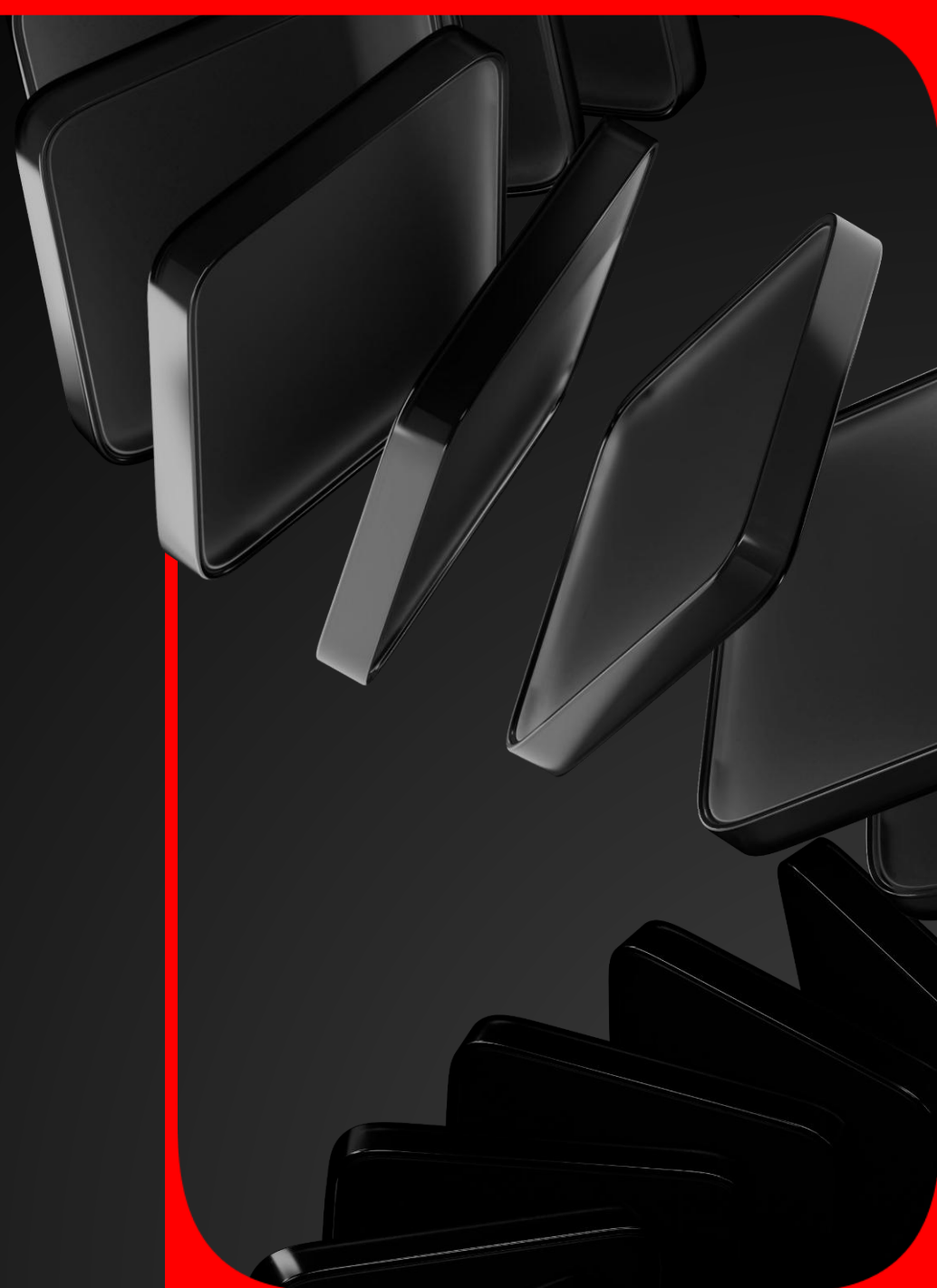




PRODUCT
BACKSTAGE



PT Application Inspector



PT Application Inspector



AppSec-платформа для управления рисками безопасности приложений, процессами безопасной разработки и цепочек поставок ПО



SAST

Высокое качество и достоверность результатов сканирования исходного кода приложения за счет большого набора и комбинации алгоритмов анализа.

Поддержка языков: Java, Scala, C#, PHP, JavaScript, TypeScript, Python, C/, C++, Go, Ruby, Kotlin, Objective-C, Swift, SQL, Solidity, 1C



DAST

Корреляция результатов статического и динамического анализа помогает быстрее выявлять действительно опасные уязвимости, не тратя время на проверку ложных срабатываний.



SCA

Находит уязвимости в зависимостях и проверяет, реально ли уязвимая функция вызывается в коде. Меньше шума, точнее приоритизация

01

PRODUCT
BACKSTAGE



Что нового в релизе 6.0

Новая архитектура платформы



ЧТО ИЗМЕНИЛОСЬ

- ✓ Разделение БД между сервисами
- ✓ Декомпозиция backend-компонентов
- ✓ Единый installer для Linux и Windows
- ✓ Унификация agent runtime
- ✓ Стандартизация adapter protocol

НОВАЯ АРХИТЕКТУРА ПЛАТФОРМЫ



Анализ конфигурационных файлов



> Независимость от формата

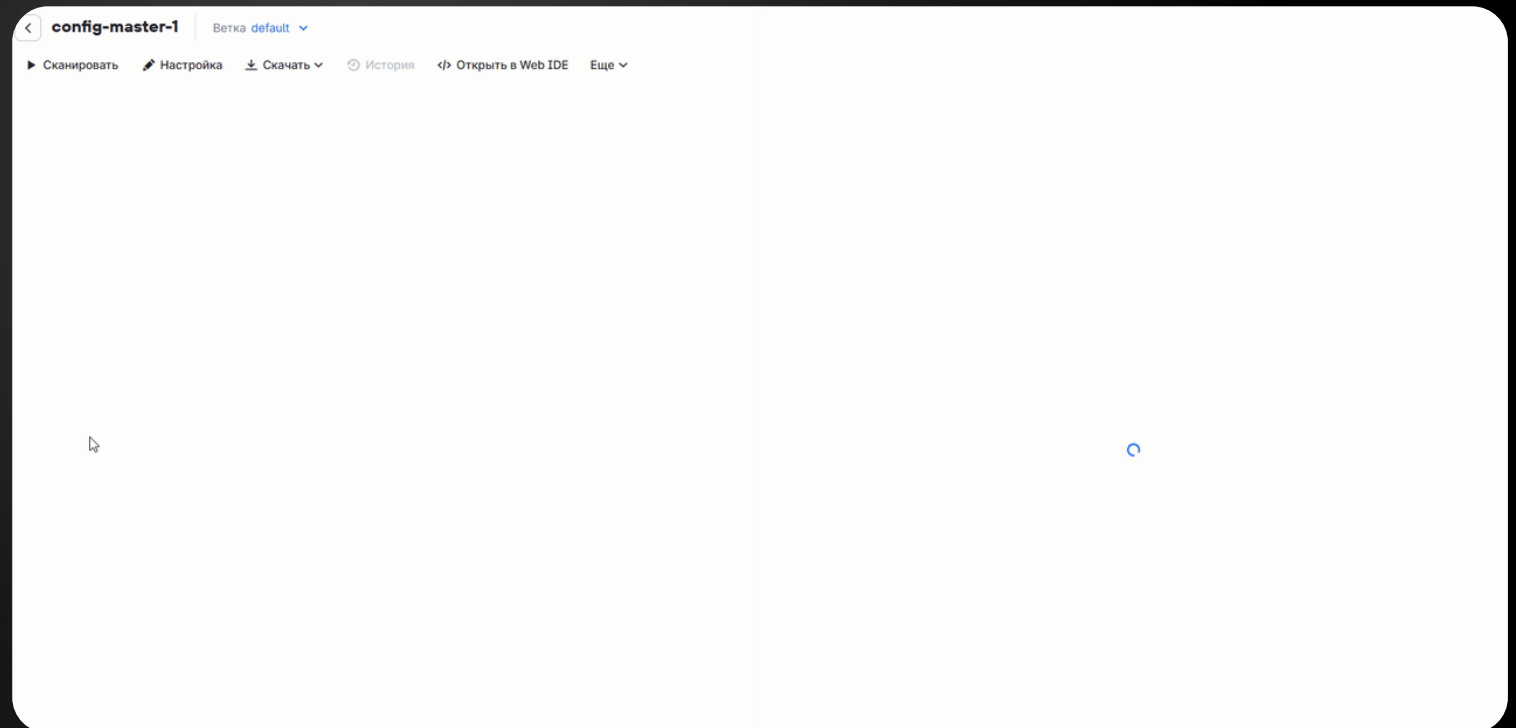
Единый подход к анализу разных типов конфигурационных файлов

> Единый механизм анализа

Общий конвейер обработки и единое представление данных

> XPath-правила

Гибкие и мощные правила для поиска уязвимостей и ошибок



Поиск секретов в коде



Модуль
не пытается заменить

Сканеры секретов для инфраструктуры

Поиск в артефактах

Технологии энтропийного анализа



Цель — реальные проблемы
в логике приложения

Меньше ложных срабатываний

Меньше ручной перепроверки

Проверка совместно с JSA

Поиск секретов в коде



Анализ вместо поиска по шаблонам

Жестко заданные пароли

Секреты в чувствительных вызовах

Конфиденциальные данные в коде



Секрет используется в коде —
реально опасен

```
55
56 // main file
57 import otherModule as other;
58
59 func main(args: string[]) {
60
61     obj->field = "o=*e31iu2&_893NNa"
62
63     other.prepareConnection(obj);
64
65 }
```

```
12 ...
13 ...
14 ...
15 func prepareConnection(input: Object) {
16     DriverManager.getConnection("http://localhost:3000", "root", obj->field)
17 }
18 ...
19 ...
20 ...
```

Источник секрета
obj->field:61

Передача в функцию
prepareConnection:63

Использование
getConnection:16

Scala



Kotlin

Приоритеты сканирования и управление очередями сканирования



Приоритеты проектов



Выделенные агенты



Ручное управление очередью



Перераспределение задач в реальном времени



Изменение порядка запуска



Переопределение приоритетов



Ручное распределение нагрузки



Теперь важные проверки не ждут в общей очереди вместе с остальными задачами. Критически важные сканирования можно запускать быстрее, когда это действительно нужно команде



PRODUCT
BACKSTAGE



ML — ПОИСК ВРЕДОНОСНОГО КОДА

molo 
■ positive technologies

ML — поиск вредоносного кода

molot



Где это особенно полезно

- Проверка сторонней разработки
- Outstaff-команды
- Open source зависимости
- Внутренние платформенные команды
- Supply chain security

Что позволяет обнаруживать

- Когику downloader и loader
- Скрытые backdoor
- Утечку данных
- Обфускацию
- Активность cryptominer
- Скрытое исполнение кода

molot показывает стабильное качество

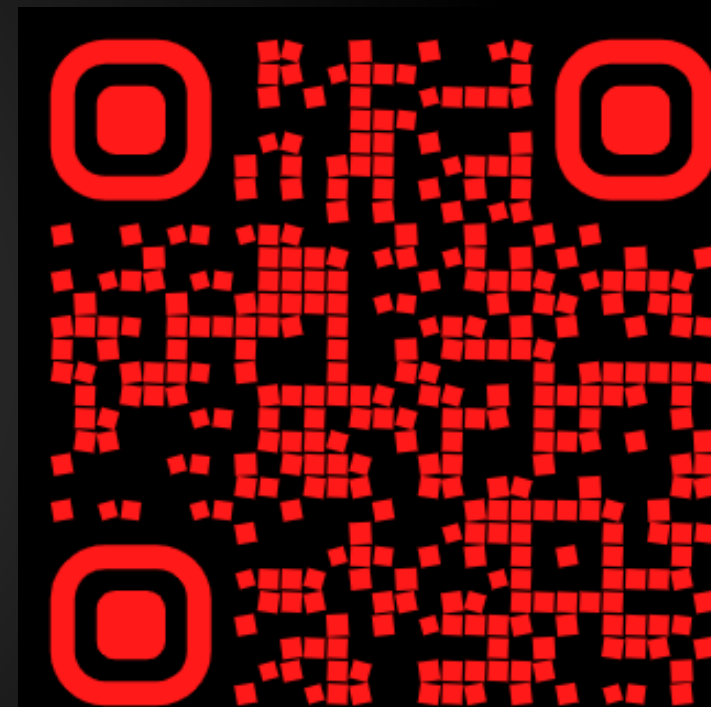


Python

ИНСТРУМЕНТ	КАЧЕСТВО
PT AI	94%
ossgadget	74%
bandit4mal	61%
semgrep	56%

JavaScript

ИНСТРУМЕНТ	КАЧЕСТВО
PT AI	94%
ossgadget	79%
semgrep	7%



800

проектов

Открытый бенчмарк

14 000

проектов

Закрытый бенчмарк

ML — поиск вредоносного кода

molot



БЕЗ ОБЛАКА.

БЕЗ LLM.

БЕЗ GPU.

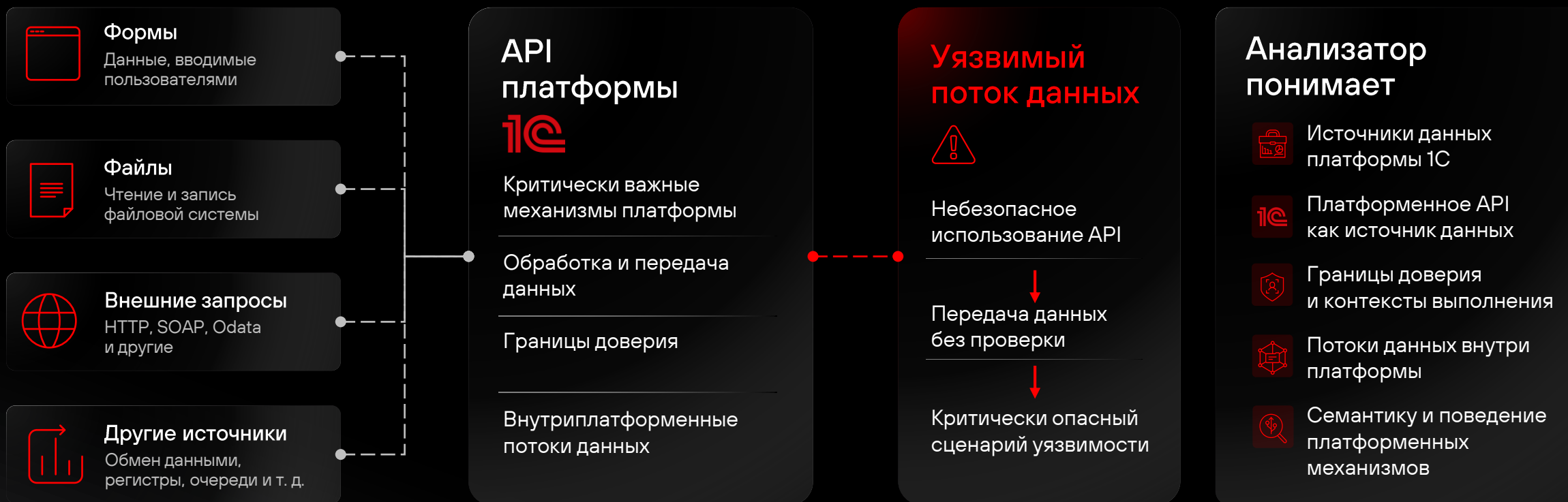
Архитектура BERT

~110M параметров

Выполнение на CPU

Полностью ON-PREM

Поддержка 1С



Понимание платформенного API 1С позволяет выявлять уязвимости

Поддержка 1С



Покрытие классов уязвимостей

ПОКАЗАТЕЛЬ	ЗНАЧЕНИЕ
Классов уязвимостей в сравнении	51
Заявлено РТ AI	28
Подтверждено конкретными находками РТ AI	28
Классы: РТ AI заявляет покрытие, у конкурентов в срезе нет	6

ПРОЕКТ	РТ AI, сек	РТ AI, VULN	КОНКУРЕНТ, сек	КОНКУРЕНТ, VULN
research-1c-synthetic	9	236	47	790
tasks	305	3338	1668	2963
IRP	106	208	360	370
SDMS	49	237	112	353
1C-PT	306	3400	1804	3164

➤ На крупных проектах РТ AI существенно быстрее

➤ Число находок сопоставимо

➤ Часть находок связана с платформенными потоками данных

Полезные ссылки



Telegram-чат
о безопасной разработке



Подробнее о продукте
PT Application Inspector



Подробнее о продукте
PT BlackBox

