

# Apple / iOS Reversing & Sandboxing

SMD, May 22, 2023

# Agenda

Overview of iOS

Reversing iOS

Overview of sandboxing

Apple (iOS) sandboxing

Reversing the Apple sandbox

Assessing / Exploiting the Apple sandbox

# iOS

popular mobile OS solution

iOS1 - 2007

yearly release

iOS16 - 2022

custom hardware, custom OS, security features

AppStore

# Security in iOS

[https://help.apple.com/pdf/security/en\\_US/apple-platform-security-guide.pdf](https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf)

app sandboxing (MAC)

privacy settings

access control

app signing, app encryption

SEP, SepOS

TouchID, FaceID

# iOS Components

firmware image

OTA updates

kernel cache

- extensions

dyld shared caches

system applications

system configurations

# iOS Reversing

<https://github.com/malus-security/iextractor>

<https://github.com/malus-security/kextractor>

<https://github.com/malus-security/sandblaster>

download firmware files

extract: kernel cache, executables, dyld\_shared\_cache, configuration files

reverse binary files (kernel cache, executables, dyld\_shared\_cache, databases)

dynamic information: processes, XPC calls, system calls, sandbox calls

# Mindset & Methodology of Reversing

always be smashing

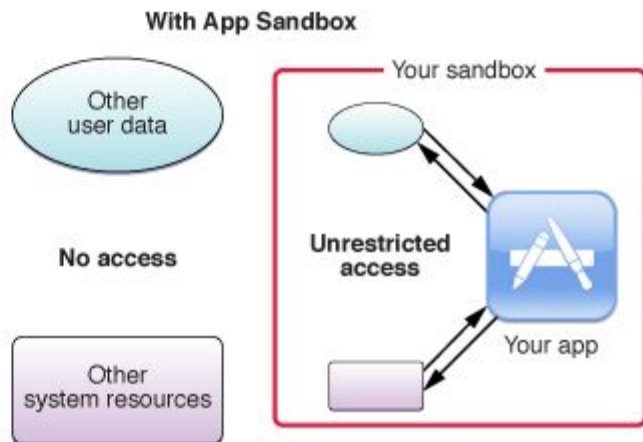
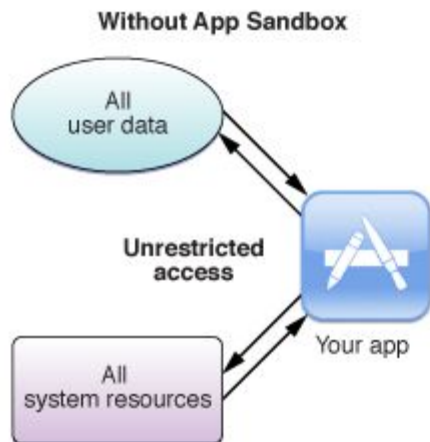
stand on the shoulder of giants

<https://www.theiphonewiki.com/>

[https://iphonedev.wiki/index.php/Main\\_Page](https://iphonedev.wiki/index.php/Main_Page)

BlackHat (and other cybersec conferences) talks

reverse, rebuild, check resulting binary with original one





# Sandboxing Questions

How to define sandbox rules?

How to store sandbox rules?

How to apply sandbox rules?

# Linux Answers

Define sandbox rules as system calls + argument filters

Store sandbox rules as commands inside program to be sandboxed

Triggered by voluntary sandbox calls from sandboxed program, applied by kernel

# Apple Sandboxing Answers

Defined as SBPL (Sandbox Profile Language) Rules (TinyScheme)

Stored in predefined sandbox profile files (.sb)

Applied voluntarily by program or enforced by kernel

# SBPL Snippet

```
(allow system-info
  (require-all
    (info-type "net.link.addr")
    (require-entitlement "fairplay-client")
    (require-not (require-entitlement "com.apple.private.MobileGestalt.AllowedProtectedKeys"))))
(allow system-privilege)
(allow system-socket
  (socket-domain AF_ROUTE)
  (require-all
    (socket-domain AF_SYSTEM)
    (socket-protocol 2))
  (require-all
    (socket-domain 39)
    (require-entitlement "com.apple.private.signing-identifier"
      (entitlement-value "com.apple.Maps"))))
```

# Mapping sandbox profiles to applications in iOS

As part of manifest file (xml, .plist)

When program is loaded, the manifest file is read and the corresponding profile is loaded

3rd party applications do not have a sandbox profile in the manifest file

the container.sb sandbox profile is applied by default

3rd party applications are matched by their location in the filesystem

# Storing & Applying iOS Sandbox Profiles

Part of the kernel cache extension (Sandbox.kext)

Binarized (compiled) and bundled together (since iOS 9)

When application is loaded (i.e. `exec()` call) - the corresponding binary sandbox profile is applied

Sandbox.kext stores the code to interpret and apply the sandbox rules

# Customizing container.sb

container.sb applied to **all** 3rd party apps: too many applications

Making it customizable / specific:

- entitlements: hard-coded key-values in manifest files

- extensions: temporary capabilities provided during runtime (may be revoked)

## Customizing container.sb (2)

```
(require-entitlement "com.apple.private.signing-identifier"  
  (entitlement-value "com.apple.webbookmarksd"))
```

```
(require-all  
  (subpath "/private/var/mobile/Media")  
  (extension "com.apple.avasset.read-only")  
  (extension-class "com.apple.mediaserverd.read")  
  (extension "com.apple.tcc.kTCCServicePhotos"))
```



# Our work (Malus Security)

Reversing iOS binary sandbox profiles (Sandblaster)

Assessing iOS sandbox profiles (Sandscout)

# Sandblaster

<https://github.com/malus-security/sandblaster>

Python

Extracts and reverses (decompiles) binary sandbox profiles

Returns SBPL profiles

# iExtractor, iExtractor-manager

<https://github.com/malus-security/iExtractor>

<https://github.com/malus-security/iExtractor-manager>

shell scripting, Python

downloads IPSW files and extracts all required files to be used by Sandblaster  
(and other tools)

# Sandscout

<https://github.com/malus-security/sandscout>

Python, Prolog

processes reversed sandbox rules as Prolog files

applies queries (on security invariants) to detect flaws

# Findings

<https://www.google.com/search?channel=fs&client=ubuntu&q=cve+ios+deshotels+deaconescu>

created PoC attacks based on Sandscout Prolog queries

submitted to Apple (responsible disclosure)

published as CVEs

# Malus Security

<https://github.com/malus-security/>

<https://discord.gg/m3qjuYHYw9>