IOS SECURITY 101-ISH

VADIM DROBININ | @VALZEVUL

ABOUT ME

https://github.com/valzevul

Overview | Repositories 37 | Projects 0 | Stars 20 | Followers 67 | Following 4

Pinned repositories

apple/swift
The Swift Programming Language
C++ 46.9k 7.4k

daltoniam/Starscream
Websockets in swift for iOS and OSX
Swift 4.7k 580

In App Purchases Sw
nd macOS 10.10+
3.8k 378

Swift 1.4k 303

Swift 177 85

Apple WWDC Scholarship winner, public speaker and freelance iOS developer.

London
Sign in to view email
http://drobinin.com/

Organizations

1,815 contributions in the last year

2019
2018
2017
2016

Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar
Mon
Wed
Fri

Learn how we count contributions.
Less More

1. WHY?

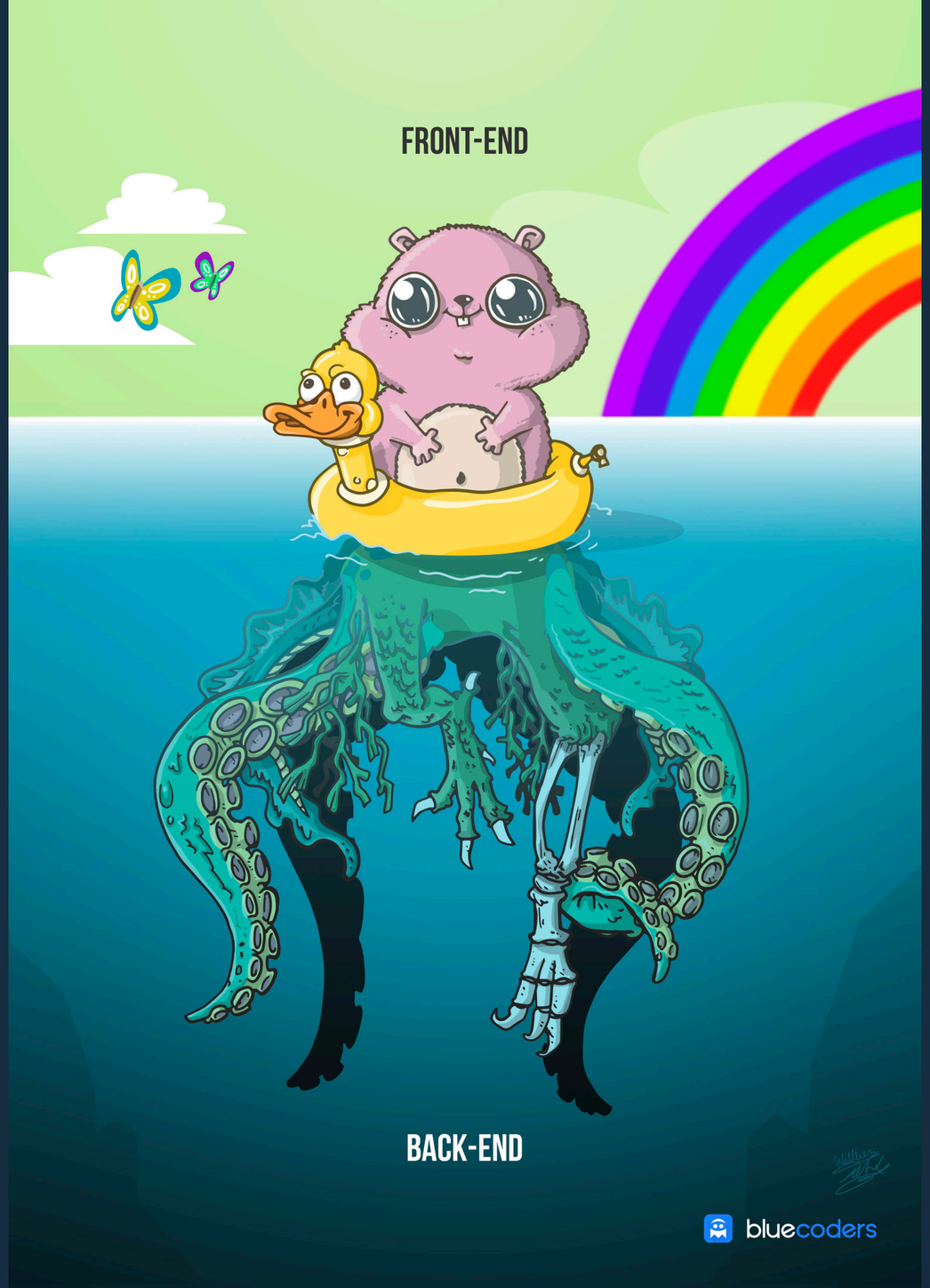
“THE AVERAGE TIME SPEND ON SMARTPHONES AND TABLETS IS 4H 33 MINS A DAY”

BankMyCell

“IN 2018, 52.2% OF ALL WEBSITE TRAFFIC WORLDWIDE WAS GENERATED THROUGH MOBILE PHONES.”

Statista

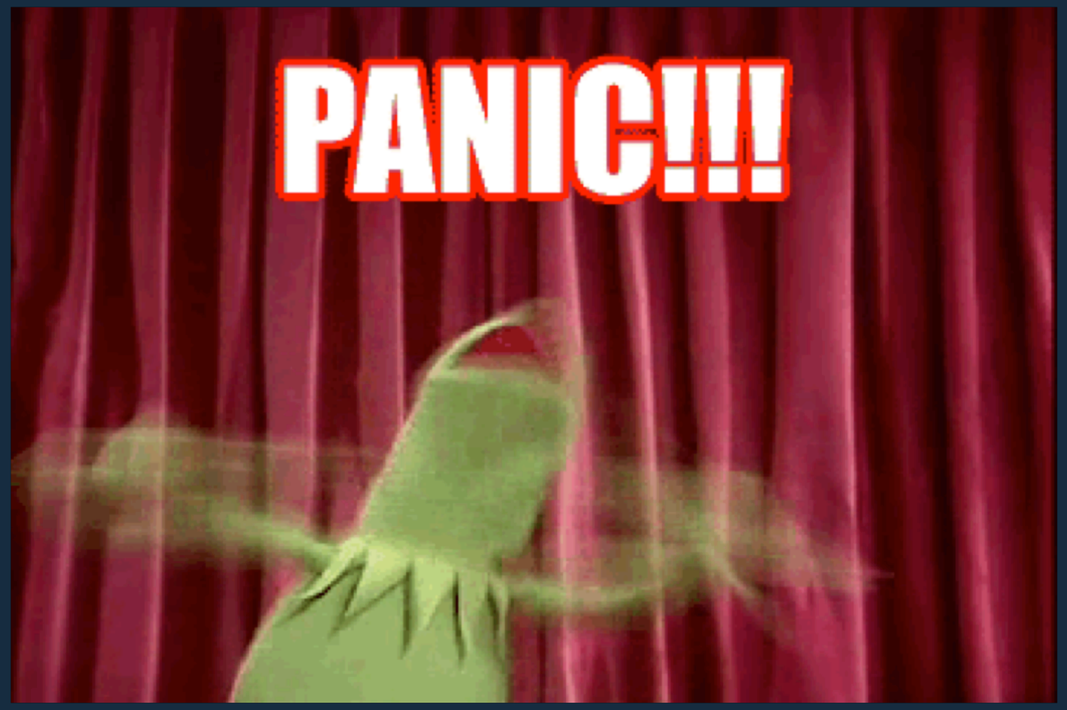
NEVER TRUST FRONTENDS



2. WHAT?

WHAT'S NOT SAFE?

- Usernames and passwords
- Location data
- Facial data
- Advertising data
- Address book entries
- Payment information
- Other personal information



OWASP*

* The Open Web Application Security Project, https://owasp.org/

ESSENTIAL PARTS

- » Device
  - » Local storage
  - » Interaction with the mobile platform
- » APIs
  - » Communication with trusted endpoints
  - » Authentication and Authorisation
- » Prevention
  - » Anti-Reversing

PLATFORM OVERVIEW

» iOS is based on Darwin, which kernel is XNU ("X is Not Unix")

» Sideload via Xcode is possible since iOS 9

» Secure boot, hardware-backed Keychain, file system encryption, update rollouts

» iOS apps are isolated from each other via Apple's iOS sandbox (“Seatbelt”)

“SEATBELT”

- OSX 10.5 “Leopard”, 2007
- Not mandatory
- Not many developers did this
- OSX 10.7 “Lion”, 2011
  - com.apple.security.app-sandbox entitlement
  - Added automatically when signed via App Store
- iOS:
  - /var/mobile/Containers and /var/Containers

SETTING UP A TESTING ENVIRONMENT

» Frida
https://www.frida.re

» Objection
https://github.com/sensepost/objection

» Wireshark
https://www.wireshark.org/download.html

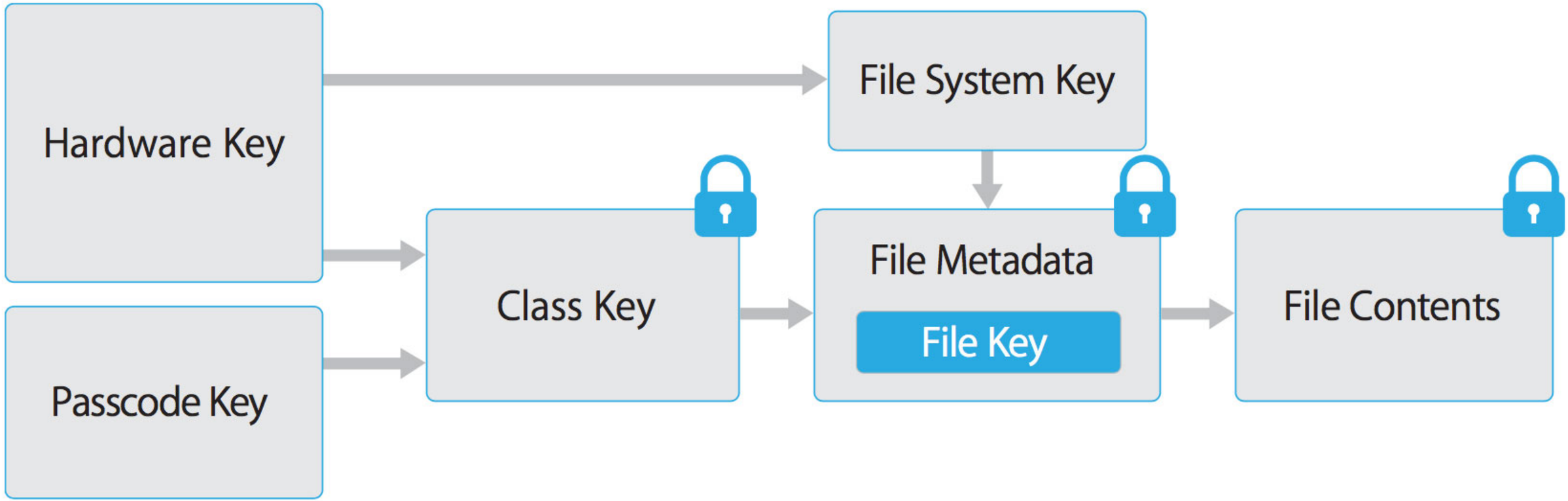
» Keychain-dumper
https://github.com/ptoomey3/Keychain-Dumper/

» Needle
https://github.com/mwrlabs/needle

AS LITTLE SENSITIVE DATA AS POSSIBLE SHOULD BE SAVED IN PERMANENT LOCAL STORAGE.

DATA PROTECTION API

DATA STORAGE ON IOS



PROTECTION CLASSES:

- » Complete Protection (NSFileProtectionComplete)
- » Protected Unless Open (NSFileProtectionCompleteUnlessOpen)
- » Protected Until First User Authentication (NSFileProtectionCompleteUntilFirstUserAuthentication)
- » No Protection (NSFileProtectionNone)

THE KEYCHAIN

» Only one Keychain is available to all apps

» Access control among apps via kSecAttrAccessGroup

» Access for items:

kSecAttrAccessibleAlways
kSecAttrAccessibleAlwaysThisDeviceOnly
kSecAttrAccessibleAfterFirstUnlock
kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly
kSecAttrAccessibleWhenUnlocked
kSecAttrAccessibleWhenUnlockedThisDeviceOnly
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly

KEYCHAIN ACCESS CONTROL FLAGS

```
kSecAccessControlDevicePasscode
kSecAccessControlTouch IDAny
kSecAccessControlTouch IDCurrentSet
kSecAccessControlUserPresence
```

HOW TO WORK WITH THE KEYCHAIN

```
func devicePasscodeEnabled() -> Bool {
    return LAContext().canEvaluatePolicy(.deviceOwnerAuthentication,
                                         error: nil)
}

let userDefaults = UserDefaults.standard
if userDefaults.bool(forKey: "hasRunBefore") == false {
     // Remove Keychain items here
     userDefaults.set(true, forKey: "hasRunBefore")
     userDefaults.synchronize() // Forces the app to update UserDefaults
}

func logout() {
    // Logout the user here
    wipeKeychain()
}
```

WHAT MIGHT GO WRONG?

» Make sure nothing sensitive (password, keys, tokens, other PII, etc) is stored in NSUserDefaults or via NSData, writeToFile, NSFileManager, CoreData, databases, etc without encryption.

» If the encryption is used, make sure the secret key is stored in the Keychain with secure settings,
ideally […]WhenPasscodeSetThisDeviceOnly.

BE CAREFUL WITH FIREBASE

- » 47% of iOS apps that connect to a Firebase database are vulnerable[1]
- » Get PROJECT_ID from GoogleService-Info.plist
- » Check https://<firebaseProjectName>.firebaseio.com/.json
- » Firebase Scanner https://github.com/shivsahni/FireBaseScanner

[1] Appthority Mobile Threat Team, Jan 2018

BE CAREFUL WITH REALM

```
// Open the encrypted Realm file where getKey()
// is a method to obtain a key from the Keychain or a server
let config = Realm.Configuration(encryptionKey: getKey())
do {
  let realm = try Realm(configuration: config)
  // Use the Realm as normal
} catch let error as NSError {
  // If the encryption key is wrong,
  // `error` will say that it's an invalid database
  fatalError("Error opening realm: \(error)")
}
```

DYNAMIC ANALYSIS VIA IMAZING

» Trigger the functionality that stores potentially sensitive data.

» Connect the iOS device and launch iMazing.

» Select the app and do "Extract App"

» Navigate to the output directory and locate $APPNAME.imazing. Rename it $APPNAME.zip.

» Unpack the zip file.

» To get Keychain items on a non-JB device, use objection

OTHER LOCATIONS OF SENSITIVE DATA

» Keyboard cache

```
textObject.autocorrectionType = .no
textObject.secureTextEntry = true
```

» Logs

» Backups

» Auto-generated (overlay) screenshots

» Memory

LOCAL AUTHENTICATION ON IOS

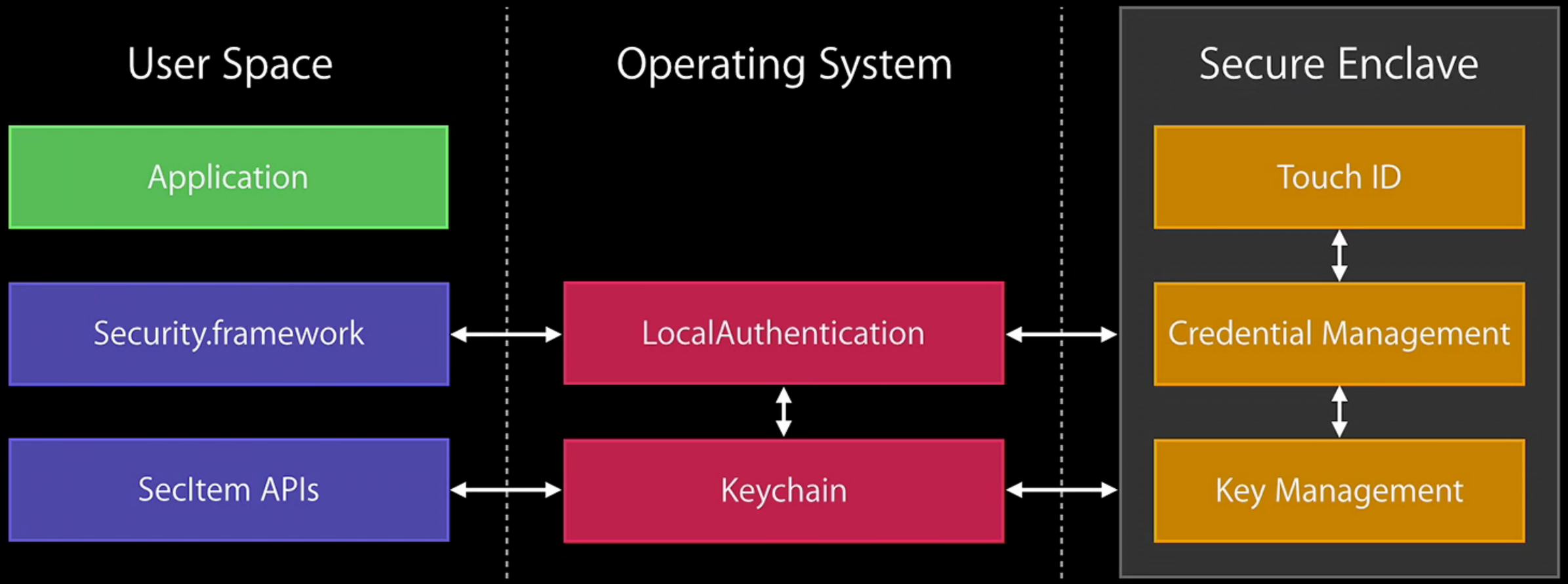
During local authentication, an app authenticates the user against credentials stored locally on the device.

» LocalAuthentication.framework
high-level API for TouchID/FaceID,

» Security.framework
low-level API for Keychain Services

IT'S SECURE, RIGHT?

IT'S SECURE, RIGHT?
NOPE.


User Space
Operating System
Secure Enclave
Application
Security.framework
SecItem APIs
LocalAuthentication
Keychain
Touch ID
Credential Management
Key Management

LOCAL AUTHENTICATION

» deviceOwnerAuthentication

» deviceOwnerAuthenticationWithBiometrics

```
LAContext().evaluatePolicy(.deviceOwnerAuthentication, localizedReason: "…") {
    success, evaluationError in
    if success {
        // Now you can trust the user
    }
}
```

» See Don't touch me that way[2] for a bypassing auth demo

[2] https://www.youtube.com/watch?v=XhXIHVGCFFM by David Lidner et al

IOS NETWORK API

App Transport Security (ATS):

» NSURLConnection, NSURLSession and CFURL

» Public hostnames (not IP addresses, unqualified domain names or TLD of .local)

» No HTTP connections

» Transport Layer Security (TLS) version >=1.2.

» Some more requirements to keys exchange

HOW TO PROTECT?

» ATS should be configured according to best practices by Apple and only be deactivated under certain circumstances.

» Don’t forget about SSL pinning; never hardcode the password though.

HOW TO PROTECT?

» If the application opens third party web sites in web views, NSAllowsArbitraryLoadsInWebContent can be used to disable ATS restrictions for the content loaded in web views.

» If the app connects to a defined number of domains under your control, configure the servers to support the ATS requirements and opt-in for the ATS requirements within the app.

HOW TO PROTECT

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
    <key>NSExceptionDomains</key>
    <dict>
        <key>example.com</key>
        <dict>
            <key>NSIncludesSubdomains</key>
            <true/>
            <key>NSExceptionMinimumTLSVersion</key>
            <string>TLSv1.2</string>
            <key>NSExceptionAllowsInsecureHTTPLoads</key>
            <false/>
            <key>NSExceptionRequiresForwardSecrecy</key>
            <true/>
        </dict>
    </dict>
</dict>
```

iOS Platform APIs

- All apps run under non-privileged mobile user
- Each app has a unique home directory and is sandboxed
- Access to protected resources or data (capabilities) is possible, but it's strictly controlled via special permissions (entitlements).

DON'T ASK FOR MORE PERMISSIONS THAN YOU ACTUALLY NEED AT THAT VERY MOMENT.

WHAT MIGHT GO WRONG?[3]

- Camera access
  - record users at any time the app is in the foreground
  - run real-time face recognition to detect facial features or expressions
  - upload the pictures/videos it takes immediately
- Photos
  - Track all users' movements based on their photos' meta
  - Track all their devices
  - Use facial recognition to find out who the user hangs out with

[3] Felix Krause, https://krausefx.com/privacy

WHAT MIGHT GO WRONG?

» MitM-attack to change the 3d-party framework

» Fake iCloud password alerts

» Inject anything into web views (if the app doesn't use SFSafariViewController)

» Screenshot typing password in app's secured fields

INTER PROCESS COMMUNICATION

» Universal Links

» Custom URL Schemes

» UIActivity Sharing

» App Extensions

» UIPasteboard

UNIVERSAL LINKS

- tg://resolve?domain=valzevul is a custom URL scheme and uses the tg:// scheme.
- https://telegram.me/valzevul is a universal link and uses the https:// scheme.
- Unique
- Secure
- Flexible
- Private

WHAT TO TEST

» Check the Associated Domains entitlement

» Retrieve the Apple App Site Association file

» Check the link receiver method

» Check the data handler method

» Check if the app is calling other app's universal links

“...DO NOT ALLOW UNIVERSAL LINKS TO DIRECTLY DELETE CONTENT OR ACCESS SENSITIVE INFORMATION ABOUT THE USER.”

Apple Documentation

UIPASTEBOARD

» Users cannot grant or deny permission for apps to read the pasteboard.

» Apple warns about persistent named pasteboards and discourages their use. Instead, shared containers should be used.

» Universal Clipboard is enabled by default and allows the general pasteboard contents to automatically transfer between devices.

CUSTOM URL SCHEMES

> "If more than one third-party app registers to handle the same URL scheme, there is currently no process for determining which app will be given that scheme."
>
> Apple Documentation

- canOpenURL will always return false for undeclared schemes
- though openURL will still open it even if LSApplicationQueriesSchemes is set
- List of URL scheme names[4]

[4] https://ios.gadgethacks.com/news/always-updated-list-ios-app-url-scheme-names-0184033/

WEB VIEWS

- » UIWebView (deprecated + impossible to turn off JS)
- » SFSafariViewController (impossible to turn off JS)
- » WKWebView:
  - » use javaScriptEnabled = false
  - » use hasOnlySecureContent = true
  - » out-of-process rendering → no memory corruption bugs

WEB VIEWS

» Topic for a separate talk.

» Native methods could be exposed through web views.

» Custom web views could ~~steal~~ store passwords, sessions, keys, etc.

» AutoFill data is available only for SFSafariViewController.

» Try WhereIsMyBrowser[5], an intentionally insecure app for training.

[5] https://github.com/authenticationfailure/WheresMyBrowser.iOS

iOS Anti-Reversing Defenses

» Jailbreak detection

» Anti-debugging checks

» File-integrity checks (source code and storage)

» Device binding

JAILBREAK DETECTION

» File-based checks

» File-permissions checks

» Protocol handlers (eg cydia://)

» Calling System APIs

DEVICE BINDING

» ❌ MAC addresses, UDID, unsafe bindings

» ✅ UIDevice.current.identifierForVendor

» ✅ Keychain + kSecAttrAccessibleWhenUnlockedThisDeviceOnly

» ✅ Google and its Instance ID for iOS

CARING ABOUT USERS

- Informing users on their private information:
  - The right to be forgotten
  - The right to correct data
  - The right to access user data
- OSS information
- Apple’s best practices (Accessibility, Localization, etc)

TESTING

» Preparation

» Intelligence Gathering

» Mapping the Application

» Exploitation

» Reporting

“TRUE EXCELLENCE AT MOBILE APPLICATION SECURITY REQUIRES A DEEP UNDERSTANDING OF MOBILE OPERATING SYSTEMS, CODING, NETWORK SECURITY, CRYPTOGRAPHY, AND A WHOLE LOT OF OTHER THINGS. ”

OWASP

NOT ENOUGH?

» iOS Security Guide by Apple[6]
Apple updates it for every version of iOS (as of now, 12.3 in May 2019)

» https://github.com/OWASP/owasp-mstg
OWASP Mobile Security Testing Guide

» Charlie Miller et al (2012) iOS Hacker's Handbook[7]

» David Thiel (2016) iOS Application Security, The Definitive Guide for Hackers and Developers[8]

» Apple Pay: Delve into the details[9]

[6] https://www.apple.com/business/site/docs/iOSSecurityGuide.pdf
[7] http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118204123.html
[8] https://www.nostarch.com/iossecurity
[9] https://drobinin.com/talks/2017/apple-pay-delve-into-the-details/

“DON'T STOP AT SECURITY TESTING. WRITE YOUR OWN APPS, COMPILE YOUR OWN KERNELS, DISSECT MOBILE MALWARE, LEARN HOW THINGS TICK.”

OWASP

QUESTIONS?

DROBININ.COM | @VALZEVUL