Demystifying Security Best Practices

Conrad Stoll

@conradstoll - conradstoll.com

Architect mutualmobile

UIKonf

So you want to make software secure?

Well, that’s easy.

Mobile devices are not secure.

We are expected to secure our software.

UnionBank®

Greenway

Cigna

ISIS™

Products are judged by the user experience, and the absence of security issues.

Security is hard.

goto fail;

Apple actually takes security very seriously.

iOS Security

February 2014

http://images.apple.com/ipad/business/docs/iOS_Security_Feb14.pdf

Security is about being responsible with a user's information.

The type of information your app handles defines how secure it needs to be.

Not everyone is building a banking app.

Everyone needs to ship software.

Security is all about making tradeoffs.

Every app we ship has an appropriate balance between security and usability.

Security Best Practices

(Demystified)

10

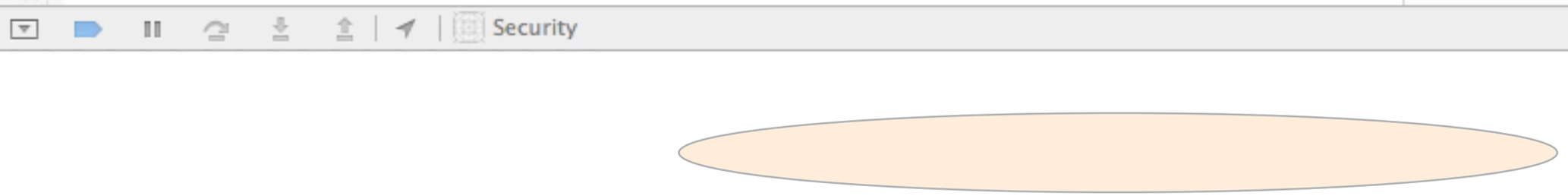
things you can do to build more secure software

Number 1:
NSLog

But logs are helpful!

What could possibly go wrong?

```
            NSLog(@"Downloading issues...");
        });
    });
}

- (void)loginWithUsername:(NSString *)username password:(NSString *)password {
    NSLog(@"Logging in with username:%@ password:%@", username, password);

    NSURLRequest *request = [[NSURLRequest alloc] initWithURL:[NSURL URLWithString:kLoginURL]];

    [NSURLConnection
     sendAsynchronousRequest:request
     queue:[NSOperationQueue mainQueue]
     completionHandler:^(NSURLResponse *response, NSData *data, NSError *connectionError) {

     }];
}

@end
```



1

Mashable

Dangers of Over Logging

Any good lumberjack knows not to over-log.

Logs can accidentally leak sensitive user data.

- User passwords
- Sensitive web requests
- Locations

Prevent production logs

Think before you log

Use a logging macro and limit the scope of production logs.

- Don’t log everything.
- Never log passwords or credit cards.
- Custom logging levels.

Logging Macro

Using a logging macro to programmatically disable logs

```
#pragma mark - LOGGING

//----  To disable all Logs in release mode ----//
#ifdef DEBUG
#define UBLog( s, ... )  NSLog(s, ## __VA_ARGS__)
#else
#define UBLog( s, ... )
#endif
```

Great case for writing tests

Run tests that verify you have no logs on your build system

We wrote a script to analyze our code and look for ‘NSLog’ in all of our files that would fail the build if it found any.

- Keeps everyone honest.
- Great for automation.
- Continuous Integration system.

```
ruby file_sieve.rb -d Classes/ -e UnionBankGlobals NSLog
```

Number 2:
Fast App Switching


PayPal
Tweetbot

Hiding Sensitive Data

Hide your kids, hide your text fields

Hiding sensitive information when the app goes to the background protects user privacy.

- Blur, Remove, Obscure

PayPal does this really well

Hiding Sensitive Data

When the app enters the background

```
#pragma mark - Security

- (void)applicationDidEnterBackground:(UIApplication *)application {
    RootViewController *rootVC = (RootViewController *)self.window.rootViewController;
    UIViewController *viewController = (UIViewController *)[rootVC.presentedViewControllers lastObject];

    /**
     If the last presented view controller is a SplashViewController we must not present the splash
     screen again.
     */
    if (![viewController isKindOfClass:[SplashViewController class]]) {
        /**
         We're presenting a splash screen when the app backgrounds as a privacy and security feature.
         This view needs to block the UI so that a screenshot of sensitive data does not accidentally
         get saved to the background.
         */
        [self presentSplashViewController];
    }
}
```

Number 3:
SSL Pinning

Easy way to ~~prevent MITM~~

(make MITM much more difficult)

Compare a server's certificate with a known certificate

Certificate Pinning

More secure, but fragile

Compare the exact certificate used in the request with the known certificate included in your app bundle.

- Certificate == Certificate?
- Easy to implement.
- Won’t work with certificate rotation.
- Communication fails after the certificate expires.

Public Key Pinning

Still secure, less fragile

Validate the key used in signed network communication with known certificate's public key.

- Public key remains the same for a longer time.
- Not as fragile.
- Best Practice.

AFNetworking

Most popular Objective-C library on GitHub!

Built in support for SSL Pinning in 1.x and 2.x.

- Super easy to use.
- Supported by v1 and v2.
- AFSecurityPolicy in v2.
- Certificate or Public Key.
- **Use latest versions!!!**

SSL Pinning Resources

Helpful sites talking about SSL Pinning

- http://www.thoughtcrime.org/blog/authenticity-is-broken-in-ssl-but-your-app-ha/
- https://www.isecpartners.com/blog/2013/february/ssl-pinning-on-ios.aspx
- http://www.doubleencore.com/2013/03/ssl-pinning-for-increased-app-security/
- http://blog.lumberlabs.com/2012/04/why-app-developers-should-care-about.html
- https://www.imperialviolet.org/2011/05/04/pinning.html
- https://www.owasp.org/index.php/Pinning_Cheat_Sheet

Number 4:
ARC PIE

An ARC PIE a day keeps memory attacks away

Being Practical

So, how likely is this really?

In all honesty, protecting against memory attacks probably is not your job as an application developer.

- No, scribbling over NSString isn't practical.
- Don't make it easy on an attacker.

Use ARC

Let go, Luke.

ARC helps make sure you're correctly managing memory.

- Release old objects.

Keep PIE Enabled

Who doesn't like PIE?

Position Independent Execution is a compiler optimization that prevents memory address attacks and is enabled by default in Xcode.

- Honestly, there's no reason to disable this…

Number 5:
Text Fields

Sensitive Autocorrect

Be careful what you allow to autocorrect

Autocorrect results are stored unencrypted on the device.

- Disabled by default on secure text fields.
- UITextInputTraits secureTextEntry property.

UIPasteBoard

Would you trust every app with your bank account number?

The pasteboard is public and a malicious app could look for sensitive data on the pasteboard.

- Avoid storing sensitive data in the pasteboard.
- Disable copying on sensitive text input fields.
- Disabled by default on secure text fields.

1Password

How does 1Password deal with UIPasteBoard?

Expire sensitive content and mark it as:

org.nspasteboard.ConcealedType

- http://nspasteboard.org
- 1Password marks content as “concealed”.
- 1Password removes content after 90 seconds by default.

UIPasteBoard Resources

Helpful sites talking about the UIPasteBoard

- http://enharmonichq.com/sharing-data-locally-between-ios-apps/

Number 6:
Local Storage

One way to improve security?

Don’t store anything locally.

Safe things to store

Most user-generated content

Storing data locally helps improve the user experience.

- Photography
- Todo list
- Public content
- Podcasts, music, videos
- Application preferences

Dangerous things to store

Personal and sensitive data

Some apps deal with sensitive data, and storing this on the device can be dangerous.

- Medical information
- Financial information
- Copyrighted content
- Corporate information
- HTTPS Traffic (Cached by default)

Implementation Trade Offs

Theoretical versus Practical

Downloading a file using NSURLSessionDownloadTask gives you the ability to download in the background, but you can’t encrypt files on the fly.

- Classic User Experience vs. Security tradeoff.
- File stored in the clear for 10 seconds?
- User is in physical possession of device?
- Dangerous to support “open in” or “open with”?

In-Memory Core Data

Core Data, but without the persistence part

Sometimes you want to use Core Data even though you can't store anything locally.

- In-Memory store is fast
- Object Graph Management
- Fetching/Sorting

Number 7:
Keychain

Device needs a PIN code

Or a really long passcode

The keychain is more secure if the user sets a passcode on the device

- No way to tell if the user set a passcode.
- You CAN enforce having a passcode via configuration profiles for enterprise apps.

Keychain is NOT foolproof

The Keychain is not a silver bullet for secure storage.

Best not to store a user's password. Try and store a hash or session token instead.

- **Vulnerable to attack!**
- (requires physical access and jailbreak)

Number 8:
Encryption

File Encryption

AES all the things!

Use the latest file encryption provided by common crypto.

- AES 256
- User generated encryption key
- Don’t store encryption key in the keychain…

Pass the salt.

Data Protection

Apple wants to make this easy for you

Data Protection APIs provide an easy to use baseline protection for files on disk.

- Available in iOS 4.0.
- Default in iOS 7.0.
- Default is relatively weak, first user authentication.
- User passcode and device key.
- Far more secure if the user sets a passcode.

Implicit Data Protection is not a fail safe

Encrypted Core Data

Core Data + Incremental Store + Encryption.
What could possibly go wrong?

Sometimes you really need an encrypted database. When you do, there's an encrypted NSIncrementalStore.

- Encrypted Core Data persistent store.
- Implementations exist.
- Encrypted Core Data is very complex.
- Consider carefully!

Encryption Resources

Helpful sites talking about encryption

- http://www.stoeger-it.de/en/secureincrementalstore/
- https://github.com/project-imas/encrypted-core-data
- http://adcdownload.apple.com//videos/wwdc_2012__sd/session_714__protecting_the_users_data.mov
- http://blog.agilebits.com/2013/03/09/guess-why-were-moving-to-256-bit-aes-keys/
- http://www.apple.com/legal/more-resources/law-enforcement/

Number 9:

Third Party Libraries

Hope is NOT a strategy

Third Party Library Security

Hope is not a strategy. Make sure libraries are secure.

You're responsible for what third party libraries are doing in your app. If you're following the rules above, make sure they are too.

- Writing anything to disk?
- Communicating over network?
- Logging to console?

“Heartbleed isn't a crypto problem. It's a software engineering problem. It's a software engineering problem related to the difficulties of writing, maintaining, and supporting a *system*…”

Jon Callas, ‘Apple and OpenSSL’
http://www.metzdowd.com/pipermail/cryptography/2014-April/020977.html

Number 10:
Have a Backup Plan

If all else fails…

The last one standing blows the bridge.

You can always use a remote server to force an update. You can lock out the app in the event of a security breach, and force users to install the update that fixes it.

- Gives you a pretty heavy hammer.
- Easy to implement.
- Important for high-risk apps like banking.

Additional Resources

Helpful places to go to learn more about security.

- http://blog.veracode.com/2014/03/introducing-the-ios-reverse-engineering-toolkit/
- https://viaforensics.com/resources/reports/best-practices-ios-android-secure-mobile-development/
- http://twit.tv/show/security-now/446
- https://www.isecpartners.com/media/12985/secure_development_on_ios.pdf
- http://images.apple.com/ipad/business/docs/iOS_Security_Feb14.pdf


SECURE
ALL THE THINGS!

Danke!

Secure all the things!

Conrad Stoll

@conradstoll - conradstoll.com

Architect mutualmobile

UIKonf

Appendix A:
NSUserDefaults

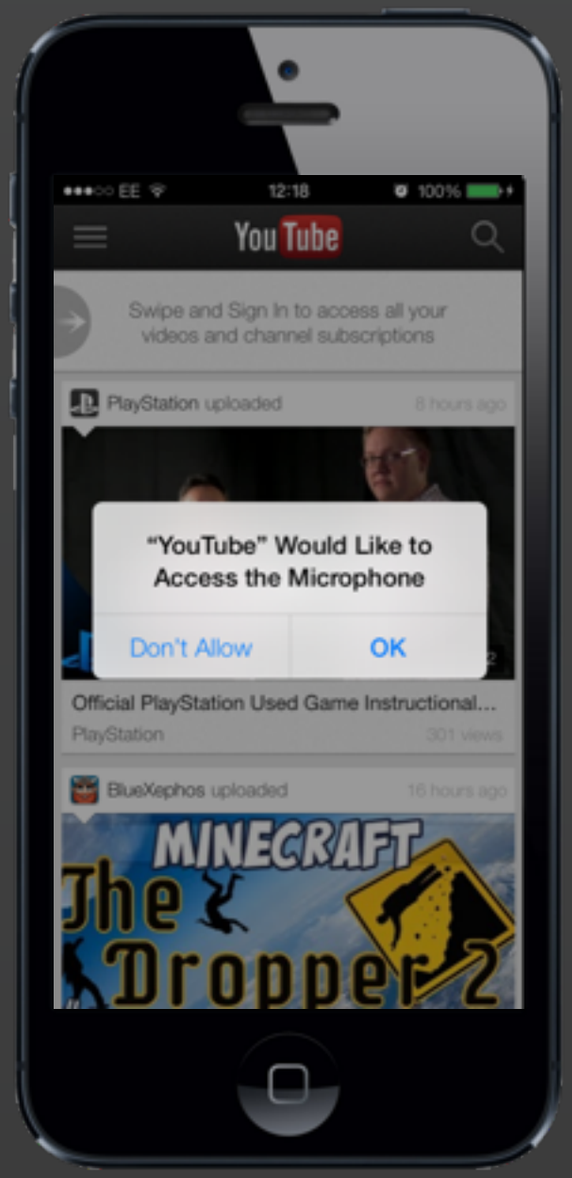
NSUserDefaults != Keychain

If you wouldn't store it in the keychain, definitely don't store it in NSUserDefaults!

NSUserDefaults is a great place to store basic app preferences or bits of state, like whether or not this is the app's first launch. But it should NOT be used to store anything sensitive.

- Unencrypted key value storage.
- Should never be used to store anything sensitive.

Appendix B:
Asking Permission


EE
12:18
100%
YouTube
Swipe and Sign In to access all your videos and channel subscriptions
PlayStation uploaded
8 hours ago
"YouTube" Would Like to Access the Microphone
Don't Allow
OK
Official PlayStation Used Game Instructional...
PlayStation
301 views
BlueXephos uploaded
16 hours ago
MINECRAFT
The Dropper 2

iOS- How to avoid the dialog “Would like to use your current location”

0

In my app, one of my input is to get the location of the user as well as the contacts of the user. This is done from the code.

When the user runs the app for the first time,they get a dialog

"AppName" would like to use your current location. I wish to avoid this dialog since this is an important data and dont want the users to accidently press "Dont Allow"

How to avoid this dialog. Could any one please let me know. Thanks

Being polite when asking for permission

How you ask matters

Many users distrust software

Remember that context for your request is very important.

- Don’t ask on app launch or login.
- Ask in response to a user action.
- Only ask for what you clearly need.
- Explain why you need permission.

Respect User Permission

Be a good house guest

Be considerate of a user's permission, and don't overstep your bounds.

- Don't copy the user's address book.
- Don't log location data to the server.
- Don't transmit user data in the clear.
- Don't store passwords on your server.

Permission Resources

Helpful sites talking about permission

- http://techcrunch.com/2014/04/04/the-right-way-to-ask-users-for-ios-permissions/

Appendix C:
UIWebView

Whitelist domains

Support safe browsing

Prevent local network spoofing attacks by only allowing UIWebView requests to specific domains.

- Compare requests made by your web view to a specific list of domains.
- Enforce HTTPS on all requests from the web view.

Whitelisting Domains

Teaching someone how not to phish

```
#pragma mark - Security

/*
 Only allow the web view to load requests to whitelisted domains in
order to prevent phishing attacks.
 */
- (BOOL)webView:(UIWebView *)webView
        shouldStartLoadWithRequest:(NSURLRequest *)request
        navigationType:(UIWebViewNavigationType)navigationType {
    if ([self shouldAllowWebViewNavigationToURLRequest:request]) {
        return YES;
    }

    [self presentUnsafeBrowsingWarningForRequest:request];

    return NO;
}
```

No SSL Pinning Support

UIWebView doesn't benefit from SSL Pinning elsewhere in your app

Web views don't get challenge callbacks, so it is hard to implement SSL Pinning with them (though not impossible).

- Try not to use web views for anything super sensitive.
- Careful relying on a web view without this.

Leave No Trace

UIWebView doesn't do a great job cleaning up after itself

Watch out for the cookies, local data, cached information, etc. that UIWebView leaves behind.

- UIWebView has a habit of caching request data
- Similar to NSURLRequest, caches by default

UIWebView Resources

Helpful sites talking about web views

- https://www.isecpartners.com/blog/2013/february/ssl-pinning-on-ios.aspx
- http://stackoverflow.com/questions/11573164/uiwebview-to-view-self-signed-websites-no-private-api-not-nsurlconnection-i

Appendix D:
Enterprise MDM

For enterprise apps…

Use MDM Features

There’s lots of powerful security features in MDM. If you’re building an enterprise app, make sure your client knows about this!

- Force device passcode
- Per-app VPN
- Configuration enforcement
- Encrypted backups
- Remote wipe