TrustKit

Code Injection on iOS 8 for the Greater Good

Alban Diquet - @nabla_c0d3
Angela Chow - @paranoid_angela
Eric Castro - @_eric_castro

black hat USA 2015

About Us

- Alban: Engineering/security lead at Data Theorem
- Eric: iOS R&D at Data Theorem
- Angela: Paranoids (security) at Yahoo

Agenda

- TrustKit: effortless SSL pinning for iOS and OS X
- Dynamic libraries and iOS 8
- Function hooking on a non-jailbroken device

Agenda

- TrustKit: effortless SSL pinning for iOS and OS X
- Dynamic libraries and iOS 8
- Function hooking on a non-jailbroken device

TrustKit

- Goal: Create an SSL pinning library for iOS
- Needed a usable solution that **works in real-world Apps**
- Collaborated with the Yahoo mobile & security teams

SSL Pinning at Yahoo

- Goal: SSL pinning for Yahoo’s mobile Apps
  - Easy project, right?

SSL Pinning at Yahoo

- Goal: SSL pinning for Yahoo's mobile Apps
    - Easy project, right?
- But...
    - Technical challenges: What and how to pin?
    - Operational challenges: How to get buy-in from product team?

Technical Challenges

- What to pin?
    - Certificate or public key?
        - Best practice is Subject Public Key Info
            - No API on iOS to extract SPKI from a certificate…
- Most libraries and examples are doing it wrong
    - Comparing the whole certificate or public key

Technical Challenges

- How to pin?
  - Find and modify every single instance of *NSURLConnection*, *NSURLSession* ?
    - Or better: use method swizzling
  - Problem: no public API for customizing certificate validation in *UIWebView*
    - Not even swizzling would work

Operational Challenges

- How to get buy-in from the product team?
  - Blocking attackers is a good cause but...

Operational Challenges

- How to get buy-in from the product team?
  - Blocking attackers is a good cause but...
    - What if we block the wrong connections?

Operational Challenges

- How to get buy-in from the product team?
  - Blocking attackers is a good cause but...
    - What if we block the wrong connections?
- Answer: a **report-only** mode
  - Shows what connections would be blocked and why
  - Easier to decide on whether pinning should be enforced or not

SSL Pinning at Yahoo

- No existing iOS library supported **any** of these requirements
  - SPKI pinning
  - Report-only mode
  - Easy to deploy but works on all networking APIs
- Met with Data Theorem and started a collaboration :)

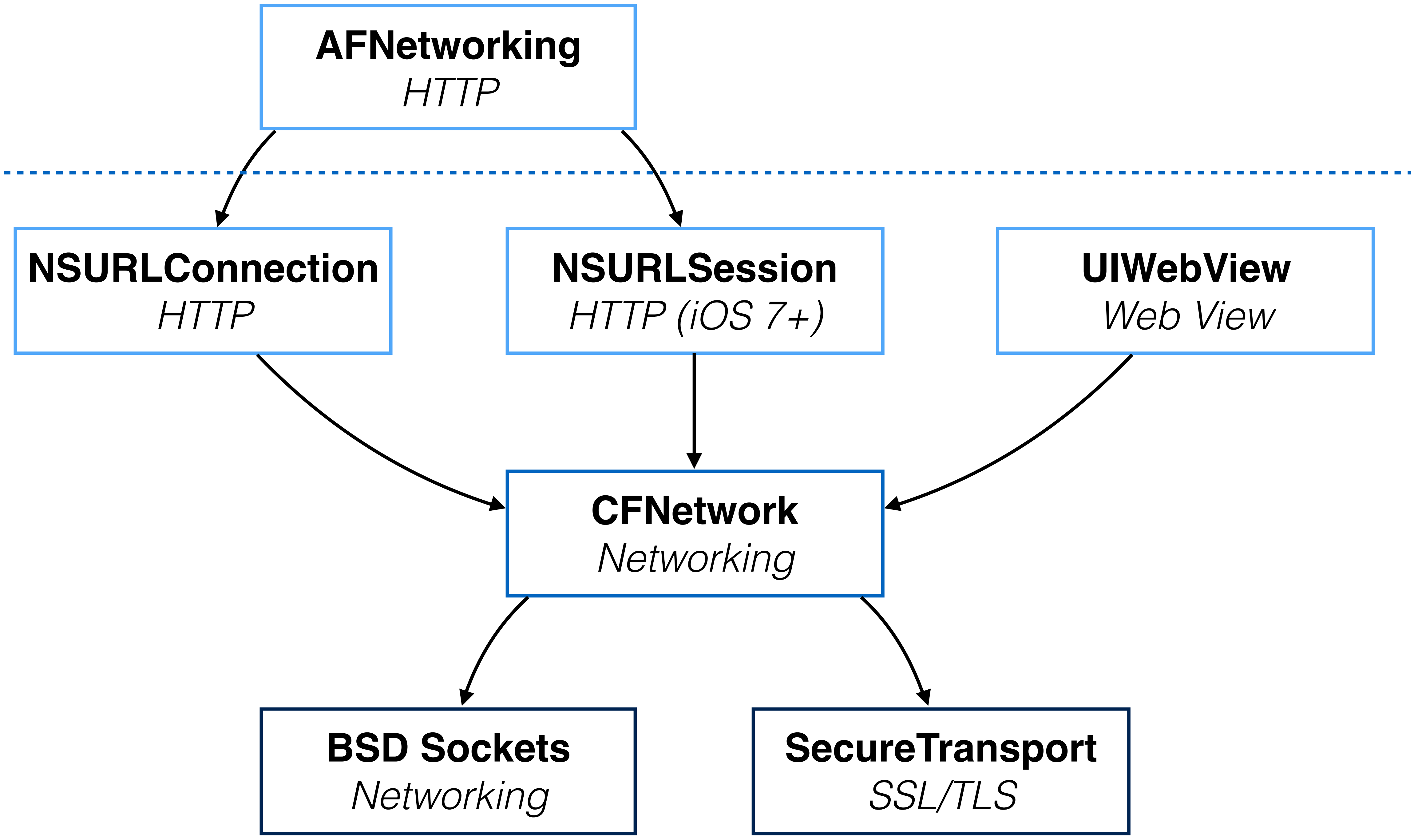
TrustKit

- We solved these challenges

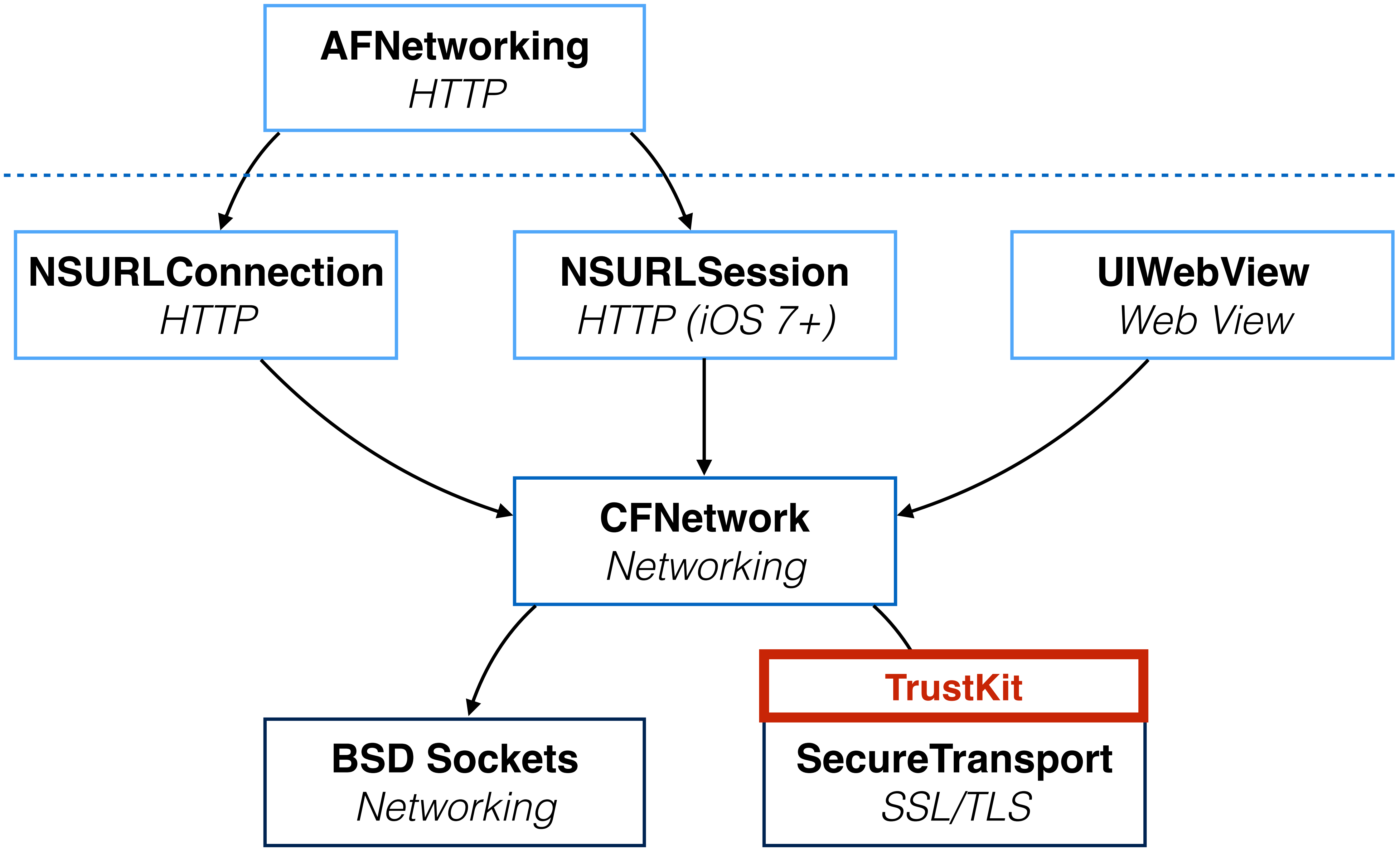
TrustKit

- We solved these challenges
    - TrustKit works transparently on all Apple APIs

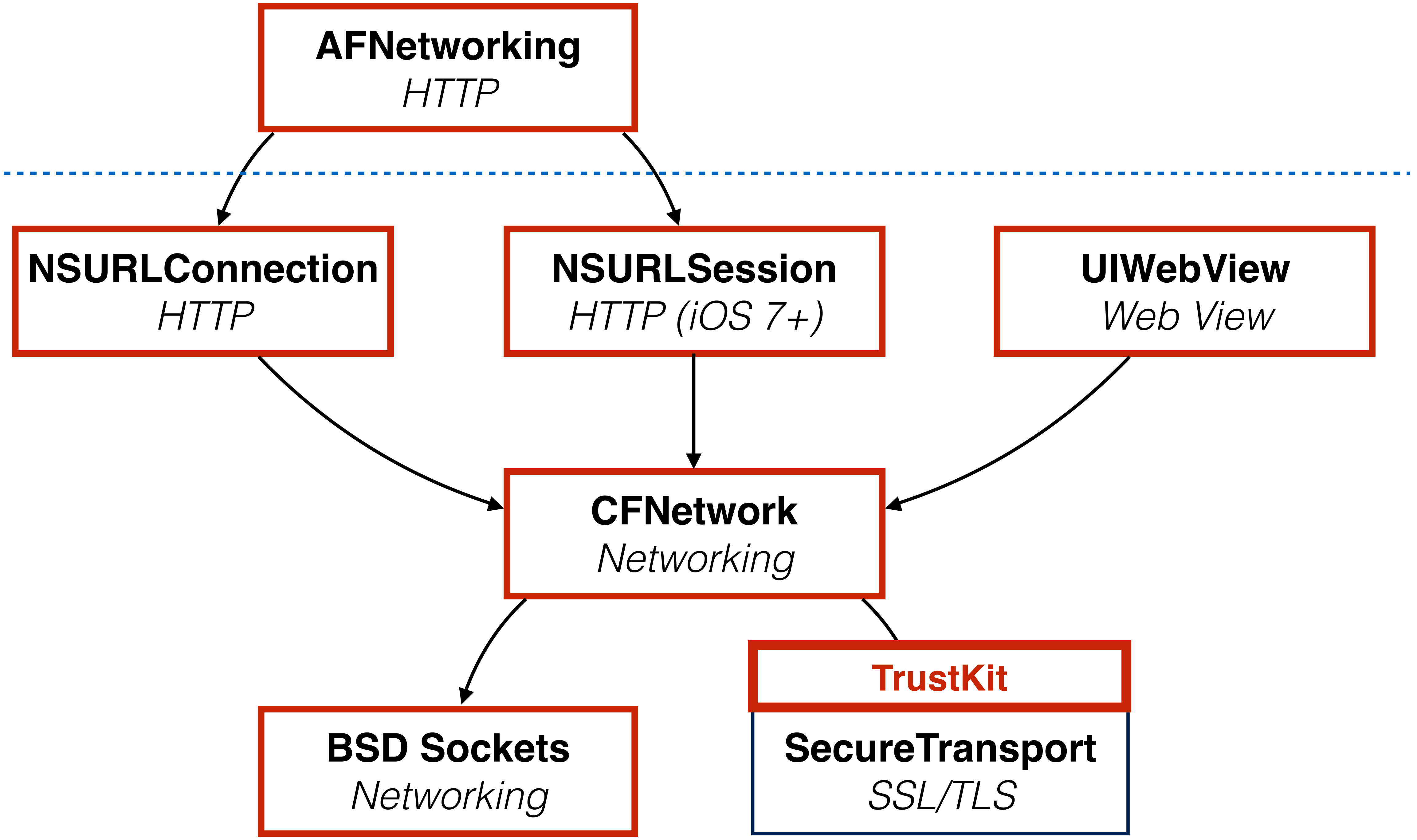
iOS Network Stack



iOS Network Stack



iOS Network Stack



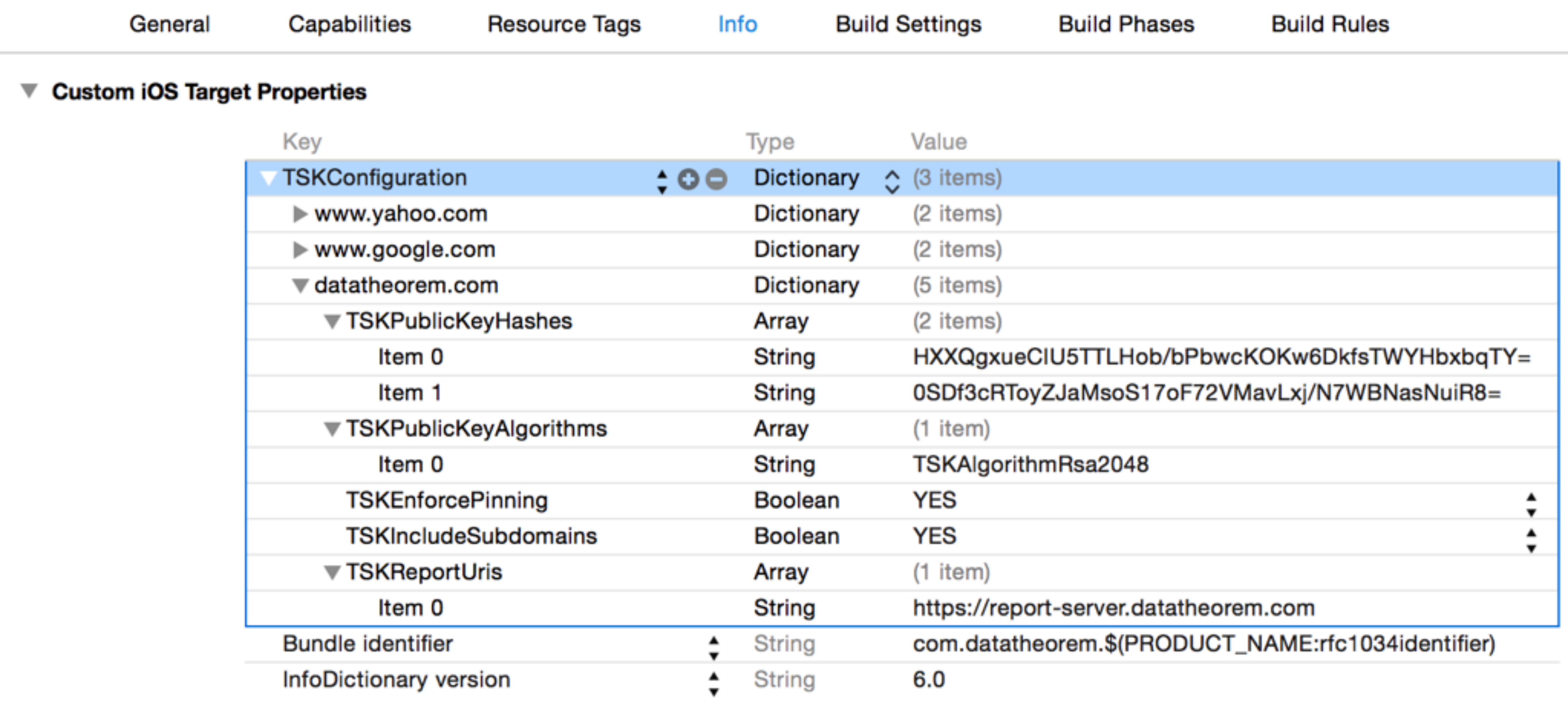
TrustKit

- We solved these challenges
  - TrustKit works transparently on all Apple APIs

TrustKit

- We solved these challenges
  - TrustKit works transparently on all Apple APIs
  - Easy configuration: set the pinning policy in the Info.plist
    - Settings are heavily based on HTTP Public Key Pinning

TrustKit



Custom iOS Target Properties

Key	Type	Value
TSKConfiguration	Dictionary	(3 items)
www.yahoo.com	Dictionary	(2 items)
www.google.com	Dictionary	(2 items)
datatheorem.com	Dictionary	(5 items)
TSKPublicKeyHashes	Array	(2 items)
Item 0	String	HXXQgxueCIU5TTLHob/bPbwcKOKw6DkfsTWYHbxbqTY=
Item 1	String	0SDf3cRToyZJaMsoS17oF72VMavLxj/N7WBNasNuiR8=
TSKPublicKeyAlgorithms	Array	(1 item)
Item 0	String	TSKAlgorithmRsa2048
TSKEnforcePinning	Boolean	YES
TSKIncludeSubdomains	Boolean	YES
TSKReportUris	Array	(1 item)
Item 0	String	https://report-server.datatheorem.com
Bundle identifier	String	com.datatheorem.$(PRODUCT_NAME:rfc1034identifier)
InfoDictionary version	String	6.0

TrustKit

- We solved these challenges
  - TrustKit works transparently on all Apple APIs
  - Easy configuration: set the pinning policy in the Info.plist
    - Settings are heavily based on HTTP Public Key Pinning

TrustKit

- We solved these challenges
  - TrustKit works transparently on all Apple APIs
  - Easy configuration: set the pinning policy in the Info.plist
    - Settings are heavily based on HTTP Public Key Pinning
  - SPKI pinning: Developer needs to specify the key algorithm

TrustKit

- We solved these challenges
  - TrustKit works transparently on all Apple APIs
  - Easy configuration: set the pinning policy in the Info.plist
    - Settings are heavily based on HTTP Public Key Pinning
  - SPKI pinning: Developer needs to specify the key algorithm
  - Report-only mode
    - Format similar to HPKP for pin failure reports

TrustKit

```
{
"port":443,
"include-subdomains":true,
"noted-hostname":"domain.com",
“hostname":"test.domain.com",
“app-bundle-id”:"com.test.testapp",
"validated-certificate-chain":
["-----BEGIN CERTIFICATE----
\nMIILyjCCCrKgAwIBAgIQQcm82qXxNZszqTblPwPAHDANBgkqhkiG9w0BAQUFADCB\r
\ntTELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQL\r
...
\nWkN/I4qtcE3vMxP8O17CkqegVaeI5nvFhca4r4f8MNYoUYT+6J07SxyA5cDsXQ==\n
-----END CERTIFICATE-----",
...
"-----BEGIN CERTIFICATE-----
\nMIIE0zCCA7ugAwIBAgIQGNrRniZ96LtKIVjNzGs7SjANBgkqhkiG9w0BAQUFADCB\r
...
LPKsEdao7WNq\n-----END CERTIFICATE-----"],
"date-time":"2015-06-29T18:12:30Z",
"known-pins":
[
"pin-sha256=\"JbQbUG5JMJUoI6brnx0x3vZF6jilxsapbXGVfjhN8Fg=\"",
“pin-sha256=\"WoiWRyIOVNa9ihaBciRSC7XHjliYS9VwUGOIud4PB18=\""
],
“app-version":"2413"
}
```

TrustKit

- We solved these challenges
    - TrustKit works transparently on all Apple APIs
    - Easy configuration: set the pinning policy in the Info.plist
        - Settings are heavily based on HTTP Public Key Pinning
    - SPKI pinning: Developer needs to specify the key algorithm
    - Report-only mode
        - Format similar to HPKP for pin failure reports

Demo

TrustKit

- We’re open sourcing TrustKit today
    - MIT License
    - https://datatheorem.github.io/TrustKit
    - Also works in OS X Apps
- More on this at the end

TrustKit

- So how does TrustKit work?
  - Leveraged techniques usually used on jailbroken iOS
    - Code injection
    - Low-level C function hooking
  - Could be applied to other things than SSL pinning

How It All Started

- iOS 8 released: **dynamic libraries** now allowed in App Store Apps!

How It All Started

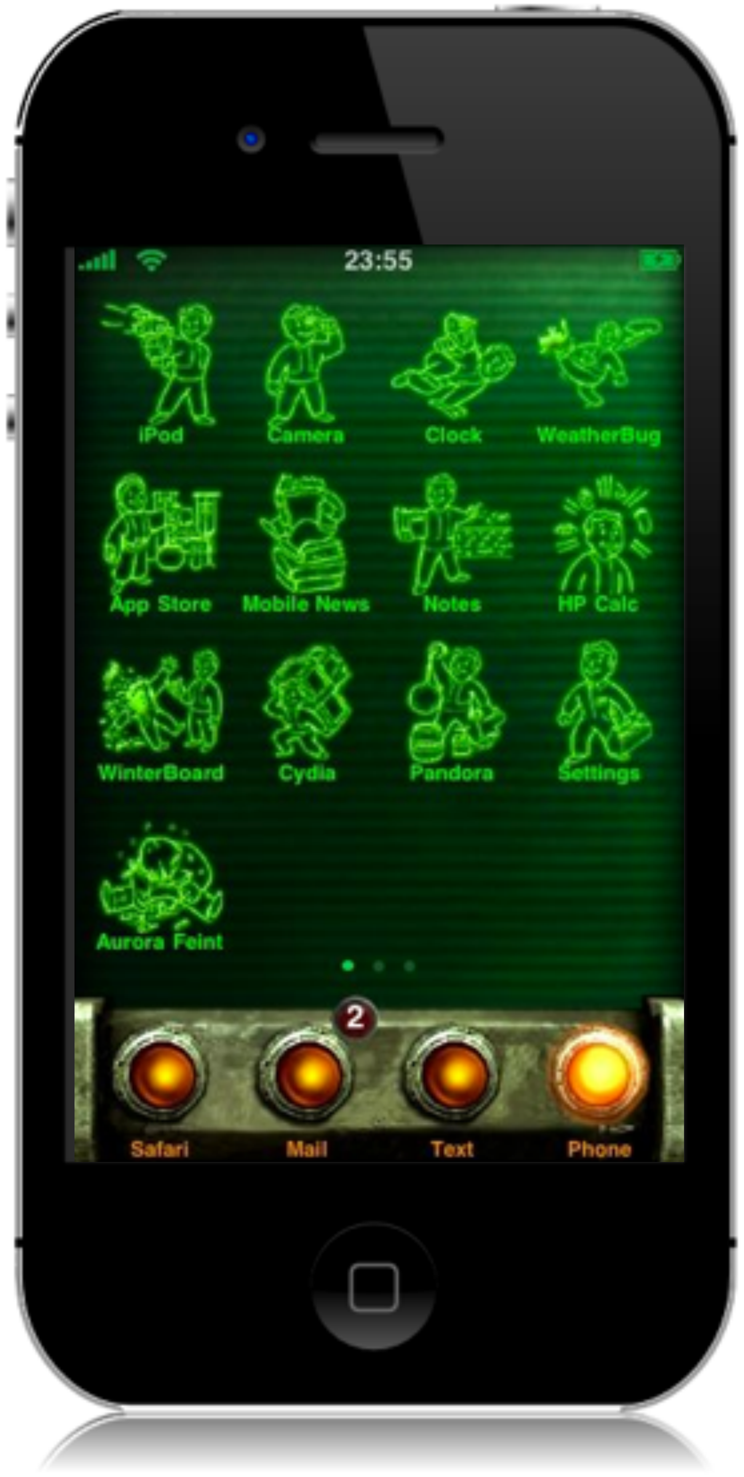
- iOS 8 released: **dynamic libraries** now allowed in App Store Apps!
- Lots of experience building Cydia “tweaks”
    - **Dynamic libraries** that modify Apps at runtime
        - Used for customization and security research
        - Implemented by hooking functions and methods

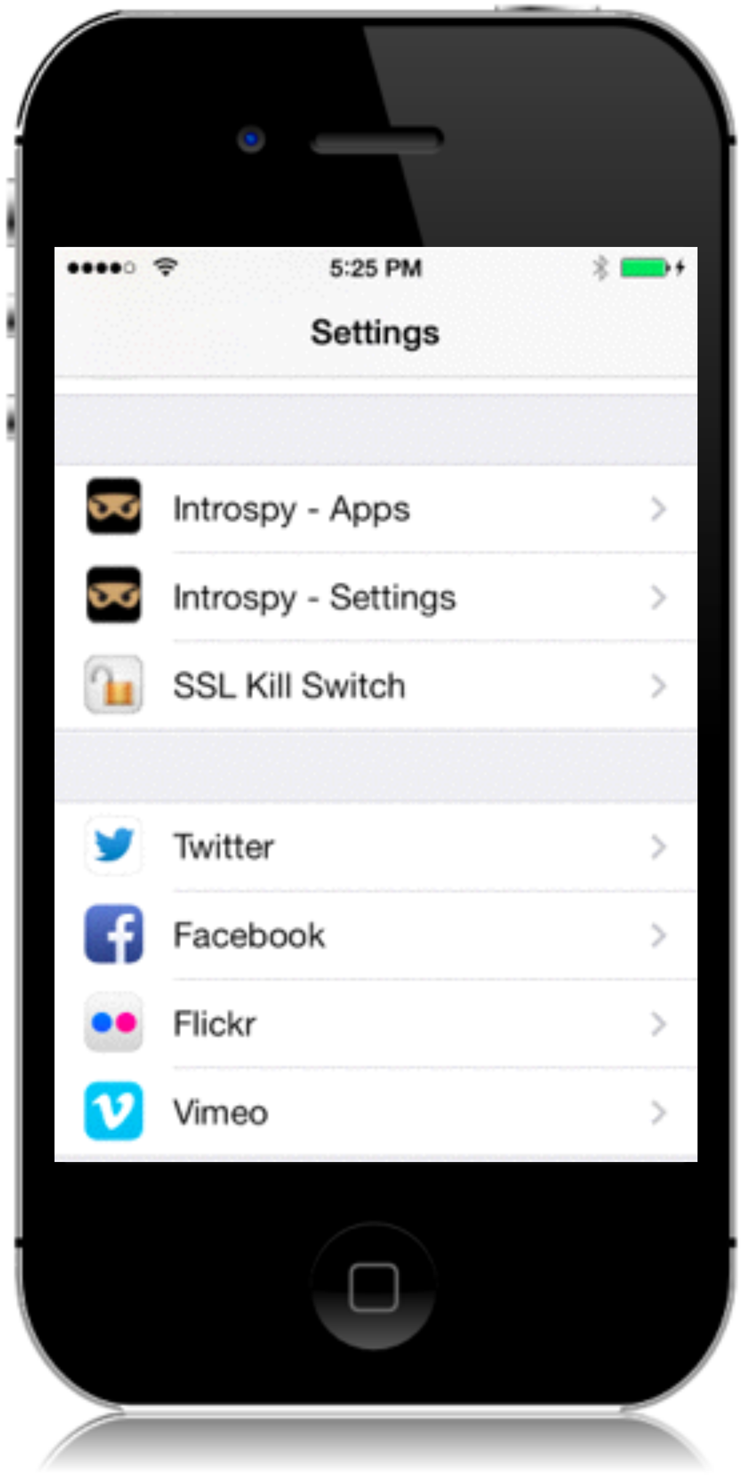
How It All Started

- iOS **c libraries** now allowed in App Sto
- Lot ing Cydia "tweaks"
  - **[** at modify Apps at runtime
    - tion and security research
    - oking functions and methods

How It All Started

- iOS c librarie p
  Sto
- Lot ing Cydi
  - [at modify
    - tion and
    - oking fu




5:25 PM

Settings

Introspy - Apps

Introspy - Settings

SSL Kill Switch

Twitter

Facebook

Flickr

Vimeo

How It All Started

- iOS 8 released: **dynamic libraries** now allowed in App Store Apps!
- Lots of experience building Cydia “tweaks”
  - **Dynamic libraries** that modify Apps at runtime
    - Used for customization and security research
    - Implemented by hooking functions and methods

How It All Started

- iOS 8 released: **dynamic libraries** now allowed in App Store Apps!
- Lots of experience building Cydia “tweaks”
  - **Dynamic libraries** that modify Apps at runtime
    - Used for customization and security research
    - Implemented by hooking functions and methods
- Can we use the same techniques within an iOS 8 App Store App on a non-jailbroken device?

How It All Started

- iOS 8 released: **dynamic libraries** now allowed in App Store Apps!
- Lots of experience building Cydia "tweaks"
  - **Dynamic libraries** that modify Apps at runtime
    - Used for customization and security research
    - Implemented by hooking functions and methods
- Can we use the same techniques within an iOS 8 App Store App on a non-jailbroken device?

Agenda

- TrustKit: effortless SSL pinning for iOS and OS X
- Dynamic libraries and iOS 8
- Function hooking on a non-jailbroken device

Agenda

- TrustKit: effortless SSL pinning for iOS and OS X
- Dynamic libraries and iOS 8
- Function hooking on a non-jailbroken device

Dylibs Before iOS 8

- Historically: no third-party dynamic libraries in Apps
    - System dylibs packaged with the OS

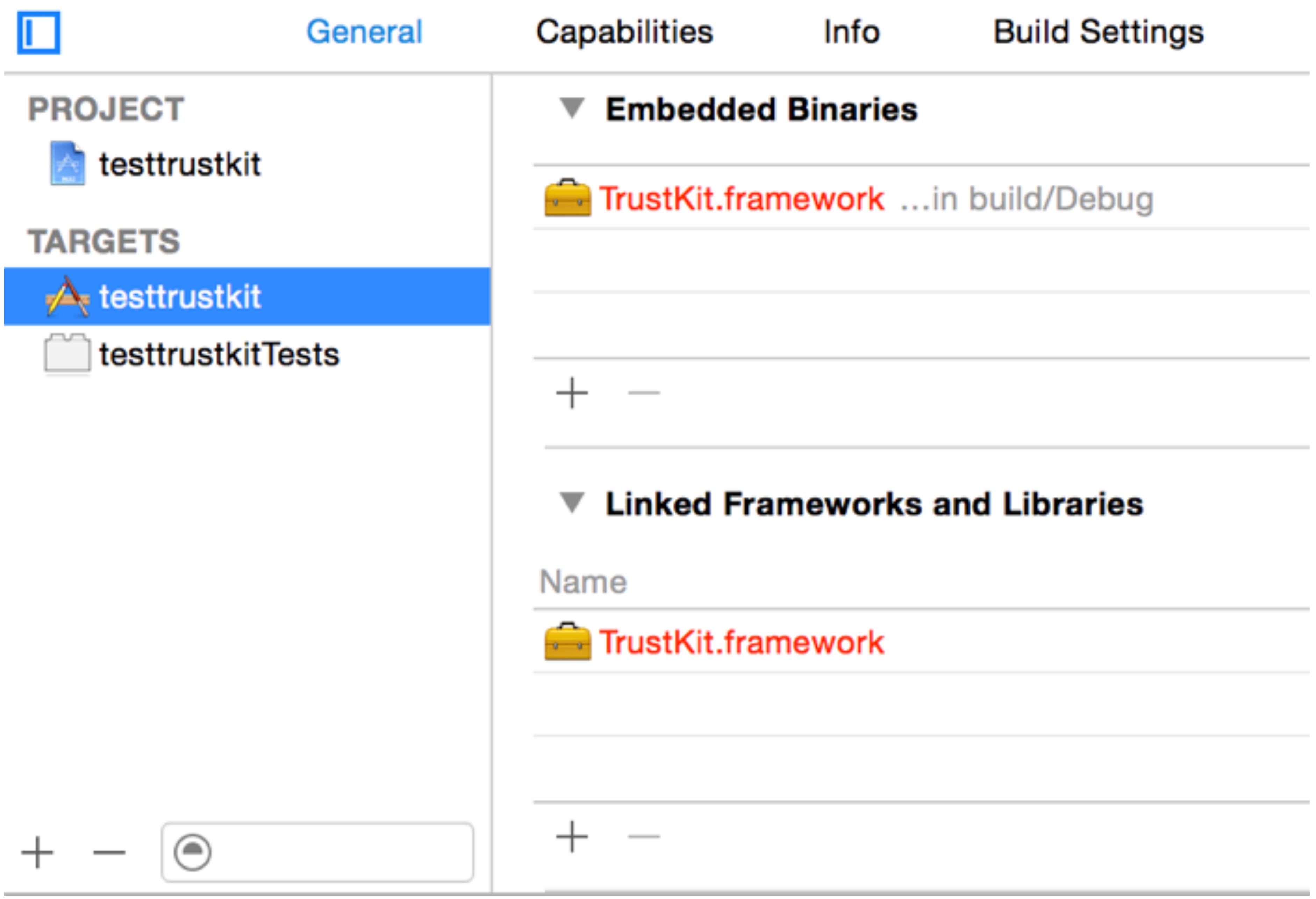
Dylibs Before iOS 8

- Historically: no third-party dynamic libraries in Apps
    - System dylibs packaged with the OS
- Developer libraries: static linking only
    - Enforced via the App Store review process

Dylibs on iOS 8

- iOS 8: dynamic libraries now accepted
  - Apple calls them “Embedded Frameworks”
- Introduced to facilitate sharing code between Apps and their App Extensions
  - But… can be used regardless of whether the App actually has an Extension

Dylibs on iOS 8



Dylibs on iOS 8

- A dynamic library dependency is created in the Mach-O binary in a “load command” structure

- Mach-O is the binary file format for programs and libraries in iOS and OS X

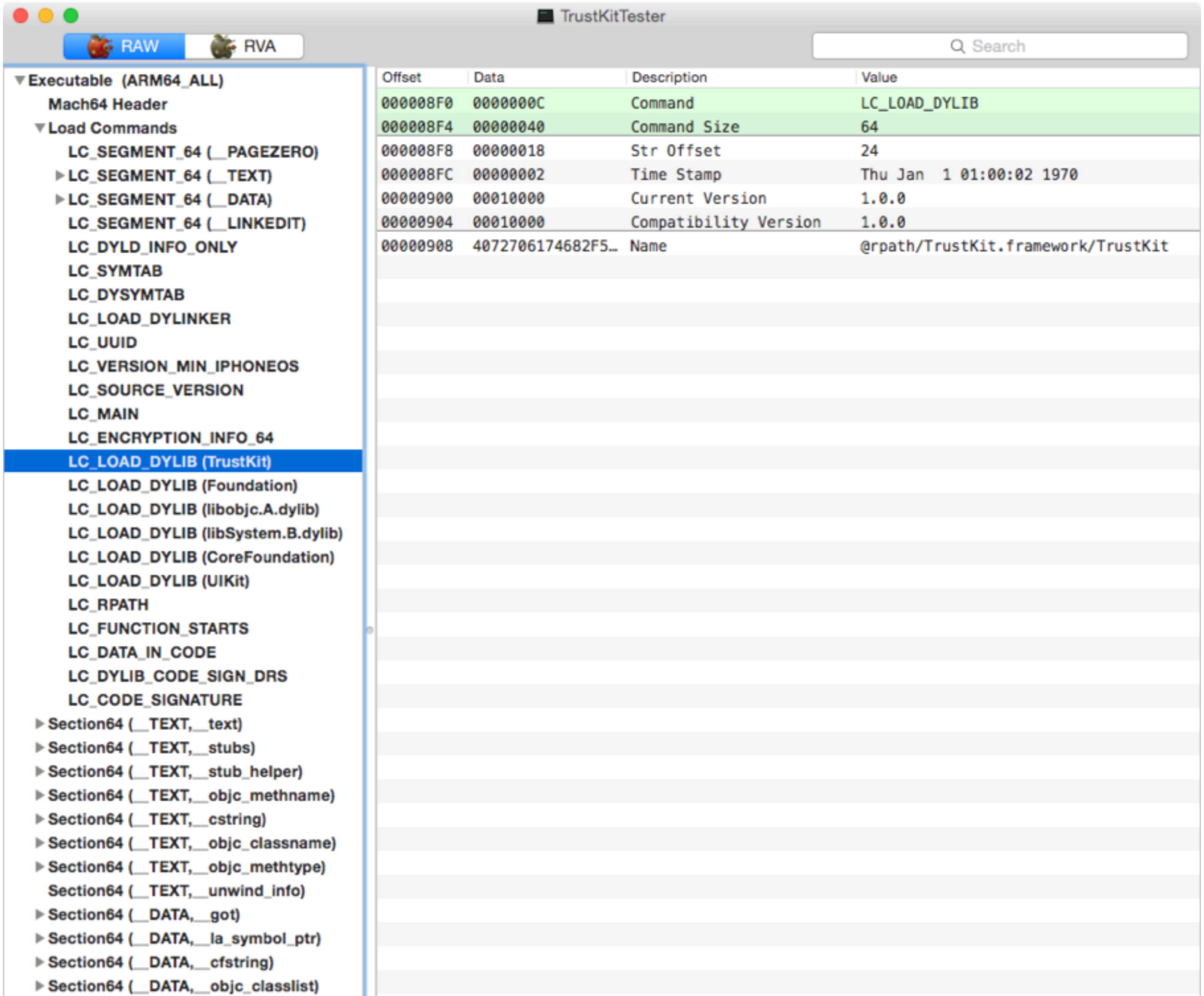
- Executables interact with “dyld” to load their library dependencies at runtime.

Dylibs on iOS 8

- Sandboxing forces our dependencies to be packaged within the app’s bundle

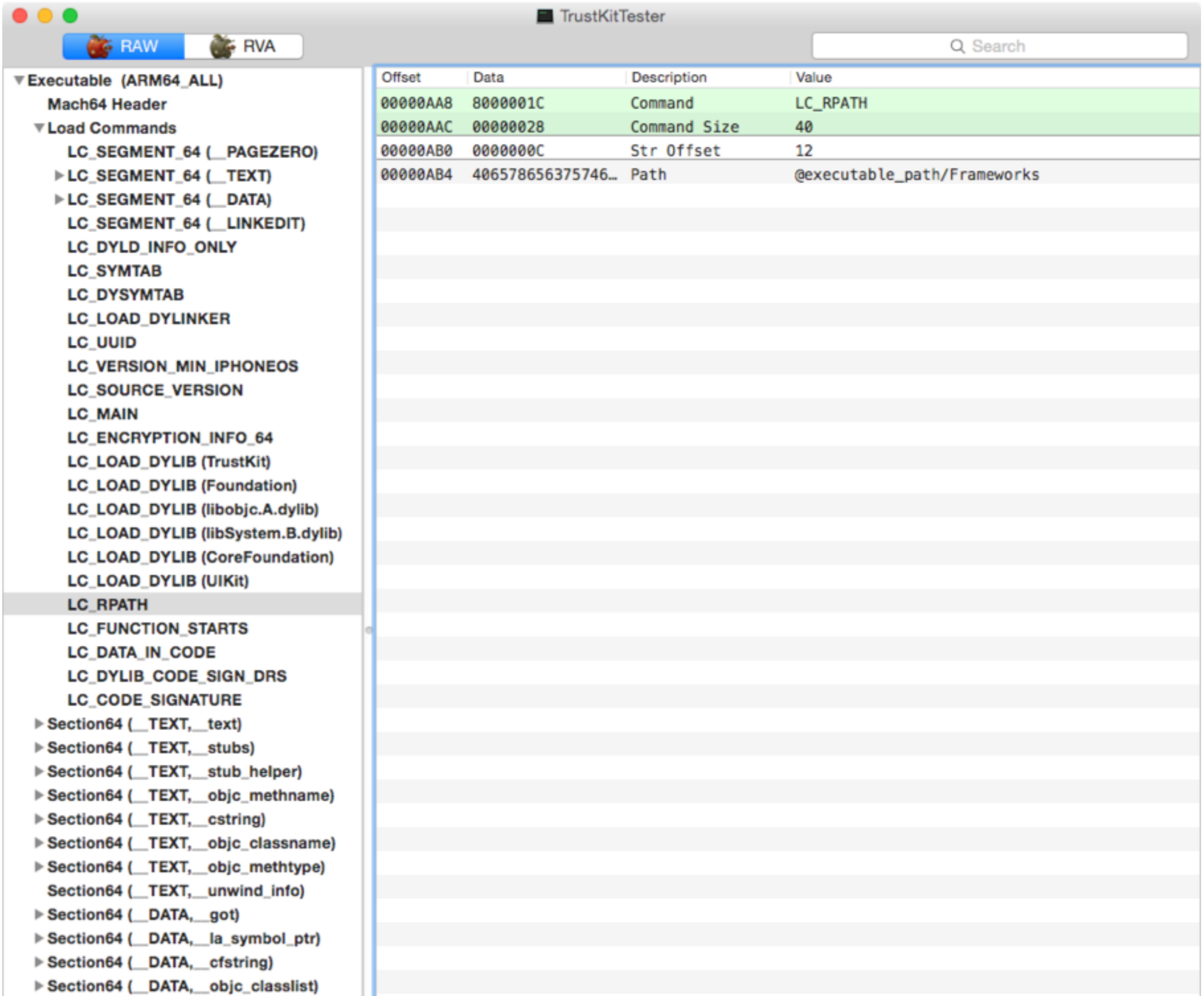
Dylibs on iOS 8

- Sandboxing forces our dependencies to be packaged within the app's bundle
- **dyld** uses prefixes inside the load command to locate them
    - **@executable_path** points to the full path where the main executable is (the **.app** folder).
    - **@rpath** defines library search path locations
        - In iOS, @rpath seems limited to one single location (a **"Frameworks"** directory inside app's bundle)



Executable (ARM64_ALL)
- Mach64 Header
- Load Commands
  - LC_SEGMENT_64 (__PAGEZERO)
  - LC_SEGMENT_64 (__TEXT)
  - LC_SEGMENT_64 (__DATA)
  - LC_SEGMENT_64 (__LINKEDIT)
  - LC_DYLD_INFO_ONLY
  - LC_SYMTAB
  - LC_DYSYMTAB
  - LC_LOAD_DYLINKER
  - LC_UUID
  - LC_VERSION_MIN_IPHONEOS
  - LC_SOURCE_VERSION
  - LC_MAIN
  - LC_ENCRYPTION_INFO_64
  - LC_LOAD_DYLIB (TrustKit)
  - LC_LOAD_DYLIB (Foundation)
  - LC_LOAD_DYLIB (libobjc.A.dylib)
  - LC_LOAD_DYLIB (libSystem.B.dylib)
  - LC_LOAD_DYLIB (CoreFoundation)
  - LC_LOAD_DYLIB (UIKit)
  - LC_RPATH
  - LC_FUNCTION_STARTS
  - LC_DATA_IN_CODE
  - LC_DYLIB_CODE_SIGN_DRS
  - LC_CODE_SIGNATURE
- Section64 (__TEXT,__text)
- Section64 (__TEXT,__stubs)
- Section64 (__TEXT,__stub_helper)
- Section64 (__TEXT,__objc_methname)
- Section64 (__TEXT,__cstring)
- Section64 (__TEXT,__objc_classname)
- Section64 (__TEXT,__objc_methtype)
- Section64 (__TEXT,__unwind_info)
- Section64 (__DATA,__got)
- Section64 (__DATA,__la_symbol_ptr)
- Section64 (__DATA,__cfstring)
- Section64 (__DATA,__objc_classlist)

Offset	Data	Description	Value
000008F0	0000000C	Command	LC_LOAD_DYLIB
000008F4	00000040	Command Size	64
000008F8	00000018	Str Offset	24
000008FC	00000002	Time Stamp	Thu Jan 1 01:00:02 1970
00000900	00010000	Current Version	1.0.0
00000904	00010000	Compatibility Version	1.0.0
00000908	4072706174682F5…	Name	@rpath/TrustKit.framework/TrustKit



```
▼Executable (ARM64_ALL)
    Mach64 Header
  ▼Load Commands
      LC_SEGMENT_64 (__PAGEZERO)
     ▶LC_SEGMENT_64 (__TEXT)
     ▶LC_SEGMENT_64 (__DATA)
      LC_SEGMENT_64 (__LINKEDIT)
      LC_DYLD_INFO_ONLY
      LC_SYMTAB
      LC_DYSYMTAB
      LC_LOAD_DYLINKER
      LC_UUID
      LC_VERSION_MIN_IPHONEOS
      LC_SOURCE_VERSION
      LC_MAIN
      LC_ENCRYPTION_INFO_64
      LC_LOAD_DYLIB (TrustKit)
      LC_LOAD_DYLIB (Foundation)
      LC_LOAD_DYLIB (libobjc.A.dylib)
      LC_LOAD_DYLIB (libSystem.B.dylib)
      LC_LOAD_DYLIB (CoreFoundation)
      LC_LOAD_DYLIB (UIKit)
      LC_RPATH
      LC_FUNCTION_STARTS
      LC_DATA_IN_CODE
      LC_DYLIB_CODE_SIGN_DRS
      LC_CODE_SIGNATURE
  ▶Section64 (__TEXT,__text)
  ▶Section64 (__TEXT,__stubs)
  ▶Section64 (__TEXT,__stub_helper)
  ▶Section64 (__TEXT,__objc_methname)
  ▶Section64 (__TEXT,__cstring)
  ▶Section64 (__TEXT,__objc_classname)
  ▶Section64 (__TEXT,__objc_methtype)
   Section64 (__TEXT,__unwind_info)
  ▶Section64 (__DATA,__got)
  ▶Section64 (__DATA,__la_symbol_ptr)
  ▶Section64 (__DATA,__cfstring)
  ▶Section64 (__DATA,__objc_classlist)
```

Offset	Data	Description	Value
00000AA8	8000001C	Command	LC_RPATH
00000AAC	00000028	Command Size	40
00000AB0	0000000C	Str Offset	12
00000AB4	406578656375746...	Path	@executable_path/Frameworks

Dylib Constructors

- Dynamic libraries can have “constructors”
- Basically a C function that is called when the library is loaded in memory
- We use it to initialize our hooks (patches) and settings
- `__attribute__((constructor)) static void initializer()`

Dylibs Recap

- By adding to the App a load command with our dylib
    - The dylib will be automatically loaded when the App starts
        - The dylib’s constructor will be run first
    - Takes care of the “injection" process

Dylibs Recap

- By adding to the App a load command with our dylib
    - The dylib will be automatically loaded when the App starts
        - The dylib’s constructor will be run first
    - Takes care of the “injection" process
- Can we create a dylib that does C function hooking?

Dylibs Recap

- By adding to the App a load command with our dylib
  - The dylib will be automatically loaded when the App starts
    - The dylib’s constructor will be run first
  - Takes care of the “injection" process
- Can we create a dylib that does C function hooking?

Agenda

- TrustKit: effortless SSL pinning for iOS and OS X
- Dynamic libraries and iOS 8
- Function hooking on a non-jailbroken device

Hooking Jailbreak-Free

- First attempt
  - Tried packaging an actual Cydia Substrate tweak into an App Store App

Substrate in an App

```
Hardware Model:      iPhone6,1
Process:             TestSubstrate [1438]
Path:                /private/var/mobile/Containers/Bundle/Application/AF0E2FD7-BA47-4E57-95ED-
B2C3D6116E62/TestSubstrate.app/TestSubstrate
Identifier:          TestSubstrate
Version:             ???
Code Type:           ARM-64 (Native)
Parent Process:      launchd [1]
Date/Time:           2015-07-16 22:57:43.529 -0700
Launch Time:         2015-07-16 22:57:43.356 -0700
OS Version:          iOS 8.4 (12H143)
Report Version:      105
Exception Type:  EXC_BAD_ACCESS (SIGKILL - CODESIGNING)
Exception Subtype: unknown at 0x0000000186b346c4
Triggered by Thread:  0
Thread 0 name:  Dispatch queue: com.apple.main-thread
Thread 0 Crashed:
0   CydiaSubstrate                0x00000001000931bc 0x100090000 + 12732
1   SSLKillSwitch.dylib           0x0000000100087d30 0x100084000 + 15664
2   dyld                          0x000000012006d234 0x12005c000 + 70196
3   dyld                          0x000000012006d3ec 0x12005c000 + 70636
[...]
```

Substrate in an App

```
Hardware Model:      iPhone6,1
Process:             TestSubstrate [1438]
Path:                /private/var/mobile/Containers/Bundle/Application/AF0E2FD7-BA47-4E57-95ED-
B2C3D6116E62/TestSubstrate.app/TestSubstrate
Identifier:          TestSubstrate
Version:             ???
Code Type:           ARM-64 (Native)
Parent Process:      launchd [1]
Date/Time:           2015-07-16 22:57:43.529 -0700
Launch Time:         2015-07-16 22:57:43.356 -0700
OS Version:          iOS 8.4 (12H143)
Report Version:      105
Exception Type:  EXC_BAD_ACCESS (SIGKILL - CODESIGNING)
Exception Subtype: unknown at 0x0000000186b346c4
Triggered by Thread:  0
Thread 0 name:  Dispatch queue: com.apple.main-thread
Thread 0 Crashed:
0   CydiaSubstrate                0x00000001000931bc 0x100090000 + 12732
1   SSLKillSwitch.dylib           0x0000000100087d30 0x100084000 + 15664
2   dyld                          0x000000012006d234 0x12005c000 + 70196
3   dyld                          0x000000012006d3ec 0x12005c000 + 70636
[...]
```

Substrate in an App

```
Hardware Model:      iPhone6,1
Process:             TestSubstrate [1438]
Path:                /private/var/mobile/Containers/Bundle/Application/AF0E2FD7-BA47-4E57-95ED-
B2C3D6116E62/TestSubstrate.app/TestSubstrate
Identifier:          TestSubstrate
Version:             ???
Code Type:           ARM-64 (Native)
Parent Process:      launchd [1]
Date/Time:           2015-07-16 22:57:43.529 -0700
Launch Time:         2015-07-16 22:57:43.356 -0700
OS Version:          iOS 8.4 (12H143)
Report Version:      105
Exception Type:  EXC_BAD_ACCESS (SIGKILL - CODESIGNING)
Exception Subtype: unknown at 0x0000000186b346c4
Triggered by Thread:  0
Thread 0 name:  Dispatch queue: com.apple.main-thread
Thread 0 Crashed:
0   CydiaSubstrate                  0x00000001000931bc 0x100090000 + 12732
1   SSLKillSwitch.dylib             0x0000000100087d30 0x100084000 + 15664
2   dyld                            0x000000012006d234 0x12005c000 + 70196
3   dyld                            0x000000012006d3ec 0x12005c000 + 70636
[...]
```

Substrate in an App

```
Hardware Model:      iPhone6,1
Process:             TestSubstrate [1438]
Path:                /private/var/mobile/Containers/Bundle/Application/AF0E2FD7-BA47-4E57-95ED-
B2C3D6116E62/TestSubstrate.app/TestSubstrate
Identifier:          TestSubstrate
Version:             ???
Code Type:           ARM-64 (Native)
Parent Process:      launchd [1]
Date/Time:           2015-07-16 22:57:43.529 -0700
Launch Time:         2015-07-16 22:57:43.356 -0700
OS Version:          iOS 8.4 (12H143)
Report Version:      105
Exception Type:  EXC_BAD_ACCESS (SIGKILL - CODESIGNING)
Exception Subtype: unknown at 0x0000000186b346c4
Triggered by Thread:  0
Thread 0 name:  Dispatch queue: com.apple.main-thread
Thread 0 Crashed:
0   CydiaSubstrate              0x00000001000931bc 0x100090000 + 12732  MSFunctionHook()
1   SSLKillSwitch.dylib         0x0000000100087d30 0x100084000 + 15664  Dylib Contructor
2   dyld                        0x000000012006d234 0x12005c000 + 70196
3   dyld                        0x000000012006d3ec 0x12005c000 + 70636
[...]
```

Substrate in an App

- SIGKILL when calling *MSFunctionHook()*
    - Substrate hooks C functions by patching the function’s prologue
    - This requires RWX memory pages
        - Not possible on a non-jailbroken device…

Substrate in an App

- SIGKILL when calling *MSFunctionHook()*
  - Substrate hooks C functions by patching the function's prologue
  - This requires RWX memory pages
    - Not possible on a non-jailbroken device…
    - …Unless running in a debugger

Hooking Jailbreak-Free

- First attempt
  - Tried packaging an actual Cydia Substrate tweak into an App Store App

Hooking Jailbreak-Free

- First attempt
    - Tried packaging an actual Cydia Substrate tweak into an App Store App
    - **Failed:** no way to package a Substrate tweak in an App Store App due to RWX requirement

Hooking Jailbreak-Free

- Second attempt
  - DYLD_INSERT_LIBRARIES and __interpose
    - Similar to LD_PRELOAD on Linux
    - Symbol rebinding: can only override exported functions

Hooking Jailbreak-Free

- Second attempt
  - DYLD_INSERT_LIBRARIES and __interpose
    - Similar to LD_PRELOAD on Linux
    - Symbol rebinding: can only override exported functions
  - Requires setting an environment variable
    - **Failed:** can't be done in an App Store App outside of Xcode

Hooking Jailbreak-Free

- Third attempt
  - Newer libraries for dynamic symbol rebinding

Hooking Jailbreak-Free

- Third attempt
  - Newer libraries for dynamic symbol rebinding
    - comex/substitute
      - Specifically *substitute_interpose_imports()*

Hooking Jailbreak-Free

- Third attempt
  - Newer libraries for dynamic symbol rebinding
    - comex/substitute
      - Specifically *substitute_interpose_imports()*
    - facebook/fishhook

Hooking Jailbreak-Free

- Third attempt
  - Newer libraries for dynamic symbol rebinding
    - comex/substitute
      - Specifically *substitute_interpose_imports()*
    - facebook/fishhook
- **Success:** We were able to create a dylib to automatically hook functions in an App Store App

Hooking Jailbreak-Free

- Third attempt
  - Newer libraries for dynamic symbol rebinding
    - comex/substitute
      - Specifically *substitute_interpose_imports()*
    - facebook/fishhook
- **Success:** We were able to create a dylib to automatically hook functions in an App Store App

Putting It All Together

- One concrete example: TrustKit for SSL pinning
  - Adding TrustKit to the App’s Xcode project:
    - Embeds the dylib in the App’s bundle
    - Adds a load command to the App’s executable

Putting It All Together

- The TrustKit dylib's constructor does all the work:
  - Reads the pinning policy from the App's Info.plist
  - Sets up the SecureTransport hooks
    - Runtime patch for *SSLHandshake()*
    - Uses *facebook/fishhook* for C function hooking

Putting It All Together

- The TrustKit dylib's constructor does all the work:
  - Reads the pinning policy from the App's Info.plist
  - Sets up the SecureTransport hooks
    - Runtime patch for *SSLHandshake()*
    - Uses *facebook/fishhook* for C function hooking
- No need to modify the App's source code or call a TrustKit initialization method!

Conclusion

- We’re open-sourcing TrustKit today - MIT license
  - Supports iOS 7+ and OS X10.9+
  - https://datatheorem.github.io/TrustKit/
- TrustKit is already live in a Yahoo App on the App Store
  - Partnered with other companies who will deploy it in their OS X and iOS Apps
- Feedback, comments and pull requests very welcome!

One Last Thing

- SSL pinning can be a challenge for security researchers
  - And is not designed to block an attacker running code as root on the device…
  - So I also released SSL Kill Switch 2
    - https://github.com/nabla-c0d3/ssl-kill-switch2
    - Added support for TrustKit Apps (and OS X)

Thanks!

black hat USA 2015