TAU TECHNOLOGIES

Требования, предъявляемые к защищенности корпоративных мобильных приложений, и методы их реализации

Александр Епифанов

Спикер

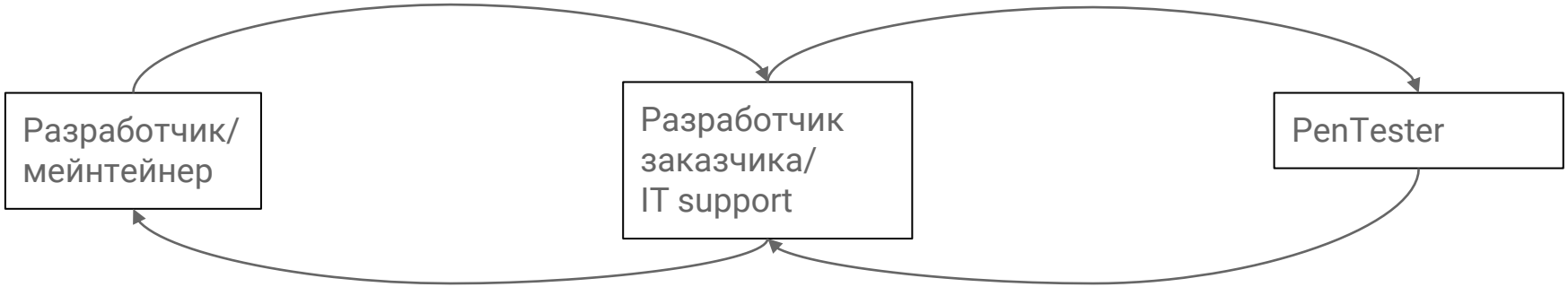
Александр Епифанов

- Соучредитель и исполнительный директор в “Tay Технологиях”.
- В разработке ПО более 20 лет, начинал с мобильных приложений для Palm OS.
- Писал софт в геймдеве, телекоме, для встраиваемых систем.
- В разработке кроссплатформенных инструментов с 2012 года.
- “Играющий тренер” в Tay с 2015 года.
- Активный контрибьютор OSS.
- Контакты тут: http://epifanoff.name/.

Оговорки и уточнения

- Мы разрабатываем и поддерживаем опенсорсный фреймворк для разработки приложений, оказываем платную поддержку для заказчиков с собственной разработкой.
- Также мы сами разрабатываем и сопровождаем мобильные приложения (на разных стеках).
- Речь о BYOD и COPE, MDM может быть опциональным, о доверенной загрузке не идёт речи, как правило.
- Речь, кроме прочего, о решениях с веб-фронтэндом.
- В основном речь про iOS и Android.
- Здесь будет дан обзор опыта работы с требованиями к безопасности с т.з разработчиков.
- В слайдах иногда используется английский язык.

Схема аудита



- Периодически (как правило, раз в год), заказчики проводят аудит безопасности своих мобильных приложений с помощью профильного подрядчика (Cylance, Veracode, Intel Security и пр.).
- Анализ проводится по методу черного ящика, либо применяется реверс инжиниринг.
- По результатам составляется отчет с выявленными проблемами и рекомендациями, который спускается к нам как к разработчикам приложения либо как к мейнтейнерам платформы.
- Нет прямой коммуникации между мейнтейнером и пен тестером.
- Есть проблемы с квалификацией и вовлеченностью специалистов заказчика.
- Бывают проблемы с квалификацией пентестеров, ведущие к формальным или нерелевантным пунктам в отчете.

Встроенные средства безопасности мобильных ОС

- Песочница.
- Безопасное хранилище для приложения.
- Шифрование пакета приложения средствами платформы (на iOS при публикации в AppStore).
- Цифровая подпись.
- Face/Touch ID.
- MDM.

На практике

Сеть

- HTTP Basic Authentication in Use:
  - CURL: `curl_easy_setopt(m_curl, CURLOPT_HTTPAUTH, CURLAUTH_DIGEST)`
  - Android: `HttpURLConnection + кастомная обвязка для Digest auth`
  - iOS: `NSURLConnectionDelegate.willSendRequestForAuthenticationChallenge`
    `NSURLAuthenticationChallenge.protectionSpace.authenticationMethod`
    `NSURLAuthenticationMethodHTTPDigest`

- SSL peer verification disabled (*по умолчанию - всегда включено):
  - Android: `WebViewClient.onReceivedSslError`
    `SslErrorHandler.proceed()`
    `кастомный X509TrustManager.checkServerTrusted()`

- SSL pinning not implemented:
  - Android: `network_security_config.xml ( API 24+ )`
  - iOS: `TrustKit: config[kTSKPinnedDomains][kTSKPublicKeyHashes]`

- SSL client auth:
  - Android: `кастомный X509TrustManager и KeyManagerFactory`

SSL peer verification disabled, иллюстрация из отчета

Реверсинг приложения, код проверки Web клиентом сертификата сервера.
В данном случае, вероятно, использовался apktool/baksmali.

RhoWebViewClient.smali - Notepad

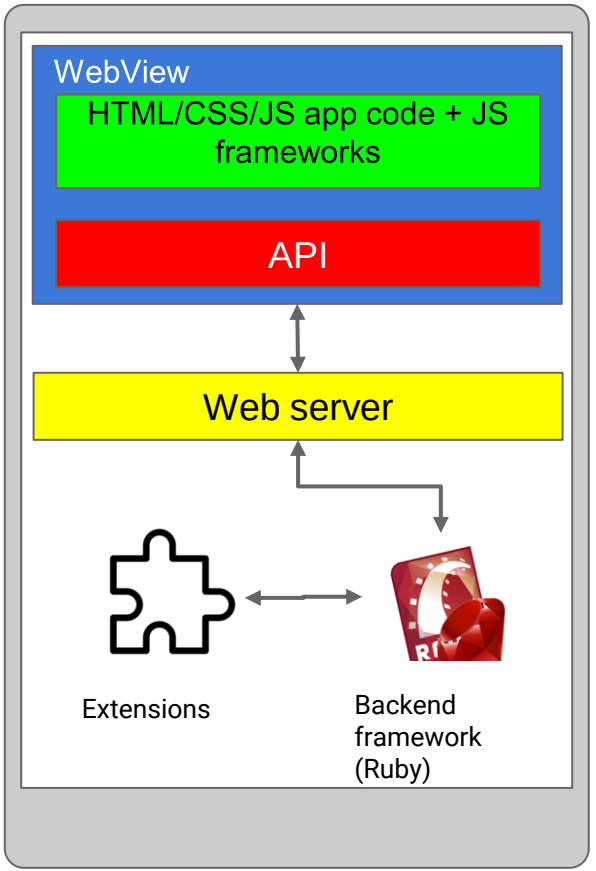
File Edit Format View Help

```
(Landroid/view/View;Lcom/rhomobile/rhodes/extmanager/IRhoExtension$IAuthRequest;)V

    goto :goto_0
.end method

.method public onReceivedSslError(Landroid/webkit/WebView;Landroid/webkit/SslErrorHandler;Landroid/net/http/SslError;)V
    .locals 3
    .param p1, "view"    # Landroid/webkit/WebView;
    .param p2, "handler"    # Landroid/webkit/SslErrorHandler;
    .param p3, "error"    # Landroid/net/http/SslError;

    .prologue
    .line 233
    const-string v1, "no_ssl_verify_peer"

    invoke-static {v1}, Lcom/rhomobile/rhodes/RhoConf;->getBool(Ljava/lang/String;)Z

    move-result v1

    if-eqz v1, :cond_0

    .line 234
    iget-object v1, p0, Lcom/rhomobile/rhodes/webview/RhoWebViewClient;->TAG:Ljava/lang/String;

    const-string v2, "Skip SSL error."

    invoke-static {v1, v2}, Lcom/rhomobile/rhodes/Logger;->D(Ljava/lang/String;Ljava/lang/String;)V

    .line 235
    invoke-virtual {p2}, Landroid/webkit/SslErrorHandler;->proceed()V
```

Особенности гибридной архитектуры

- При старте приложение запускает HTTP(S) сервер на локалхосте.
- UI работает в стандартном WebView компоненте.
- Весь веб-контент приложения находится внутри пакета (APK, IPA …), т.е. не происходит взаимодействия с внешними ресурсами.
- За веб-сервером работает машина Ruby.
- Код на Ruby (в нашем случае) при сборке компилируется в ISEQ. Это можно назвать обфускацией т.к. усложняет реверсинг (тем более что по iSEQ нет официальной документации и инструментария).

```
127.0.0.1:12345
-> GET /app/MyBusinessEntitity/list?some&more&args
 socket(-ish) -> web server -> Ruby code [ -> native code ]
<- <html><body>...
```



WebView в гибридном приложении

- setJavaScriptEnabled Web View RCE Vulnerability CVE-2013-4710:
  Запрос разработчика заказчика:
  *“We have a reported security issue under Web View RCE Vulnerability CVE-2013-4710. The security team suggested to **setJavaScriptEnabled** to **false** in order to mitigate this vulnerability. We need to know where to add this flag and what is the affection of such change **knowing that we are using javascript in our app**.”*

- Exposed Dangerous Method or Function (CWE ID 749)(1 flaw) addJSInterface:
  Выдержка из отчета:
  “Description
  The application provides an API or similar interface to a dangerous method or function that is not properly restricted.”

WebView в гибридном приложении

- Protection Mechanism Failure (CWE ID 693):

  Разработчик:

  “Basically its asking to implement safebrowsing in android webview.
  https://developer.android.com/reference/android/webkit/WebSettings#setSafeBrowsingEnabled(boolean)
  ”

  Документация Android

  Sets whether Safe Browsing is enabled. Safe Browsing allows WebView to protect against malware and phishing attacks ***by verifying the links****.*

Такие “уязвимости”, очевидно, связаны с архитектурой платформы и не подлежат устранению, что мы регулярно объясняем разработчикам и поддержке заказчиков.

Локальный (на 127.0.0.1) сервер в гибридном приложении

Основная проблема:
к слушающему сокету может подключиться внешний процесс, что потенциально нарушает песочницу: нужен способ убедиться что клиент “наш”, т.е. находится в том же процессе что и сервер.

Решения:

- Рандомный порт.
- Полный перехват веб запросов - не всегда возможно, могут быть ограничения по протоколу.
- Генерация токена доступа на уровне HTTP.
- Двусторонняя TLS аутентификация с использованием автоматически генерируемых самоподписанных сертификатов.

Локальные данные

- Шифрование БД, встроенные средства SQLite (AES256):
  Ключ хранится в защищенном хранилище устройства.
- Шифрование бандла (данные + бизнес код) при сборке AES256:
  Ключ вшит в приложение.
- Запрет на снятие скриншотов в приложении:
  - Android:

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_SECURE, WindowManager.LayoutParams.FLAG_SECURE);
```

  - iOS:

```
- (void)applicationWillResignActive:(UIApplication *)application {
            [self blurCurrentView];
 }
```

Локальные данные

- Очистка клипборда при деактивации приложения:
  - Android:

```
ClipboardManager clipMan =    (ClipboardManager)getContext().getSystemService(Context.CLIPBOARD_SERVICE);
ClipData clip = ClipData.newPlainText("", "");
clipMan.setPrimaryClip(clip);
```

  - iOS:

```
UIPasteboard *pasteboard = [UIPasteboard generalPasteboard];
[pasteboard setValue:@"" forPasteboardType:UIPasteboardNameGeneral];
```

- Авто логаут по home:
  Отказались по причине ужасного UX.
- Пароли и логины в памяти:
  Храним только между вводом пользователя и отправкой на сервер, обфусцируем (xor, base64)

Защита от реверсинга / безопасность кода

- ISeq (Instructions sequence):
  Байт-код для Ruby машины - обфускация+ускорение.
- Proguard:
  Обфускация Java. Сложно применять там где используется рефлексия или JNI.
- YUI compressor:
  Обфускация и минификация JS/CSS.
- Шифрование бандла:
  Дополнительно шифруются iseq и js.

Защита от реверсинга на рантайме

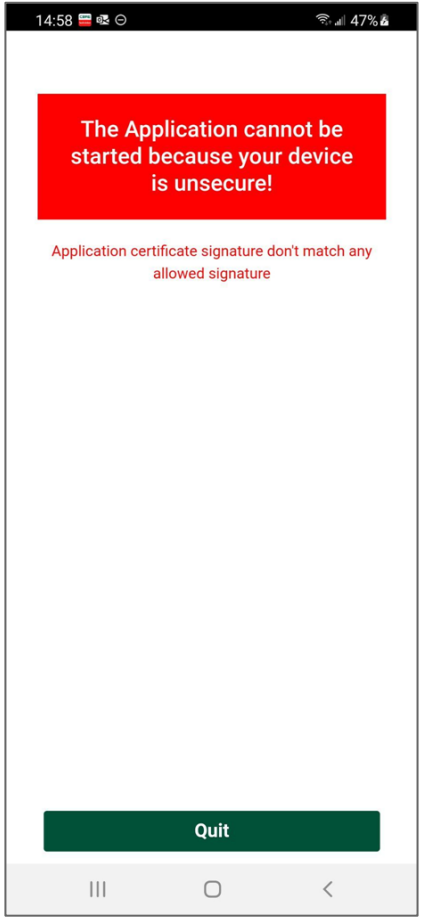
- Обнаружение root на Android.
- Обнаружение jailbreak на iOS.
- Проверка подписи пакета.

```
android:
  security:
    allowed_cert_signatures:
    - "4o7xYWLVqbE+lK020bKX0+wnM48="
```

- Проверка пакета-установщика по имени.

```
android:
  security:
    allowed_installers:
      - 'com.android.vending'  #идентификатор установщика Android
```

- Детектирование отладчика.
- Детектирование запуска на эмуляторе.



Немного кода

```
        private static boolean checkRootMethod1() {
            String buildTags = android.os.Build.TAGS;
            return buildTags != null && buildTags.contains("test-keys");
        }

        private static boolean checkRootMethod2() {
            String[] paths = { "/system/app/Superuser.apk", "/sbin/su", "/system/bin/su",
"/system/xbin/su", "/data/local/xbin/su", "/data/local/bin/su", "/system/sd/xbin/su",
                    "/system/bin/failsafe/su", "/data/local/su", "/su/bin/su"};
            for (String path : paths) {
                if (new File(path).exists()) return true;
            }
            return false;
        }
```

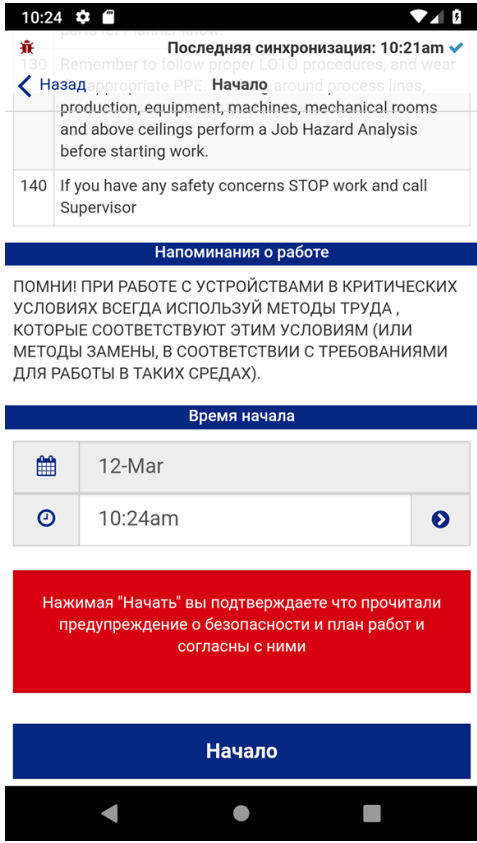
Еще немного кода

```
private static boolean checkEmulatorMethod2() {
     return Build.FINGERPRINT.startsWith("generic")
          || Build.FINGERPRINT.startsWith("unknown")
          || Build.MODEL.contains("google_sdk")
          || Build.MODEL.contains("Emulator")
          || Build.MODEL.contains("Android SDK built for x86")
          || Build.MANUFACTURER.contains("Genymotion")
          || (Build.BRAND.startsWith("generic") && Build.DEVICE.startsWith("generic"))
          || "google_sdk".equals(Build.PRODUCT)
          || "goldfish".equals(Build.HARDWARE);
 }
```

Недобросовестные действия пользователя

Кейз: приложение для техников:

- Техник обрабатывает поступающие заявки с помощью мобильного приложения.
- Оплата техника зависит от времени потраченного на заявку.
- При старте и завершении заявки приложение фиксирует время (и, следовательно, продолжительность работы).
- Системным часам нельзя доверять: техник может подкрутить их и добавить время к заявке = убыток для заказчика.
- Приложение может работать офлайн, без соединения с сервером.



Недобросовестные действия пользователя

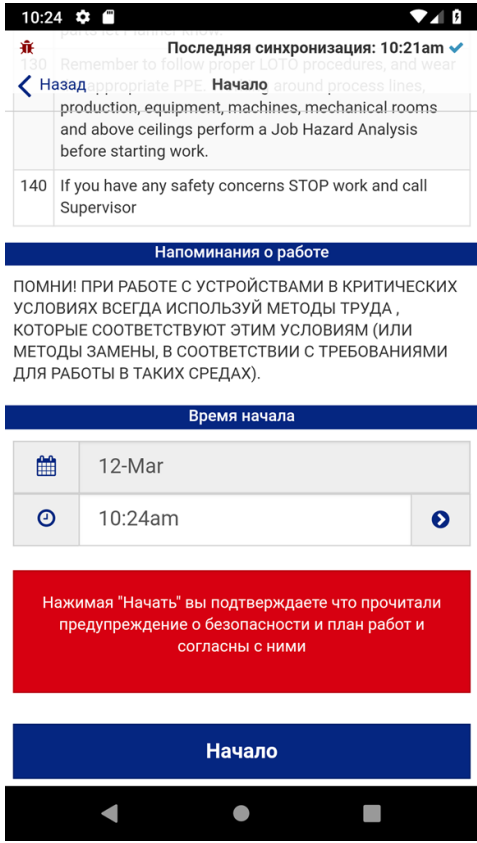
Решение:

- При запуске приложения требуем логиниться (т.е. храним сессию только пока приложение запущено).
- При логине сервер присылает настоящее время.
- Привязываем системные тики (миллисекунды от загрузки устройсва) к полученному серверному времени.
- При фиксировании времени работы учитываем только системные тики, не ориентируясь на часы.

Android: `android.os.SystemClock.elapsedRealtime()`

iOS:

```
static int64_t us_since_boot() {
            struct timeval boottime;
            int mib[2] = {CTL_KERN, KERN_BOOTTIME};
            size_t size = sizeof(boottime);
            int rc = sysctl(mib, 2, &boottime, &size, NULL, 0);
            if (rc != 0)  return 0;
            return boottime.tv_sec * 1000000 + boottime.tv_usec;
}
```



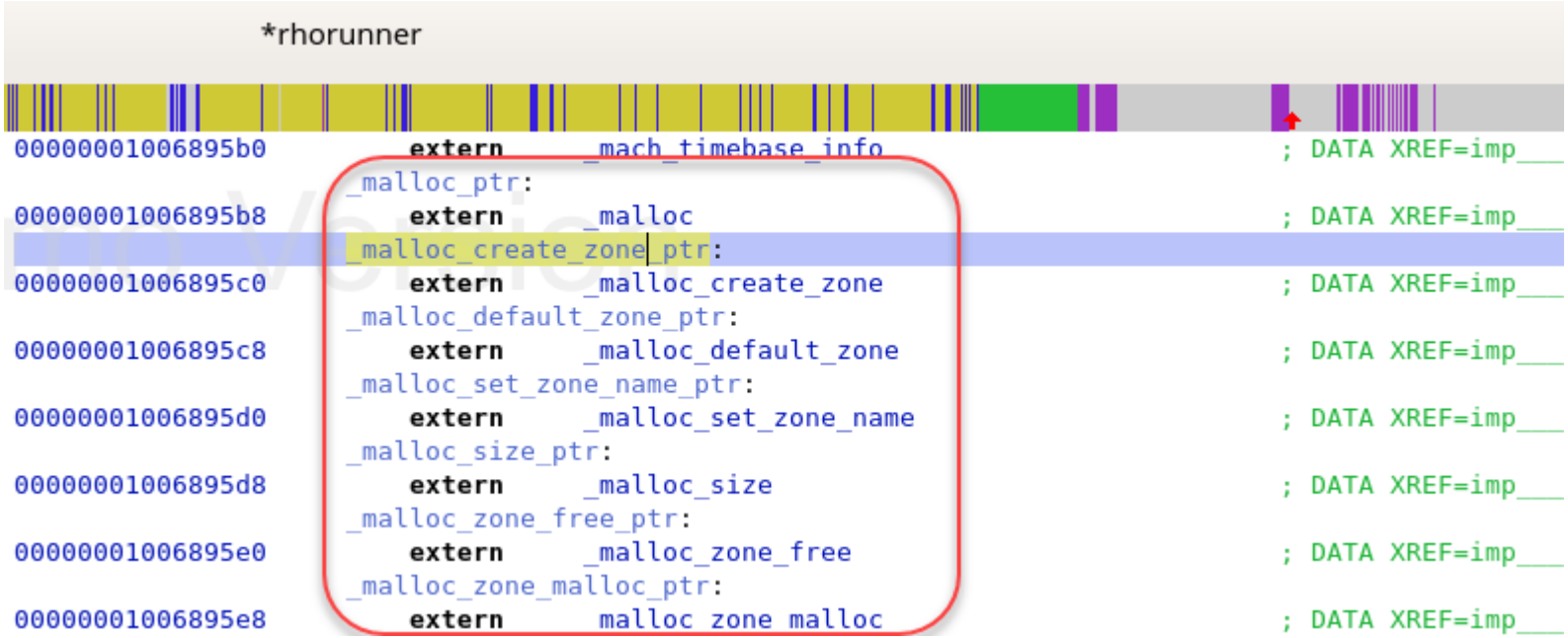
Странное…

- Заменить вызовы libc в сторонних зависимостях:
  например 2500 вызово malloc на calloc в сторонних зависимостях типа CURL и SQLite.
- ProGuard не покрывает 100% Java кода (в опенсорс платформе!).
- Литералы типа “password”, “http”, найденные внутри бинарного файла репортятся как “хранение захардкоженного пароля в открытом виде” или “использование незащищенного соединения”.

На такие пункты отвечаем о сложности и нецелесообразности реализации, с просьбой предоставить PoC реальной уязвимости. На этом, как правило, всё заканчивается.

Просят заменить 2500 вызовов malloc на calloc

IDA Pro?



Выводы

- Периодические аудиты приложений дают полезную обратную связь и помогают развивать платформу с т.з безопасности и поддерживать высокий уровень защищенности.
- Данные ценнее кода.
- Не все требования ИБ выполнимы или целесообразны, но такие как правило носят характер low-level рекомедаций.
- Реализованные обновления и дополнительные механизмы защиты становятся автоматически доступны для всех приложений при обновлении платформы.
- Взломать можно всё.
  - особенно, если злоумышленник имеет физический доступ к мобильному устройству.

“I have to say, I am impressed with the technology your company has created. RhoMobile is cool.”

Aaron Bryson
Director of Information Security @ Blue Lava, Inc
Technical Director @ Cylance (Blackberry Cybersecurity)

Ссылки

RhoMobile / Tau Платформа:
https://tau-platform.com/ru/developers/downloads/
https://tau-platform.com/ru/products/rhomobile/
https://tau-platform.com/ru/products/rhoconnect/

Репозитории:
https://github.com/rhomobile/rhodes
https://github.com/rhomobile/rhoconnect

RhoBrowser / Tau Браузер:
https://tau-platform.com/ru/products/rhobrowser/
https://rhobrowser-demo-static.s3.us-east-1.amazonaws.com/rhobrowser_demo_signed_master_JDK1.8.apk

Документация:
http://docs.tau-platform.com/en/7.4/home
http://docs.tau-platform.com/en/7.4/guide/apisummary
http://docs.tau-platform.com/en/7.4/guide/welcome

Дополнительные примеры приложений :
https://github.com/tauplatform/rhodes-system-api-samples
https://github.com/tauplatform/kitchensinkRuby
https://github.com/tauplatform/inventoryDemo-mobileApp
https://github.com/tauplatform/kitchensinkJS
https://github.com/tauplatform/universal-push-example

Контакты:

info@tau-platform.com