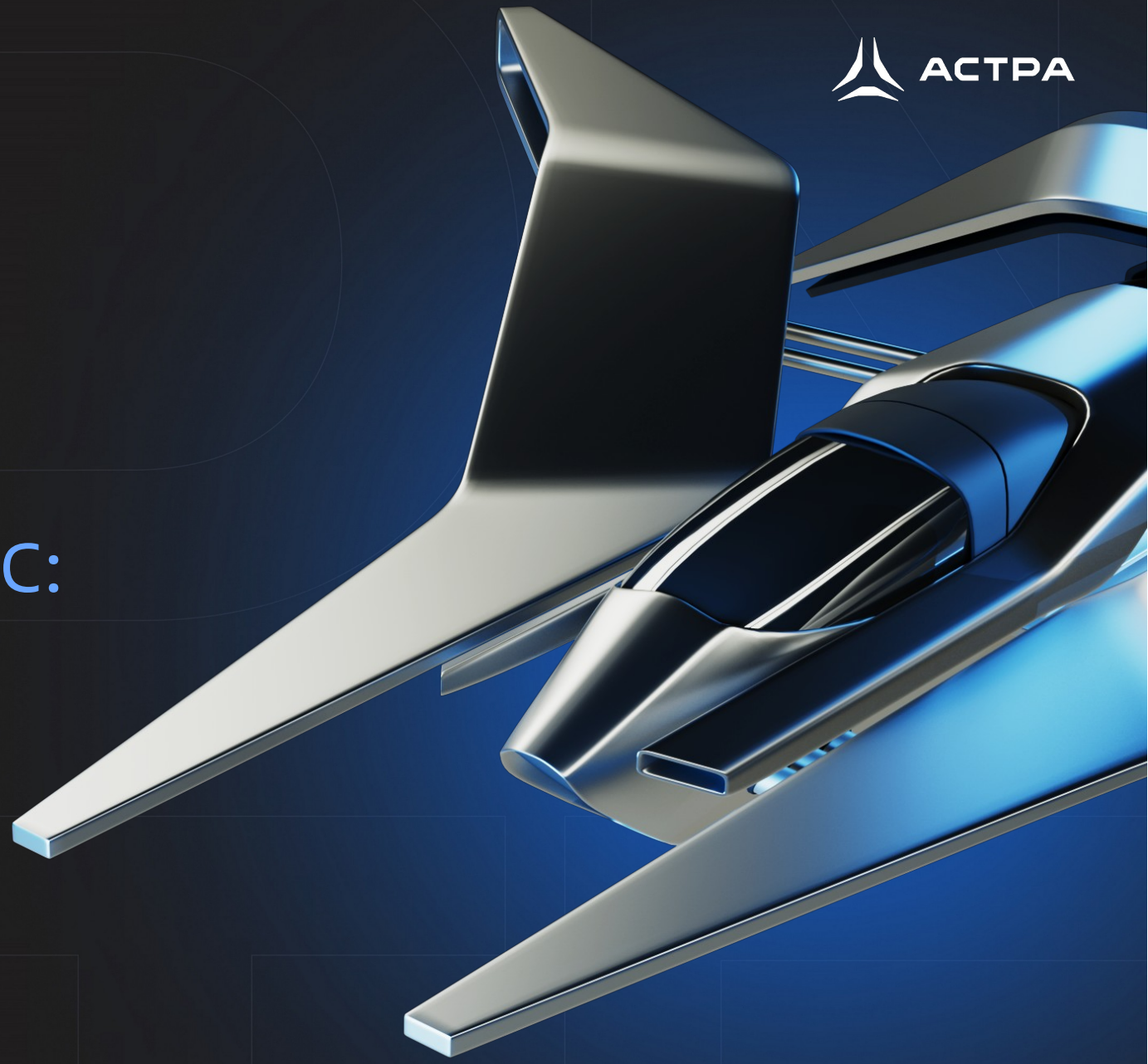
АСТРА

Разработка безопасной ОС: вызовы и решения



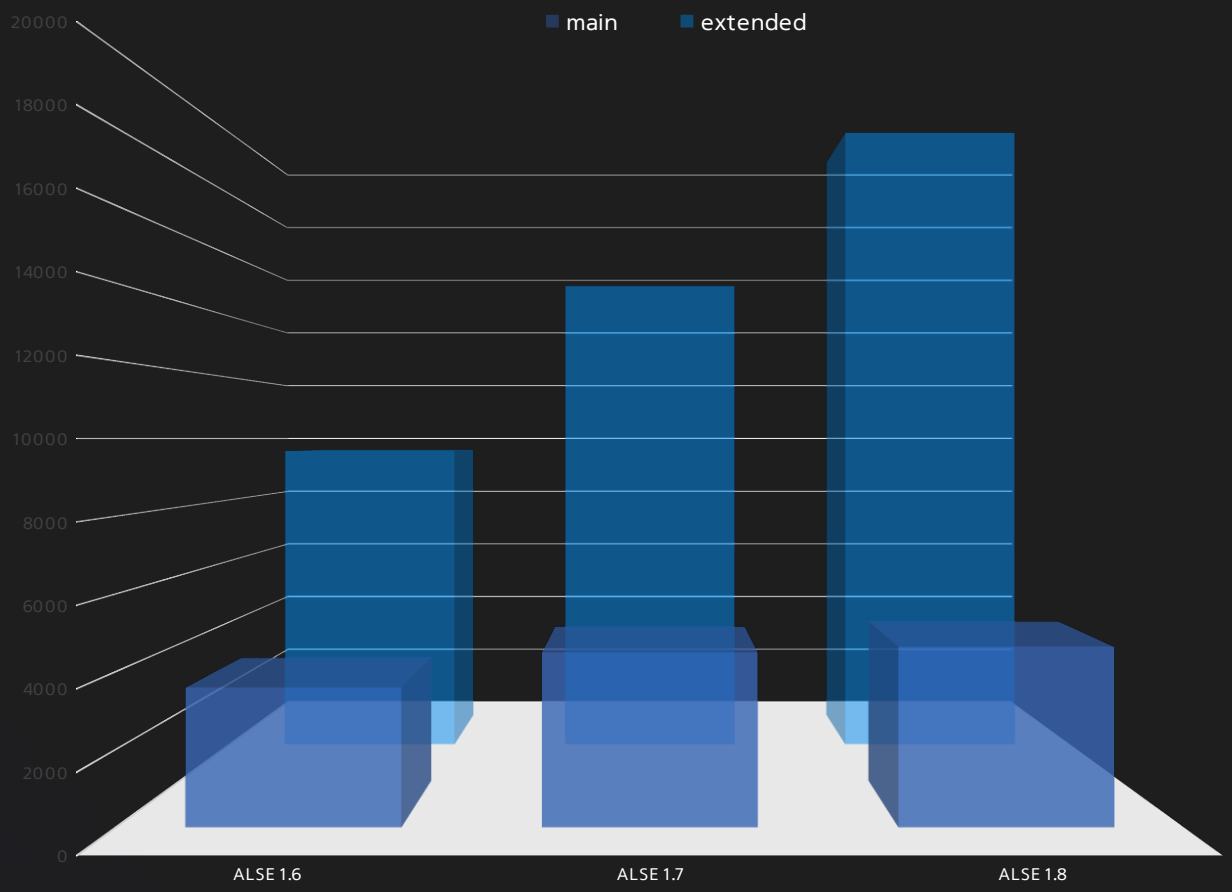
Владимир Тележников

Директор департамента
анализа безопасности

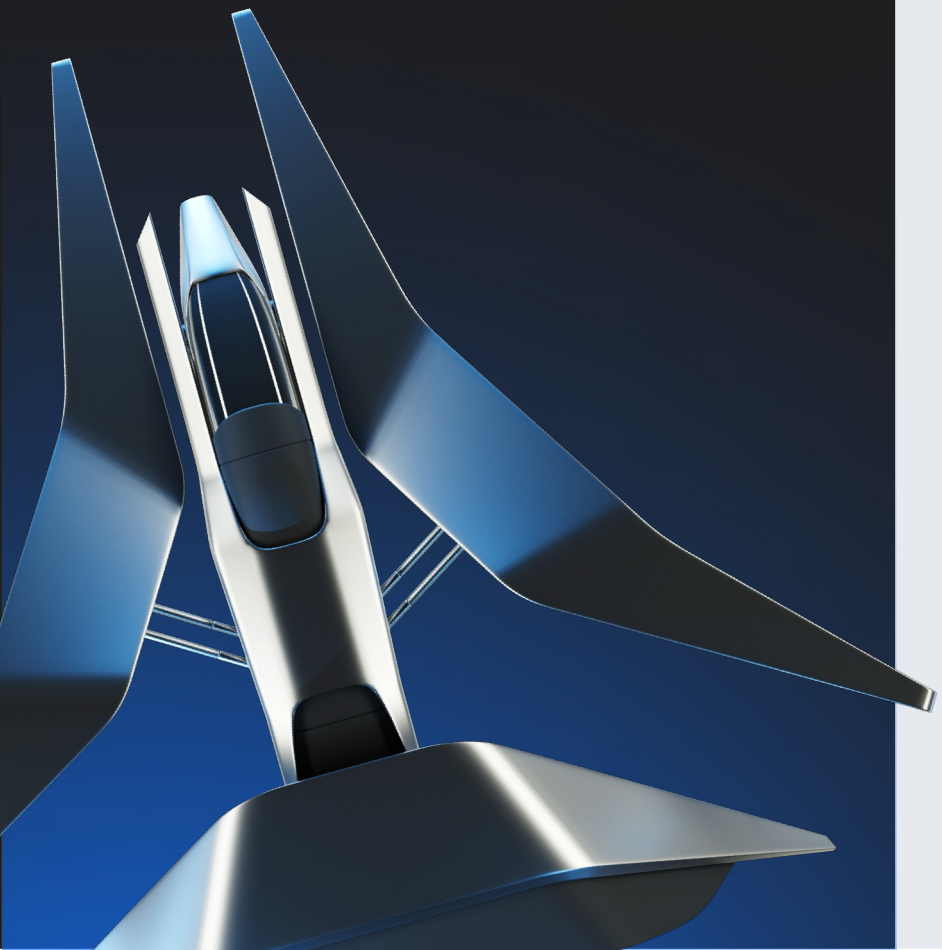
2025 год

Основной вызов - «ООО»
Объем объекта оценки

- **ALSE 1.6:**
  Репозиторий main - более 3 600 пакетов
  Репозиторий extended - более 9 600 пакетов
- **ALSE 1.7:**
  Репозиторий main — более 4 500 пакетов
  Репозиторий extended — более 15 000 пакетов
- **ALSE 1.8:**
  Репозиторий main — более 4 650 пакетов
  Репозиторий extended — более 20 000 пакетов



Модель угроз и поверхность атаки



Модель угроз

- включает более 160 актуальных УБИ
- оценивает каждую УБИ в отдельности и в рамках укрупненных групп
- постоянно актуализируется с учетом расширения сценариев применения
- формируется с учетом встроенных механизмов защиты

Детализация объектов воздействия

- ядро и модули ядра ОС
- средства защиты информации
- сетевые сервисы
- системное ПО
- графическая среда
- интерпретаторы
- прикладное ПО

Поверхность атаки

- детализируется с учетом уровня критичности последствий реализации УБИ:
  - высокий
  - средний
  - низкий
- пересматривается в ходе проведения анализа безопасности

Механизмы защиты ОС

- применимы для нейтрализации более 150 УБИ
- оцениваются по уровню эффективности
- развиваются для нейтрализации УБИ с учетом актуальной МУ
- составляют поверхность атаки

Основные компоненты СЗИ ОС

МРД - мандатное управление доступом
(защита информации различных уровней конфиденциальности)

МКЦ - мандатный контроль целостности
(защита целостности программной среды, в т.ч. от вирусов и закладок)

ЗПС - замкнутая программная среда
(защита от модификации и подмены приложений)

Режим «Киоск»
(«белый» список разрешённых к запуску приложений)

Адаптированная контейнерная виртуализация
(контроль целостности, проверка уязвимостей, организация «песочницы»)

Защищенная среда виртуализации
(изоляция процессов, контроль целостности, поддержка МРД, интеграция с МКЦ)

Защищенная СУБД
(контроль целостности, ролевое управление доступом, поддержка МРД)

Защищенная графическая среда

Контроль съемных носителей

Защищенный web-сервер и сервер почты

...

Дискреционное управление доступом
(защита информации одного уровня конфиденциальности)

Идентификация и аутентификация

Расширенный аудит событий безопасности

Блокировка интерпретаторов
(включая bash и макросы)

Регламентный контроль целостности

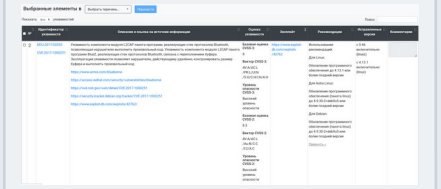
Ядро ОС с интегрированной поддержкой комплекса СЗИ

Методология разработки СЗИ как безопасного ПО

Научное и методическое обеспечение

Развитие нормативной базы разработки и обеспечения доверия к ПО

Непрерывный анализ уязвимостей в ПО

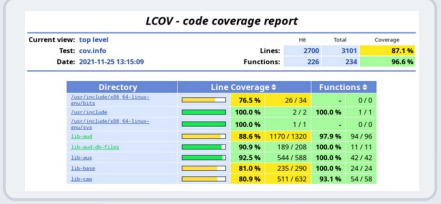


Разработка, развитие и верификация МРОСЛ ДП-модели

Сбор и аналитическая обработка результатов анализа программного кода

Разработка технологий МО (ML) и ИИ, перспективных для применения в СЗИ

Статический и динамический анализ программного кода, его верификация



Конструирование, макетирование и специфицирование перспективных СЗИ

5

Включаем в состав дистрибутива?

- Анализ зависимостей и лицензий
- Архитектурный анализ
- Анализ влияния на поверхность атаки
- Анализ влияния на ФБ
- Анализ возможности интеграции с СЗИ
- Анализ качества сопровождения проекта

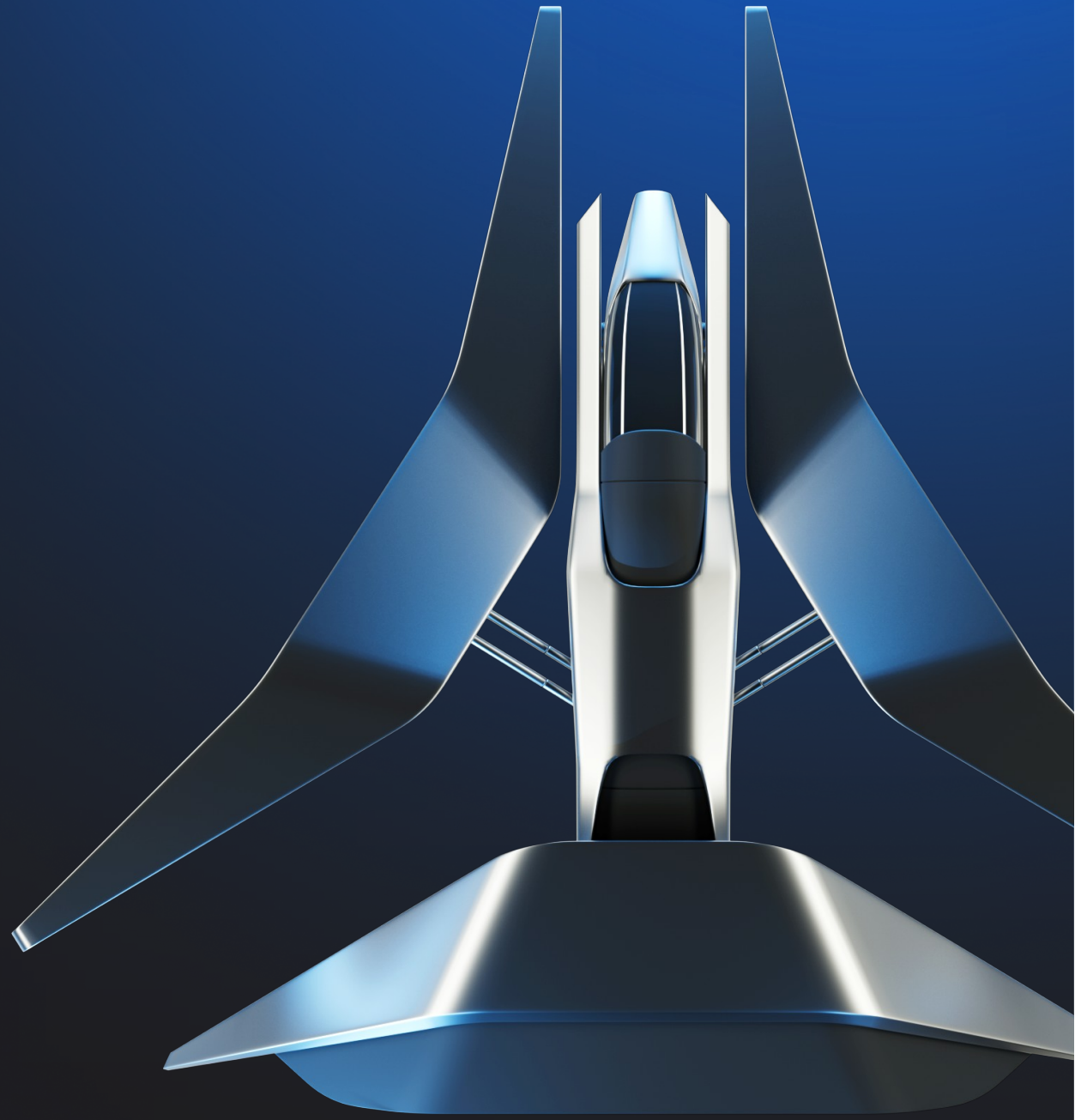
Управление уязвимостями

- Постоянный контроль
- Приоритизация уязвимостей
- Подготовка машиночитаемых форматов данных
- Подготовка МР по нейтрализации УБИ
- Взаимодействие с БДУ ФСТЭК России
- Взаимодействие с вендорами VM-решений

oval-описания:

пример применения:

Статический анализ



Статический анализ. Чем?

- Качество детектирования ошибок
- Гибкость конфигурации
- Скорость сканирования
- Возможность доработки правил
- Возможность переноса разметки
- База агрегации результатов

Статический анализ. Как?

Разработчики

Security-чемпион

Проект №1

Анализ кода

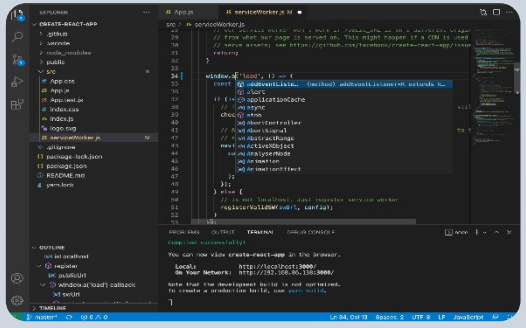
```
Comparing results:
Found new markers: 2
      HANDLE_LEAK.EX: 1
      UNCHECKED_FUNC_RES.STAT: 1
Found missing markers: 10
      DEREF_AFTER_NULL: 1
      DEREF_OF_NULL.EX: 1
      DEREF_OF_NULL.RET.STAT: 1
      TAINTED_PTR: 1
      UNCHECKED_FUNC_RES.LIB.STRICT: 3
      UNCHECKED_FUNC_RES.STAT: 3
```

Подготовка анализаторов

C++

SVACE

Разметка срабатываний



Разметка срабатываний

- CRITICAL (детекторов: 2, маркеров: 4)
  - DEREF_AFTER_NULL.EX (3)
    - ls.c : 1603
    - setparsec.c : 352
    - pdp-ls.c : 1627
  - DEREF_OF_NULL.EX (1)
    - tcpip6_mac.c : 614
- MAJOR (детекторов: 2, маркеров: 3)
  - INTEGER_OVERFLOW (2)
    - tcpip_mac.c : 156

AppSec

Взаимодействие с сообществом

Центр исследований безопасности системного ПО

30+ компаний

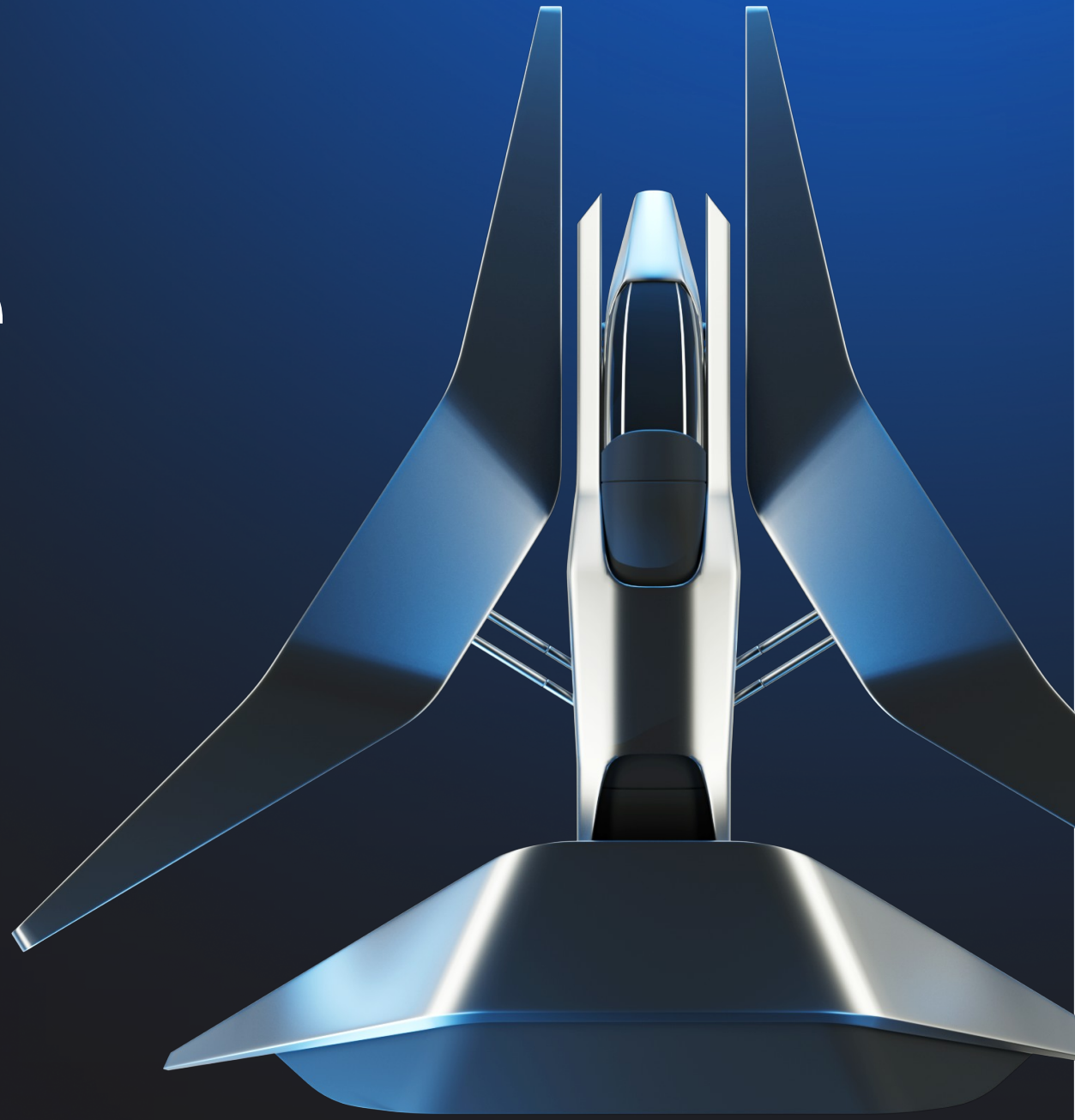
ИСП РАН

База Данных Доверия

Зоны ответственности:

- команда разработки
- AppSec

Фаззинг-тестирование



Фаззинг. Чем?

Качество генерационных и мутационных алгоритмов

Гибкость конфигурации

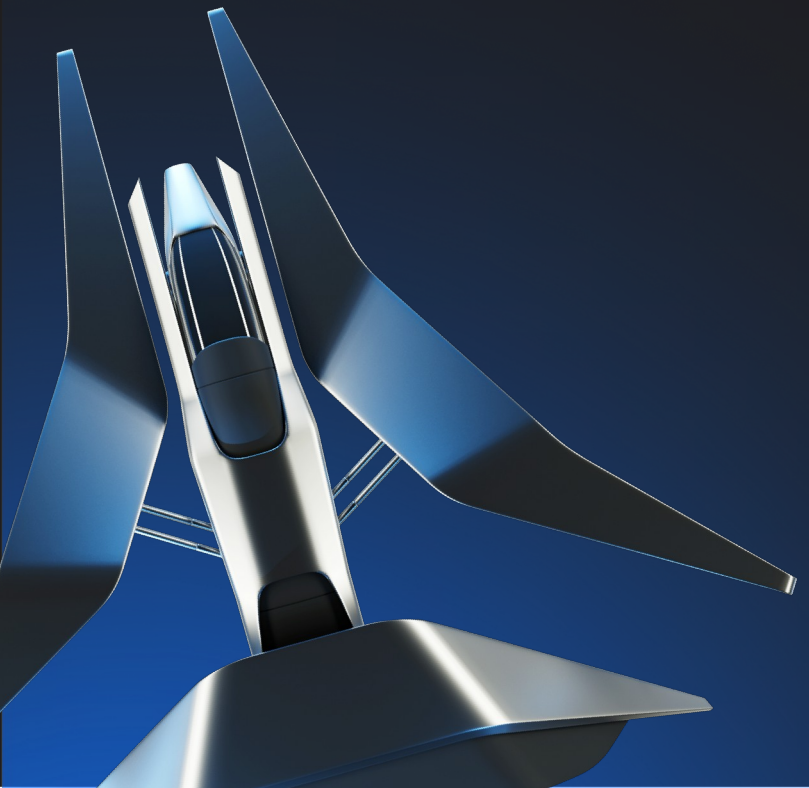
Скорость тестирования

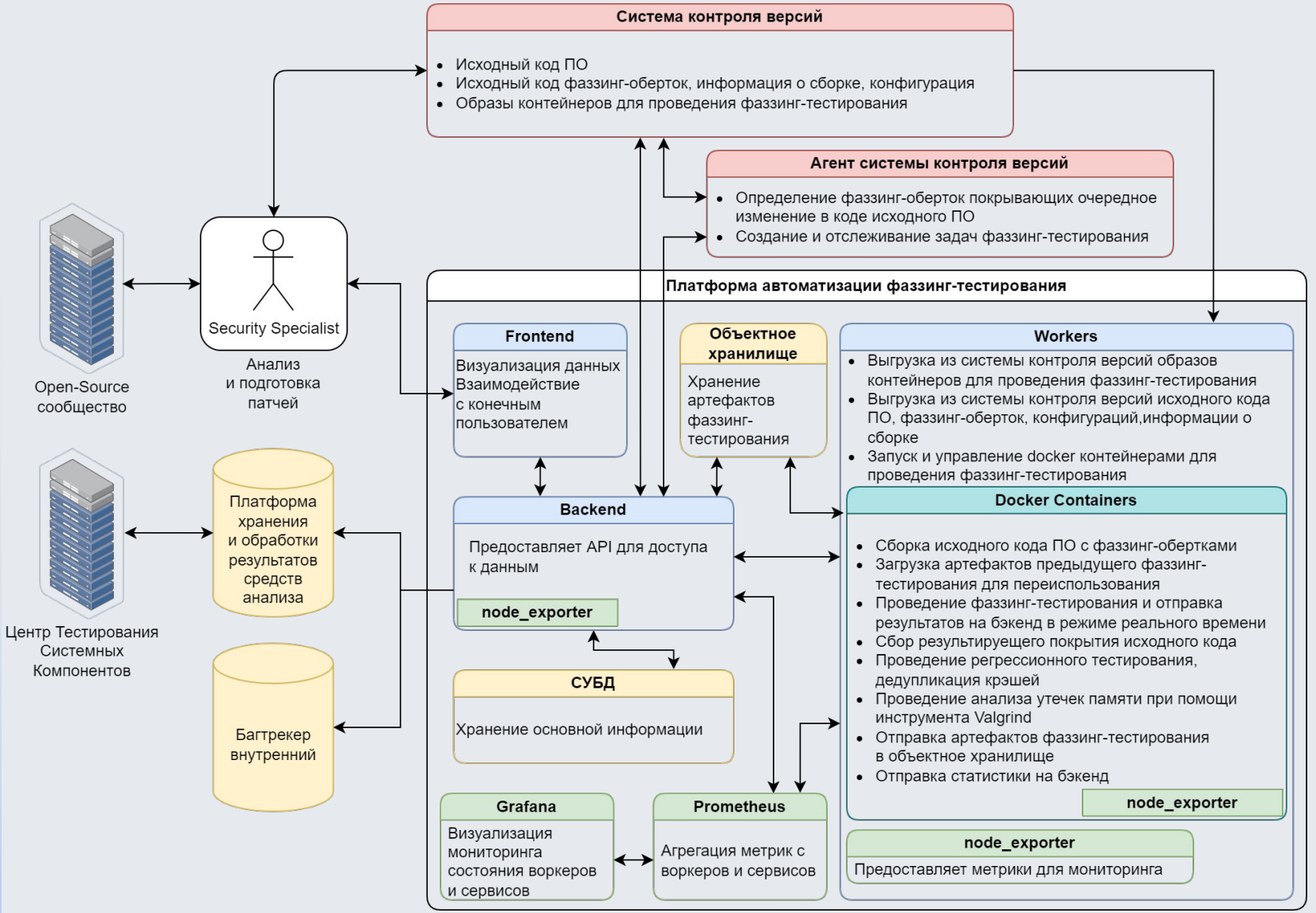
Интеграция с другими инструментами динамического анализа

Обратная связь по покрытию кода

База агрегации результатов

Фаззинг. Как?

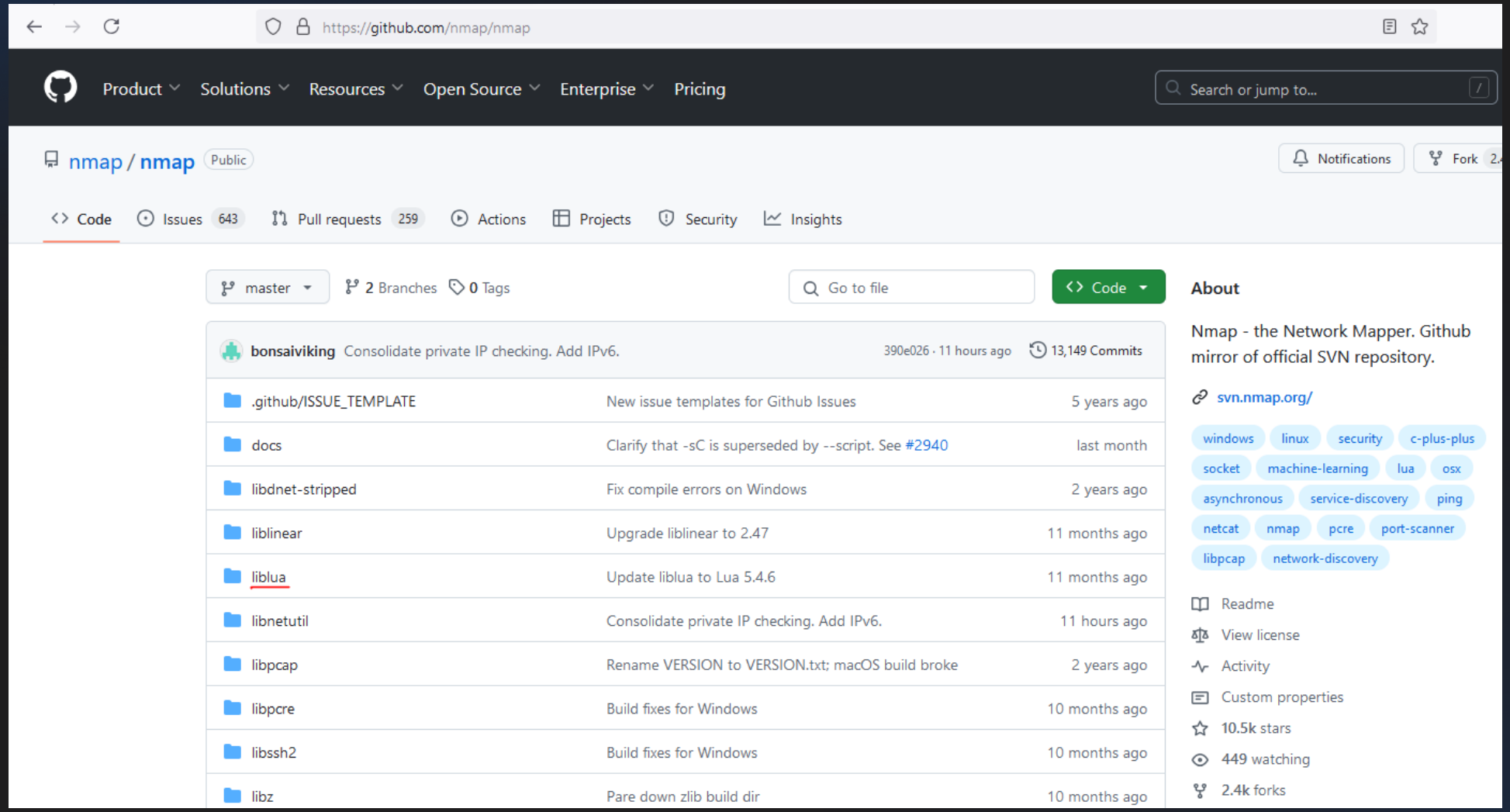



Система контроля версий
Исходный код ПО
Исходный код фаззинг-оберток, информация о сборке, конфигурация
Образы контейнеров для проведения фаззинг-тестирования
Агент системы контроля версий
Определение фаззинг-оберток покрывающих очередное изменение в коде исходного ПО
Создание и отслеживание задач фаззинг-тестирования
Платформа автоматизации фаззинг-тестирования
Open-Source сообщество
Security Specialist
Анализ и подготовка патчей
Frontend
Визуализация данных Взаимодействие с конечным пользователем
Объектное хранилище
Хранение артефактов фаззинг-тестирования
Workers
Выгрузка из системы контроля версий образов контейнеров для проведения фаззинг-тестирования
Выгрузка из системы контроля версий исходного кода ПО, фаззинг-оберток, конфигураций,информации о сборке
Запуск и управление docker контейнерами для проведения фаззинг-тестирования
Backend
Предоставляет API для доступа к данным
node_exporter
Docker Containers
Сборка исходного кода ПО с фаззинг-обертками
Загрузка артефактов предыдущего фаззинг-тестирования для переиспользования
Проведение фаззинг-тестирования и отправка результатов на бэкенд в режиме реального времени
Сбор результирующего покрытия исходного кода
Проведение регрессионного тестирования, дедупликация крэшей
Проведение анализа утечек памяти при помощи инструмента Valgrind
Отправка артефактов фаззинг-тестирования в объектное хранилище
Отправка статистики на бэкенд
node_exporter
Платформа хранения и обработки результатов средств анализа
Центр Тестирования Системных Компонентов
СУБД
Хранение основной информации
Багтрекер внутренний
Grafana
Визуализация мониторинга состояния воркеров и сервисов
Prometheus
Агрегация метрик с воркеров и сервисов
node_exporter
Предоставляет метрики для мониторинга

Не только программные ошибки

- Минимизация полномочий
- Настройка конфигураций
- Настройка компиляторов
- Харденинг ядра и иного ПО
- Выявление нежелательного контента
- Развитие СЗИ

Выявление «интересных» зависимостей



Выявление «интересных» зависимостей

```
"main.o",
"-lnsock",
"-lnbase",
"libpcre/.libs/libpcre2-8.a",
"libpcap/libpcap.a",
"-lz",
"libnetutil/libnetutil.a",
"./libdnet-stripped/src/.libs/libdnet.a",
"./liblua/liblua.a",
"./liblinear/liblinear.a",
"-ldl",
"-lstdc++",
"-lm",
"-lgcc_s",
"-lgcc",
"-lc",
"-lgcc_s",
"-lgcc",
"/usr/lib/gcc/x86_64-linux-gnu/8/crtend.o",
"/usr/lib/gcc/x86_64-linux-gnu/8/../../../x86_64-linux-gnu/crtn.o"
],
```

«?» - SCA

«?» - анализ конфигурации сборки ПО

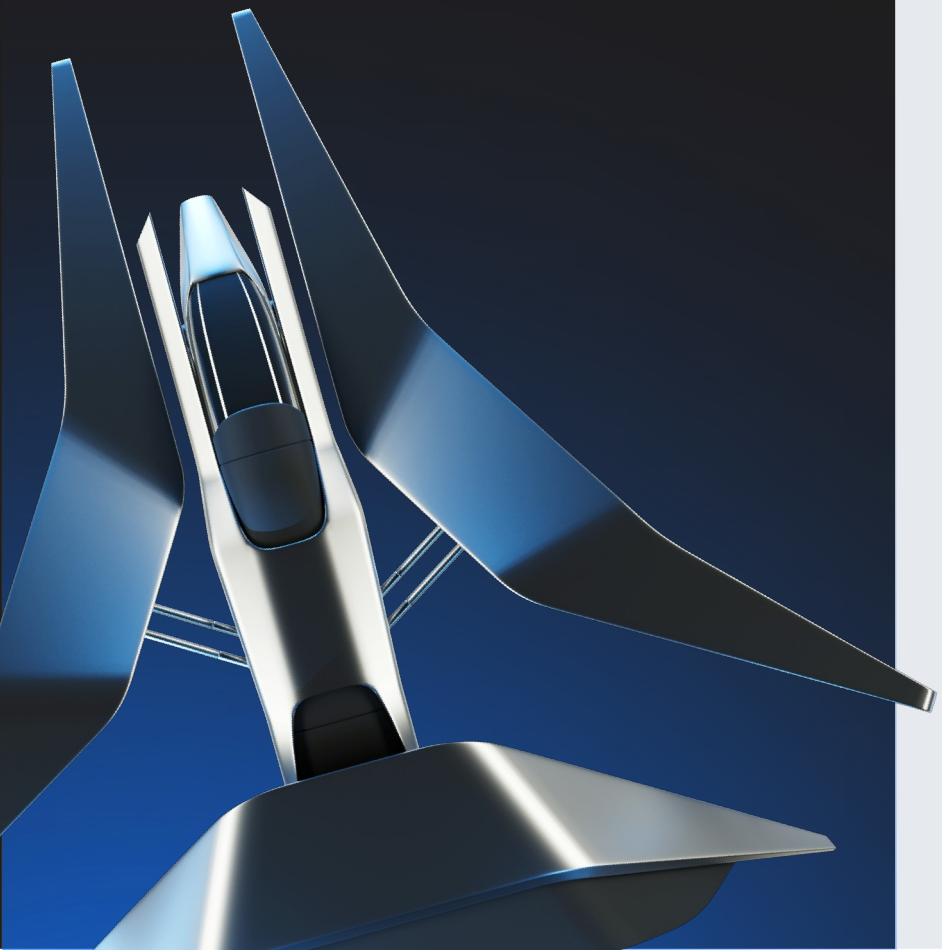
«+» - контроль сборки, сбор логов и их анализ

«+» - buildography: инструмент идентификации реквизитов сборки (ИСП РАН)

Блокировка интерпретаторов

- Исключение «лишних» зависимостей
- Удаление «избыточного» функционала
- Блокировка квази-интерпретаторов
- Интеграция с механизмом ЗПС
- Исключение из состава ОС
- Доработка самих интерпретаторов

Выявление нежелательного контента



Пропаганда

- pgadmin4-server
- gufw
- fbreader
- libjs-backbone
- gitlab-1st
- ...

«18+ и онлайн-казино»

- python-matplotlib-data
- xterm
- pcsc-tools
- xmlrpc-epi
- vim-runtime
- ansible
- ...

Ссылки на вредоносные ресурсы

- libimage-exiftool-perl
- python3-mutagen
- ruby3.1
- ...

Аномальная сетевая активность

- pgadmin4-desktop
- ltsp-cluster-nxloadbalancer
- adwaita-icon-theme
- ...
-

Основные инструментальные средства анализа

АСТРА

Композиционный и статический анализ:

- Syft, Trivy, **ProtoPack, AVM, VulScan**
- Svace
- ClangSA
- AppScreener
- AK-BC 3
- CodeQl, semgrep, shellcheck

Динамический анализ:

- **Санитайзеры:** asan, lsan, kasan, ubsan, ...
- **Отладчики:** gdb, strace, ltrace, valgrind, uftrace
- **Сбор покрытия:** lcov, gcov, afl-cov
- **Фаззинг ядра:** syzkaller, **syz-ci**
- **Фаззинг:** crusher, afl++, libFuzzer, Sydr, **AutoFuzz**
- **А также:** klee, symcc, casr, ...

Средства верификации:

- Frama-C
- Verified Software Toolchain (VST)
- WP
- Why3
- Coq

А также:

- **Средства анализа ПА:** AttackSurfaceAnalyzer*, ...
- **Средства анализа помеченных данных:** «Блесна»
- **Средства тестирования на проникновение:** LinPEAS, Zap, Kali Linux*
- **Средства выявления нежелательного контента**
- **Средства антивирусного сканирования**
- **...**

Где взять ресурсы?

- Непрерывное повышение компетенций
- Взаимодействие с ЦИБ СПО
- Технологическое партнерство
- Работа с учебными организациями
- Ведение научных проектов
- Bug Bounty

СПАСИБО
ЗА ВНИМАНИЕ