

УЯЗВИМОСТИ ПОД КОНТРОЛЕМ:

Как автоматизировать анализ кода
и обеспечить безопасность разработки

ЛЕВ АРМАНШИН

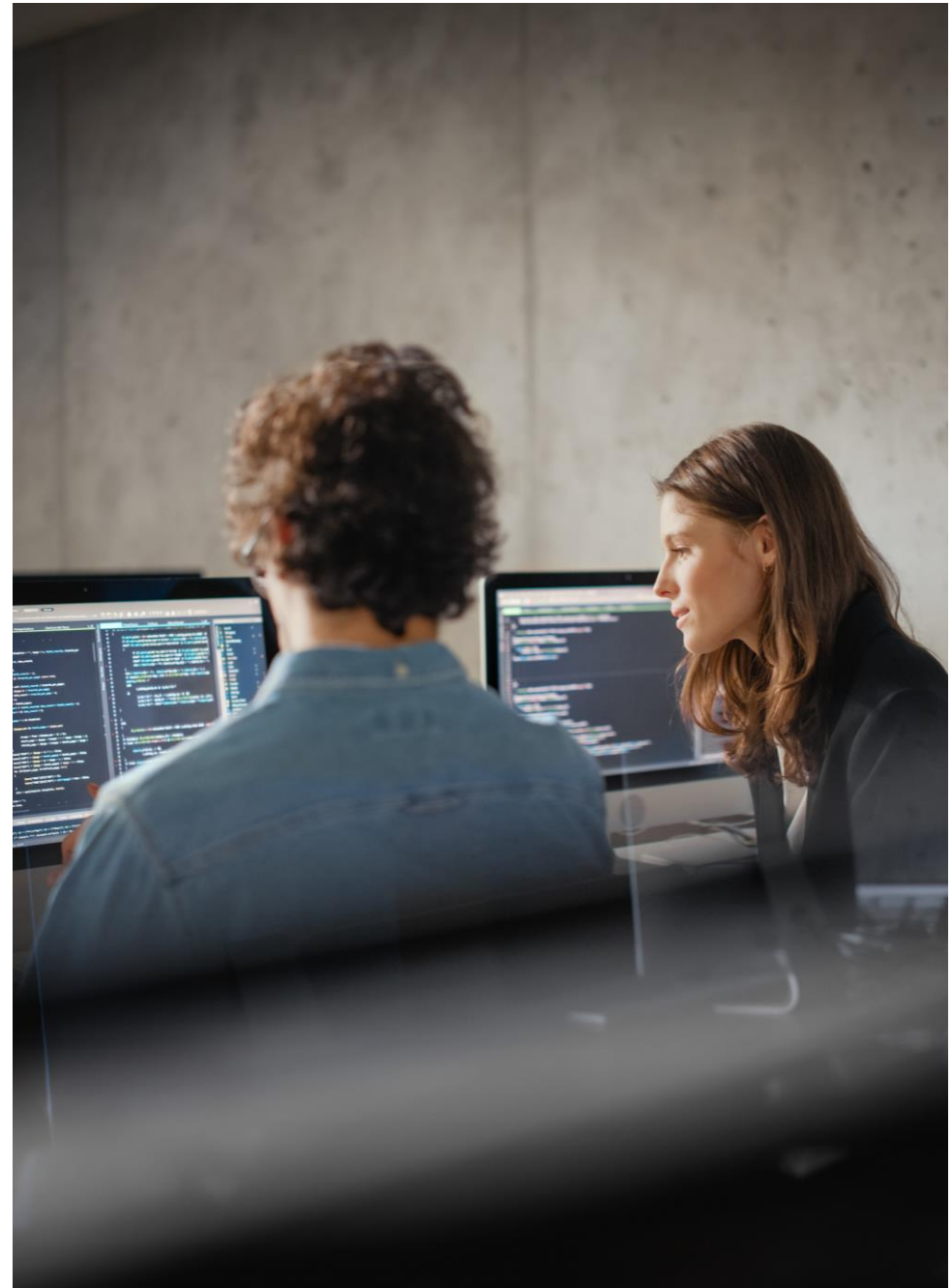
Эксперт центра разработки решений по контролю безопасности ПО
ГК «Солар»

Зачем проверять безопасность приложений

Обеспечение безопасности должно начинаться с **анализа уязвимых точек, которые могут стать началом атаки** – например, с анализа безопасности приложений.

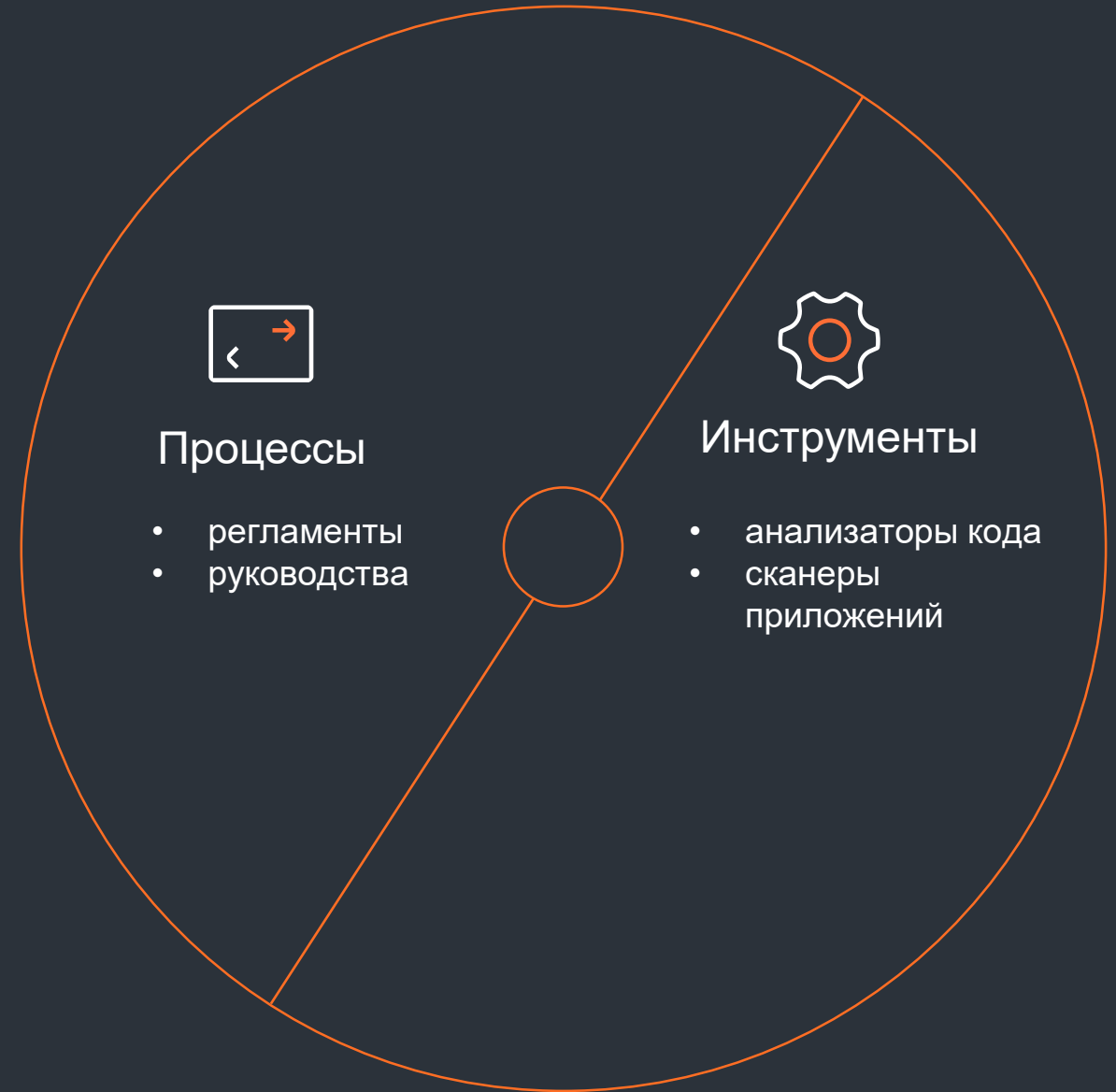
54%

инцидентов происходят из-за уязвимостей в веб-приложениях



Из чего состоит безопасная разработка

- Команда разработчиков должна быть замотивирована писать код безопасно
- Это возможно только если безопасность не тормозит процесс разработки
- Для этого нужны автоматизированные инструменты





СТАТИЧЕСКИЙ АНАЛИЗ КОДА (SAST)

Проверяет исходный или бинарный код приложения на наличие уязвимостей как на ранних, так и на более поздних этапах разработки ПО



ДИНАМИЧЕСКИЙ АНАЛИЗ КОДА (DAST)

Проверяет работающее приложение в конце жизненного цикла разработки ПО или когда оно уже установлено в инфраструктуре



АНАЛИЗ СТОРОННИХ КОМПОНЕНТОВ (OSA)

Выявляет угрозы, связанные с уязвимостями и бэкдорами в open-source-библиотеках, и другие опасности использования сторонних компонентов в коде

Статический анализ кода (SAST)

Статический анализ кода (SAST) позволяет выявить уязвимости и недеklarированные возможности в коде приложений.

[01]

Максимальное количество выявленных уязвимостей за счет анализа исходного кода

[02]

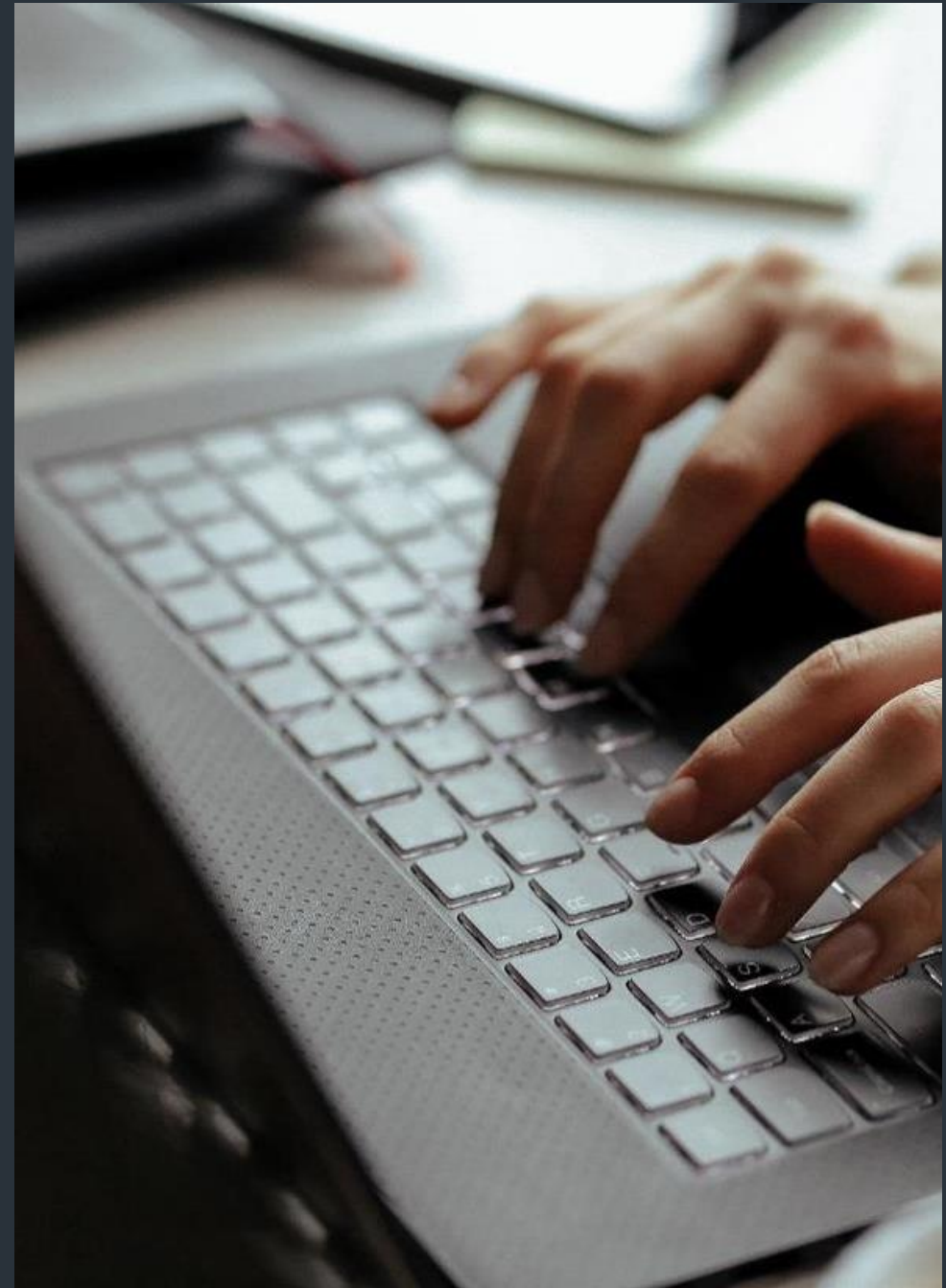
Выявление уязвимостей на ранних этапах разработки

[03]

Возможность инкрементального сканирования

[04]

Большой процент ложных срабатываний



Как работает SAST

The screenshot displays the Solar appSec Scanner interface. At the top, the navigation bar includes 'Домашняя страница', 'Проекты', 'Группы проектов', 'Правила и наборы', and 'О продукте'. The main content area shows the path 'Проекты SAST > Test Application > Подробные результаты'. A summary table indicates the scan date as '7/7 12.10.2023 13:28:31' and provides a color-coded bar for the results: 159 total findings, 15 critical, 116 medium, 25 low, and 3 info.

Всего	Критический	Средний	Низкий	Инфо
159	15	116	25	3

The left sidebar lists various vulnerabilities, with 'Внедрение в SQL-запрос' (SQL Injection) highlighted. The main panel shows the source code of 'PYTHON_INJECTION_SQL.py:6' with a detected vulnerability:

```
5 # <yes> <report> PYTHON_INJECTION_SQL c8d15e
6 store.executeSQL(query)
7 # <no> <report>
8 store.executeSQL('SELECT * FROM table')
9
10 # <yes> <report> PYTHON_INJECTION_SQL f5a2f6
```

Below the code, a detailed description of the vulnerability is provided:

Описание уязвимости | Пример | Рекомендации | Ссылки | Классификации | Трасса | Управление уязви

Возможно внедрение в SQL-коде. Злоумышленник может обойти механизм аутентификации, получить доступ ко всем записям базы данных, выполнить вредоносный код с правами приложения.

Атаки типа “внедрение кода на стороне клиента” (client side code injection) занимают первое место в рейтинге уязвимостей web-приложений OWASP Top 10 2017 и седьмое место в рейтинге OWASP Mobile Top 10 2014. Уровень возможного ущерба от такой атаки зависит от работы валидации пользовательского ввода и механизмов защиты файлов.

Внедрение SQL -кода происходит, когда запрос к базе данных формируется на основе данных из ненадёжного

Динамический анализ кода (DAST)

Позволяет выявлять уязвимости непосредственно при выполнении программы



DAST имитирует вредоносные внешние атаки, использующие распространенные уязвимости для компрометации приложения



Использование при отсутствии исходного кода



Не привязан к языкам программирования



Минимальное количество ложных срабатываний



Возможно большое количество ненайденных уязвимостей и НДВ

~20%

кода разработчики пишут сами

Open Source Analysis (OSA) - комплексный анализ для защиты от угроз, связанных с open-source-компонентами.

РЕШАЕМЫЕ ЗАДАЧИ

- Выявление всех сторонних компонентов в приложениях
- Своевременное выявление уязвимостей и бэкдоров в open-source-библиотеках
- Предотвращение угроз и снижение рисков ИБ из-за заимствования кода

РЕЛЕВАНТНЫЕ ВИДЫ АНАЛИЗА



АНАЛИЗ СОСТАВА ПО (SCA)



АНАЛИЗ ЦЕПОЧКИ ПОСТАВОК ПО (SCS)



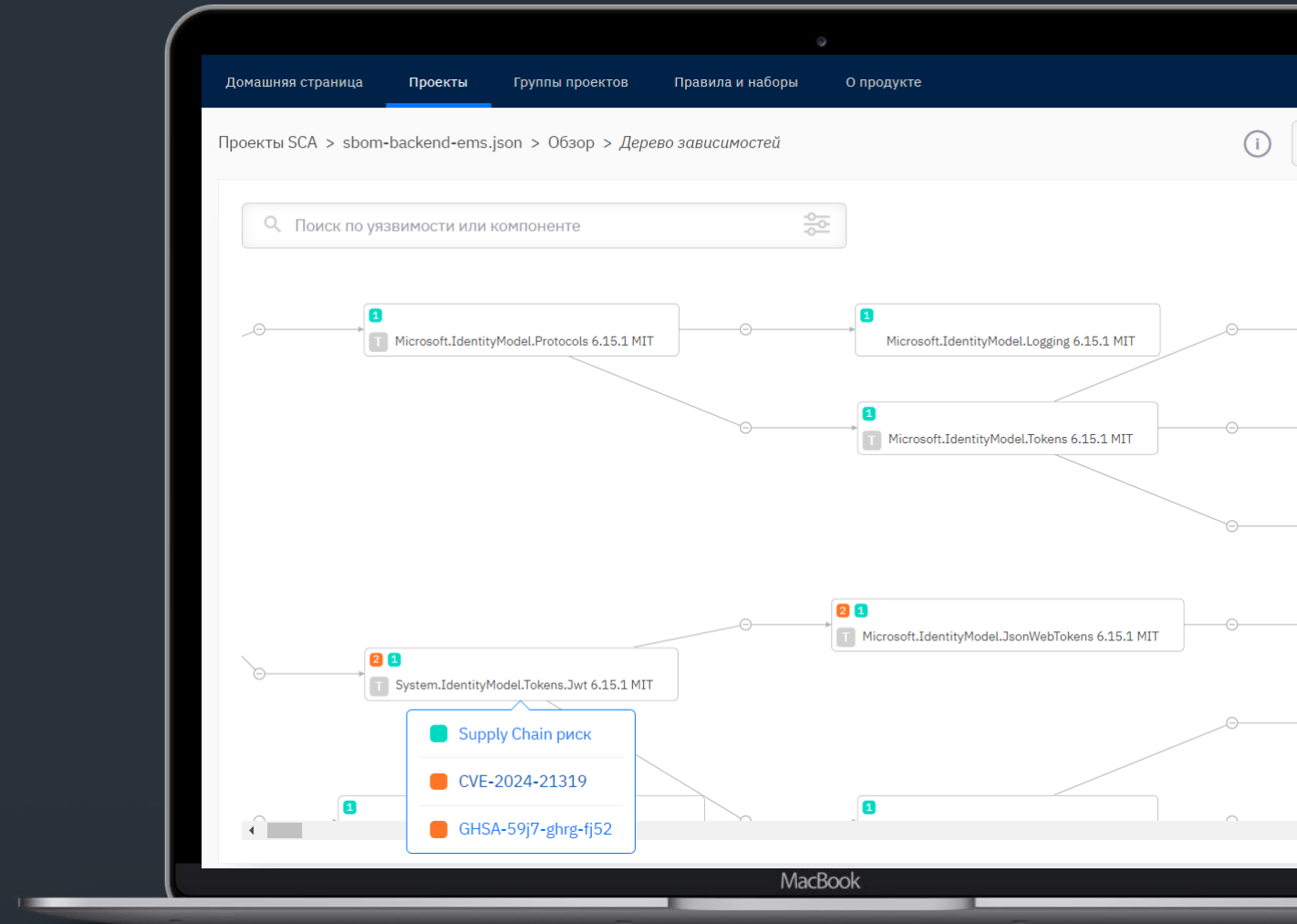
АНАЛИЗ ЛИЦЕНЗИОННЫХ РИСКОВ

Анализ состава ПО (SCA)

Анализ состава ПО (SCA) позволяет проверять безопасность сторонних библиотек, используемых разработчиками в своем ПО.

РЕШАЕМЫЕ ЗАДАЧИ:

- Обнаружение и отслеживание всех сторонних компонентов в ПО
- Выявление известных уязвимостей и закладок в сторонних библиотеках
- Предотвращение угроз и снижение рисков ИБ из-за заимствования кода



Анализ цепочки поставок ПО (SCS)

Анализ цепочки поставок (Supply Chain) выявляет потенциально опасные библиотеки, даже если известные уязвимости в них пока не обнаружены.

Скриншот интерфейса SOLAR SCA. Вверху меню: Домашняя страница, Проекты, Группы проектов, Правила и наборы, Администрирование, О продукте. В левом меню: Домашняя страница, Проекты, Группы проектов, Правила и наборы, Администрирование, О продукте. В центре: Проекты SCA > test_sbom_request.json > Подробные результаты. Вверху: Всего 186, Критический 23, Средний 58, Низкий 53, Инфо 52, 1. Поиск по уязвимости или компоненте. ВСЕ КОМПОНЕНТЫ. Список компонентов: event-pubsub 5.0.3 (Supply Chain риск), js-queue 2.0.2, cov2badge 0.1.2, vanilla-test 1.4.8, esbuild 0.12.28, js-message 1.0.7, node-cmd 4.0.0, node-http-server 8.1.4, strong-type 1.0.1, peacenotwar 9.1.5, cyclonedx-library 2.1.0, node-pre-gyp 1.0.10, semver 7.5.2. В центре: Описание уязвимости, Управление уязвимостью, Таск-менеджер. Компоненте назначена оценка: 1.4. Автор библиотеки недоверенный. Таблица метрик:

Метрика	Оценка
Популярность ⓘ	2.5
Авторский состав ⓘ	0.0
Активность сообщества ⓘ	3.0
Заинтересованность в безопасности ⓘ	Нет
Библиотека создана недавно ⓘ	Нет
Первая версия подозрительно высокая ⓘ	Нет
Единственный проект этого разработчика ⓘ	Нет
% пул-реквестов не выполняют требование рецензии от двух человек ⓘ	0 %

РЕШАЕМЫЕ ЗАДАЧИ:

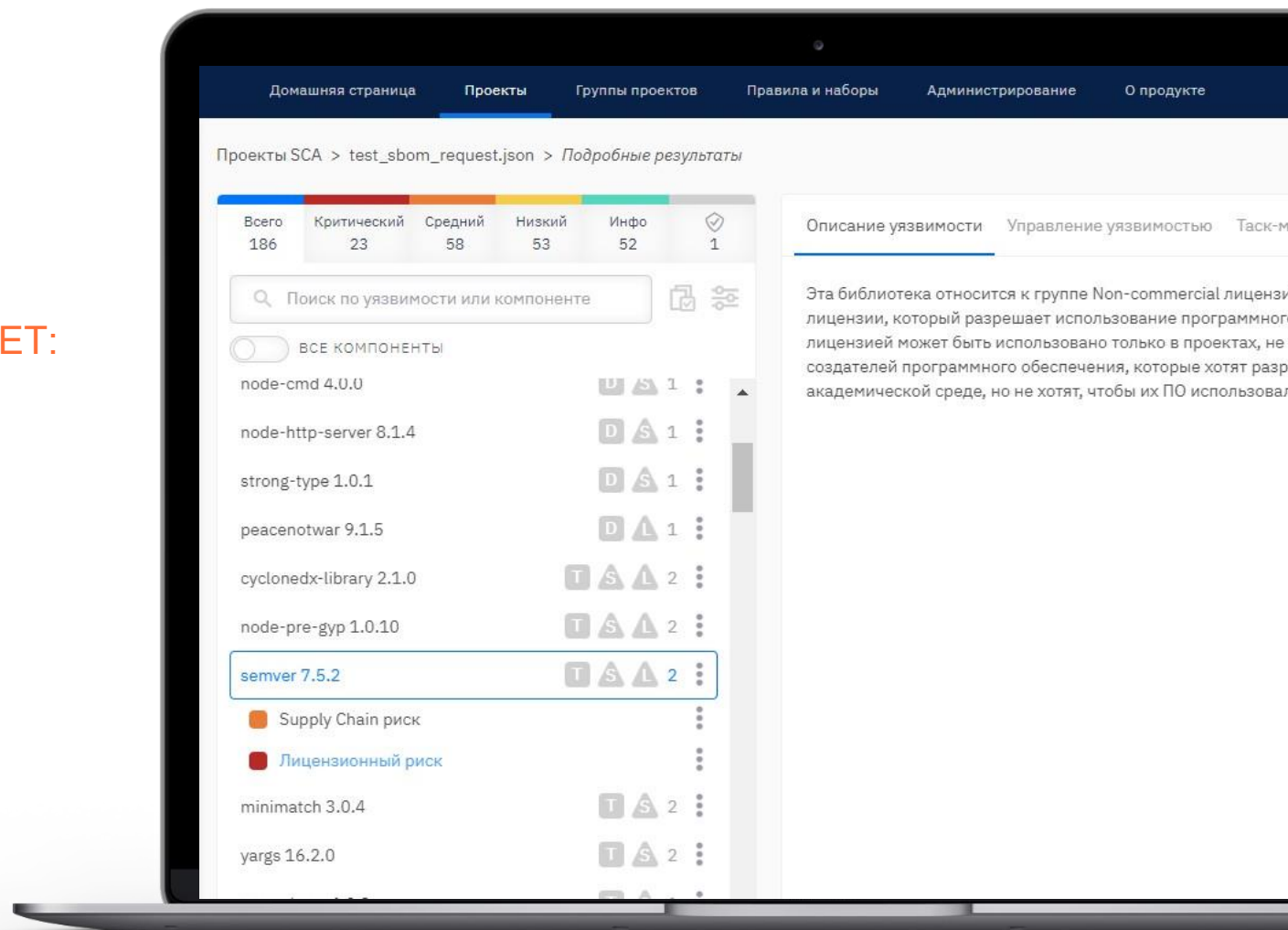
- Мониторинг подозрительной активности в используемых сторонних библиотеках
- Снижение рисков ИБ из-за использования заимствованного кода
- Защита от атак supply chain



Не весь open source распространяется свободно!

ОЦЕНКА ЛИЦЕНЗИОННЫХ РИСКОВ ОБЕСПЕЧИВАЕТ:

- Мониторинг лицензионной политики стороннего ПО в вашем приложении
- Снижение юридических рисков, связанных с лицензированием open source: отсутствие лицензии, несовместимость лицензий разных компонентов и т. д.
- Рекомендации по устранению найденных угроз



SOLAR APPSCREENER



СВЯЗЬ МОДУЛЕЙ



Корреляция SAST и DAST:

Результаты статического анализа можно верифицировать с помощью DAST.

Это помогает выделить приоритетные уязвимости и устранить их в первую очередь



Комбинированный анализ OSA и SAST:

SAST проверяет, представляют ли угрозу уязвимости, найденные OSA-анализом.

Это сокращает ложные срабатывания и выявляет только истинные уязвимости.

Интеграция со средствами разработки

РЕПОЗИТОРИИ



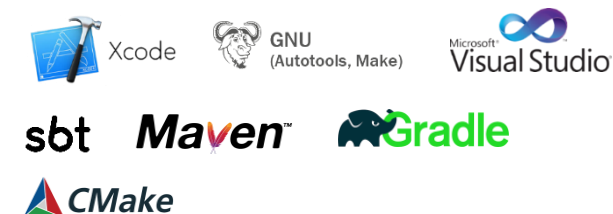
VCS-ХОСТИНГИ



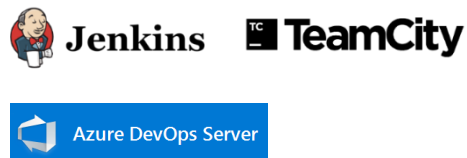
СРЕДСТВА РАЗРАБОТКИ



СРЕДСТВА СБОРКИ



СЕРВЕРЫ CI/CD



ОТСЛЕЖИВАНИЕ ЗАДАЧ

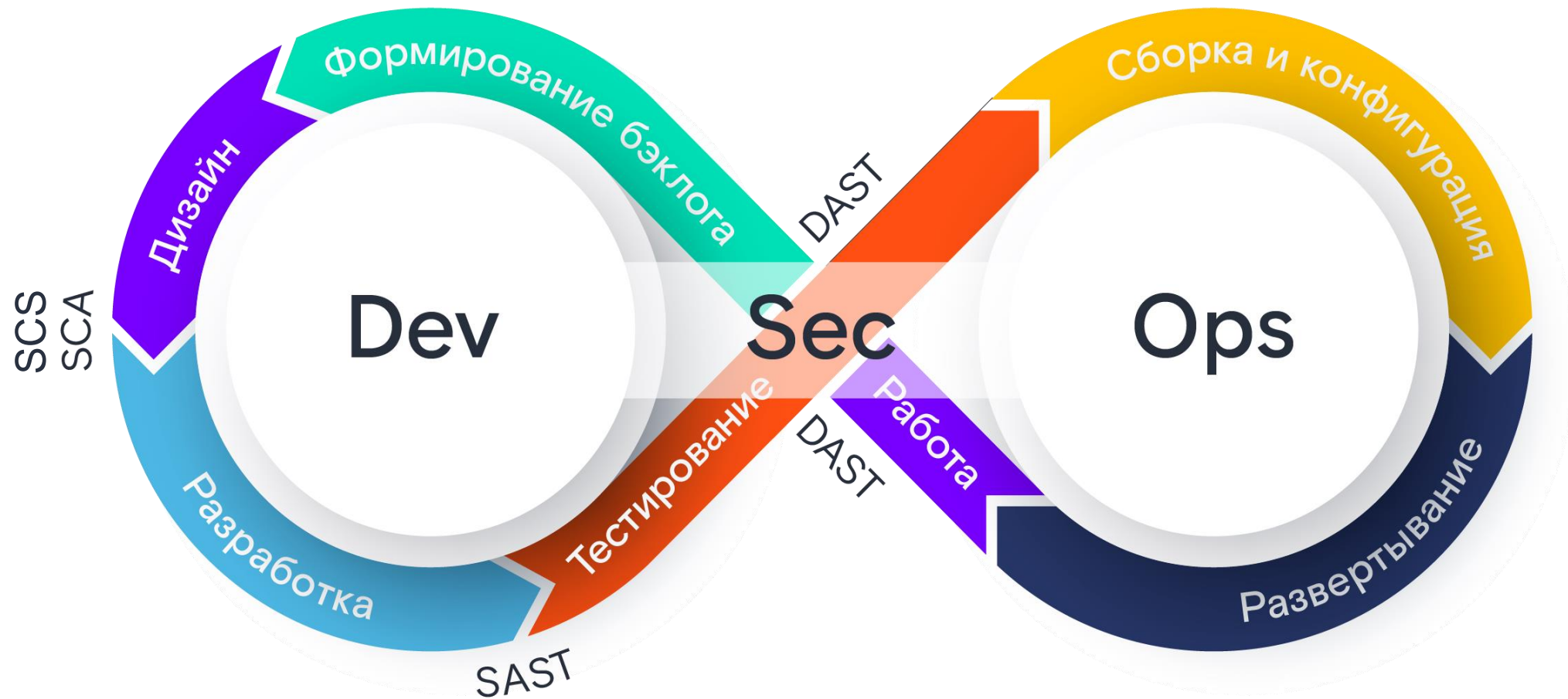


АНАЛИЗ КОДА



Широкие возможности по дополнительной интеграции и автоматизации благодаря **открытому API**.
Включает в себя **JSON API** и **CLI**.

Как встроить инструменты в цикл безопасной разработки



Подход Solar appScreener

Solar appScreener – комплексное решение для контроля безопасности ПО, которое включает технологии статического (SAST), динамического анализа (DAST) и анализа сторонних компонентов (OSA).



Сочетает преимущества всех ключевых методов анализа в одном решении



Позволяет быстро и удобно управлять сканированиями из единого интерфейса



Автоматизирует анализ кода, чтобы освободить ресурсы ваших специалистов и сэкономить время



Удобно встраивается в цикл разработки ПО



Бесплатно проверьте ваш код с Solar appScreener

СПАСИБО ЗА ВНИМАНИЕ!



ЛЕВ АРМАНШИН

Эксперт центра разработки решений по контролю
безопасности ПО, ГК «Солар»

l.armanshin@rt-solar.ru