## How to Inject Code into Mach-O Apps. Part II.

jon-gabilondo-angulo-7635.medium.com/how-to-inject-code-into-mach-o-apps-part-ii-ddb13ebc8191

Jon Gabilondo

July 25, 2022

## **Dynamic Code Injection Techniques**



Organismo-App-inspector within Microsoft Word 2018.

## Update !

This article was written while finding a solution to inject Organismo framework, or any other library, into hardened applications. The article described a journey to overcome the limitations that prevent code injection, until we stumbled into the final hurdle of <u>AMFI</u> and the Gatekeeper.

Gladly, a little research on AMFI brought a surprising solution. You may jump to the end of the article to find out the solution to inject code freely on any Mac App. However the content until there is really interesting if you are into the details of the OS X (& iOS) system security.

In Part I we saw how easy it is to inject code into Mac Apps, from Calculator to Mail, even more surprisingly, into Microsoft Apps like Word 2018. Why would such important applications not have a simple protection (hardening) against external code injection is not easy to understand. In fairness we must say that in Part I we worked with the condition of disabling System Integrity Protection (SIP) which is a major security layer on OS X.

In Part I we used a dynamic code injection techique using the DYLD\_INSERT\_LIBRARIES environment variable, an old property of the Dynamic Linker 'dyld' to load external libraries.

However, as one would expect, the simple dynamic code injection used in Part I would not succeed in (important) Apps such as iTunes, Xcode, Photos... These Apps are hardened to instruct the 'dyld' to disable the injection defined by environment variables as well as to reject code that does not match code signature of the Application.

In this story we will walk towards finding a solution to inject external code into hardened Apps like iTunes and Xcode.

### **Hardened Runtime**

The Hardened Runtime, along with System Integrity Protection (SIP), protects the runtime integrity of your software by preventing certain classes of exploits, like code injection, dynamically linked library (DLL) hijacking, and process memory space tampering.

There are two ways to harden your Application, the official one is by using entitlements and a less common one by creating a \_\_\_\_RESTRICT segment in the Mach-O binary.

To harden your Application in Apple's official way navigate in Xcode to your target's Signing & Capabilities information and click the + button. In the window that appears, choose Hardened Runtime.



Then add the capability 'Hardened Runtime". Check in 'Build Settings' that "Hardened Runtime" is enabled.



Find Apple's documenttion here:

#### Hardened Runtime Entitlements

<u>The Hardened Runtime, along with System Integrity Protection (SIP),</u> protects the runtime integrity of your software by...

developer.apple.com

#### All About Notarization - WWDC 2019 - Videos - Apple Developer

Notarization is all about identifying and blocking malicious Mac software prior to distribution, without requiring App...

developer.apple.com

#### Hardening by Entitlements

We've seen that Xcode allows to harden one Application in a simple manner, but how are they applied to the binary and where are they kept in a binary.

Let's start by retrieving the the entitlements of any Application binary, this is done using *codesign*, lets' do it on Music.app:

```
$ codesign -d --entitlements :-
/System/Applications/Music.appExecutable=/System/Applications/Music.app/Contents/MacOS
```

This is the output of the *codesign* command containing the xml list with the entitlements of the Music.app:

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST</pre> 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd"><plist version="1.0"><dict> <key>com.apple.PairingManager.Read</key><true/> <kev>com.apple.PairingManager.RemovePeer</kev><true/> <key>com.apple.PairingManager.Write</key><true/> <key>com.apple.amp.artwork.client</key><true/><key>com.apple.amp.devices.client</key> <true/><key>com.apple.amp.library.client</key><true/><key>com.apple.applicationidentifier</key><string>com.apple.Music</string> <key>com.apple.authkit.client.internal</key><true/><key>com.apple.avfoundation.allowsystem-wide-context</key><true/><key>com.apple.avfoundation.allows-access-to-devicelist</key><true/><key>com.apple.avfoundation.allows-set-output-device</key><true/> <key>com.apple.cdp.recoverykey</key><true/><key>com.apple.mediaremote.allow</key> <array><string>TVPairing</string></array> <key>com.apple.private.accounts.allaccounts</key><true/> <key>com.apple.private.applemediaservices</key><true/><key>com.apple.private.apsconnection-initiate</key><true/><key>com.apple.private.audio.notification-wakeaudio</key><true/><key>com.apple.private.bmk.allow</key><true/> <key>com.apple.private.commerce</key><array> <string>Accounts</string></array> <key>com.apple.private.fpsd.client</key><true/> <key>com.apple.private.notificationcenter-system</key><array> <dict> <key>identifier</key> <string>com.apple.appstoreagent</string> </dict></array> <key>com.apple.private.rtcreportingd</key><true/> <key>com.apple.private.security.storage.mobilesync.heritable</key><true/> <key>com.apple.private.sqlite.sqlite-encryption</key><true/> <key>com.apple.private.tcc.allow</key><array> <string>kTCCServiceAddressBook</string> <string>kTCCServicePhotos</string> <string>kTCCServiceAppleEvents</string> <string>kTCCServiceSystemPolicyAllFiles</string> <string>kTCCServiceCamera</string> </array><key>keychain-access-groups</key><array> <string>com.apple.pairing</string> <string>com.apple.airplay</string> <string>apple</string></array></dict></plist>

At runtime the OSX/iOS needs to validate the operations and resources accessed by the executable, for that it needs to know what has the App been validated for, via the entitlements. The entitlements are kept in the binary Mach-O structure itself, in the 'Code Signature' setion.

		Music			
RVA			Q, key		¢
Sectioned ( DATA _ ship const)	pFile	Data LO	Data HI	Value	
Section64 (DATAobjc_const)	ATRODEAA	FA DE AC CO 00 03 RE AD	00 00 00 01 00 00 00 00		
P Section64 (_DATAobjc_sellers)	01890F10	00 00 00 00 00 00 00 00 00 00	00 03 74 EC 00 00 00 00	+	
Section64 (_DATAobjc_protorets)	01000520	00 00 00 20 00 00 00 02	00 03 74 FC 00 00 00 03	1	
P Section64 (DATA,objc_classifiers)	01000120	00 03 73 50 00 01 00 00	00 03 70 03 FA 02 00 02		
<pre>&gt; Section64 (DATA,objc_superrets)</pre>	01090730		00 01 20 00 00 00 01 10 00 00 10 00 01 00 00 00		
Section64 (_DATA,_objc_ivar)	01090740	29 92 96 96 99 99 99 99	00 00 10 92 01 09 07 00		
Section64 (_DATA,_objc_data)	01890750	20 02 09 00 00 00 00 00	00 00 00 00 00 00 00 00		
Section64 (_DATA,_data)	01890760	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		
Section64 (DATA,thread_vars)	0189DF70	00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		
Section64 (DATA,thread_data)	01890F80	00 00 00 01 00 0A 0F 00	00 00 00 00 63 6F 6D 2E		
Section64 (RESTRICT,restrict)	0189DF90	61 70 70 6C 65 2E 4D 75	73 69 63 00 74 41 05 F6	apple.Music.tA.	
* Dynamic Loader Info	01B9DFA0	6A 8A 25 42 68 2C 08 8C	71 5A ED 3A 5B DD 2F CA	j.%Bh,qZ.:[./.	
▶ Rebase Info	01B9DFB0	2F 43 4F 04 E6 23 37 A6	86 07 82 63 00 00 00 00	/C0#7c	
▶ Binding Info	01B9DFC0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		
▶ Weak Binding Info	01B9DFD0	00 00 00 00 00 00 00 00	00 00 00 00 20 32 02 EF	2	
► Lazy Binding Info	01B9DFE0	02 7A 91 B3 35 60 21 54	A4 17 20 D6 F8 BF 4E ED	.z5`!TN.	
* Function Starts	01B9DFF0	F0 85 E0 C3 A6 36 F5 DB	64 07 59 61 6D 51 74 93	6d.YamQt.	
Functions	0189E000	D4 79 88 A8 F6 C3 66 AB	33 FA 11 4A C0 D1 CF 46	.yf.3JF	
▷ Symbol Table	0189E010	EC E2 06 BA F6 5E B5 5C	4B CD DF 84 8D AB 23 1A	*.	
Dynamic Symbol Table	0189E020	76 F8 70 20 C7 50 23 28	1A 4C 6A 7A 4A 5F 3C 5B	v.p .P#(.LjzJ_<[	
String Table	01B9E030	05 38 F7 16 86 10 A5 0E	9A 80 8B 5A F6 29 74 D8	.8Z.)t.	
Code Signature	0189E040	77 28 A9 B7 10 7B 28 41	B4 04 E5 DE 50 6D F2 6B	w({(APm.k	

The Code Signature section in Music.app binary.

Entitlements is therefore a simple xml list of keys defining the entitlement and its properties. Although Apple documents and offers a handful of them to programmers, internally there might be hundreds.

The assignment of entitlements to an App is a matter of defining the entitlements xml file in the *codesign* command:

\$ codesign --entitlements entitlements.xml -f -s "iPhone Distribution: Company (XYZ)" Payload/Example.appAn interesting option when resigning an App that has already the entitlements set use the flag:--preserve-metadata=entitlements

## Hardened by \_\_\_RESTRICT

A **\_\_\_\_RESTRICT** segment is a section in the Mach-O binary file that can be created at link time. The section has no content. It acts like a flag to instruct the 'dyld' to perform code signature validation to all code loaded to the process.

See bellow how does the \_\_\_\_RESTRICT section look like in iTunes:





Creating the \_\_\_\_RESTRICT segment in Xcode is actually very easy. It is done by adding the following flags into your "Other Linker Flags":

-Wl,-sectcreate,\_\_RESTRICT,\_\_restrict,/dev/null

### Why Is the Injection Rejected

The \_\_\_RESTRICT segment flags the '*dyld*' to activate the code signature validation for all library to be loaded.

We saw in Part I how the injection in iTunes failed with the following message:

"dyld warning: could not load inserted library into hardened process because no suitable image found. Code signature in framework not valid for use in process using Library Validation"

<pre>[jongabilondo@jon=macbook=pro:~\$ DYLD_INSERT_LIBRARIES=/Users/jongabilondo/Library/Developer/Xcode/DerivedData/Organismo-ablrcuiefmvkmwflhidfive] [zgoic/Build/Products/Debug/Organismo=mac.framework/Versions/A/Organismo=mac /Applications/iTunes.app/Contents/HacOS/iTunes [dyld: warning: could not load inserted library '/Users/jongabilondo/Library/Developer/Xcode/DerivedData/Organismo=ablrcuiefmvkmwflhidfivezgoic/ Build/Products/Debug/Organismo=mac.framework/Versions/A/Organismo=mac' into bardened process because no suitable image found. Did find:</pre>
/Users/jongabilondo/Library/Developer/Xcode/DerivedData/Organismo-ablrcuiefmvkmwflhidfivezgoic/Build/Products/Debug/Organismo-mac.frame
work/Versions/A/Organismo-mac: code signature in (/Users/iongabilond/Library/Developer/Xcode/DerivedData/Organismo-maltruiefmytkmwflhidfivezgoi -/Ruild/Products/Debug/Organismo-mac, framework/Versions/A/Organismo-mac) not valid for use in process
is a platform binary, but mapped file is not
/Users/jongabilondo/Library/Developer/Xcode/DerivedData/Organismo-ablrcuiefmvkmwflhidfivezgoic/Build/Products/Debug/Organismo-mac.frame work/Unrapismo-mac.stati/ failed with erron=1
/Users/jongabilondo/Library/Developer/Xcode/DerivedData/Organismo-ablrcuiefmvkmwflhidfivezgoic/Build/Products/Debug/Organismo-mac.frame
work/Versions/A/Organismo-mac: code signature in (/Users/jongabilondo/Library/Developer/Xcode/DerivedData/Organismo-ablrcuiefnvkmwflhidfivezgoi
is a platform binary, but mapped file is not

If we analyse the code signature of iTunes and Organismo we can see they obviously have a different code signing values (certificates):

Executable=/Applications/iTunes.app/Contents/MacOS/iTunes Identifier=com.apple.iTunes Formatmap bundle with Mach-0 thin (x86\_64) CodeDirectory v=20100 size=266721 flags=0x2000(library-validation) hashes=8328+5 location=embedded Platform identifier=7 Hash type=sha256 size=32 CandidateCDHash sha256=bdd6819ddb2907046eb6dc29d150d5178459d391 Hash choices=sha256 CDHash=bdd6819ddb2907046eb6dc29d150d5178459d391 Signature size=4485 Authority=Software Signing Authority=Apple Code Signing Certification Authority Authority=Apple Root CA Info.plist entries=42 TeamIdentifier=not set Sealed Resources version=2 rules=13 files=4142 Internal requirements count=1 size=64

iTunes code signature values.

```
Identifier=com.organismo-mobile.Organismo
Format=bundle with Mach-0 thin (x86_64)
CodeDirectory v=20200 size=1054 flags=0x0(none) hashes=27+3 location=embedded
Hash type=sha256 size=32
CandidateCDHash sha256=76ced99ac03a03a0e6f1188c6b2c48a943bb7348
Hash choices=sha256
CDHash=76ced99ac03a03a0e6f1188c6b2c48a943bb7348
Signature size=4738
                                                    MR)
Authority=Apple Development: Jon Gabilondo (HSI
Authority=Apple Worldwide Developer Relations Certification Authority
Authority=Apple Root CA
Signed Time=17 Sep 2019 at 16:22:49
Info.plist entries=19
TeamIdentifier=NR
                       ZC
Sealed Resources version=2 rules=13 files=4
Internal requirements count=1 size=196
```

Organismo code signature values.

#### **Approaches to Overcome Hardening**

In order to overcome the hardening barrier, these are the ideas that come to my mind. From foolish to realistic, they are:

- 1. Build our own tweaked 'dyld' and instruct the Application to use it, instead of the default '/usr/lib/dyld'.
- 2. Codesign all software components with your own Certificate.
- 3. Remove the \_\_\_\_RESTRICT segment.

## **Option 1. Create a Custom dyld**

#### macOS 10.14.5 - Source

#### <u>Please be advised that unless your final product is also open source,</u> <u>incorporating open source software containing...</u>

#### opensource.apple.com

second, as we saw in part I, the Mach-O binary of every App specifies what dynamic linker must be used. In Mac OS X is always '/usr/lib/dyld':

Executable (X86_64)	Offset	Data	Description	Value
Mach64 Header	00000B38	0000000E	Command	LC LOAD DYLINKER
▼ Load Commands	00000B3C	00000020	Command Size	32
LC_SEGMENT_64 (PAGEZERO)	00000B40	0000000C	Str Offset	12
► LC_SEGMENT_64 (TEXT) ► LC_SEGMENT_64 (DATA) LC_SEGMENT_64 (LINKEDIT) LC_DYLD_INFO_ONLY LC_SYMTAB LC_DYSYMTAB LC_LOAD_DYLINKER	•	2F7573722F6C69622F64796	Name	/usr/lib/dyld

As of today the <u>macOS 10.14.5 open source</u> site has more than one hundred projects. The Dynamic Linker of OS X and iOS is right there. Impressive.

	😫 < 🗦 🛅 dyld		< 🛆 >
r 📴 dyld	Ger	eral Signing & Capabilities Resource Tags Bui	Id Settings Build Phases Build Rules
testing     configs	PROJECT	Basic Customized Al Combined Levels	+ Qr Search
doc     include	TARGETS	* Architectures	dvid
Products     dyld3	update_dyld_share	Additional SDKs Architectures	Standard Architectures (64-bit Intel) + \$(ARCHS_STANDA
Isunch-cache     Frameworks	dyld	Base SDK W Build Active Architecture Only	macosx.internal (SDK not found) C C
	Ibdyld.dylb	Release	Yes 0 No 0
	multi_dyld_shared	Supported Platforms Valid Architectures	watchos macosx iphonesimulatornano iphonesimulator i386 x86_64
	update_dyld_sim_s_	▼ Assets	

Well, first problem:

Base SDK MacOSX.internal (SDK not found)

MacOSX.internal... Okay... I understand, Apple has its own SDK, makes sense. Can I get it ? Obviously not. So lets set the Base SDK to what we've got by default: 'macOS'.

What happens from then on is a never ending compilation attempts with missing includes: \_simple.h, Block\_private.h, coreSymbolicationDyldSupport.h, CrashReporterClient.h, cs\_blobs, objc-shared-cache, etc. etc.

One can painstakingly find the missing includes from different packages of the open source, until it makes no sense to continue. There must be a better way to do it, but it's nowhere to be found.

This option ends here. One wonders how does Apple understand open source.

#### **Option 2. Re-Codesigning with Your Certificate**

We assume that the Organismo framework is rejected by the 'dyld' because its code signing certificate is different from the one of iTunes. Organismo uses your Developer Certificate, iTunes uses Apple's Private Certificate.

Therefore we assume that code signing iTunes with our Dev Certificate, the 'dyld' will pass the code integrity validation when loading Organismo.

Re-codesigning any App is as simple as (remember to work on a copy of the App !):

```
$ codesign --deep --force --preserve-metadata=entitlements -s 'Mac Developer: Jon
Gabilondo (HSU.....MR)' /Users/jongabilondo/Desktop/iTunes.app
```

Checking the signature with 'codesign -dvvv' we can actually see that the signature is now 'Jon Gabilondo', which is what we wanted. Let's run the modified iTunes to see if it runs properly, before we attempt the injection:

\$ DYLD\_INSERT\_LIBRARIES=/path\_to/Organismo-mac.framework/Versions/A/Organismo-mac /Users/jongabilondo/Desktop/iTunes.app/Contents/MacOS/iTunes

	iTunes cannot be opened because of a problem.
<u>_</u>	Check with the developer to make sure iTunes works with this version of macOS. You may need to reinstall the application. Be sure to install any available updates for the application and macOS.
	Click Report to see more detailed information and send a report to Apple.
?	Ignore Report

The assumption of having the same code signature was right, partially. It did not consider the ramifications of dylib dependencies that will always end up in an Apple system dylib with Apple's Signature. Like we see in the error report of the modified iTunes, the dylibs of iTunes have Jon's signature, but its dependencies always end up in Apple's OSX /usr/lib dylibs.

Option 2 not good either.



Error loading dylib because binary is restricted.

It is always very useful to read Apple's :

## macOS Code Signing In Depth

#### <u>Code signing is a facility by which developers can assign a digital identity to</u> <u>their programs. Apple provides the...</u>

developer.apple.com

#### **Option 3. Removing the \_\_\_RESTRICT segment**

I'm left with the last bullet. If the \_\_\_RESTRICT segment is the flag that is activating the code signature validation, we might just remove it from the Mach-O binary.

This approach requires static code modification, which obviuously causes the binary (checksums) invalidation. But we know how to recodesign an App with our Dev Certificate so there should be no problem.

### Removing the \_\_\_RESTRICT segment

Several tools could be used to modify a Mach-O binary file, you can choose your favourite. Here I will use iHex editor. Find the \_\_\_RESTRICT segment and rename it , for instance to: \_\_XESTRICT. Save the file and codesign it with your Dev Certficate.

Hex	Text	Find		RES										
	Re	place												
						0								
						L	Replace All	Replace	R	eplac	e & Find	a   P	revious	Ne
2327	00000000	31 31 01	UL	04033103	0001/3/3	12030013	JI JI 4441	34410000			ojc_c		(CI)	UNIN
2352	00000000	000000	00	70715501	01000000	80090000	00000000	7071FF01			pq		A	pq
2380	03000000	646355	22	75706572	72656673	000000000	54410000	00000000		hic		fr	DATA	
2408	00000000	CA035F	13	01000000	120300/3	00000000	54410000	00000000	0	bjc_	super	rers.	DATA	
2450	00000000	P07AFF	00	00000000	10040000	00000000	P07AFF01	03000000		•2			•2	
2404	64635560	766172	00	00000010	55554441	54410000	00000000	00000000	ic	iver		DA.	ТА	
2520	04055F09	010000	00	E8120000	00000000	087EEE01	00000000	00000000	16-	Lvur		UA		
2520	000000000	010000	00	000000000	00000000	007777701	SESECCE 2	CAC25ECA						obic
2576	61746100	000000	00	SESEAAA1	54410000	00000000	00000000	00025504	a+ a		DA	ТА		50 J C
2604	01/40100	E03300	00	00000000	0002EE01	030000000	00000000	00927701	acu	+3	0A	<u>،</u>		
2632	00000000	000000	00	000000000	000021101	SESE6461	74610000	00000000		+5			data	
2660	000000000	5E5E44	41	54410000	00000000	00000000	E0C5FF01	010000000		D	ATA		uuuu t~	•
2688	70580100	000000	000	FACSEFAL	04000000	00000000	000000000	00000000	n٢		+~.		+~	
2716	000000000	000000	00	000000000	5E5E7468	72656164	5E766172	73000000	۴L		• *	th	read v	ars
2744	5F5F4441	544100	00	000000000	00000000	50210102	01000000	18000000	D	ΔΤΔ			PI	
2772	00000000	502101	02	03000000	000000000	00000000	13000000	00000000		PI				
2800	000000000	000000	000	5E5E7468	72656164	5E646174	61000000	5E5E4441		•••	th	read	data	
2828	54410000	000000	00	00000000	68210102	01000000	10000000	00000000	TA			h!		
2856	68210102	030000	000	00000000	00000000	11000000	00000000	00000000	hl					
2884	00000000	5F5F63	6F	6D6D6F6E	00000000	00000000	5F5F4441	54410000		c	ommon			DATA
2912	00000000	000000	000	80210102	01000000	38000200	00000000	00000000			Ä!	1	8	
2940	04000000	000000	000	00000000	01000000	00000000	00000000	00000000					-	
2968	5F5F6273	730000	000	00000000	00000000	5F5F4441	54410000	00000000	b	55			DATA	
2996	00000000	C02103	02	01000000	84D00500	00000000	00000000	04000000		21	1	Ň		
3024	00000000	000000	000	01000000	00000000	00000000	00000000	19000000						
3052	98000000	SFSF52	45	53545249	43540000	00000000	00000902	01000000	ò	R	ESTRI	СТ		
3080	00000000	000000	000	00300102	00000000	00000000	00000000	07000000			0			
3108	03000000	010000	00	04000000	5F5F7865	73747269	63740000	00000000				xe:	strict	
3136	5F5F5845	535452	49	43540000	00000000	00000902	01000000	00000000	X	ESTR	ICT			
3164	00000000	003001	02	00000000	00000000	00000000	00000000	00000000		0				
3192	00000000	000000	00	19000000	48000000	SFSF4C49	4E4B4544	49540000			1	н.	LINK	EDIT
3220	00000000	000009	02	01000000	00A00B00	00000000	00300102	00000000				t	0	
3248	409D0B00	000000	000	07000000	01000000	00000000	00000000	22000080	00					
3276	30000000	003001	.02	F0C70000	F0F70102	20080000	10000202	F8000000	0	0	<b>\$</b> << (	4 <sup>-</sup>	ÿ -	-
3304	08D10202	601F01	00	00000000	00000000	02000000	18000000	500E0602	-	•				Р
3332	650C0000	F42407	02	484F0100	08000000	50000000	00000000	01000000	e	Ù\$	HO	1	Р	
3360	01000000	000000	00	01000000	640C0000	00000000	00000000	00000000				d		
3388	00000000	000000	000	00000000	A0D40602	15140000	00000000	00000000				t'		
3416	00000000	000000	000	0E000000	20000000	00000000	2F757372	2F6C6962					/u	sr/l
3444	2F64796C	640000	000	00000000	18000000	18000000	2C702757	5943383B	/dy	ld			, p	'WYC
3472	85DE5B80	363CBD	44	32000000	20000000	01000000	000E0A00	000E0A00	Öfi	Ä6<Ω	D2			
3500	01000000	030000	000	0001C201	2A000000	10000000	05149000	03B00400			- '	•		ê ∞
3528	28000080	180000	000	66C1F500	00000000	00000000	00000000	00000000	(	Ä	fjì			
3556	70000000	180000	000	02000000	0000EB02	00000100	2F537973	74656D2F	р			Î	/S	yste
aned In	nt Clin	tle )	(se	lect less d	ata)									(
gricali			100	1001 1000 U								_		0

Renaming the \_\_\_\_RESTRICT segment of iTunes Mach-O binary.

Now let's inject Organismo into iTunes:

\$ DYLD\_INSERT\_LIBRARIES=/path-to/Organismo-mac.framework/Versions/A/Organismo-mac /Users/jongabilondo/Desktop/iTunes.app/Contents/MacOS/iTunes

This time Organismo framework was successfully loaded into iTunes.



## **Update for Catalina!**

Mac OSX Catalina has added new system protections. After changing the RESTRICT section the Apps and re-codesigning with the Dev Certificate it does result in a non valid App.

Music.app crashes on start. The logs hint to the the **taskgated-helper** and **amfid** processes. The error says: **Disallowing com.apple.Music because no eligible provisioning profiles found.** 

Here the Console logs:

error 11:06:47.743379+0200 error 11:06:47.743805+0200 default 11:06:47.743834+0200 default 11:06:47.743870+0200 default 11:06:47.743884+0200 default 11:06:47.744097+0200 default 11:06:47.744129+0200 default 11:06:47.744136+0200 default 11:06:47.744201+0200

The taskgated process belong to a daemon that can be found in:

/System/Library/LaunchDaemons/com.apple.taskgated.plistand the related:/System/Library/LaunchDaemons/com.apple.taskgated-helper.plist

Which executables are : /usr/libexec/taskgated and /usr/libexec/tasgated-helper

Rey TY Root E POSIXSpawnType S	/pe Dictionary	(5 items)	
POSIXSpawnType S	Dictionary	(5 items)	
POSIXSpawnType 5	tring		
	, un 9	Interactive	
EnableTransactions E	Boolean	YES	
Label S	String	com.apple.taskgated	
▼ MachServices E	Dictionary	(1 item)	
▼ com.apple.taskgated [	Dictionary	(1 item)	
TaskSpecialPort N	lumber	9	
▼ ProgramArguments A	\rray	(1 item)	
Item 0 S	String	/usr/libexec/taskgated	

The **amfid** is a daemon of the Apple Mobile File Integrity found in

/System/Library/LaunchDaemons/com.apple.MobileFileIntegrity.plist which launches /usr/libexec/amfi.

Key		Type		Value
Root	0	Dictionary	0	(6 items)
Label		String		com.apple.MobileFileIntegrity
▼ MachServices		Dictionary		(1 item)
▼ com.apple.MobileFileIntegrity		Dictionary		(1 item)
HostSpecialPort		Number		18
▼ LaunchEvents		Dictionary		(1 item)
▼ com.apple.fsevents.matching		Dictionary		(2 items)
▼ com.apple.MobileFileIntegrity.CodeRequirementPrefsChanged		Dictionary		(1 item)
Path		String		/Library/Preferences/com.apple.security.coderequirements.plist
▼ com.apple.MobileFileIntegrity.LibraryValidationPrefsChanged		Dictionary		(1 item)
Path		String		/Library/Preferences/com.apple.security.libraryvalidation.plist
		Array		(1 item)
Item 0		String		/usr/libexec/amfid
POSIXSpawnType		String		Interactive
EnablePressuredExit		Boolean		YES

This is a very interesting article on AMFI:

## AMFI: checking file integrity on your Mac

# Digging around looking at signature checking for apps in Mojave brought me in contact with a part of macOS with which I...

#### eclecticlight.co

We can assume that these two Apple Security daemons are rejecting our codesigned Music.app because the entitlements it requires are not accepted for our Dev Certificate. To confirm this we can create an entitlements file based on Music.app's entitlements and removing the entries that start *com.apple.private*.\*. The resulting Music.app will pass the AMFI and Gatekeeper security validation and it will launch, but it will fail in operations that require those entitlements.

These is a great WWDC video on Gatekeeper and other security updates for Catalina:

## Advances in macOS Security - WWDC 2019 - Videos - Apple Developer

We are on a journey to continuously improve macOS security, with a particular focus on preventing malware and...

developer.apple.com

#### **The Solution**

The ending of the first version of the article was this one:

For my purpose of making Organismo App Inspector still work in Catalina and coming versions, the direction to take is to get dyld, AMFI and the Gatekeeper 'out of the way', i.e. to disable their integrity validation.

The expression to 'get them out of the way', could have not been more premonitory. See the entry on AMFI on <u>iphonewiki</u>.

The amfi kext recognizes quite a few boot-args, including: — Allowing the above to proceed even without entitlement — Allowing any digital signature on code, not just Apple's — disable amfi — Disable code signing enforcement — Debug code signing

We can set a boot-arg to disable AMFI daemon process completely !

% sudo nvram boot-args="amfi\_get\_out\_of\_my\_way=0x1"(careful with the quotes if you copy-paste.)

Reboot. The AMFI will disappear from your system. Organismo is free to be injected into any App.

% DYLD\_INSERT\_LIBRARIES=/path/to/Organismo-mac.framework/Versions/A/Organismo-mac /System/Applications/Music.app/Contents/MacOS/Music



Organismo injected into Music.app

## Thanks !

I hope you enjoyed it. You may explore it yourself using Organismo.

## JonGabilondoAngulo/Organismo-Lib

# Organismo framework to be injected into Mac Apps to explore them at runtime...

github.com