Экспресс 42

История одного проекта:

встраиваем безопасность по командным топологиям

Экспресс 42

Шмыр Василий

Техлид по практикам DevOps

- 15+ лет опыта в проектировании и внедрении сложной информационной инфраструктуры в крупнейших компаниях РФ
- Веду проекты полного цикла: от архитектуры до промышленной эксплуатации
- Формирую инженерные команды и процессы поставки с нуля
- В «Экспресс 42» с 2025 года: развиваю практики поставки и оптимизирую поток ценности для заказчиков

О компании «Флант»

Экспресс 42

17+
лет опыта
в Open Source

С 2017
года используем
Kubernetes в production

№1
контрибьютор в проекты
CNCF из России

500+
сотрудников

>260
компаний-пользователей

В топе
вендоров ИТ-решений для банков*
и промышленности**

Реестр
российского ПО

Лицензии и сертификат
ФСТЭК России

АРПП «Отечественный
софт»

* Рейтинг «Крупнейшие ИТ-вендоры в банках», TAdviser, 2024
** Рейтинг «Крупнейшие ИТ-вендоры в промышленности», TAdviser, 2024

ФЛАНТ

Синергия опыта вендора, интегратора, сервисной и консалтинговой компании

Deckhouse

Deckhouse — продуктовое подразделение, разработчик продуктов для построения надёжной enterprise-инфраструктуры

DaaS

DaaS — комплексное DevOps-сопровождение инфраструктуры в режиме 24/7 силами выделенной DevOps-команды

Экспресс 42

«Экспресс 42» — DevOps-консалтинг. От анализа узких мест в ИТ-процессах до создания роадмапа изменения ИТ для реализации цифровой трансформации

Контекст

Главный вызов:

ИБ сегодня — безусловный приоритет, но бывает так, что безопасность становится главным «узким местом» для цифрового бизнеса

Решение:

Мы нашли ответ в методологии Team Topologies

О чём поговорим:

01 Почему ИБ и разработка говорят на разных языках?

02 Как лечить организационные конфликты через архитектуру команд?

03 Реальный кейс: как падение одного сервиса на 12 часов заставило пересобрать ИБ с нуля

Параллельные системы координат

Во многих ИТ-организациях ИБ и производство живут в параллельных системах координат, которые слабо пересекаются

Реальность ИБ

Метрики привязаны к compliance-чек-листам, регуляторным требованиям, количеству закрытых уязвимостей и минимизации рисков любой ценой

Реальность цифрового производства

Метрики привязаны к Time-to-Market, скорости поставки фич, SLA инфраструктуры и стабильности CI/CD-пайплайна

Системный дефект

Без чётких организационных границ ИБ вынужденно работает как «пожарная команда»: тушит локальные инциденты на релизах вместо проектирования системно защищённой архитектуры

Роль ИБ в цифровом бизнесе

Безопасность больше не является поддерживающей функцией.
Сегодня это базовое условие непрерывности бизнеса. Любой критический инцидент, связанный с ИБ, любая успешная атака злоумышленников приносят прямые коммерческие убытки

Репутационный коллапс

Мгновенный отток клиентов
из-за потери доверия к бренду

Финансовые потери

Прямые потери от простоя цифровых сервисов, штрафы регуляторов и затраты на ликвидацию последствий

Точка невозврата

Риск полной остановки жизни продукта или закрытия компании

Главный конфликт и вызов

Продукт

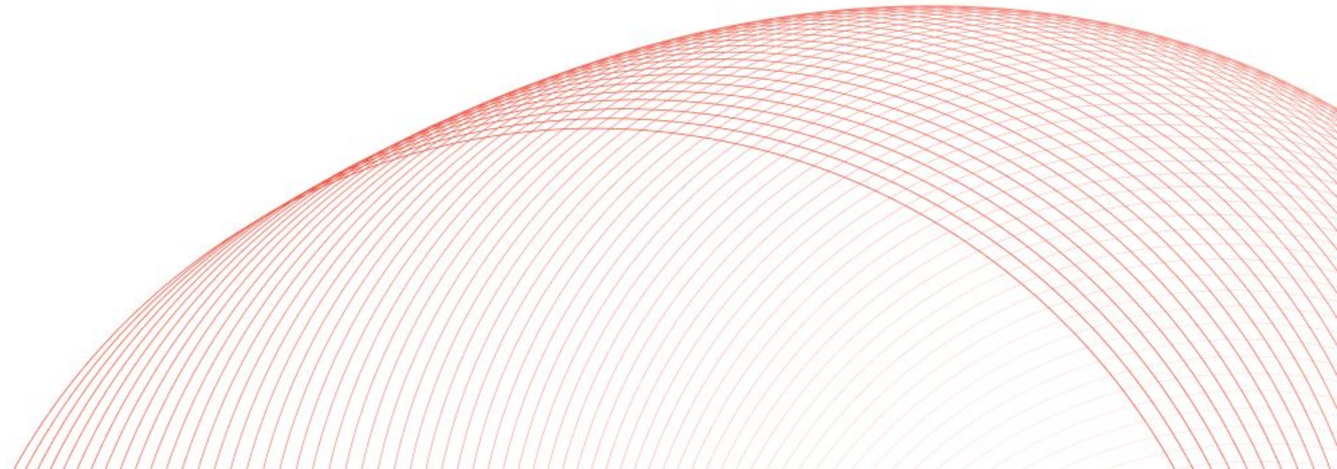
Требует максимальной скорости изменений для выживания на рынке

ИБ

Требует верификации безопасности каждого шага для выживания в правовом и техническом поле

Вызов

Как спроектировать архитектуру команд и распределить зоны ответственности так, чтобы продуктовые стримы, платформы и ИБ двигались к единой бизнес-цели, не блокируя друг друга?



Фундаментальный принцип:

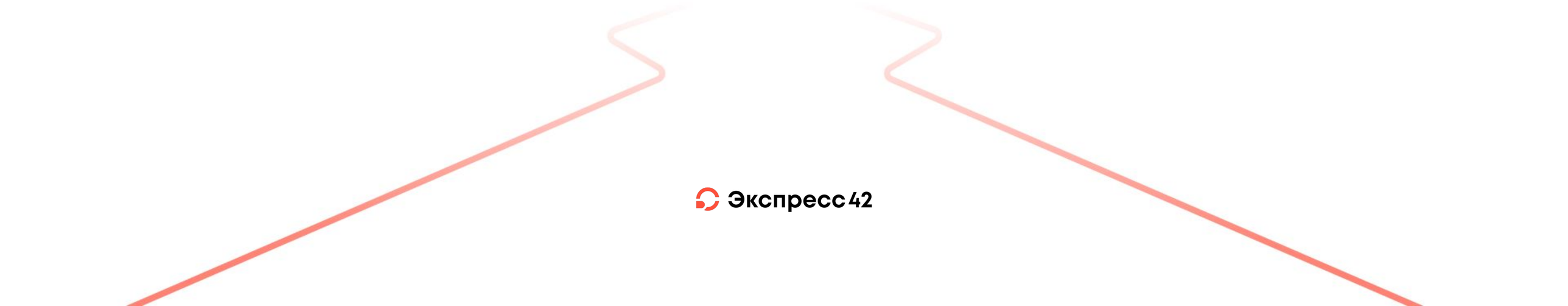
Коммуникации = Архитектура

> Организации, проектирующие системы,
> ограничены рамками копирования
> своих собственных структур коммуникаций

Закон Мелвина Конвея (1967 г.)

Заказчик

ACME Corp



Опять во всём виновата ИБ

Запрос бизнеса

ИБ тормозит релизы, требования непонятны,
компания теряет деньги

Инцидент-триггер

Падение HashiCorp Vault силами самой ИБ

Эффект домино

Деградация 12 часов → Нет оповещения об инциденте →
Попытка релиза «вслепую» → Старый сервис лёг,
новый не поднялся → Полный паралич доставки изменений

Диагноз

ИБ попала в системную ловушку. Команды пытались быть
одновременно платформой, консультантами и разработчиками

Итог

Размытие фокуса → Запредельная когнитивная нагрузка →
Технический инцидент

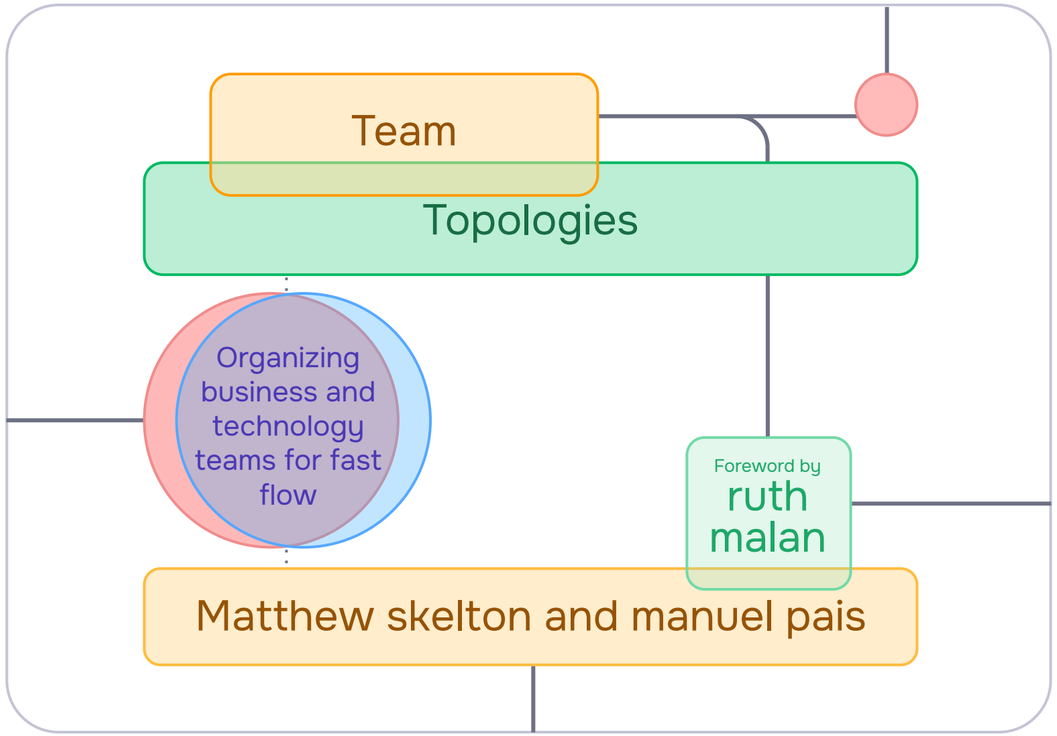
Отсутствие явных разграничений зон ответственности ведёт к инцидентам

Team Topologies

Это концептуальный фреймворк
и методологический подход
к организационному дизайну
компаний

Три главные цели фреймворка

- Снижение когнитивной нагрузки
- Ускорение потока изменений
- Ясные границы и инженерный подход к коммуникациям



Team Topologies

Основные сущности

Stream-Aligned Team

Четыре типа команд

01 Stream-aligned — отвечают за конкретный поток ценности (фичи, пользовательский опыт)

Team Topologies

Основные сущности

Stream-Aligned Team

Platform Team

Четыре типа команд

01 Stream-aligned — отвечают за конкретный поток ценности (фичи, пользовательский опыт)

02 Platform — предоставляют внутренние сервисы «как услугу» (X-as-a-Service), чтобы стрим-команды не изобретали велосипед

Team Topologies

Основные сущности

Stream-Aligned Team

Platform Team

Enabling Team

Четыре типа команд

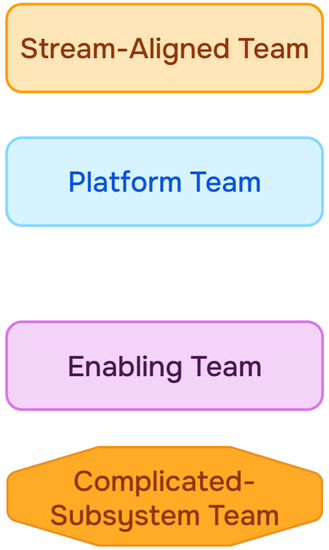
01 Stream-aligned — отвечают за конкретный поток ценности (фичи, пользовательский опыт)

02 Platform — предоставляют внутренние сервисы «как услугу» (X-as-a-Service), чтобы стрим-команды не изобретали велосипед

03 Enabling — помогают, обучают, закрывают пробелы в экспертизе, но не делают работу за других

Team Topologies

Основные сущности



Четыре типа команд

01 Stream-aligned — отвечают за конкретный поток ценности (фичи, пользовательский опыт)

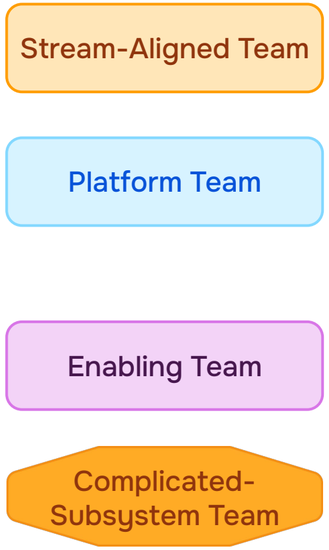
02 Platform — предоставляют внутренние сервисы «как услугу» (X-as-a-Service), чтобы стрим-команды не изобретали велосипед

03 Enabling — помогают, обучают, закрывают пробелы в экспертизе, но не делают работу за других

04 Complicated-subsystem — узкоспециализированная экспертиза, которая должна быть сконцентрирована в одной команде

Team Topologies

Основные сущности



Четыре типа команд

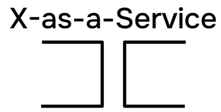
01 Stream-aligned — отвечают за конкретный поток ценности (фичи, пользовательский опыт)

02 Platform — предоставляют внутренние сервисы «как услугу» (X-as-a-Service), чтобы стрим-команды не изобретали велосипед

03 Enabling — помогают, обучают, закрывают пробелы в экспертизе, но не делают работу за других

04 Complicated-subsystem — узкоспециализированная экспертиза, которая должна быть сконцентрирована в одной команде

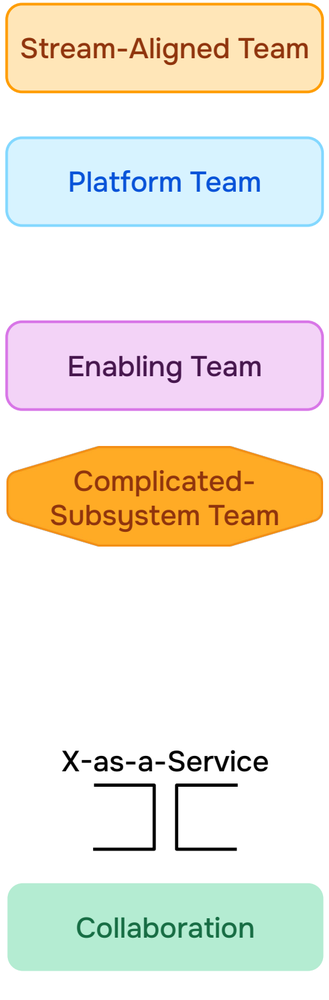
Три режима взаимодействия



01 X-as-a-Service — чёткий контракт, API, самообслуживание, минимум встреч

Team Topologies

Основные сущности



Четыре типа команд

01 Stream-aligned — отвечают за конкретный поток ценности (фичи, пользовательский опыт)

02 Platform — предоставляют внутренние сервисы «как услугу» (X-as-a-Service), чтобы стрим-команды не изобретали велосипед

03 Enabling — помогают, обучают, закрывают пробелы в экспертизе, но не делают работу за других

04 Complicated-subsystem — узкоспециализированная экспертиза, которая должна быть сконцентрирована в одной команде

Три режима взаимодействия

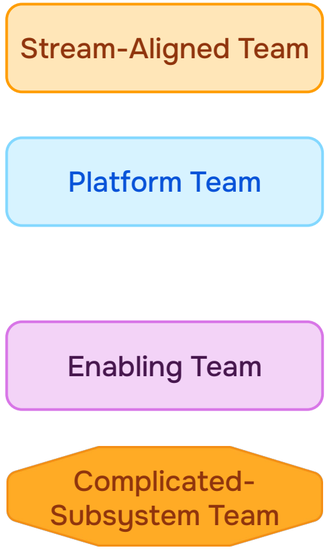
01 X-as-a-Service — чёткий контракт, API, самообслуживание, минимум встреч

02 Collaboration — временная совместная работа для решения сложной задачи, имеющей понятную ценность

Team Topologies

Основные сущности

Четыре типа команд



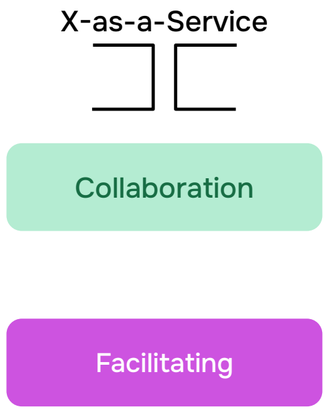
01 Stream-aligned — отвечают за конкретный поток ценности (фичи, пользовательский опыт)

02 Platform — предоставляют внутренние сервисы «как услугу» (X-as-a-Service), чтобы стрим-команды не изобретали велосипед

03 Enabling — помогают, обучают, закрывают пробелы в экспертизе, но не делают работу за других

04 Complicated-subsystem — узкоспециализированная экспертиза, которая должна быть сконцентрирована в одной команде

Три режима взаимодействия



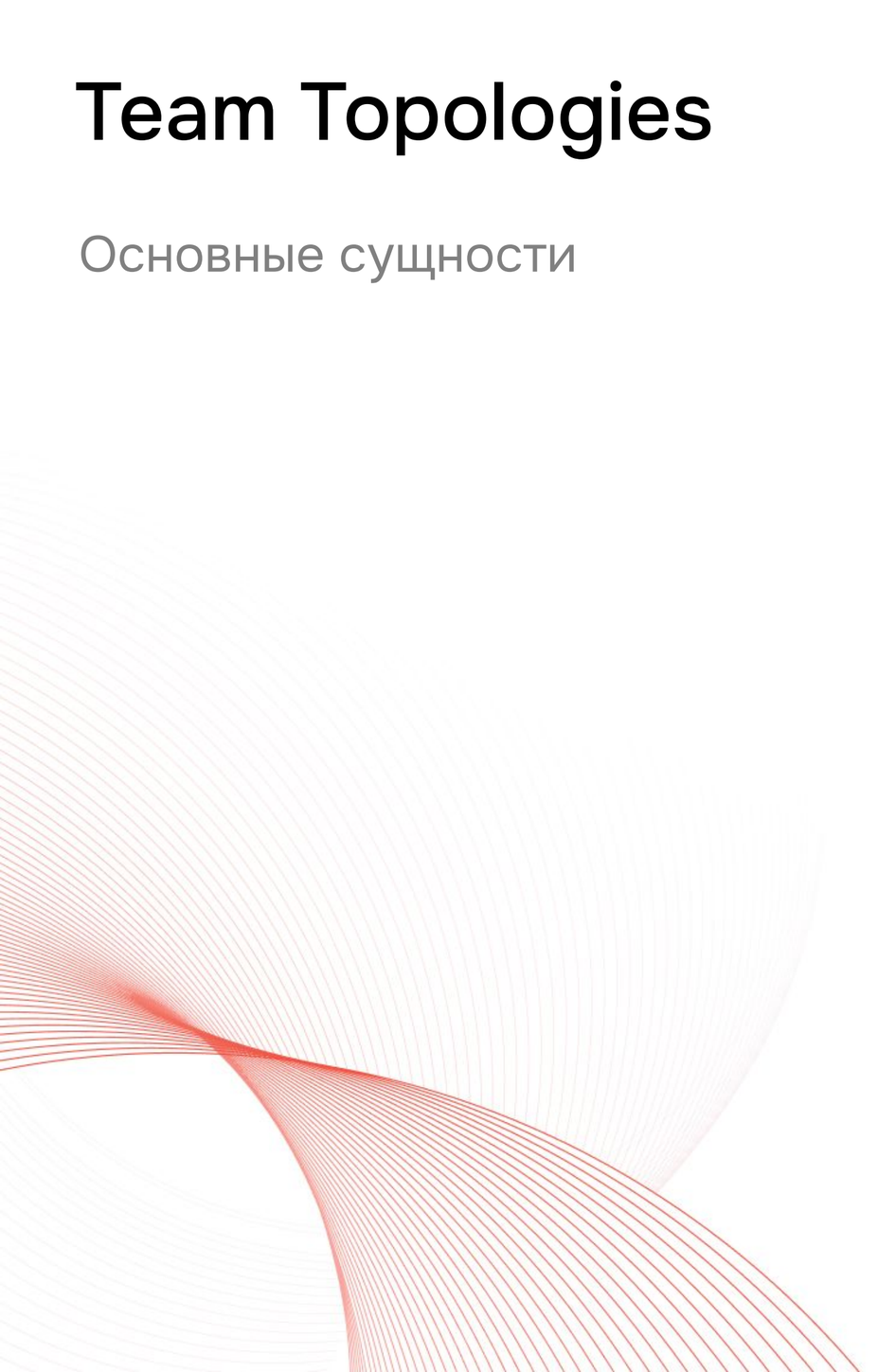
01 X-as-a-Service — чёткий контракт, API, самообслуживание, минимум встреч

02 Collaboration — временная совместная работа для решения сложной задачи, имеющей понятную ценность

03 Facilitating — наставничество, снятие блоков, экспертная поддержка

Team Topologies

Основные сущности



Три типа когнитивной нагрузки

Внутренняя Intrinsic

Связана с базовыми навыками разработчика

Внешняя/побочная Extraneous

Связана с окружением, процессами и рутиной

Полезная Germane

Связана с бизнес-логикой и решением реальных задач клиента

Какие типы команд применимы в ИБ

И как их правильно классифицировать?

Platform Team команда платформенных сервисов

Когда ИБ-команда становится платформой

Если она берёт на себя предоставление готовых сервисов для других команд компании, например:

- предоставление инфраструктурных сервисов (SIEM, ASOC, PKI)
- поддержка сканеров безопасности (SAST, SCA, DAST)
- создание инструментов для других команд (IAM, сертификаты, секреты)

В каком режиме взаимодействия команда должна работать

X-as-a-Service
(платформа как сервис)

Ключевые требования: к команде

- Чёткие SLA и публичные метрики доступности
- Self-service-интерфейс (API, портал, CLI)
- Публичная документация и changelog
- Минимум ручных согласований
- Потребители используют сервис автономно

Какие типы команд применимы в ИБ

И как их правильно классифицировать?

Enabling Team команда экспертной поддержки/обучения

Если главная цель команды — не делать работу руками, а повышать безопасность чужими руками

- Обучение разработчиков безопасной разработке (DevSecOps)
- Внедрение security-стандартов и отраслевых best practices
- Консультирование продуктовых команд на ранних этапах
- Развитие сообщества Security Champions внутри продуктовых команд
- Проведение аудитов и оценка рисков

В каком режиме взаимодействия команда должна работать

Facilitating (фасилитация/наставничество)

Ключевые требования

- Наставничество и менторство
- Обучение и передача знаний
- Помощь в решении конкретных проблем
- Временное вовлечение, но никогда не делает работу за другие команды

Какие типы команд применимы в ИБ

И как их правильно классифицировать?

Complicated Subsystem Team — команда сложных подсистем

Когда команда уходит в узкую специализацию. Сюда относятся редкие эксперты, чьи знания невозможно «размазать» по всей компании

- Занимается узкоспециализированной экспертизой: криптографией, Reverse Engineering, анализом вредоносного ПО (Malware Analysis), тестами на проникновение (Pentest)
- Разрабатывает сложные компоненты, требующие глубоких научных или технических знаний

В каком режиме взаимодействия команда должна работать

Collaboration (ограниченно по времени, например для интеграции компонента)

Что это значит

- Другие команды обращаются, только когда это критично
- Экспертиза концентрируется в одном месте
- Команда работает изолированно над своими задачами, а другие подразделения привлекают её только тогда, когда это критически необходимо для интеграции сложного узла

Какие типы команд применимы в ИБ

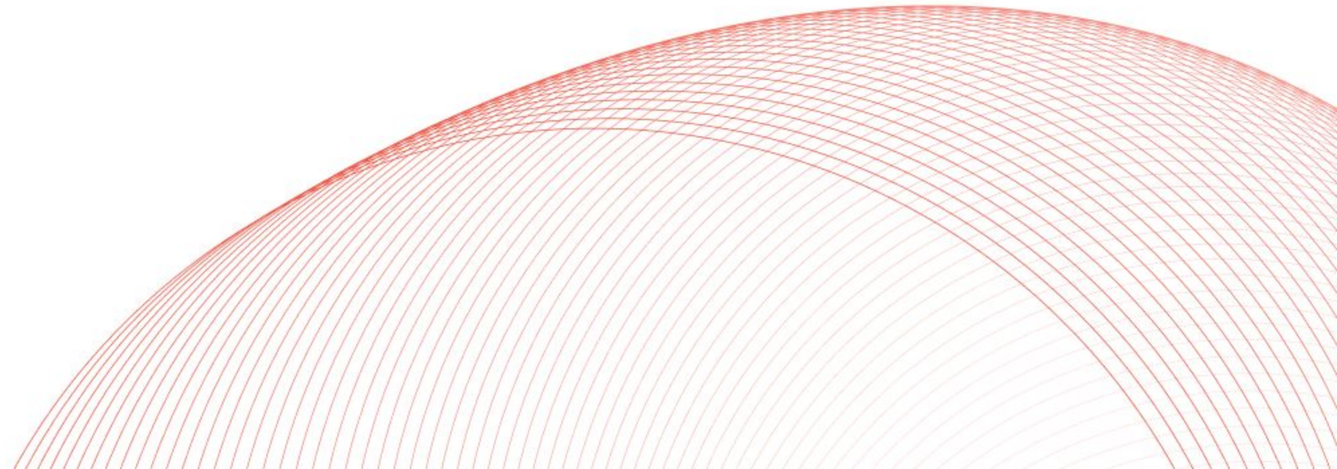
И как их правильно классифицировать?

Причины, почему ИБ — это **не** Stream-aligned

01 У ИБ нет своего независимого потока бизнес-ценности (Value Stream)

02 Разные скорость и метрики успеха

03 ИБ по своей природе — это сервисная или контролирующая функция



Team Topologies

Это диагностический инструмент, который позволяет увидеть системные проблемы в организации до того, как они превратятся в инциденты

Как это работает?

01 Выявляете несоответствие между типом команды и её реальной деятельностью

02 Видите скрытые зависимости

03 Обнаруживаете размытие фокуса

04 Видите размытие границ ответственности и «узкие горлышки» межкомандного взаимодействия

Текущее состояние и ключевые проблемы

- **Фикция платформы**

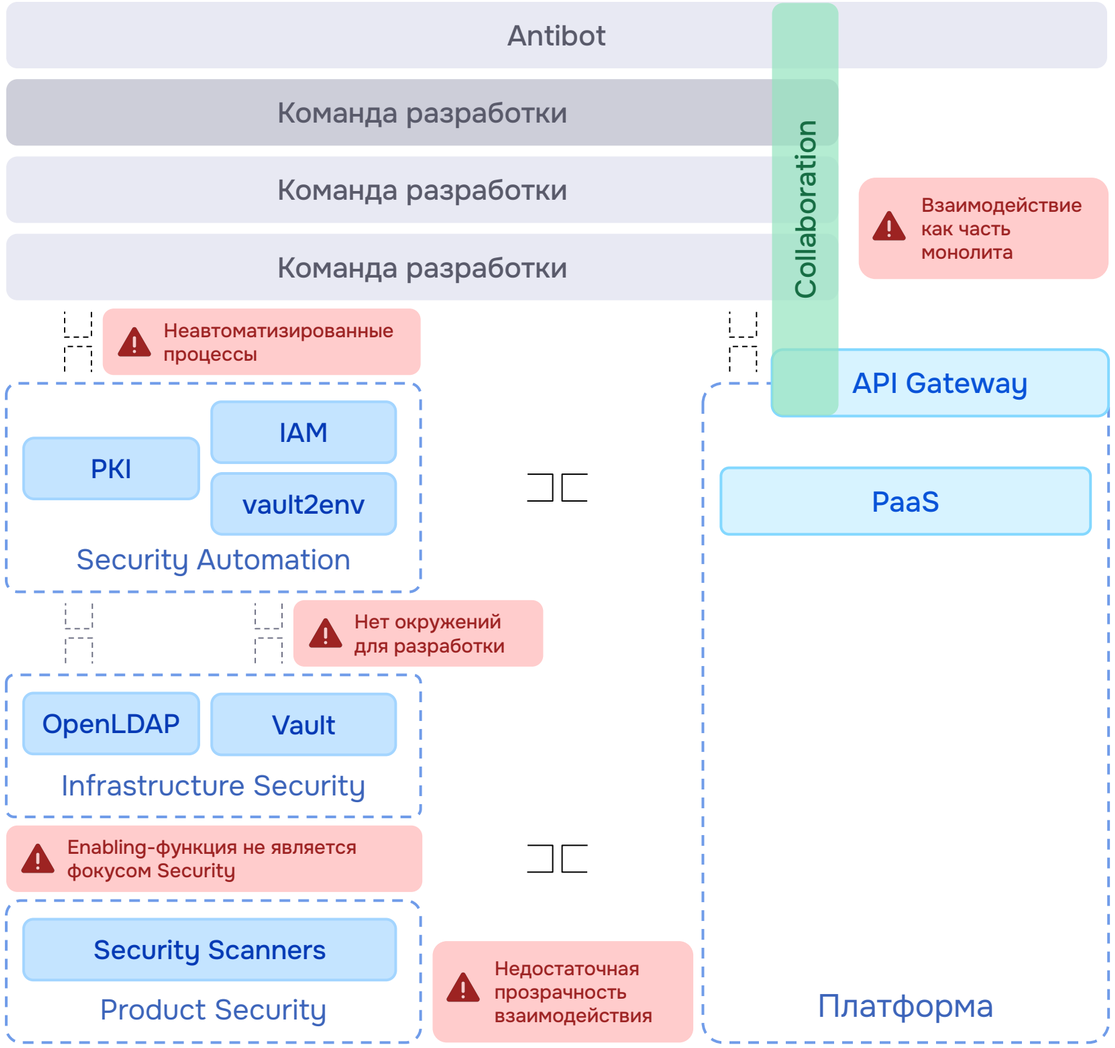
  Сервисы предоставляются через ручные операции вместо модели X-as-a-Service

- **Блокирующие процессы**

  Взаимодействие требует постоянной постановки задач и ручного труда

- **Размытие фокуса**

  Продуктовые и инфраструктурные команды перегружены развитием платформ в ущерб обучению (Enabling)

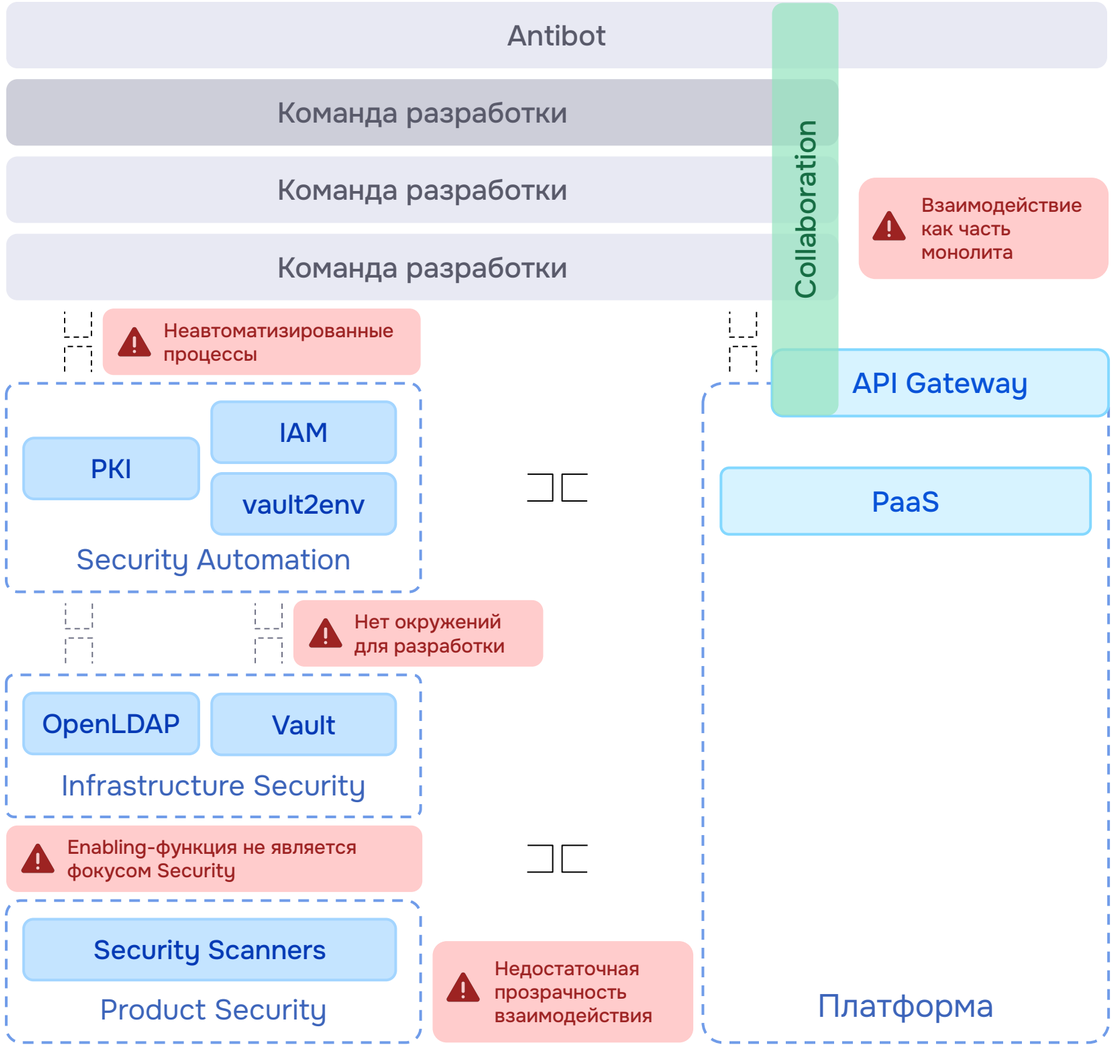


Главный вывод:

команды не соответствуют принципам Team Topologies

Проблемы Antibot

- **Архитектурный тупик**
  Сервис привязан к релизному циклу API Gateway
- **Зависимость от монолита**
  Невозможность независимого деплоя
- **Размытие ответственности**
  Команда сама пишет правила распознавания ботов для продуктовых команд
- **Избыточный Collaboration**
  Чрезмерное взаимодействие с Stream-aligned- и платформенной командами

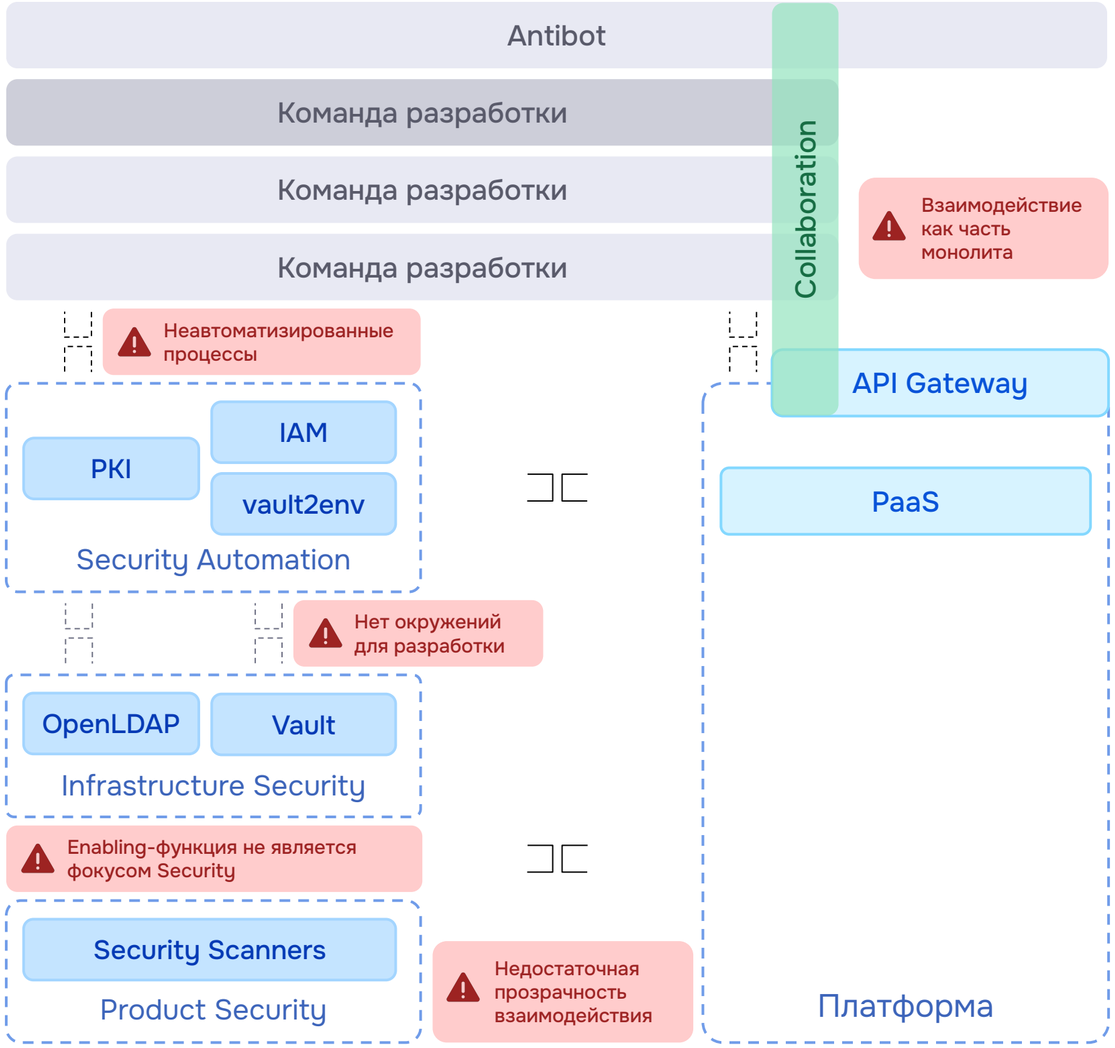


Главный вывод:

избыточное взаимодействие
и жёсткая привязка к монолиту

Проблемы Security Automation

- **Security Automation (PKI)**
  Ручной выпуск и перевыпуск сертификатов
- **Security Automation (IAM)**
  Проверка работоспособности через ручные тикеты в Service Desk
- **Infrastructure Security**
  Отсутствие тестовых окружений для Security Automation и PaaS
- **Следствие**
  Изменения тестируются прямо в продакшене, что повышает риск аварий

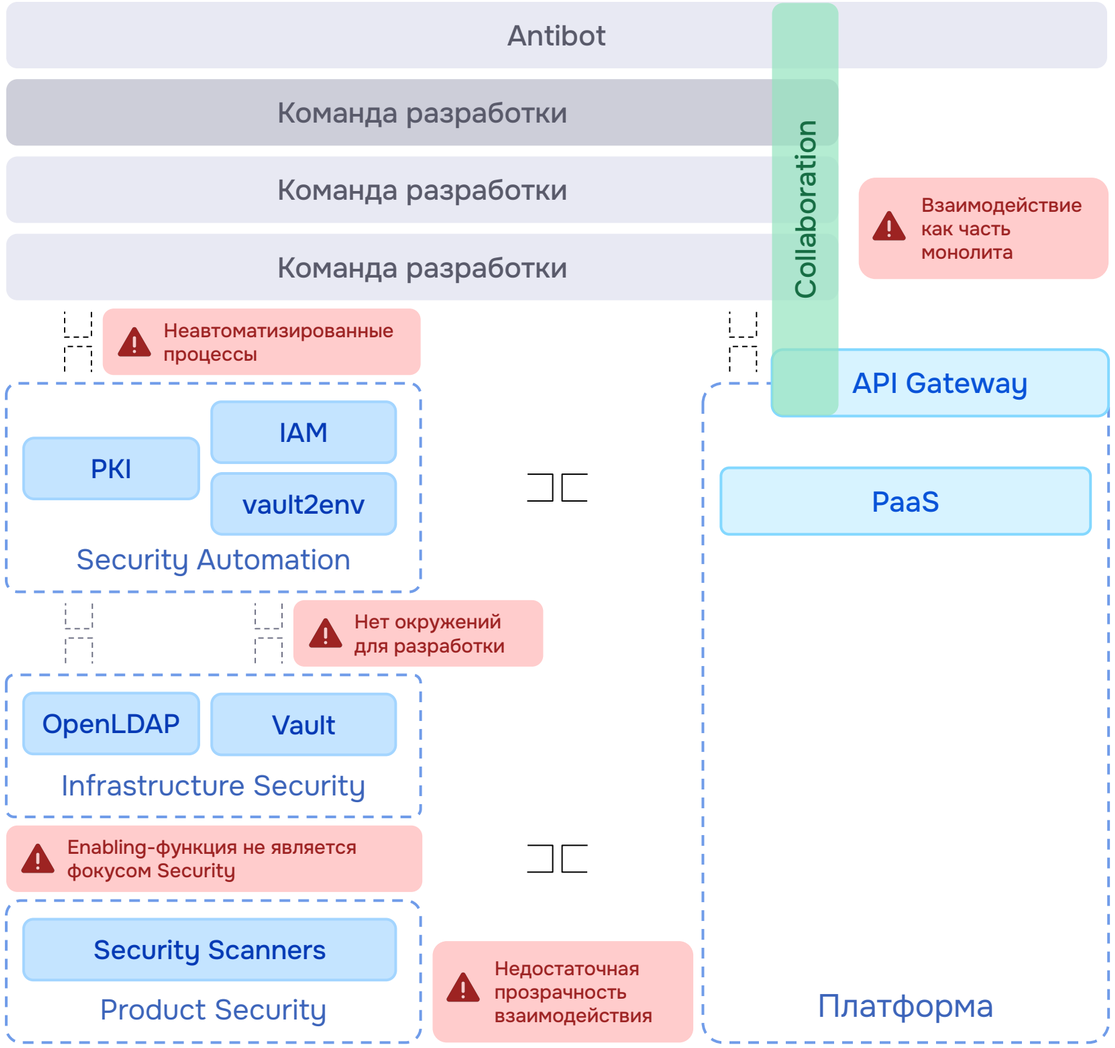


Главный вывод:

ручные процессы в ИТ-безопасности тормозят разработку и увеличивают риски

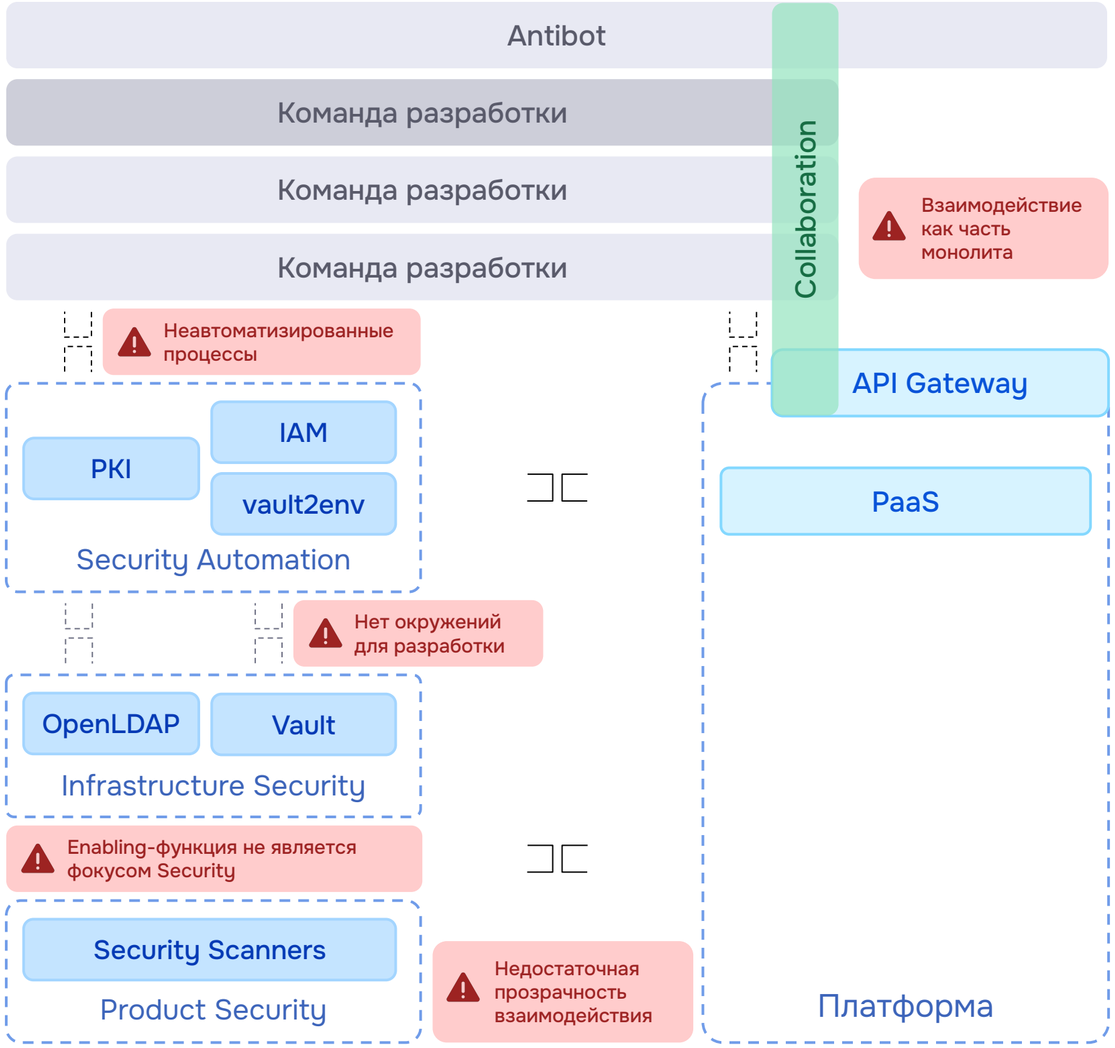
Проблемы Infrastructure Security

- Не имеют и не предоставляют тестовых окружений
- Отсутствует понимание границ ответственности на уровне инфраструктуры
- Отсутствует понимание верхней границы нагрузки, не приводящей к деградации сервиса



Проблемы Product Security

- **«Слепая зона» сервисов**
  Нет API и статусных страниц для диагностики
- **Критический сбой**
  Вкладка «Безопасность» в PaaS полностью отключена
- **Перегрузка функций**
  Попытка совместить развитие платформы с консультированием разработчиков



Инциденты | Кейс 1

«Кто виноват в ботах?»

Суть проблемы

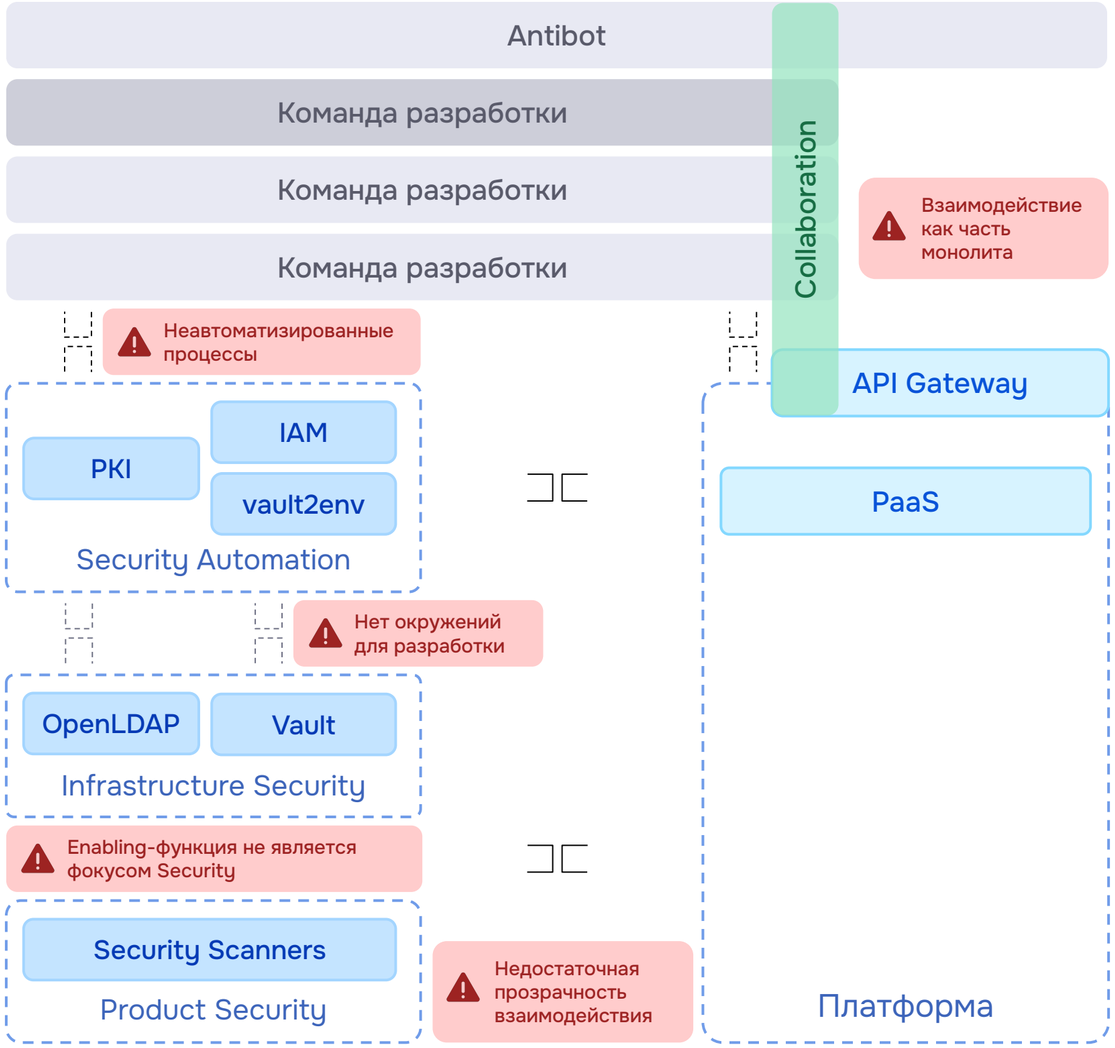
Команда Antibot сама писала бизнес-правила фильтрации для разных направлений («Авто», «Электроника»)

Следствие для бизнеса

Ошибочные правила блокировали реальных клиентов и снижали продажи

Конфликт интересов

Бизнес винил ИБ в падении выручки.
ИБ физически не могла знать специфику каждого продукта



Инциденты | Кейс 2

«12 часов без Vault»

Суть проблемы

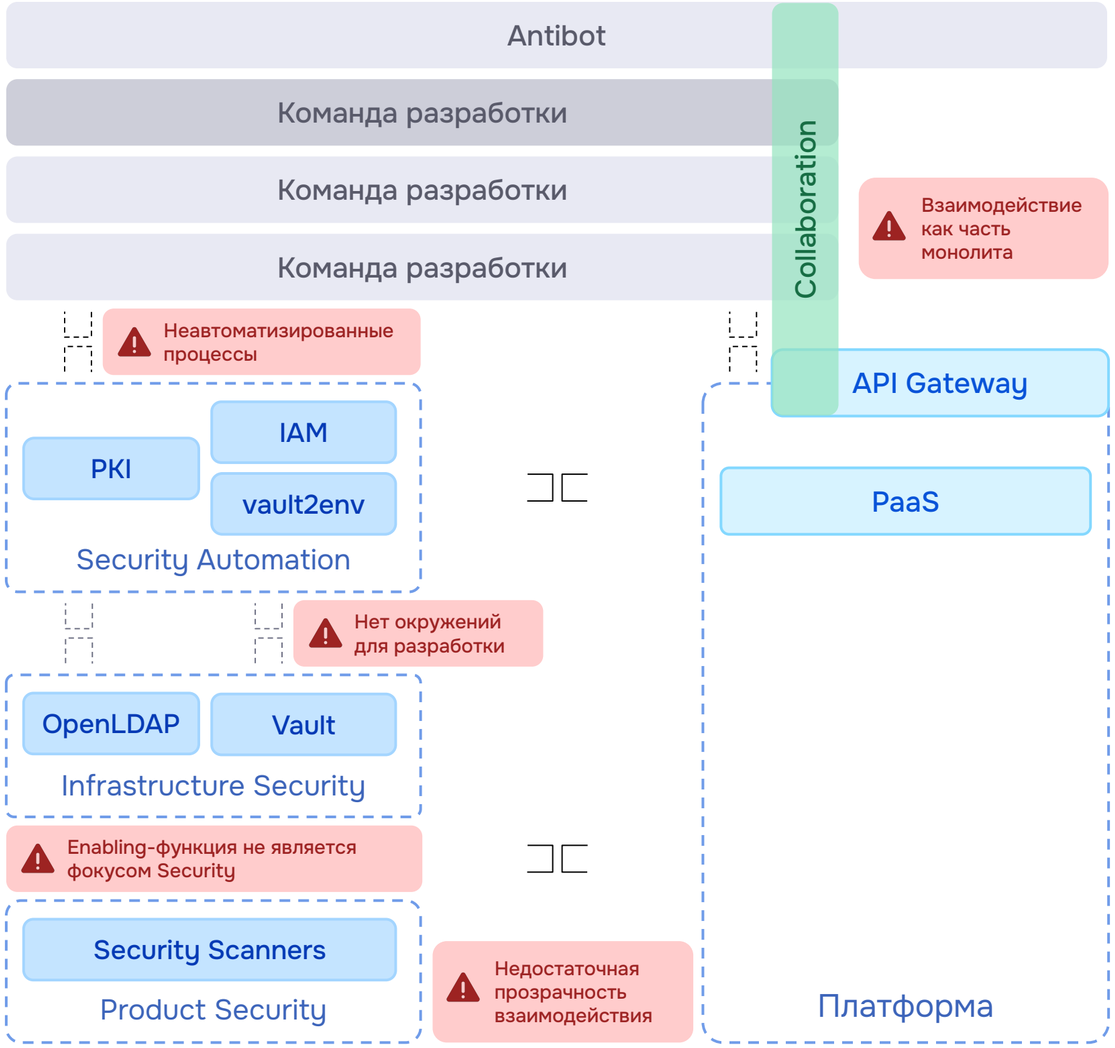
Изменения тестировались «на глаз» в продакшене, Vault жил на одной ВМ с чужими сервисами без лимитов и бэкапов

Следствие для бизнеса

12 часов полного простоя доставки (паралич деплоев во всей компании из-за недоступности секретов), каскадное падение зависимых от Vault сервисов привело к значительным финансовым потерям

Конфликт интересов

Команда разработки писала плагины, считая Vault чужой инфраструктурой, а Infrastructure Security не знала о нагрузках от новых плагинов



Что мы обнаружили по результатам диагностики

- Ручные процессы вместо автоматизации
- Размытие границ ответственности внутри ИБ-команд
- Архитектурный тупик и «вечный Collaboration»
- Фикция платформенного взаимодействия
- Разрозненные практики поставки
- Утонувшая экспертиза

Как это починить?

Обратный манёвр Конвея

Суть концепции:

закон Конвея работает
в обе стороны. Чтобы получить
правильную архитектуру систем,
нужно осознанно спроектировать
коммуникации в компании

- **Проектирование**

  Сначала мы проектируем целевые сервисы,
  API и границы отказоустойчивости,
  затем под них формируем команды

- **Ликвидация оргдолга**

  Избавляемся от «вечного Collaboration»
  и ручных согласований на уровне людей,
  а не только в коде

- **ИБ как фундамент**

  Безопасность перестаёт быть внешней
  «надстройкой-контролёром» и становится
  частью надёжной инфраструктуры

Как это работает на практике?

Обратный манёвр Конвея

? **Нужны независимые релизы?**

✓ Создаём автономные продуктовые команды (Stream-aligned team)

? **Нужно общее хранилище секретов?**

✓ Создаём платформенную команду (Platform team) с чётким контрактом X-as-a-Service

? **Нужно массовое обучение разработчиков?**

✓ Создаём обучающую команду (Enabling team)

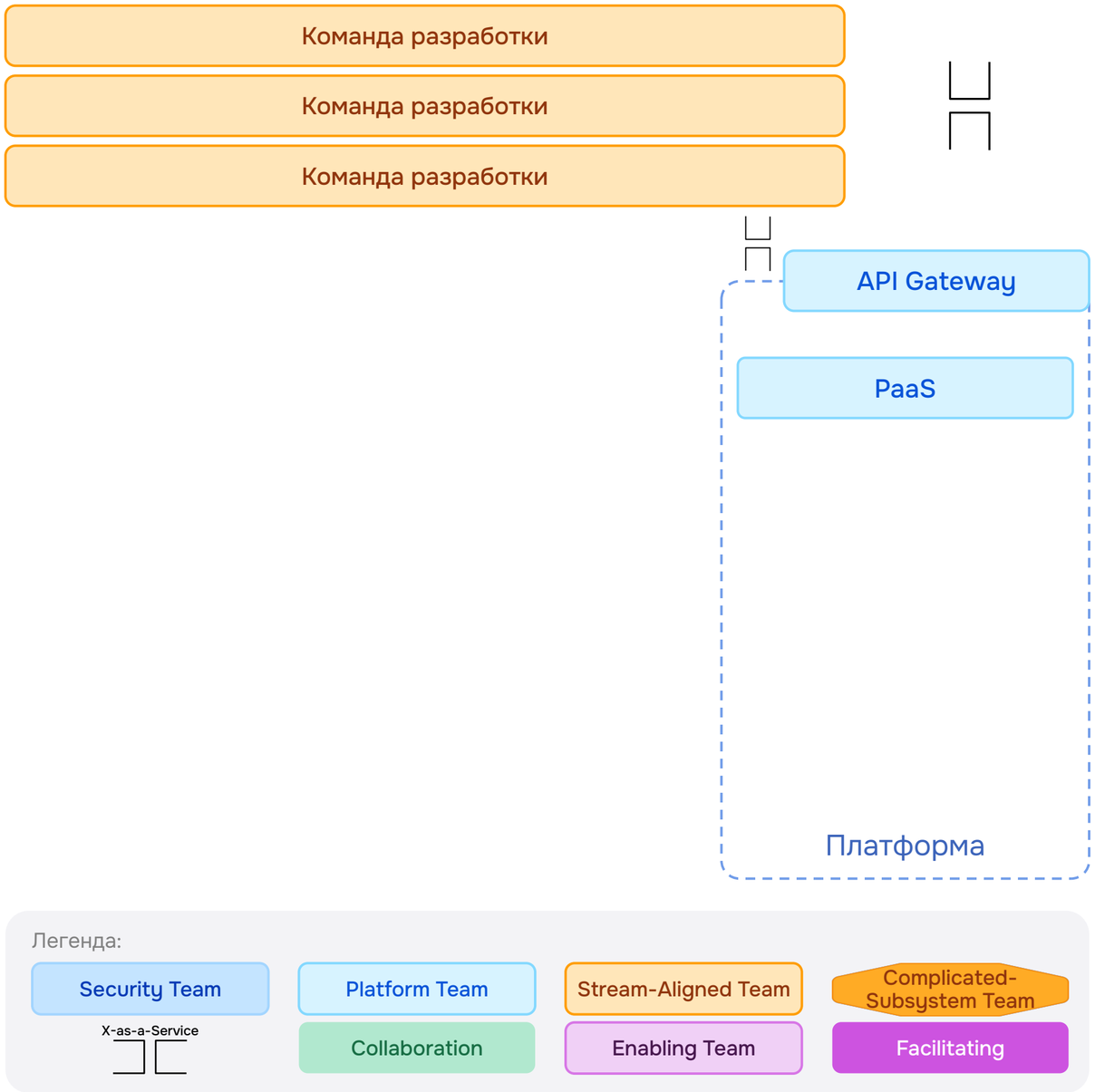
? **Требуются узкоспециализированные знания и компетенции?**

✓ Создаём команду сложной подсистемы (Complicated-subsystem team)

Механика перехода к целевому состоянию

Platform (Команда api-gateway)

- Публикует чёткий контракт: (версия рантайма, API, SLA)
- Предоставляет self-service-пайплайн для валидации и деплоя
- Предоставляет тестовые окружения

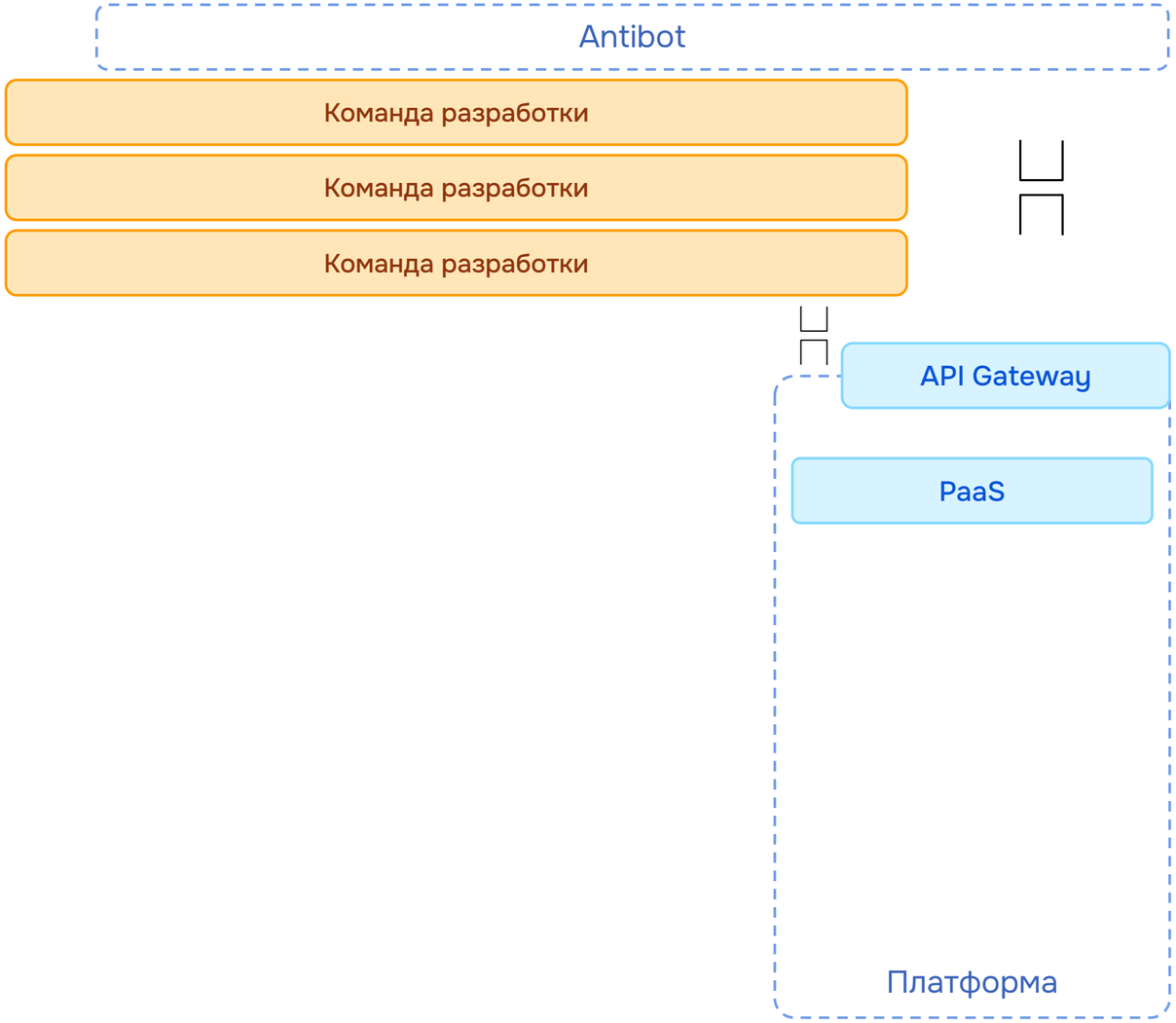


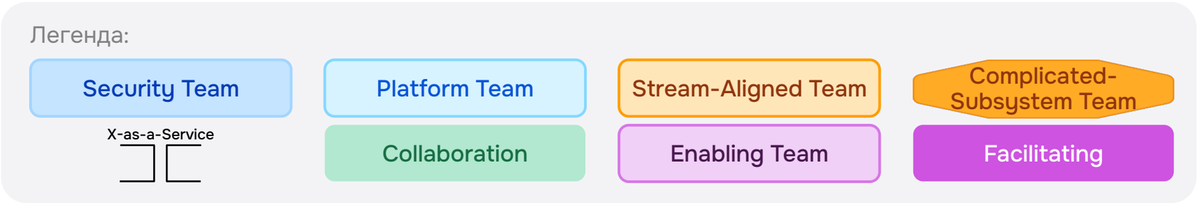
Механика перехода к целевому состоянию

Antibot становится платформенной командой, так как пишет сервис для расширения возможностей существующей платформы.

Разрабатывает движок строго под контракт другой платформенной команды

- Проходит автоматические контрактные тесты
- Деплоит автономно — без согласований с командой api-gateway
- Снимает с себя ответственность за адаптацию решения для конкретных бизнес-задач продуктовых команд

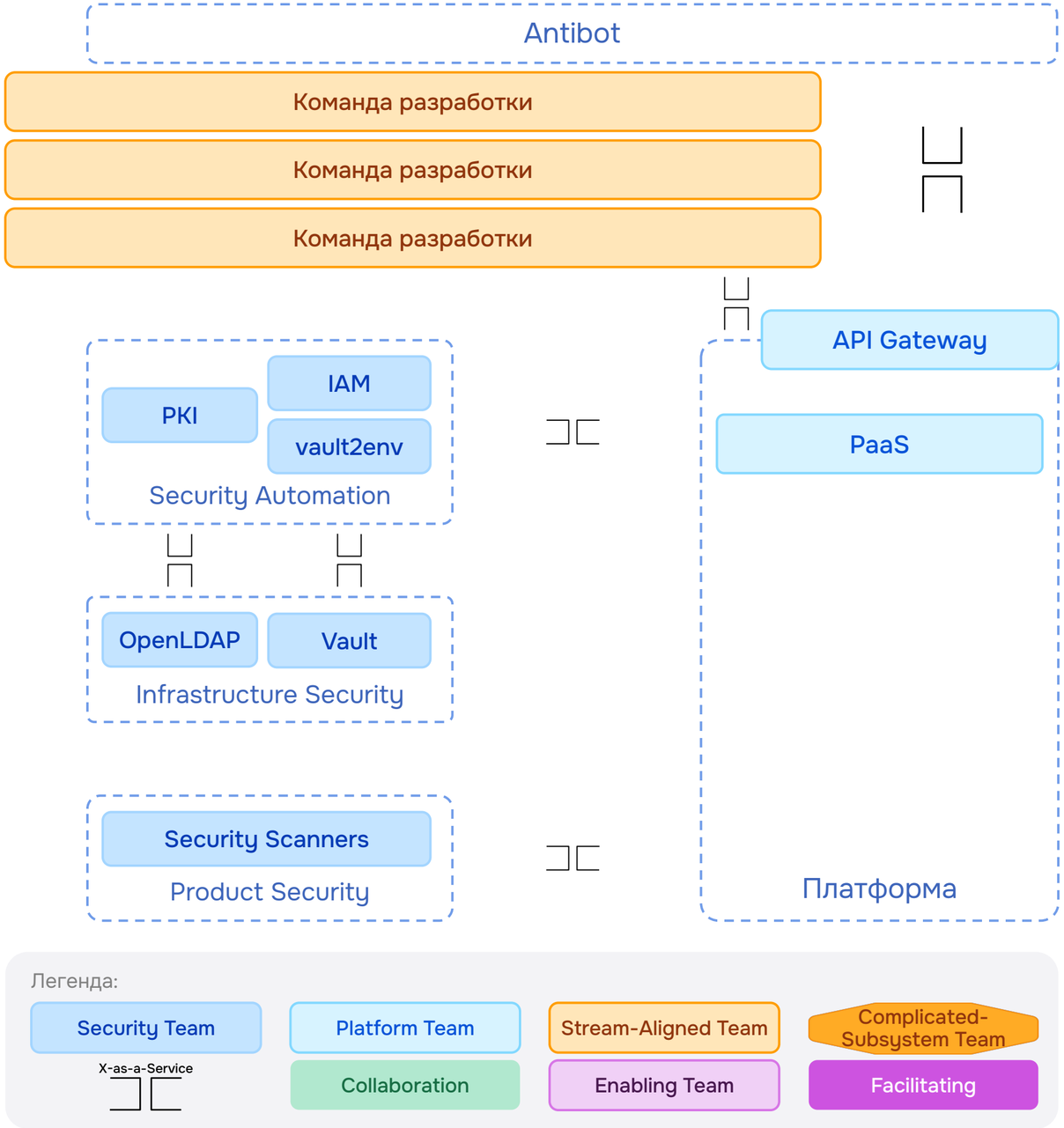




Механика перехода к целевому состоянию

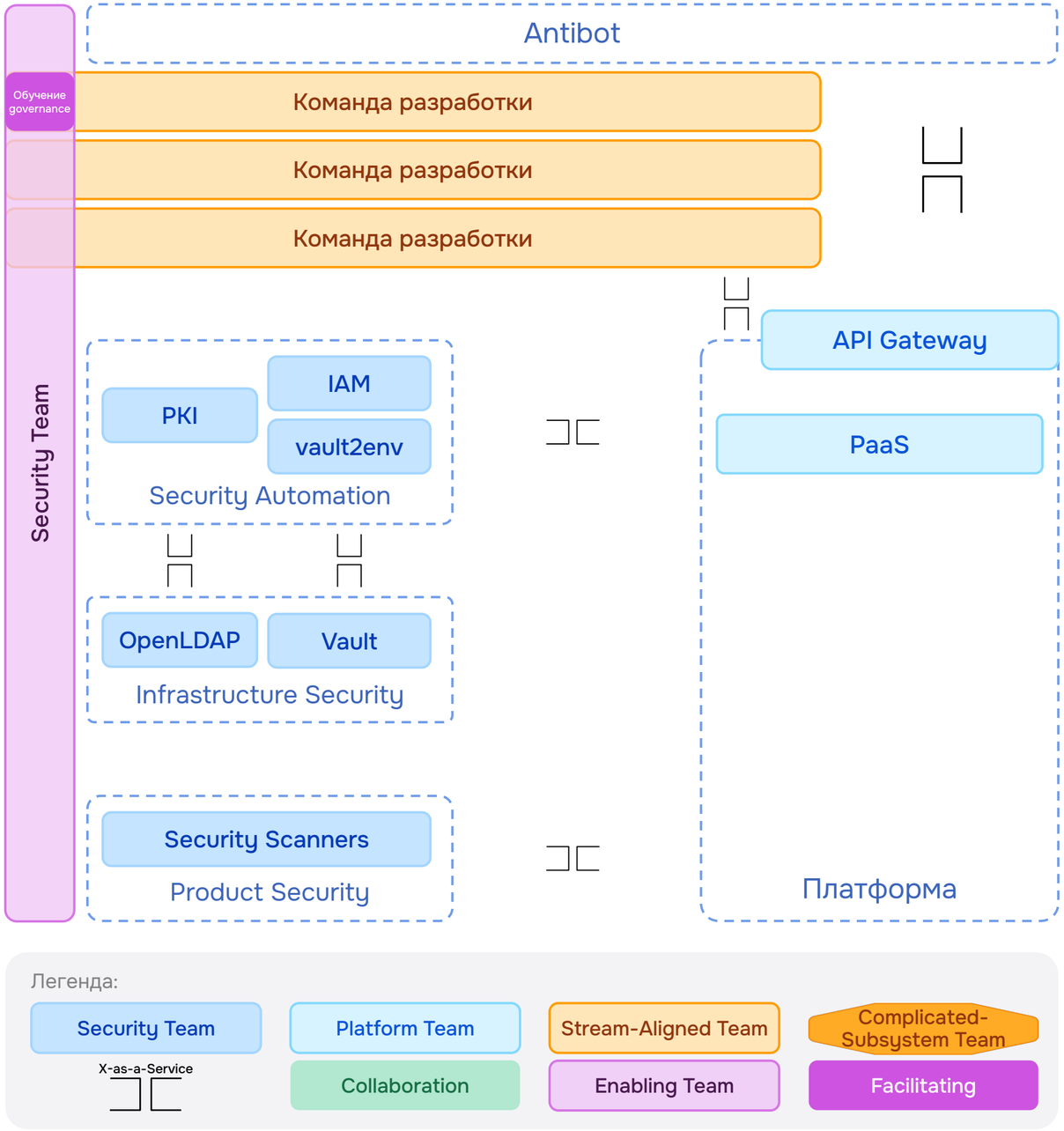
Все платформенные сервисы ИБ в режим XaaS

- Контракты опубликованы
- Чёткие границы ответственности и SLA
- Предоставляются тестовые окружения и проводятся нагрузочные тестирования



Фокус
на Enabling-функции

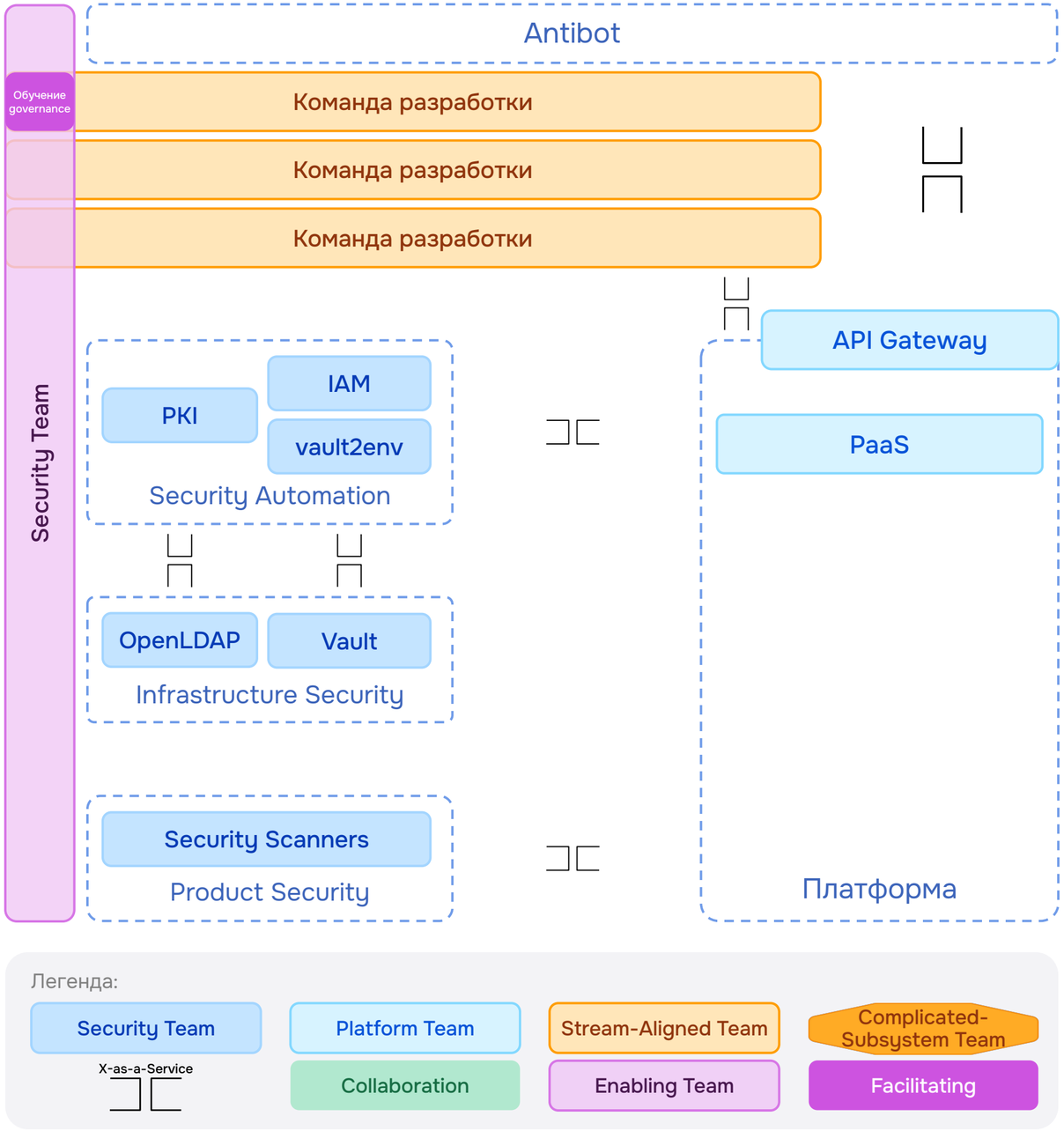
- Разрабатывает и транслирует стандарты: чёткие, автоматизированные политики (что и как должно быть защищено)
- Развивает институт Security Champions
- Проводит воркшопы и архитектурные ревью на этапе дизайна: фасилитирует, а не делает за команды; помогает встроить безопасность в процесс
- Создаёт самообслуживаемые ресурсы: публичная документация, гайды по secure coding, шаблоны конфигураций, — чтобы команды могли учиться и применять знания автономно
- Работает в режиме Facilitating: наставничество, консультации, transfer знаний. Collaboration — только временно, с жёстким дедлайном выхода
- Метрика успеха: автономность продуктовых команд в вопросах безопасности



Механика перехода к целевому состоянию

Продуктовые команды

- Подключают Antibot SDK
- Описывают правила детекта самостоятельно и несут за это ответственность
- Применяют через self-service → бизнес сам управляет паттернами под свою конверсию



Результаты

- Уменьшение когнитивной нагрузки
- Устранение человеческого фактора через контракты (SLA/API)
- Эффект рычага (масштабируемость через Enabling)
- Правильная мотивация бизнеса (возврат ответственности)

Итог

Перевод в формат «Продукт как Сервис» и жёсткие рамки для Collaboration снижают методологические риски. ИБ перестаёт быть «узким горлышком» и становится фундаментом, на котором строятся фичи

Выводы

01

Структура определяет архитектуру и культуру

02

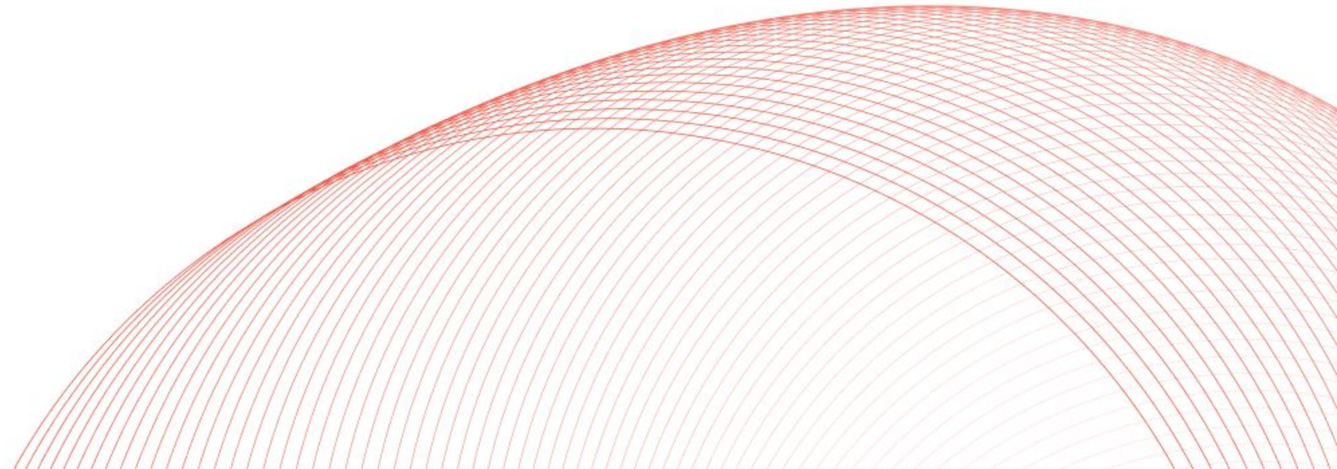
ИБ как сервис, а не как барьер

03

Уход от ручных операций и согласований

04

Строгое разграничение ответственности



Послесловие

Три принципа, которые меняют всё:

- Конкретные типы команд и чёткие границы ответственности
- Осознанно спроектированные коммуникации
- Эволюция: оргструктура — это живой организм. Команды должны менять форму и тип взаимодействия по мере изменения бизнес-контекста

Экспресс 42

Приходите за экспертизой!

Telegram-канал «Экспресс 42»

Официальный сайт Team Topologies

Репозиторий на GitHub с шаблонами фигур — векторные исходники и требования к геометрии схем

input@express42.com

express42.com