатории

Испытательная лаборатория (аккредитованная ФСТЭК) выполняет основной объем проверок. В зависимости от уровня защиты (класса/уровня доверия), это может занять **6–15 человекомесяцев** работы экспертов лаборатории:

- Анализ исходного кода (статический и динамический).
- Проверка на отсутствие недекларированных возможностей (НДВ).
- Функциональное тестирование средств защиты информации (СЗИ).

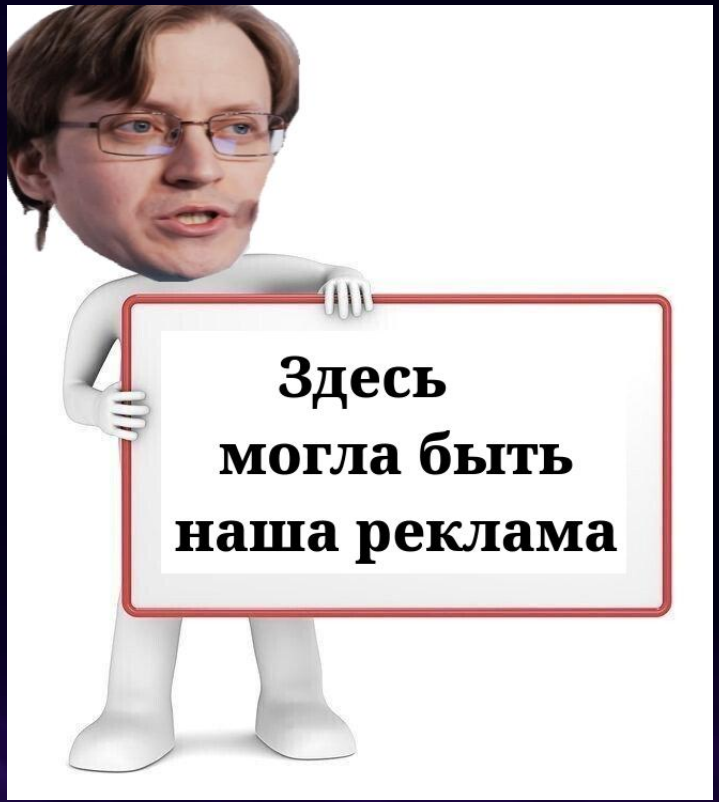
3. Орган по сертификации и экспертиза

На финальном этапе эксперты органа по сертификации проверяют отчеты лаборатории. Это добавляет еще **1–3 человекомесяца** трудозатрат со стороны проверяющих структур.

Результат

Сертификат соответствия средств защиты информации требованиям по безопасности информации

(мы очень хотели получить к БЕКОН-у, но не успели)



Вместо заключения

БЕКОН'26

Ожидание

- Должны быть процессы, а не просто единоразовый сбор “артефактов” для получения “бумажки”
- Выход на сертификацию с предварительной подготовкой, иначе сертификация может занять больше времени
- Сертификацию не сможет сделать один человек
- Быть готовым к изменяющимся требованиям в процессе сертификации

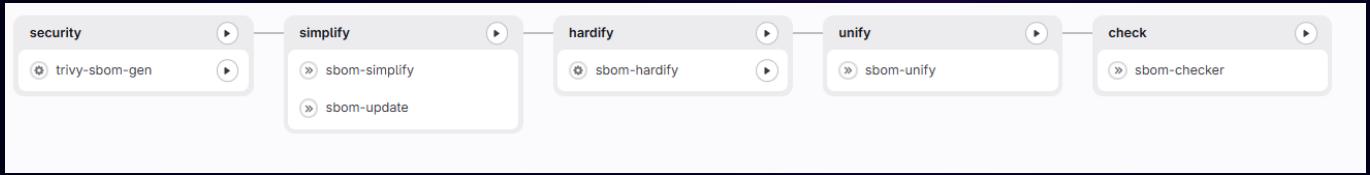
Реальность

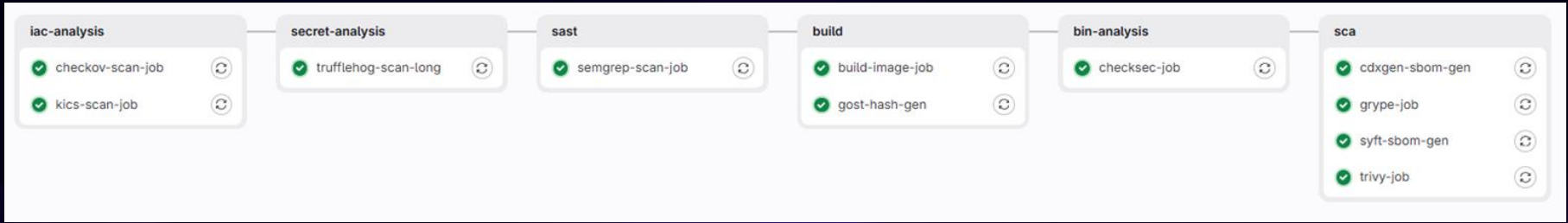
- У нас были процессы до начала взаимодействия с лабораторией
- Мы выходили подготовленными, но в процессе обнаружились “нюансы”
- Я один пытался, но не смог, поэтому спасибо команде, которая всячески помогала и поддерживала
- Быть готовым к изменяющимся требованиям в процессе сертификации

Бонус

БЕКОН'26

«Demo certification pipeline» – пример пайплайна, этапы которого можно использовать как «отправные точки» при сертификации





https://gitlab.com/demo-certification-pipeline

БЕКОН'26

Всем спасибо!

Анатолий Карпенко (Luntry)
site: luntry.ru
tg: @rusdacent

Андрей Слепых (Фобос-НТ)
site: fobos-nt.ru
tg: @SlepoyPew

БЕКОН'26

Вопросы?