

Hack in the (sand)Box

(The Apple Sandbox - five years later)

Jonathan Levin

<http://NewOSXBook.com/>

<http://technogeeks.com>

The Apple Sandbox

- Introduced way back in Mac OS 10.5 as “Seatbelt”
 - Very naive implementation originally, bypassed and opt-in
- Revamped in Mac OS 10.7 as “The App Sandbox”
 - Stronger implementation, introducing containers
 - Opt-in for Apple’s own binaries and apps
 - Mandatory for Mac App Store apps (but not for DMG based)
- Far stronger still in iOS
 - Mandatory for all third party applications
 - Evolved beyond MacOS implementation

Sandbox versions

Version	OS Version	Notable Features
..	OS X 10.5/iOS 1-3	Initial version, white list approach
..	OS X 10.6/iOS 4	
165	OS X 10.7/iOS 5	Basic containers
220	OS X 10.8/iOS 6	Sandbox exceptions
278-300	OS X 10.9/iOS 7	IOKit get property, vnode renaming
358	OS X 10.10/iOS 8	Rootless (introduction, non-enforcing), get-task, AMFI integration (in OS X version), kexts (kind of)
460	OS X 10.11/iOS 9	Rootless enforcement, container manager Host special ports, kexts, OSX NVRAM finally protected Policy moved to __DATA.__const (iOS 9.2)
592	OS X 10.12/iOS 10	Container Manager enforcement (iOS) User data items

So Why Are We Here?

- Last actual research conducted in 2011:
 - Dionysus Balazakis seminal work - “The Apple Sandbox”
- Very little further research – partial, unpublished or both
- Sandbox has evolved by leaps and bounds
 - Further evolves in iOS 10 and MacOS 12
 - Provides “System Integrity Protection” as of MacOS 11 (not yet iOS)
- Provides first, strongest, and sometimes last line of defense
 - Tons of exploitable bugs in services and kexts blocked by sandbox
 - Breaking out of the sandbox is toughest stage of jailbreaking.
 - ... And eight of you here voted for this talk 😊

Plan

- Prerequisite: MACF
- MacOS (“App Sandboxing”)
- *OS (Containers)
- Reversing (MacOS, iOS implementations)
- Sandbox APIs

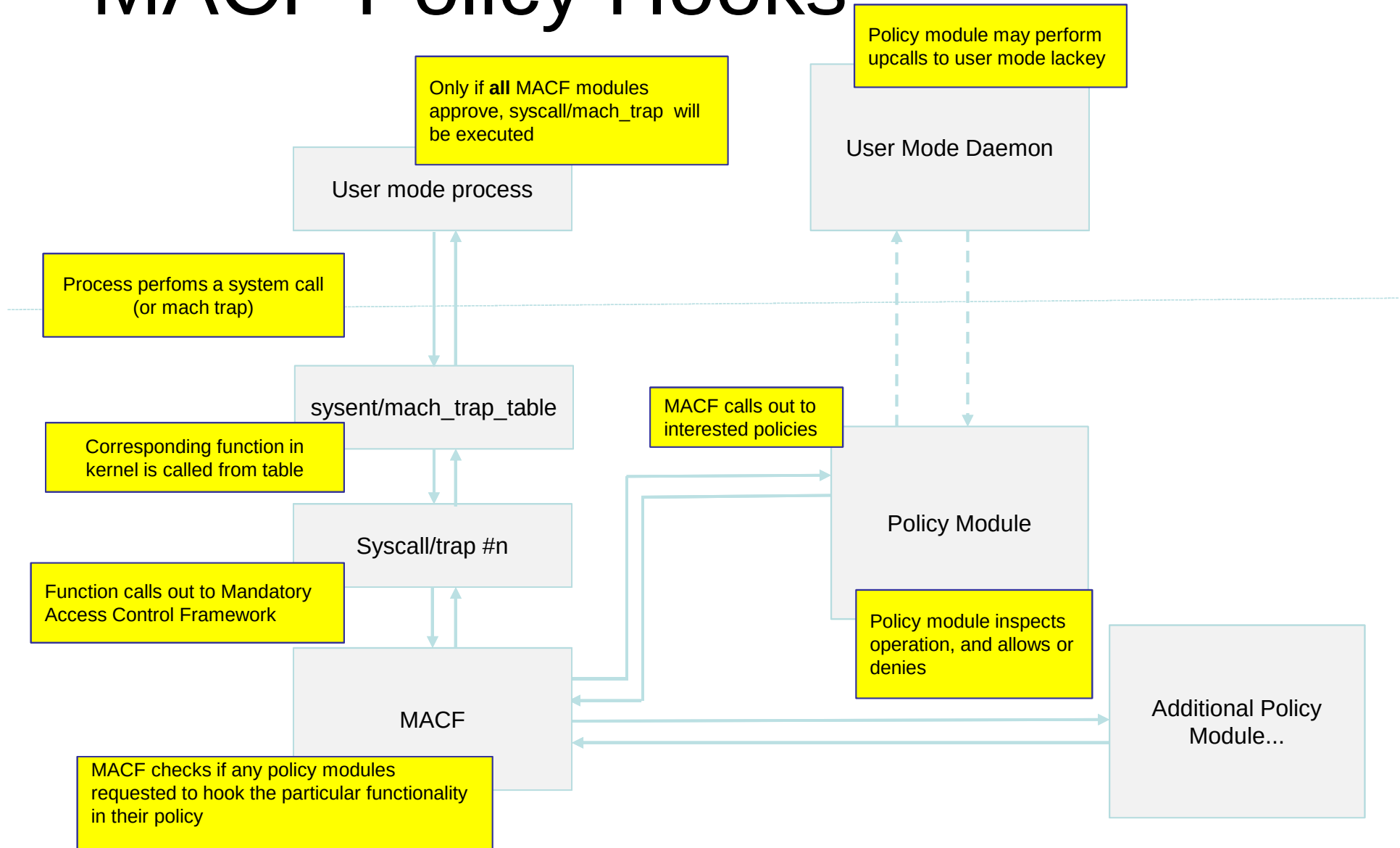
You’re welcome to follow along:

<http://NewOSXBook.com/articles/hitsb.html>

Prerequisite: MACF

- Mandatory Access Control Framework serves as substrate
 - XNU's implementation modeled after FreeBSD's
 - Compare - SELinux/SEAndroid
- Simple idea, powerful impact:
 - Kernel extensions provide a “policy” and call `mac_policy_register`
 - Policy contains “hooks” (callbacks)
 - Depending on process label, callbacks get invoked
 - Kernel extension gets to inspect operation arguments
 - Return 0 to allow, non-zero to thwart operation
 - All registered hooks must allow operation.

MACF Policy Hooks



MACF Policy Modules

- Serves as basis for virtually all of Apple's OS Security
- Currently 5 known policy modules:

Kext	Oses	# Ops	Purpose
Quarantine	MacOS	~15-17	Gatekeeper. Sort of.
MCXALR	MacOS	1	Managed Client Extensions (MDM/Parental Controls)
TMSafetyNet	MacOS	~26	TimeMachine hooks on file access
AMFI	All (OSX >=10.10)	~8-13	Enforce code signing, some entitlements & Mach ports
Sandbox	All	130+	Confine, strangle and block Applications at every turn

- Labels can define which policy, if any, will take effect
 - Process can be execed into label with `mac_execve(#380)`
 - `posix_spawnattrs` can similarly enforce sandbox
 - Sandbox has own `spawnattrs` (for specific container or profile)

Sandboxing

- Original sandbox approach – “seatbelt” – opt in:
 - You’d have to ask to confined (like, want to go to jail!)
 - Like its namesake, most people find it borderline troublesome.
- Contemporary sandbox approach is radically different:
 - You are either containerized or you are not:
 - Voluntary: because you are a responsible developer
 - Semi-voluntary: Code signature or location (Apple controlled)
 - Non-voluntary: Based on install location (*OS)
 - If containerized, Sandbox intercepts all important operations
 - Definition of important keeps increasing to include more..
 - Operation assessed versus a profile, or entitlements

MacOS : App Sandboxing

- Sandbox no longer requires `sandbox_init` – but signature
 - This way Apple, not developer, can enforce sandboxing
 - In iOS, `/var/mobile/Containers/Bundle` location auto-sandboxes
- In MacOS, `com.apple.security.app-sandbox` sandboxes
- `com.apple.application-identifier` for container
 - Otherwise defaults to `CFBundleIdentifier` from App's Info.plist
- `com.apple.application-groups` (~10.7.5, 10.8.3 and later)
 - `~/Library/Group Containers/...`

1) Process loads libSystem.B

libSystem.B.initializer

libdispatch_init

2) libSystem Initializer calls libsecinit

libsecinit_initializer

libsecinit_initialize_once

_libsecinit_setup_secinitd_client

xpc_copy_entitlements_for_pid

xpc_pipe_routine

_libsecinit_setup_app_sandbox

__sandbox_ms

5) If decision is affirmative, libsecinit voluntarily imposes sandbox on process

3) Libsecinit registers with securityd

```
<key>SECINITD_REGISTRATION_MESSAGE_SHORT_NAME_KEY</key>
<string>TextEdit</string>
<key>SECINITD_REGISTRATION_MESSAGE_IS_SANDBOX_CANDIDATE_KEY</key>
<bool>true</bool>
<key>SECINITD_REGISTRATION_MESSAGE_ENTITLEMENTS_DICT_KEY</key>
<dict> ... Entitlement dictionary</dict>
```

```
<key>SECINITD_REPLY_MESSAGE_CONTAINER_ID_KEY</key>
<string>com.apple.TextEdit</string>
<key>SECINITD_REPLY_MESSAGE_QTN_PROC_FLAGS_KEY</key>
<integer>10</integer>
<key>SECINITD_REPLY_MESSAGE_CONTAINER_ROOT_PATH_KEY</key>
<string>/Users/morpheus/Library/Containers/com.apple.TextEdit/Data</string>
<key>SECINITD_REPLY_MESSAGE_SANDBOX_PROFILE_DATA_KEY</key>
<data>0x00003a014c004d0.. Compiled sandbox profile...</data>
<key>SECINITD_REPLY_MESSAGE_VERSION_NUMBER_KEY</key>
<integer>1</integer>
<key>SECINITD_MESSAGE_TYPE_KEY</key>
<integer>2</integer>
<key>SECINITD_REPLY_FAILURE_CODE</key>
<integer>0</integer>
```

4) Securityd decides whether or not process needs to be sandboxed

s
e
c
i
n
i
t
d

MacOS : App Sandboxing

- Containers created at ~/Library/Containers/{CFBundleIdentifier}
- All Structured the same way:
 - Container.plist: metadata (in bplist00 format)
 - Identity (Unicode, Base64)
 - Compiled profile (SandboxProfileData, base64)
 - SandboxProfileDataValidationInfo (long dict...)
 - Version (36 = MacOS 10, 38 = MacOS 11, 39 = MacOS 12)
 - Data: Directory structure, mimicking user's home directory:
 - .CFUserTextEncoding
 - Documents
 - Library
 - Music
 - Desktop
 - Downloads
 - Movies
 - Pictures

MacOS : App Sandboxing

- Data directories are often symbolic links(!)
 - SandboxProfileDataValidationRedirectablePathsKey limits links
- Metadata also holds entitlements, and other parameters
 - SandboxProfileDataValidationEntitlementsKey
 - SandboxProfileDataValidationParametersKey

iOS: Containers

Listing xx-conta: Traditional App directories vs. the Containers of iOS 8-9

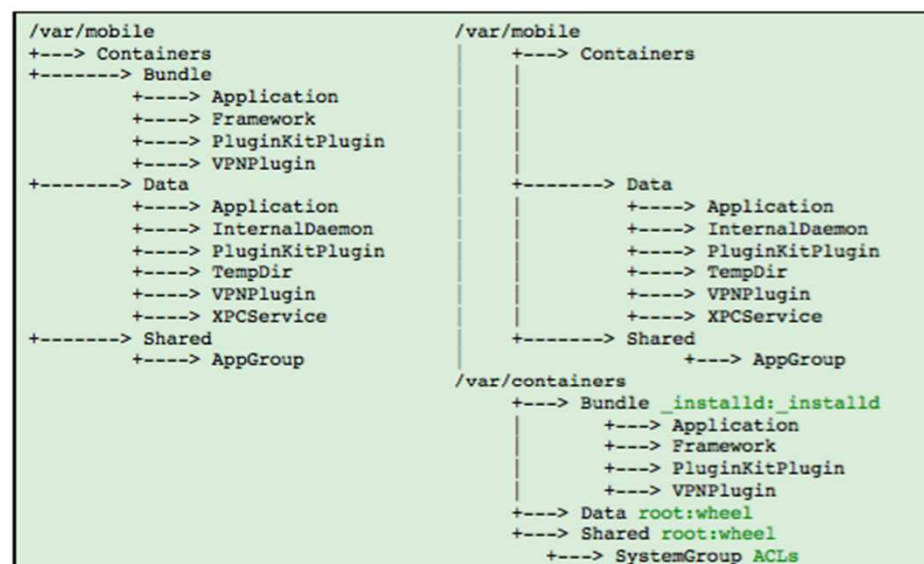
<pre> /var/mobile/Applications +--> UUID-OF-APP +---> appName.app/ +---> Documents/ +---> Library/ +---> Application Support/ +---> Caches/ +---> Cookies/ +---> Preferences/ +---> StoreKit/ +---> iTunesArtwork +---> iTunesMetadata.plist +---> tmp/ </pre>	<pre> /var/mobile/Containers/ +--> Bundle/ +--> Application/ +---> UUID-OF-APP +---> appName.app/ +--> Data/ +--> Application/ +---> UUID2-OF-APP/ +---> Documents/ +---> Library/ +---> StoreKit +---> iTunesArtwork +---> iTunesMetadata.plist +---> tmp; </pre>
--	---

- Also allows for shared containers
 - Apps with same team-id can share data

iOS 10 Containers

iOS 10 continues the evolution of containers, by once again moving Application static data to /var/containers, leaving /var/mobile/Containers with just Data/ and Shared/. The Application/ sub-directory structure has also been chown(2)ed to _installld. This is likely in anticipation of full multi-user capabilities.

Listing xx-conta: Containers in iOS 8-9 vs. those in 10



Another interesting change in iOS10 is the inclusion of a new SystemGroup/ shared container, which uses for the first time Access Control Lists (ACLs), as shown in Output xx-1:

Output xx-contacts: Access Control Lists on the Shared/SystemGroup containers

```

# As of iOS 10, shared system group containers also have ACLs
iPhone:/var/containers root# ls -le Shared/SystemGroup/
drwxr-xr-x+ 3 root wheel 136 Jul 7 12:40 6244C5EB-F346-43B5-A6A9-C269A6D02730
0: allow list,add_file,search,delete,add_subdirectory,delete_child,readattr,writeattr,
readextattr,writeextattr,readsecurity,writesecurity,chmod,file_inherit,directory_inherit,only_inherit
1: allow add_file,add_subdirectory,readextattr,writeextattr
...
drwxr-xr-x+ 3 root wheel 136 Jul 7 12:40 systemgroup.com.apple.pisico.suinfo
0: allow list,add_file,search,delete,add_subdirectory,delete_child,readattr,writeattr,
readextattr,writeextattr,readsecurity,writesecurity,chmod,file_inherit,directory_inherit,only_inherit
1: allow add_file,add_subdirectory,readextattr,writeextattr
  
```

iOS: Containers

- The sandboxd has been entirely removed in iOS as of 9.x
 - Still used in MacOS, primarily for tracing
- New daemon – containermanagerd – takes over
 - Part of mobilecontainer private framework
 - Communicates with user mode (installd, etc) over XPC port
 - Communicates with kernel mode (kext) over Special Port #25
 - MIG message 0x13392fd4 (322514900)
 - Contains sb_packbuff payload of kernel requests

#	CM_KERN_REQUEST_..
0	.._CODE_SIGNATURE_ID
1	.._CONTAINER_ID
2	.._APPLICATION_ID
3	.._UID
4	.._APP_GROUP_ID
5	.._CONTAINER_TYPE
6?	.._PERSONA_ID
7?	.._SYSTEM_GROUP_CONTAINER_ID

AMFI

- Sandbox and AMFI make good bedfellows
- AMFI ensures signature, provides entitlement services
- Sandbox depends on AMFI (as of 358 in MacOS)

```
morpheus@zephyr(~)$ kextstat
19 2 0xffffffff7f8100f000 0xd000 0xd000 com.apple.driver.AppleMobileFileIntegrity (1.0.5) <7 6 5 4 3 2 1>
22 2 0xffffffff7f8101c000 0x5000 0x5000 com.apple.kext.AppleMatch (1.0.0d1) <4 1>
23 1 0xffffffff7f81021000 0x17000 0x17000 com.apple.security.sandbox (300.0) <22 19 7 6 5 4 3 2
```

- iOS Sandbox uses specific entitlements:
 - seatbelt-profiles — assign a particular profile to binary
 - com.apple.private.security.container-required - Sandboxes built-in apps

Deconstructing Sandbox

- MacOS Sandbox.kext can serve as a good reference
 - Largely same codebase, with some differences, but symbolicated
- Joker can auto-symbolicate plenty*:
 - Stubs to kernel functions
 - Entire MACF Policy (120+ functions!)
- Can get other functions (no names, yet) with jtool:
 - `grep BL.*0x | cut -dx -f2` then feedback to companion file
 - About 150 additional functions revealed by this method
- Important functions (e.g. `smalloc`, `sfree`) yield rest.
 - `Hook_policy_syscall` especially important (for `mac_policy_syscall`)

* - Joker 3 can now handle split kexts from XNU 3750+!

Sandbox MACF Policy Hooks

- Most MACF Policy hooks call `cred_sb_evaluate`
 - 1st argument (in R0/X0/RDI) is MACF's
 - 2nd argument (in R1/X1/ESI) encodes operation number

```
morpheus@Zephyr (../10) %jtool -d _mpo_priv_check com.apple.security.sandbox.kext
Opened companion File: ./com.apple.security.sandbox.kext.ARM64.D6145CC4-1EDA-34AF-A613-A0E613FE791F
Disassembling from file offset 0x75304, Address 0xffffffff006b9a304 to next function
_mpo_priv_check:
ffffffff006b9a304      STP     X28, X27, [SP, #-48]!      ;
ffffffff006b9a308      STP     X20, X19, [SP, #16]      ;
ffffffff006b9a30c      STP     X29, X30, [SP, #32]      ;
ffffffff006b9a310      ADD     X29, SP, #32             ; $$ R29 = SP + 0x20
ffffffff006b9a314      SUB     SP, SP, 224             ; SP -= 0xe0 (stack frame)
ffffffff006b9a318      MOV     X19, X1                 ; --X19 = X1 = ARG1
ffffffff006b9a31c      MOV     X20, X0                 ; --X20 = X0 = ARG0
ffffffff006b9a320      ADD     X0, SP, #0              ; $$ R0 = SP + 0x0
ffffffff006b9a324      ORR     W2, WZR, #0xe0          ; -->R2 = 0xe0
ffffffff006b9a328      MOVZ    W1, 0x0                 ; -->R1 = 0x0
ffffffff006b9a32c      BL      _memset.stub           ; 0xffffffff006ba834c
; R0 = _memset.stub(SP + 0x0, 0x0, 224);
ffffffff006b9a330      ORR     W8, WZR, #0xf           ; -->R8 = 0xf
ffffffff006b9a334      STR     W8, [SP, #96]           ; *(SP + 0x60) =
ffffffff006b9a338      STR     W19, [SP, #104]         ; *(SP + 0x68) =
ffffffff006b9a33c      ORR     W1, WZR, #0x7c          ; -->R1 = 0x7c
ffffffff006b9a340      ADD     X2, SP, #0              ; $$ R2 = SP + 0x0
ffffffff006b9a344      MOV     X0, X20                 ; --X0 = X20 = ARG0
ffffffff006b9a348      BL      _cred_sb_evaluate       ; 0xffffffff006b96c70
; R0 = _cred_sb_evaluate(ARG0, 0x7c, SP + 0x0);
ffffffff006b9a34c      SUB     X31, X29, #32           ; SP = R29 - 0x20
ffffffff006b9a350      LDP     X29, X30, [SP, #32]      ;
ffffffff006b9a354      LDP     X20, X19, [SP, #16]      ;
ffffffff006b9a358      LDP     X28, X27, [SP], #48      ;
ffffffff006b9a35c      RET                                ;
_mpo_priv_grant:
```

Sandbox MACF Policy Hooks

- Operation numbers correspond to hard-coded names
 - Can also be found in older `libsandbox.1.dylib`
 - Removed (precompiled) into 570+
 - Names can be found in `kext's __DATA__CONST.__const`
 - Not going away since they are needed for APIs
 - There are more operations than there are MACF hooks
 - Some are callable from user mode by apps (e.g. `AppleEvents`, `TCC`)

Sandbox MACF Policy Hooks

- cred_sb_evaluate calls sb_evaluate
 - 1st parameter is sandbox obtained from label_get_sandbox
 - Operation as 2nd Parameter
 - Buffer as 3rd Parameter

```
morpheus@Zephyr (../10) %jtool -d _cred_sb_evaluate com.apple.security.sandbox.kext
Opened companion File: ./com.apple.security.sandbox.kext.ARM64.D6145CC4-1EDA-34AF-A613-A0E613FE791F
Disassembling from file offset 0x71c70, Address 0xffffffff006b96c70 to next function
_cred_sb_evaluate:
ffffffff006b96c70      STP    X22, X21, [SP, #-48]!      ;
ffffffff006b96c74      STP    X20, X19, [SP, #16]      ;
ffffffff006b96c78      STP    X29, X30, [SP, #32]      ;
ffffffff006b96c7c      ADD    X29, SP, #32             ; $$ R29 = SP + 0x20
ffffffff006b96c80      MOV    X20, X2                  ; --X20 = X2 = ARG2
ffffffff006b96c84      MOV    X21, X1                  ; --X21 = X1 = ARG1
ffffffff006b96c88      STR    X0, [X20, #16]           ;= X0 ARG0
ffffffff006b96c8c      LDR    X0, [X0, #120]           ; R0 = *(ARG0 + 120)
ffffffff006b96c90      BL     _label_get_sandbox        ; 0xffffffff006b96584
ffffffff006b96c94      MOV    X19, X0                  ; --X19 = X0 = 0x0
ffffffff006b96c98      MOV    X1, X21                  ; --X1 = X21 = ARG1
ffffffff006b96c9c      MOV    X2, X20                  ; --X2 = X20 = ARG2
ffffffff006b96ca0      BL     _sb_evaluate              ; 0xffffffff006b96ec0
ffffffff006b96ca4      MOV    X20, X0                  ; --X20 = X0 = 0x0
; // if (R19 == 0) then goto 0xffffffff006b96cb4
ffffffff006b96ca8      CBZ    X19, 0xffffffff006b96cb4 ;
ffffffff006b96cac      MOV    X0, X19                  ; --X0 = X19 = 0x0
ffffffff006b96cb0      BL     _sandbox_release          ; 0xffffffff006b96254
ffffffff006b96cb4      MOV    X0, X20                  ; --X0 = X20 = 0x0
ffffffff006b96cb8      LDP    X29, X30, [SP, #32]      ;
ffffffff006b96cbc      LDP    X20, X19, [SP, #16]      ;
ffffffff006b96cc0      LDP    X22, X21, [SP], #48      ;
ffffffff006b96cc4      RET                             ;
```

Sandbox MACF Policy Hooks

- `cred_sb_evaluate` derives credentials, and calls `eval` *
 - May or may not report sandbox violations (based on argument to check)

```
morpheus@Zephyr (../10) % jtool -d _sb_evaluate com.apple.security.sandbox.kext | grep BL
Opened companion File: ./com.apple.security.sandbox.kext.ARM64.D6145CC4-1EDA-34AF-A613-A0E613FE791F
Disassembling from file offset 0x79ec0, Address 0xffffffff006b9eec0 to next function
ffffffff006b9eec      BL      _derive_cred      ; 0xffffffff006b9f0fc
ffffffff006b9ef20      BL      _kauth_cred_proc_ref.stub ; 0xffffffff006ba81cc
ffffffff006b9ef3c      BL      _OSCompareAndSwapPtr.stub ; 0xffffffff006ba7f44
ffffffff006b9ef48      BL      _kauth_cred_unref.stub    ; 0xffffffff006ba81d8
ffffffff006b9ef70      BL      _eval              ; 0xffffffff006b9f164
ffffffff006b9efbc      BL      _eval              ; 0xffffffff006b9f164
ffffffff006b9f010      BL      _derive_vnode_path ; 0xffffffff006b9e590
ffffffff006b9f030      BL      _derive_socket_info    ; 0xffffffff006ba1030
ffffffff006b9f070      BL      _sb_trace            ; 0xffffffff006ba1fcc
ffffffff006b9f0bc      BL      _sb_report           ; 0xffffffff006ba161c
ffffffff006b9f0cc      BL      _free_filter_context   ; 0xffffffff006b9e410
morpheus@Zephyr (../10) %
```

- Evaluation first attempted against `platform_profile`
- Can default to specific process-defined (container) profile

* - MacOS implementation slightly different (includes `csr_check`, etc). iOS also inlines `eval_filter` into `eval`

Reversing Profiles

- Sandbox Profiles are written in tinyScheme (UGH!)
 - In MacOS – plaintext, in /System/Library/Sandbox/Profiles
 - Per framework profiles also exist for Apple's frameworks
 - in iOS – compiled & built-in!
- The gist:
 - (version 1) (only version supported)
 - (deny default) (least privilege)
 - (allow) (selectively allow APIs)
 - (deny) (selectively disallow APIs)
- Can apply and trace using `sandbox-exec`:

```
(version 1)
(trace "/tmp/appTrace.sb")
```

Sandbox-exec

- Simple binary (300-500 lines of ASM)

```
morpheus@zephyr (~)$ sandbox-exec
Usage: sandbox-exec [options] command [args]
Options:
  -f profile-file      Read profile from file.
  -n profile-name      Use pre-defined profile.
  -p profile-string    Specify profile on the command line.
  -D key=value         Define a profile parameter.
Exactly one of -f, -n, -p must be specified.
```

- MacOS 11 adds undocumented “-t” for tracing
 - Tracing broken in iOS with the removal of sandboxd ☹
- Closed source – but....
 - Fully compatible clone at <http://NewOSXBook.com/tools/sob.html>
 - Will also dump compiled profile in /tmp
 - Provides first implementation of sandbox-exec for iOS!

Built-in Profiles

- MacOS originally had 4 “built-in” profiles
 - Weren’t so useful in the first place and largely deprecated
- iOS extends that to dozens of profiles
 - Can be found in kext
 - Can also be found in iOS’s libsandbox.1.dylib
 - AGXCompilerService ... wifiFirmwareLoader
- Built-in profiles are precompiled
 - Originally, maintained by sandboxd
 - In iOS 9+, maintained inside kext (___TEXT.__const)

```
morpheus@Zephyr [~/10] %jtool -v -dD ___TEXT.__const com.apple.security.sandbox.kext | grep -A 4 -a A.G.X.C
Opened companion File: ./com.apple.security.sandbox.kext.ARM64.D6145CC4-1EDA-34AF-A613-A0E613FE791F
Dumping from address 0xffffffff00630e620 (Segment: ___TEXT.__const) to end of section
Address : 0xffffffff00630e620 = Offset 0x520
0xffffffff006353360: 13 00 00 00 41 47 58 43      A G X C
0xffffffff006353368: 6f 6d 70 69 6c 65 72 53    o m p i l e r S
0xffffffff006353370: 65 72 76 69 63 65 00 00    e r v i c e
0xffffffff006353378: 0e 00 00 00 48 2f 64 65      H / d e
0xffffffff006353380: 76 2f 70 74 6d 78 0f 00    v / p t m x
```

Containerizing Applications (iOS)

mpo_cred_label_update_execve hook

MACF calls sandbox, because it registered hook

amfi_copy_seatbelt_profile_names

Get <seatbelt-profiles> entitlement, if any

PE_I_can_haz_debugger

Allows debug_mode (if boot arg was set, non-issue)

get_signing_identifier

Gets application-identifier (from entitlement)

get_container_required_entitlement

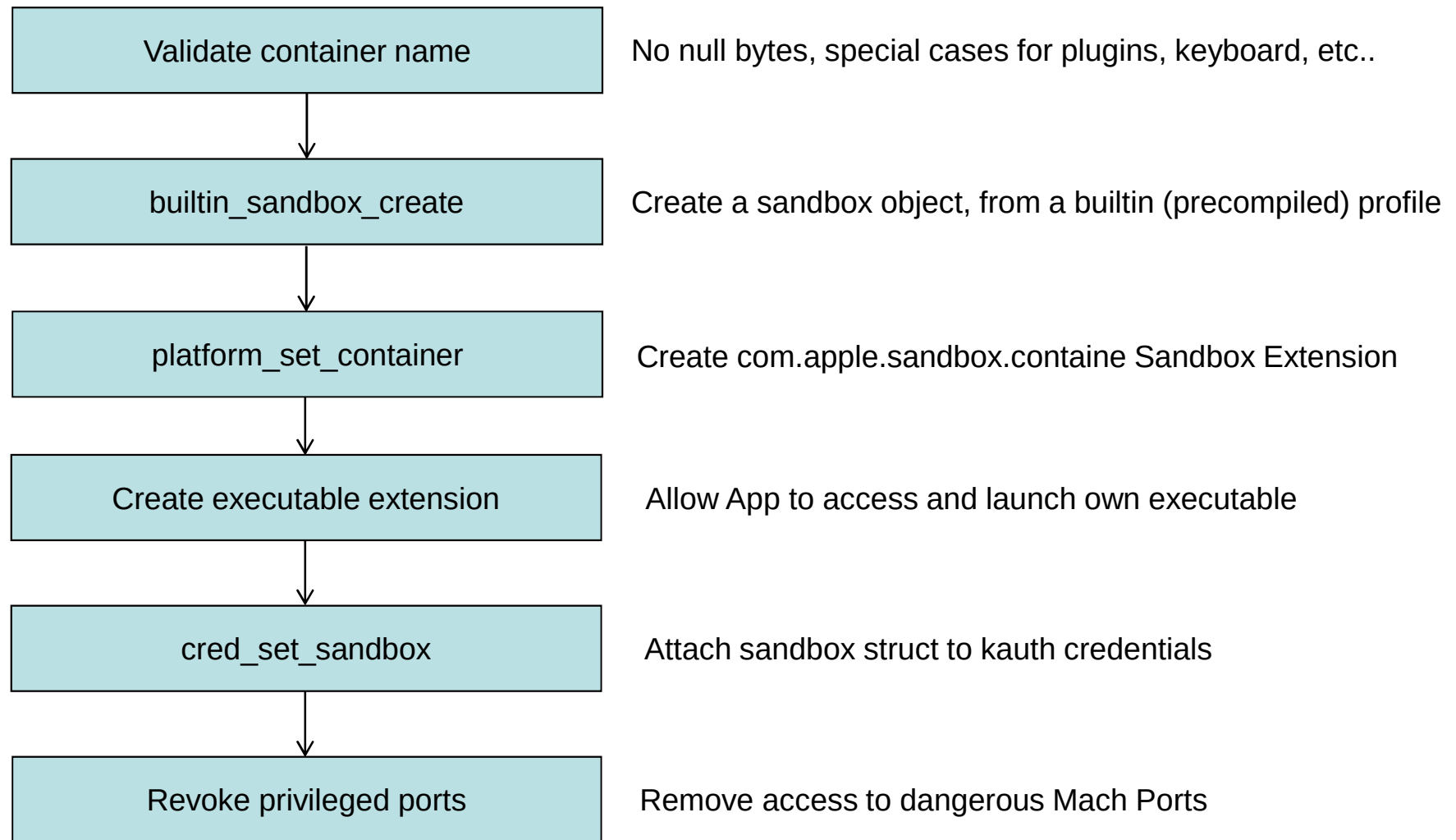
Get com.apple.private.security.container-required ent

check sandbox spawnattrs

Several validations on spawn attributes

Upcall to container manager

Containerizing Applications (iOS)



Sandbox APIs

- Sandbox usermode APIs provided by two libraries:
 - /usr/lib/system/libsystem_sandbox.dylib
 - Re-exported by LibSystem.B.dylib
 - Mostly direct APIs to kext
 - /usr/lib/libsandbox.1.dylib
 - Profile compilation
 - TinyScheme implementation statically linked in
 - Plenty of Scheme strings/profile definitions in __TEXT.__const
- Containment (often) performed over mac_execve()
- KEXT APIs invoked over macf_syscall()

Sandbox APIs

- `mac_syscall` (#381) used extensively:
 - Allows `ioctl(2)` style multiplexing of syscalls provided by a kext
 - Generic mechanism, used by all policy modules
 - On kext end, `hook_policy_syscall` enables multiplexing
 - Different offerings in MacOS and *OS

Sandbox APIs

- Syscall implementations differ in between OSes, versions!

Op	Sandbox function	Purpose
0-1	<code>_set_profile[_builtin]</code>	Set a profile (=label & containment) of a process
2	<code>_check</code>	Check if operation is allowed in confines of sandbox
3	<code>_note</code>	Attaches a note (memory buffer) to sandbox (offset 0x80)
4	<code>_container_path_for_pid</code>	Retrieve container path for a given PID
5-7	<code>_extension_issue/consume/release</code>	<i>Issue, apply and remove a temporary exception</i>
8-9	<code>_extension_update_file[_with_new_type]</code>	<i>Update/twiddle extension</i>
10-11	<code>_suspend/unsuspend</code>	Suspend/resume sandbox checks for PID*
13-15	<code>_policy_syscall</code> related...	iOS, routed to container manager
16	<code>_inspect</code>	Dump tons of great information on SB.
17	<code>profile_dump</code>	Dumps compiled profile for a PID (MacOS, AppleInternal** ☹)
19	<code>_vtrace[enable disable report]</code>	Trace operation to a buffer. Not on iOS ☹
21	<code>_rootless_allows_task_for_pid</code>	Does current policy allow <code>task_for_pid</code> call?

- Get a more accurate list with jtool's switch detection (ARM64)

* - Don't get excited. Process can only do it on itself, if entitled as a sandbox-manager *and* another exception entitlement..

** - `csr_check(0x01)` – can be tweaked via direct access to NVRAM

Sandbox APIs

- `sandbox_check` especially useful:
 - Widely used in tweaks to gauge sandbox restrictions
 - Commonly used with `SANDBOX_CHECK_NO_REPORT`
 - Performs check silently, without any user-mode output

Listing xx-sbchk: Demonstrating the `sandbox_check` function

```
int port_denied = sandbox_check (pid,  
                                "mach-lookup",  
                                SANDBOX_FILTER_RIGHT_NAME | SANDBOX_CHECK_NO_REPORT,  
                                "com.apple.....");  
  
int read_denied = sandbox_check (pid,  
                                "file-read-data",  
                                SANDBOX_FILTER_PATH | SANDBOX_CHECK_NO_REPORT,  
                                "path/to/file");
```

- Really useful for probing container XPC/file restrictions
 - Much more reliable than decompiling!
- Sandbox 570+ adds `sandbox_check_bulk`

Demo: sbtool

```
root@Padishah (/var/root)# ps -ef | grep MobileSafari$
  501  3427      1   0 Mon06AM ??          0:00.44 /Applications/MobileSafari.app/MobileSafari
root@Padishah (/var/root)# sbtool 3427 mach | grep No | head -5
Checking Mach services for 3427....
com.apple.Preferences.gsEvents: Nope
com.apple.mobilemail.gsEvents: Nope
com.apple.timezoneupdates.tzd.server: Nope
com.apple.streaming_zip_conduit: Nope
com.apple.pfd: Nope
root@Padishah (/var/root)# sbtool 3427 mach | grep Yep | head -5
Checking Mach services for 3427....
com.apple.voiceservices.tts: Yep
com.apple.nehelper: Yep
com.apple.coremedia.videocompositor: Yep
com.apple.coremedia.mutablecomposition: Yep
com.apple.managedconfiguration.mdmdpush-prod: Yep
```


Sandbox APIs - undocumented

- `sandbox_inspect_pid` super useful, but undocumented:
 - Available in *OS as of somewhere in 460 (iOS 9.something)

```
int sandbox_inspect_pid(int pid,    /* in */  
                        char **buf, /* out */  
                        int *size); /* out */
```

- Implemented via `__sandbox_ms (... , 0x10)`;
 - Very valuable information on process, directly from `kext`
- Requires root privileges (or AppleInternal build)

Demo: sbtool

```
root@Padishah (/var/root)# sbtool 3427 inspect
MobileSafari[3427] sandboxed.
size = 439166
container = /private/var/mobile/Containers/Data/Application/607D2C61-76F7-49AF-B3FB-B6B4BE45AA47
sb_refcount = 210
profile = container
profile_refcount = 56
extensions (3: class: com.apple.sandbox.executable) {
    file: /Applications/MobileSafari.app (unresolved); flags=0
}
extensions (5: class: com.apple.sandbox.system-container) {
    file: /private/var/containers/Data/System/738391BB-914B-4AEF-88CE-D8758754CCBD (unresolved); flags=0
}
extensions (5: class: com.apple.security.exception.mach-lookup.global-name) {
    mach: com.apple.mobile.keybagd.xpc; flags=0
    mach: com.apple.parsec.subscription.service.internal; flags=0
    mach: com.apple.SafariCloudHistoryPushAgent; flags=0
    mach: com.apple.Safari.SafeBrowsing.Service; flags=0
}
extensions (7: class: com.apple.security.exception.files.absolute-path.read-only) {
    file: /private/var/mobile/Library/Caches/com.apple.storeservices (unresolved); flags=0
}
extensions (8: class: com.apple.sandbox.container) {
    file: /private/var/mobile/Containers/Data/Application/607D2C61-76F7-49AF-B3FB-B6B4BE45AA47 (unresolved); flags=0
}
```

Sandbox Extensions

Extensions allow exceptions to a given profile

iOS apps get the “standard extensions”:

- com.apple.sandbox.executable
- com.apple.sandbox.container
- com.apple.sandbox.application-group

Sandbox Extensions

Apple's App provide even more extensions for themselves:

Extension
com.apple.security.exception.shared-preference.read-write
com.apple.sandbox.application-group
com.apple.tcc.kTCCServiceAddressBook
com.apple.sandbox.executable
com.apple.app-sandbox.read
com.apple.security.exception.mach-lookup.global-name
com.apple.security.exception.iokit-user-client-class
com.apple.security.exception.files
com.apple.sandbox.container

Sandbox Extensions

- Before sandboxing, caller can set extensions (unless forbidden)

Table xx-isexts: Extension issuance APIs in libsystem_sandbox.dylib

#	sandbox_extension_issue_...	Provides
0	...file[_with_new_type]	Access to named files/directories
1	...mach	Access to named Mach/XPC ports
2	...iokit_user_client_class	Access named IOUserClient (IOServiceOpen())
	...iokit_registry_entry_class	Permission to iterate the IORegistry for specific class (570)
3	...generic	Extensions which don't fall into other categories
4	...posix_ipc	Access to named POSIX IPC object (UN*X sockets, etc)

- Extensions are issued by sandbox kext as “tokens”
 - Hmac_sha1 with secret value (not exposed to user space)

Take Aways

- If you're even loosely interested in OSX/iOS:
 - The sandbox is the first, possibly last line of security
 - In iOS, provides the most important obstacle to jailbreaking
 - In MacOS, containerizes AppStore Apps, and implements SIP
- <http://NewOSXBook.com/articles/hitsb.html>
 - Source of sandbox_exec clone
 - Sbttool – open source
 - Ongoing documentation on profile reversing
 - Fully symbolicated companion file for iOS 10 kext

Suggested Links

- <http://NewOSXBook.com/> - MOXiI, 2nd Edition
 - Volume III (Security & Insecurity) available for pre-order!
- <http://NewOSXBook.com/forum> - Open forum for MOXiI
- <http://Technogeeks.com/OSXRE> - Related Training