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

Zhushou Tang, Shanghai Jiao Tong University and PWNZEN InfoTech Co., LTD; Ke Tang, Shanghai Jiao Tong University; Minhui Xue, The University of Adelaide; Yuan Tian, University of Virginia; Sen Chen, Nanyang Technological University; Muhammad Ikram, Macquarie University; Tielei Wang, PWNZEN InfoTech Co., LTD; Haojin Zhu, Shanghai Jiao Tong University

Presenter: Yujun Zhang
yzha725@aucklanduni.ac.nz

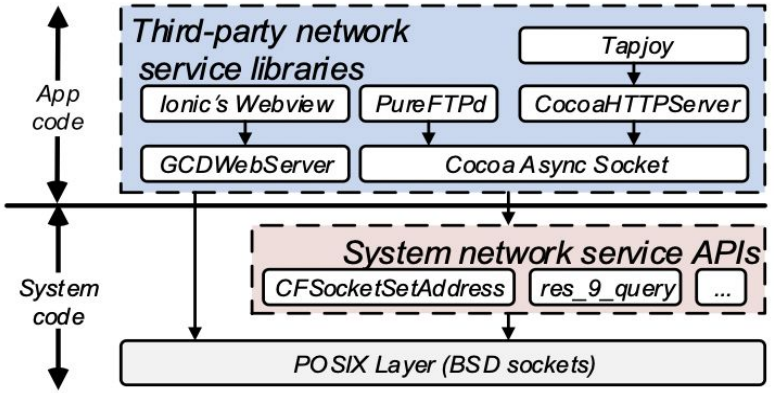
Background

- In April 2021, iOS has 26.99% of worldwide market share
- Network Services are used in multiple purposes with smart devices:
  - Online data storage
  - Smart 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)

iOS Network Structure

- Top-down Network Services
  - See Figure
- Service build on top of BSD sockets
  - Both System APIs and 3rd party Libraries
  - Directly or indirectly
  - See Figure

Network service	Possible mistakes	Vulnerability impact
Resources / functionalities	(M1) Over resource / functionalities	(A1) Privacy leakage / command execution
Access control	(M2) Weak / no access control	(A2) Easily bypass / unauthorized access
Communication protocol	(M3) Bugs in the implementation	(A3) DoS / RCE
Open port	(M4) Interface misconfiguration	(A4) Attack surface exposure

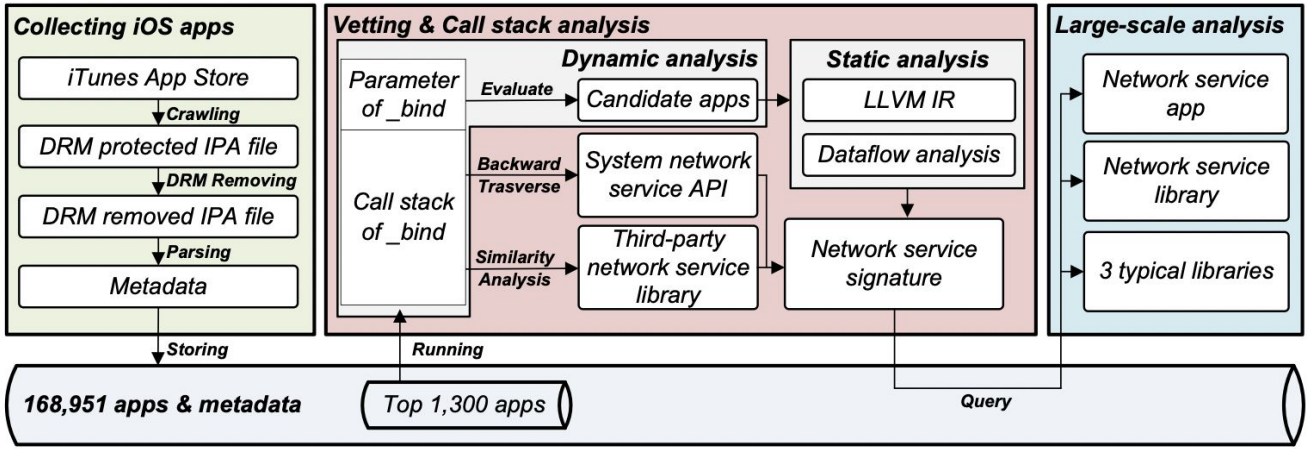


Challenges

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

In This Paper

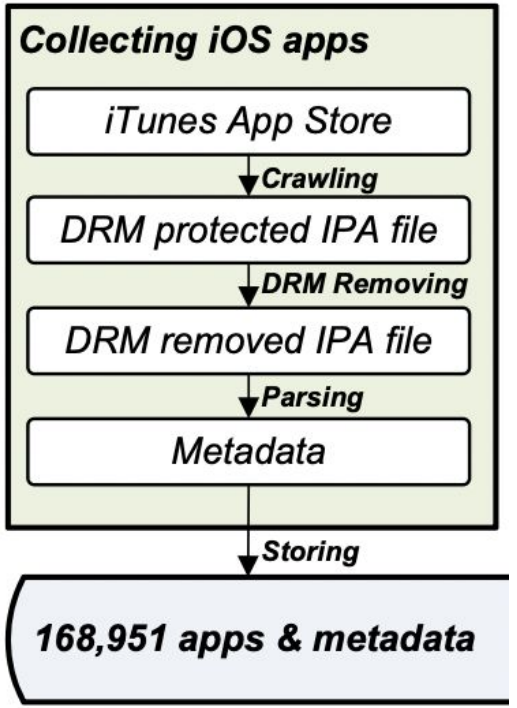
- An efficient way to collect iOS apps
- Systematic characterization and analysis of network services of iOS apps
- Identify new vulnerabilities of iOS apps

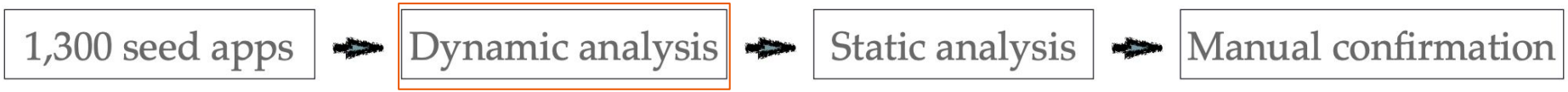


1,300 seed apps → Dynamic analysis → Static analysis → Manual confirmation

App Collection

- Collecting IDs from iTunes Searching API
  - Collected 168,951 apps in 30 days
- Download Apps using a headless-downloader
  - For purchasing and downloading DRM protected Apps
  - Leverages iTunes .dll flies of iTunes for Windows
- Decrypting the executable using jailbroken iOS device
- Parsing the Executable using JTool
- Select Seed Apps.





Analysis - Dynamic

- **Selected 1300 Seed Apps**
  - Top 20 free apps from each category
  - 480 from China and 820 from US
- **Check which app provides a network service by inspecting network interface**
  - By Implement an “Add-on” on Jailbroken device
  - Redirect and parse `_bind` API calls
- **Call stack extraction**
  - Exposed and preserved by the “Add-on”

* _bind calls is used by developers to pass rich parameters to limit the access scope of network services interfaces.

1,300 seed apps → Dynamic analysis → Static analysis → Manual confirmation

Dynamic Analysis Result

Three categories of network interfaces:

- The **dynamic port** to which a socket binds is usually used for in-app communication.
- Attack target network service on **loopback interface** is not practical in iOS .
- For apps that use the **LAN interface**, we run static analysis on them later.

	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%)**

1,300 seed apps → Dynamic analysis → Static analysis → Manual confirmation

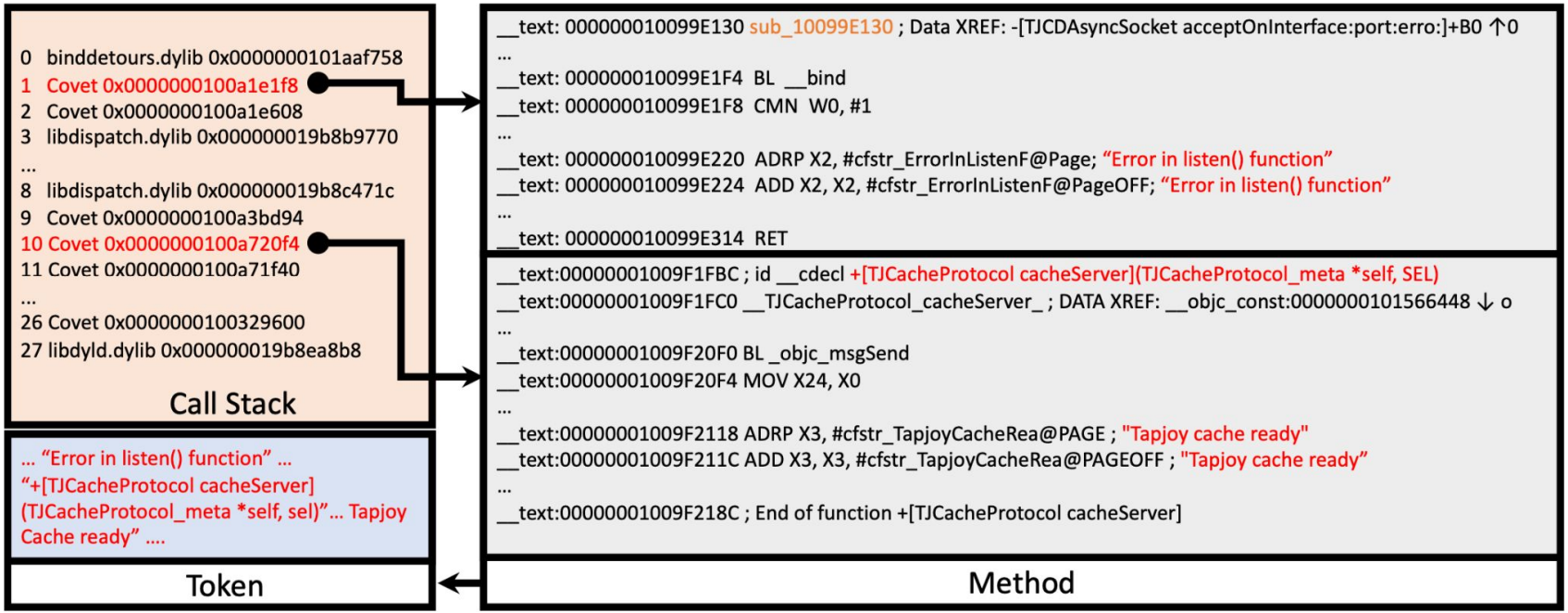
Analysis - Static and Manual

- **Static Analysis Pipeline**

- **“Dynamic first, static later, and manual confirmation last”**
  - Dynamic analysis rapidly check for misconfigured network interface
  - Static analysis is more time consuming but fine-grained
  - Manual confirmation used to verify the identified vulnerabilities

1,300 seed apps → Dynamic analysis → Static analysis → Manual confirmation

Deeper scope - Building Signature for Network services



1,300 seed apps → Dynamic analysis → Static analysis → Manual confirmation

Vetting results

- **By Dynamic vetting process**
  - 172 unique apps, 13.2% of collected total provide network services
  - 65 from China and 107 form US
- **By Static and manual analysis**
  - 11 (9.7%) of the 113 candidate apps have vulnerabilities
  - Include Waze, QQBrowser, Now, Scout GPS and Youku

	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%)**

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: 103079215120
```

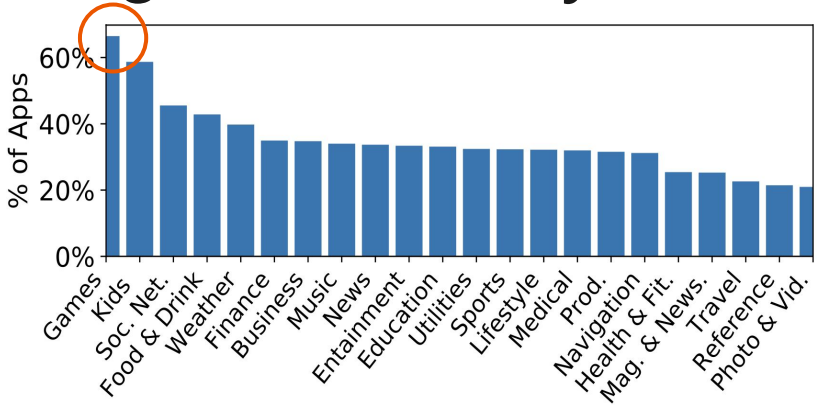
PATH #2

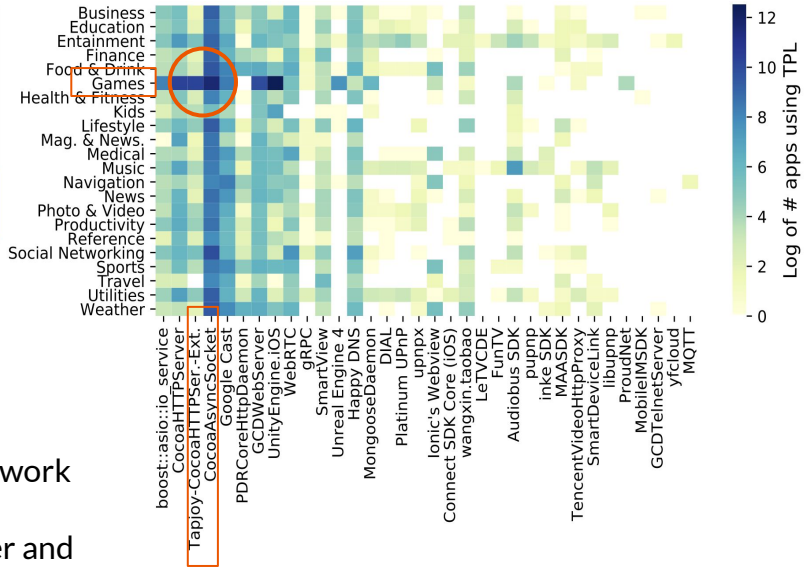
Trace

```
-[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
```

Figure: example of static analysis result of NOW app

Larger-Scale Analysis Result - Ecosystem of iOS





Remarkable:

- Apps in Game category are inclined to provide network services
- The Game category mainly uses CocoaHTTPServer and GCDWebServer libraries may lead to the library misuse.

Found Vulnerabilities - Seed Apps

- Connected with a IOT device with no/week access control
  - Waze, Scout GPS Link
- Served as a command server to execute command per the client's request
  - QQBrowser, Taobao4iPhone, Youku
- Served as a file server to share files between PC and iOS device
  - Now
- Served as a CDN node to share videos with other peer devices
  - Amazon Prime Video, QQSports

Found Vulnerabilities - Network Service Libraries

- Using the vulnerable Weblink library
  - WebLink series libraries such as WebLink for KENWOOD, WebLink for JVC
- Abusing the out-of-date vulnerable portable UPNP library
  - iMediaShare, Flipps TV, Fite TV
- Misuse of the GCDwebserver Library
  - JDRead, QQMail

Conclusions

- Remarkable
  - Network service is wildly used in iOS applications.
  - Vulnerabilities are wildly exist in these network services.
  - Vulnerabilities: 11 in 1,300 seed apps, 92 in 168,951 apps
- Mitigation Strategies
  - Developers should use loopback interface as much as possible to avoid unnecessary or misuse of LAN interface
  - Stricter firewall strategies and block unknown connection attempts from same LAN network
  - Less usage of misused or vulnerable third party libraries

Thank you!

Question time!