2 ИЮНЯ 2026, МОСКВА, ЛОФТ ГОЭЛРО

БЕКОН LUNTRY '26

ЕДИНСТВЕННАЯ КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Как не убить кластер и безопасность исключениями

Валерий Кунавин | Ozon Банк

БЕКОН'26

Как не убить кластер и безопасность исключениями

Валерий Кунавин

Старший инженер

Сегодня с вами

Валерий
Старший инженер в ОЗОН Банке
~4 года опыта работы с K8s
Спикер ZeroNights, БеКон 2024, Код ИБ



ЦА доклада

Те, кто только начинает защищать K8s.

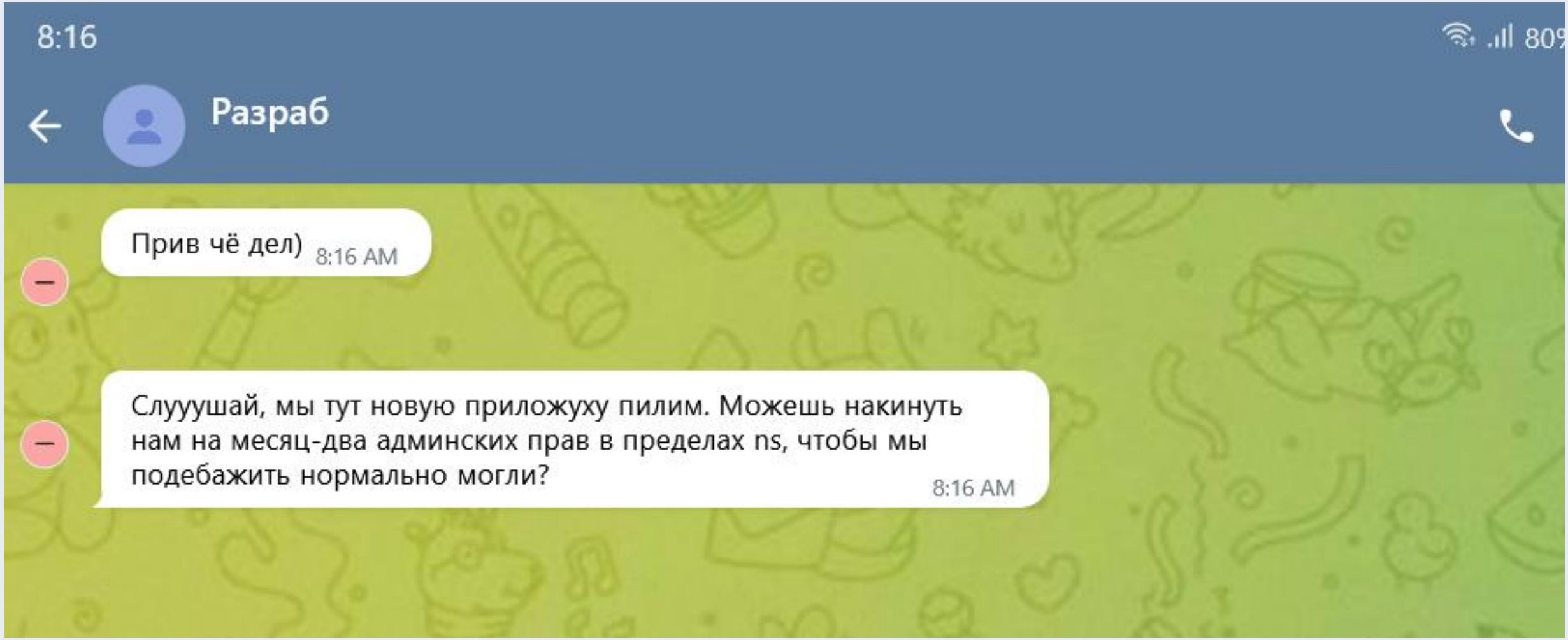
Те, кто уже преисполнился защитными мерами.

Те, кому не хочется согласовывать исключения.

Те, кому хочется согласовывать исключения, но страшновато.

Те, кто хочет посмотреть на часто возникающие ситуации, связанные с исключениями, и сделать «дома» по-своему.




8:16
Разраб
Прив чё дел) 8:16 AM
Слууушай, мы тут новую приложуху пилим. Можешь накинуть нам на месяц-два админских прав в пределах ns, чтобы мы подебажить нормально могли? 8:16 AM



В меме использован кадр из фильма «Вилли Вонка и шоколадная фабрика»

Общий подход к исключениям

1. IaC. Даже если сейчас вручную, потом – IaC.
2. IaC — круто, но реальность все равно в кластере.
3. Любые исключения проходят согласование у безопасников.
4. Вне продуктива тоже.
5. Чем сильнее **формализован и документирован** процесс выдачи исключений, тем меньше крови потеряют разработчики.
6. Чем этот процесс более зрелый, тем меньше крови потеряют админы и безопасники.



БЕКОН'26

RBAC



1

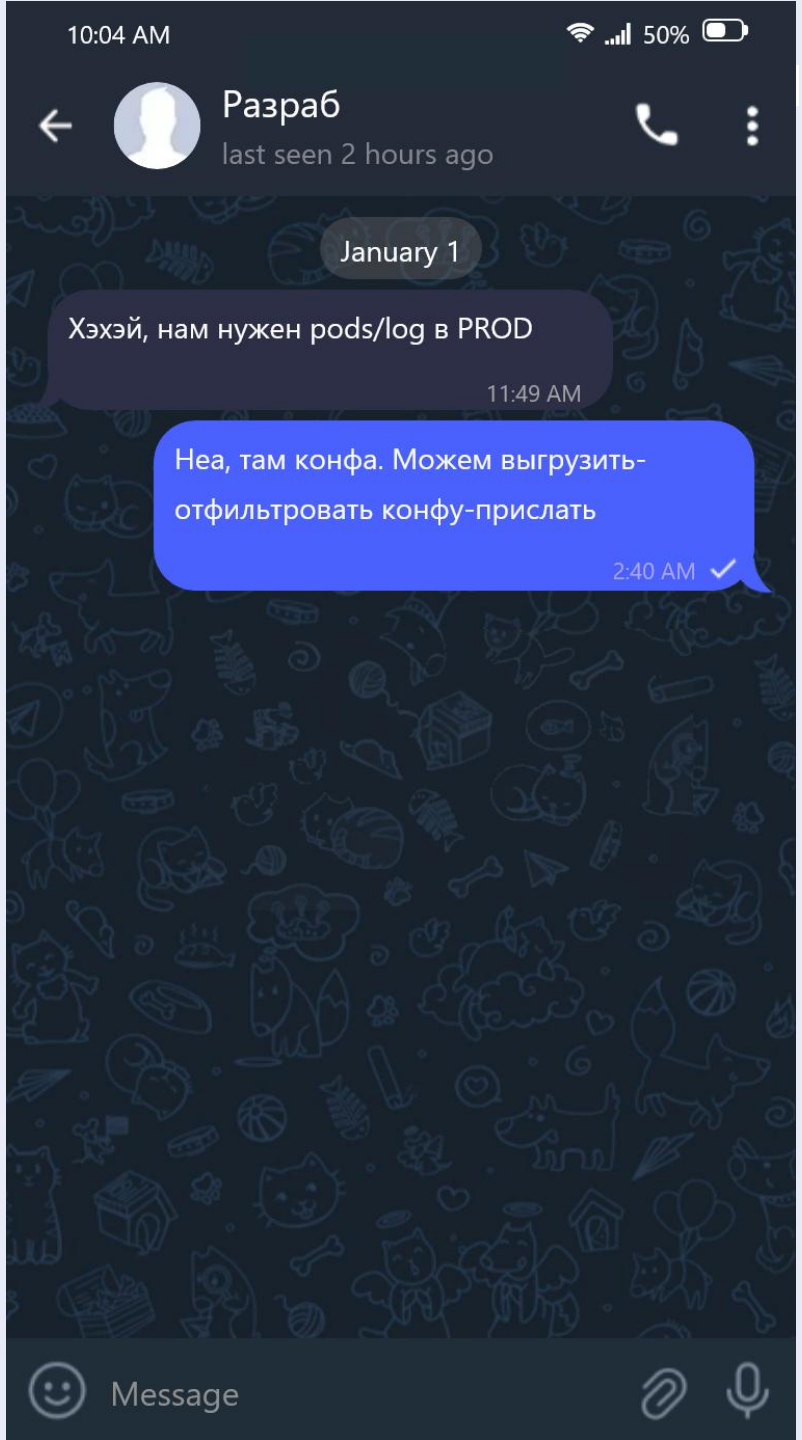
Когда могут понадобиться расширенные права?

- Админские навороты (сторонние или свои).
- Сервис-аккаунты CI/CD.
- Некоторые команды разработки (привет, мы ML, дайте доступ к логам).
- Ранние этапы развития безопасности в кластере.



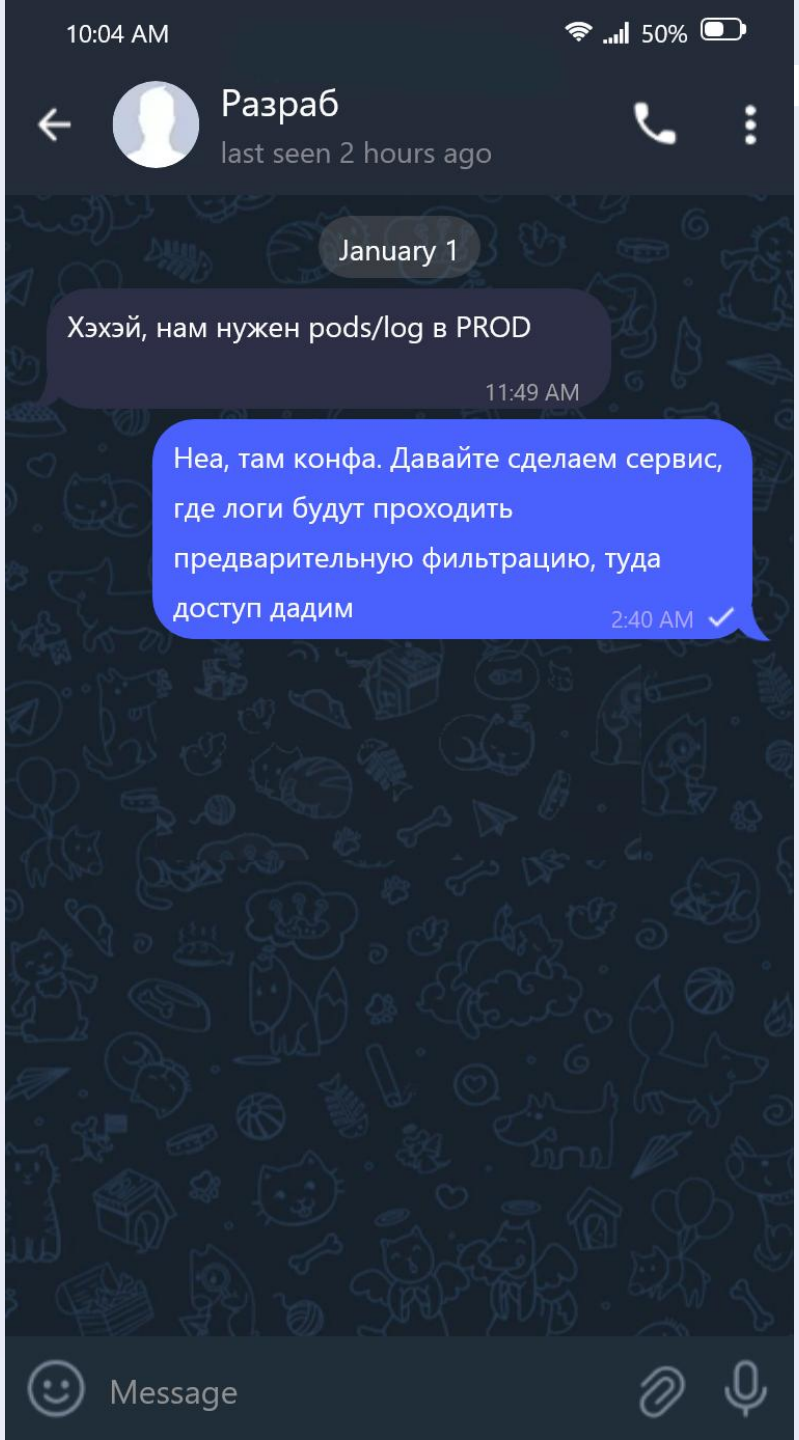
На что обратить внимание?

1. Список людей, которые имеют доступ к конфиденциальной информации, лучше держать небольшим.



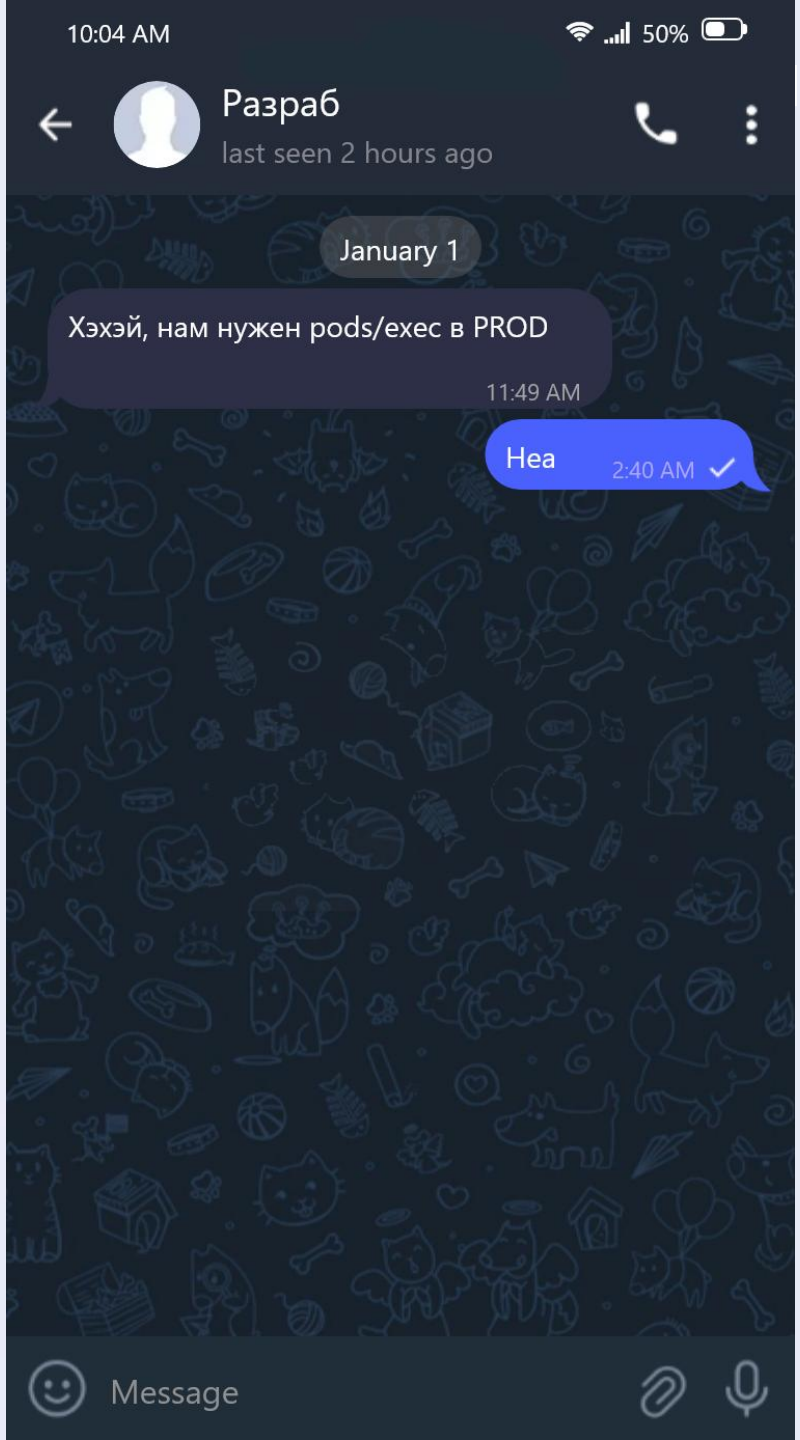
На что обратить внимание?

1. Список людей, которые имеют доступ к конфиденциальной информации, лучше держать небольшим.
2. Какие-то доступы можно дать в другом месте **(своя разработка или готовые системы).**



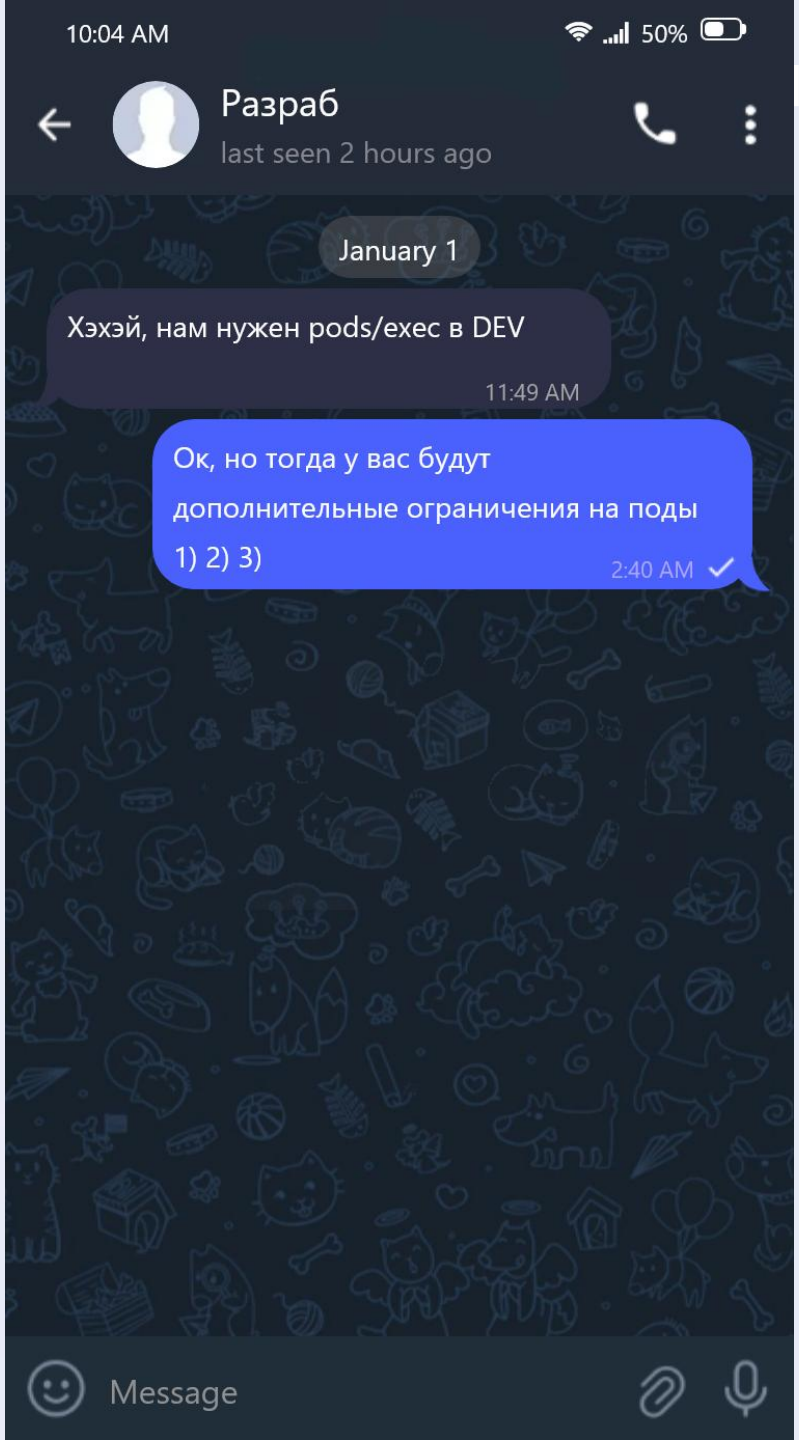
На что обратить внимание?

1. Список людей, которые имеют доступ к конфиденциальной информации, лучше держать небольшим.
2. Какие-то доступы можно дать в другом месте **(своя разработка или готовые системы).**
3. Для некоторых привилегий не нужны компромиссы, и так всё понятно.



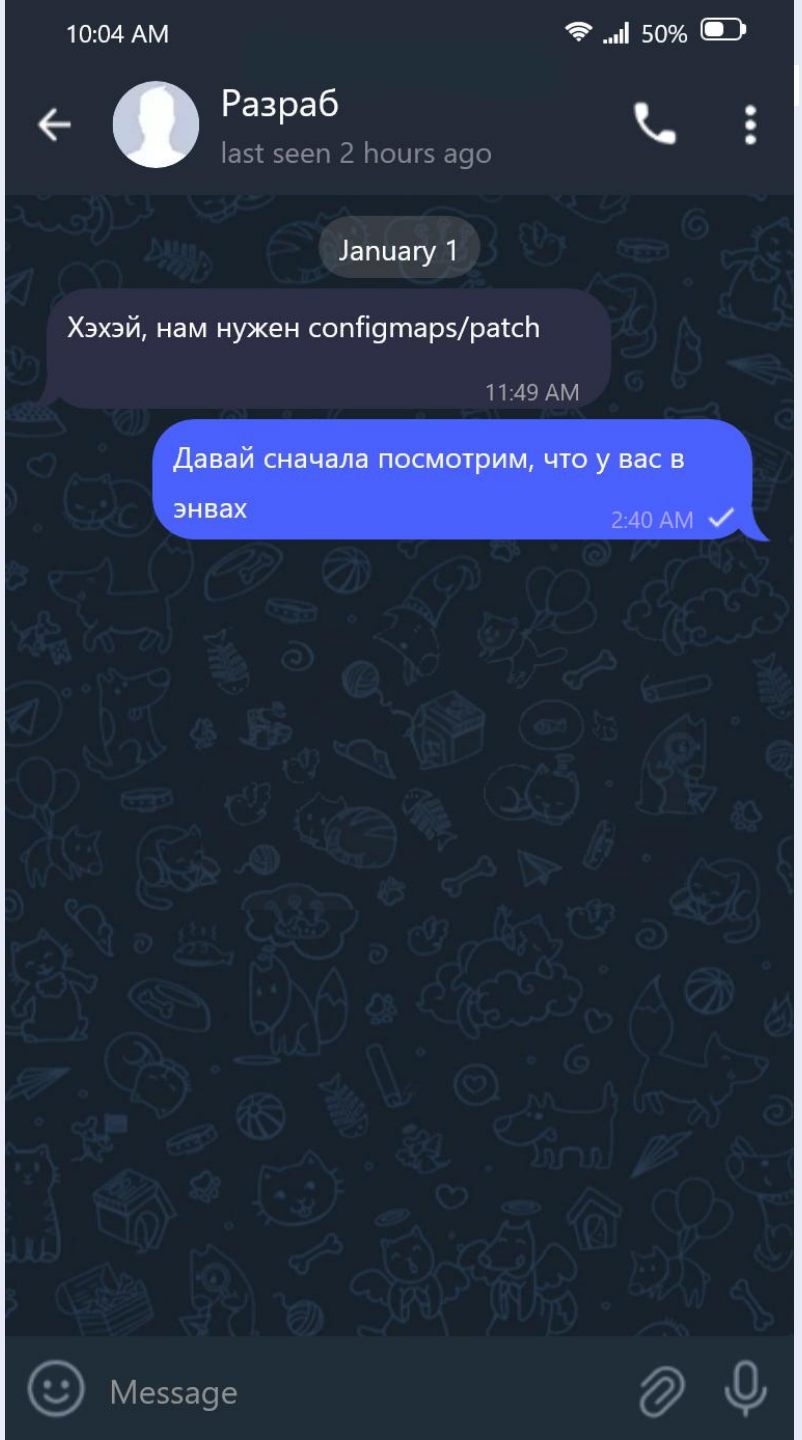
На что обратить внимание?

1. Список людей, которые имеют доступ к конфиденциальной информации, лучше держать небольшим.
2. Какие-то доступы можно дать в другом месте **(своя разработка или готовые системы).**
3. Для некоторых привилегий не нужны компромиссы, и так всё понятно. Впрочем, от ландшафта тоже зависит (и от вашей модели угроз).



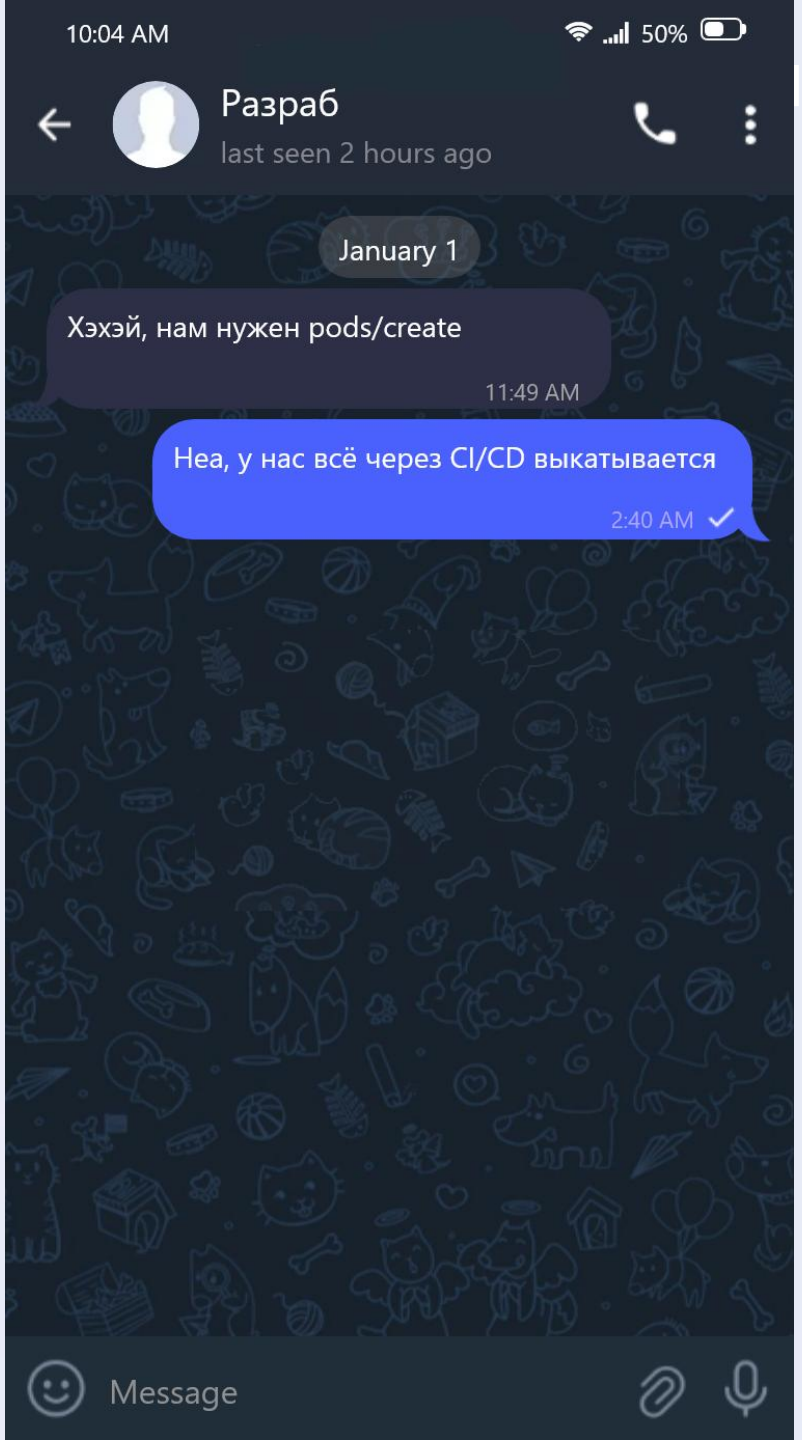
На что обратить внимание?

1. Список людей, которые имеют доступ к конфиденциальной информации, лучше держать небольшим.
2. Какие-то доступы можно дать в другом месте **(своя разработка или готовые системы).**
3. Для некоторых привилегий не нужны компромиссы, и так всё понятно. Впрочем, от ландшафта тоже зависит (и от вашей модели угроз).
4. Некоторые привилегии не считаются критичными, но могут повлиять на работу приложения.



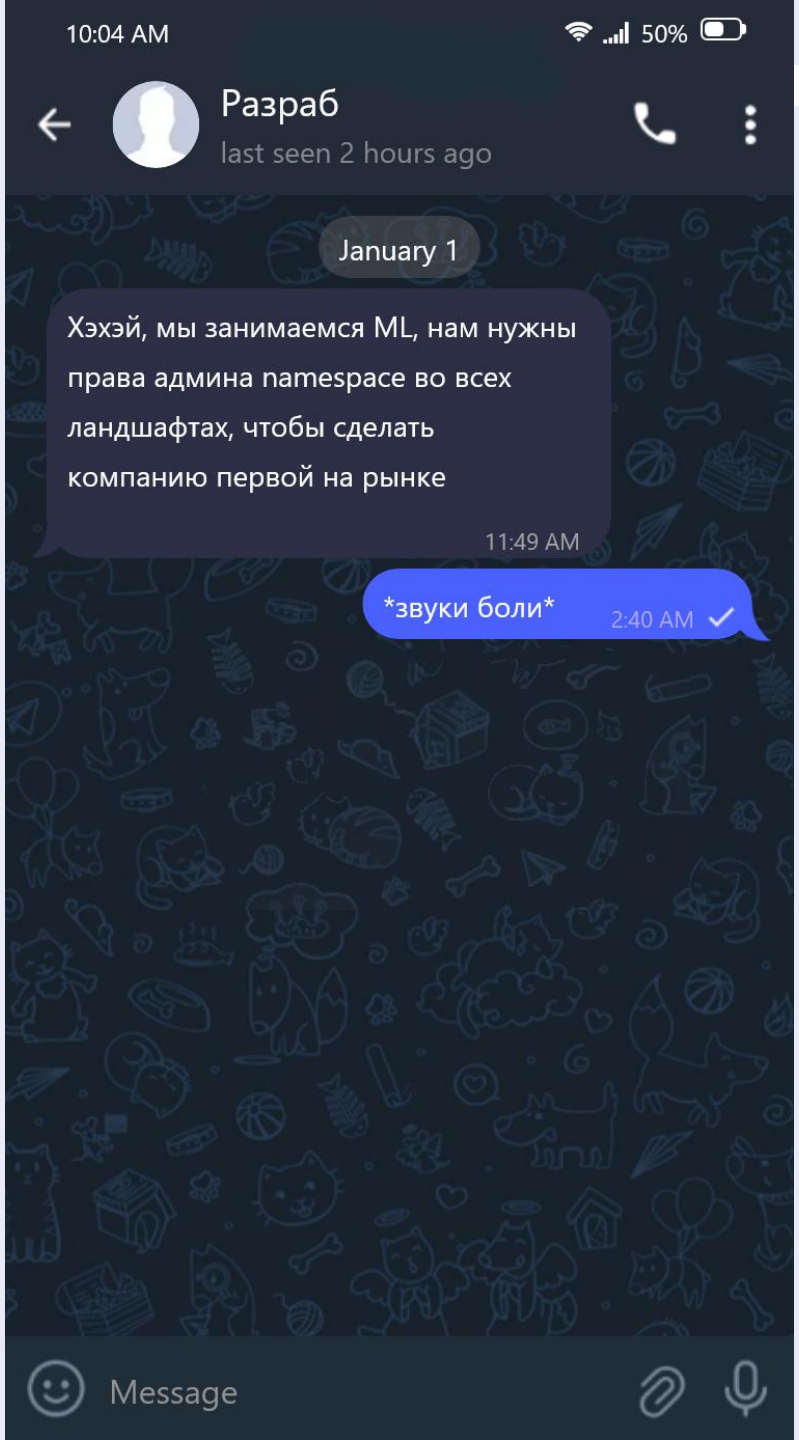
На что обратить внимание?

1. Список людей, которые имеют доступ к конфиденциальной информации, лучше держать небольшим.
2. Какие-то доступы можно дать в другом месте **(своя разработка или готовые системы).**
3. Для некоторых привилегий не нужны компромиссы, и так всё понятно. Впрочем, от ландшафта тоже зависит (и от вашей модели угроз).
4. Некоторые привилегии не считаются критичными, но могут повлиять на работу приложения.
5. Не нужно дублировать процессы.



На что обратить внимание?

1. Список людей, которые имеют доступ к конфиденциальной информации, лучше держать небольшим.
2. Какие-то доступы можно дать в другом месте **(своя разработка или готовые системы).**
3. Для некоторых привилегий не нужны компромиссы, и так всё понятно. Впрочем, от ландшафта тоже зависит (и от вашей модели угроз).
4. Некоторые привилегии не считаются критичными, но могут повлиять на работу приложения.
5. Не нужно дублировать процессы.
6. Есть ситуации, когда потребности бизнеса важнее, но что поделать.



Общий подход к исключениям в RBAC

1. Не стесняемся делать исключения, при этом не забываем торговаться.
2. Даём необходимый минимум.
3. Следим за консистентностью (pods/create + disabled Policy Engine = boom).
4. Активно применяем компенсирующие меры:
    - дополнительные ограничения Security Context;
    - перенос нагрузок в более защищенные тенанты.
5. Постоянно повышаем осведомлённость.
6. Внедрили исключения - подключили сторонние средства защиты (exec pod -> HIDS).



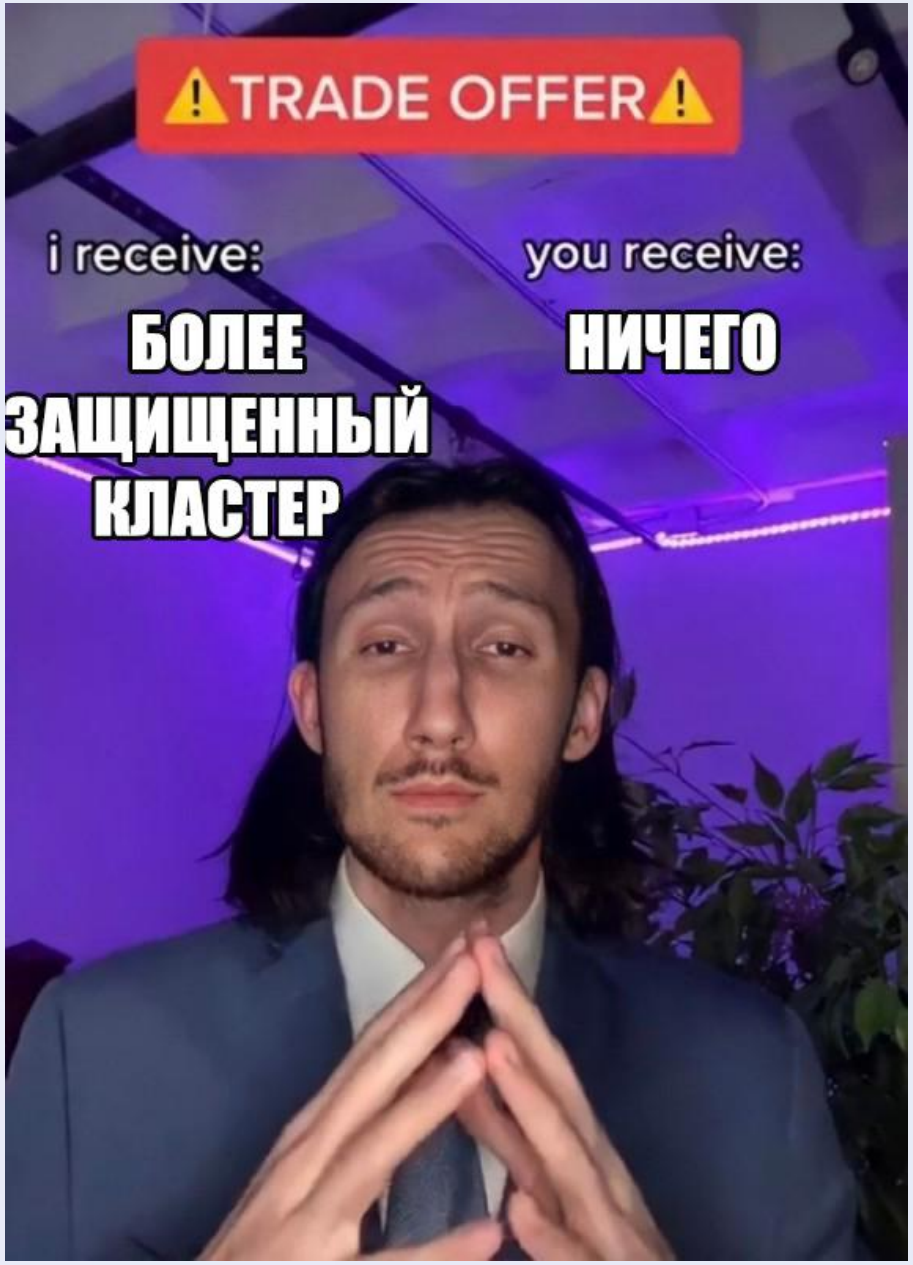
Общий подход к исключениям в RBAC

1. Не стесняемся делать исключения, при этом не забываем торговаться.
2. Даём необходимый минимум.
3. Следим за консистентностью (pods/create + disabled Policy Engine = boom).
4. Активно применяем компенсирующие меры:
    - дополнительные ограничения Security Context;
    - перенос нагрузок в более защищенные тенанты.
5. Постоянно повышаем осведомлённость.
6. Внедрили исключения - подключили сторонние средства защиты (exec pod -> HIDS).
7. Временные исключения не должны стать постоянными.



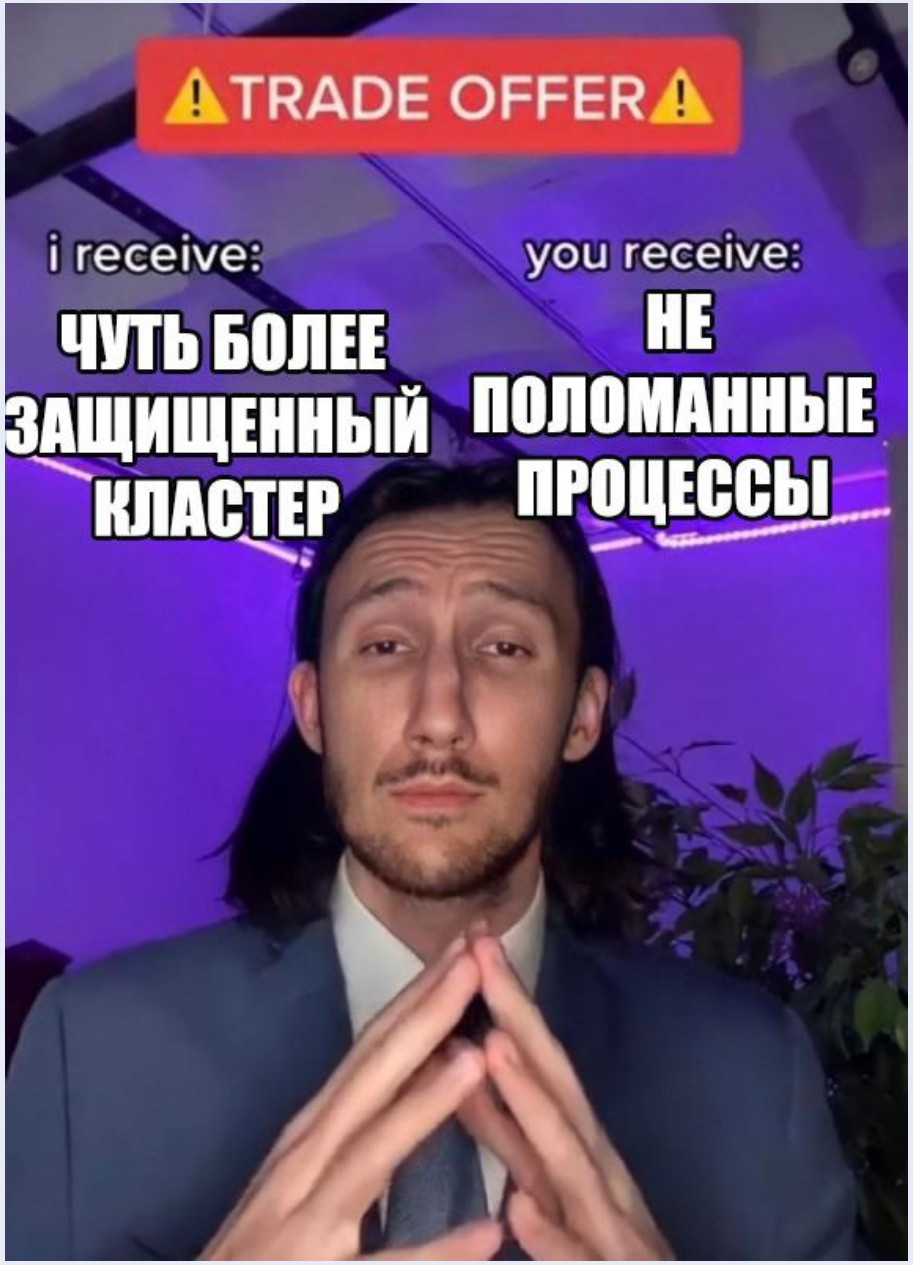
Торговаться? А как это?

1. «Окей, вы можете делать X, но мы тогда ещё делаем Y»
2. «Хм, а может, сделаем систему для X? Вам так даже удобнее будет, да и нам нервы не тратить»



Торговаться? А как это?

1. «Окей, вы можете делать X, но мы тогда ещё делаем Y»
2. «Хм, а может, сделаем систему для X? Вам так даже удобнее будет, да и нам нервы не тратить»
3. «Мы понимаем, что вам это нужно, но с правами на X вы ещё можете сделать Y, а нам бы этого не хотелось. У нас есть механизм Z для защиты от Y, если поможете нам его настроить, то права менять не будем»



Что добавлять в манифесты

- Лейбл с типом исключения - постоянное / временное.
- Время окончания действия - относительное / абсолютное.
- ID задачи, на основе которой сделали исключение.
- Владелец риска по исключению, если непонятно из контекста.

```
ex-type: temporary
ex-deadline: "2026-04-22T08:22:24Z"
ex-task: K8SEC-1337
ex-owner: iiivanov

...
janitor/expires: 2020-01-17T15:14:38Z
cleanup.kyverno.io/ttl: 2h
k8s-ttl-controller.twin.sh/ttl: "1h"
```

Что добавлять в манифесты

- Лейбл с типом исключения - постоянное / временное
- Время окончания действия - относительное / абсолютное.
- ID задачи, на основе которой сделали исключение.
- Владелец риска по исключению, если непонятно из контекста.

+ если есть автоматический уборщик, проверить/порезать его права

```
{{ if .Values.clusterRole.create }}
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
...
rules:
  - apiGroups:
      - "*"
    resources:
      - "*"
    verbs:
      - "get"
      - "list"
      - "delete"
...
{{ end }}
```

БЕКОН'26

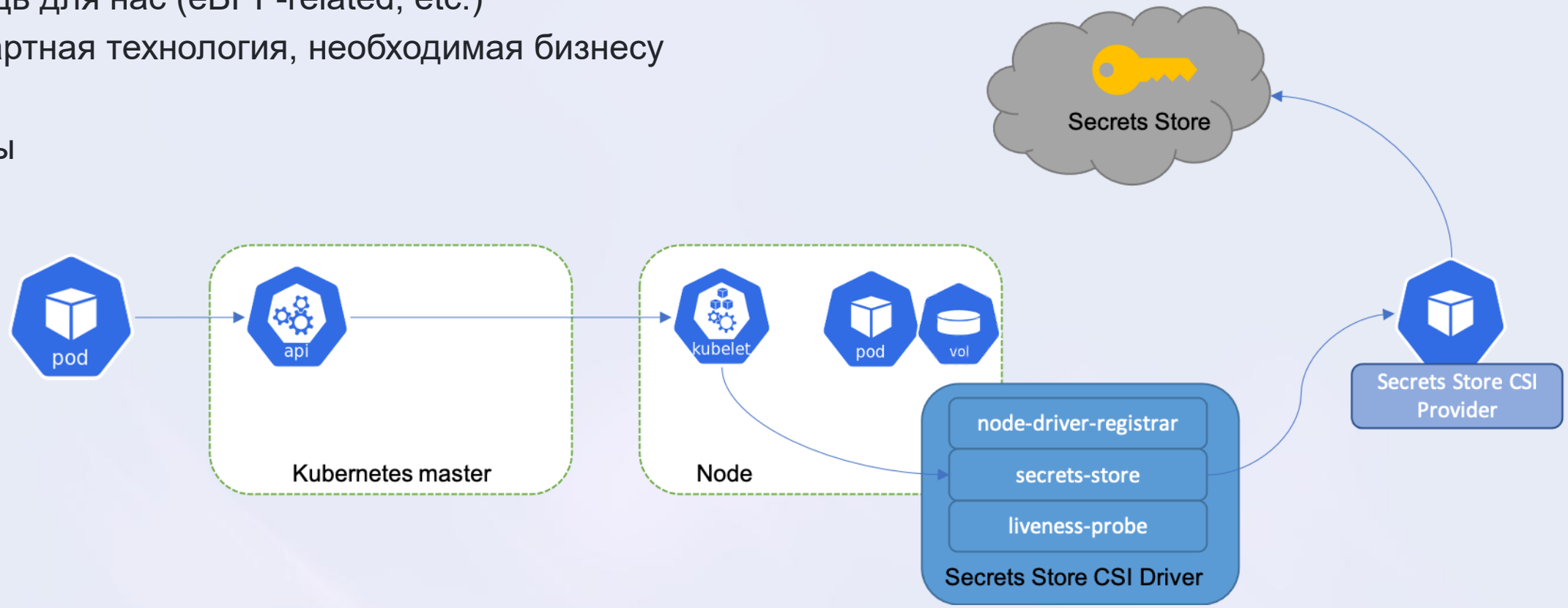
Policy Engine



2

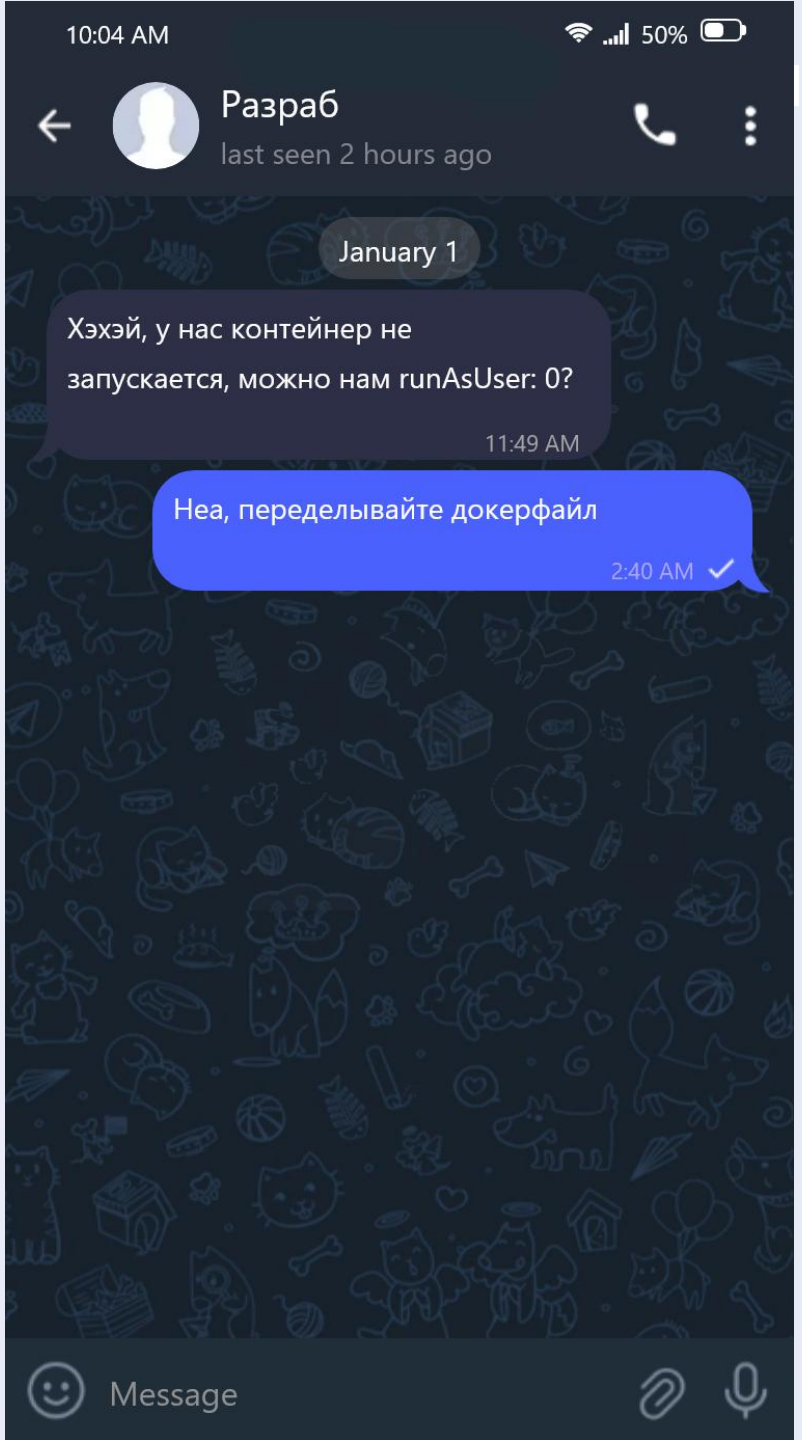
Когда могут понадобиться исключения?

- Системные нагрузки (CNI, Volume Provider)
- Что-нибудь для админов (метрики, мониторинг)
- Что-нибудь для нас (eBPF-related, etc.)
- Нестандартная технология, необходимая бизнесу
- Легаси
- Монолиты



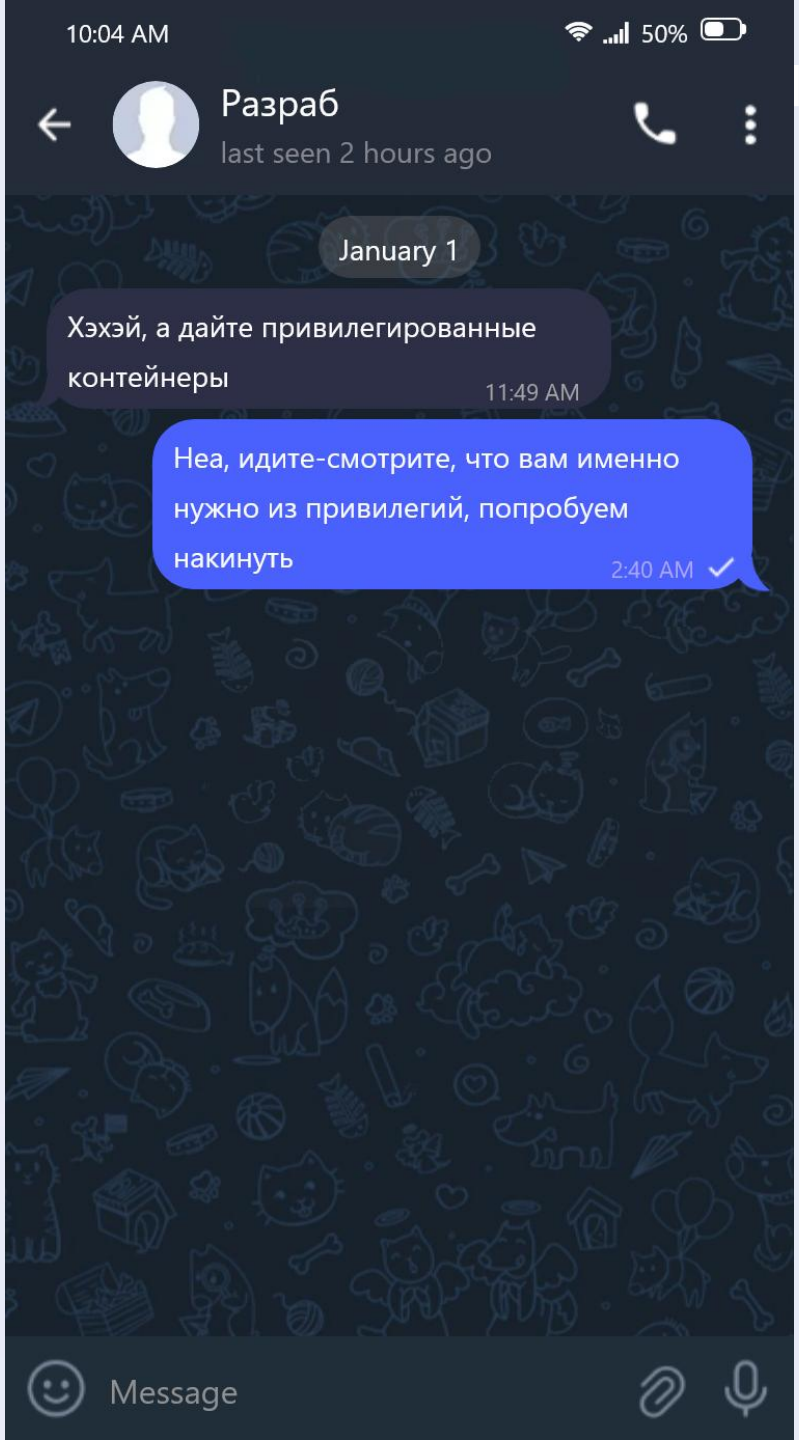
На что обратить внимание?

1. На начальном этапе внедрения Security Context команды могут упираться. К счастью, приложение почти всегда можно поправить под наши нужды.



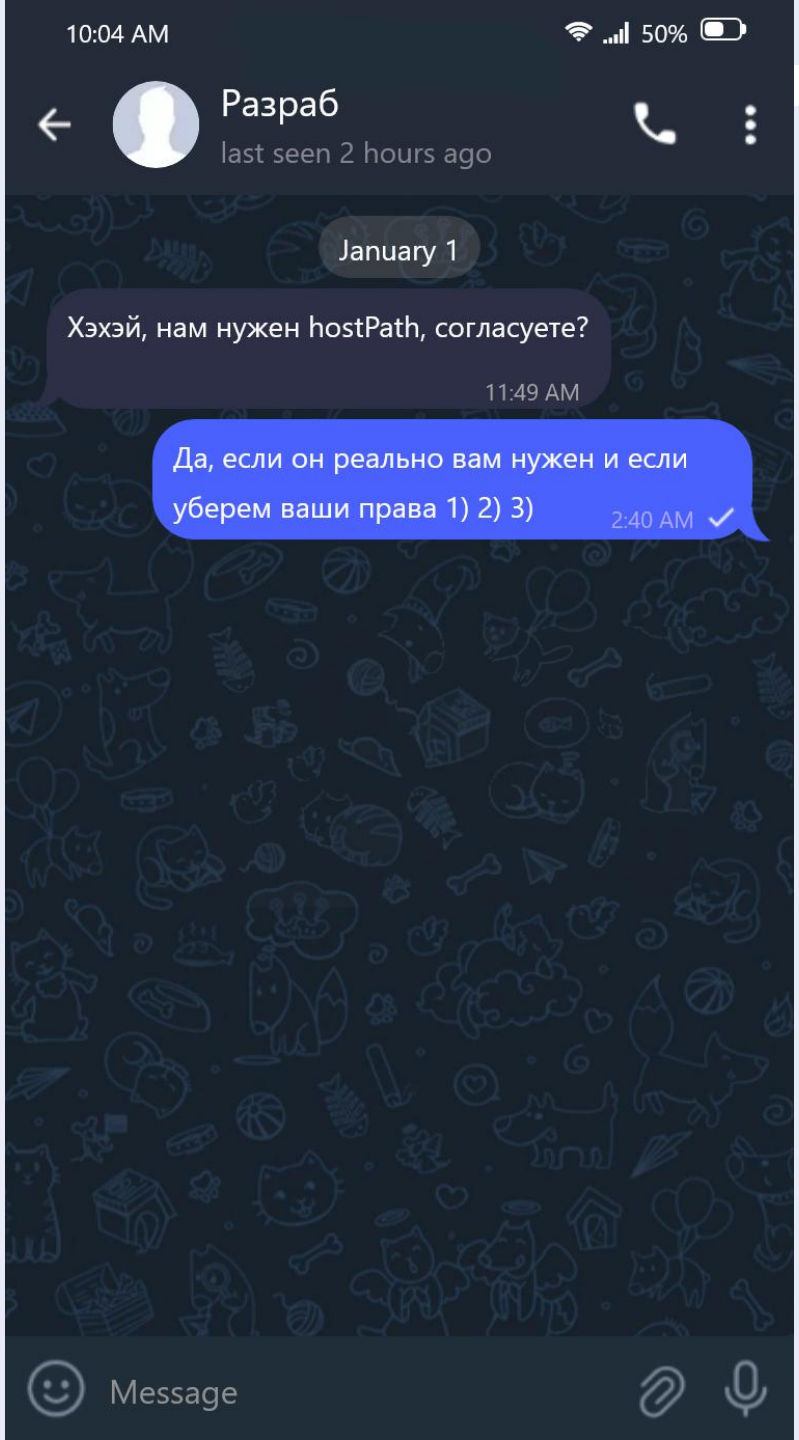
На что обратить внимание?

1. На начальном этапе внедрения Security Context команды могут упираться. К счастью, приложение почти всегда можно поправить под наши нужды.
2. Какие-то привилегии давать прям явно не стоит.



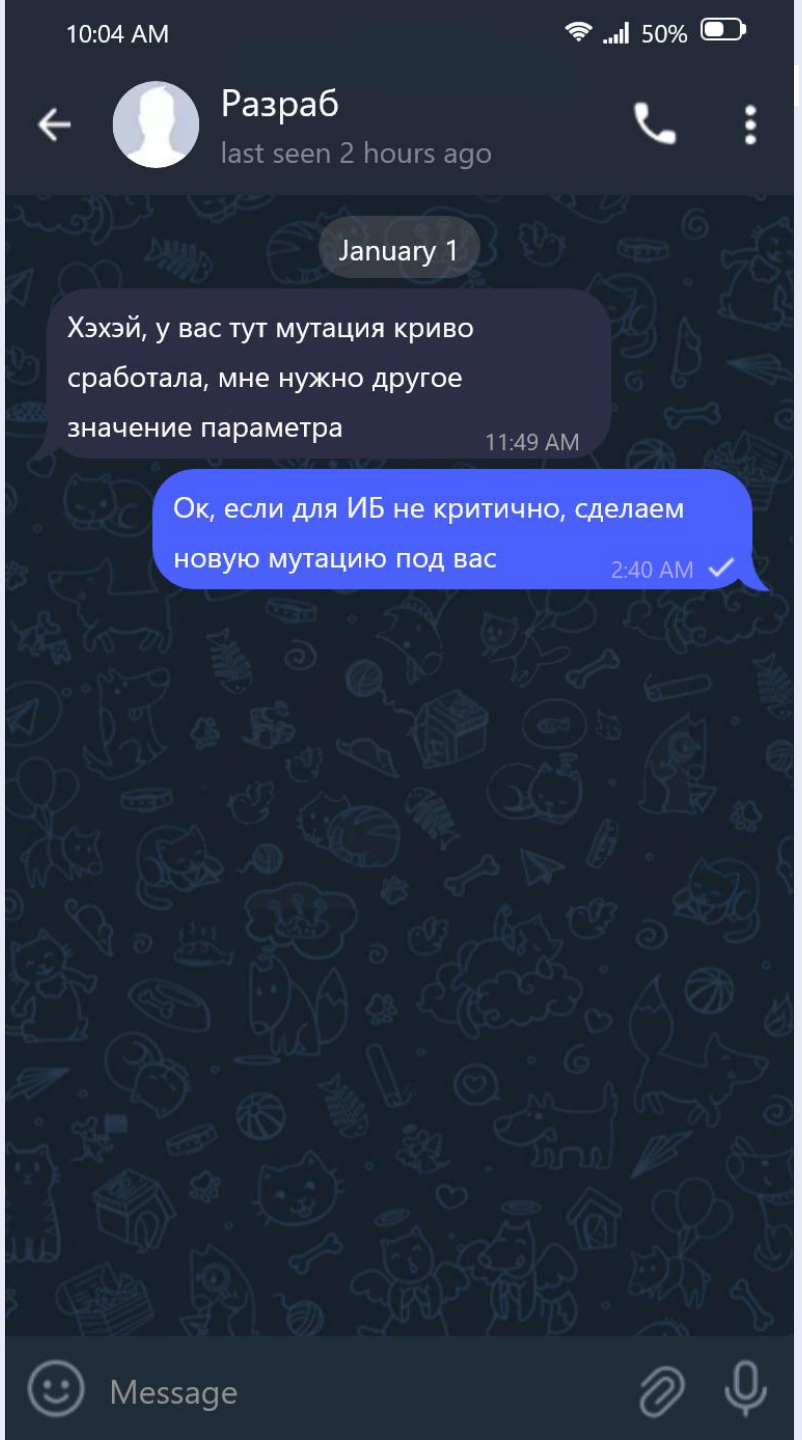
На что обратить внимание?

1. На начальном этапе внедрения Security Context команды могут упираться. К счастью, приложение почти всегда можно поправить под наши нужды.
2. Какие-то привилегии давать прям явно не стоит.
3. Лучше потратить время на более детальную проработку исключений, чем просто убрать проверяющее правило.



На что обратить внимание?

1. На начальном этапе внедрения Security Context команды могут упираться. К счастью, приложение почти всегда можно поправить под наши нужды.
2. Какие-то привилегии давать прям явно не стоит.
3. Лучше потратить время на более детальную проработку исключений, чем просто убрать проверяющее правило.
4. Бонусные очки за взаимодействие с командами **не** в стиле «У вас не работает — ваши проблемы».



А может, у вас уже есть исключения?

Что будет, если убрать label?

А можно ли его убрать? А кому?

А как это сработает на Deployment? Cronjob?

```
match:
  any:
  - resources:
    kinds:
    - Pod
    namespaceSelector:
      matchLabels:
        kyverno: enabled
```

Вариант - исключить namespace целиком


```
apiVersion: config.gatekeeper.sh/v1alpha1
kind: Config
metadata:
  name: config
  namespace: "gatekeeper-system"
spec:
  match:
    - excludedNamespaces: ["kube-*", "my-namespace"]
      processes: ["*"]
    - excludedNamespaces: ["audit-webhook-sync-excluded-ns"]
      processes: ["audit", "webhook", "sync"]
    - excludedNamespaces: ["mutation-excluded-ns"]
      processes: ["mutation-webhook"]
...
```

Вариант - исключить namespace целиком

```
kind: PolicyException
...
spec:
  exceptions:
    - policyName: psa
      ruleNames:
        - restricted
  match:
    any:
      - resources:
          namespaces:
            - delta
  podSecurity:
    - controlName: 'Running as Non-root'
```

Вариант - исключить по wildcard

- Компромиссно-удобно
- Эффективно?

```
kind: PolicyException
...
  namespace: exclusions
...
  exceptions:
  - policyName:
    disallow-host-namespaces
    ruleNames:
    - host-namespaces
  match:
  ...
    names:
    - pod-name*
```

Вариант - исключить по wildcard

- От регулярки зависит, насколько сложно обойти исключение

```
podSecurity:
- controlName: Capabilities
  images:
  - '*/istio/proxyv2*'
  - '*/linkerd/proxy-init*'
  restrictedField:
    ...capabilities.add
  values:
  - NET_ADMIN
  - NET_RAW
```

Вариант - исключить конкретный объект

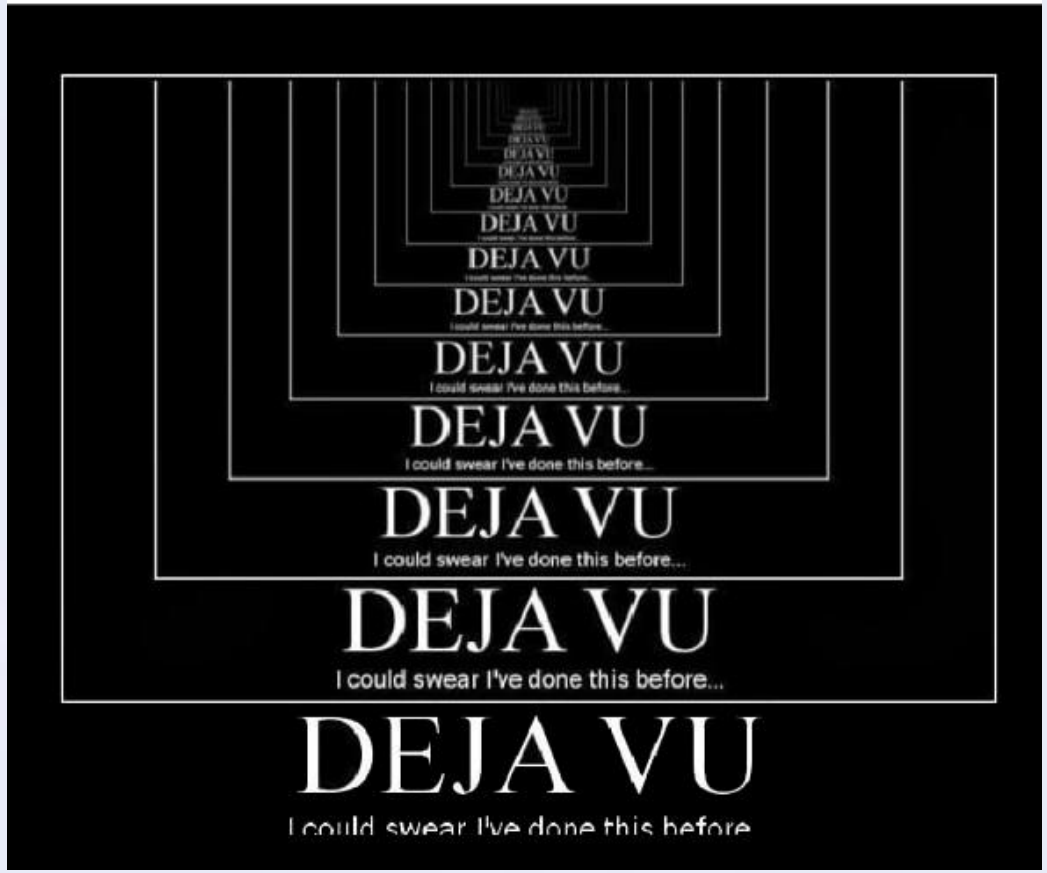
- Здорово, но чего-то ещё не хватает

```
spec:
  policyRefs:
  - name: require-prod-label
    kind: ValidatingPolicy
matchConditions:
- name: skip-by-name
  expression:
    "object.metadata.name ==
      'skipped-deployment'"
```

Вариант - исключить конкретный объект и прилепить TTL

- Ну мёд
- Так, а что делать с IaC?

TTL прошёл.
Политика опять накатилась.



Общий подход к исключениям в Policy Engine

1. Стремимся делать исключения атомарными.
2. Регулярки - с осторожностью + протестировать.
3. Закрываем всё, что может помочь обойти Policy Engine.
4. Даже постоянные исключения стоит периодически пересматривать.
5. Если есть автоудаление исключений, информируем команду.
6. Если используем готовые наборы правил, **проверяем**, что там есть.
7. Soft auto-delete > hard auto-delete.

```
...
name: require-image-tag
match:
  any:
    resources:
      kinds:
      - Pod
    validate:
      message: "An image tag is required."
      foreach:
      - list: "request.object.spec.containers"
        pattern:
          image: "*:*"
...
```

БЕКОН'26

Network Policy



3

Когда могут понадобиться исключения?

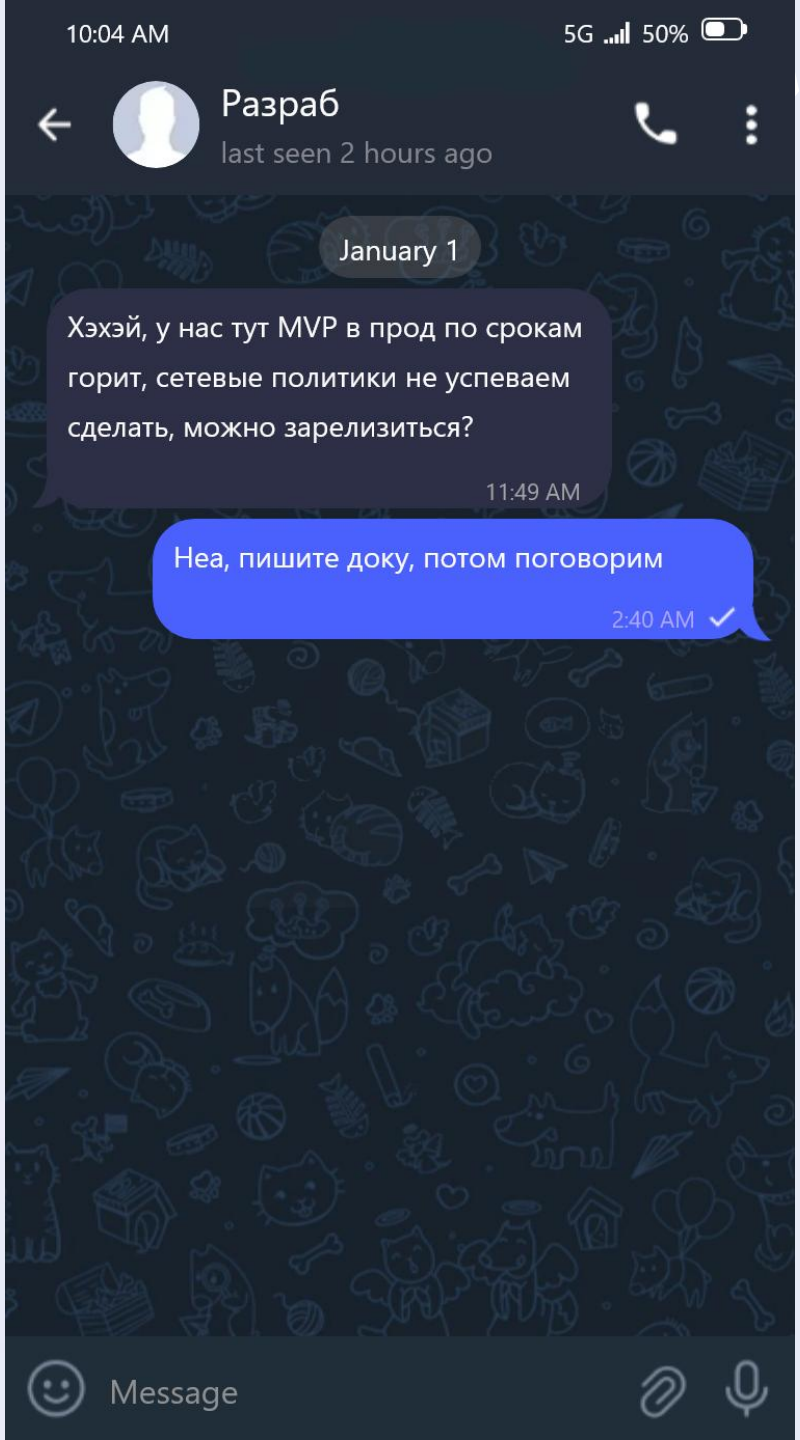
- Разработчики ещё не определились, какой микросервис куда ходит.
- Или «забыли» разработать доку (особенно инфровые сервисы).
- Или «Аааааааааа, срочно разработать и выкатить, всё потом оформим»
- Или «Да тут минорные изменения, жалко, что ли?»
- Интеграционные сервисы.



Использована картинка из мультфильма «Вовка в тридевятом царстве»

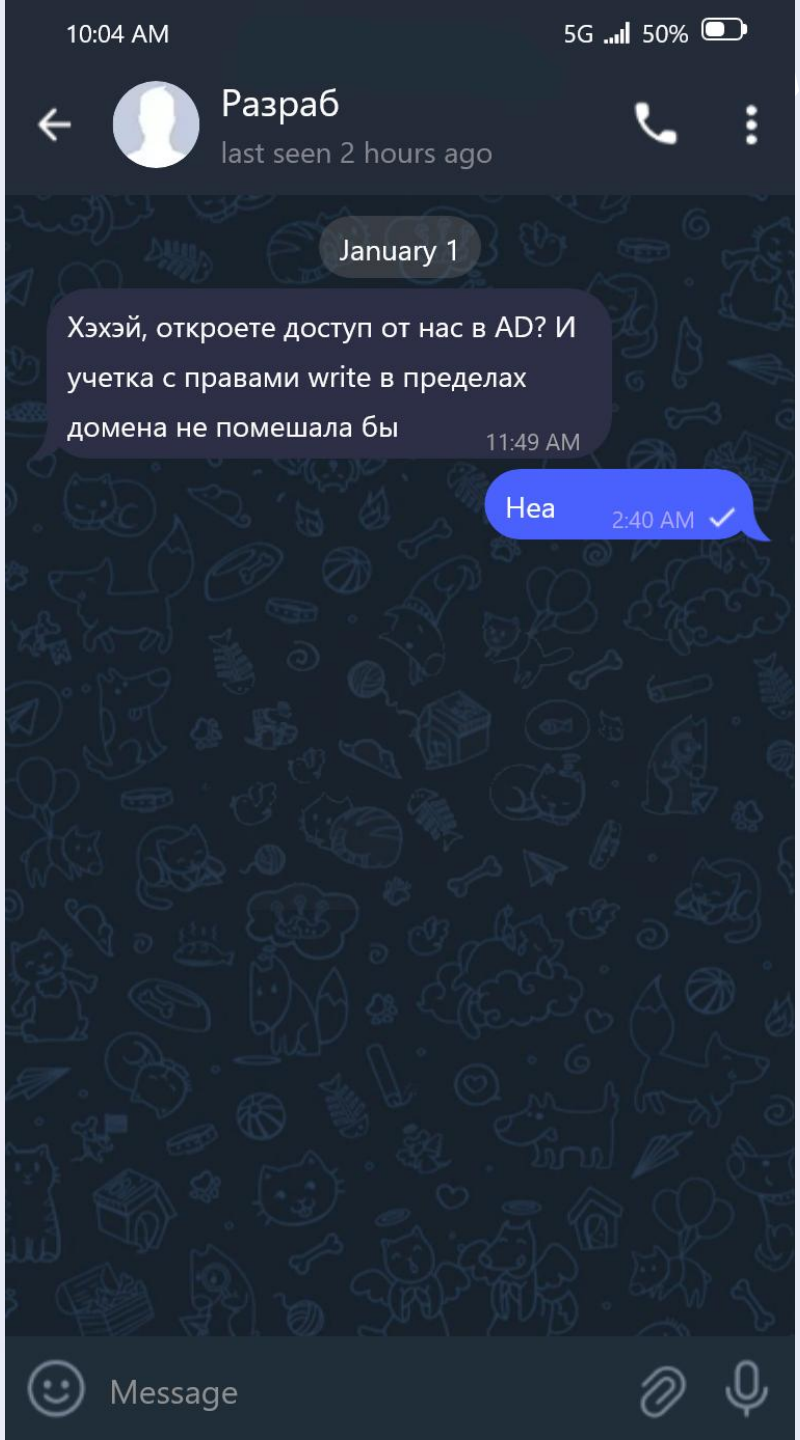
На что обратить внимание?

1. Чем лучше документация приложения, тем проще написать под него сетевые политики.



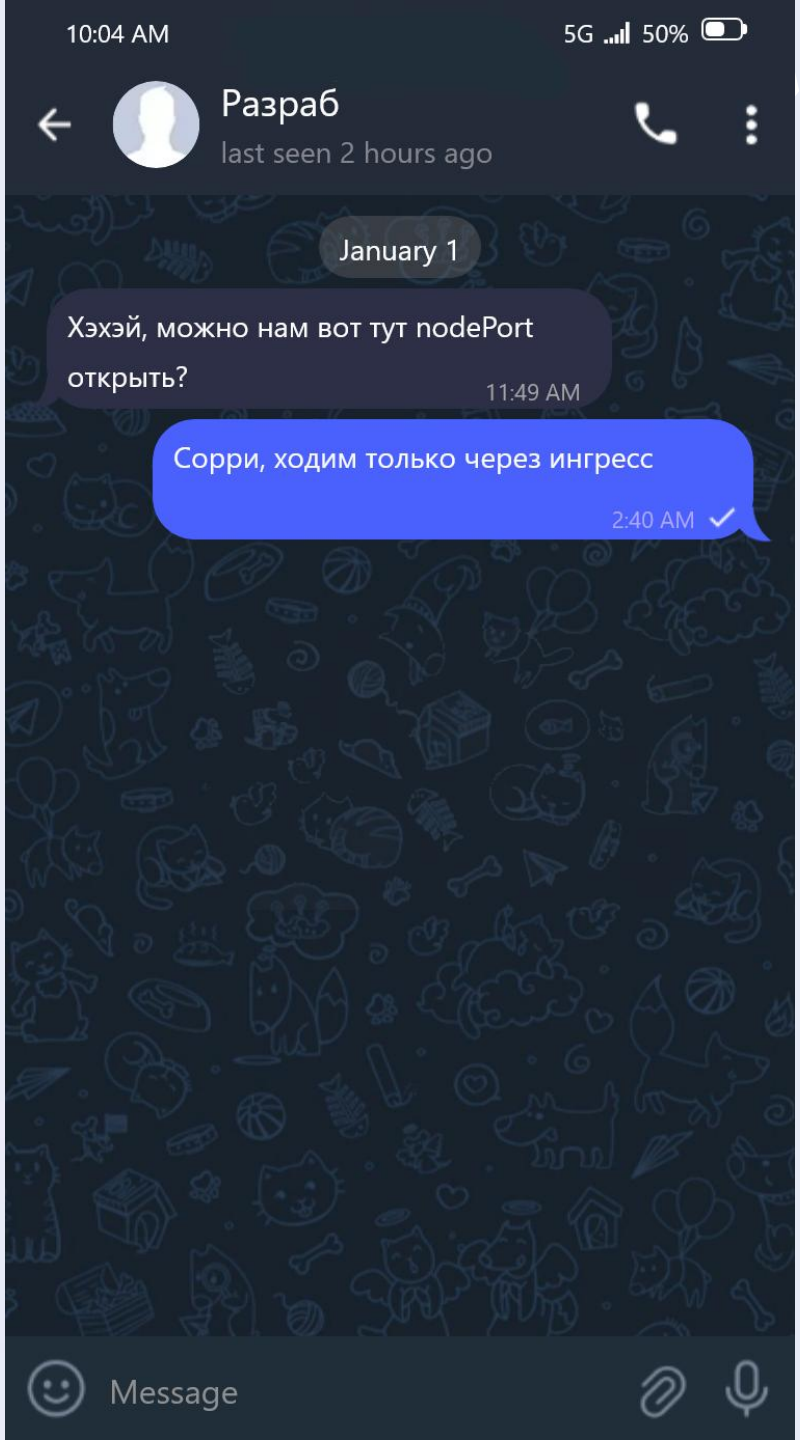
На что обратить внимание?

1. Чем лучше документация приложения, тем проще написать под него сетевые политики.
2. Любой доступ до инфраструктурных сервисов - сильное расширение поверхности атаки.



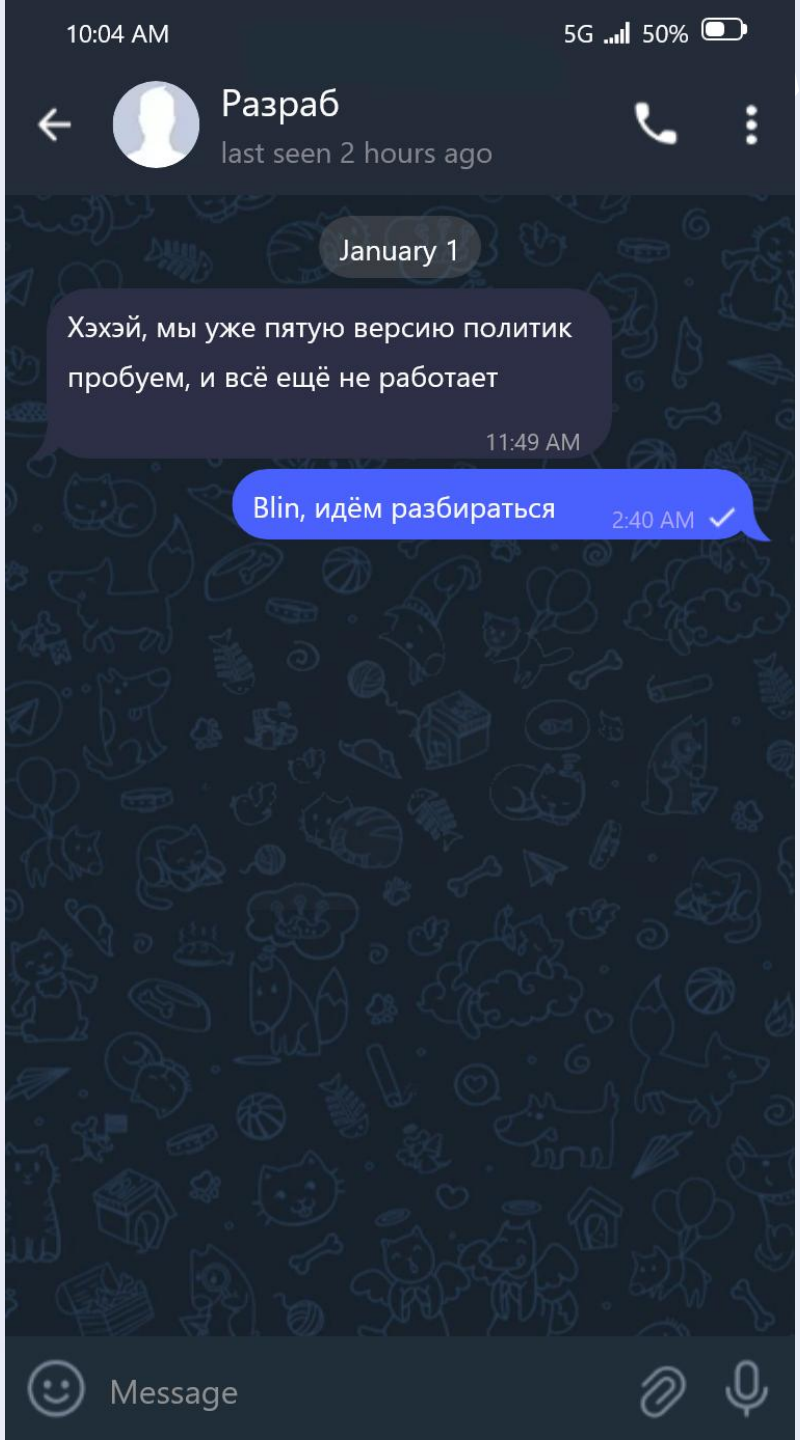
На что обратить внимание?

1. Чем лучше документация приложения, тем проще написать под него сетевые политики.
2. Любой доступ до инфраструктурных сервисов - сильное расширение поверхности атаки.
3. Некоторые варианты сетевого доступа не обрабатываются сетевыми политиками.



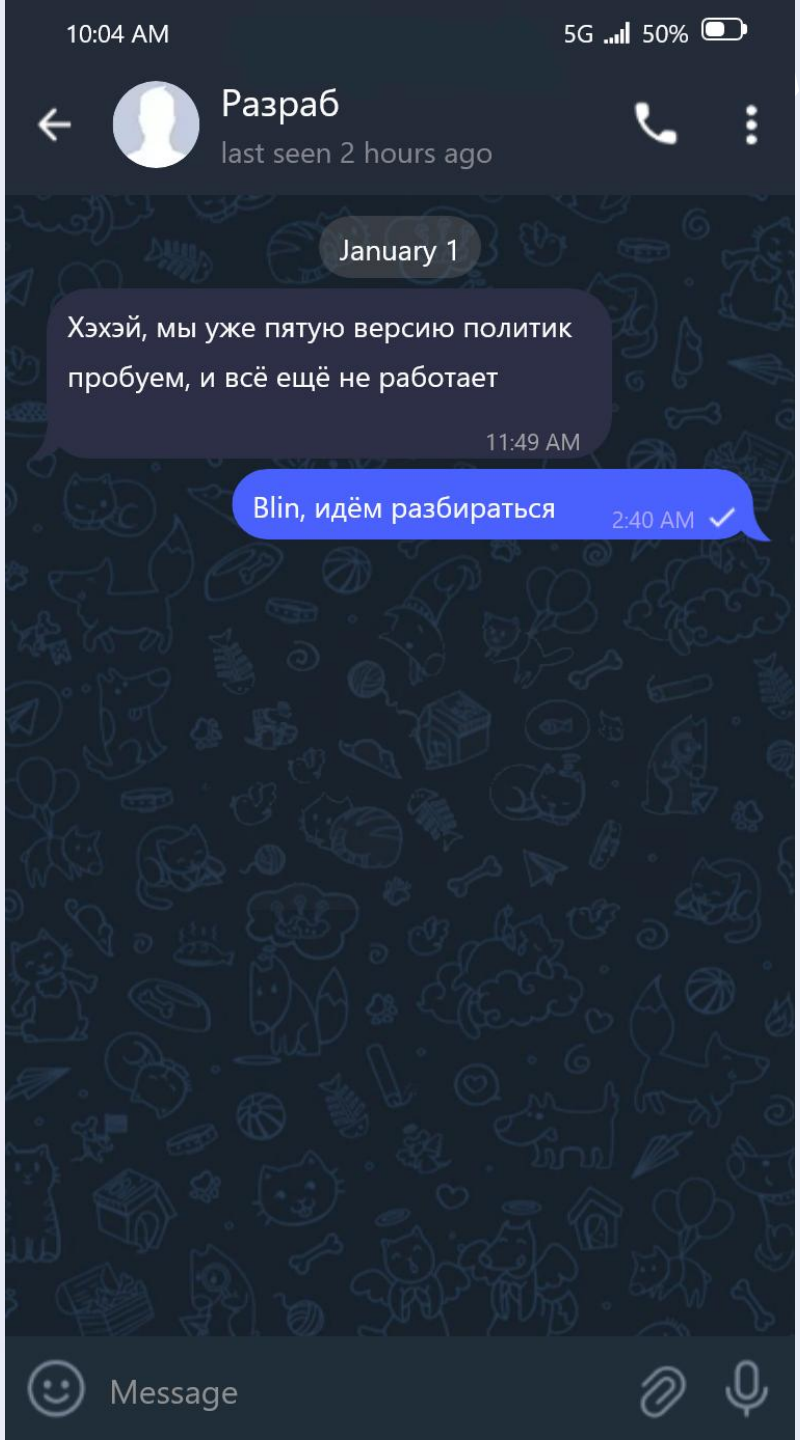
На что обратить внимание?

1. Чем лучше документация приложения, тем проще написать под него сетевые политики.
2. Любой доступ до инфраструктурных сервисов - сильное расширение поверхности атаки.
3. Некоторые варианты сетевого доступа не обрабатываются сетевыми политиками.
4. Чрезмерная детализация правил, если не автоматизирована, может тормозить процесс.



На что обратить внимание?

1. Чем лучше документация приложения, тем проще написать под него сетевые политики.
2. Любой доступ до инфраструктурных сервисов - сильное расширение поверхности атаки.
3. Некоторые варианты сетевого доступа не обрабатываются сетевыми политиками.
4. Чрезмерная детализация правил, если не автоматизирована, может тормозить процесс.
5. Ручная работа увеличивает вероятность ошибок.



Как автоматизировать?

- Сделать стандартные профили доступа для разных типов приложений.
- Бегать и исправлять слишком широкие доступы.
- Бегать и добавлять исключения.



В меме использована картинка из мультсериала «Симпсоны»

Как автоматизировать?

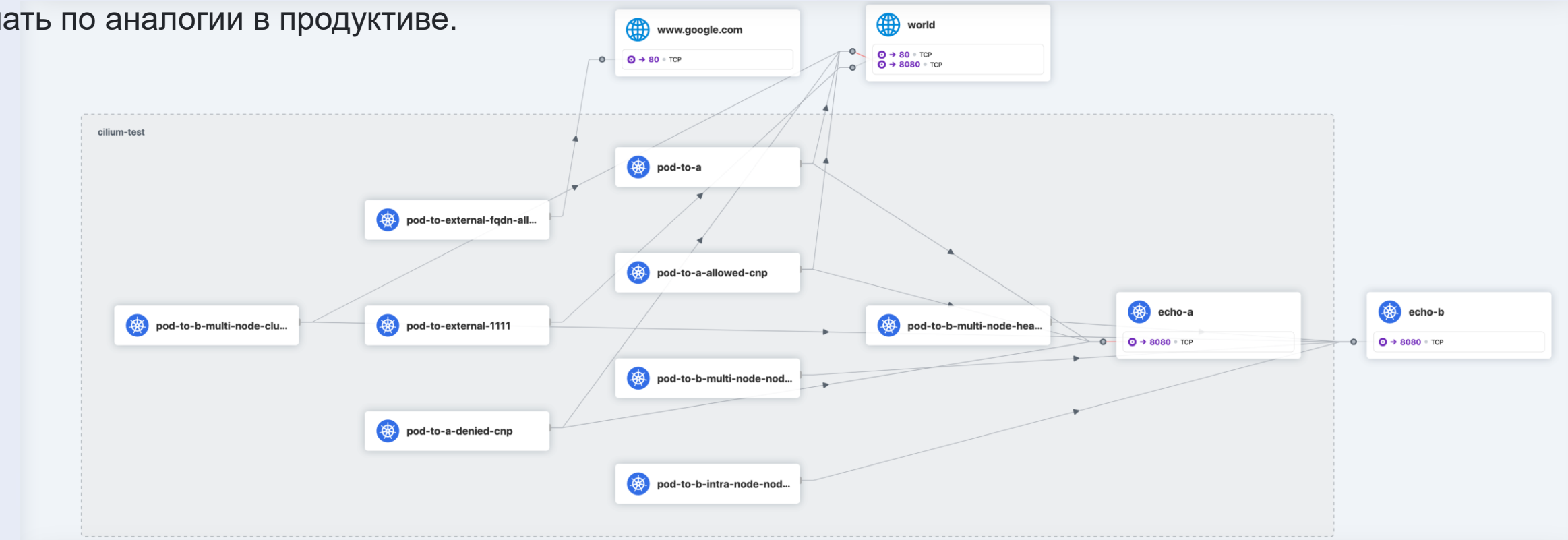
- Заставить разработчиков дополнять документацию.
- Генерировать политики на основе документации.
- Ловить и исправлять ошибки разработчиков в постоянном режиме.
- Вспоминать, что из сгенерированного относится к исключениям.



В меме использованы кадры из фильма «Волшебная страна»

Как автоматизировать?

- Развернуть приложение в тестовой среде.
- Собрать трафик (например, в hubble для cilium).
- Валидировать правила.
- Сделать по аналогии в продуктиве.



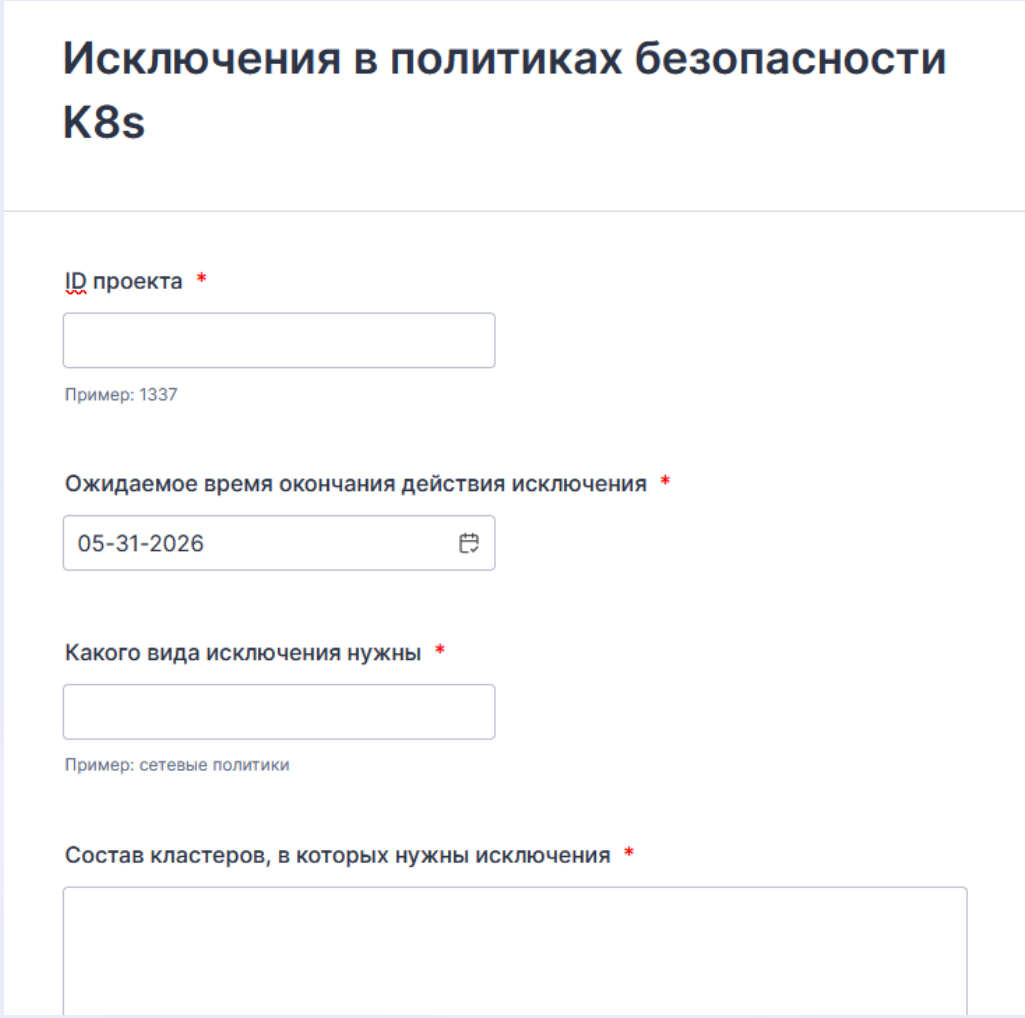
Общий подход к исключениям в сетевые политики

1. Выше детализация — больше мороки с исключениями.
2. Ниже детализация — больше мороки с исключениями.
3. Лучше документация — лучше политики.
4. Прежде всего решаем вопросы с доступом к системным ns/pod.
5. Автоматизируем!



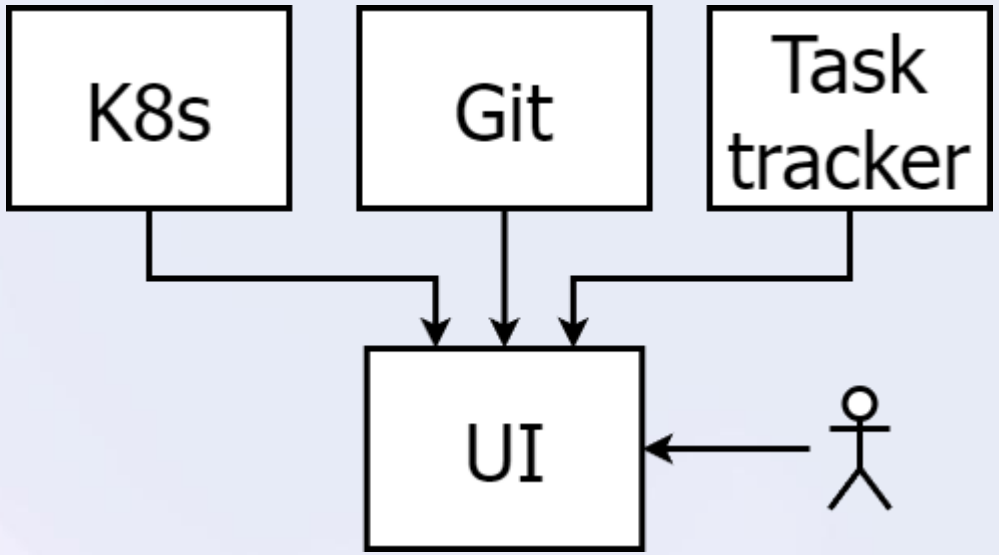
Делаем процесс удобным для команд

- Шаблон задачи + документация.
- Понятный процесс и критерии согласования.
- Сопровождение, особенно на ранних этапах.

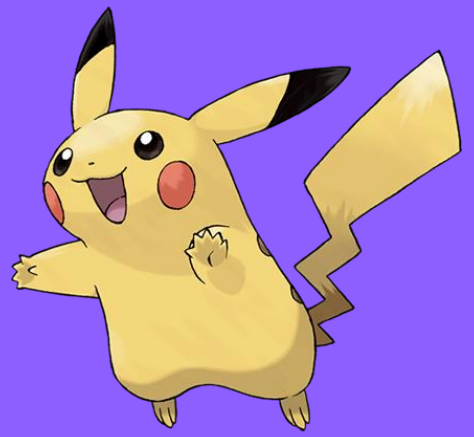


Контролируем исключения

- Собираем данные из нескольких источников.
- В K8s — контроллер / оператор / скрипт.
- В Task tracker — через эпики / отдельный проект.
- В Git — скан репозиториев с IaC.
- Внимательно следим за дрифтом исключений.



Эволюция процесса

			
Формализация	Средняя	Высокая	Высокая
Автоматизация	Низкая	Средняя	Высокая
Покрытие	Низкое	Среднее	Полное
Отчетность	Нет	Периодическая	Real-time
Метрики	Нет	Количество + статус	+ статистика просроченных + компенсирующие меры + продления

Использованы картинки из аниме «Pokemon»

Выводы

БЕКОН'26

Исключения сделать можно, и они будут даже эффективными, но для этого нужно время, заинтересованность безопасника в «сделать по красоте» и навыки.

Чтобы сделать процесс выдачи исключений удобнее для команд, нужно дать им знать, как запросить исключение (шаблон заявки + TTL).

Если не смотреть на исключения системно, они вряд ли получатся эффективными.

БЕКОН'26

КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Спасибо за внимание!



@V_Kunavin