iOS Security Framework

Research Experience

Costin Carabaș

Background

iOS General Information

- Based on Darwin OS (open source)
- XNU kernel - open source
- iOS -> Darwin <=> Android -> Linux
- Closed source software

iOS Applications

Application Development

- Xcode & Objective-C or Swift
- Public frameworks vs Private frameworks
- Public frameworks allowed by developers
- Private frameworks present in System Apps and Services

iOS Applications

Application Runtime

- User: **mobile**
- Filesystem: /var/container/Bundle/ Application/<UUID>
- Apple Sandbox (iOS and macOS)
  - Sandbox **profile** (set of rules)
  - Allow/Deny low-level actions (system calls): file op, IPC, network
  - *Sandblaser (Deaconescu et. al)* reverse to human readable

iOS Applications

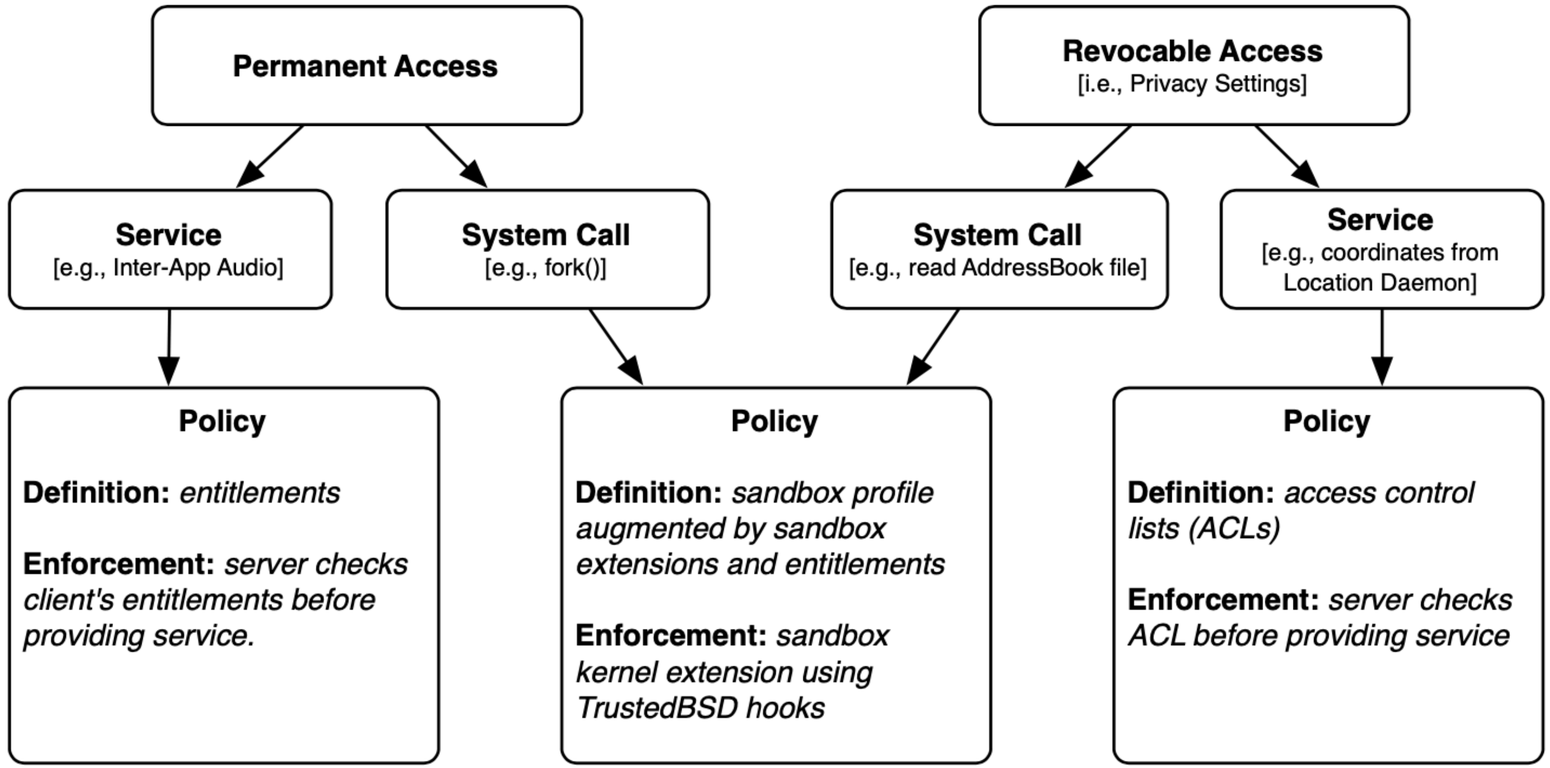
SandBox Profile Language (SBPL)

```
( allow file-read*
  ( require-all
    ( subpath "/Media/Safari" )
    ( require-not
      ( literal "/Media/Safari/secret.txt" )
    )
    ( require-entitlement
      "private.signing-identifier"
      ( require-any
        (entitlement-value "mobilesafari" )
        (entitlement-value "safarifetcherd" )
) ) ) )
```

Mobilesafari or safarifetcherd can read files in /Media/Safari, but they cannot read /Media/Safari/secret.txt

Access control mechanisms

- Entitlements
- Sandboxing
- Privacy Settings



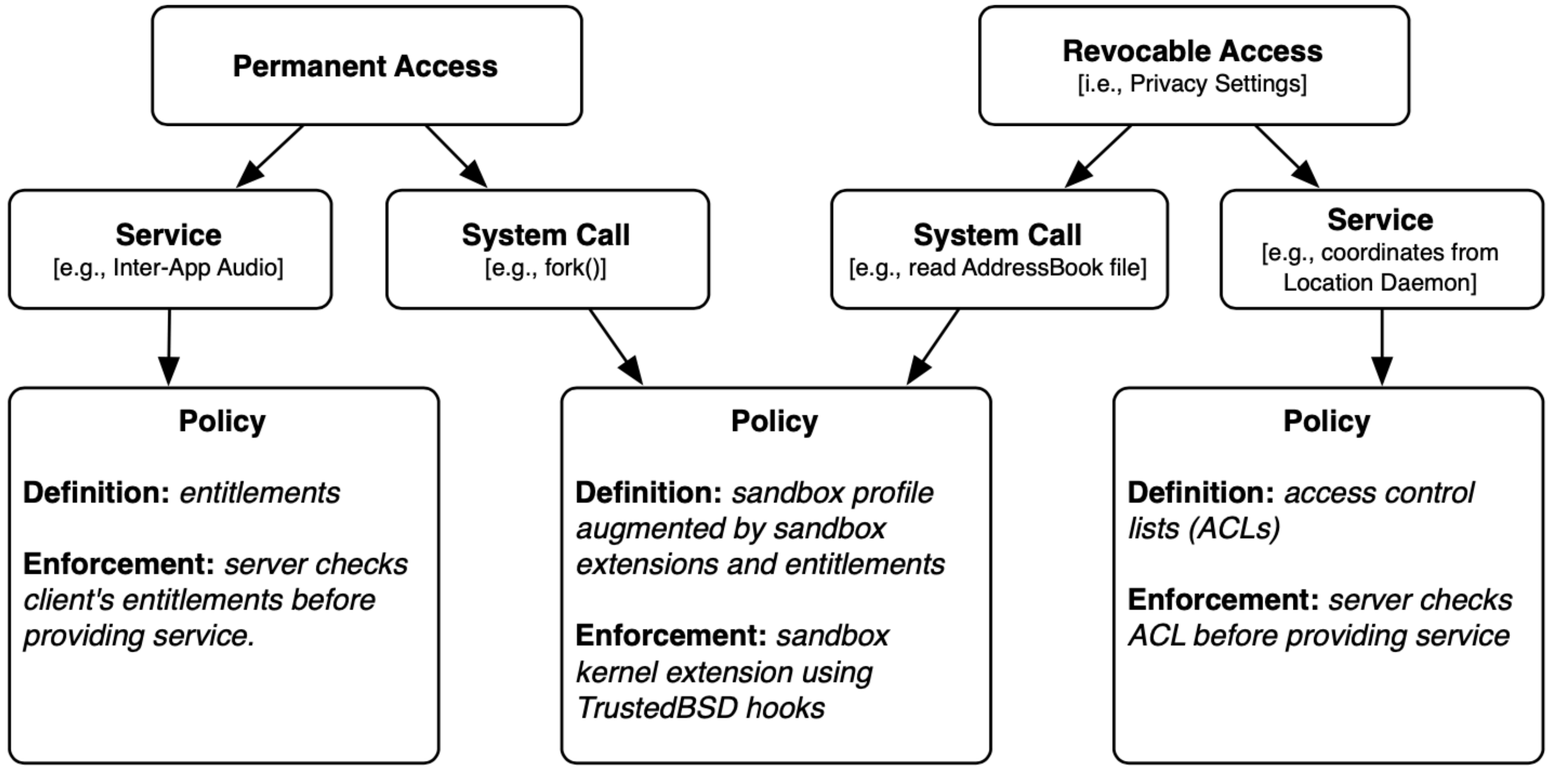
Access control mechanisms

Entitlements

- Permanent (cannot be granted/revoked)
- Cannot be changed at runtime (i.e. by an attacker)
- Can be changed only by officially updating an app
- Trust is moved to developers
- Public, private, semi-private entitlements
- Wang et al. (2013) bypass this limitation and call private APIs (*Jekyll apps*)

Access control mechanisms

- Entitlements
- Sandboxing
- Privacy Settings



Access control mechanisms

SandBox Profile Language (SBPL)

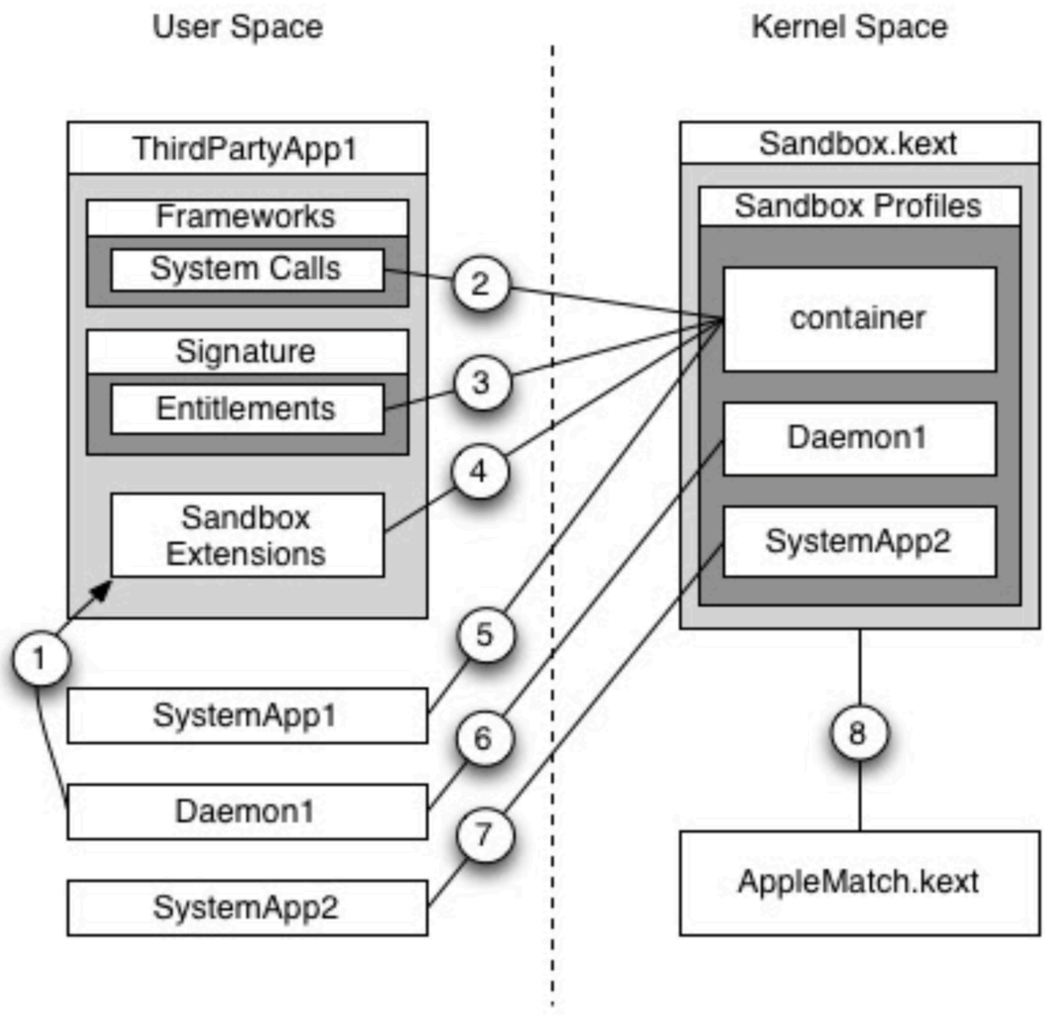
```
( allow file-read*
  ( require-all
    ( subpath "/Media/Safari" )
    ( require-not
      ( literal "/Media/Safari/secret.txt" )
    )
    ( require-entitlement
      "private.signing-identifier"
      ( require-any
        (entitlement-value "mobilesafari" )
        (entitlement-value "safarifetcherd" )
) ) ) )
```

Mobilesafari or safarifetcherd can read files in /Media/Safari, but they cannot read /Media/Safari/secret.txt

iOS Applications

Sandbox Profile

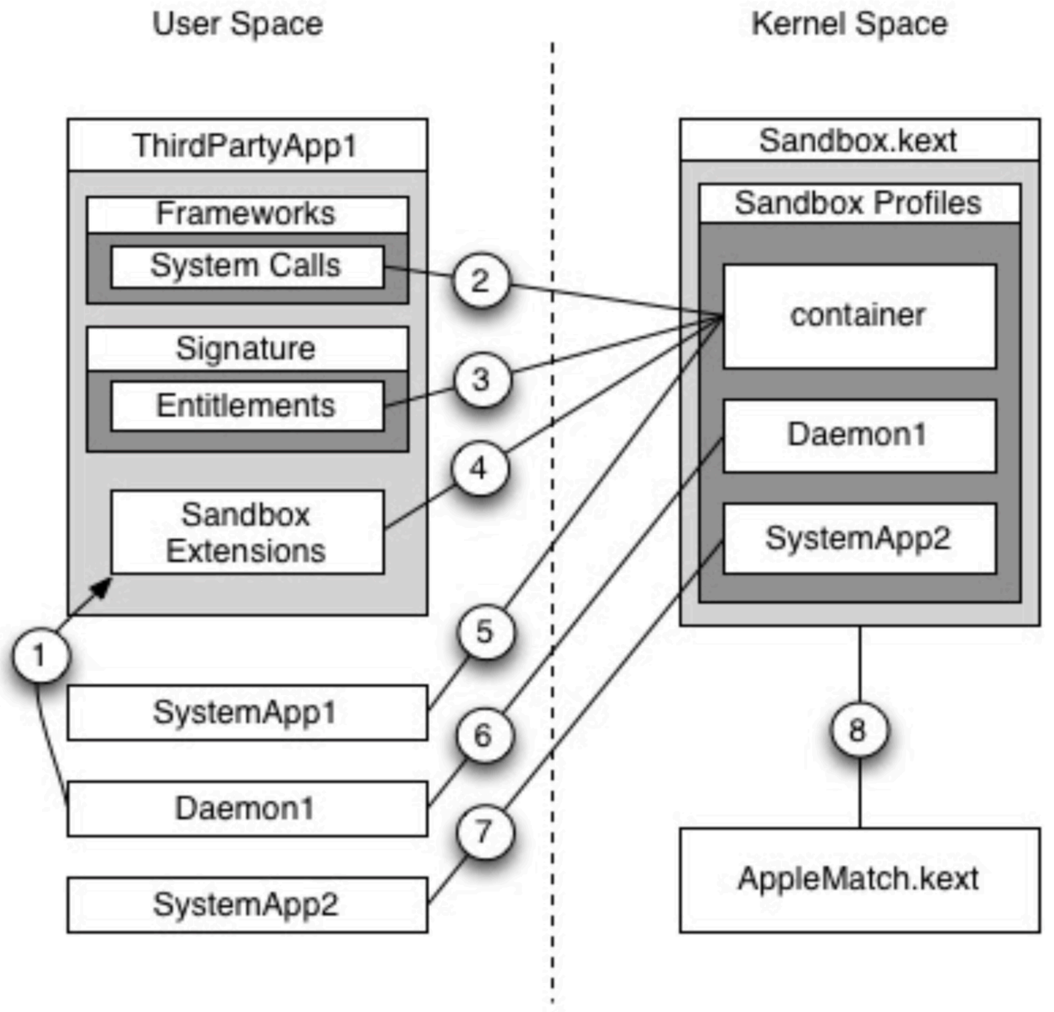
- Written in SBPL (SandBox Profile Language)
- Allows acces via capabilities: **entitlements** and **sandbox extensions**
- **Entitlements** - hardcoded in the signed app executable (static)
- **Sandbox extensions** - non-permanent, revokable tokens (dynamic)



Access control mechanisms

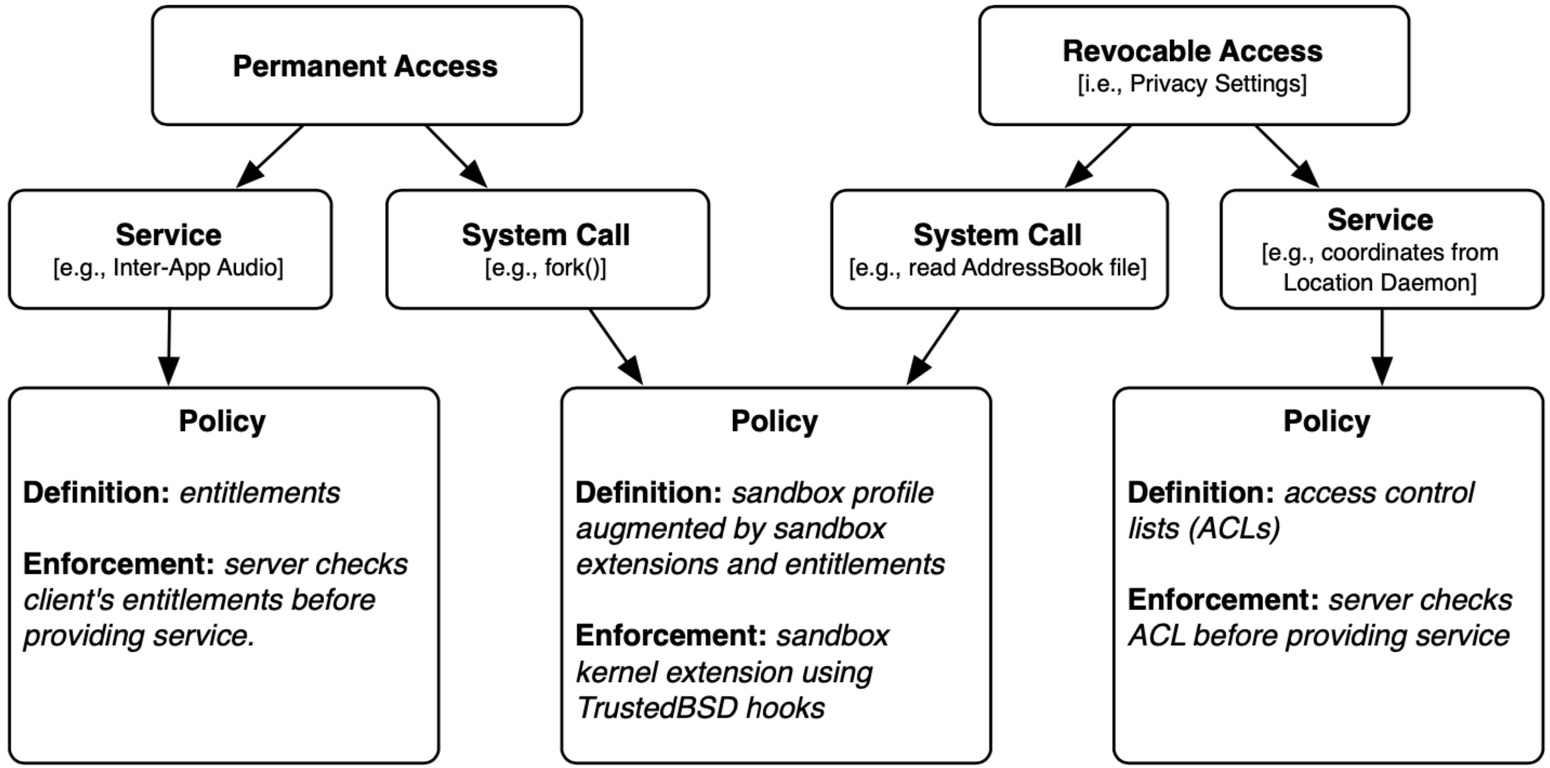
Sandboxing

- Allow/Deny system calls performed by apps (sandboxed processes)
- Sandbox Kernel Extension - decision maker
- Each app - sandbox profile
- *Adding Custom Sandbox Profiles to iOS Apps (Mad Corneci et. al)*
- *Sandscout (Deshotels et. al)* - 6 CVEs in **container** sandbox profile



Access control mechanisms

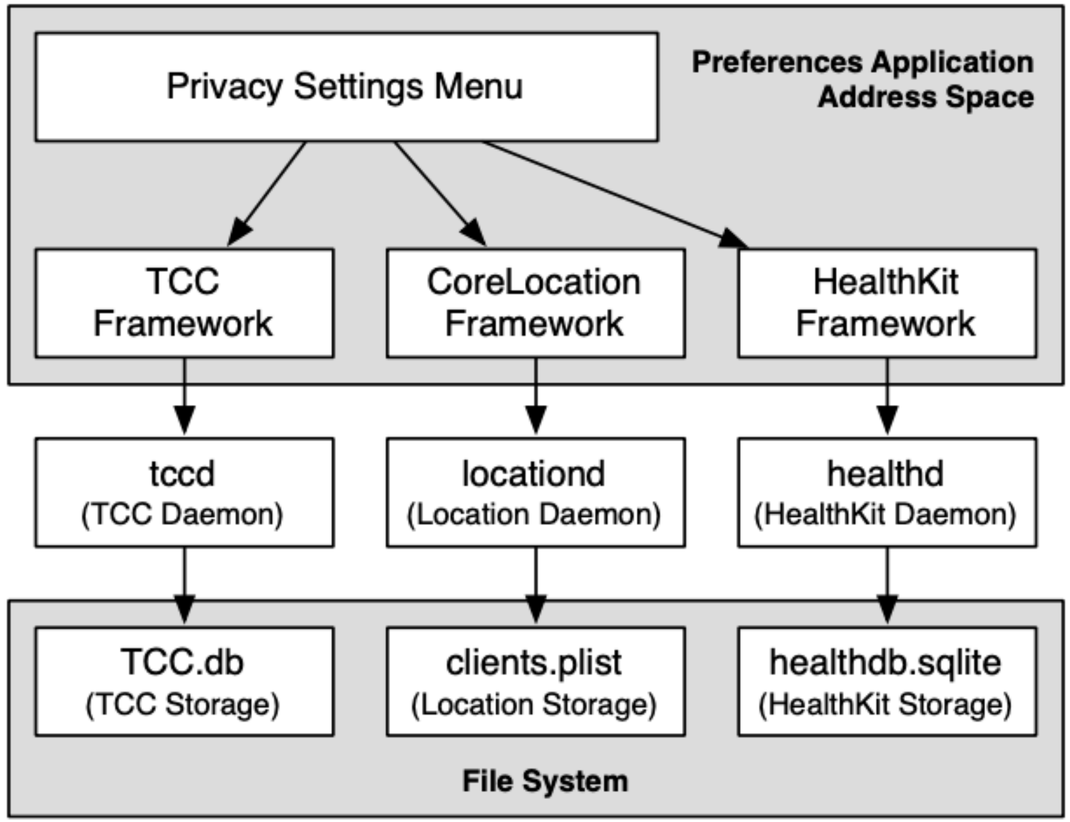
- Entitlements
- Sandboxing
- Privacy Settings



Access control mechanisms

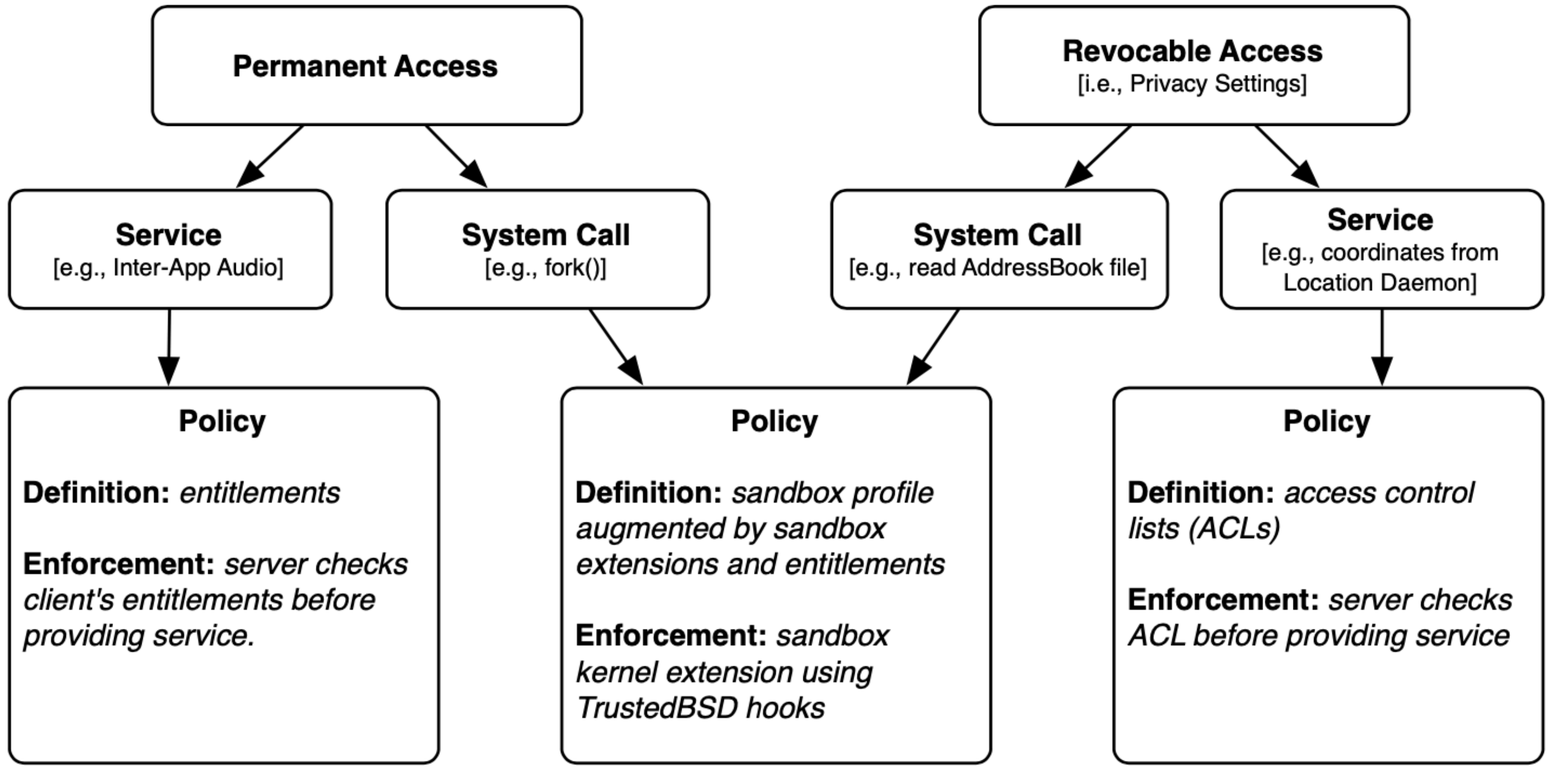
Privacy Settings

- Settings App -> Privacy
- Can be changed at any time by the user
- User-friendly
- Managed by Transparency, Consent and Control (TCC)
- Apps work even without permissions from Privacy Settings (Google Maps)
- Grants sandbox extensions to apps



Access control mechanisms

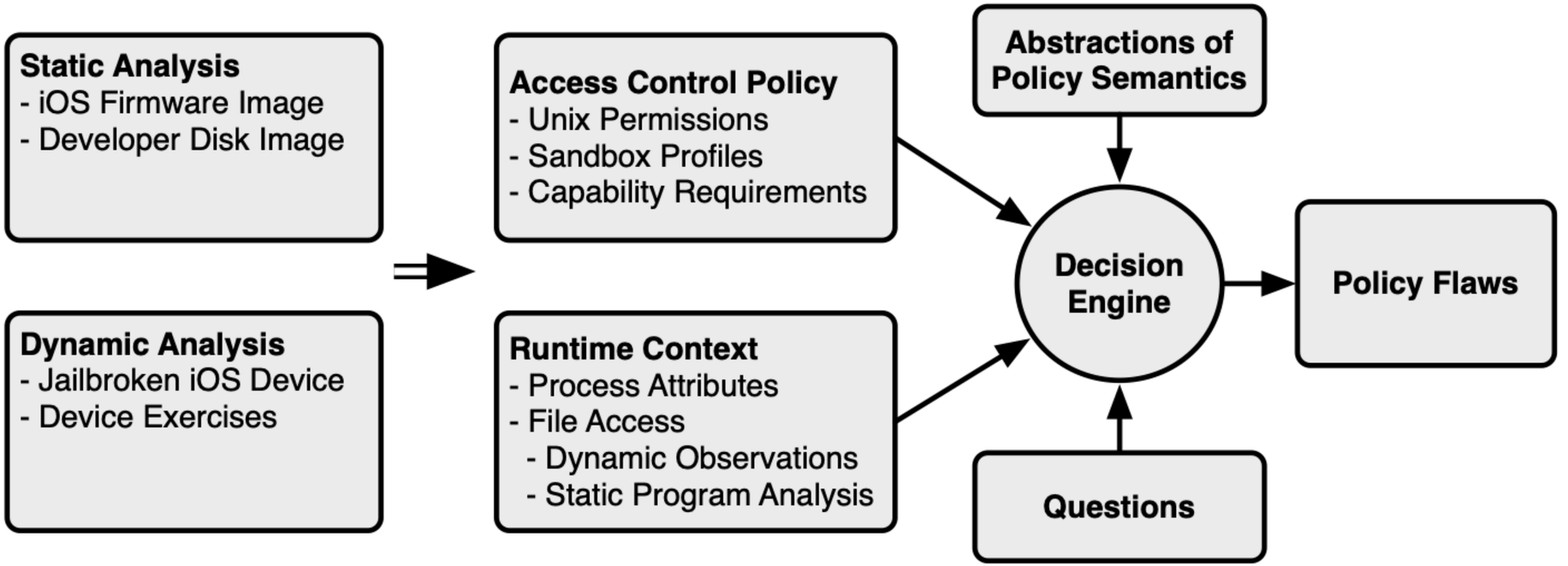
- Entitlements
- Sandboxing
- Privacy Settings
- UNIX permissions



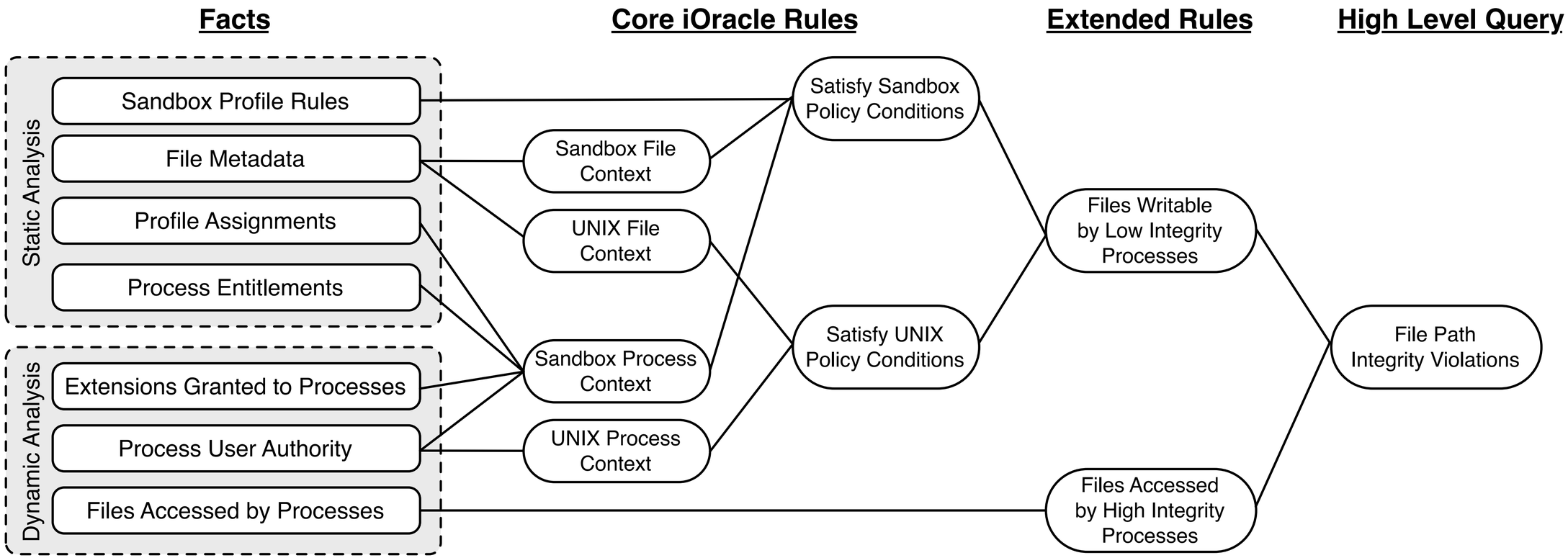
iOracle

- Models policies and runtime context
- Access control mechanisms
- iOS firmware
- Developer resources
- Jailbroken devices
- iOracle: Automated Evaluation of Access Control Policies in iOS (Deshotels et. al)

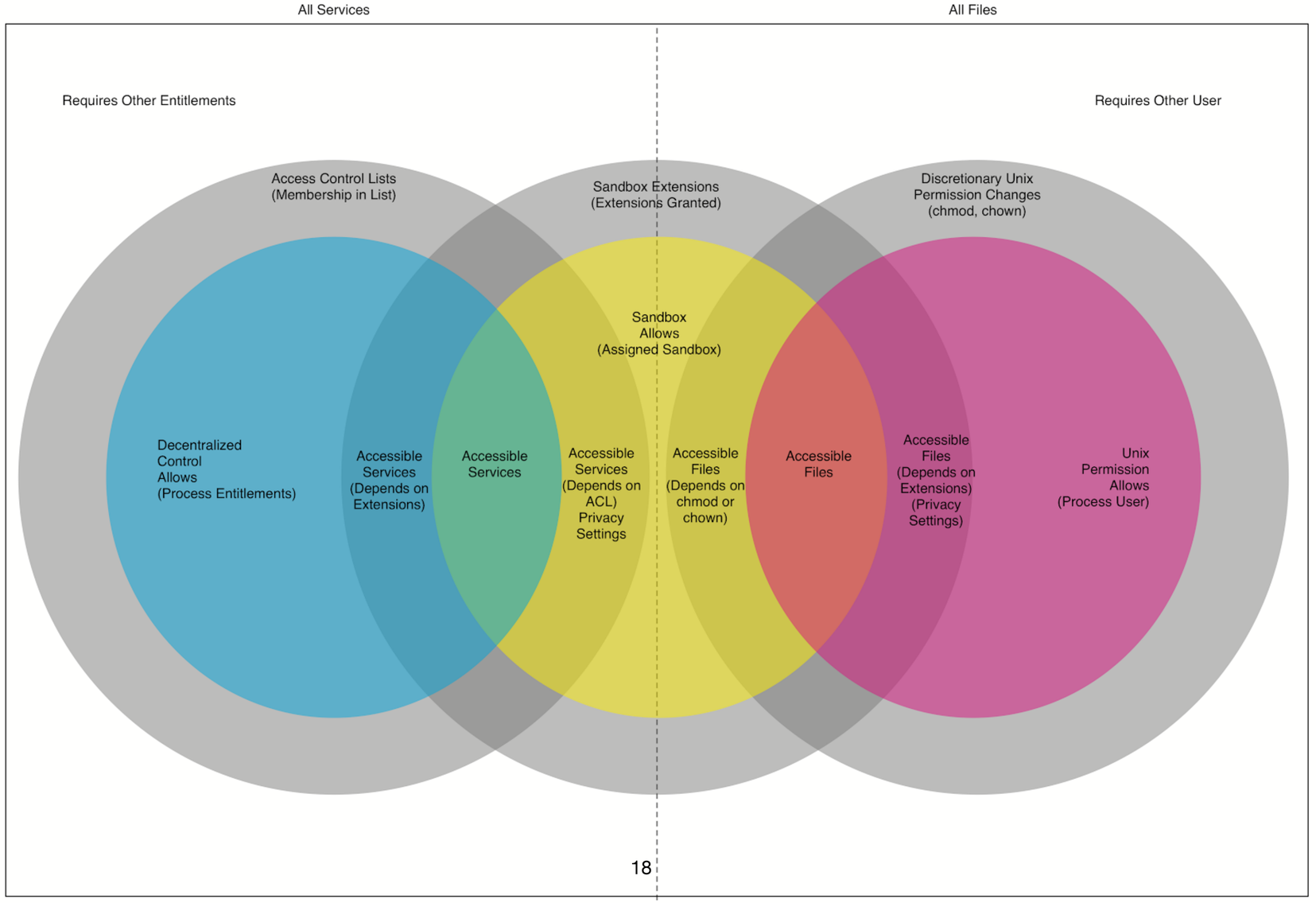
iOracle: Automated Evaluation of Access Control Policies in iOS



iOracle: Automated Evaluation of Access Control Policies in iOS



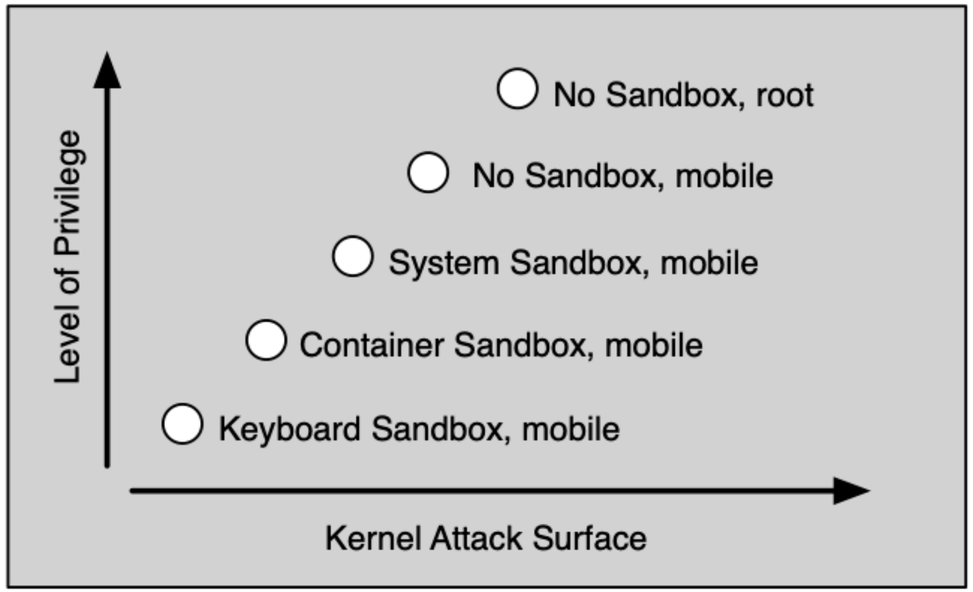
iOracle: Automated Evaluation of Access Control Policies in iOS



iOracle - Model

- Define tables of facts
  - parent(alice,bob)
  - parent(bob,charlie)
- Define rules that abstract those facts
  - grandparent(A,C):- parent(A,B), parent(B,C).
- Make queries about facts and rules
  - ?- grandparent(alice,X).
  - Query is satisfied by matching X to Charlie
- Target: ?- access(process(Proc),operation(“file-read”),file(“**superSecret.txt”**))

iOracle: Automated Evaluation of Access Control Policies in iOS

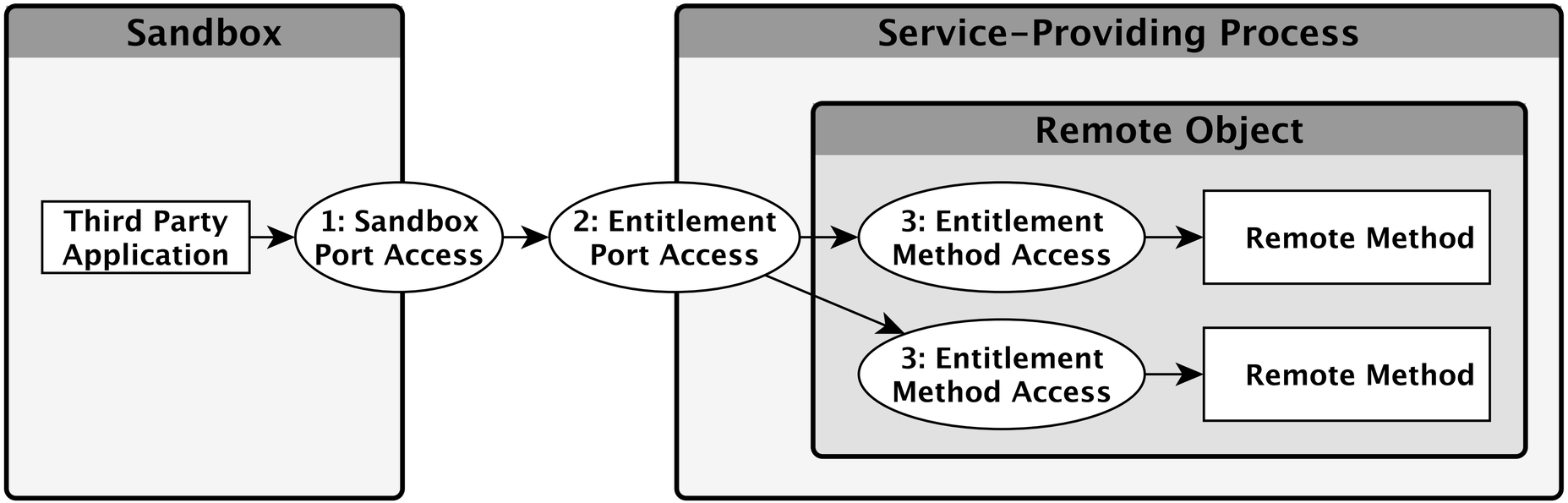


iOS Runtime

Inter-Process Communication

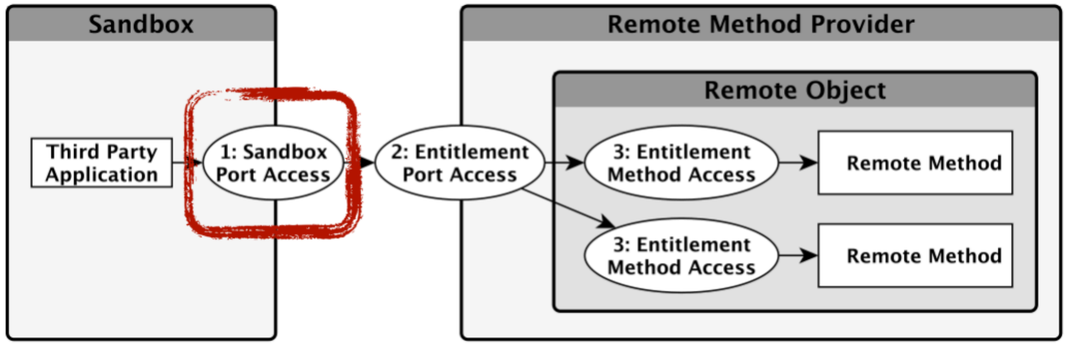
- Mach Ports
- Distributed Notifications
- Distributed Objects
- AppleEvents & AppleScript
- Pasteboard
- **XPC** (**Kobold, IEEE S&P 2020**) - 3 CVEs

Kobold


Sandbox
Third Party Application
1: Sandbox Port Access
Service-Providing Process
2: Entitlement Port Access
Remote Object
3: Entitlement Method Access
Remote Method
3: Entitlement Method Access
Remote Method

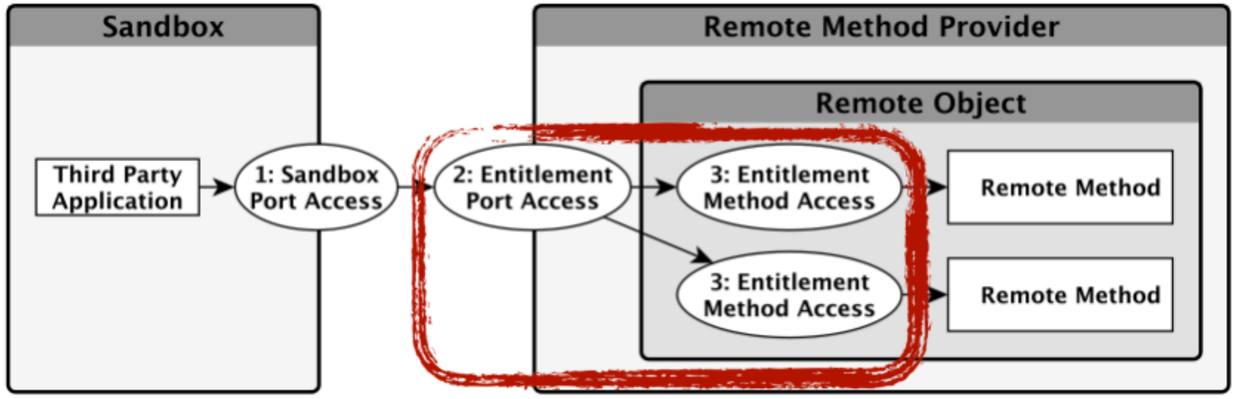
Kobold

```
(allow mach-lookup(
  require-all
   (global-name "siri.vocabularyupdates")
   (require-entitlement “com.apple.developer.siri”)
 )
)
```



Kobold

```
//psuedocode for entitlement check in daemon logic
entitlement_key = "health.stats"
entitlement_value = "weight"
if check_value(entitlement_name, client) == entitlement_value:
    send_weight(client)
else:
    send_error(client, "missing entitlement")
```

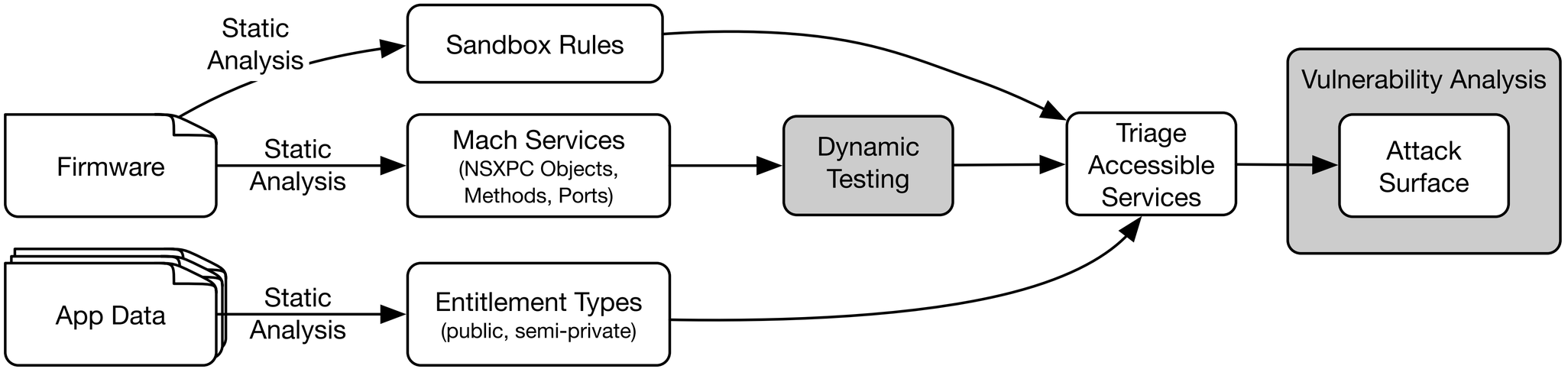


Kobold

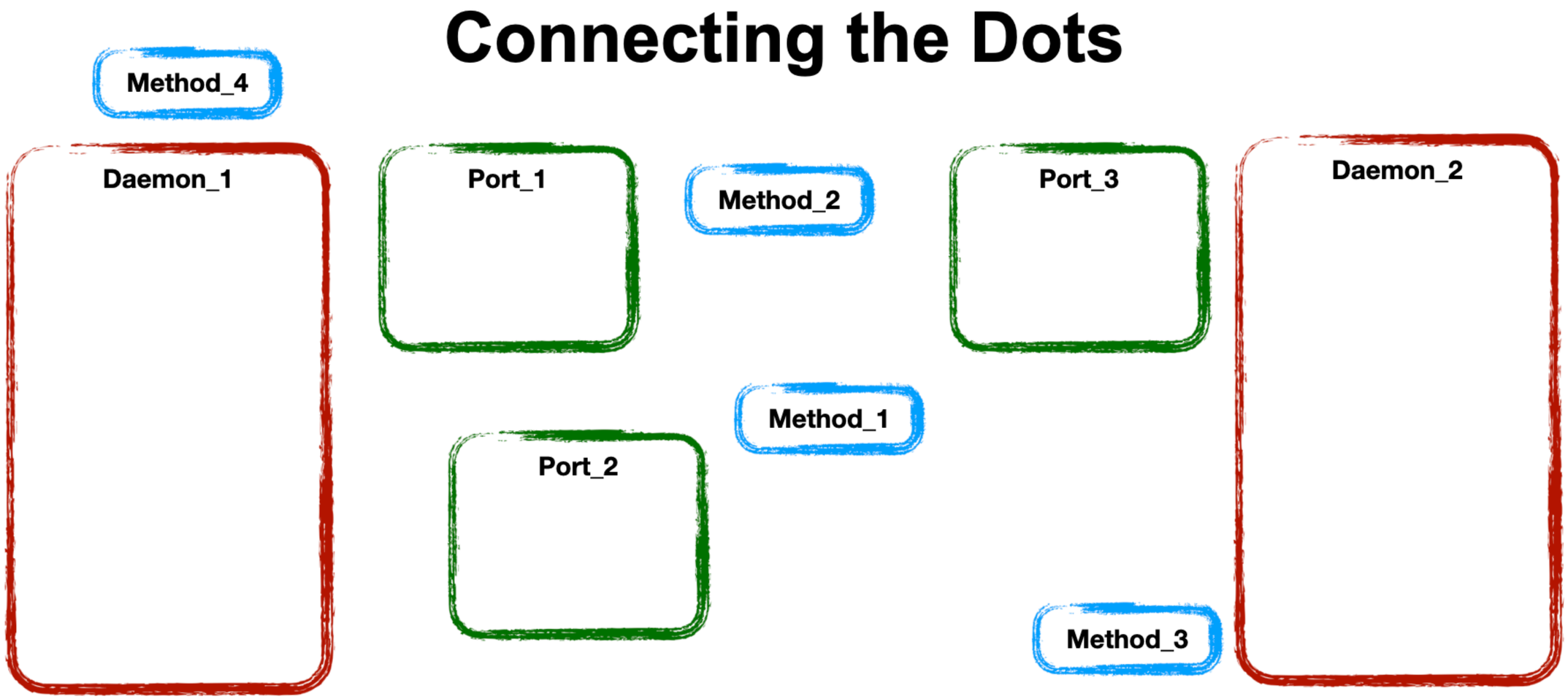
Research Questions

- Which NSXPC methods are accessible to third party apps?
- Which entitlements are available to third party apps?
- Of these accesible NSXPC methods, which are dangerous?

Kobold

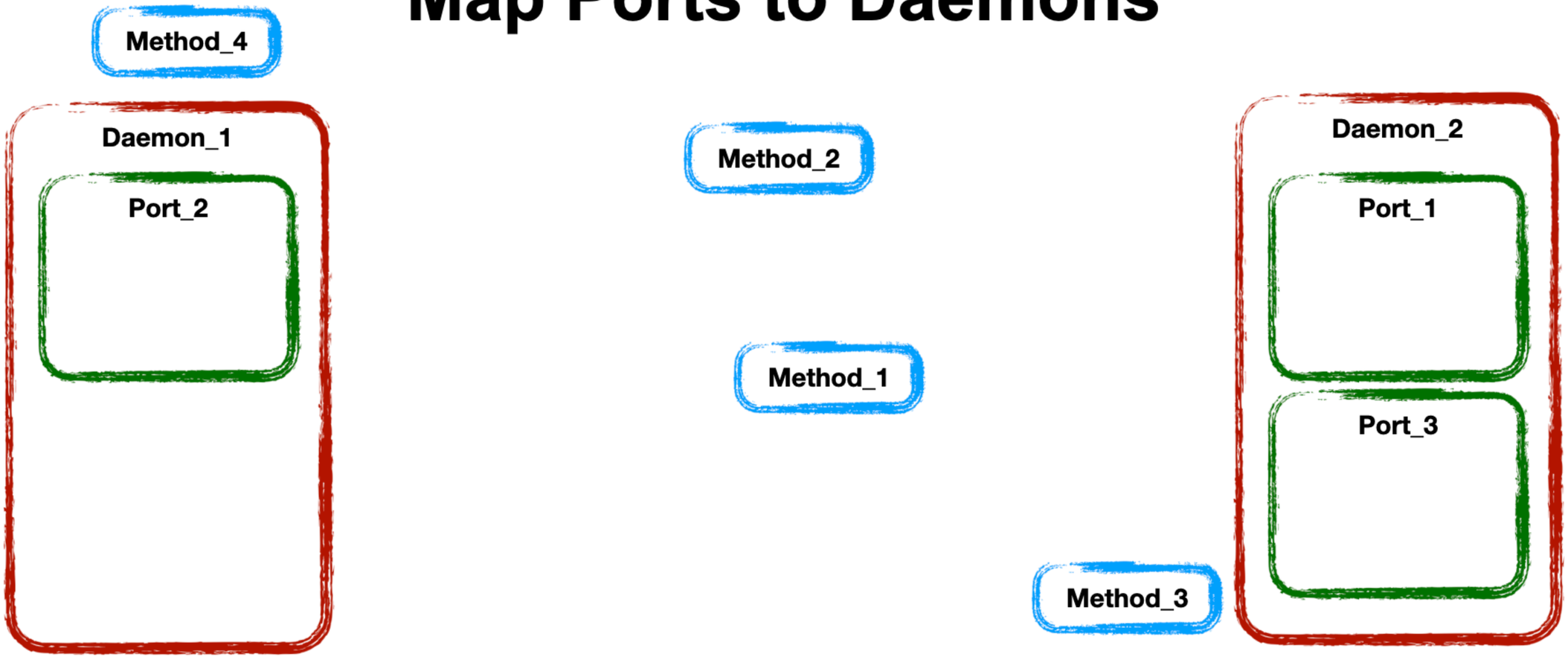

Static
Analysis
Sandbox Rules
Firmware
Static
Analysis
Mach Services
(NSXPC Objects,
Methods, Ports)
Dynamic
Testing
Triage
Accessible
Services
Vulnerability Analysis
Attack
Surface
App Data
Static
Analysis
Entitlement Types
(public, semi-private)

Kobold


Connecting the Dots
Method_4
Daemon_1
Port_1
Method_2
Port_3
Daemon_2
Method_1
Port_2
Method_3

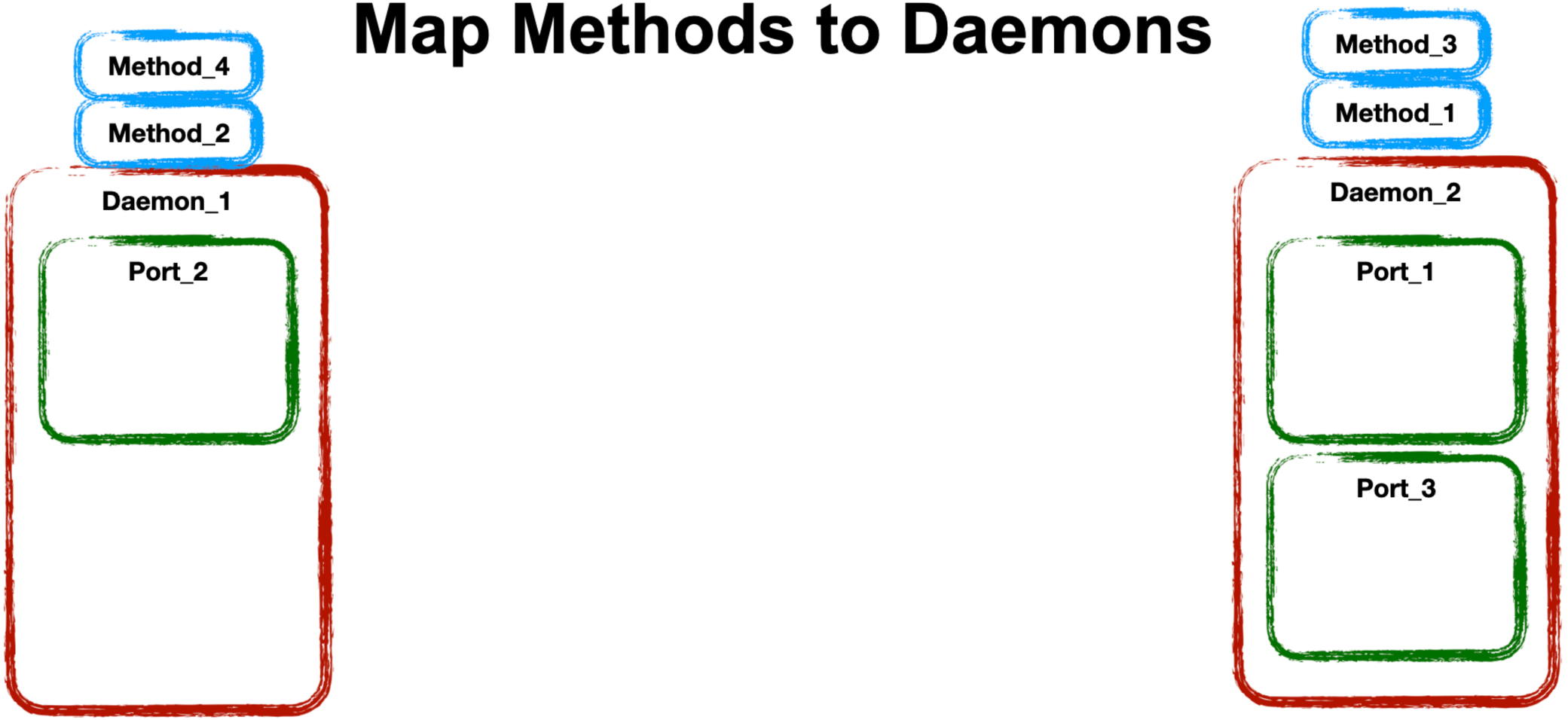
Kobold

Map Ports to Daemons

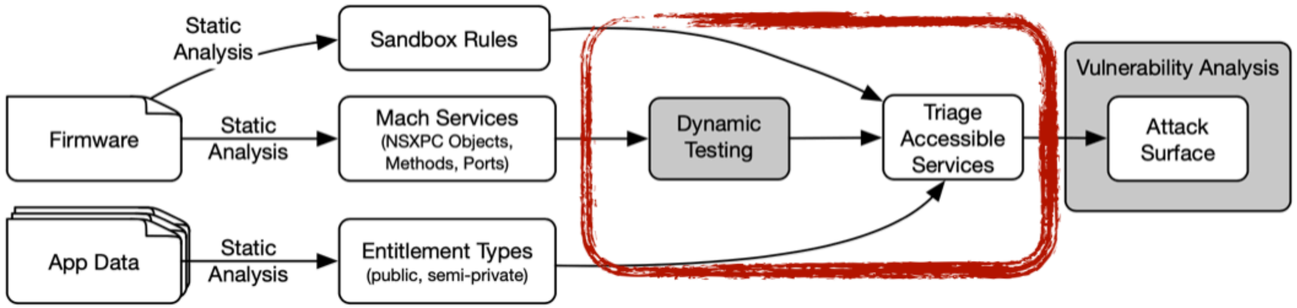


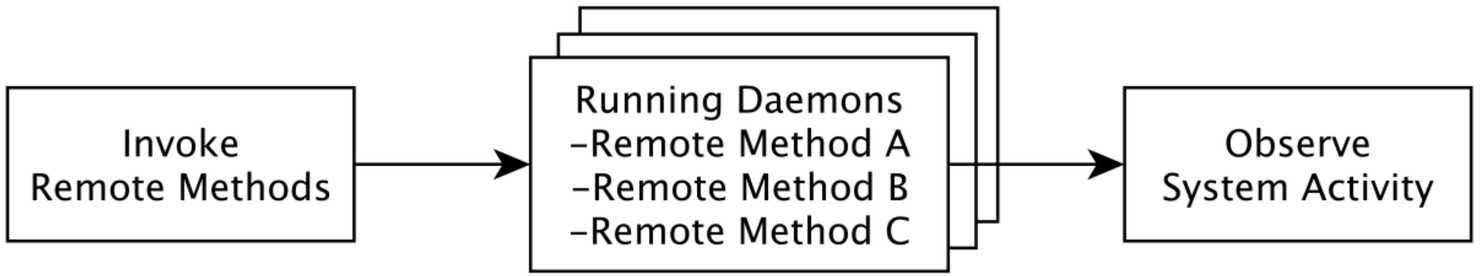
Kobold

Map Methods to Daemons



Kobold


Static Analysis
Sandbox Rules
Firmware
Static Analysis
Mach Services (NSXPC Objects, Methods, Ports)
Dynamic Testing
Triage Accessible Services
Vulnerability Analysis
Attack Surface
App Data
Static Analysis
Entitlement Types (public, semi-private)


Invoke Remote Methods
Running Daemons
-Remote Method A
-Remote Method B
-Remote Method C
Observe System Activity

Kobold

Running the Code

Kobold

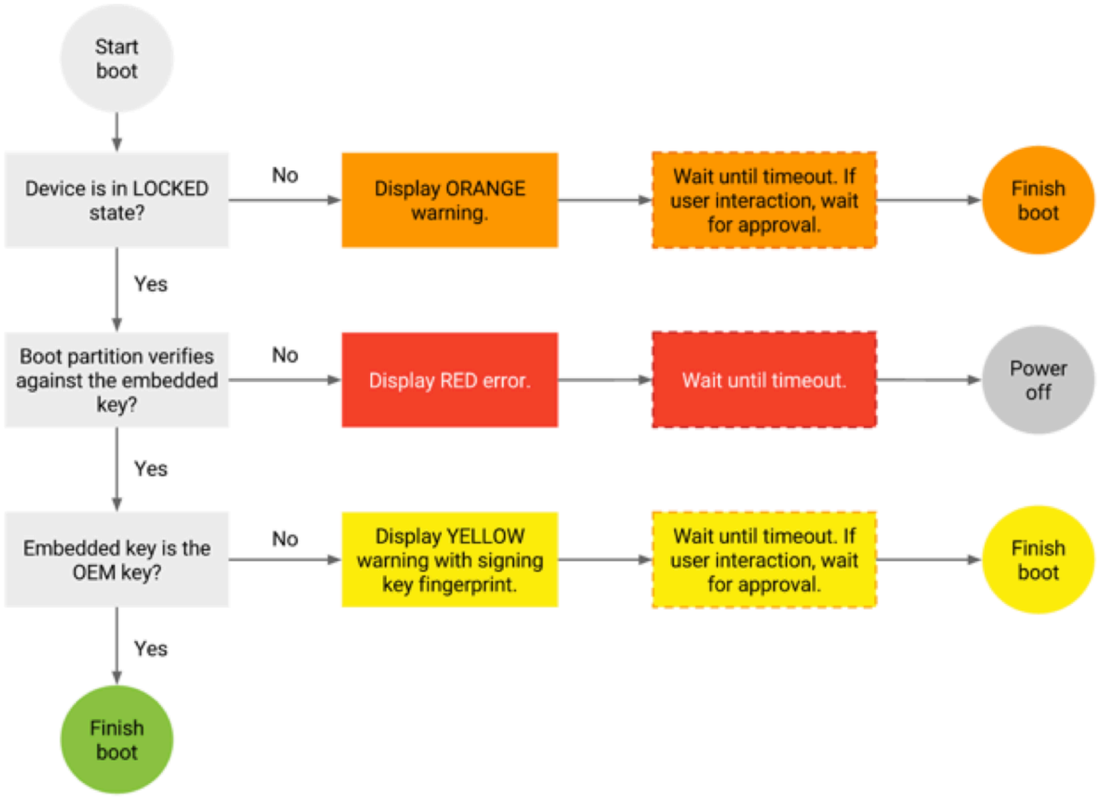
Effect	Method	Mach-Port
Leak names of installed apps with File Providers	dumpStateTo:completionHandler:	com.apple.FileProvider
Device fingerprinting	dumpStateTo:completionHandler:	com.apple.FileProvider
Infer file names in File Providers	dumpStateTo:completionHandler:	com.apple.FileProvider
Activate microphone	startRecordingFor PendingDictationWithLanguageCode: options:speechOptions:reply:	com.apple.assistant.dictation
Disable Text Replacement	setKeyboardShortcutsAllowed:completion:	com.apple.managedconfiguration.profiled.public
Disable Dictation	setDictationAllowed:completion:	com.apple.managedconfiguration.profiled.public
Block access to all websites	addBookmark:completion:	com.apple.managedconfiguration.profiled.public

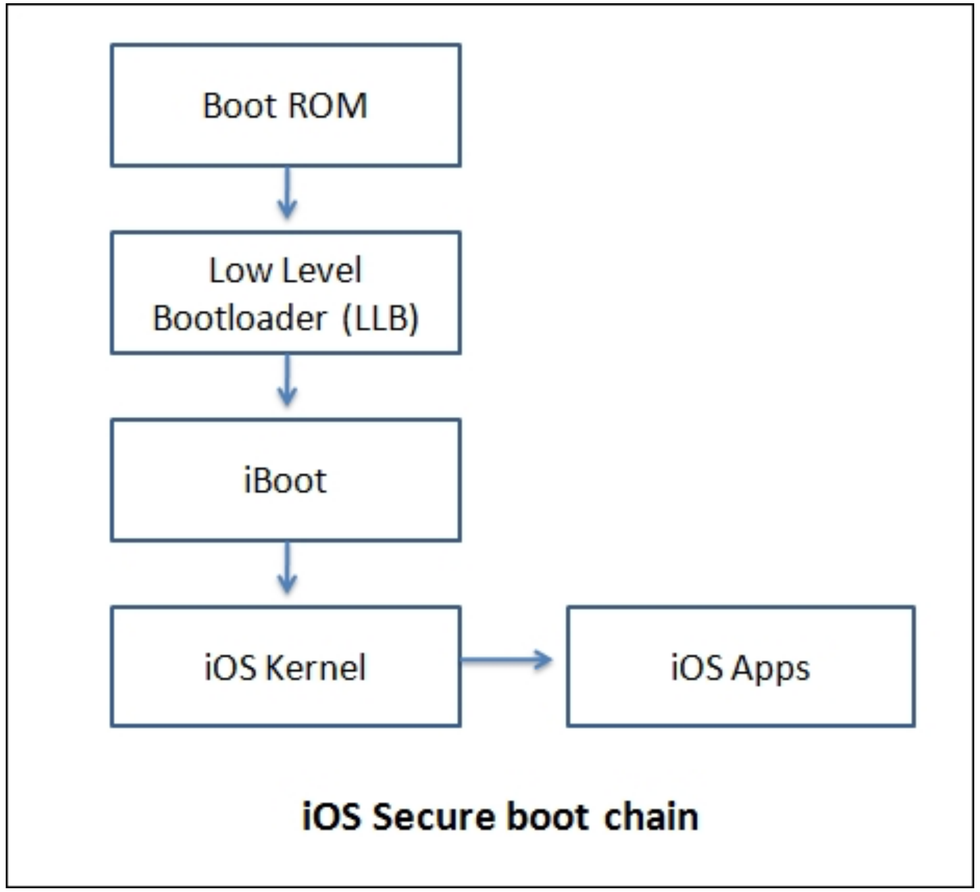
Android vs iOS

Android vs iOS

- **Boot up process**

Android

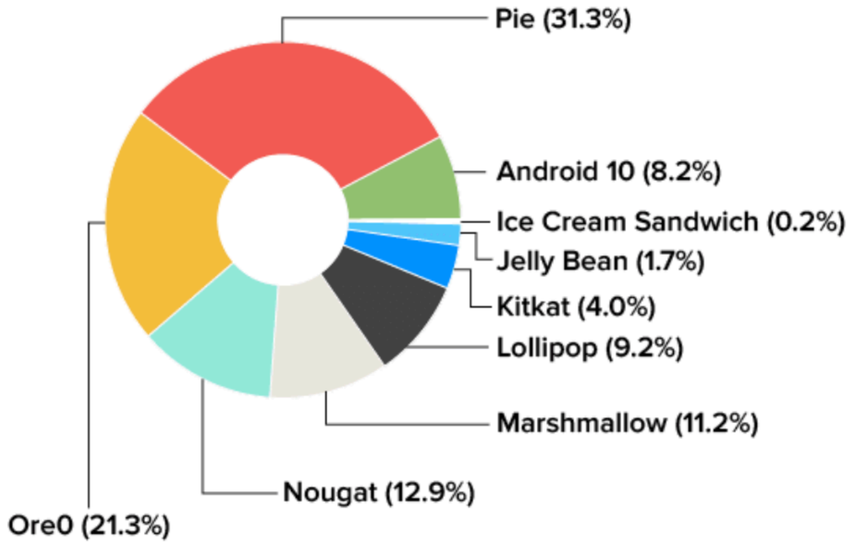




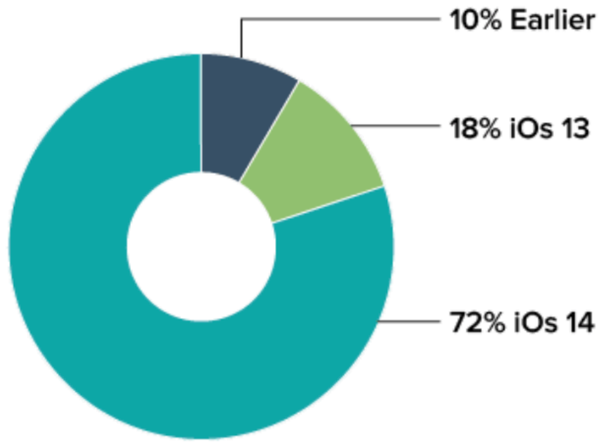
Android vs iOS

Mobile OS fragmentation

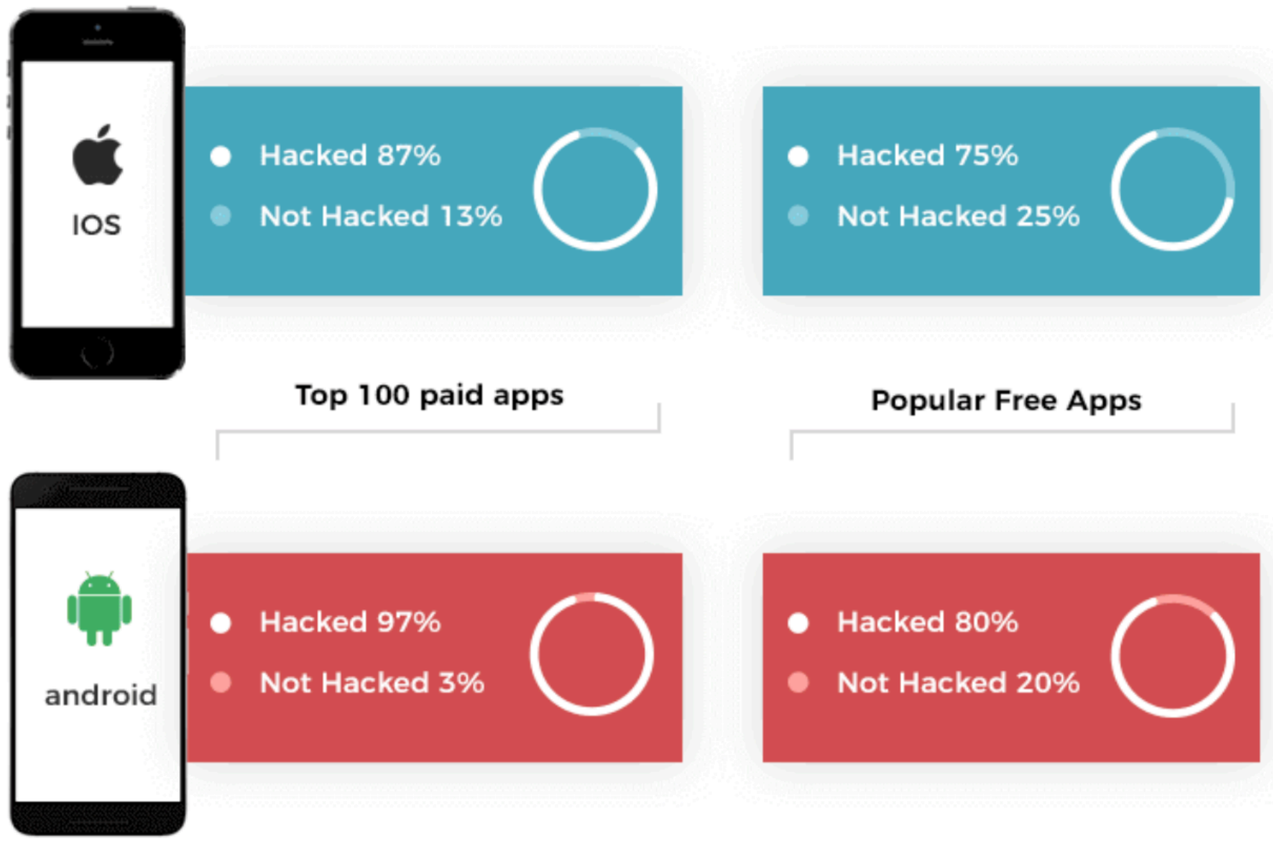
Android



iOS



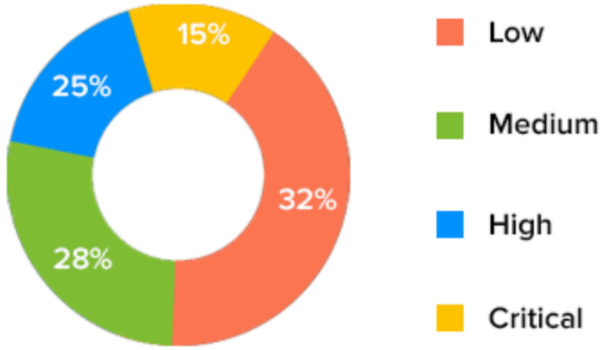
Android vs iOS

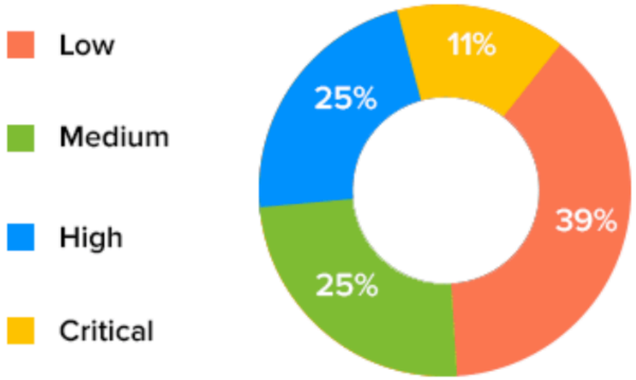


Android vs iOS

40%

of the detected vulnerabilities on iOS tested applications were found to be critical or high severity





36%

of the detected vulnerabilities on Android tested applications were found to be critical or high severity

Summary

- iOS Overview
- Application Development
- Application Runtime
- Sandbox Profile
- Privacy Settings
- General access to System Resources
- IPC
- Protecting User Data

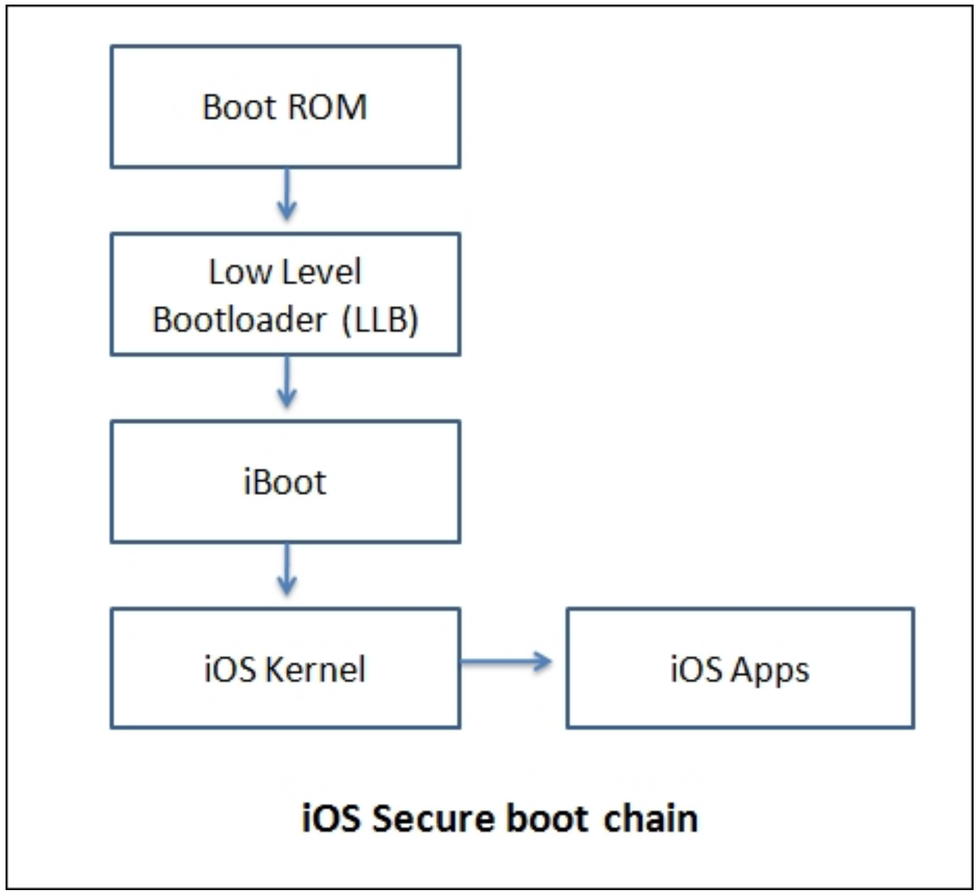
Resources:

https://www.theiphonewiki.com

https://github.com/malus-security

iOS - Secure Boot

- Bootrom
  - read-only program
  - Stores the Apple root CA public key
  - Loads iBoot
- Bootloader
  - iBoot
  - Checks the integrity of OS Kernel
  - Loads the kernel



- Bypassing the kernel integrity check - Jailbreak (rooting in Android)
- Usually **after** Secure Boot

Protecting User Data

Authentication

- Boot time:
  - Protected by hardware-based encryption
  - Decryption key available if PIN entered
  - Failure to enter PIN - all data lost
  - PIN not stored on device (one-way function -> key stored)
- After booting:
  - Touch ID, Face ID - stored on the device (Secure Enclave)

Protecting User Data

Secure Enclave

- Hardware-based key manager
- Similar to ARM’s TrustZone + proprietary code
- Runs on Secure Enclave Processor (SEP) - separated from app CPU
- Used by Apple Pay