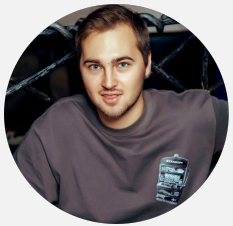
Information Security

WILDBERRIES

Проектируем безопасный сервис аутентификации



Александр Савин
InfraSec Lead

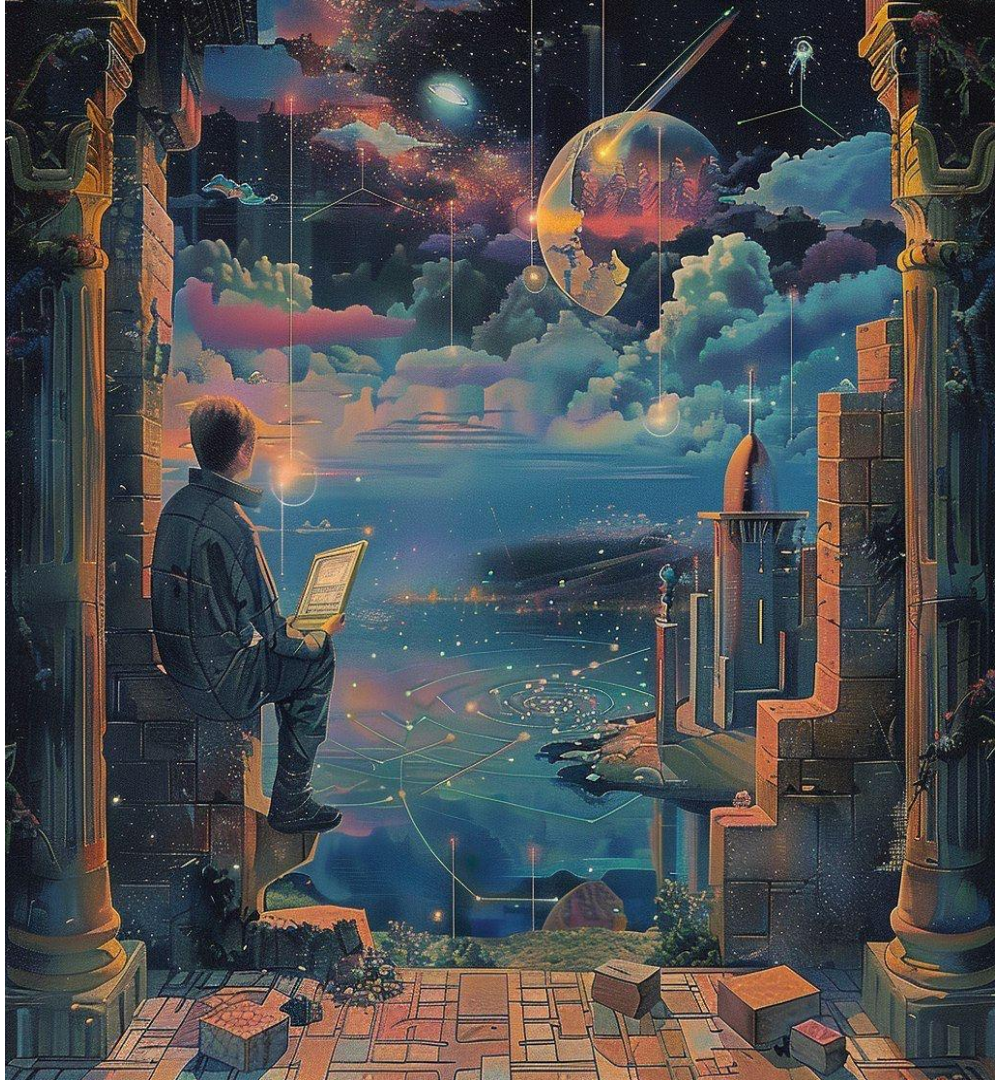
О спикере

Александр Савин
InfraSec Lead

- Работаю в Wildberries с 2022 года
- С февраля 2024 занимаюсь безопасностью Инфраструктуры
- Ранее развивал процессы Application Security

Аутентификация в современных приложения

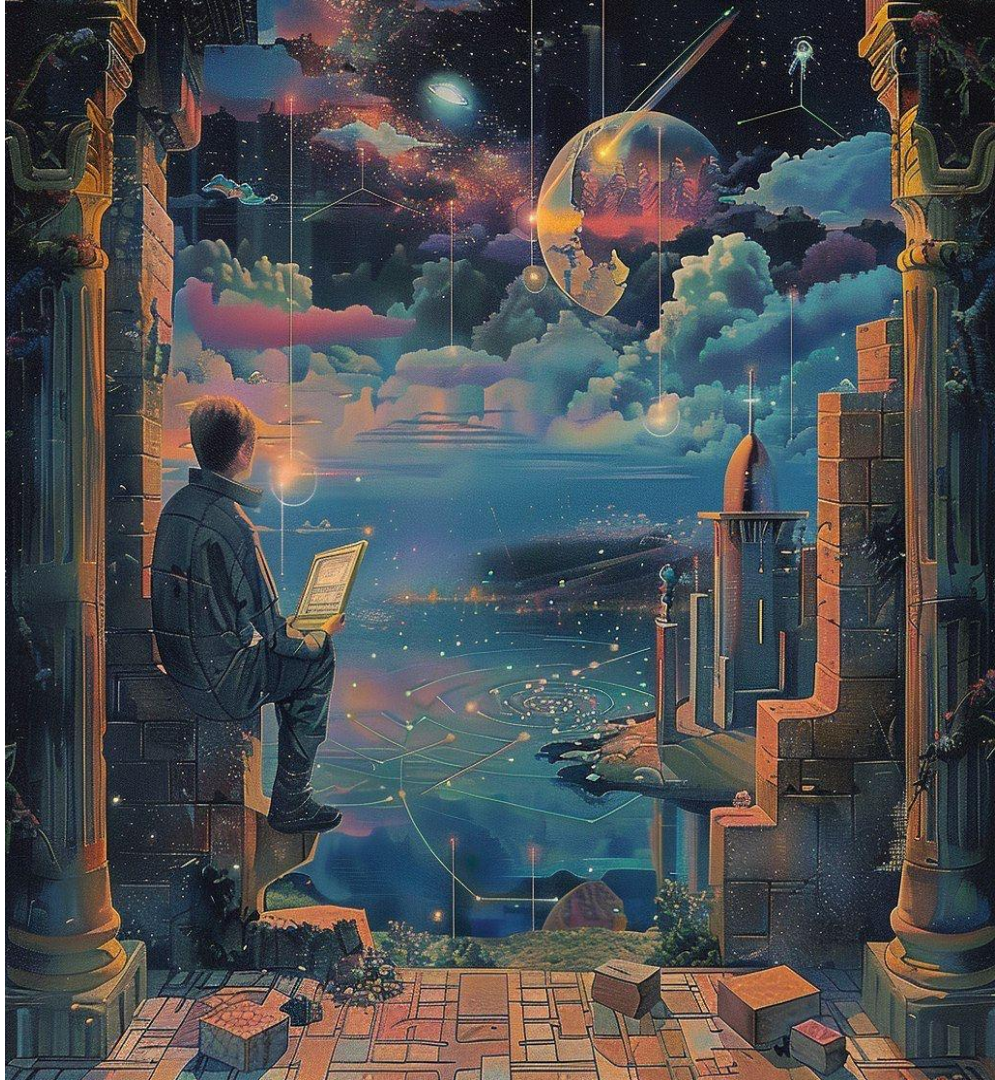
Современные веб приложения не обходятся без сервиса аутентификации пользователей



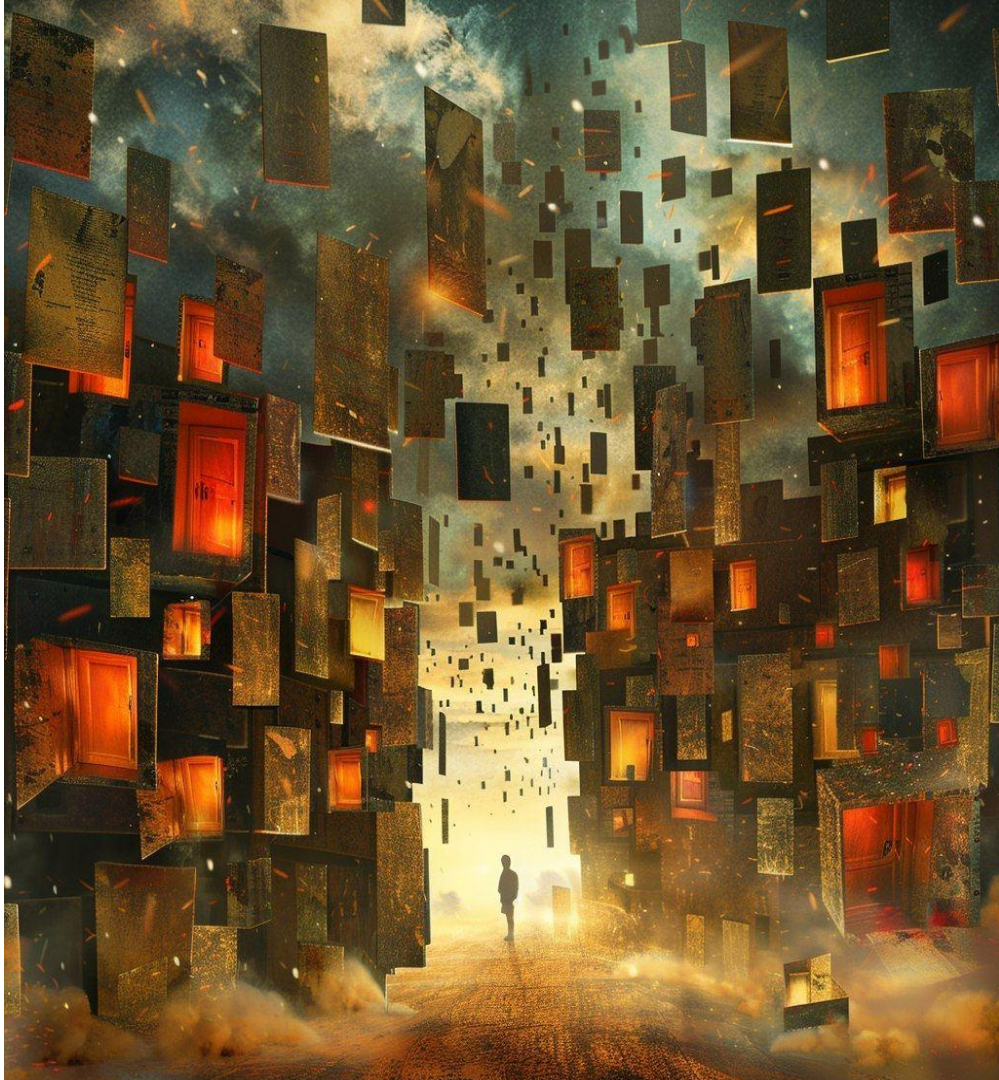
Аутентификация в современных приложения

Современные веб приложения не обходятся без сервиса аутентификации пользователей

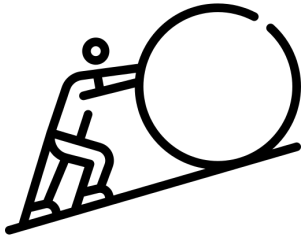
Сервис критичен с точки зрения ИБ



У каждого приложения компании может быть свой сервис аутентификации



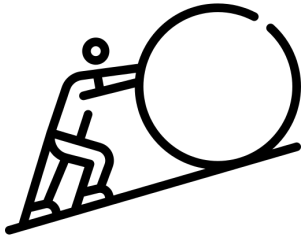
Минусы разрозненной аутентификации



Сложно развивать ИБ фичи

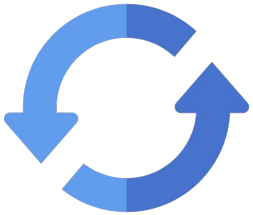
При внедрении нового функционала в сервис аутентификации одного приложения, другим придется делать тот же функционал повторно.

Минусы разрозненной аутентификации



Сложно развивать ИБ фичи

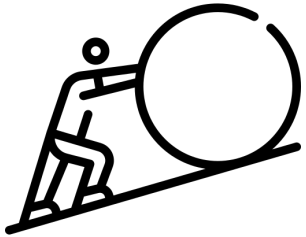
При внедрении нового функционала в сервис аутентификации одного приложения, другим придется делать тот же функционал повторно.



Отсутствует возможность Force Update

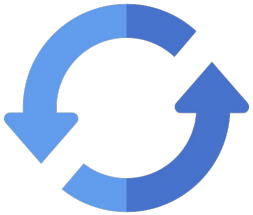
При обнаружении проблем с безопасностью в одном из приложений мы не сможем одним фиксом митигировать подобные баги в остальных.

Минусы разрозненной аутентификации



Сложно развивать ИБ фичи

При внедрении нового функционала в сервис аутентификации одного приложения, другим придется делать тот же функционал повторно.



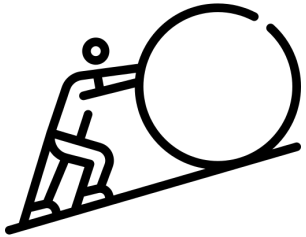
Отсутствует возможность Force Update

При обнаружении проблем с безопасностью в одном из приложений мы не сможем одним фиксом митигировать подобные баги в остальных.

Множество паролей

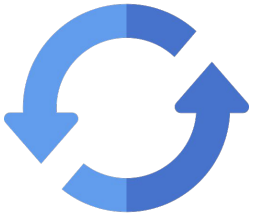
Каждому пользователю необходимо создавать новый пароль для каждого приложения компании.

Минусы разрозненной аутентификации



Сложно развивать ИБ фичи

При внедрении нового функционала в сервис аутентификации одного приложения, другим придется делать тот же функционал повторно.



Отсутствует возможность Force Update

При обнаружении проблем с безопасностью в одном из приложений мы не сможем одним фиксом митигировать подобные баги в остальных.

Множество паролей

Каждому пользователю необходимо создавать новый пароль для каждого приложения компании

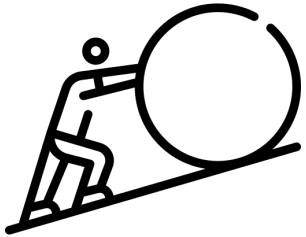
Подробное логирование

При расследовании подозрительных событий необходимо иметь подробный лог аутентификации. Добиться хорошо подробного лога в N сервисах почти невозможно.

Каким иным подходом можно воспользоваться?

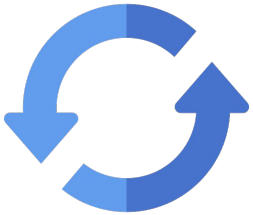
Единый сервис аутентификации

Минусы разрозненной аутентификации



~~Сложно развивать ИБ фичи~~

При внедрении нового функционала в сервис аутентификации одного приложения, другим придется делать тот же функционал повторно.



~~Отсутствует возможность Force Update~~

При обнаружении проблем с безопасностью в одном из приложений мы не сможем одним фиксом митигировать подобные баги в остальных.

~~Множество паролей~~

Каждому пользователю необходимо создавать новый пароль для каждого приложения компании.

~~Подробное логирование~~

При расследовании подозрительных событий необходимо иметь подробный лог аутентификации. Добиться хорошо подробного лога в N сервисах почти невозможно.

Требования ИБ

Сессионные токены

Сессионные токены в WEB приложении

1

Access/Refresh токены

Время жизни Access:

10 – 30 минут

Время жизни Refresh:

6 – 12 месяцев

Сессионные токены в WEB приложении

1

Access/Refresh токены

Время жизни Access:

10 – 30 минут

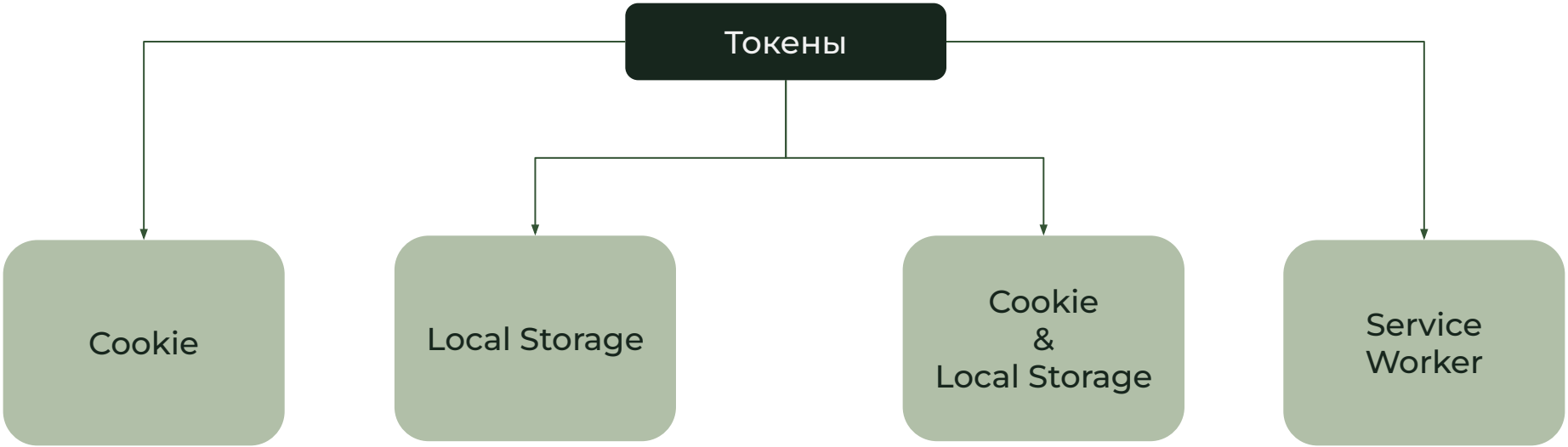
Время жизни Refresh:

6 – 12 месяцев

2

Где будем хранить токены?

Где можно хранить токены в WEB приложении



Плюсы

1) Защитные флаги:
   - SameSite
   - httpOnly
   - Secure

2) Автоматически отправляются со всеми запросами

Минусы

1) Не можем гибко проставлять cookie на определенные поддомены, только wildcard: *.example.com

Плюсы

1) Хранится для определенного origin
2) Токены будут отправляться к кастомных заголовках -> получим защиту от CSRF атак
3) Разработчикам легко управлять токенами

Минусы

1) Нет защитных флагов:
   - SameSite
   - httpOnly
   - Secure
2) Уязвимо к XSS

Плюсы

1) Не уязвим к XSS атакам, если используется только для хранения токенов

Минусы

1) На одном скоупе может быть только один SW

2) При падении SW отвалится аутентификация
3) SW могут использовать сторонние библиотеки
4) При обновлении SW пользователей будет разлогинивать

Refresh Token

1) В Cookie
2) С защитными флагами:
   a) httpOnly
   b) SameSite
   c) Secure
3) Выставлен только на путь обновления токенов и logout

Cookie & Local Storage

Доступ осуществляется по паре:

- Access token (Local Storage)
- Key (Cookie)

Cookie Key:

- рандомная строка
- с защитными флагами
- Wildcard scope

```
{"key": "446498cb-e44e-4eed-abb3-
547b0ee603ca"}

{
  "id": "1234567890",
  "name": "John Doe",
  "iat": 1516239022,
  "H(key)":
"a5a70bc36ecc3ec8259d2e28424792a34bc
bc3d2ea1858c70d01f0cf1308fd4a2"
}
```

Плюсы

1) При XSS злоумышленник не сможет воспользоваться украденным токеном
2) Токены будут отправляться в кастомных заголовках -> защитимся от CSRF
3) При компрометации поддомена и получении cookie key злоумышленник не получит доступ к другим доменам

Минусы

1) При компрометации поддомена и наличии приложения уязвимого к XSS атакам злоумышленник сможет украсть сессию пользователя

Сессионные токены в WEB приложении

1

Access/Refresh

Время жизни Access:

10 – 30 минут

Время жизни Refresh:

6 – 12 месяцев

2

Где будем хранить токены?

Refresh в cookie

Access в Local Storage & Cookie

Сессионные токены в WEB приложении

1

Access/Refresh

Время жизни Access:

10 – 30 минут

Время жизни Refresh:

6 – 12 месяцев

2

Где будем хранить токены?

Refresh в cookie

Access в Local Storage & Cookie

3

Stateless/stateful

Refresh – stateful

Access – stateless

Сессионные токены в WEB приложении

1

Access/Refresh

Время жизни Access:

10 – 30 минут

Время жизни Refresh:

6 – 12 месяцев

2

Где будем хранить токены?

Refresh в cookie

Access в Local Storage & Cookie

3

Stateless/stateful

Refresh – stateful

Access – stateless

4

Различные ключи для различных приложений

Если был скомпрометирован токен приложения A, то остальные приложения не пострадали.

Получив токен от приложения A, злоумышленник не сможет получить доступ к приложению B.

При компрометации ключа от приложения A, остальные приложения не пострадают

Отказ от паролей

Отказ от паролей

OTP длинной в 6 символов

Код должен быть одноразовым.

Код должен жить не более 5 минут.

У пользователя должно быть максимум 5 попыток ввода кода.

Отказ от паролей

OTP длинной в 6 символов

Код должен быть одноразовым.

Код должен жить не более 5 минут.

У пользователя должно быть максимум 5 попыток ввода кода.

Канал отправки кода

Push сообщение в собственное приложение или SMS с одноразовым кодом.

Имея собственное приложение можно использовать его и меньше тратиться на SMS сообщения.

Отказ от паролей

OTP длинной в 6 символов

Код должен быть одноразовым.

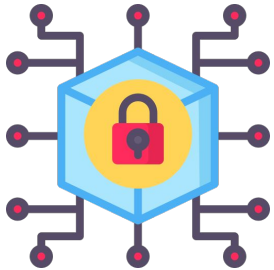
Код должен жить не более 5 минут.

У пользователя должно быть максимум 5 попыток ввода кода.

Канал отправки кода

Push сообщение в собственное приложение или SMS с одноразовым кодом.

Имея собственное приложение можно использовать его и меньше тратиться на SMS сообщения.



Криптостойкий алгоритм генерации OTP

Для генерации одноразового пароля должен использоваться криптостойкий алгоритм с высокоэнтропийными параметрами.

Отказ от паролей

OTP длинной в 6 символов

Код должен быть одноразовым.

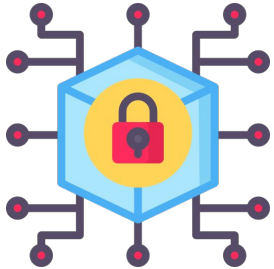
Код должен жить не более 5 минут.

У пользователя должно быть максимум 5 попыток ввода кода.

Канал отправки кода

Push сообщение в собственное приложение или SMS с одноразовым кодом.

Имея собственное приложение можно использовать его и меньше тратиться на SMS сообщения.



Криптостойкий алгоритм генерации OTP

Для генерации одноразового пароля должен использоваться криптостойкий алгоритм с высокоэнтропийными параметрами.

Одноразовые пароли не должны логироваться

Хорошей практикой является иметь отдельный более подробный лог для нужд ИБ.
Но даже в таком логе не должен сохраняться ОТП.

Лимиты

На какие действия накладывать лимиты?

1

Запрос ОТП

Возможность запросы одноразового кода должна быть ограничена во избежание:

- Излишних трат на отправку смс
- Спама пользователей
- Переборных атак

2

Подтверждение аутентификации

Для защиты от brute force атак необходимо ограничить пользователю количество попыток входа в приложение

На кого можем накладывать лимиты?

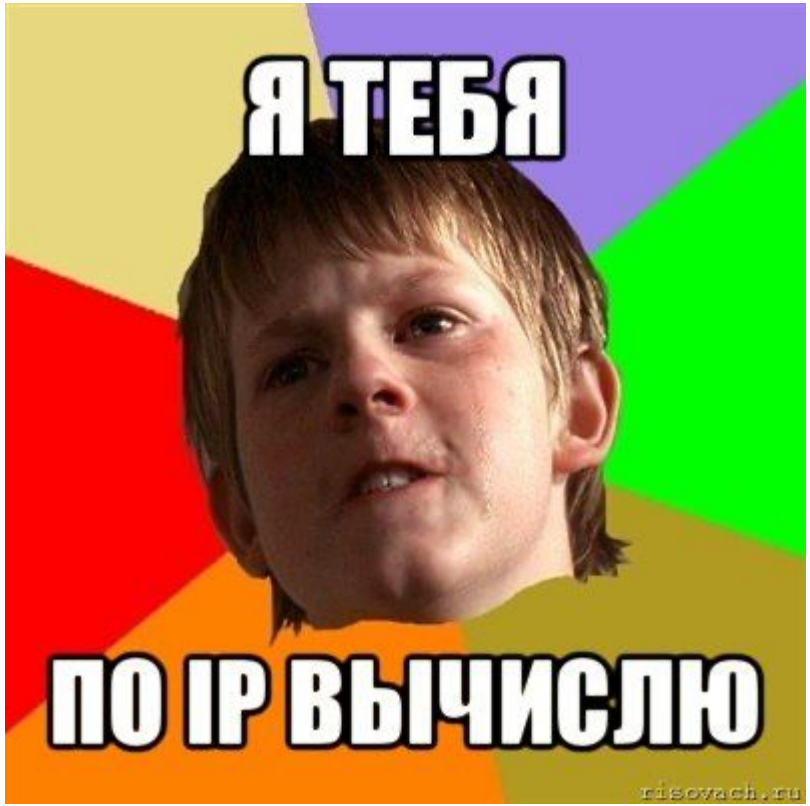
1. Пользователь

На кого можем накладывать лимиты?

1. ~~Пользователь~~

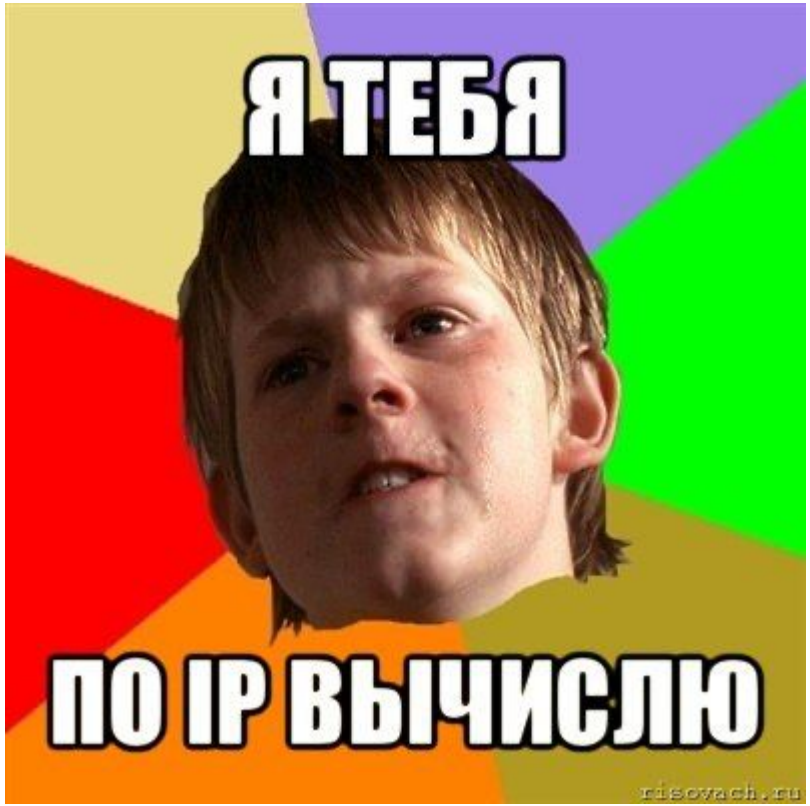
На кого можем накладывать лимиты?

1. ~~Пользователь~~
2. IP-адрес

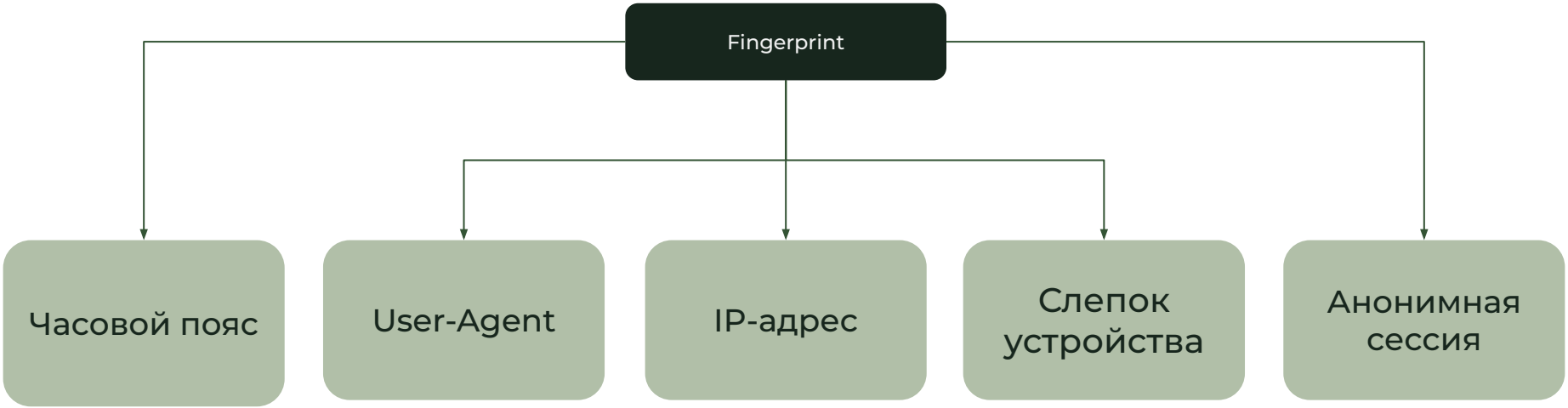


На кого можем накладывать лимиты?

1. ~~Пользователя~~
2. IP-адрес
3. Fingerprint

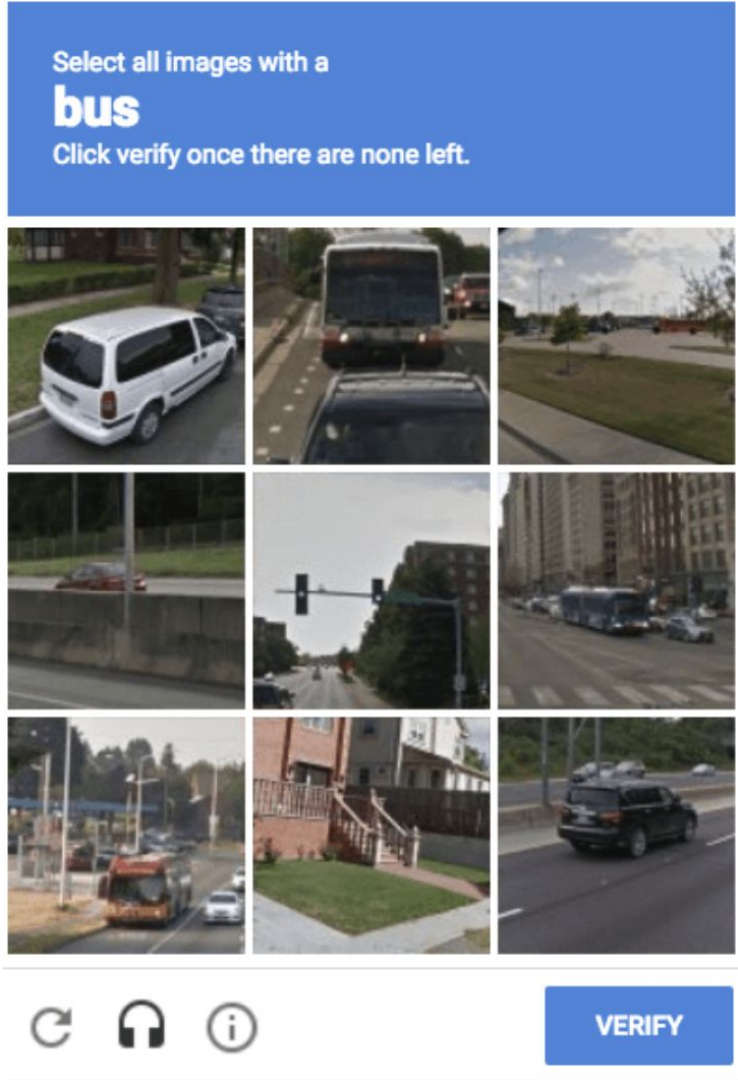


Отпечаток пользователя

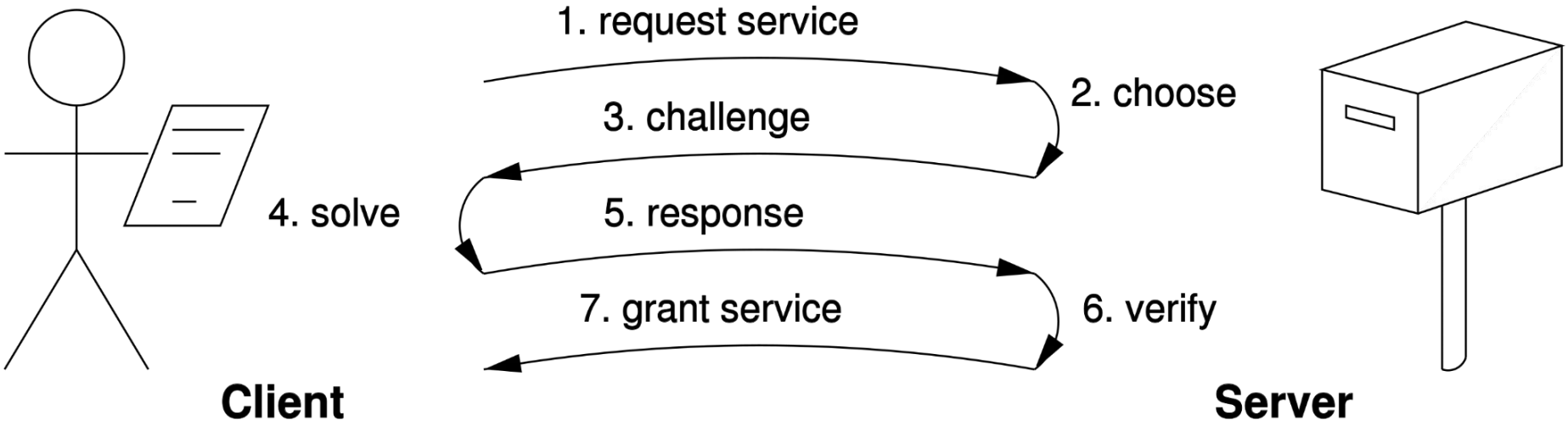


Captcha

Completely
Automated
Public
Turing test to tell
Computers and Humans
Apart



Proof of Work



Как накладывать лимиты

1. На критичные действия

Как накладывать лимиты

1. На критичные действия
2. Для отпечатка пользователя

Как накладывать лимиты

1. На критичные действия
2. Для отпечатка пользователя
3. Лимиты должны быть ступенчатыми

Как накладывать лимиты

1. На критичные действия
2. Для отпечатка пользователя
3. Лимиты должны быть ступенчатыми
4. Лимиты могут отличаться для Push и SMS

Как накладывать лимиты

1. На критичные действия
2. Для отпечатка пользователя
3. Лимиты должны быть ступенчатыми
4. Лимиты могут отличаться для Push и SMS
5. Проверять подозрительных пользователей капчей, а не банить

Детектирование подозрительных событий

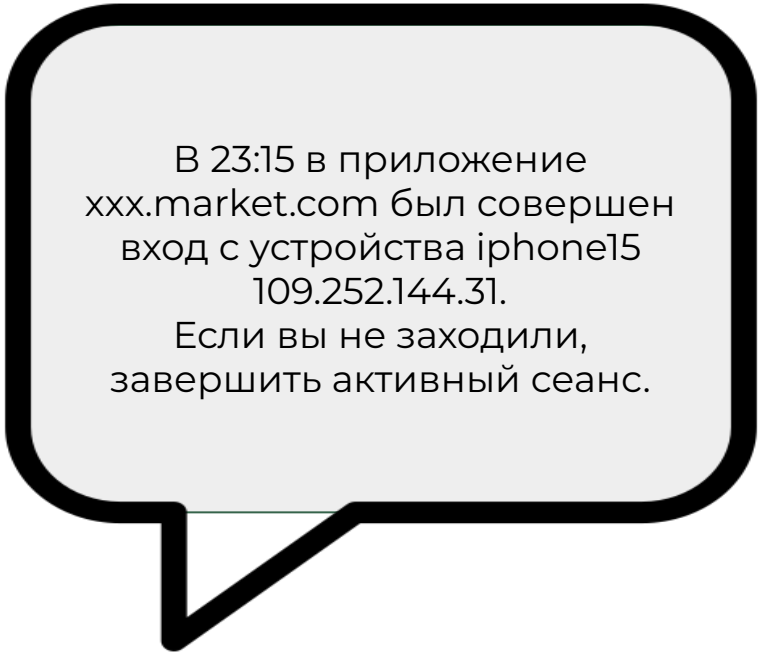
Детектирование подозрительных событий

- Вход с нового устройства
- Изменение геолокации
- Сильное изменение отпечатка пользователя
- Изменение региона во время аутентификации

Уведомления

Уведомление о входе в аккаунт или о подозрительной активности должно содержать:

- Название приложение
- Название устройства
- Время активности
- Ссылка или кнопка разлогина



Завершение сессии

Активные сеансы

Safari 17.1
Desktop Mac OS X
Текущий сеанс

Россия, Москва
109.252.144.31

Wildberries 643
iPhone15,2
12.03.2024 09:00:24

Россия, Москва
176.59.56.126

Chrome 120.0
Desktop Mac OS X
01.03.2024 20:51:38

Россия,
178.57.90.218

Завершить другие сеансы

Ещё

2FA

1

Пароль

Так как первым фактором у нас является OTP, то стандартный пароль может быть вторым фактором.

На пароль можно накладывать меньше ограничений, чем когда он единственный фактор.

энтропия важнее спец символов.

2FA

1

Пароль

Так как первым фактором у нас является OTP, то стандартный пароль может быть вторым фактором.

На пароль можно накладывать меньше ограничений, чем когда он единственный фактор.

энтропия важнее спец символов.

2

TOTP

Пользователи любят сохранять seed в сообщения в мессенджеры или в сторонние аутентификаторы

2FA

1

Пароль

Так как первым фактором у нас является OTP, то стандартный пароль может быть вторым фактором.

На пароль можно накладывать меньше ограничений, чем когда он единственный фактор.

энтропия важнее спец символов.

2

TOTP

Пользователи любят сохранять seed в сообщения в мессенджеры или в сторонние аутентификаторы

3

Email

Удобный, но не самый безопасный способ.

Почтовые аккаунты часто взламывают.

2FA

1

Пароль

Так как первым фактором у нас является OTP, то стандартный пароль может быть вторым фактором.

На пароль можно накладывать меньше ограничений, чем когда он единственный фактор.

энтропия важнее спец символов.

2

TOTP

Пользователи любят сохранять seed в сообщения в мессенджеры или в сторонние аутентификаторы.

3

Email

Удобный, но не самый безопасный способ.

Почтовые аккаунты часто взламывают.

4

Hardware Key

Дорогой, но эффективный способ. Можем предложить особенным пользователям.

Также можно реализовать поддержку webauthn для устройств с встроенным аппаратным ключом.

2FA не для login-а

Для критичных действий
Доступ к критичному функционалу можно предоставлять только после ввода второго фактора

Для проверки подозрительной активности
При подозрительной активности пользователя можно запросить второй фактор для подтверждения личности

Проведение операции заблокировано для продолжения работы подтвердите свою личность

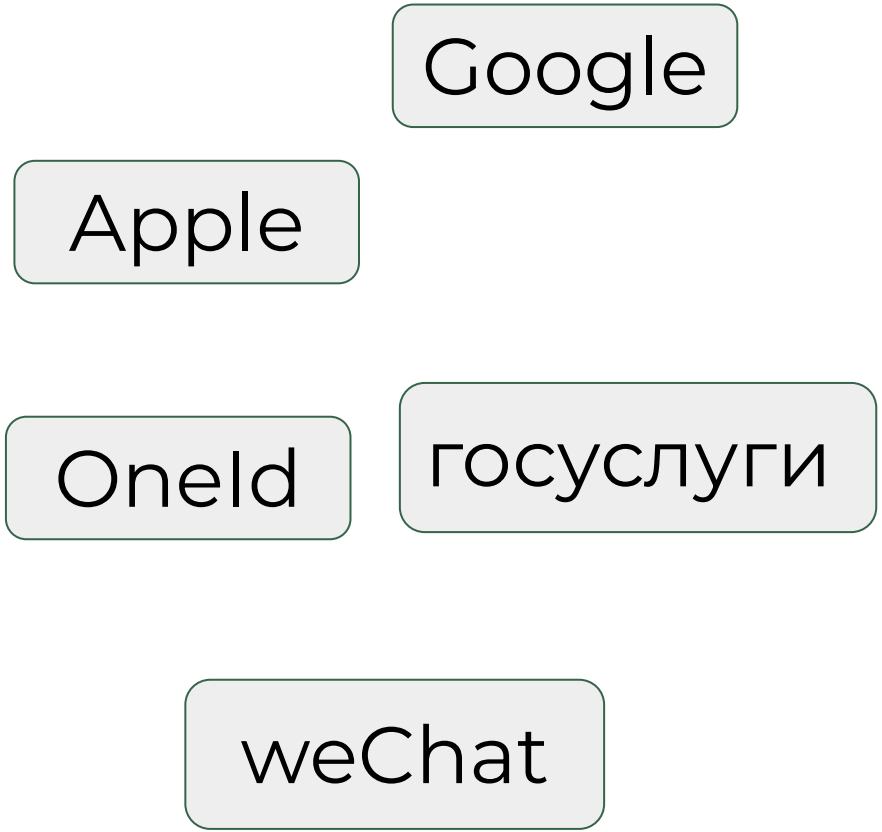
Вход через сторонних провайдеров

Вход через сторонних провайдеров

Несколько аккаунтов у одного пользователя
Размножается сущность пользователя, если не мержить аккаунты, и их становится тяжело контролировать

Тяжело отслеживать
Становится необходимо искать все аккаунты одного пользователя

Компрометация сторонних провайдеров
Взлом УЗ google приведет к получению доступа злоумышленником аккаунта в вашем приложении



Что можно сделать?

1

Принудительно связывать при первом входе

Хороший способ для ИБ и только для ИБ :)

Пользователь попытается пойти через стороннее приложение, а по итогу всё равно придется проходить стандартный flow

2

Связывать по запросу

Добавить функционал связывания аккаунтов.

Административный интерфейс

Административный интерфейс

Узкий круг доступа
В административный интерфейс такого сервиса должны иметь доступ только команда ИБ и руководитель разработки

Просмотр сессий
Должно быть возможно просматривать информацию о сессиях для расследования подозрительной активности

Завершение сессий
Необходимо уметь завершать сессии, для разрыва сеанса злоумышленника с аккаунтом пользователя

Логи

Logs

Логи важны
Отсутствие логов сделает невозможным расследования инциденты

Отдельный лог для ИБ
Лог с более подробной информацией

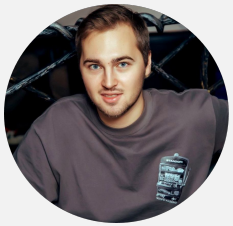
В логах не должно быть секретной информации

Итоги

Итоги

- Единый сервис
- Надежное хранилище токенов
- Отсутствие паролей
- Лимиты
- Детектирование подозрительных действий
- 2FA
- Сторонние провайдеры
- Административный интерфейс
- Логи

А теперь ваши вопросы



Александр Савин
InfraSec Lead

telegram
@Savin_A_S

linkedin
a.savin