UserGate

Практический опыт по ускоренному внедрению РБПО в UserGate

Дмитрий Ткачев
Директор по разработке

20.02.2026

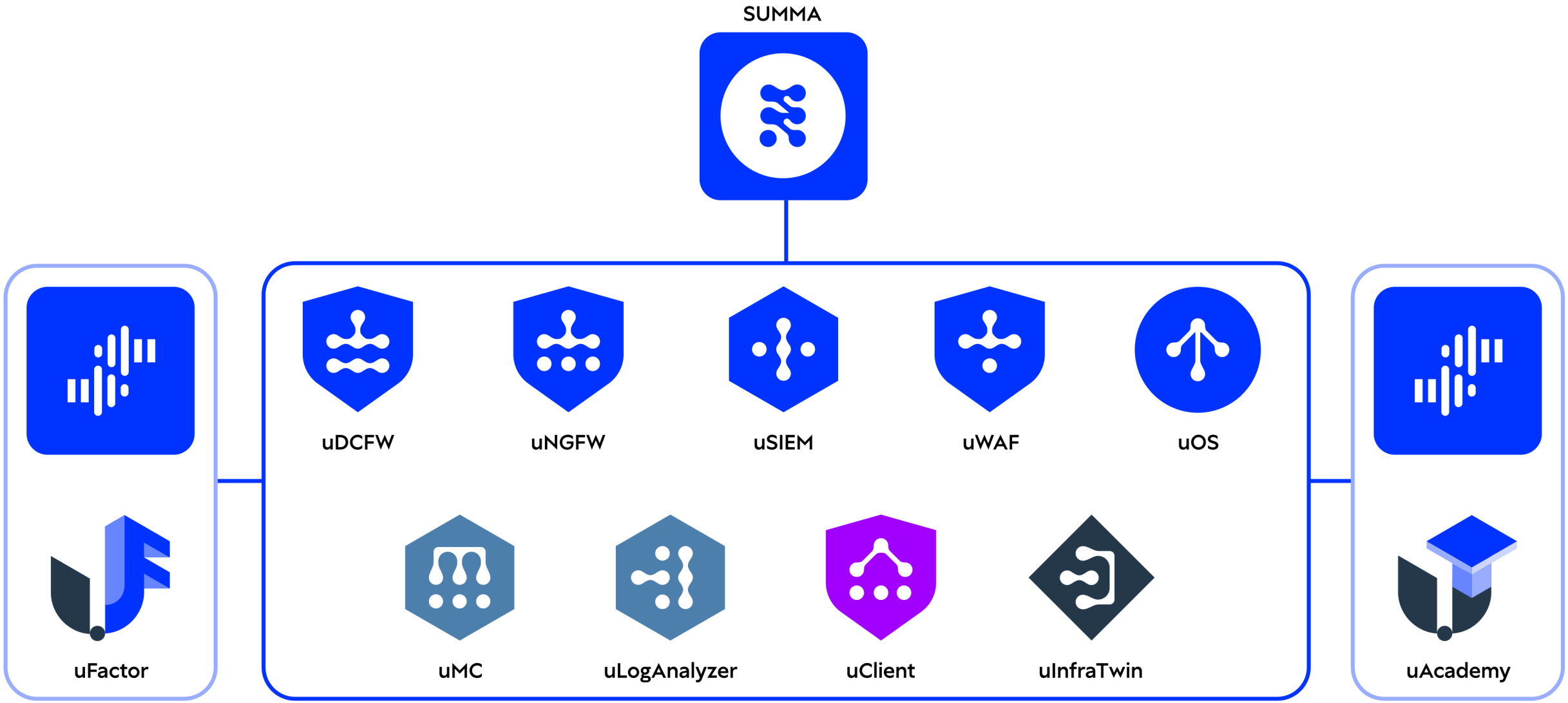
ОБ АВТОРЕ

UserGate

Ткачев Дмитрий

- 20 лет в разработке ПО
- Технический опыт – разработка, проектирование систем, reverse engineering
- Управление командами и подразделениями разработки с 2012 года
- В UserGate с мая 2024, директор дисциплины программного обеспечения

ЭКОСИСТЕМА USERGATE SUMMA



АППАРАТНЫЕ ПЛАТФОРМЫ USERGATE

КРУПНЫЕ СЕТИ, ЦОДЫ

E1000

E3000

F8000

E1010

E3010

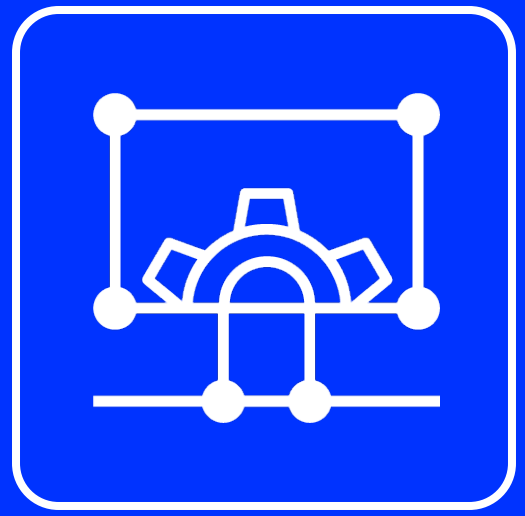
F8010

FG

ДИСКЛЕЙМЕР

01 В докладе мы рассказываем, каким образом внедряем стандарт РБПО в Компании, процесс еще не завершен, сейчас находимся на начальном этапе выездного аудита.

02 Мы не знаем универсальные решения по внедрению, но знаем, что сработало конкретно у нас в UserGate.

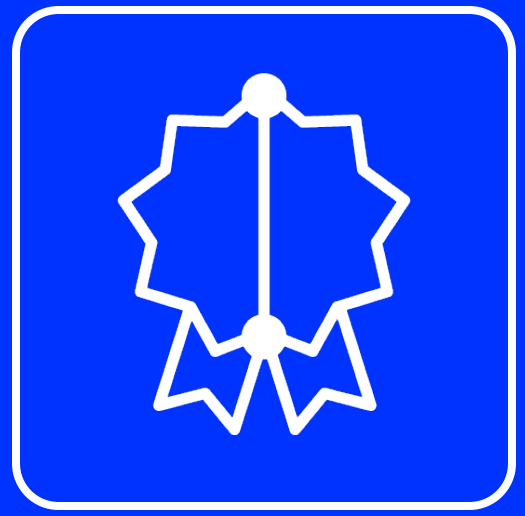


UserGate

Нанести пользу производственным процессам и продуктам компании

ТЕХНИЧЕСКИЙ СТЕК РАЗРАБОТКИ

- Erlang
- C - много Linux Kernel
- C++
- Rust
- Golang
- Python
- Своя OS UGOS, на базе linux с особенностями
- Большая история и кодовая база, 15+ лет



ТОЧКА СТАРТА

Точкой старта считаем май-июнь 2024, проведение независимого аудита процессов разработки компанией Фобос-НТ

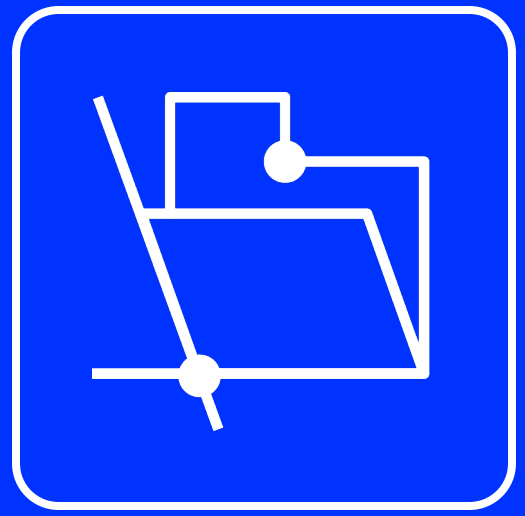
- Средняя оценка процессов:
  0.8 - недостаточный уровень реализации
- Объективная картина состояния
  25 процессов разработки РБПО
- Выявлены критичные зоны для изменений
- Четкие рекомендации и указание несоответствий

ПЕРВЫЙ ШАГ - РАЗРАБОТКА РЕГЛАМЕНТОВ

UserGate

Письменная и четкая формализация регламентов - не про бюрократию

- Регламенты синхронизированы с разделами ГОСТ Р 56939-2024
- Четкие договоренности: как вести разработку далее
- Обеспечение единого понимания стандартов во всей компании



25 ПРОЦЕССОВ РБПО

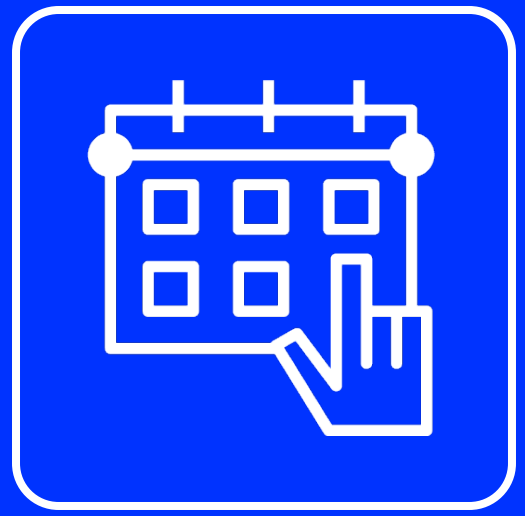
UserGate

1. План развития РБПО
2. Обучение сотрудников
3. Управление ТБ ПО
4. Управление конфигурацией в рамках ЖЦ ПО
5. Управление жизненным циклом дефектов
6. Разработка, уточнение и анализ архитектуры
7. Моделирование угроз и разработка описания
8. Оформление исходного кода и описания
9. Проведение экспертизы исходного кода ПО
10. Проведение статистического анализа исходного кода ПО
11. Проведение динамического анализа кода ПО
12. Использование системы безопасной сборки
13. Обеспечение безопасности сборочной среды
14. Доступ к исходному коду ПО и обеспечения
15. Использование секретов
16. Композиционный анализ
17. Проверка кода на предмет внедрения вредоносного
18. Проведение функционального тестирования
19. Проведение нефункционального тестирования
20. Проведение приемочного тестирования
21. Обеспечение целостности ПО, передаваемого пользователям
22. Безопасная поставка ПО пользователям
23. Техническая поддержка
24. Поиск ошибок и уязвимостей в ПО при его эксплуатации
25. Вывод ПО из эксплуатации

UserGate

06 Разработка, уточнение и анализ архитектуры

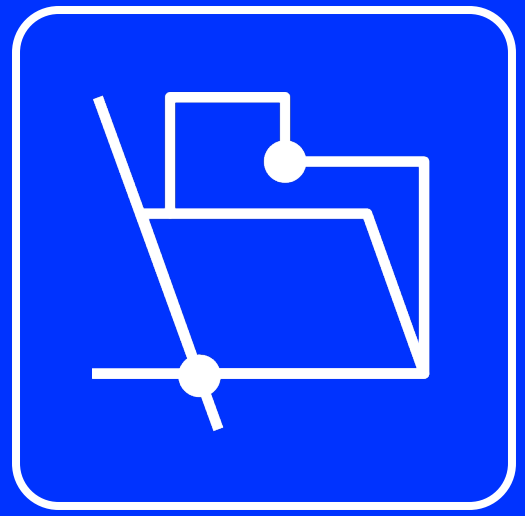
- Design Review и ADR (Architecture Decision Records)
- Моделирование угроз (OWASP)
- Security-требования
- Системный анализ и требования безопасности
- Appsec встроены на этапе проработки требований и дизайн-ревью



UserGate

10 Проведение статического анализа

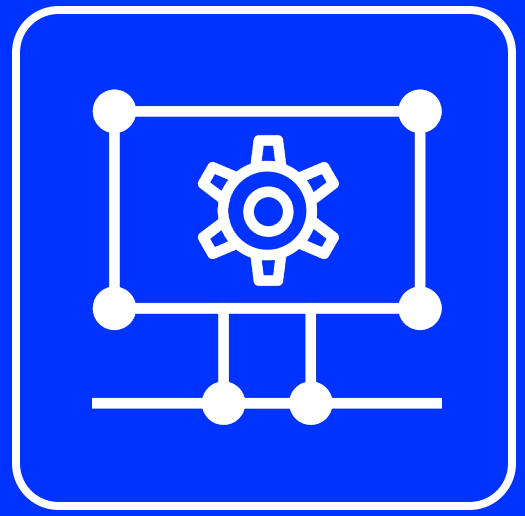
- **Инструменты первый подход:**
  - SonarQube - для всех продуктов/языков
- **Сложности**
  - Не поддерживает Erlang (у нас много)
  - Не удовлетворяет требованиям РБПО по C/C++



UserGate

10 Проведение статического анализа

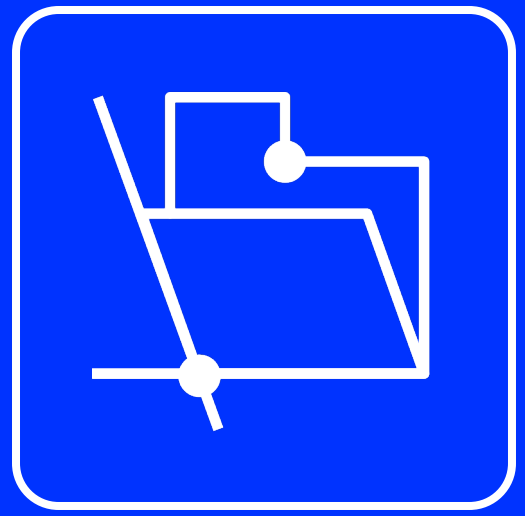
- **Итоговая конфигурация**
  - Svace - для всех языков, кроме Erlang
  - Erlang Dialyzer - для кода на Erlang
  - SonarQube - вторым эшелоном, ловит Security



UserGate

10 Проведение статического анализа

- **Особенности Svace**
  - Конфигурации Svace на первом этапе
  - Svace нужно много времени проверку
  - Не получается проверить diff кода
- **Особенности Dialyzer**
  - Большая кодовая база, много сработок, не могли блокировать Pipeline
  - Внедряли через «неухудшение»



UserGate

10 Проведение статического анализа

Пример разбора сработок в UGOS (Linux Kernel)

- **Текущие шаги**
  - Исправлены сработки в Linux Kernel
  - Работаем над исправлением в нашем коде
  - Начинаем работы по исправлению в сторонних библиотеках

Statistics by checkers

363351-NGFW-target-linux-compile-X64

Severity	Reliability	Total	Undecided	Confirmed
Critical	VeryHigh	21 (21)		
Critical	High	29 (29)		
Critical	Average	31 (31)		
Critical	Unknown	13 (13)		

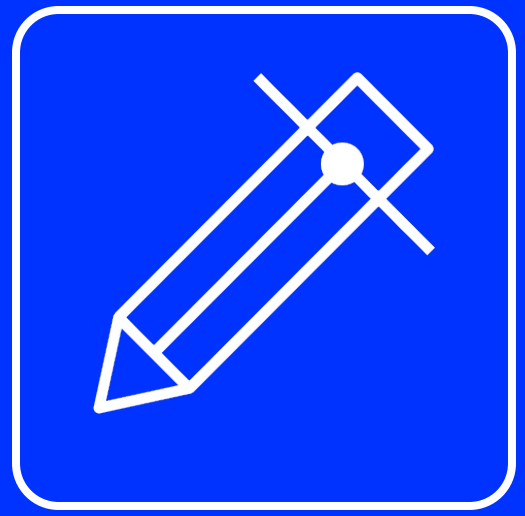
10 Проведение статического анализа

- **Статический анализ заимствованных компонентов**
  - Инструмент - Svace
  - Объем работы - большой
- **Рекомендации**
  - Для новых проектов крайне внимательно добавлять новые библиотеки

UserGate

11 Проведение динамического анализа

- Появилась практика Application Security
- Построение поверхности атаки
- В первую очередь проводим fuzzing по поверхности
- Выбираем функции, которые могут принимать данные
- Используем AFL++
- На этапе design review видим новый интерфейс и готовим fuzzing тесты
- Закрываем разработку функциональности после завершения fuzzing тестирования.

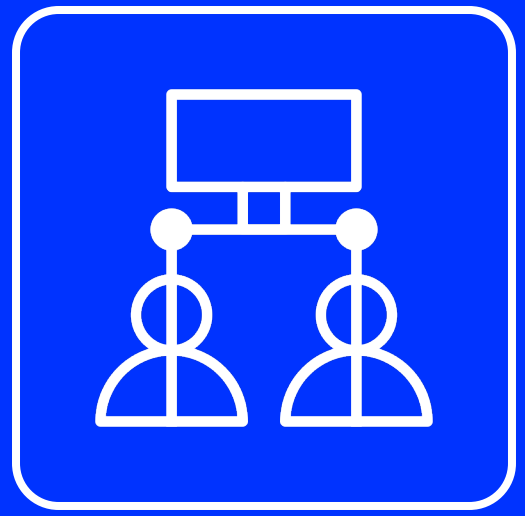


UserGate

11 Проведение динамического анализа

Оценка защищенности продуктов перед релизом

- Команда экспертов проверяет релизы перед публикацией
- Проверяем свои релизы и продаем услуги динамического анализа на рынок



UserGate

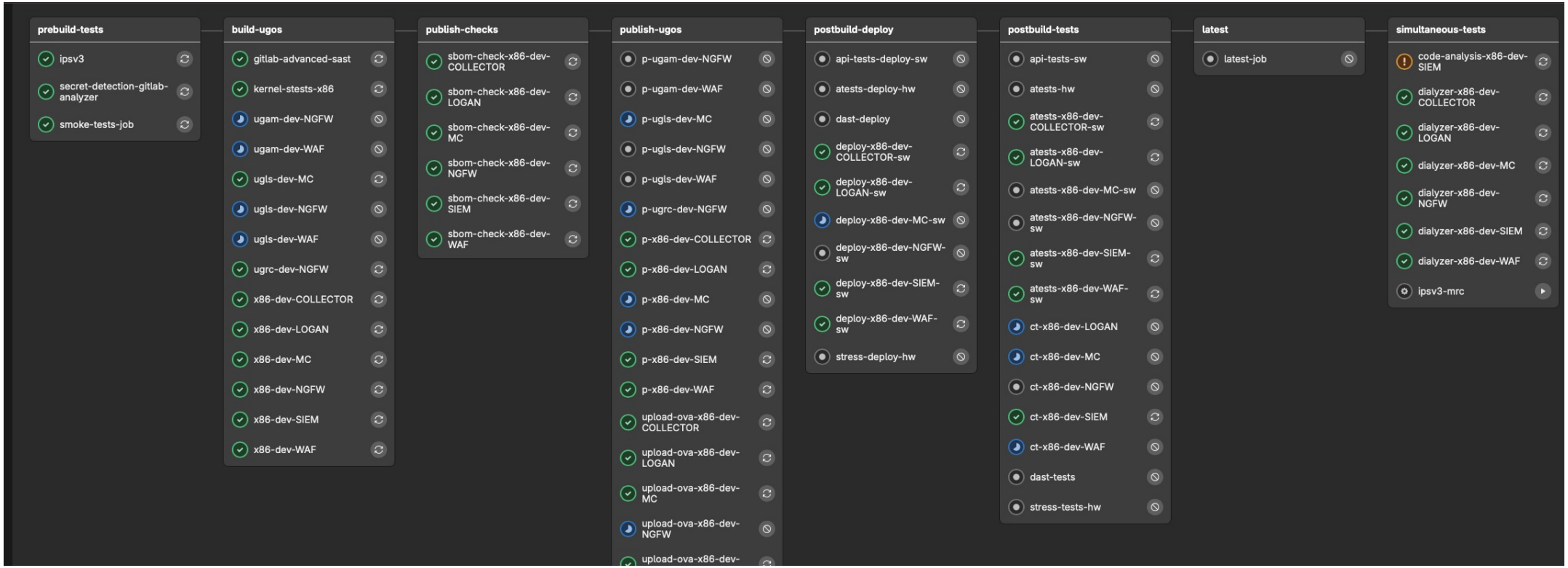
12 Использование системы безопасной сборки

- Sonatype Nexus Repository для контроля компонентов
- Генерация SBOM (ППК)
- Безопасная сборка без доступа к внешней среде
- **Общий подход - не делаем документы по запросу. Если есть запрос, то включаем генерацию в процесс сборки**

- **Валидируем в pipeline корректность SBOM по стандартам ИСП РАН**
- При ухудшении по тестам pipeline завершается ошибкой
- Один из этапов - разметка зависимых компонентов по Svace

UserGate

12 Использование системы безопасной сборки



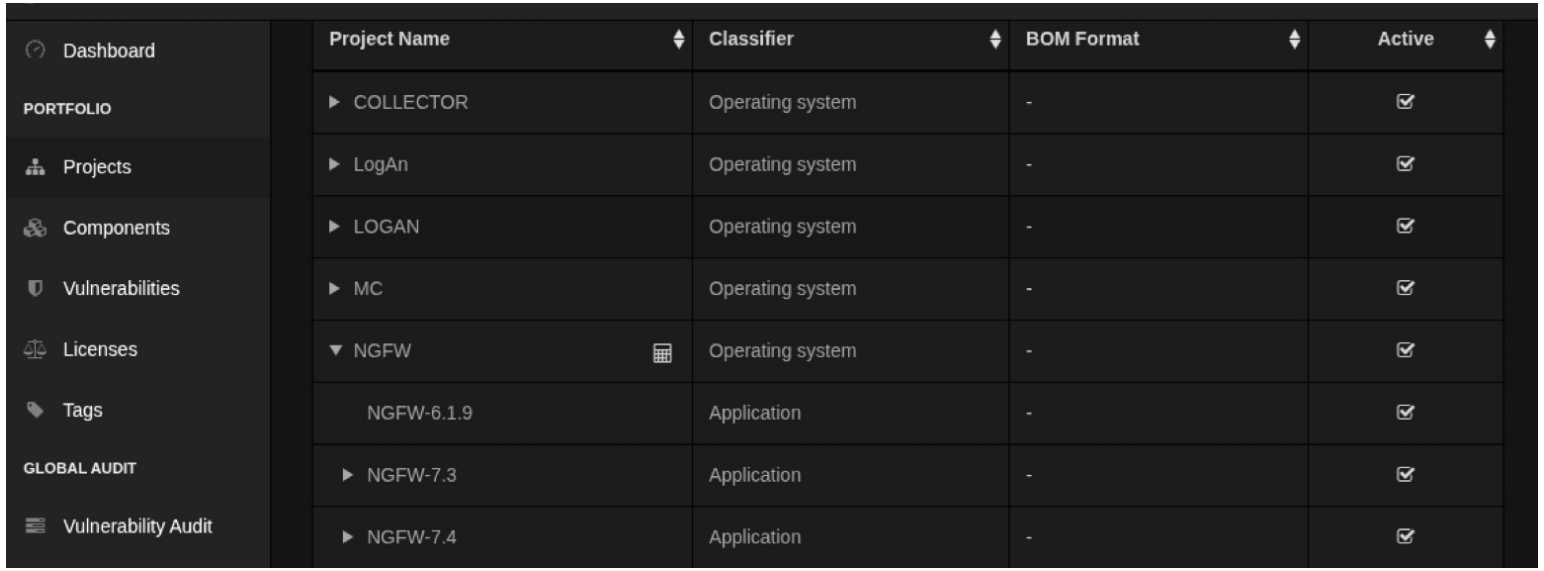
UserGate

15 Использование секретов

- Секреты в коде обнаруживаем на pre-commit (даже не попадает в git) и pre-build и в MP
- Используем Gitleaks – open source инструмент
- Пробовали TruffleHog (сильно перегружен), Talisman (неудобная интеграция на pre-commit)
- Секреты в Hashicorp vault

16 Композиционный анализ

- Dependency track – в пайплайне, если есть новая уязвимость, то блокировка MR
- Новые компоненты в хранилище артефактов только после проверки безопасности
- Получаем на ежедневной основе актуальную информацию об уязвимостях



UserGate

16 Композиционный анализ

- Разработали скрипты сбора информации об уязвимостях из источников

libxml2 2.13.9

Regressions

- valid: Don't add ids when validating entity content
- io: Fix reading from pipes like stdin on Windows
- parser: Fix handling of invalid char refs in recovery mode

Security

- regexp: Avoid integer overflow and OOB array access
- tree: Guard against atype corruption
- [CVE-2025-49794] [CVE-2025-49796] schematron: Fix xmlSchematronReportOutput
- [CVE-2025-49795] schematron: Fix null pointer dereference leading to DoS (Michael Mann)
- [CVE-2025-6170] Fix potential buffer overflows of interactive shell (Michael Mann)
- [CVE-2025-6021] tree: Fix integer overflow in xmlBuildQName

Bug fixes

- save: Fix serialization of attribute defaults containing <

Improvements

- parser: Fix xmlSaturatedAddSizeT argument type

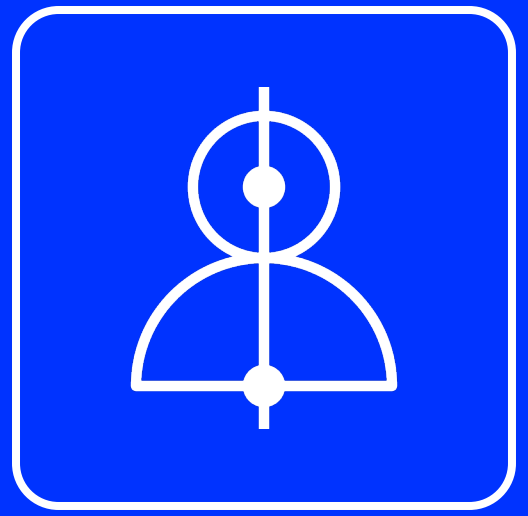
Build systems and portability

- meson: Add libxml2 part of include dir to pc file (Heiko Becker)
- cmake: Fix installation directories in libxml2-config.cmake
- io: Fix linkage of __xml*BufferCreateFilename functions

URL: http://gitlab.gnome.org/GNOME/libxml2/-/releases/v2.13.9

КОМАНДЫ, КОТОРЫМ ПРИШЛОСЬ МЕНЯТЬСЯ

- **Development –**
  внедрение новых стандартов в разработке, архитектуре, документации
- **DevOps –**
  огромный объем работы по автоматизации
- **AppSec –**
  SAST/DAST анализ, ревью кода, контроль зависимостей
- **QA & QC –**
  автоматизация контроля качества
- **GR –**
  все регламентные и регуляторные вопросы
- **HR –** обучение сотрудников



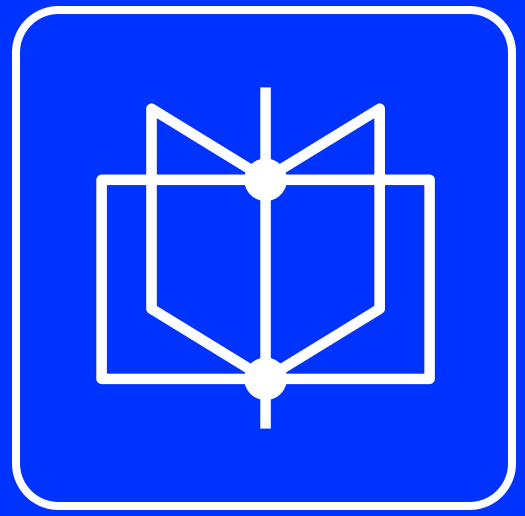
УРОКИ, КОТОРЫЕ МЫ ВЫУЧИЛИ

UserGate

01	Честный внешний аудит - фундамент успеха
02	Формализация регламентов до начала внедрения
03	Автоматизация решает - без CI/CD/DevOps невозможно масштабировать
04	Технических препятствий нет, на некоторые задачи надо больше времени
05	Кросс-функциональное сотрудничество

БЛИЖАЙШИЕ ШАГИ

- Проведение пилота с CodeScoring
- Периодические внутренние аудиты процессов
- Разработка внутренних курсов обучения по отдельным процессам и инструментам



UserGate

Текущая точка спустя 1.5 года – выход на сертификацию

UserGate

Для UserGate РБПО —
это инвестиции в бизнес,
а не просто сертификация

UserGate

ВОПРОСЫ?

СВЯЗАТЬСЯ С НАМИ

UserGate

Контакты для связи

Дмитрий Ткачев
Директор по разработке

e-mail: DTkachev@usergate.com

UserGate

СПАСИБО ЗА ВНИМАНИЕ