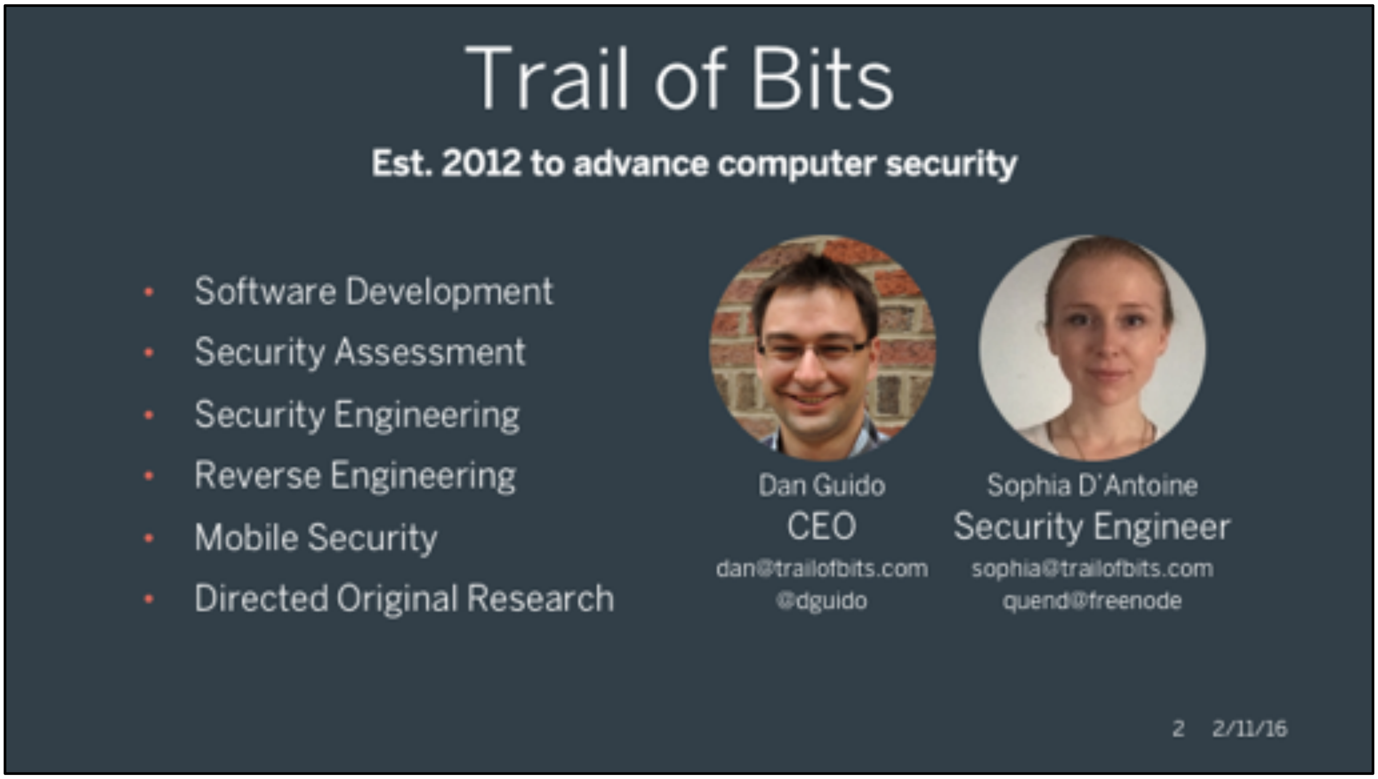
TRAIL OF BITS

iOS Application Security

1 2/11/16



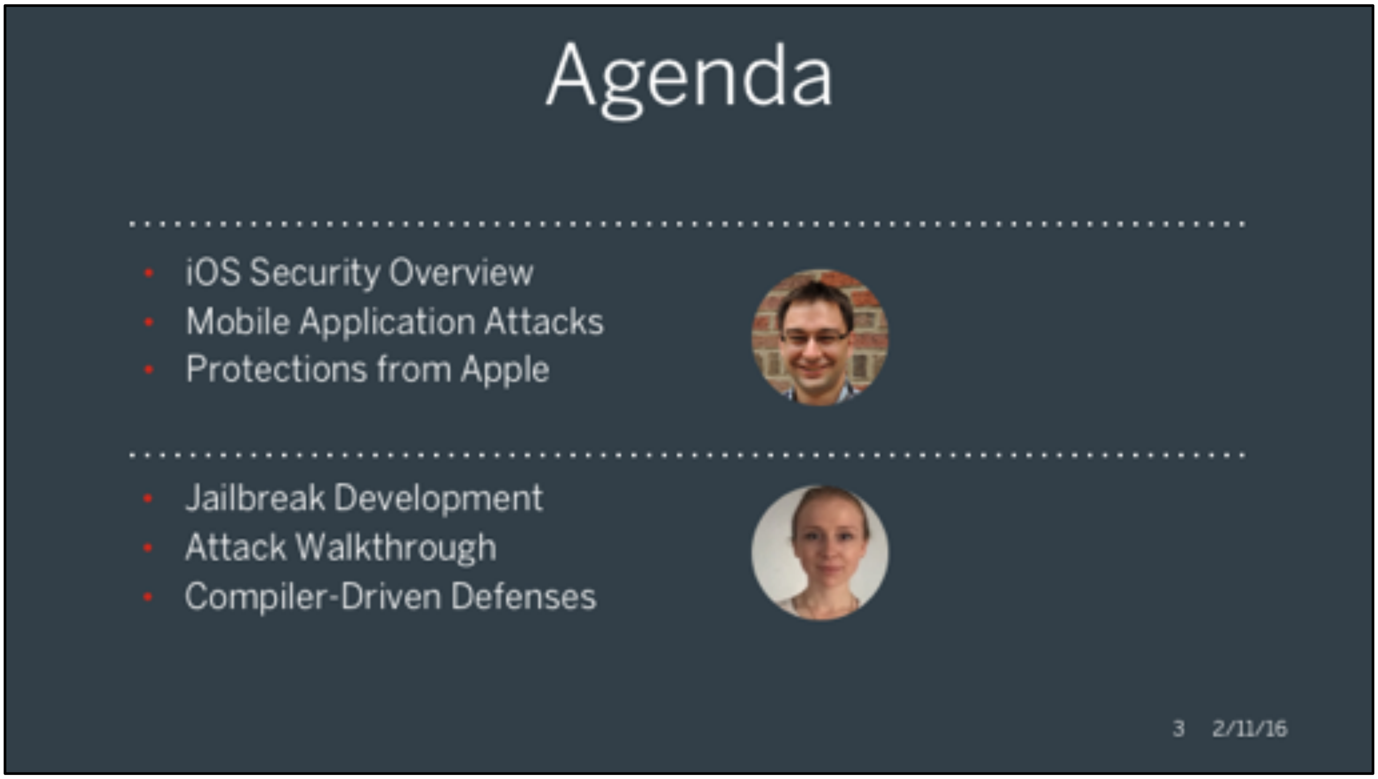
Trail of Bits

Est. 2012 to advance computer security

- Software Development
- Security Assessment
- Security Engineering
- Reverse Engineering
- Mobile Security
- Directed Original Research

Dan Guido
CEO
dan@trailofbits.com
@dguido

Sophia D'Antoine
Security Engineer
sophia@trailofbits.com
quend@freenode

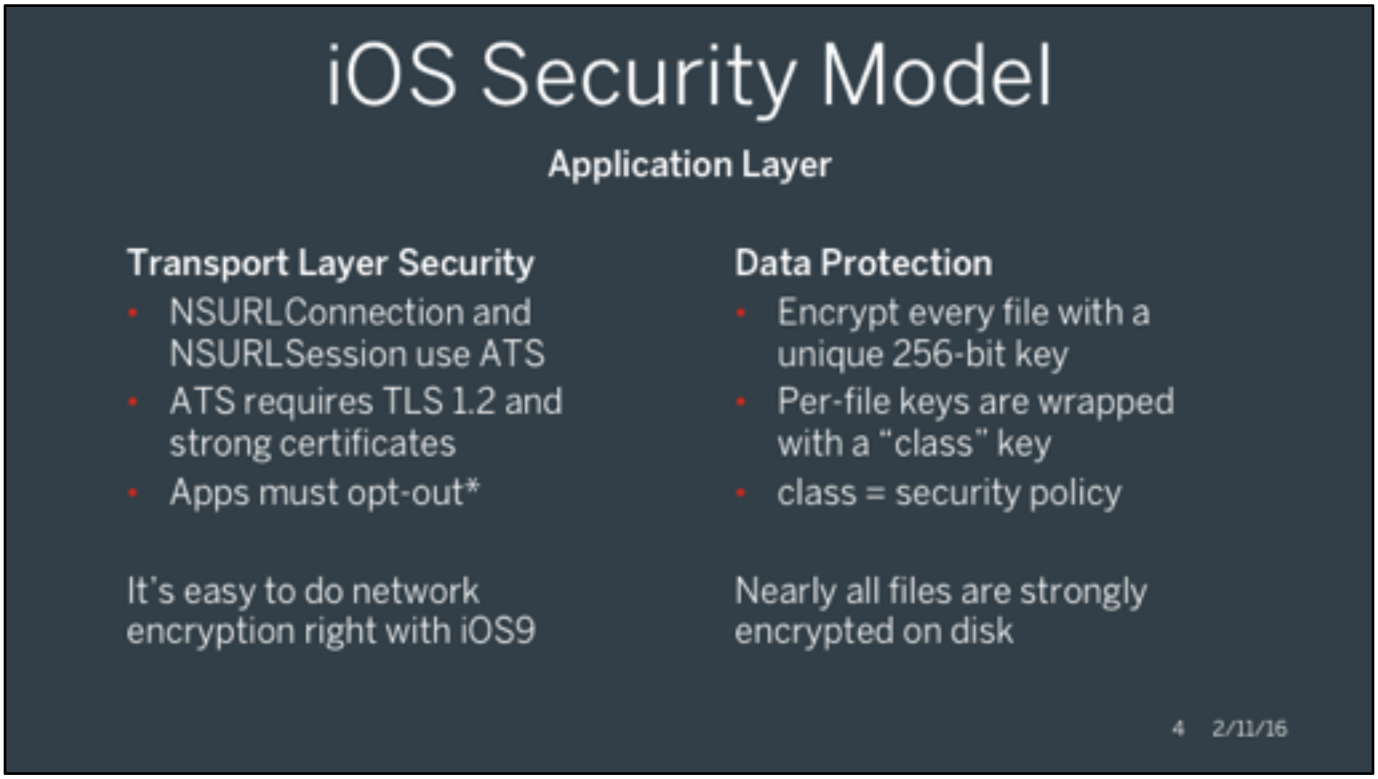


Agenda

- iOS Security Overview
- Mobile Application Attacks
- Protections from Apple

- Jailbreak Development
- Attack Walkthrough
- Compiler-Driven Defenses

3 2/11/16



iOS Security Model

Application Layer

Transport Layer Security

- NSURLConnection and NSURLSession use ATS
- ATS requires TLS 1.2 and strong certificates
- Apps must opt-out*

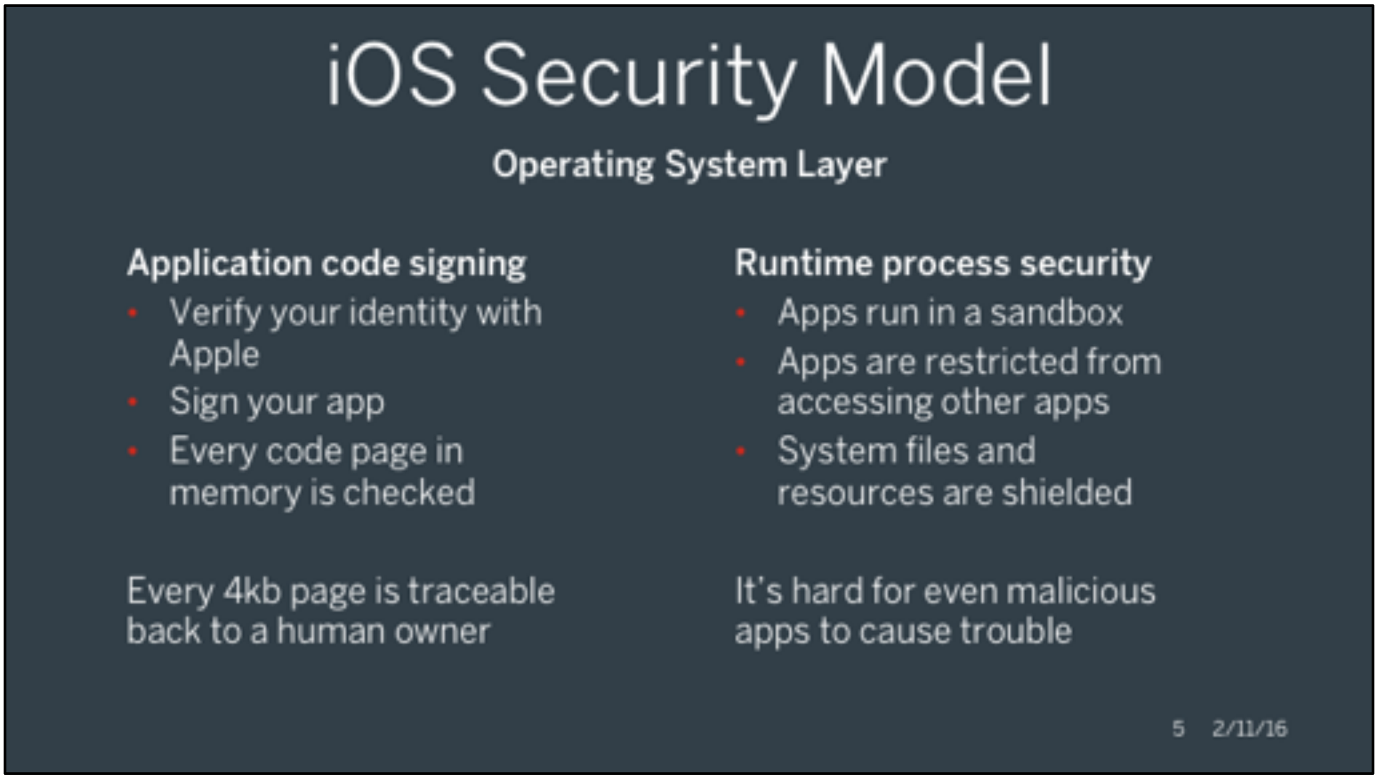
It's easy to do network encryption right with iOS9

Data Protection

- Encrypt every file with a unique 256-bit key
- Per-file keys are wrapped with a "class" key
- class = security policy

Nearly all files are strongly encrypted on disk

Here are some security buzzwords you may have heard before and what they mean
These are features Apple uses to protect you



iOS Security Model

Operating System Layer

Application code signing

- Verify your identity with Apple
- Sign your app
- Every code page in memory is checked

Every 4kb page is traceable back to a human owner

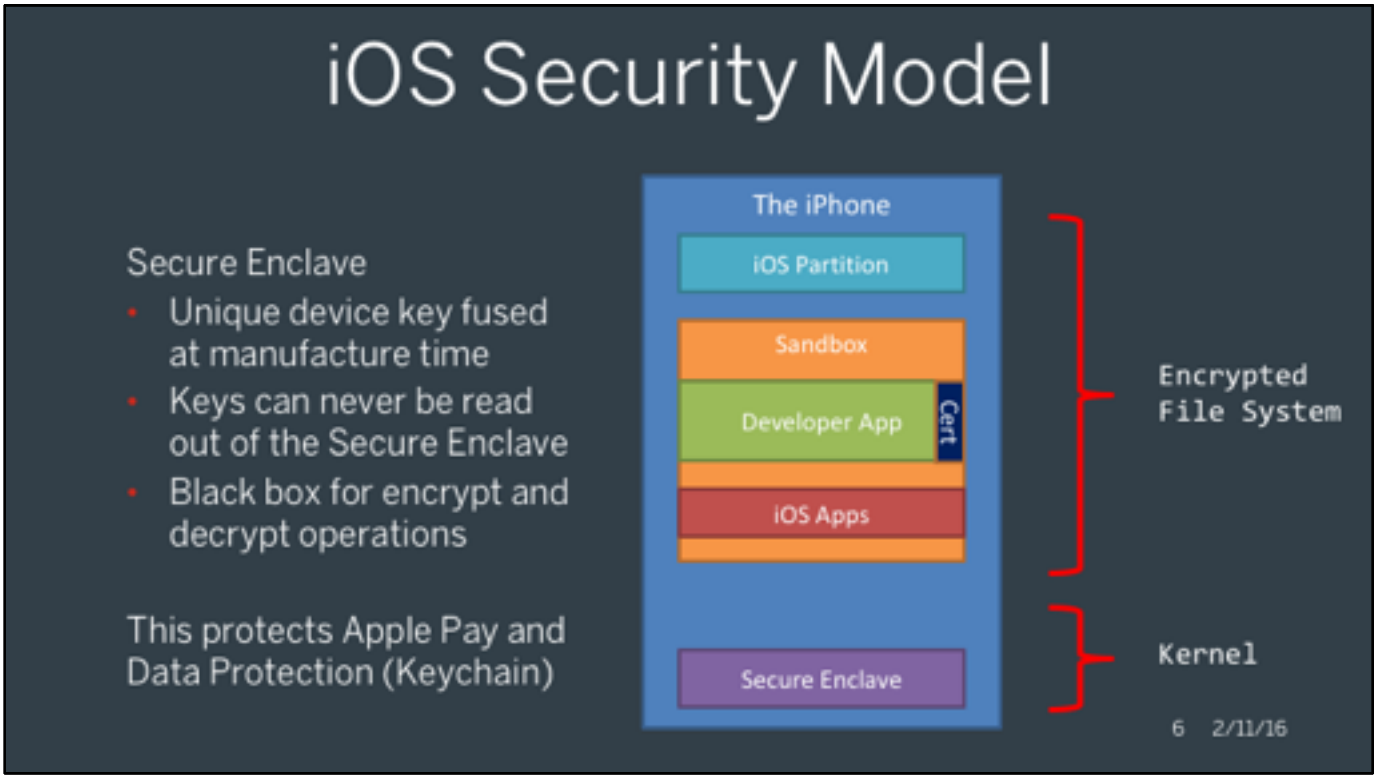
Runtime process security

- Apps run in a sandbox
- Apps are restricted from accessing other apps
- System files and resources are shielded

It's hard for even malicious apps to cause trouble

5 2/11/16

These are features that Apple uses to protect themselves



iOS Security Model

Secure Enclave

- Unique device key fused at manufacture time
- Keys can never be read out of the Secure Enclave
- Black box for encrypt and decrypt operations

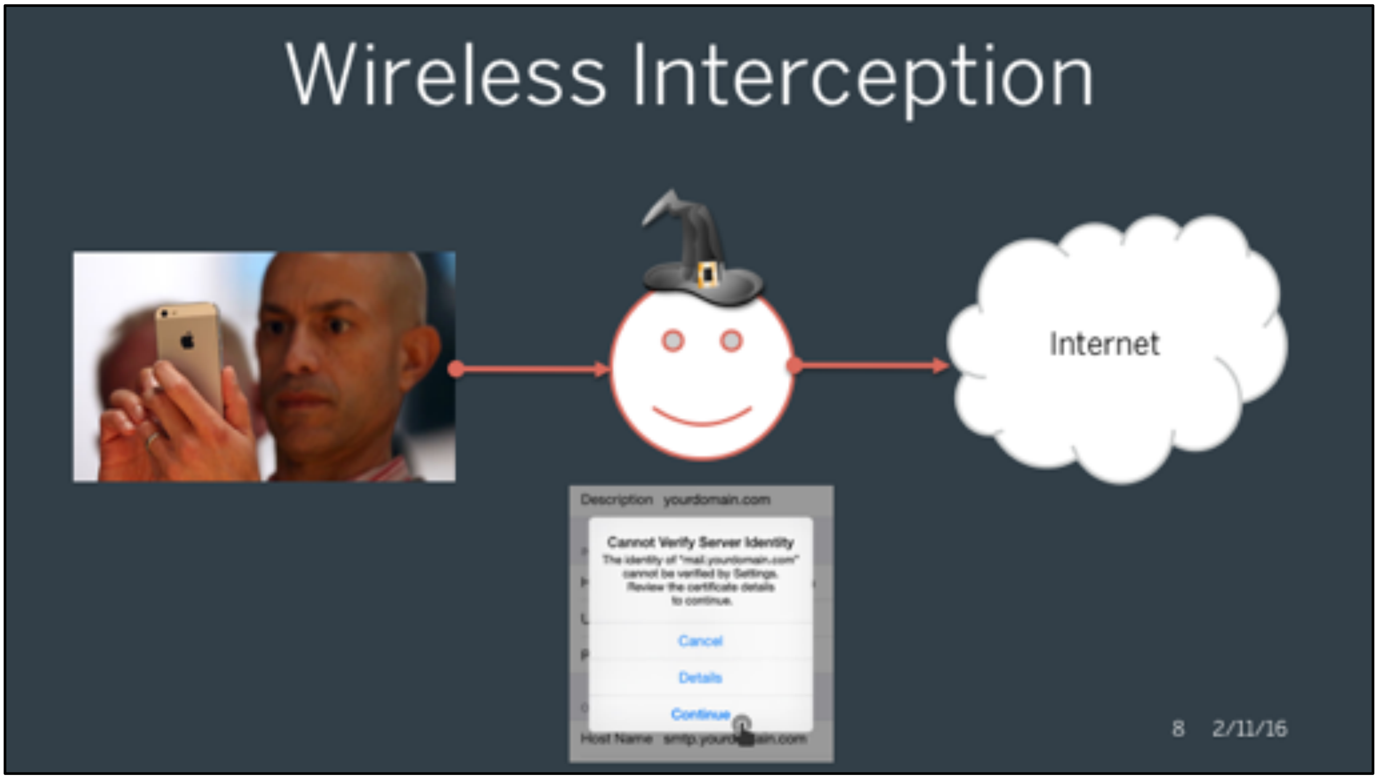
This protects Apple Pay and Data Protection (Keychain)

6 2/11/16

New stuff! Secure Enclave is how they protect Apple Pay.

iOS Application Security

Wireless Interception



Description yourdomain.com

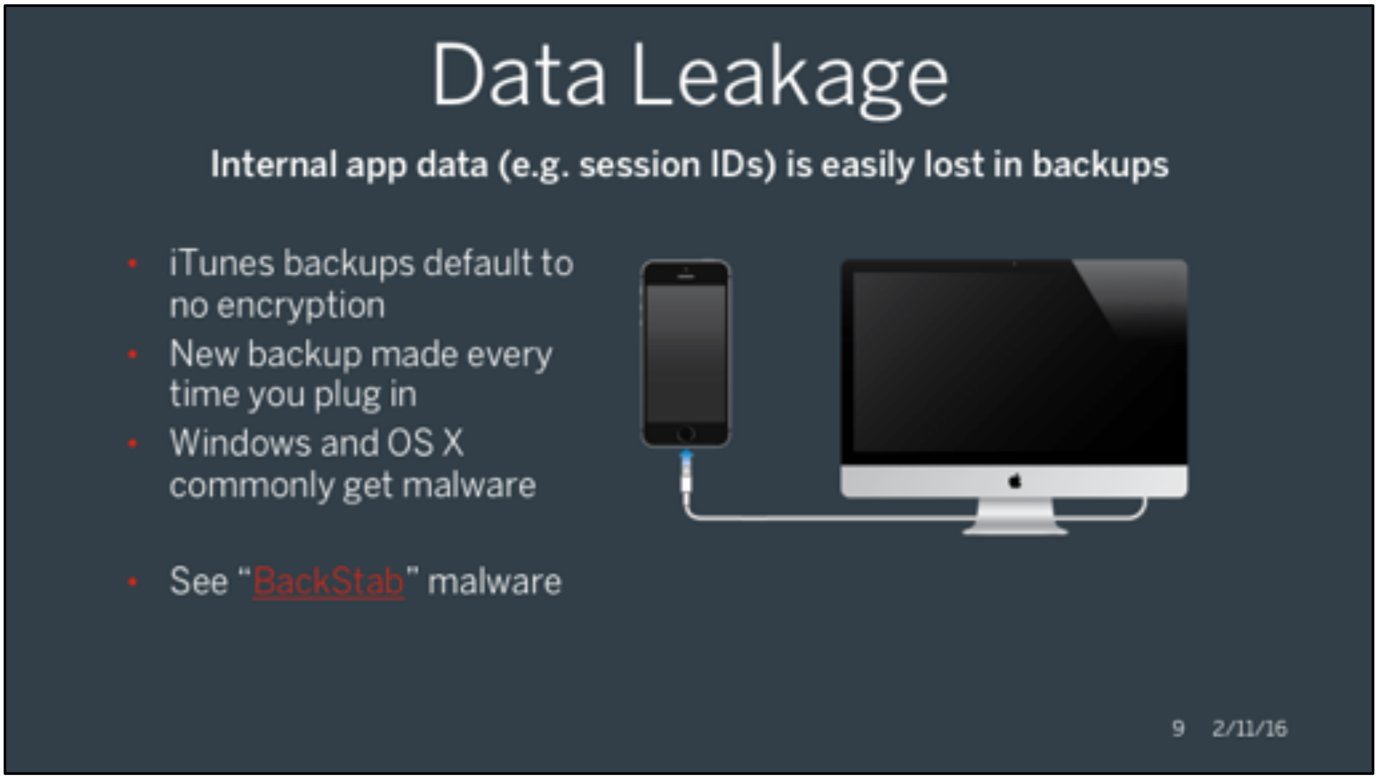
Cannot Verify Server Identity

The identity of "mail.yourdomain.com" cannot be verified by Settings. Review the certificate details to continue.

Host Name smtp.yourdomain.com

8 2/11/16

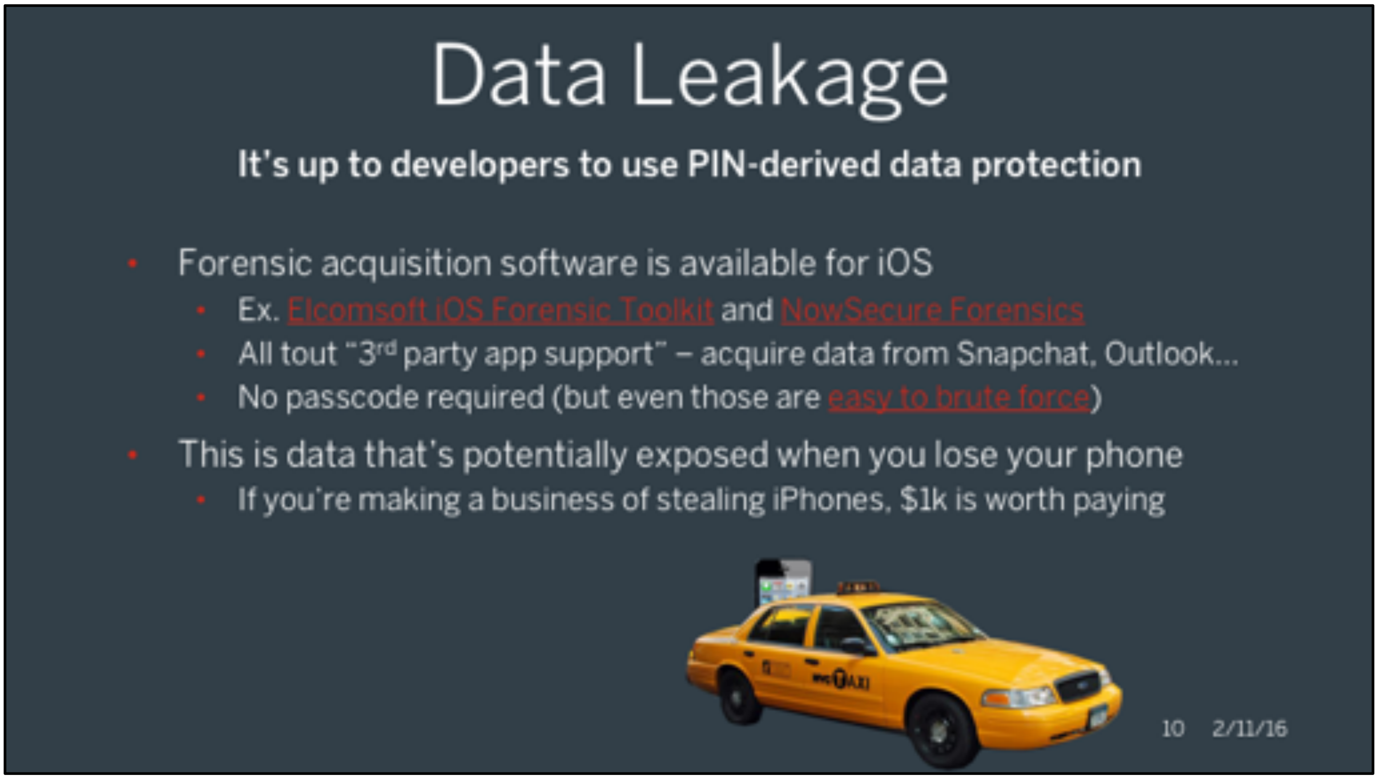
MITM / Proxy
Reverse engineer an API spec from network traffic



Data Leakage

Internal app data (e.g. session IDs) is easily lost in backups

- iTunes backups default to no encryption
- New backup made every time you plug in
- Windows and OS X commonly get malware
- See "BackStab" malware

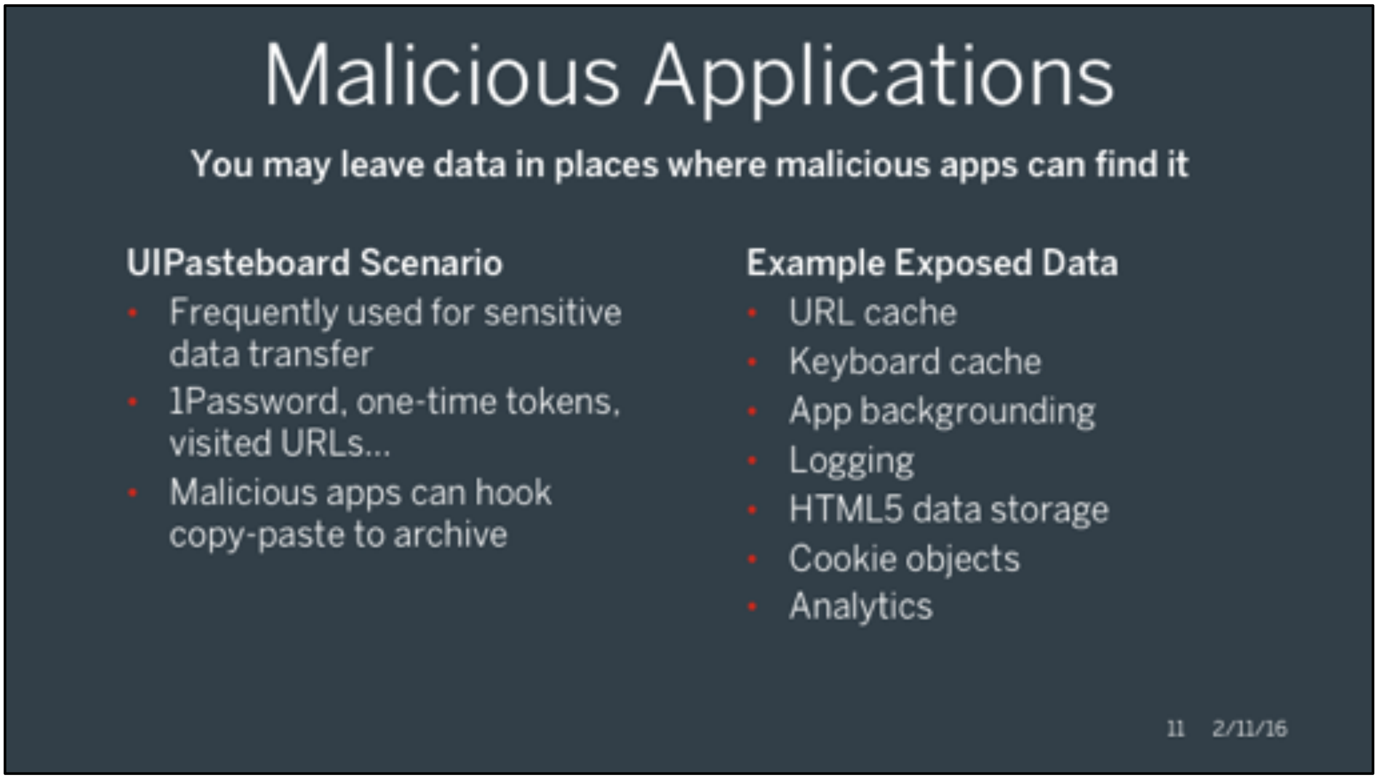


Data Leakage

It's up to developers to use PIN-derived data protection

- Forensic acquisition software is available for iOS
  - Ex. Elcomsoft iOS Forensic Toolkit and NowSecure Forensics
  - All tout "3rd party app support" – acquire data from Snapchat, Outlook…
  - No passcode required (but even those are easy to brute force)
- This is data that's potentially exposed when you lose your phone
  - If you're making a business of stealing iPhones, $1k is worth paying

10 2/11/16



Malicious Applications

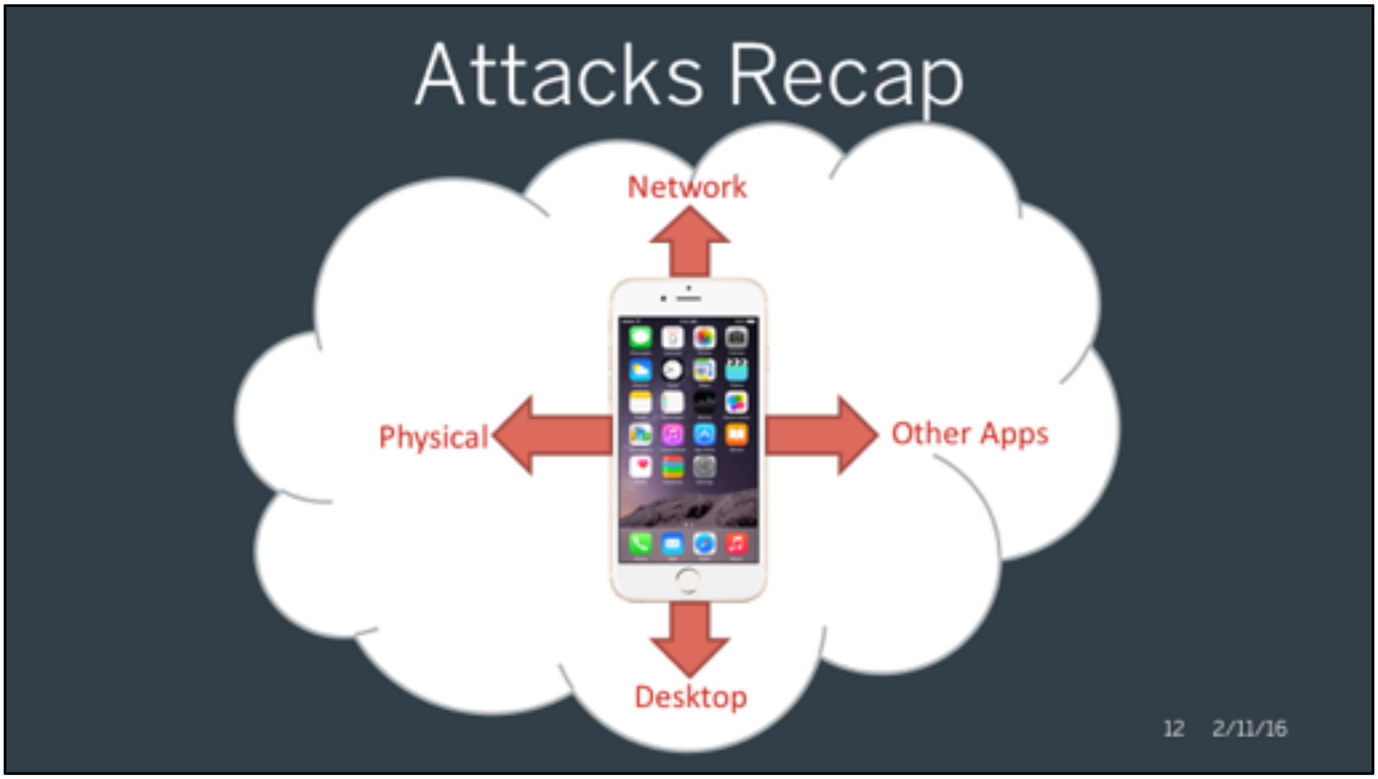
You may leave data in places where malicious apps can find it

UIPasteboard Scenario

- Frequently used for sensitive data transfer
- 1Password, one-time tokens, visited URLs…
- Malicious apps can hook copy-paste to archive

Example Exposed Data

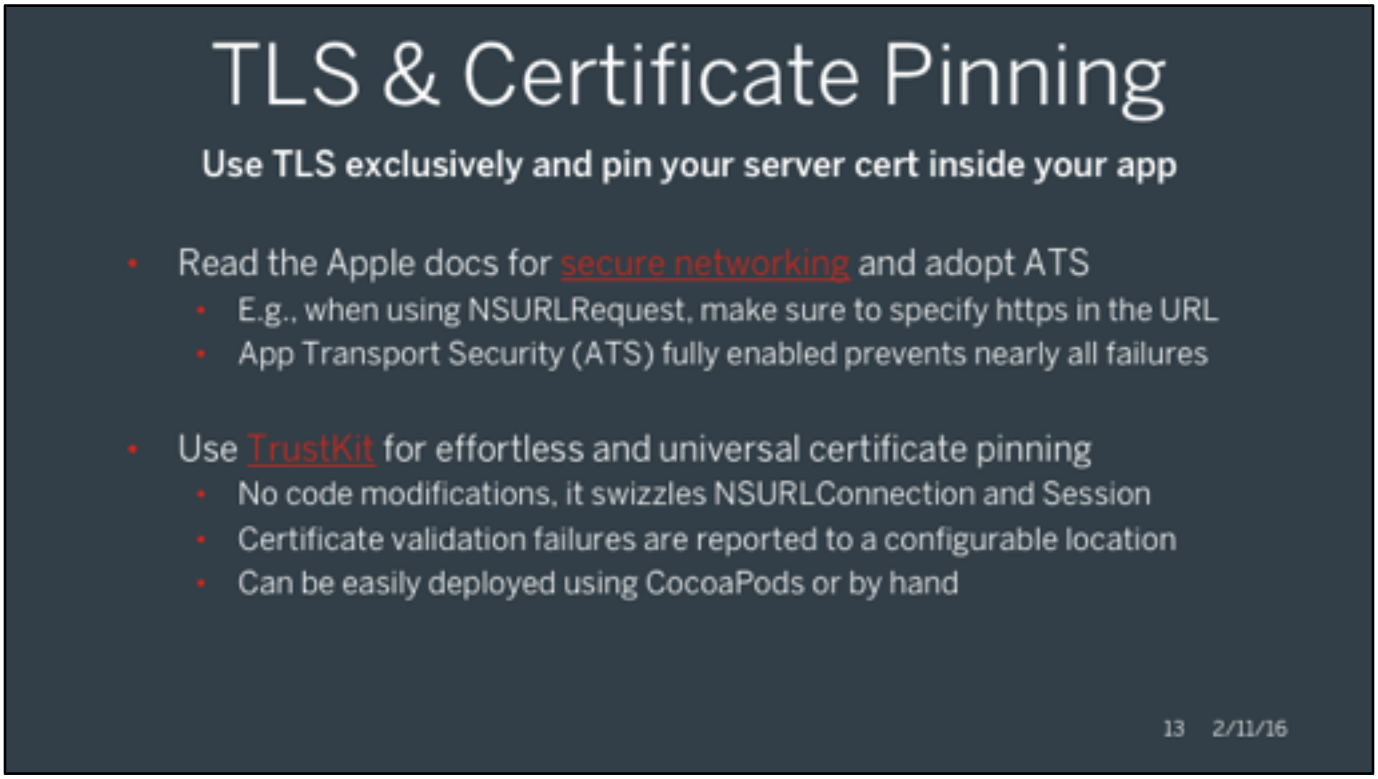
- URL cache
- Keyboard cache
- App backgrounding
- Logging
- HTML5 data storage
- Cookie objects
- Analytics



Attacks Recap

12 2/11/16

Pervasive data leakage via Network, Desktop, Physical, App



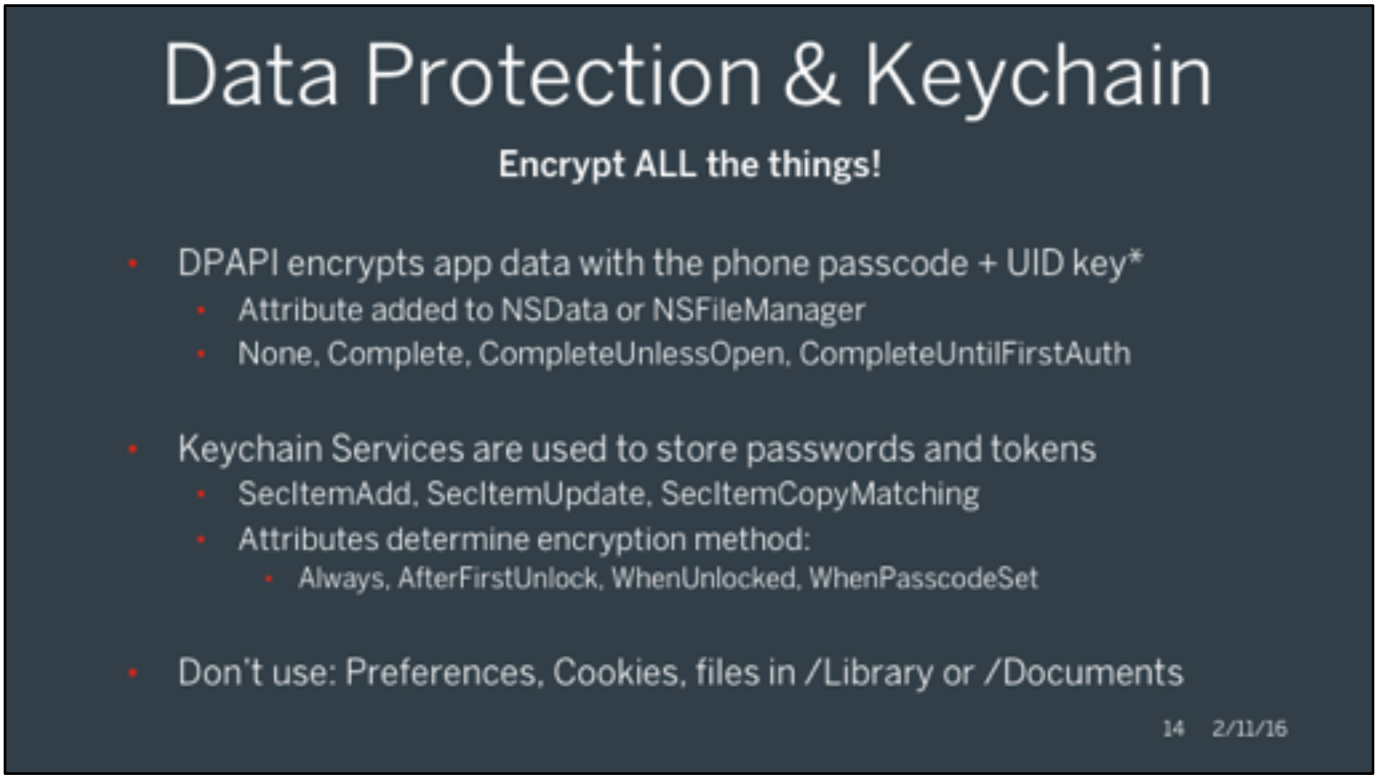
TLS & Certificate Pinning

Use TLS exclusively and pin your server cert inside your app

- Read the Apple docs for secure networking and adopt ATS
  - E.g., when using NSURLRequest, make sure to specify https in the URL
  - App Transport Security (ATS) fully enabled prevents nearly all failures
- Use TrustKit for effortless and universal certificate pinning
  - No code modifications, it swizzles NSURLConnection and Session
  - Certificate validation failures are reported to a configurable location
  - Can be easily deployed using CocoaPods or by hand

13 2/11/16

https://developer.apple.com/library/ios/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple_ref/doc/uid/TP40009251-SW33



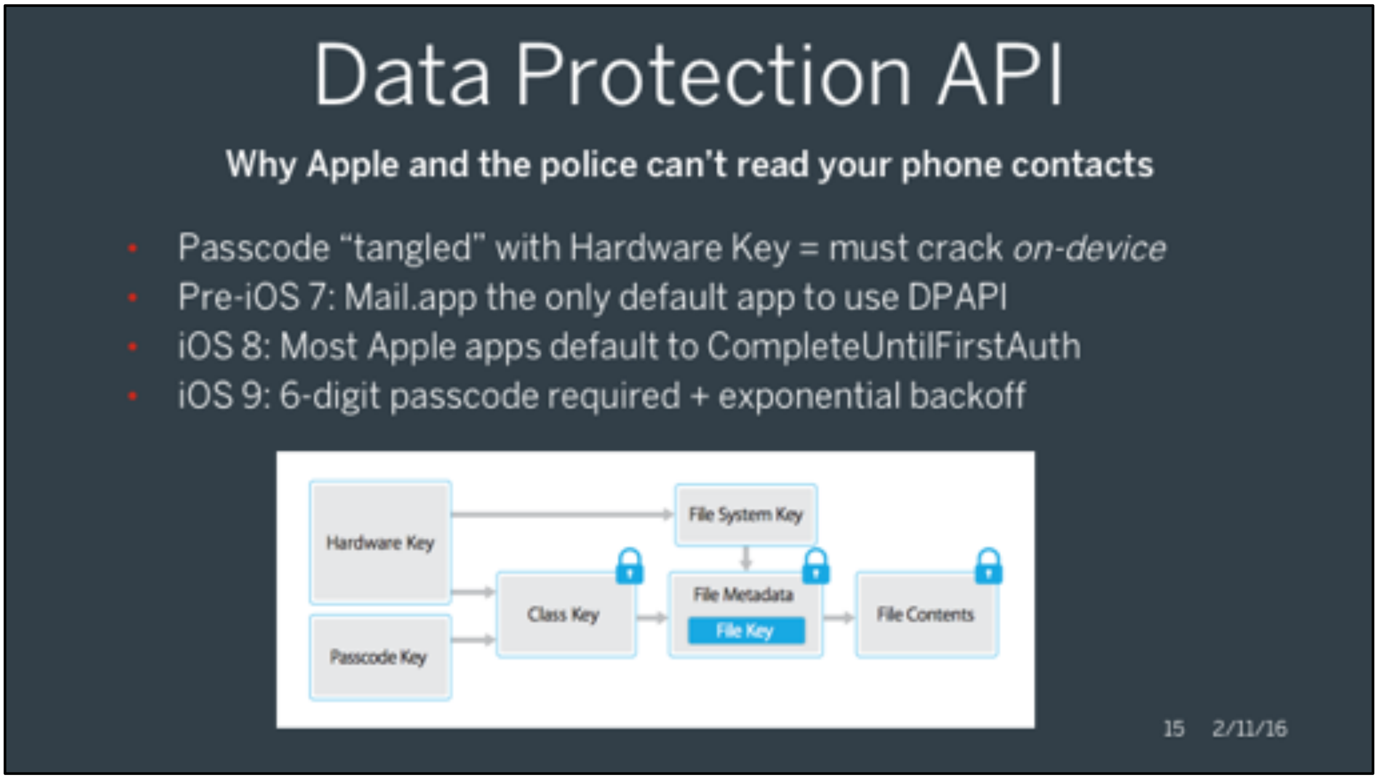
Data Protection & Keychain

Encrypt ALL the things!

- DPAPI encrypts app data with the phone passcode + UID key*
  - Attribute added to NSData or NSFileManager
  - None, Complete, CompleteUnlessOpen, CompleteUntilFirstAuth
- Keychain Services are used to store passwords and tokens
  - SecItemAdd, SecItemUpdate, SecItemCopyMatching
  - Attributes determine encryption method:
    - Always, AfterFirstUnlock, WhenUnlocked, WhenPasscodeSet
- Don't use: Preferences, Cookies, files in /Library or /Documents

14 2/11/16

https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/StrategiesforImplementingYourApp/StrategiesforImplementingYourApp.html#//apple_ref/doc/uid/TP40007072-CH5-SW21



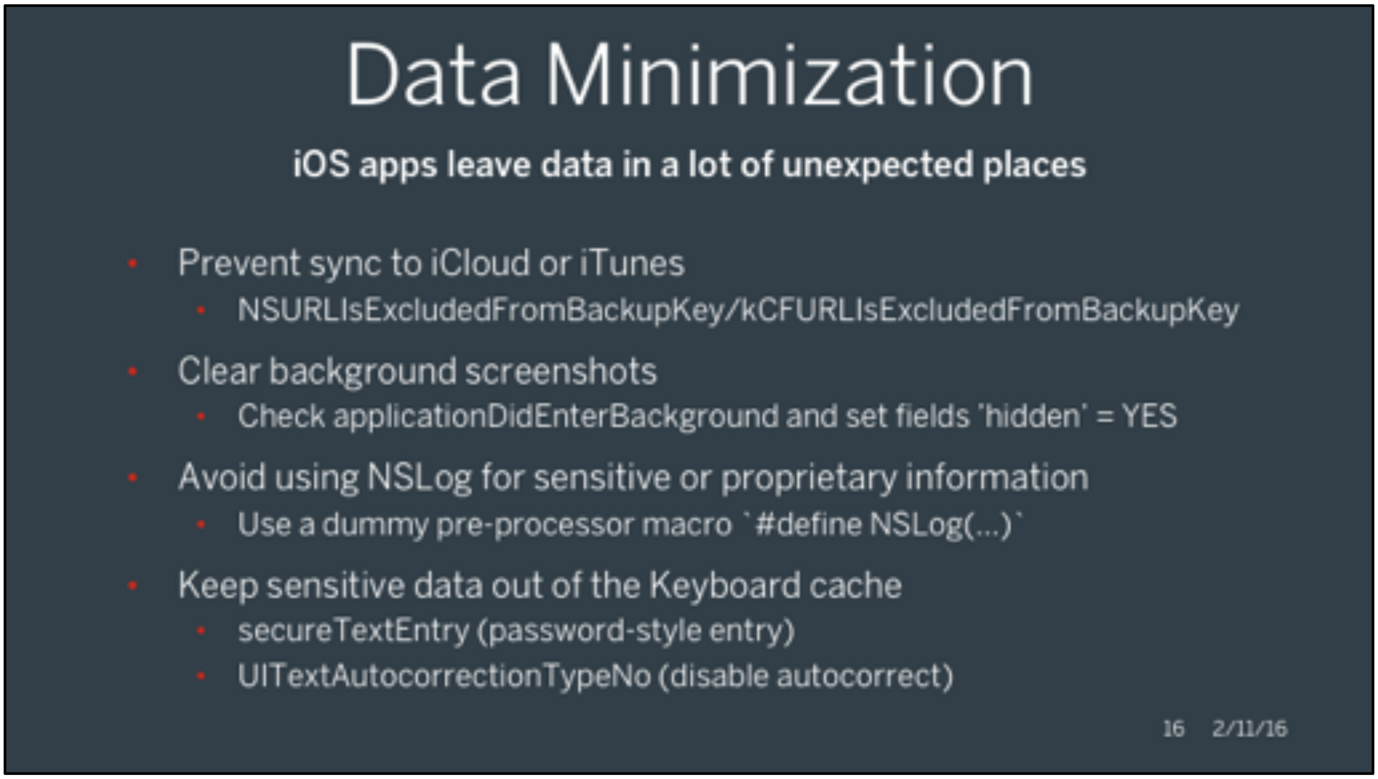
Data Protection API

Why Apple and the police can't read your phone contacts

- Passcode "tangled" with Hardware Key = must crack *on-device*
- Pre-iOS 7: Mail.app the only default app to use DPAPI
- iOS 8: Most Apple apps default to CompleteUntilFirstAuth
- iOS 9: 6-digit passcode required + exponential backoff

15 2/11/16

UID stored in Secure Enclave, exponential backoff per attempt, 4 digits were unfeasible, 6 is astronomical
After 9 guesses, it locks you out for an hour each time
http://blog.cryptographyengineering.com/2014/10/why-cant-apple-decrypt-your-iphone.html



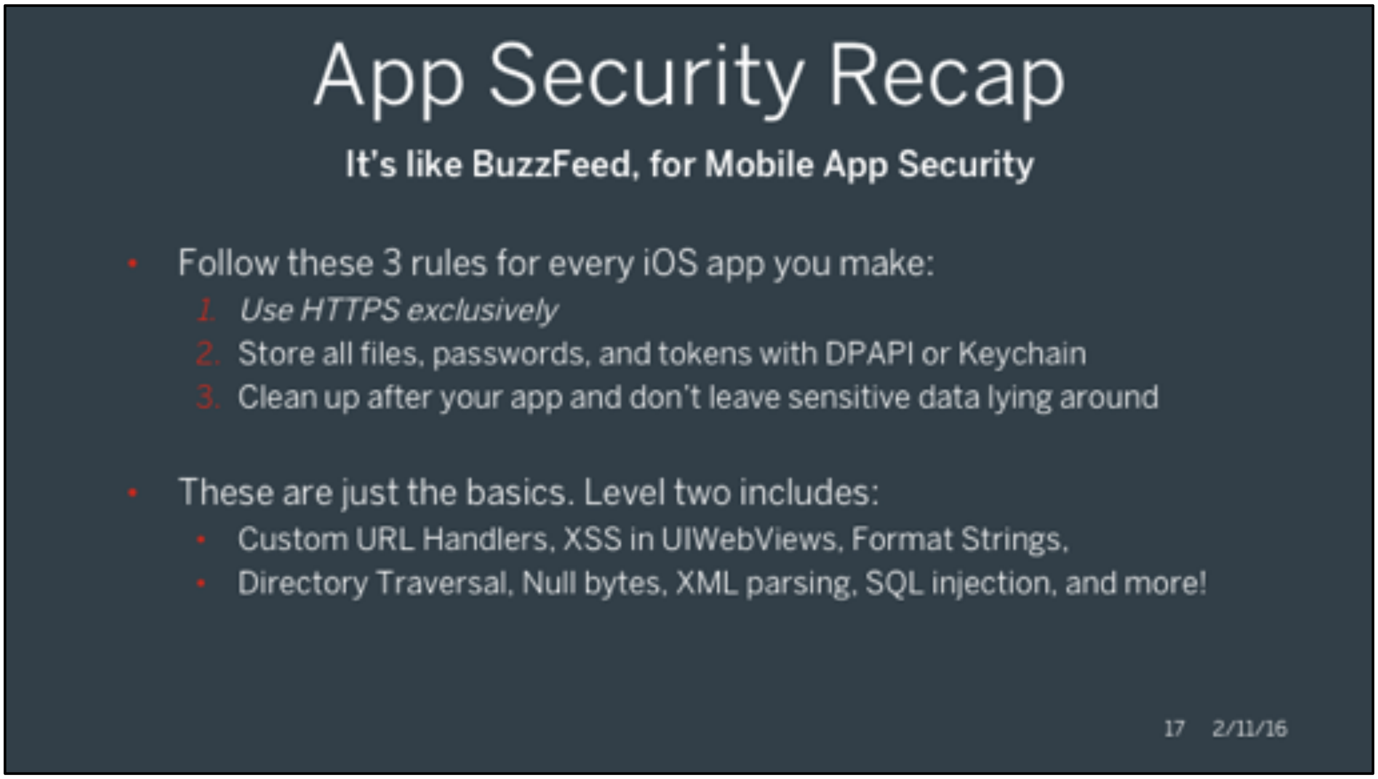
Data Minimization

iOS apps leave data in a lot of unexpected places

- Prevent sync to iCloud or iTunes
  - NSURLIsExcludedFromBackupKey/kCFURLIsExcludedFromBackupKey
- Clear background screenshots
  - Check applicationDidEnterBackground and set fields 'hidden' = YES
- Avoid using NSLog for sensitive or proprietary information
  - Use a dummy pre-processor macro `#define NSLog(...)`
- Keep sensitive data out of the Keyboard cache
  - secureTextEntry (password-style entry)
  - UITextAutocorrectionTypeNo (disable autocorrect)

16 2/11/16

https://developer.apple.com/library/ios/qa/qa1719/_index.html
https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITextInputTraits_Protocol/#//apple_ref/occ/intfp/UITextInputTraits/autocorrectionType



App Security Recap

It's like BuzzFeed, for Mobile App Security

- Follow these 3 rules for every iOS app you make:
  1. *Use HTTPS exclusively*
  2. Store all files, passwords, and tokens with DPAPI or Keychain
  3. Clean up after your app and don't leave sensitive data lying around

- These are just the basics. Level two includes:
  - Custom URL Handlers, XSS in UIWebViews, Format Strings,
  - Directory Traversal, Null bytes, XML parsing, SQL injection, and more!

17 2/11/16

Many recent iOS 9 security enhancements:

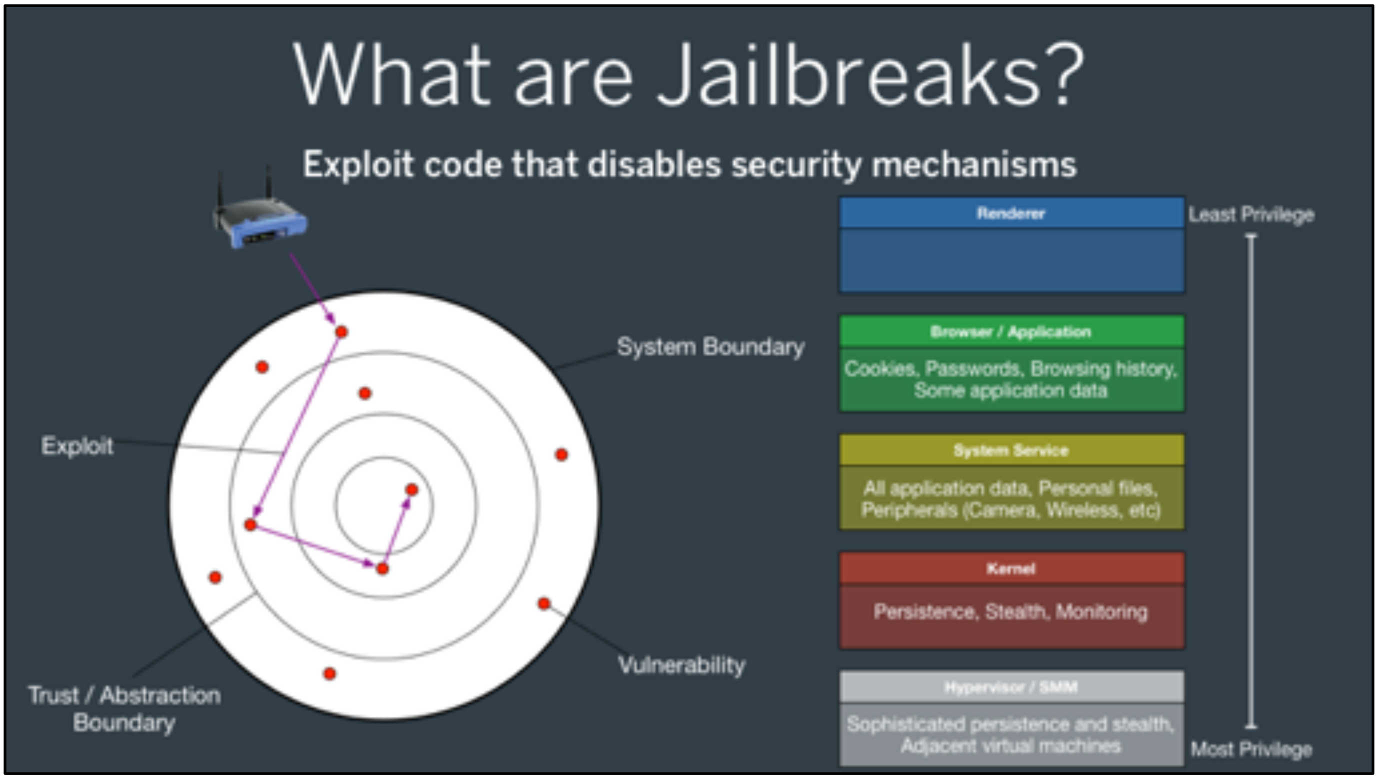
- System pasteboard can't be read when app is in background
- openURL has extra prompts for URL handlers now

http://www.andreas-kurtz.de/2014/09/malicious-apps-ios8.html

https://stackoverflow.com/questions/33408048/ios-9-pasteboard-not-able-to-write-in-pasteboard-while-in-background

Jailbreaks

18 2/11/16



What are Jailbreaks?

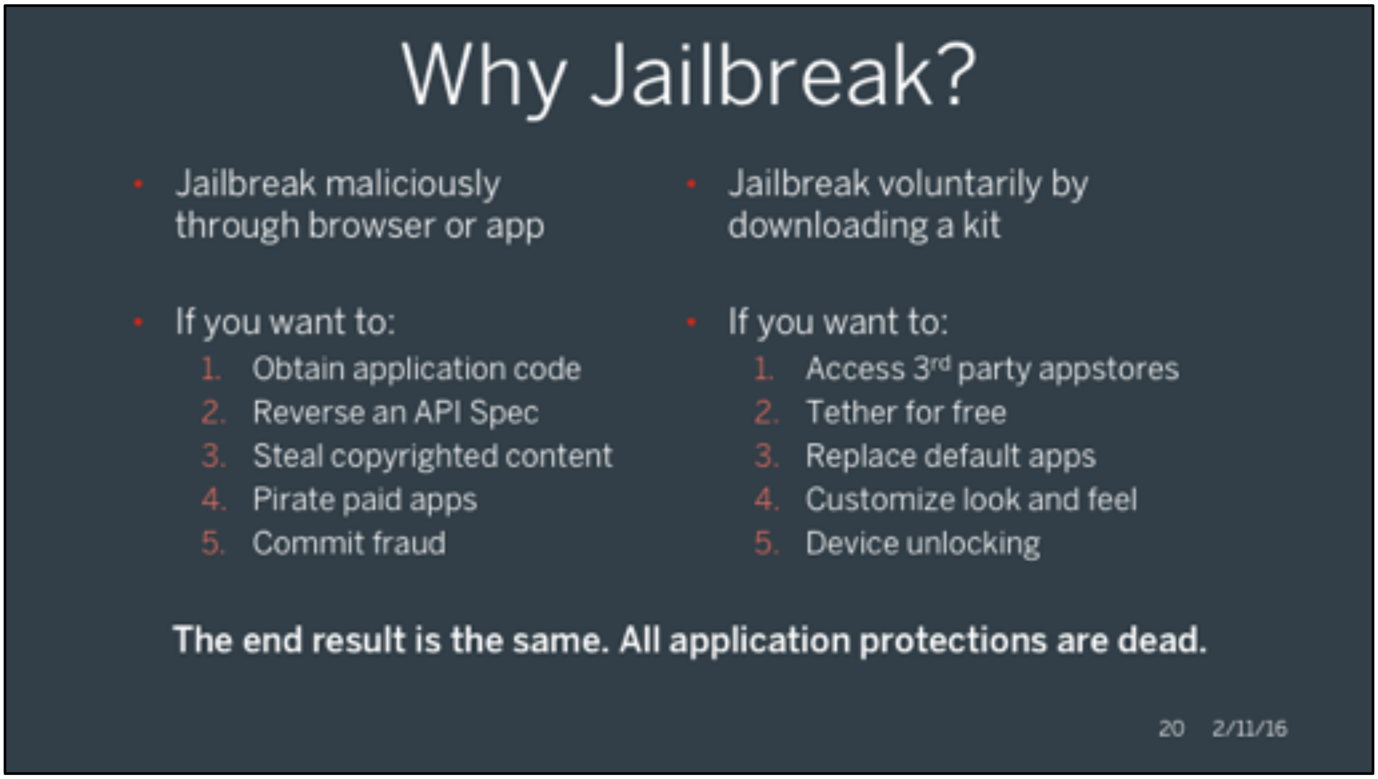
Exploit code that disables security mechanisms

Renderer	
Browser / Application	Cookies, Passwords, Browsing history, Some application data
System Service	All application data, Personal files, Peripherals (Camera, Wireless, etc)
Kernel	Persistence, Stealth, Monitoring
Hypervisor / SMM	Sophisticated persistence and stealth, Adjacent virtual machines

Least Privilege

Most Privilege

Apple makes this really hard. Teams of people working for months on end, bypassing layer after layer of protections.



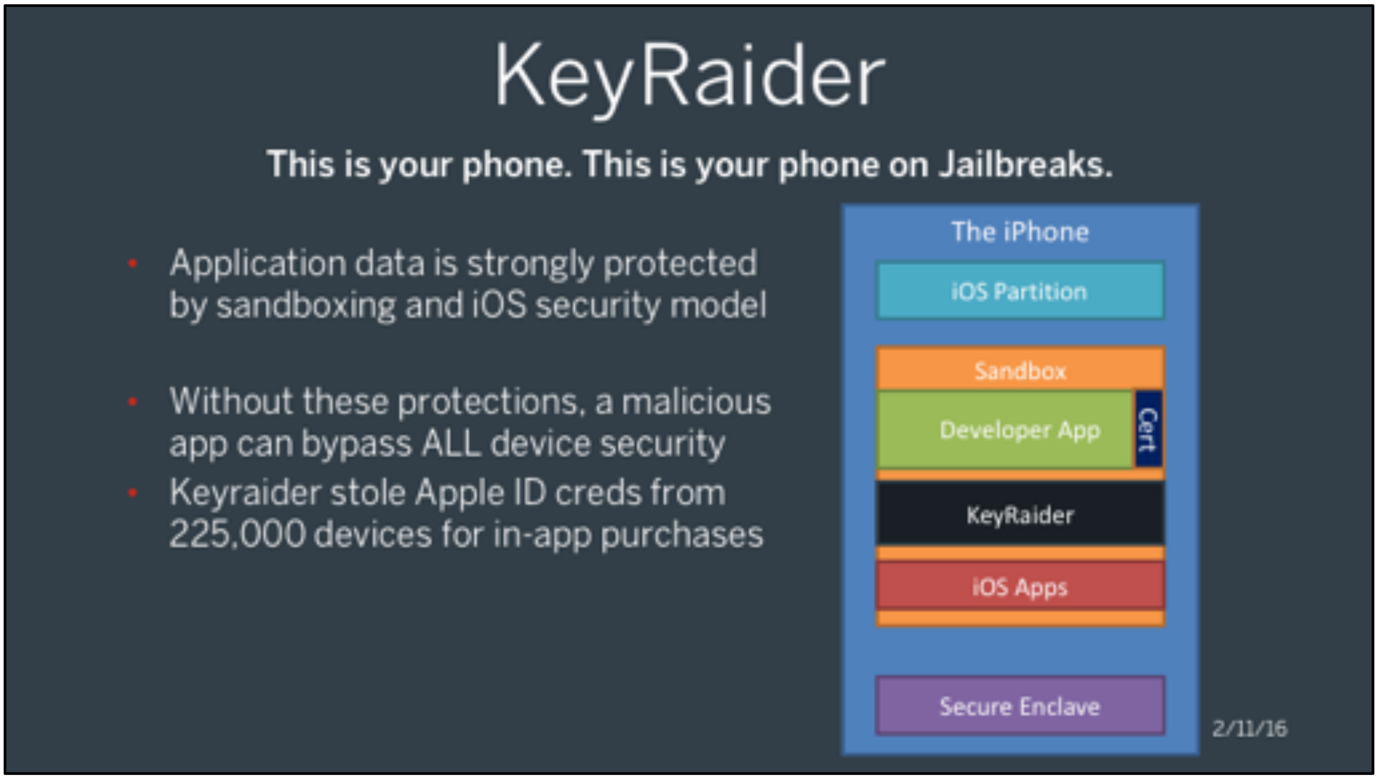
Why Jailbreak?

- Jailbreak maliciously through browser or app
- If you want to:
  1. Obtain application code
  2. Reverse an API Spec
  3. Steal copyrighted content
  4. Pirate paid apps
  5. Commit fraud

- Jailbreak voluntarily by downloading a kit
- If you want to:
  1. Access 3rd party appstores
  2. Tether for free
  3. Replace default apps
  4. Customize look and feel
  5. Device unlocking

The end result is the same. All application protections are dead.

20 2/11/16



KeyRaider

This is your phone. This is your phone on Jailbreaks.

- Application data is strongly protected by sandboxing and iOS security model
- Without these protections, a malicious app can bypass ALL device security
- Keyraider stole Apple ID creds from 225,000 devices for in-app purchases

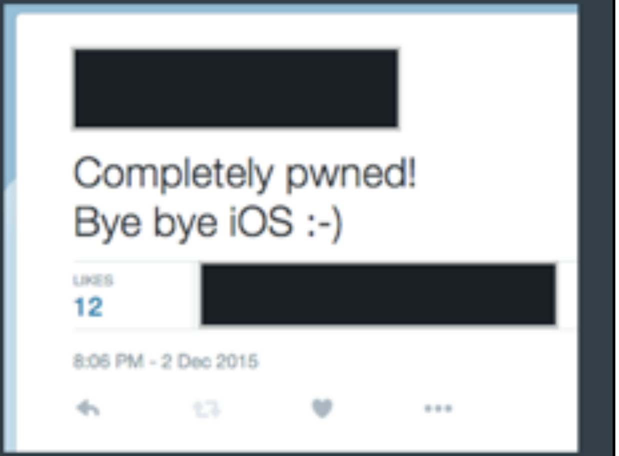
2/11/16

http://researchcenter.paloaltonetworks.com/2015/08/keyraider-ios-malware-steals-over-225000-apple-accounts-to-create-free-app-utopia/

Jailbreak Takeaways

You cannot always depend on Apple's APIs to protect you

1. Users will 'attack' themselves
   - Up to 7 million voluntarily jailbreak
2. Existing Jailbreaks used for attacks
   - Delivered via browser or user-generated app content
3. New Jailbreaks not always public
   - Active community of people, knowledge, tools for hacking iOS



[redacted]

Completely pwned!
Bye bye iOS :-)

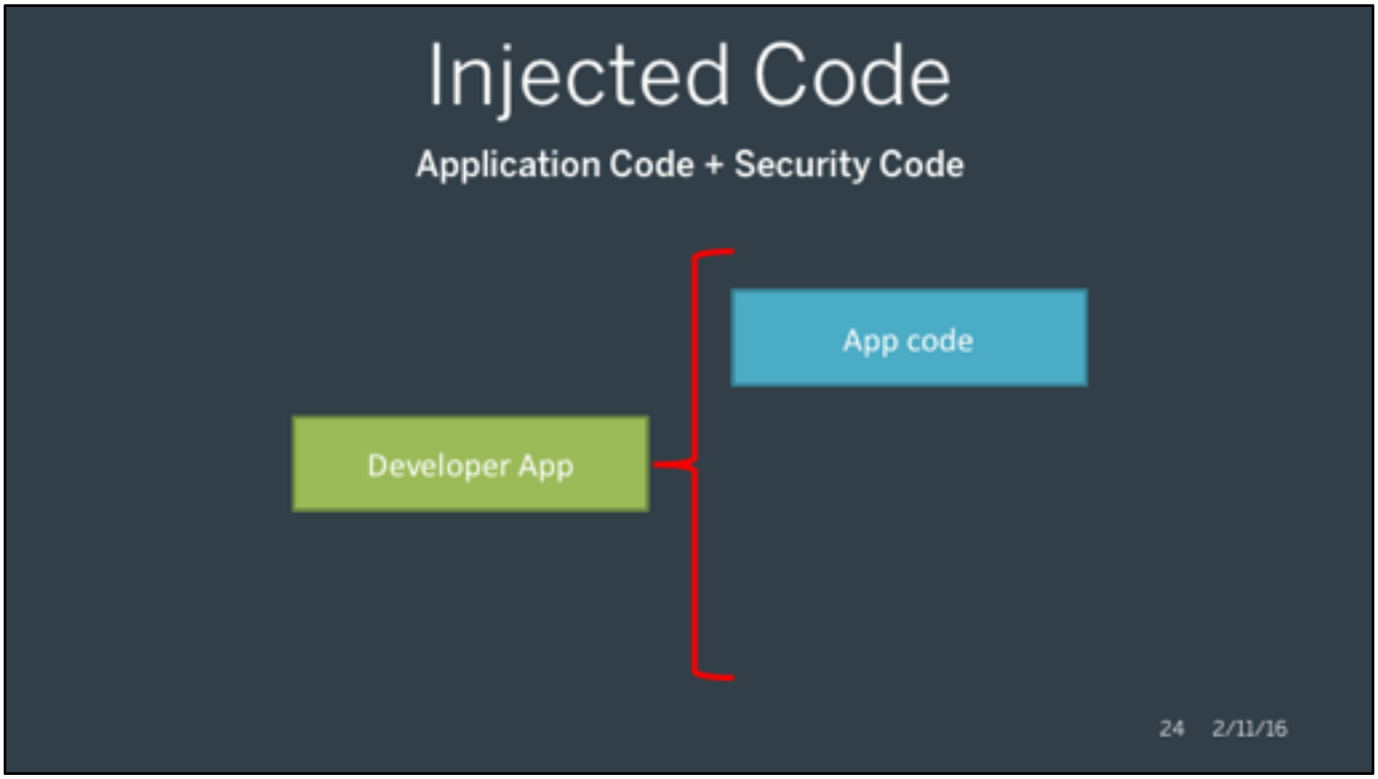
LIKES
12

[redacted]

8:06 PM - 2 Dec 2015

Jailbreak Protections

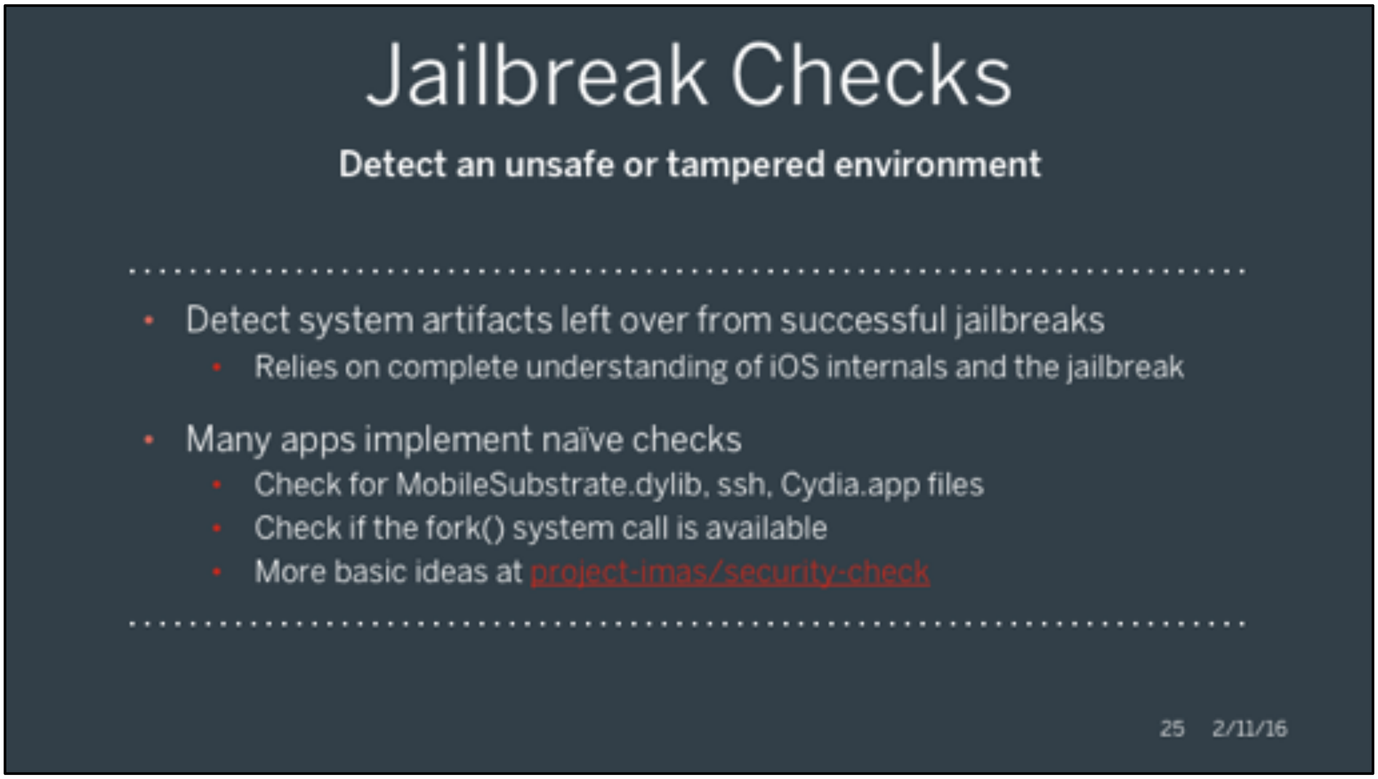
23 2/11/16



Injected Code

Application Code + Security Code

24 2/11/16

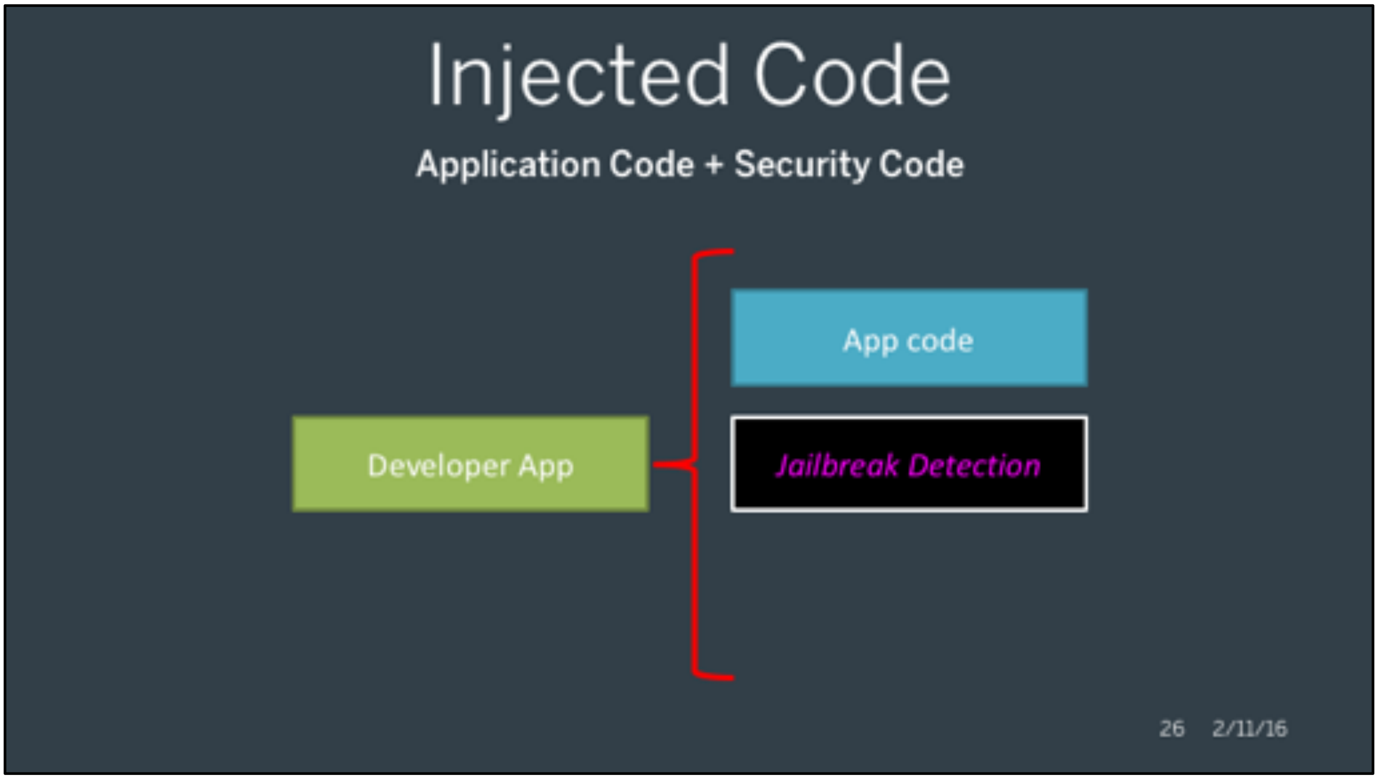


Jailbreak Checks

Detect an unsafe or tampered environment

- Detect system artifacts left over from successful jailbreaks
  - Relies on complete understanding of iOS internals and the jailbreak
- Many apps implement naïve checks
  - Check for MobileSubstrate.dylib, ssh, Cydia.app files
  - Check if the fork() system call is available
  - More basic ideas at project-imas/security-check

25 2/11/16



Injected Code

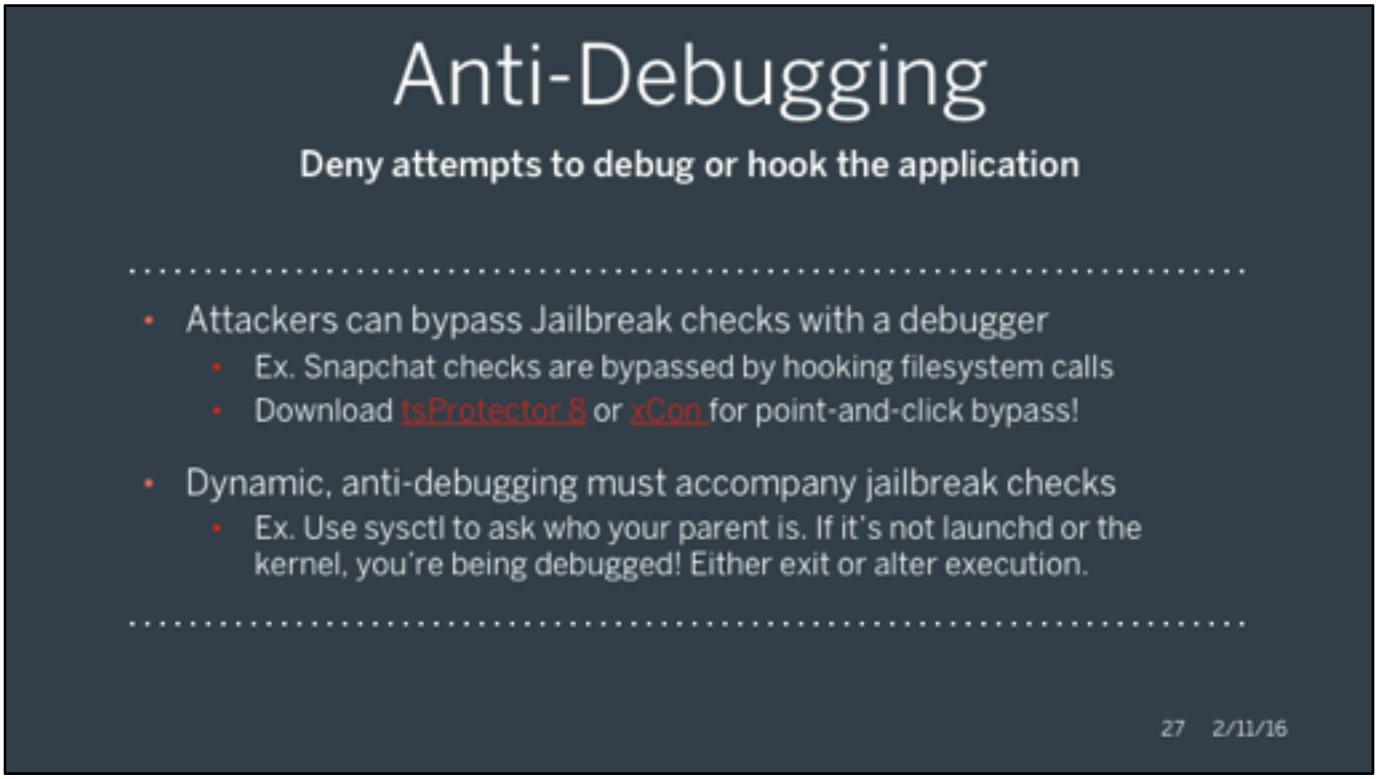
Application Code + Security Code

Developer App

App code

Jailbreak Detection

26 2/11/16

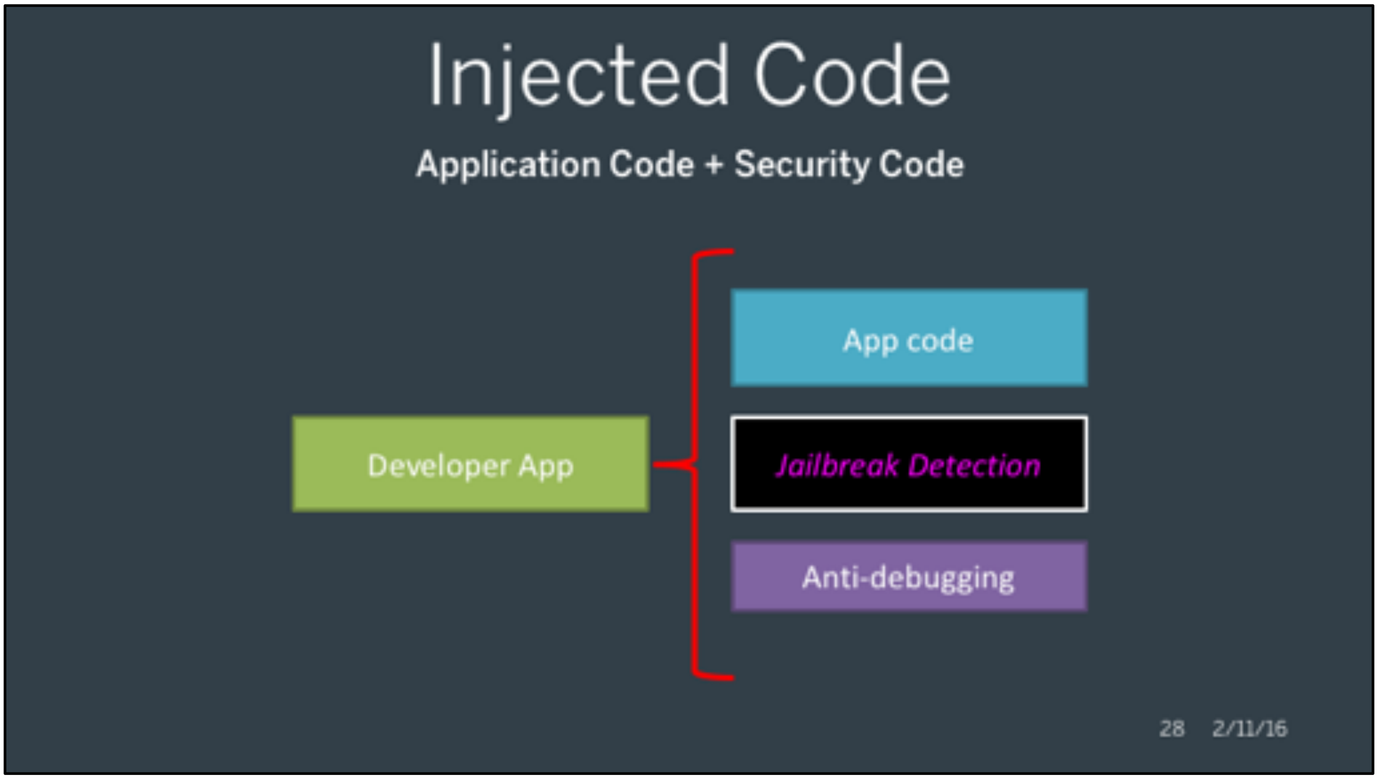


Anti-Debugging

Deny attempts to debug or hook the application

- Attackers can bypass Jailbreak checks with a debugger
  - Ex. Snapchat checks are bypassed by hooking filesystem calls
  - Download tsProtector 8 or xCon for point-and-click bypass!
- Dynamic, anti-debugging must accompany jailbreak checks
  - Ex. Use sysctl to ask who your parent is. If it's not launchd or the kernel, you're being debugged! Either exit or alter execution.

27 2/11/16



Injected Code

Application Code + Security Code

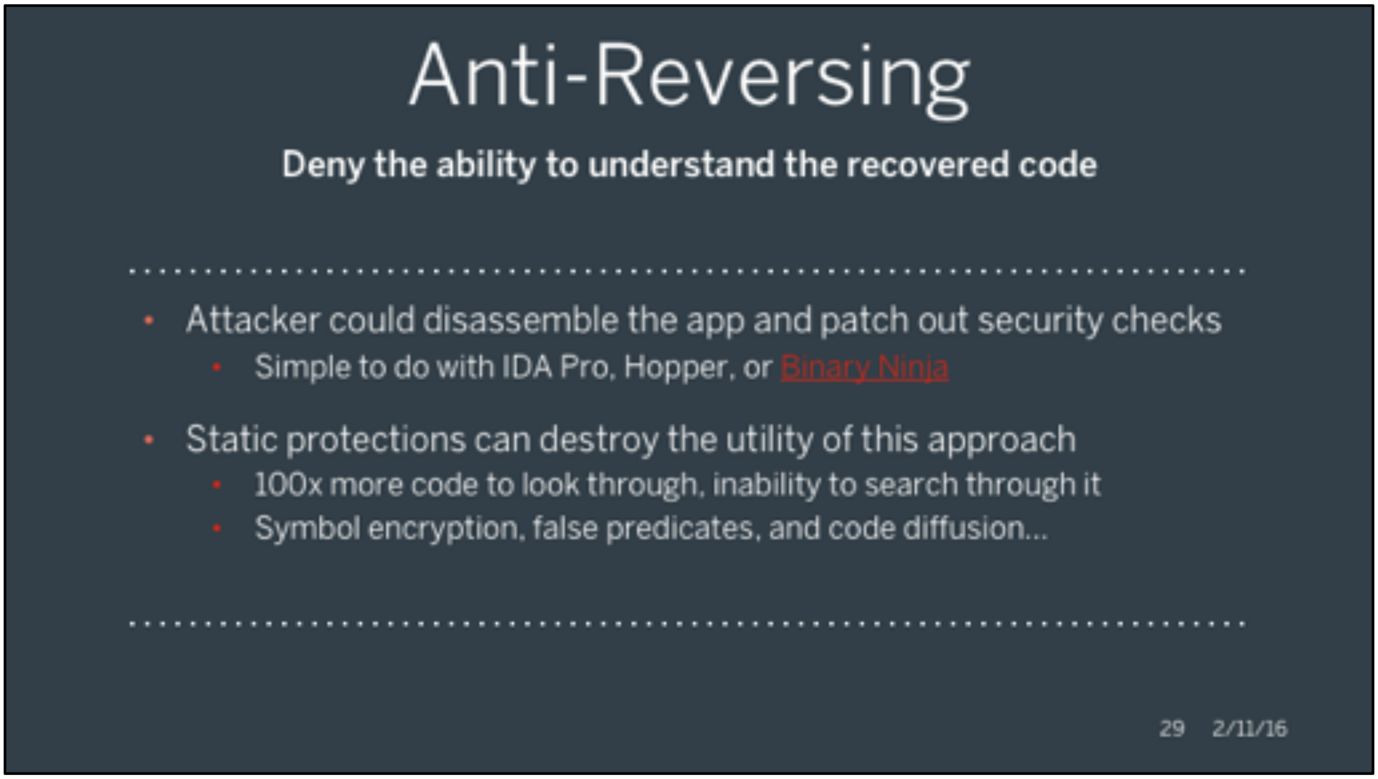
Developer App

App code

Jailbreak Detection

Anti-debugging

28 2/11/16



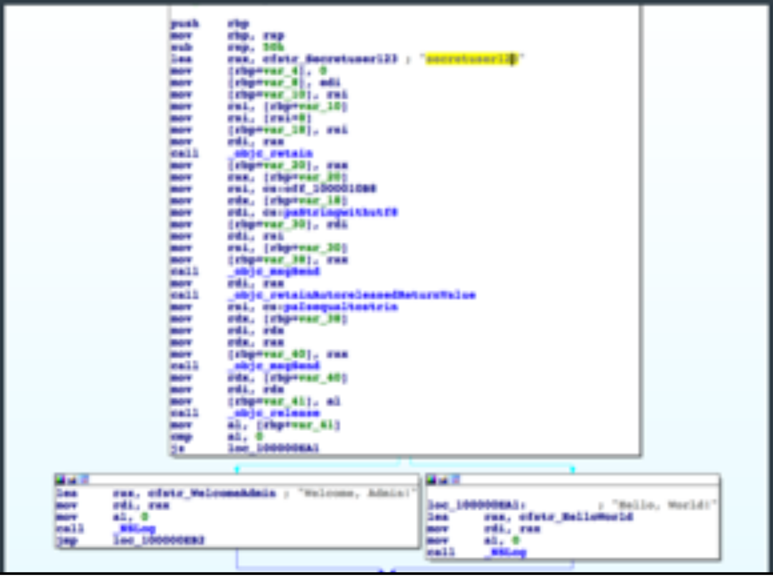
Anti-Reversing

Deny the ability to understand the recovered code

- Attacker could disassemble the app and patch out security checks
  - Simple to do with IDA Pro, Hopper, or Binary Ninja
- Static protections can destroy the utility of this approach
  - 100x more code to look through, inability to search through it
  - Symbol encryption, false predicates, and code diffusion...

Anti-Reversing

Original application as viewed in IDA Pro



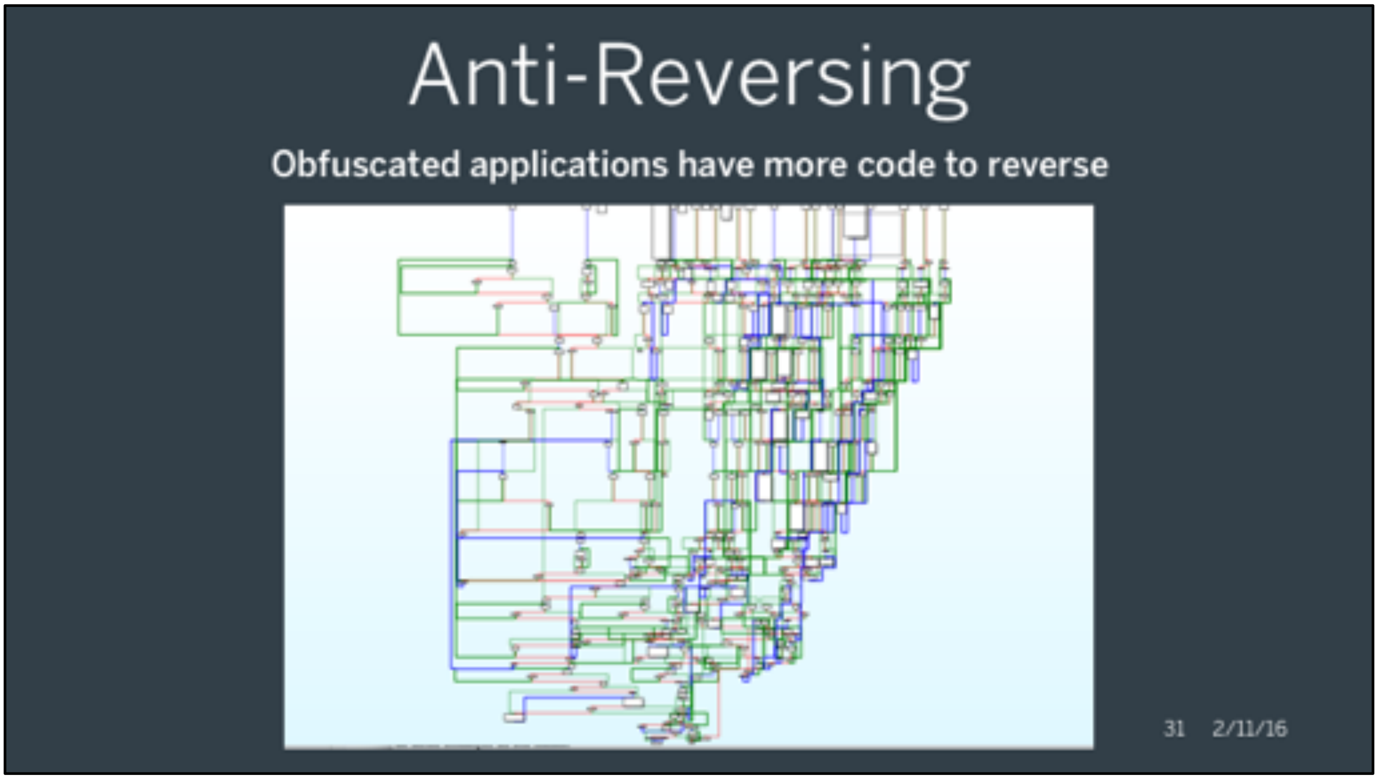
```
push    rbp
mov     rbp, rsp
sub     rsp, 50h
lea     rax, cfstr_Secretuser123 ; "secretuser123"
mov     [rbp+var_4], 0
mov     [rbp+var_8], edi
mov     [rbp+var_10], rsi
mov     rsi, [rbp+var_10]
mov     rsi, [rsi+8]
mov     [rbp+var_18], rsi
mov     rdi, rax
call    _objc_retain
mov     [rbp+var_20], rax
mov     rax, [rbp+var_20]
mov     rsi, cs:off_100001088
mov     rdx, [rbp+var_18]
mov     rdi, cs:paStringwithutf8
mov     [rbp+var_30], rdi
mov     rdi, rsi
mov     rsi, [rbp+var_30]
mov     [rbp+var_38], rax
call    _objc_msgSend
mov     rdi, rax
call    _objc_retainAutoreleasedReturnValue
mov     rsi, cs:paIsequaltostrin
mov     rdx, [rbp+var_38]
mov     rdi, rdx
mov     rdx, rax
mov     [rbp+var_40], rax
call    _objc_msgSend
mov     rdx, [rbp+var_40]
mov     rdi, rdx
mov     [rbp+var_41], al
call    _objc_release
mov     al, [rbp+var_41]
cmp     al, 0
jz      loc_100000EA1
```

```
lea     rax, cfstr_WelcomeAdmin ; "Welcome, Admin!"
mov     rdi, rax
mov     al, 0
call    _NSLog
jmp     loc_100000EB2
```

```
loc_100000EA1:                  ; "Hello, World!"
lea     rax, cfstr_HelloWorld
mov     rdi, rax
mov     al, 0
call    _NSLog
```

30 2/11/16

Further explain symbol encryption, predicate insertion, and code diffusion aided by a graphic
Performance



Anti-Reversing

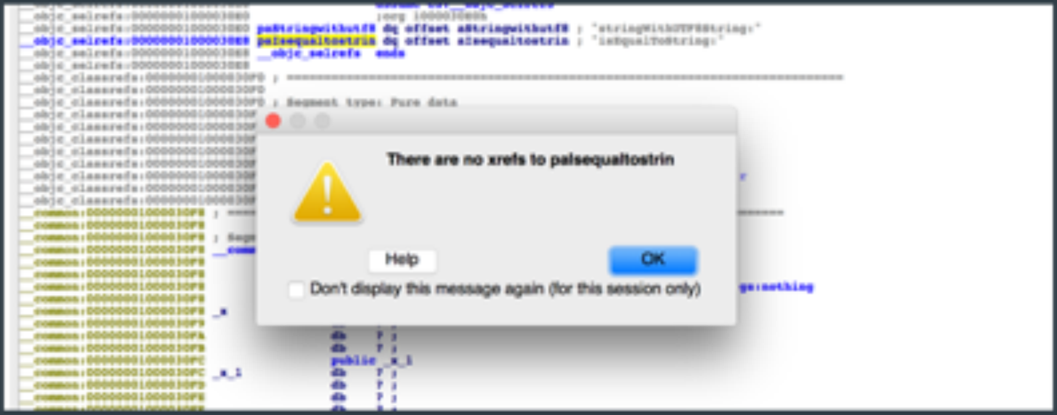
Obfuscated applications have more code to reverse

31 2/11/16

Further explain symbol encryption, predicate insertion, and code diffusion aided by a graphic
Performance

Anti-Reversing

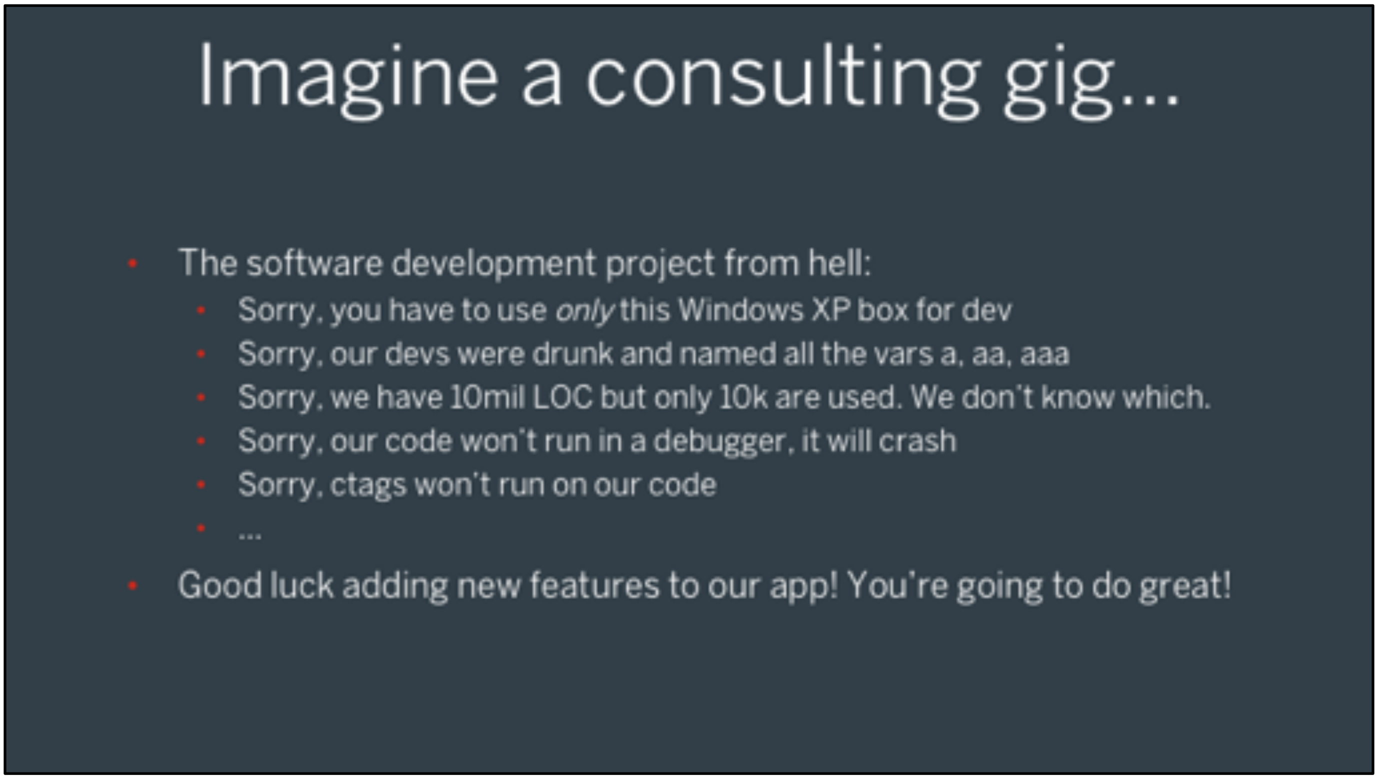
Symbol encryption makes binary code navigation impossible



There are no xrefs to palsequaltostrin

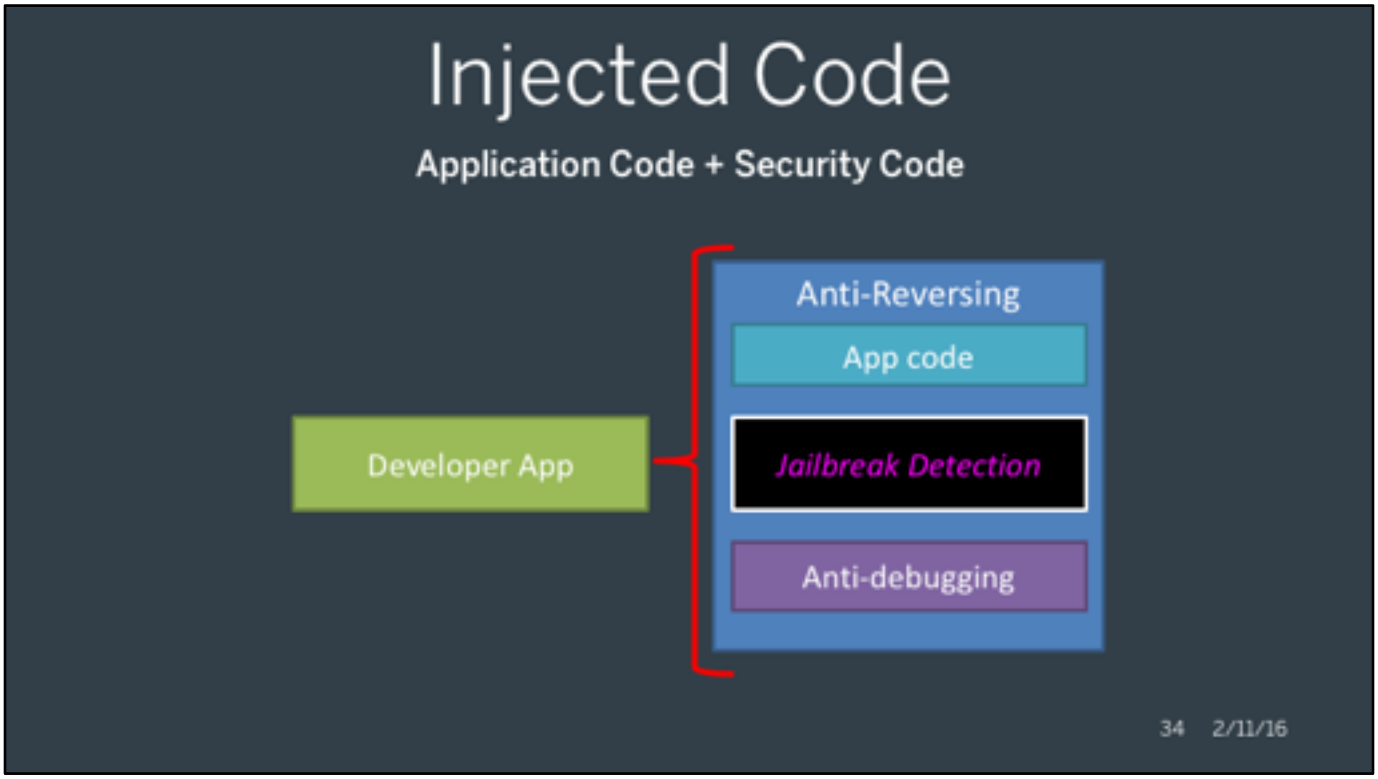
☐ Don't display this message again (for this session only)

Further explain symbol encryption, predicate insertion, and code diffusion aided by a graphic
Performance



Imagine a consulting gig…

- The software development project from hell:
  - Sorry, you have to use *only* this Windows XP box for dev
  - Sorry, our devs were drunk and named all the vars a, aa, aaa
  - Sorry, we have 10mil LOC but only 10k are used. We don't know which.
  - Sorry, our code won't run in a debugger, it will crash
  - Sorry, ctags won't run on our code
  - …
- Good luck adding new features to our app! You're going to do great!



Injected Code

Application Code + Security Code

Developer App

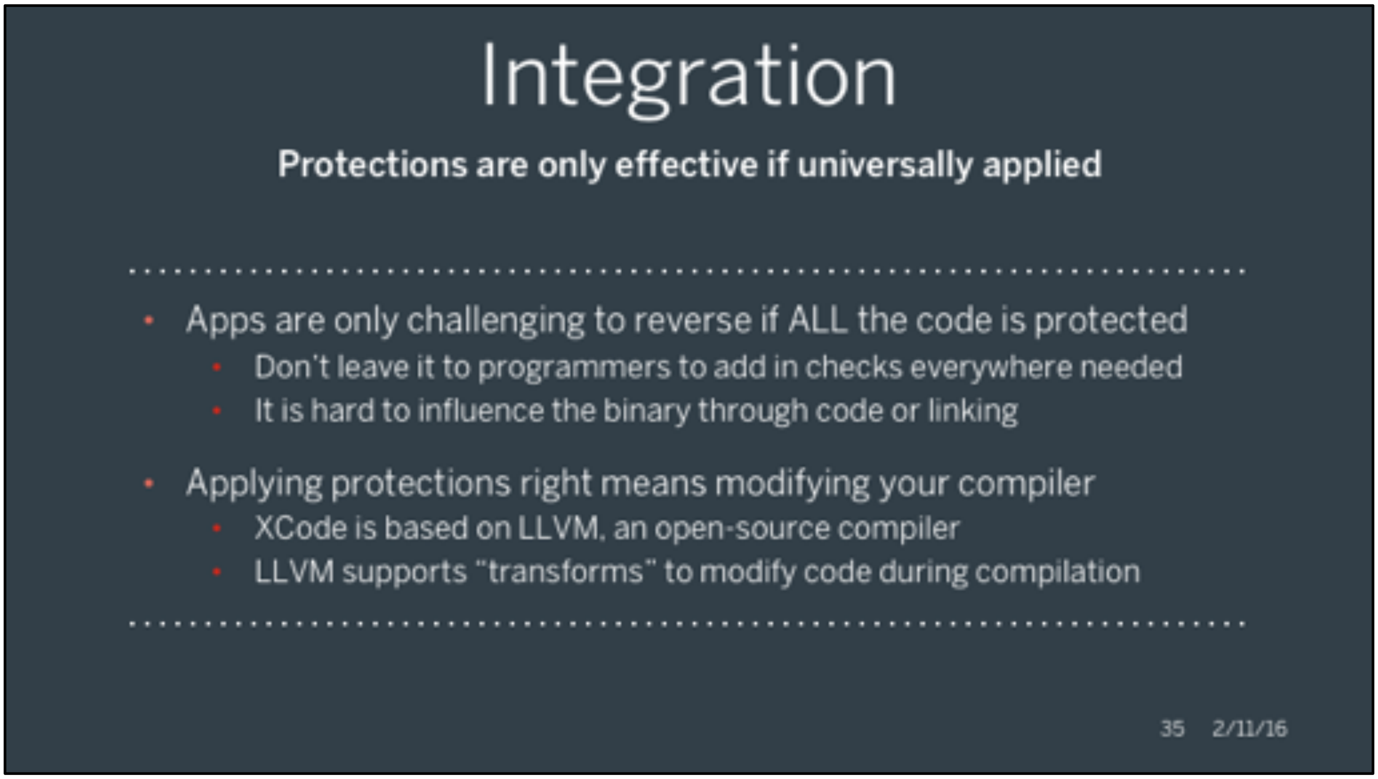
Anti-Reversing

App code

Jailbreak Detection

Anti-debugging

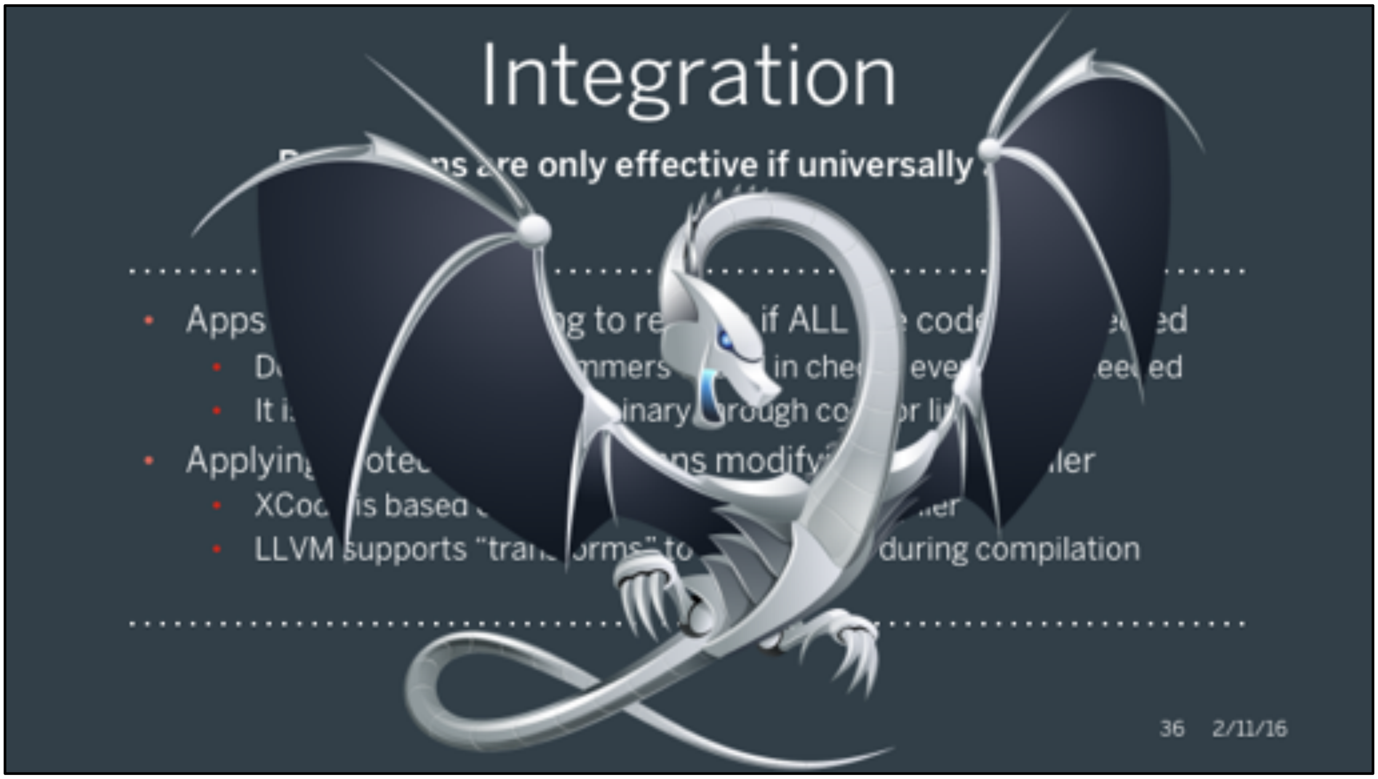
34 2/11/16



Integration

Protections are only effective if universally applied

- Apps are only challenging to reverse if ALL the code is protected
  - Don’t leave it to programmers to add in checks everywhere needed
  - It is hard to influence the binary through code or linking
- Applying protections right means modifying your compiler
  - XCode is based on LLVM, an open-source compiler
  - LLVM supports “transforms” to modify code during compilation

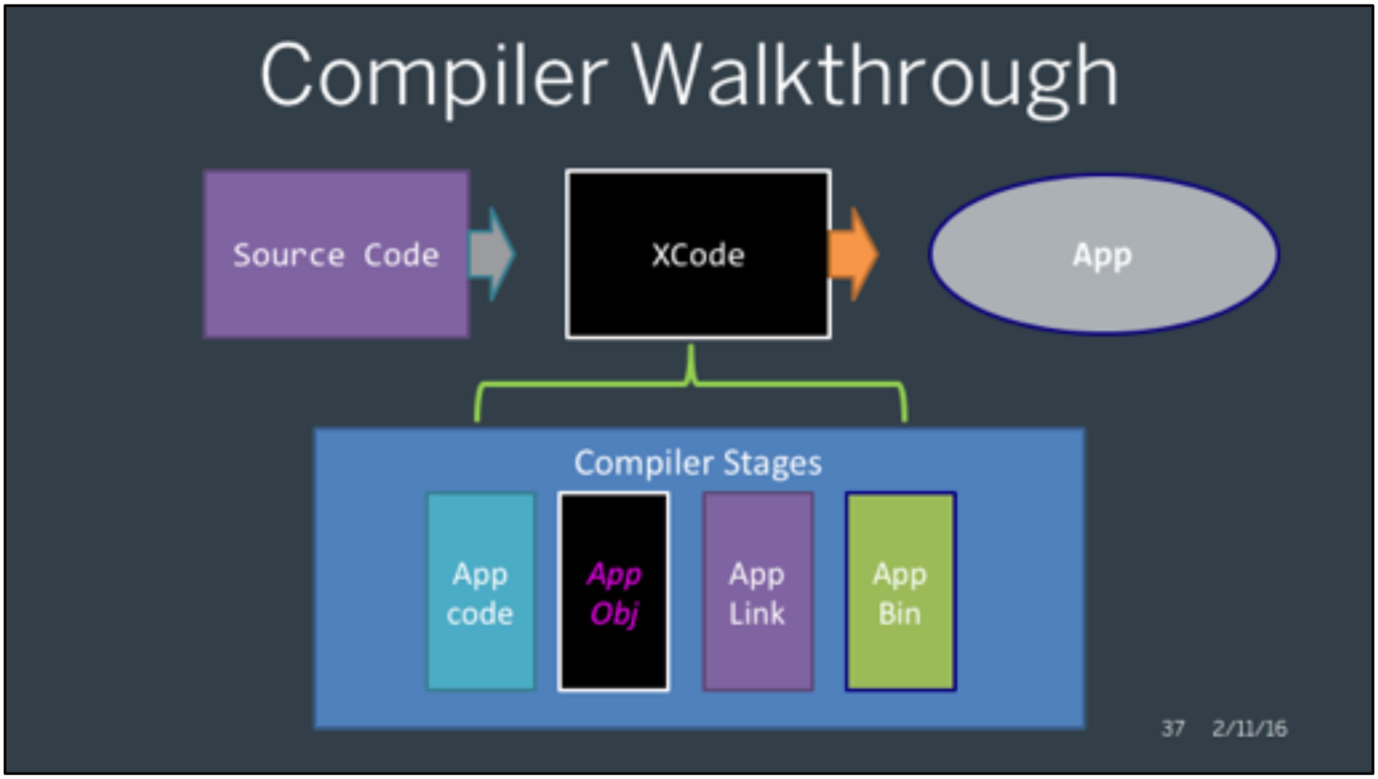


Integration

[illegible]ns are only effective if universally [illegible]

- Apps [illegible]g to re[illegible] if ALL [illegible]e code[illegible]eded
  - D[illegible]mmers [illegible] in che[illegible] eve[illegible]eeded
  - It i[illegible]inary [illegible]rough co[illegible]or li[illegible]
- Applying [illegible]otec[illegible]ns modify[illegible]iler
  - XCod[illegible] is based [illegible]ler
  - LLVM supports “transforms” to [illegible] during compilation

36 2/11/16



Compiler Walkthrough

Source Code

XCode

App

Compiler Stages

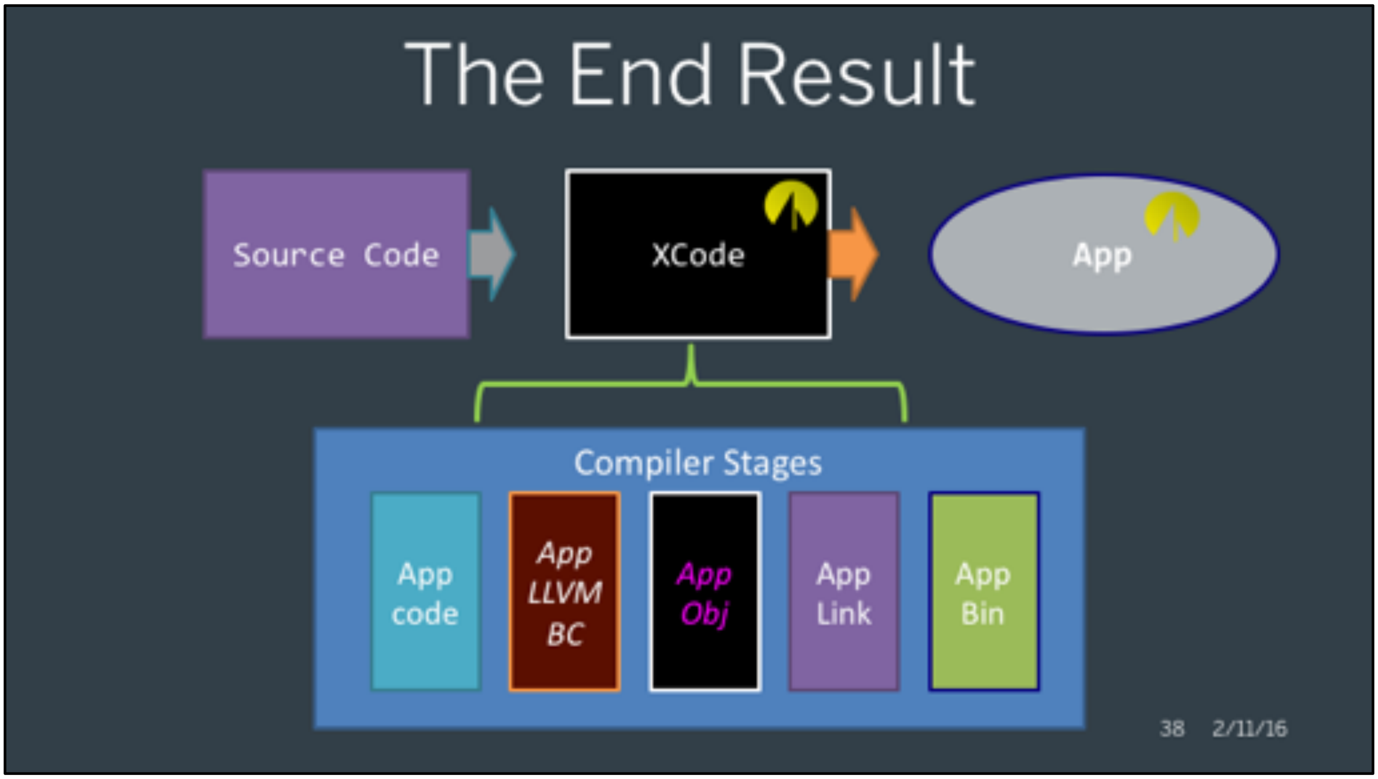
App code

App Obj

App Link

App Bin

37 2/11/16



The End Result

Source Code

XCode

App

Compiler Stages

App code

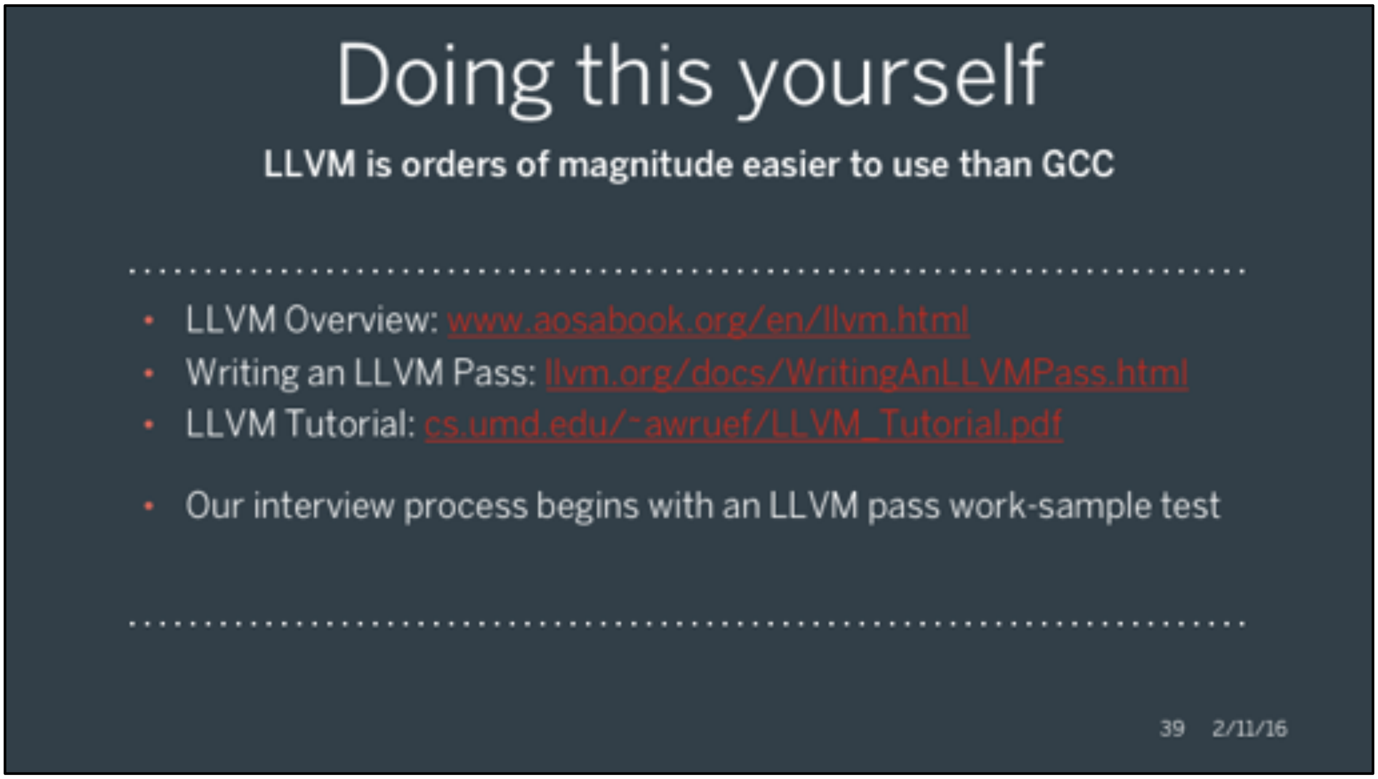
App LLVM BC

App Obj

App Link

App Bin

38 2/11/16



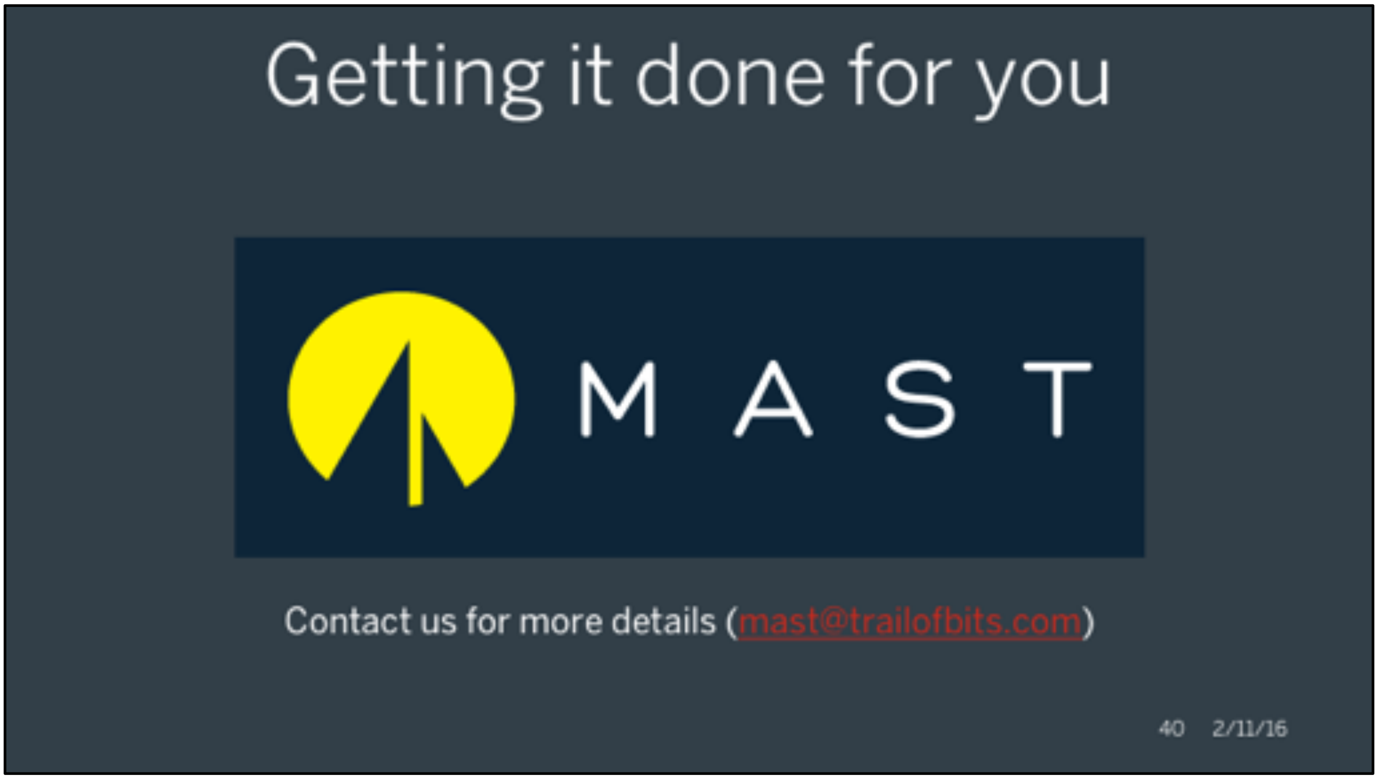
Doing this yourself

LLVM is orders of magnitude easier to use than GCC

- LLVM Overview: www.aosabook.org/en/llvm.html
- Writing an LLVM Pass: llvm.org/docs/WritingAnLLVMPass.html
- LLVM Tutorial: cs.umd.edu/~awruef/LLVM_Tutorial.pdf
- Our interview process begins with an LLVM pass work-sample test

39 2/11/16

Alex w/ GCC VulnCheck vs InternChallenge w/ LLVM



Getting it done for you

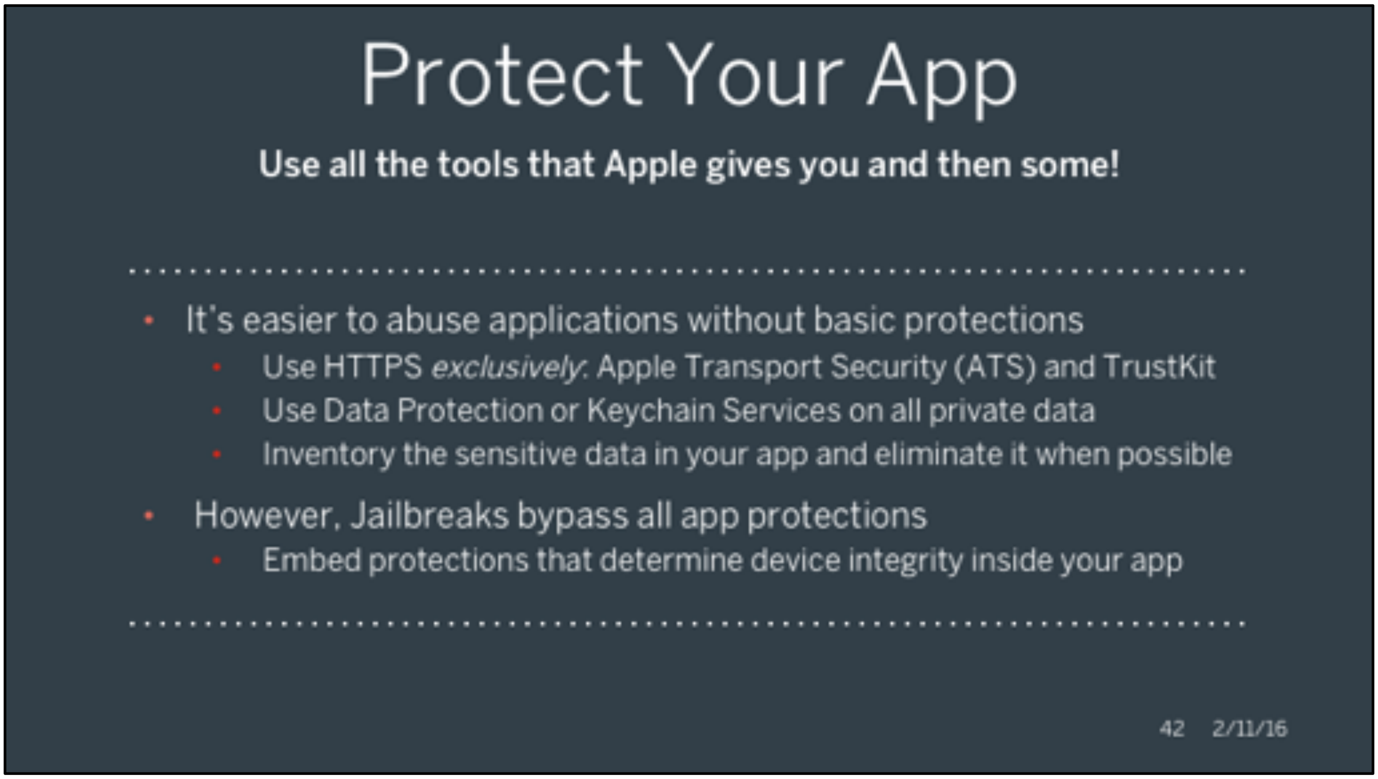
MAST

Contact us for more details (mast@trailofbits.com)

Alex w/ GCC VulnCheck vs InternChallenge w/ LLVM

Conclusions

41 2/11/16



Protect Your App

Use all the tools that Apple gives you and then some!

- It's easier to abuse applications without basic protections
  - Use HTTPS *exclusively*: Apple Transport Security (ATS) and TrustKit
  - Use Data Protection or Keychain Services on all private data
  - Inventory the sensitive data in your app and eliminate it when possible
- However, Jailbreaks bypass all app protections
  - Embed protections that determine device integrity inside your app

42 2/11/16

Questions?

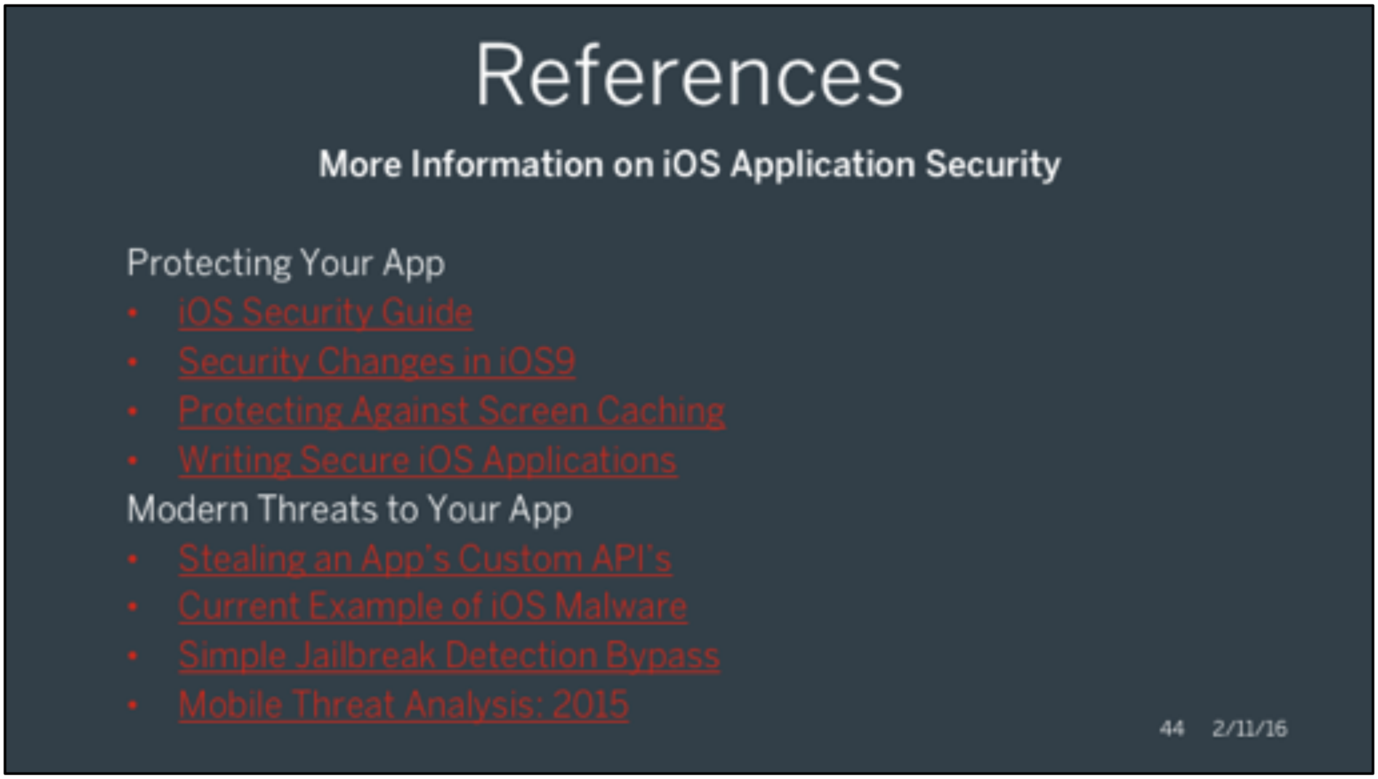
Thanks for having us!

- If you like this kind of work, come to Empire Hacking
- If you have high security needs, talk with us about MAST
- If auth is more your thing, www.passwordlessapps.com

Dan Guido
CEO
dan@trailofbits.com
@dguido

Sophia D'Antoine
Security Engineer
sophia@trailofbits.com
quend@freenode

43 2/11/16



References

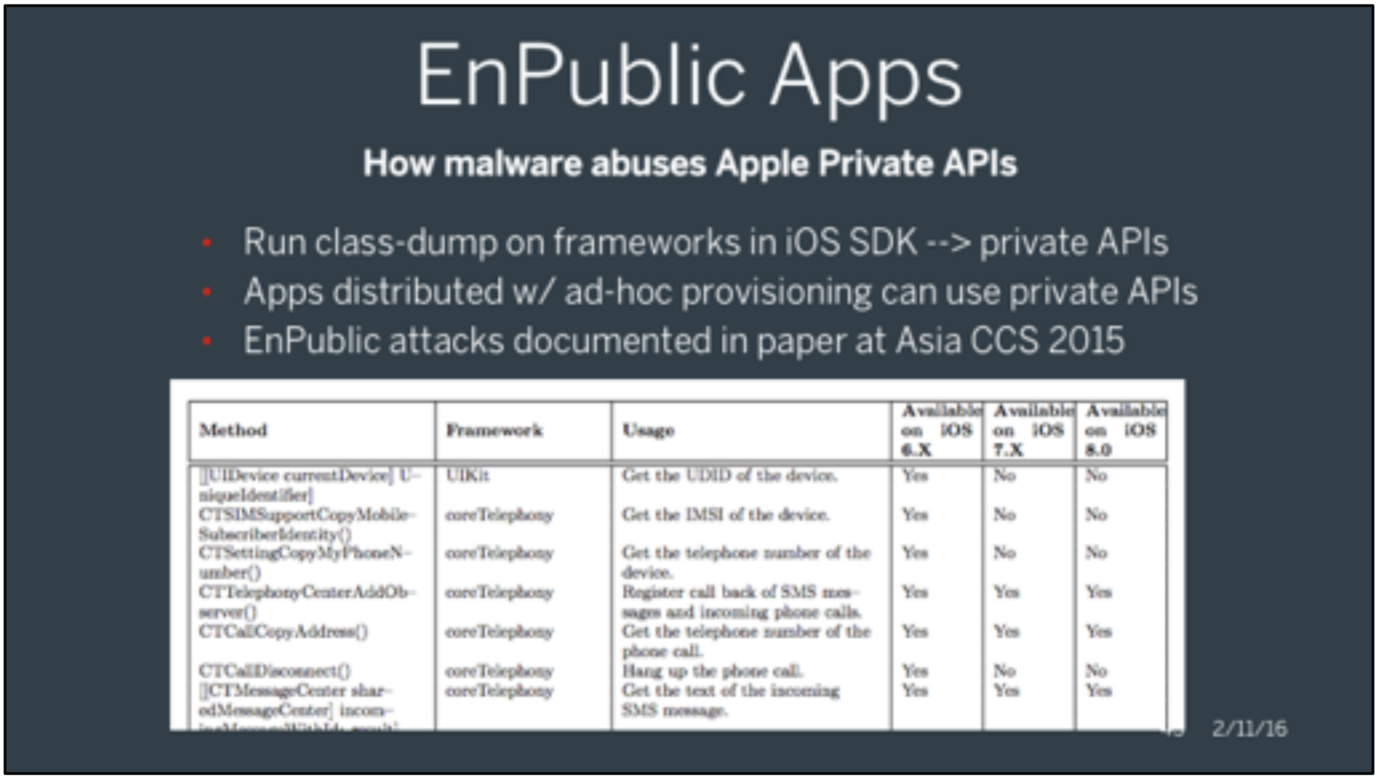
More Information on iOS Application Security

Protecting Your App

- iOS Security Guide
- Security Changes in iOS9
- Protecting Against Screen Caching
- Writing Secure iOS Applications

Modern Threats to Your App

- Stealing an App's Custom API's
- Current Example of iOS Malware
- Simple Jailbreak Detection Bypass
- Mobile Threat Analysis: 2015



EnPublic Apps

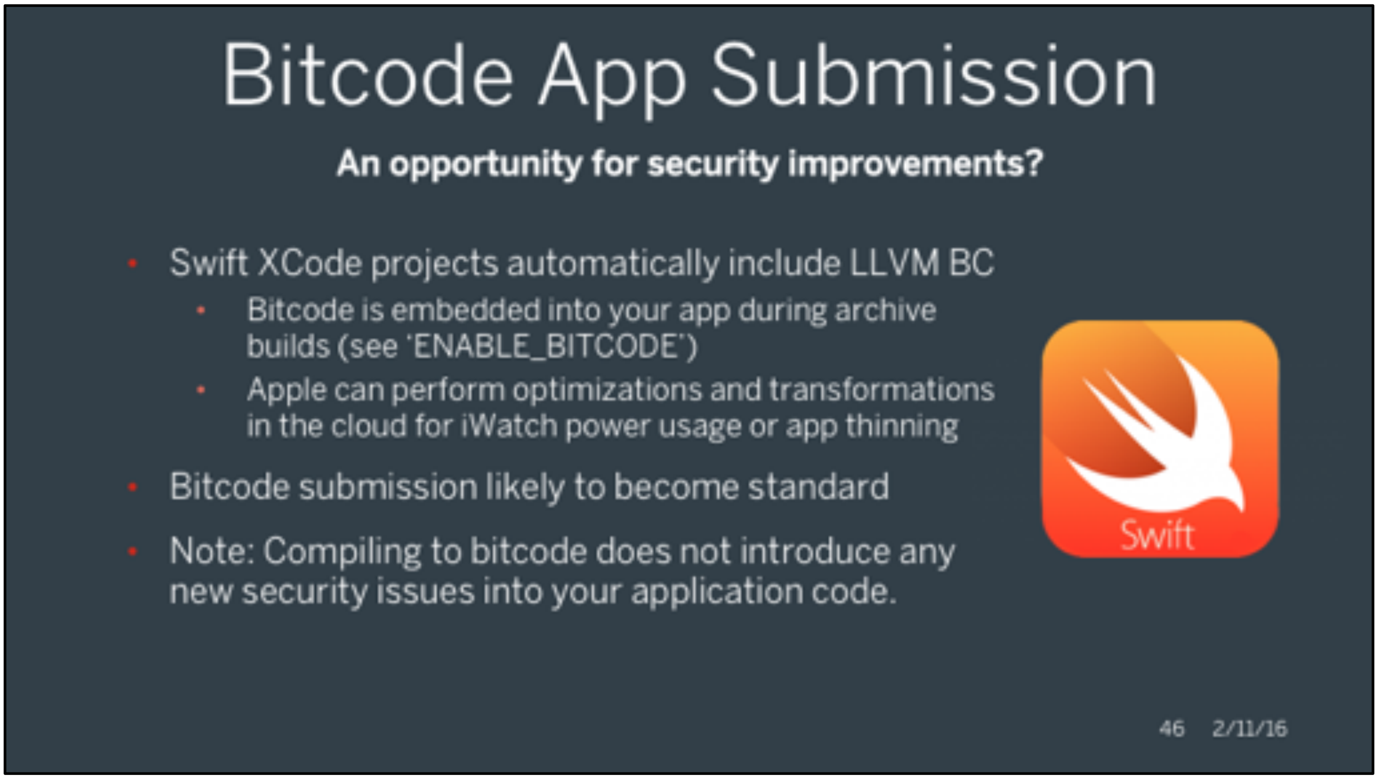
How malware abuses Apple Private APIs

- Run class-dump on frameworks in iOS SDK --> private APIs
- Apps distributed w/ ad-hoc provisioning can use private APIs
- EnPublic attacks documented in paper at Asia CCS 2015

Method	Framework	Usage	Available on iOS 6.X	Available on iOS 7.X	Available on iOS 8.0
[[UIDevice currentDevice] UniqueIdentifier]	UIKit	Get the UDID of the device.	Yes	No	No
CTSIMSupportCopyMobileSubscriberIdentity()	coreTelephony	Get the IMSI of the device.	Yes	No	No
CTSettingCopyMyPhoneNumber()	coreTelephony	Get the telephone number of the device.	Yes	No	No
CTTelephonyCenterAddObserver()	coreTelephony	Register call back of SMS messages and incoming phone calls.	Yes	Yes	Yes
CTCallCopyAddress()	coreTelephony	Get the telephone number of the phone call.	Yes	Yes	Yes
CTCallDisconnect()	coreTelephony	Hang up the phone call.	Yes	No	No
[[CTMessageCenter sharedMessageCenter] incom	coreTelephony	Get the text of the incoming SMS message.	Yes	Yes	Yes

2/11/16

http://www.cse.cuhk.edu.hk/~cslui/PUBLICATION/ASIACCS15.pdf
Masque attack vulns patched out in 8.x series
https://www.fireeye.com/blog/threat-research/2015/06/three_new_masqueatt.html



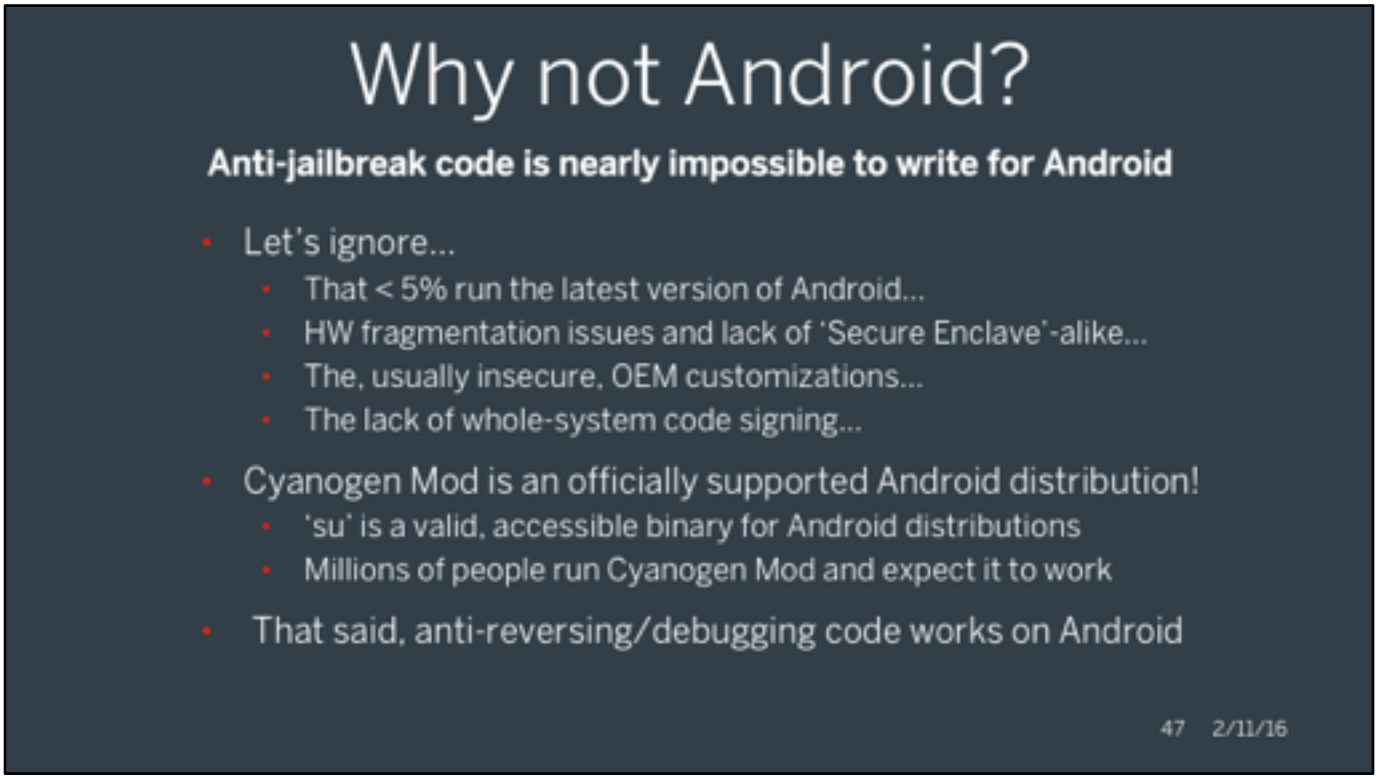
Bitcode App Submission

An opportunity for security improvements?

- Swift XCode projects automatically include LLVM BC
  - Bitcode is embedded into your app during archive builds (see 'ENABLE_BITCODE')
  - Apple can perform optimizations and transformations in the cloud for iWatch power usage or app thinning
- Bitcode submission likely to become standard
- Note: Compiling to bitcode does not introduce any new security issues into your application code.

46 2/11/16

https://developer.apple.com/library/tvos/documentation/IDEs/Conceptual/AppDistributionGuide/AppThinning/AppThinning.html#//apple_ref/doc/uid/TP40012582-CH35-SW2

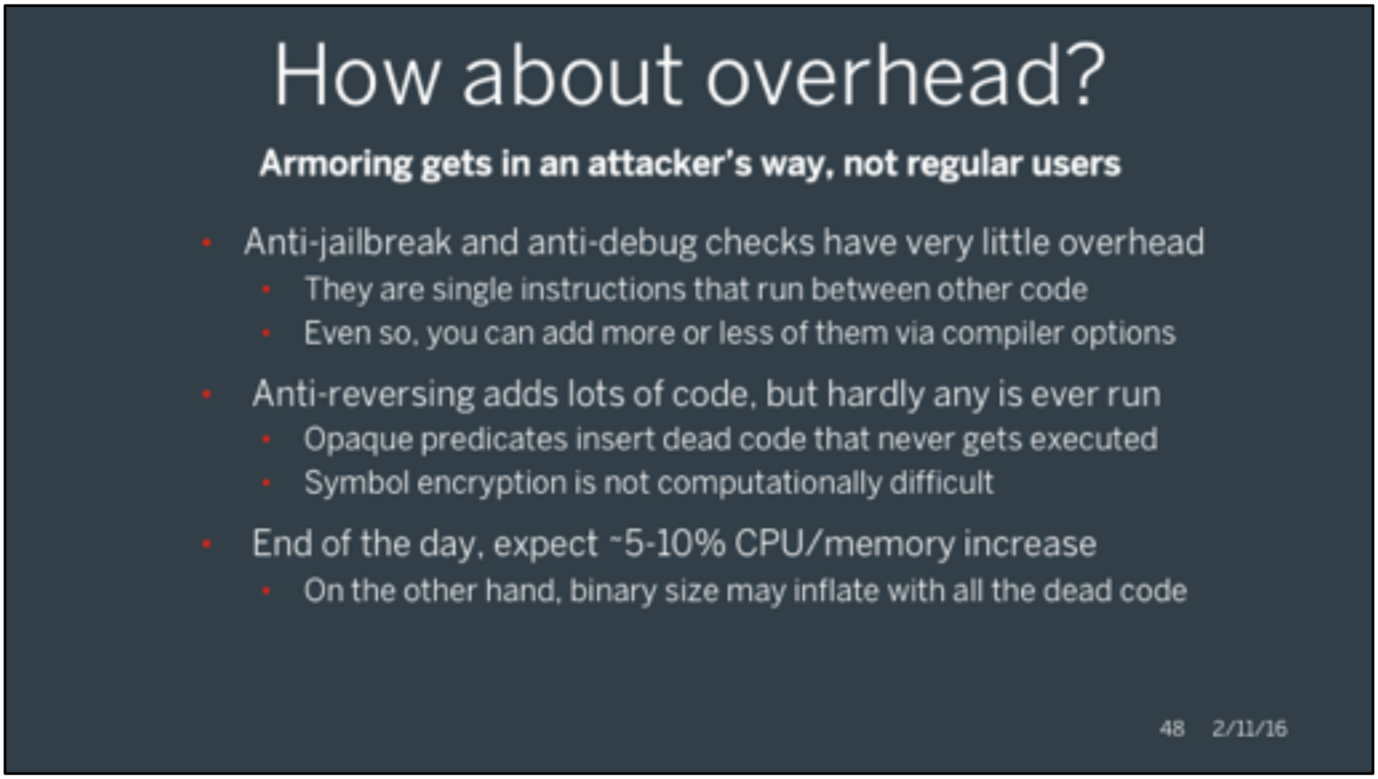


Why not Android?

Anti-jailbreak code is nearly impossible to write for Android

- Let's ignore…
  - That < 5% run the latest version of Android…
  - HW fragmentation issues and lack of 'Secure Enclave'-alike…
  - The, usually insecure, OEM customizations…
  - The lack of whole-system code signing…
- Cyanogen Mod is an officially supported Android distribution!
  - 'su' is a valid, accessible binary for Android distributions
  - Millions of people run Cyanogen Mod and expect it to work
- That said, anti-reversing/debugging code works on Android

47 2/11/16



How about overhead?

Armoring gets in an attacker's way, not regular users

- Anti-jailbreak and anti-debug checks have very little overhead
  - They are single instructions that run between other code
  - Even so, you can add more or less of them via compiler options
- Anti-reversing adds lots of code, but hardly any is ever run
  - Opaque predicates insert dead code that never gets executed
  - Symbol encryption is not computationally difficult
- End of the day, expect ~5-10% CPU/memory increase
  - On the other hand, binary size may inflate with all the dead code