AXIOM JDK

на страже безопасности Java

РБПО с AXIOM JDK

AXIOM JDK

Задаем направление Java

AXIOM JDK

1995-2010

Проделали большой путь:
от Java1.0a2 до создания центра разработки в Санкт-Петербурге

2017

Создание команды AXIOM JDK

2018

AXIOM JDK PRO

- Выпуск российской платформы Java.
- 100% исходного кода Java SE открыто в OpenJDK, включая коммерческую функциональность.

2019

Изменение лицензии Oracle: Java стала платной для коммерческого использования

2022

Российская Java в ответственных системах:

- Карты «МИР»,
- СБП,
- 1С,
- СПбМТСБ,
- ЕДИНЫЙ ЦУПИС

2023

- Отечественная сертификация Java-специалистов.
- Платформа для Kaspersky.
- Выход на широкий корпоративный рынок.

2024

ЭКОСИСТЕМА
Успехи наших клиентов

2025

30 лет
Опыт разработки Java

Национальный стандарт РФ ГОСТ Р 56939-2024

"Защита информации.
Разработка безопасного программного обеспечения. Общие требования”

(введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 24 октября 2024 г. N 1504-ст)

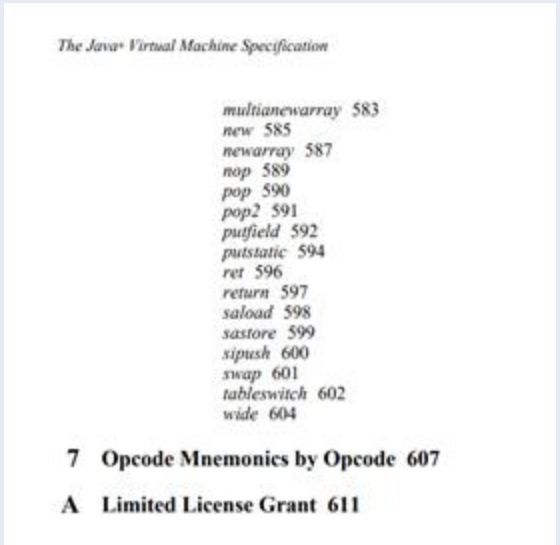
4.1 ОСНОВНОЙ ЦЕЛЬЮ РАЗРАБОТКИ БЕЗОПАСНОГО ПО ЯВЛЯЕТСЯ:

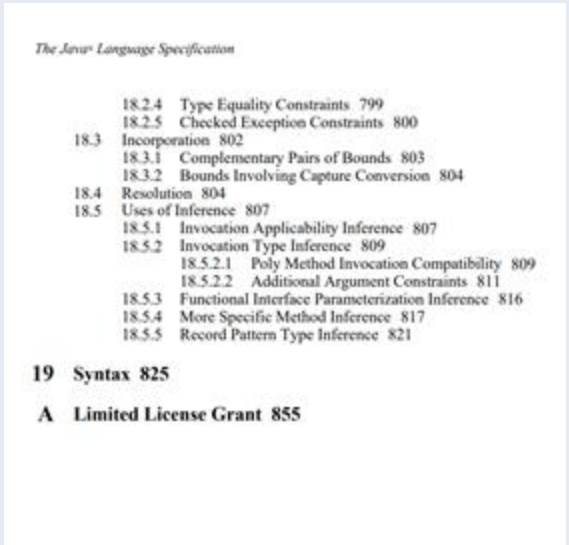
- Выявление недостатков, в том числе уязвимостей, в разрабатываемом ПО.
- Снижение количества недостатков, в том числе уязвимостей ПО.
- Снижение ущерба от невыявленных уязвимостей ПО.
- Оперативное устранение выявляемых уязвимостей в ПО.

https://base.garant.ru/410749342/#friends

AXIOM JDK

Объем спецификации продуктов





2
документа

1500+
страниц

4300+
функциональных требования для одной версии

100 000+
тестов

Знание кодовой базы продуктов – это необходимость

- Объем верифицированного исходного кода российского дистрибутива среды исполнения (JRE) составляет
  16 млн. строк
- У нас 6 LTS-релизов **94 млн. строк**
- Используем тесты для подтверждения соответствия каждому пункту спецификации

JDK – это сложнейший продукт. Всего на исследование кода ушло

100+ инженеро-лет

Продукты реализуют спецификацию, в которой
>1500 страниц,
например, JVM 23

Идентифицированная проблема:

При особой, очень редкой конфигурации параметров сервера приложений Tomcat появляется возможность обойти проверку пароля.

Решение:

Добавили выброс исключения для соответствия спецификации, устранив проблему.

Опыт больших вендоров и развитие тестовой базы

- Опыт международных компаний
- JTReg – регрессионные тесты
- **jtreg** поддерживает до 35 000 тестов, написанных для JDK
- На прогон всей кодовой базы 365 дней в год

- **Опыт международных компаний предполагает максимальное покрытие возможностей ПО тестами.**
- Соответствие спецификации выполняется с помощью внутренних тестов.
- **Регрессионные,** модульные (unit), функциональные тесты.
- **Мы используем jtreg** — набор тестов, используемый в тестовой среде OpenJDK.
- Мы учитываем **все ~~(баги)~~ ошибки** и пополняем набор регрессионных тестов, контролируем процесс исправления.
- Наш код покрыт тестами на 69%. У ядра линукс этот показатель – 39%. Мы стремимся достичь 80%.

AXIOM JDK

Продукт необходимо тестировать – это непросто

Svace ИСП РАН

SpotBugs Find bugs in Java Programs

CodeScoring

dependency track

AFL++

Jazzer

Статический анализ

- Более 100 тыс. срабатываний
- Стратегия обработки
  Critical with Very High and High Availability
  98 срабатываний
- 26 дефектов исправлены в JDK 8
- 22 дефекта исправлены в JDK 11

Композиционный анализ

- Проверка OpenSource кода

Регрессионные тесты

- Jtreg | https://openjdk.org/jtreg
- Дополняет функциональные тесты

Фаззинг и санитайзеры

- GNU Compiler Collection https://gcc.gnu.org
- AddressSanitizer, ASan утечки памяти
- UndefinedBehaviorSanitizer, UBSan

Планируем провести исследования применения ИИ автоматизации анализа результатов тестов

Проверка ПО – это технически сложный и трудоемкий процесс

AXIOM JDK

Требования ФСТЭК и что мы обнаружили

- Применение фаззинга и санитайзеров позволяет выявить скрытые дефекты, возможности обхода безопасности и повышения привилегий
- Мутация входа для генерации комбинаций
- Статический анализ + динамический анализ

Идентифицированная проблема

“Undefined Behavior Sanitizer”**:**

В C1 JIT идентифицировано знаковое переполнение значения int. На некоторых платформах оно приводит к возникновению исключений на уровне оборудования. Если его не обработать, то это приводит к ошибкам

```
runtime error: signed integer overflow:
2147483647 + 1
cannot be represented in type 'int'
```

Решение:

Добавили обработку исключения для поддержки платформ и для корректной работы компилятора C1

Комбинация методов = Результат

AXIOM JDK

AXIOM РБПО и Доверенный репозиторий

Процесс повторяется для каждого последующего обновления версий

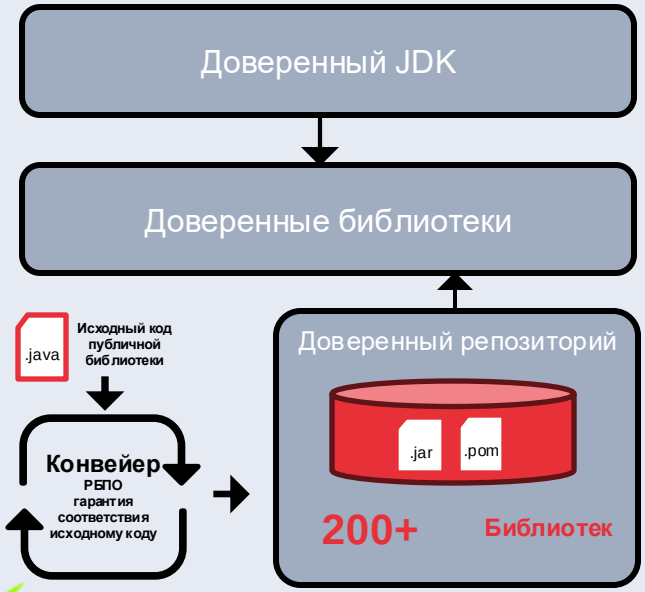
Также процесс сертификации (инспекционный контроль) для каждого обновления / релиза версий Axiom JDK Certified / Libercat Certified

AXIOM JDK

Проверенные средства сборки – это необходимость

- Средства сборки собираем с помощью нашего безопасного компилятора Java который уже прошел проверку.
- Пользуемся ими для компиляции библиотек в доверенном репозитории.

- Разработали доверенный репозиторий Java-библиотек.
- Сделали конвейер – его скорость планируется вывести на 100 библиотек в день.



Добавляем инструменты исследования кода к этому конвейеру

Gradle – собран и успешно им пользуемся. Готовы передавать его заказчиками во 2 квартале 2025 года.

Maven – частично используем, но существует проблема множества версий. Задача более сложная и мы над ней работаем.

Gradle

Apache Maven

AXIOM JDK

Автоматизация выпуска релизов

- **Классический пример: сборка происходит из исходников OpenJDK с помощью ./configure && make**
  Результат выгружается в публичные ресурсы, нет гарантии соответствия исходному коду.

- **Профессиональный подход – отказоустойчивая инфраструктура для выпуска продукта и обновлений**
  Требуется комплект оборудования и ПО с достаточным набором аппаратных ресурсов.

- **OpenJDK:** git
  jdk/jdk
  jdk/jdk21u
  jdk/jdk17u
  jdk/jdk11u
  jdk/jdk8u
  jdk/jdk7u
  jdk/jdk6u

- **Сборки:**
  

  >500 нод

- **Тесты:**
  - Функциональные
  - Регрессионные
  - Производительность

  >200 конфигураций

- **Репозитории:**
  - Linux
  - Container Registry
  - Кабинет поддержки
  - Axiom Repo
  - REST API

AXIOM JDK

Реализация режима ЗПС

Нужно гарантировать, что после разработки все файлы подписаны

Java в режиме ЗПС ОС:

- Интеграция с механизмами ОС, например, в Astra Linux.
- Проверка целостности исполняемых и *.so файлов JDK, в частности libjvm.so.

Axiom JDK Certified:

- Целостность jar и class файлов JDK.
- Целостность jar и class файлов и запускаемого пользовательского приложения.
- jar и class файлы, не подписанные надлежащим образом, не будут выполнятся.
- При включенной ЗПС запуск исполняемых файлов и загрузка исполняемых библиотек возможна только в том случае, если они подписаны электронной цифровой подписью (ЭЦП) на доверительном ключе.
- ЗПС обеспечивает защиту от загрузки произвольного исполняемого файла или библиотеки, не обладающих корректной ЭЦП.

Libercat в режиме ЗПС: (закончена реализация и сейчас идет тестирование)

Оперируемые сущности: war, ear, jar, zip, class файлы.

Проблемы:

- Libercat делает распаковку war, ear, jar на диск. Появляются неподписанные сущности.
- Ограничения Java: не поддерживаются сложные URL (со схемой “jar:jar”).

Решения:

- Создание структур в оперативной памяти вместо распаковки на диск (без подписи не исполняется).
- Реализованы специальные обработчики URL, работающие в памяти вместо обращения к файловой системе.

Последствия: падение производительности для архивов с большим количеством файлов.

AXIOM JDK

Соответствие с циклом РБПО ГОСТ

- У нас уже реализовано 23 процесса из 25 РБПО по ГОСТ

Процесс по ГОСТ	Статус в Axiom
5.1 Планирование процессов разработки безопасного программного обеспечения	Реализовано
5.2 Обучение сотрудников	Реализовано
5.3 Формирование и предъявление требований безопасности к программному обеспечению	Реализовано
5.4 Управление конфигурацией программного обеспечения	Реализовано
5.5 Управление недостатками и запросами на изменение программного обеспечения	Реализовано
5.6 Разработка, уточнение и анализ архитектуры программного обеспечения	Реализовано
5.7 Моделирование угроз и разработка описания поверхности атаки	Реализовано
5.8 Формирование и поддержание в актуальном состоянии правил кодирования	Реализовано
5.9 Экспертиза исходного кода	Реализовано
5.10 Статический анализ исходного кода	Реализовано
5.11 Динамический анализ кода программы	Реализовано
5.12 Использование безопасной системы сборки программного обеспечения	Реализовано
5.13 Обеспечение безопасности сборочной среды программного обеспечения	Реализовано
5.14 Управление доступом и контроль целостности кода при разработке программного обеспечения	Реализовано

Планируем:

- Использование безопасного компилятора ИСП РАН.
- Перевод системы хранения кода на GitFlic.

Процесс по ГОСТ	Статус в Axiom
5.15 Обеспечение безопасности используемых секретов	Реализовано
5.16 Использование инструментов композиционного анализа	Реализовано
5.17 Проверка кода на предмет внедрения вредоносного программного обеспечения через цепочки поставок	Реализовано
5.18 Функциональное тестирование	Реализовано
5.19 Нефункциональное тестирование	Реализовано
5.20 Обеспечение безопасности при выпуске готовой к эксплуатации версии программного обеспечения	Реализовано
5.21 Безопасная поставка программного обеспечения пользователям	Реализовано
5.22 Обеспечение поддержки программного обеспечения при эксплуатации пользователями	Реализовано
5.23 Реагирование на информацию об уязвимостях	Реализовано
5.24 Поиск уязвимостей в программном обеспечении при эксплуатации	Не реализовано
5.25 Обеспечение безопасности при выводе программного обеспечения из эксплуатации	Не реализовано

Вывод:

Внедрение РБПО –

это процесс постоянного совершенствования

AXIOM JDK

на страже безопасности Java