iOS, Your OS, Everybody's OS: Vetting and Analyzing Network Services of iOS Applications

Presenter: Ken Fang

May 18, 2021

Introduction

- **Network service** is used to provide networked data storage, or other online functionality to apps.

Introduction

- **Network service** is used to provide networked data storage, or other online functionality to apps.
- **Network service** is wildly used in mobile applications.

Introduction

- **Network service** is used to provide networked data storage, or other online functionality to apps.
- **Network service** is wildly used in mobile applications.
  - Mobile Device Management.

Introduction

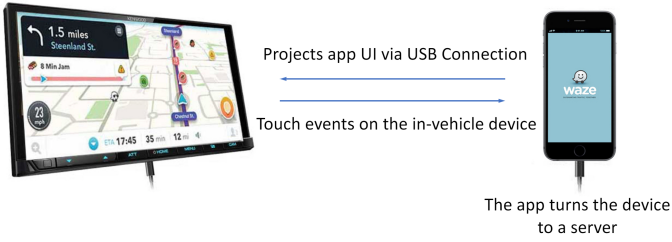
- **Network service** is used to provide networked data storage, or other online functionality to apps.
- **Network service** is wildly used in mobile applications.
  - Mobile Device Management.
  - Receive command from a peripheral equipment (IoT device).

Introduction

- **Network service** is used to provide networked data storage, or other online functionality to apps.
- **Network service** is wildly used in mobile applications.
  - Mobile Device Management.
  - Receive command from a peripheral equipment (IoT device).
  - File delivering, voice calls, etc., (Point-to-Point Network).

Introduction

- **Network service** is used to provide networked data storage, or other online functionality to apps.
- **Network service** is wildly used in mobile applications.
  - Mobile Device Management.
  - Receive command from a peripheral equipment (IoT device).
  - File delivering, voice calls, etc., (Point-to-Point Network).
  - File or media sharing, etc., (Content Delivery Network).



Motivation

- Recent research evaluating the security of open port usage in Android apps has demonstrated new attack avenues that can exploit the vulnerability of network services and access unauthorized sensitive data previously unthought of.

Motivation

- Recent research evaluating the security of open port usage in Android apps has demonstrated new attack avenues that can exploit the vulnerability of network services and access unauthorized sensitive data previously unthought of.
- There are many works on how to solve limitations on static analysis android apps such as handle dynamic code loading, complex implicit control/data flows, or advanced code obfuscation.

Motivation

- Recent research evaluating the security of open port usage in Android apps has demonstrated new attack avenues that can exploit the vulnerability of network services and access unauthorized sensitive data previously unthought of.
- There are many works on how to solve limitations on static analysis android apps such as handle dynamic code loading, complex implicit control/data flows, or advanced code obfuscation.

What about iOS?

Goals

- Vetting and analyzing network services of iOS applications.

Goals

- Vetting and analyzing network services of iOS applications.
- Understanding network service of iOS ecosystem.

Challenges

- Public repository of iOS apps is not available

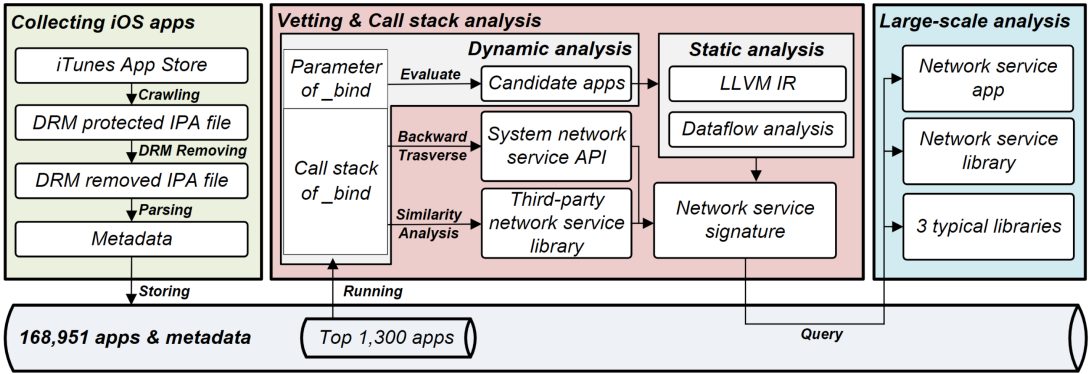
Challenges

- Public repository of iOS apps is not available
- Practical tools for automatically analyzing iOS apps are not well developed.

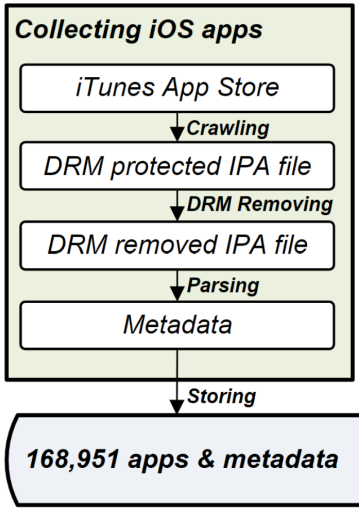
Challenges

- Public repository of iOS apps is not available
- Practical tools for automatically analyzing iOS apps are not well developed.
- Network service libraries of iOS apps are not available.

Design Overview

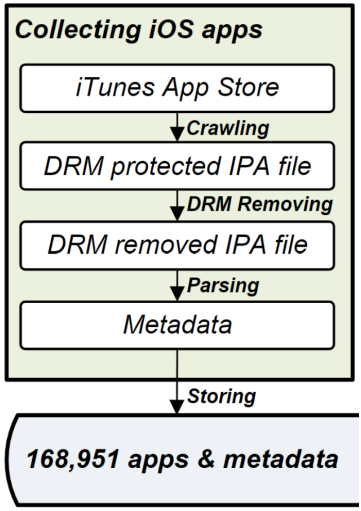


Collecting iOS Apps



- Download apps from iTunes app store utilizing iTunes' dll file

Collecting iOS Apps



- Download apps from iTunes app store utilizing iTunes' dll file
- Decrypt the app by only delivering the executable

Collecting iOS Apps – cont

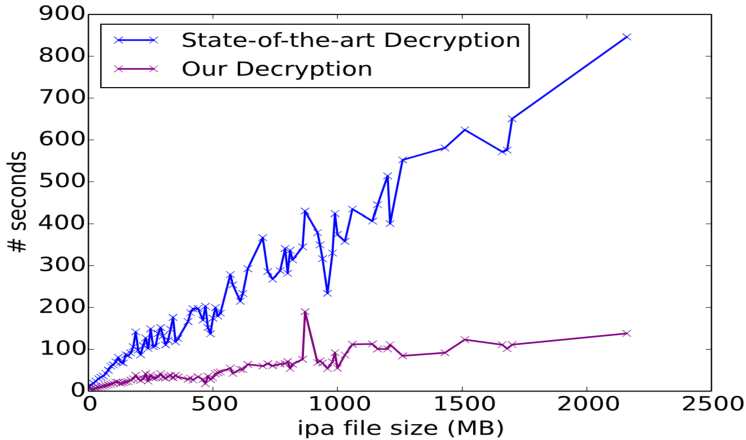


Figure: Performance of decryption

Vetting

- Dynamic analysis

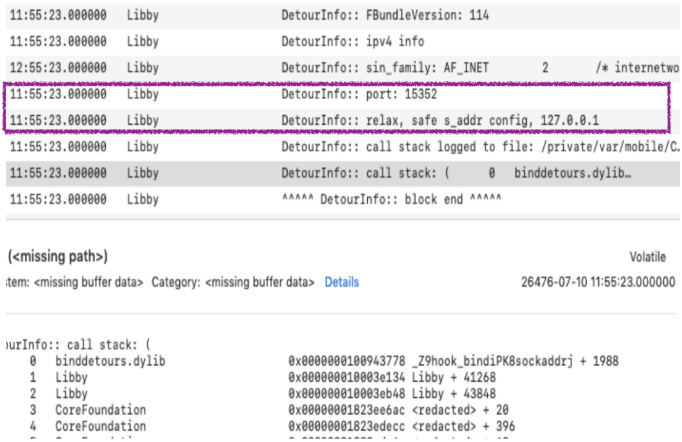
Vetting

- Dynamic analysis
- Static analysis

Vetting

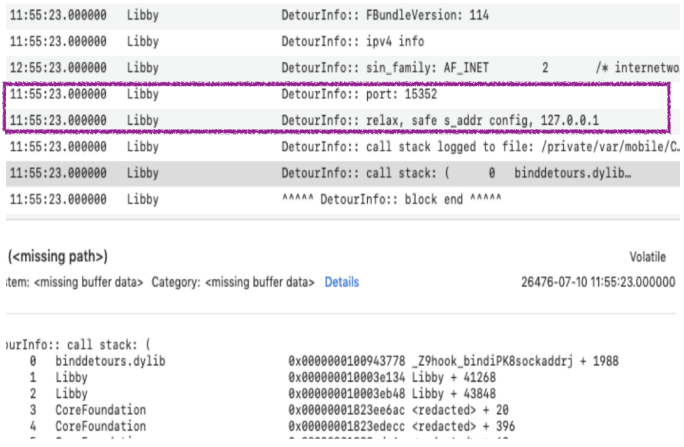
- Dynamic analysis
- Static analysis
- Manual confirmation

Dynamic analysis



- From inspection of network interface, we can find out which app provide network service
  - Deploy an addon on jailbroken iOS devices to redirect _bind API calls
  - Fetch parameters of _bind API

Dynamic analysis



- From inspection of network interface, we can find out which app provide network service
  - Deploy an addon on jailbroken iOS devices to redirect _bind API calls
  - Fetch parameters of _bind API
- Call stack extraction
  - The call stack is maintained by the addon for network service library analysis

Result of Dynamic Analysis

	Dynamic Port (0)	Loopback Interface (e.g., 127.0.0.1)	LAN Interface
China (480)	16 (3.33%)	14 (2.91%)	**51 (11.04%)**
United States (820)	42 (5.12%)	43 (5.24%)	**62 (7.01%)**
Total (1,300)	58 (4.46%)	57 (4.38%)	**113 (8.69%)**

- Dynamic port is for in-app communication

Result of Dynamic Analysis

	Dynamic Port (0)	Loopback Interface (e.g., 127.0.0.1)	LAN Interface
China (480)	16 (3.33%)	14 (2.91%)	**51 (11.04%)**
United States (820)	42 (5.12%)	43 (5.24%)	**62 (7.01%)**
Total (1,300)	58 (4.46%)	57 (4.38%)	**113 (8.69%)**

- Dynamic port is for in-app communication
- In iOS, attack service bind on loopback is impossible.

Result of Dynamic Analysis

	Dynamic Port (0)	Loopback Interface (e.g., 127.0.0.1)	LAN Interface
China (480)	16 (3.33%)	14 (2.91%)	**51 (11.04%)**
United States (820)	42 (5.12%)	43 (5.24%)	**62 (7.01%)**
Total (1,300)	58 (4.46%)	57 (4.38%)	**113 (8.69%)**

- Dynamic port is for in-app communication
- In iOS, attack service bind on loopback is impossible.
- The app that need static analysis are apps listen on LAN interface.

Static Analysis

Researcher solution:

- Supplement semantics of more ARM instructions to the decompiler to reduce memory usage

Static Analysis

Researcher solution:

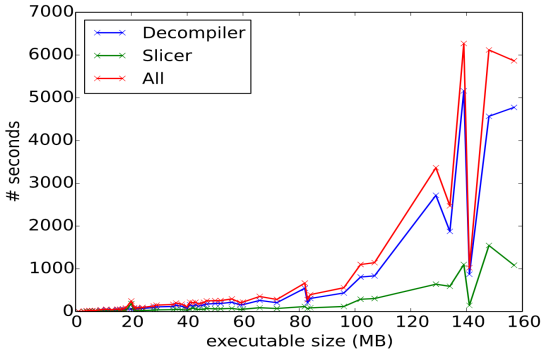
- Supplement semantics of more ARM instructions to the decompiler to reduce memory usage
- Convert inter-procedural data-flow analysis to on-demand inter-procedural to speed up analysis performance

Static Analysis

Researcher solution:

- Supplement semantics of more ARM instructions to the decompiler to reduce memory usage
- Convert inter-procedural data-flow analysis to on-demand inter-procedural to speed up analysis performance
- Formulate and specify rules for the misuse of network services

Static Analysis – result



Performance overview

PATH #1

Trace

-[GCDWebUploader initWithUploadDirectory:]0x7ff2c5f953a0 call void @objc_msgSend(%regset* %0)(store i64 %X0_6421, i64* %X3_ptr, align 4)
Called:
-[GCDWebServer addGETHandlerForBasePath:directoryPath:indexFilename:cacheAge:allowRangeRequests:]

-[GCDWebUploader initWithUploadDirectory:]0x7ff2c5f94290 store i64 %X0_6421, i64* %X3_ptr, align 4(store i64 %X0_6421, i64* %X3_ptr, align 4)

-[GCDWebUploader initWithUploadDirectory:]0x7ff2c5d59b58 %X8_3778 = load i64, i64* undef, align 1(@100 = external global i1)

0x7ff2c5f807e0 i64 4295171188(@100 = external global i1)
Load from 0x100031C74: 10307921512

PATH #2

Trace

-[AppDelegate application:didFinishLaunchingWithOptions:]0x7ff2c67305b0 call void @objc_msgSend(%regset* %0)(store i64 %X0_7536, i64* %X3_ptr, align 4)
Called:
-[GCDWebServer addGETHandlerForBasePath:directoryPath:indexFilename:cacheAge:allowRangeRequests:]

-[AppDelegate application:didFinishLaunchingWithOptions:]0x7ff2c6730310 store i64 %X0_7536, i64* %X3_ptr, align 4(store i64 %X0_7536, i64* %X3_ptr, align 4)

-[AppDelegate application:didFinishLaunchingWithOptions:]0x7ff2c672f490 call void @NSHomeDirectory(%regset* %0)(@98 = external global i1)
Called:
NSHomeDirectory

Analysis result

Manual confirmation

- Automated analysis can't verify specified vulnerabilities.

Manual confirmation

- Automated analysis can't verify specified vulnerabilities.
- Mainly focus on remote vulnerabilities for exploits.

Manual confirmation

- Automated analysis can't verify specified vulnerabilities.
- Mainly focus on remote vulnerabilities for exploits.
- For example, vulnerable service exposed to cellular network is high risk, medium risk for exposed to Wi-Fi network.

Signature of Network Services

Since call stacks don't have enough information, we need to map them to token.

Signature of Network Services – cont

- Third-party libraries have a larger weight on similarity ratio of two different call stacks.
- Researchers duplicate the token multiple times according to the weight. Then calculate the similarity of two new tokens.

Result of signature comparison

Library (a.k.a., Framework)	Signatures	Location	China (480)	United States (820)	1,300 apps	168,951 apps
libSystem	_bind	Symbol Table	**353 (73.54%)**	**331 (40.37%)**	**684 (52.62%)**	**69,238 (40.98%)**
libresolv	_res_9_nquery	Symbol Table	56 (11.67%)	1 (0%)	57 (4.38%)	1,481(0.88%)
CoreFoundation	_CFSocketSetAddress	Symbol Table	112 (23.33%)	57 (6.95%)	169 (13%)	11,965 (7.08%)
GameKit (1)	_OBJC_CLASS_$_GKLocalPlayer	Symbol Table	0 (0%)	10 (1.22%)	10 (0.77%)	2,673 (1.58%)
	localPlayer	String Table				
	registerListener:	String Table				
GameKit (2)	_OBJC_CLASS_$_GKMatchmaker	Symbol Table	1 (0%)	12 (1.46%)	13 (1%)	5,580 (3.3%)
	sharedMatchmaker	String Table				
	setInviteHandler:	String Table				
MultipeerConnectivity	_OBJC_CLASS_$_MCSession	Symbol Table	10 (2.08%)	3 (0.37%)	13 (1%)	604 (0.36%)

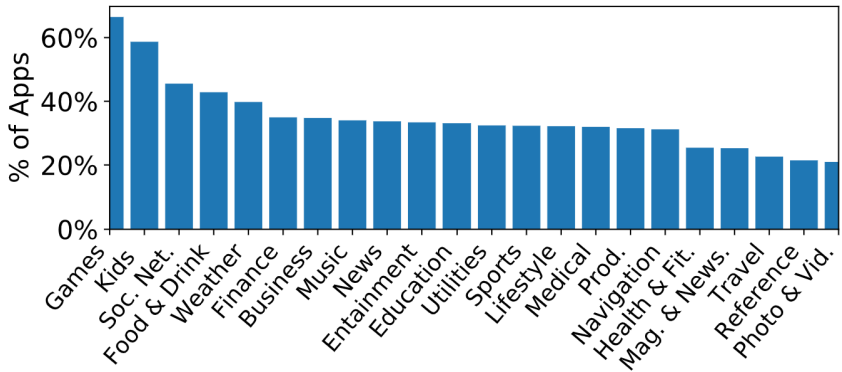
- Over 40.98% apps use system APIs to provide network services.

Result of signature comparison

Library (a.k.a., Framework)	Signatures	Location	China (480)	United States (820)	1,300 apps	168,951 apps
libSystem	_bind	Symbol Table	**353 (73.54%)**	**331 (40.37%)**	**684 (52.62%)**	**69,238 (40.98%)**
libresolv	_res_9_nquery	Symbol Table	56 (11.67%)	1 (0%)	57 (4.38%)	1,481(0.88%)
CoreFoundation	_CFSocketSetAddress	Symbol Table	112 (23.33%)	57 (6.95%)	169 (13%)	11,965 (7.08%)
GameKit (1)	_OBJC_CLASS_$_GKLocalPlayer	Symbol Table	0 (0%)	10 (1.22%)	10 (0.77%)	2,673 (1.58%)
	localPlayer	String Table				
	registerListener:	String Table				
GameKit (2)	_OBJC_CLASS_$_GKMatchmaker	Symbol Table	1 (0%)	12 (1.46%)	13 (1%)	5,580 (3.3%)
	sharedMatchmaker	String Table				
	setInviteHandler:	String Table				
MultipeerConnectivity	_OBJC_CLASS_$_MCSession	Symbol Table	10 (2.08%)	3 (0.37%)	13 (1%)	604 (0.36%)

- Over 40.98% apps use system APIs to provide network services.
- Apps in China are more likely to provide network service than apps in US.

Overall result



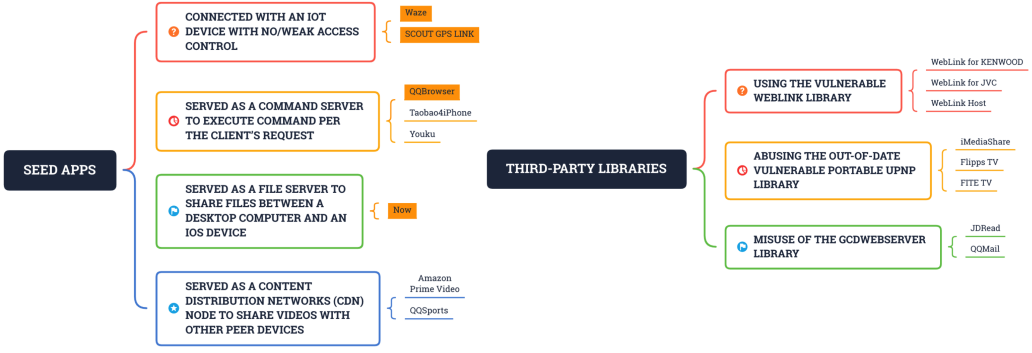
Game App are the most likely to provide network services.

Overall result

#	Library/API	Library/API
1	Tapjoy-CocoaHTTPServer-Extension	CocoaHTTPServer
2	Tapjoy-CocoaHTTPServer-Extension	CocoaAsyncSocket
3	PDRCoreHttpDaemon	_CFSocketSetAddress
4	Ionics_Webview	GCDWebServer
5	CocoaHTTPServer	CocoaAsyncSocket
6	Happy_DNS	_res_9_nquery
7	MAASDK	CocoaAsyncSocket
8	Ionics_Webview	_bind
9	wangxin.taobao	_CFSocketSetAddress
10	MongooseDaemon	_bind
11	CocoaAsyncSocket	_bind
12	Tapjoy-CocoaHTTPServer-Extension	_bind
13	CocoaHTTPServer	_bind
14	TencentVideoHttpProxy	CocoaAsyncSocket
15	Platinum_UPnP	_bind
16	GCDWebServer	_bind
17	upnpx	_bind
18	DIAL_UPnP	_bind
19	WebRTC	_bind
20	SmartDeviceLink	_bind
21	Connect_SDK_Core_(iOS)	DIAL
22	FunTV	CocoaAsyncSocket
23	Unreal_Engine_4	Game_Kit_(2)
24	TencentVideoHttpProxy	CocoaHTTPServer
25	wangxin.taobao	_bind
26	UnityEngine.iOS	_bind

Figure: Connection between third-party network service and system network service APIs

Vulnerabilities in iOS Apps



Summary & Mitigation

- App developers should avoid use LAN interface as much as possible.

Summary & Mitigation

- App developers should avoid use LAN interface as much as possible.
- Network admin of Wifi and cellular network should enable strict firewall policy to whitelist connection between same LAN network.

Summary & Mitigation

- App developers should avoid use LAN interface as much as possible.
- Network admin of Wifi and cellular network should enable strict firewall policy to whitelist connection between same LAN network.
- Mobile OS vendor should implement a host-based firewall.