



# iOS SDK 1.1.5.1 for Xcode

## User Guide



This document is intended only for authorized individuals of AppSealing customer.  
Unauthorized distribution of any parts of this document to other parties is prohibited.

The software referenced herein, this User Guide, and any associated documentation is provided to you pursuant to the agreement between your company, governmental body or other entity ("you") and INKA Entworks Corporation ("AppSealing") under which you have received a copy of AppSealing Licensed Technology and this User Guide (such agreement, the "Agreement"). Defined terms not defined herein shall have the meanings ascribed to them in the Agreement. In the event of conflict between the terms of this User Guide and the terms of the Agreement, the terms of the Agreement shall prevail. Without limiting the generality of the remainder of this paragraph, (a) this User Guide is provided to you for informational purposes only, (b) your right to access, view, use, and copy this User Guide is limited to the rights and subject to the applicable requirements and limitations set forth in the Agreement, and (c) all of the content of this User Guide constitutes "Confidential Information" of AppSealing (or the equivalent term used in the Agreement) and is subject to all of the limitations and requirements pertinent to the use, disclosure and safeguarding of such information. Permitting anyone who is not directly involved in the authorized use of AppSealing Licensed Technology by your company or other entity to gain any access to this User Guide shall violate the Agreement and subject your company or other entity to liability therefore.

## Copyright Information

Copyright © 2000-2021 INKA Entworks Corporation. All rights reserved.

AppSealing® is a trademark of INKA Entworks Corporation in the South Korea and/or other countries.

macOS™ and Xcode® are trademarks of Apple Inc., registered in the United States and other countries.

All other trademarks are the property of their respective owners.

## Disclaimer

The remainder of this User Guide notwithstanding, this User Guide is provided "as is", without any warranty whatsoever (including that it is error-free or complete). This User Guide contains no express or implied warranties, covenants or grants of rights or licenses, and does not modify or supplement any express warranty, covenant or grant of rights or licenses that is set forth in the Agreement. This User Guide is current as of the date set forth in the header that appears above on this page, but may be modified at any time without prior notice. Future revisions and updates of this User Guide shall be distributed as part of AppSealing SDK new releases. INKA Entworks shall under no circumstances bear any responsibility for your failure to operate AppSealing Licensed Technology in compliance with the then-current version of this User Guide. Your remedies with respect to your use of this User Guide, and INKA Entworks' liability for your use of this User Guide (including for any errors or inaccuracies that appear in this User Guide) are limited to those remedies expressly authorized by the Agreement (if any).

## Contact Information

Address: [06109] 1F 608, Nonhyeon-ro, Gangnam-gu, Seoul, South Korea

Technical support: [dev@appsealing.com](mailto:dev@appsealing.com)

Website: [www.appsealing.com](http://www.appsealing.com)

## 목차

요구 사항 ..... 4

### Part 1. AppSealing SDK의 위치

|  |   |
|--|---|
| 1-1 AppSealing SDK 프로젝트 폴더에 복사 .....             | 4 |
| 1-2 AppSealing SDK 압축 해제 .....                   | 5 |
| 1-3 프로젝트 Bundle ID 와 SDK에 등록된 Bundle ID 비교 ..... | 6 |

### Part 2. 프로젝트에 AppSealing 파일 추가

|   |    |
|---|----|
| 2-1 Xcode project 열기 .....  | 7  |
| 2-2 'File >> Add Files to "TestApp_Swift..."'를 이용하여 파일 추가 .....   | 7  |
| 2-3 Bridging header file에 AppSealing header 추가 (Swift 프로젝트) ..... | 10 |

### Part 3. 프로젝트 "Build Settings" 수정

|  |    |
|--|----|
| 3-1 프로젝트 메인 target 선택 .....  | 11 |
| 3-2 Build Settings "Other Linker Flags" 설정 .....                     | 11 |
| 3-3 Build Settings "Architectures – Excluded Architectures" 설정 ..... | 13 |
| 3-4 앱 빌드 모드에 따른 주의 사항 .....  | 16 |
| 3-5 앱 무결성 및 인증서 검증 정보 생성 .....                                       | 17 |
| 3-6 재서명 앱 App Store Connect 업로드 .....                                | 23 |

### Part 4. 보안 위협 안내를 위한 GUI 추가

|                                    |    |
|------------------------------------|----|
| 4-1 앱에서 UIAlertController 출력 ..... | 28 |
|------------------------------------|----|

### Part 5. AppSealing 기기 고유 ID 획득

|                       |    |
|-----------------------|----|
| 5-1 기기 고유 ID 확인 ..... | 31 |
|-----------------------|----|

### Part 6. 문제 해결

|   |    |
|---|----|
| 6-1 Xcode Build error (1) Use of undeclared type 'AppSealingInterface' .....  | 33 |
| 6-2 Xcode Build error (2) Undefined symbols for architecture arm64:<br>"_OBJC_CLASS_\$_AppSealingInterface" .....                     | 34 |
| 6-3 Xcode Build error (3) ld: library not found for -lStaticAppSec_Debug .....  | 36 |
| 6-4 Xcode Build error (4/5) Undefined symbols for architecture arm64:<br>"ObjC_IsAbnormalEnvironmentDetected()", "Appsealing()" ..... | 38 |
| 6-6 Execution error (1) 앱이 실행되고 곧바로 종료됩니다 .....   | 41 |
| 6-7 Execution error (2) 앱이 실행 중 갑자기 종료됩니다 .....   | 42 |
| 6-8 무결성/인증서 스냅샷 생성 스크립트가 실행되지 않습니다 .....  | 43 |
| 6-9 Continuous Integration Tools에서 AppSealing SDK 사용 .....  | 43 |

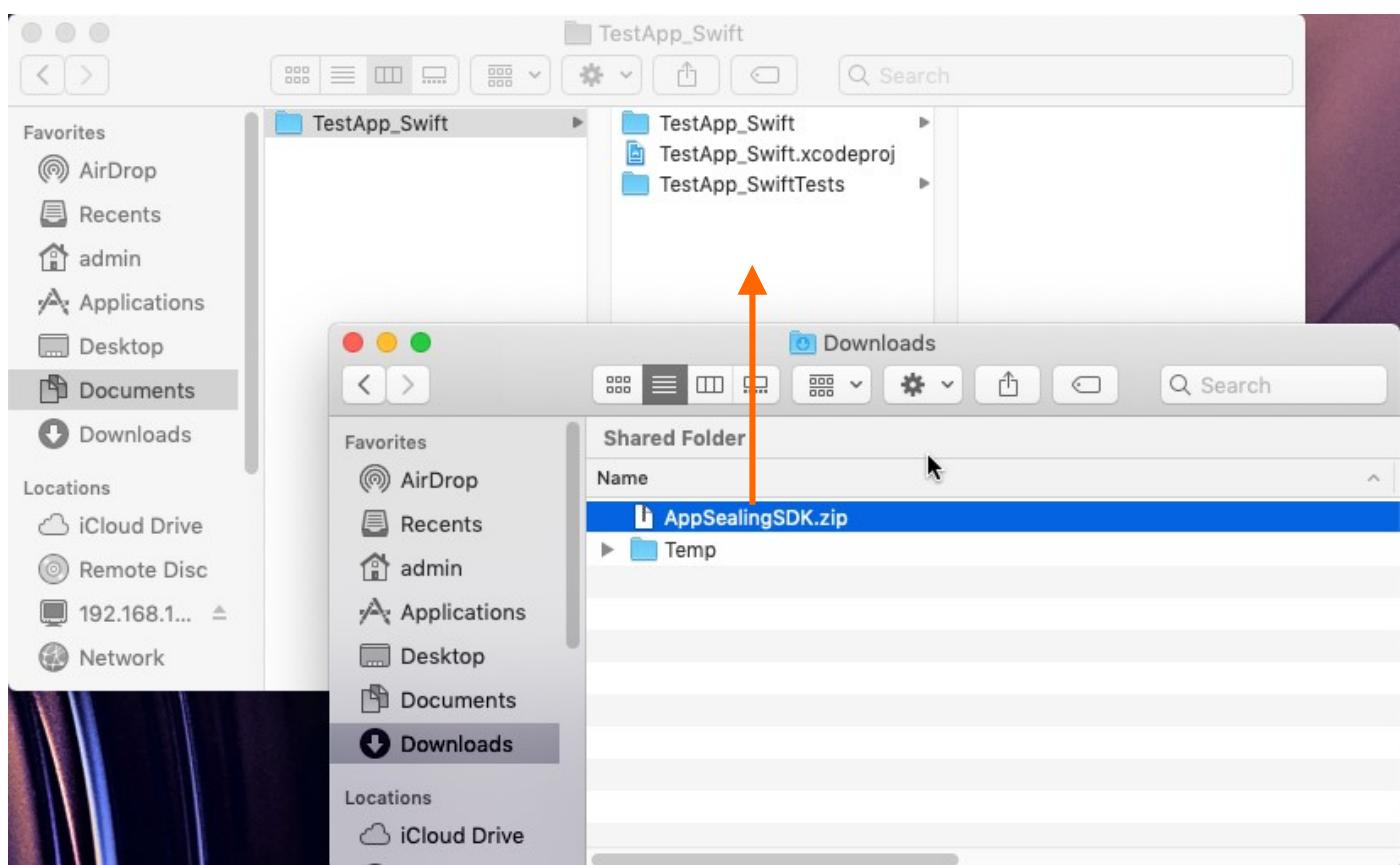
## Prerequisite

- Xcode 10.1 이상
- iOS 9.0 이상 기기

## Part 1. AppSealing SDK의 위치

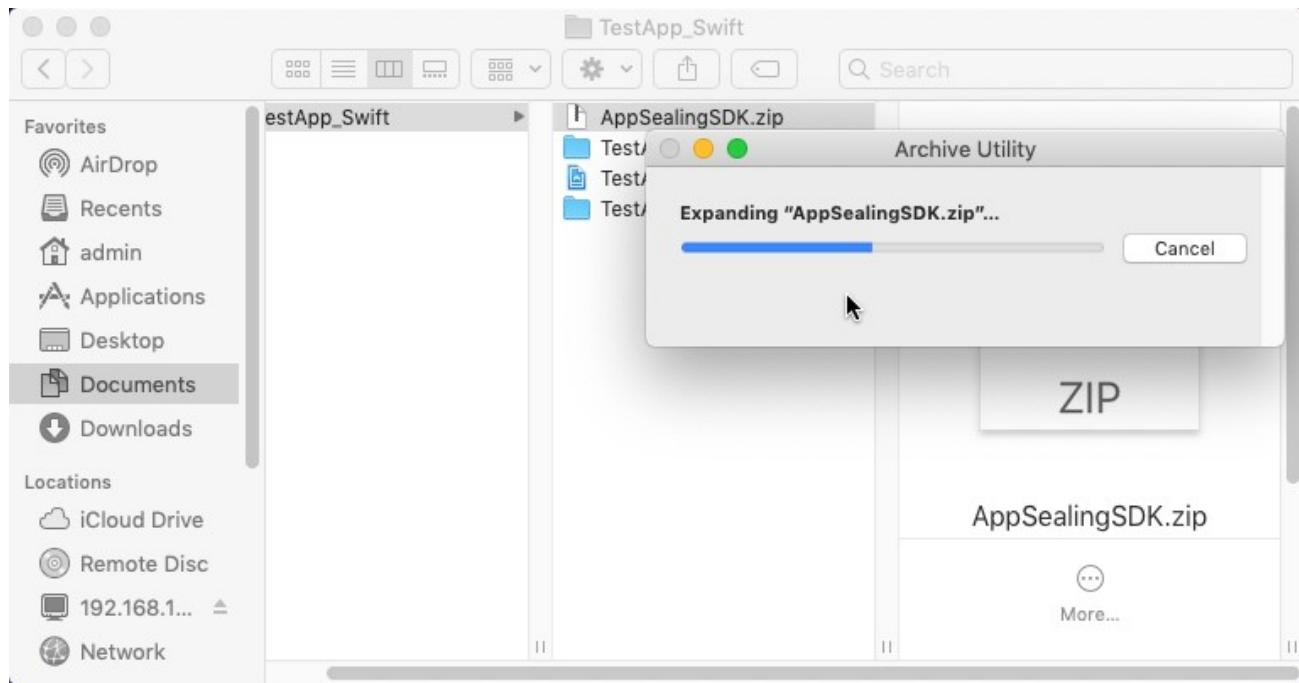
AppSealing SDK zip 파일을 다운로드한 후 Xcode 프로젝트 폴더에 압축을 해제합니다. 이 문서에서 안내되는 샘플 프로젝트 'TestApp\_Swift'는 "~/Documents/TestApp\_Swift" 경로에 위치합니다.

### 1-1 AppSealing SDK 프로젝트 폴더에 복사



## 1-2 AppSealing SDK 압축 해제

복사한 AppSealingPlugin.zip 파일을 더블클릭 하여 압축을 해제합니다.



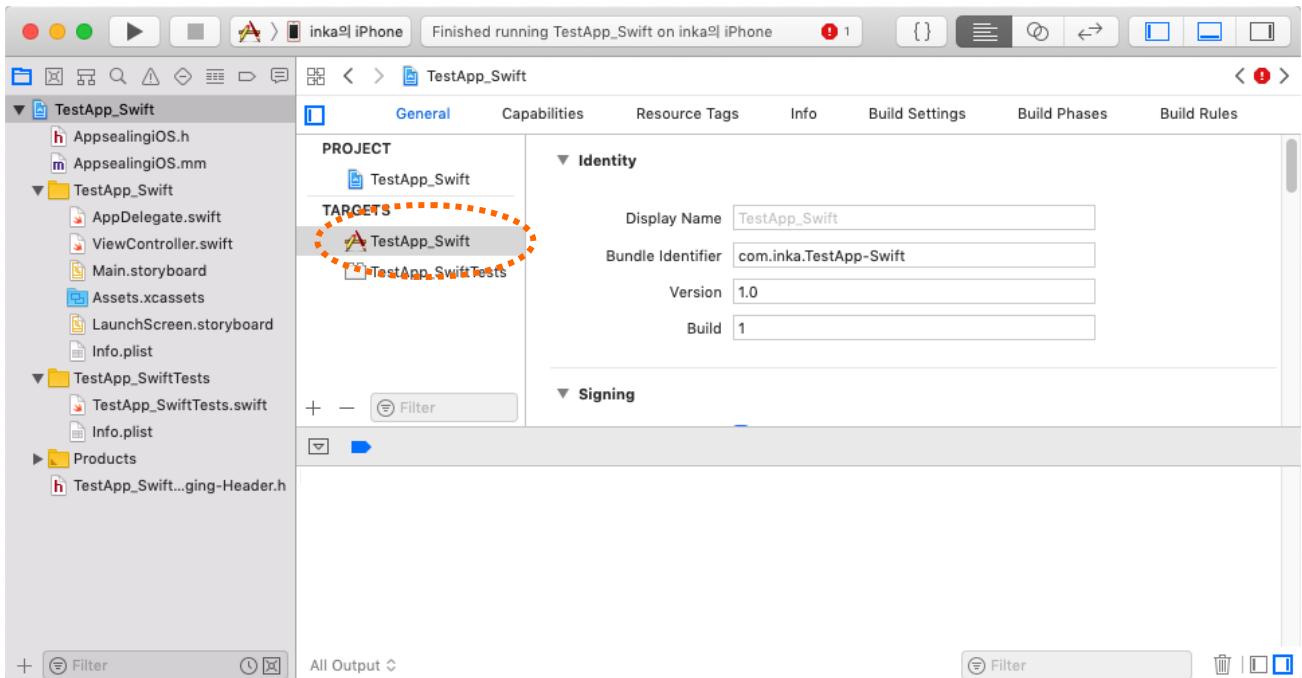
Zip 파일 압축 해제 후 다음과 같은 구조처럼 모든 파일이 생성되는지 확인하십시오.

| Name   | Size      | Kind         |
|--|-----------|--------------|
| AppSealingSDK                                    | --        | Folder       |
| Documents  | --        | Folder       |
| iOS_AppSealingSDK_Use..._for_XcodeProject_JP.pdf | 3.9 MB    | PDF          |
| iOS_AppSealingSDK_Use..._for_XcodeProject_KR.pdf | 2.8 MB    | PDF          |
| iOS_AppSealingSDK_Use..._for_XcodeProject_EN.pdf | 3.7 MB    | PDF          |
| Libraries  | --        | Folder       |
| AppsealingiOS.mm                                 | 3 KB      | ObjC++       |
| AppsealingiOS.h                                  | 925 bytes | C He...ource |
| libStaticAppSec.a                                | 79.9 MB   | Document     |
| libStaticAppSec_Debug.a                          | 99.3 MB   | Document     |
| appsealing.lic                                   | 89 bytes  | TextE...ment |
| Release-Notes.txt                                | 738 bytes | Plain Text   |
| Tools  | --        | Folder       |
| generate_hash                                    | 12 KB     | TextE...ment |

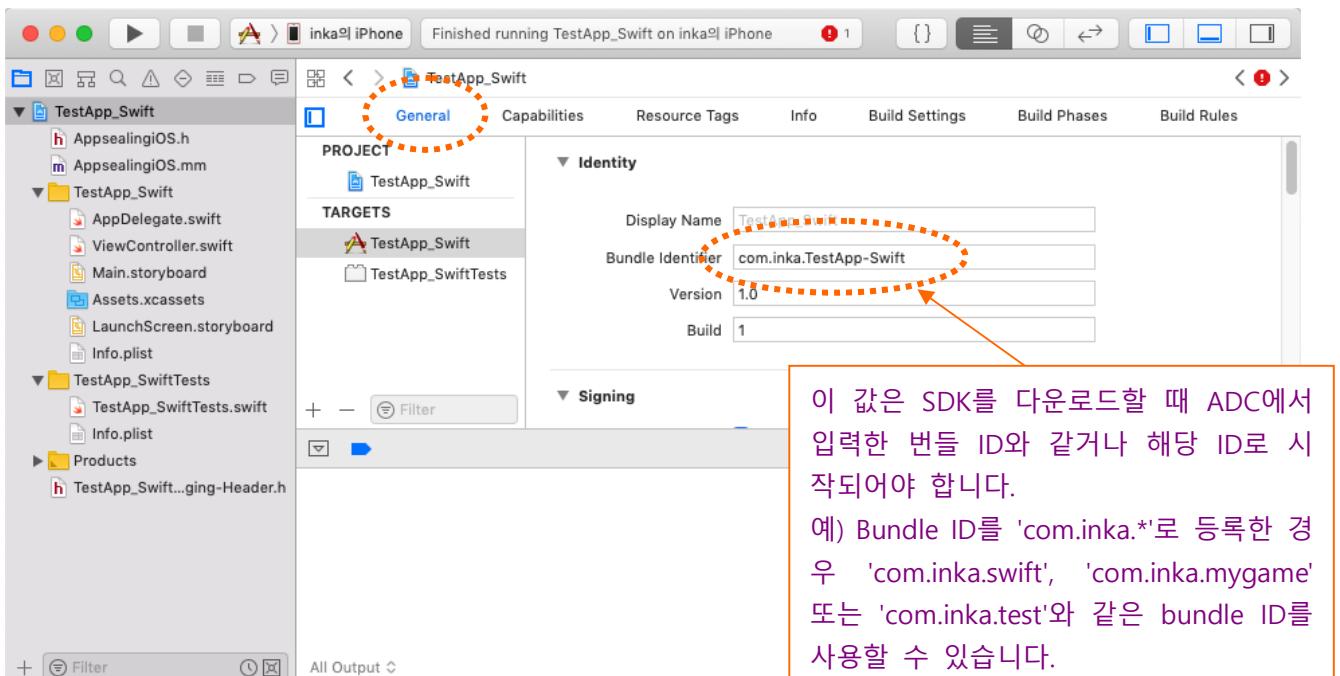
파일이 없는 경우 SDK를 다시 내려 받거나 zip 파일 압축을 다시 해제해 보십시오.

### 1-3 프로젝트 Bundle ID 와 SDK에 등록된 Bundle ID 비교

SDK zip 파일을 압축 해제하고 프로젝트의 bundle ID가 유효한지 확인하십시오. Xcode를 열고 왼쪽 패널에서 프로젝트를 선택하고 Targets 목록에서 프로젝트 이름 (TestApp\_Swift)를 선택하십시오.

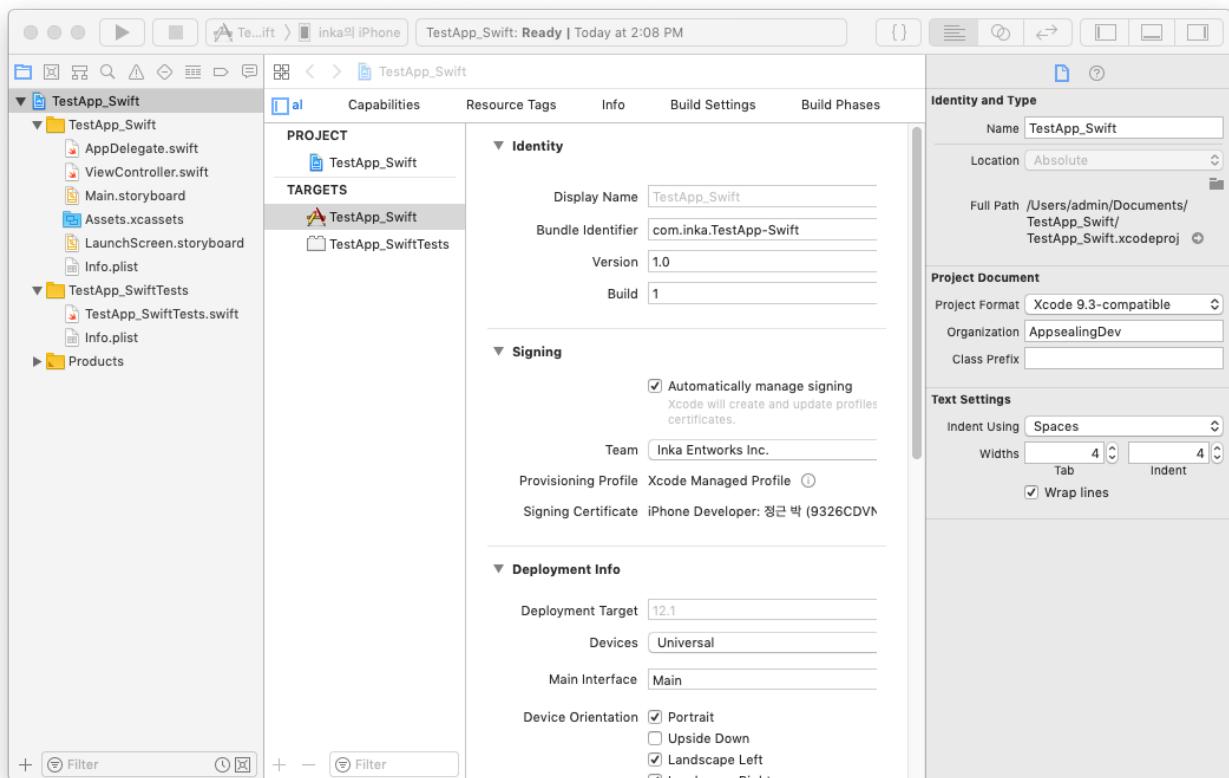


그 다음 “General” 탭을 선택하고 bundle ID가 AppSealing Developer Center (ADC)에서 SDK를 다운로드할 때 입력한 값과 같거나 해당 값으로 시작되는지 확인하십시오.

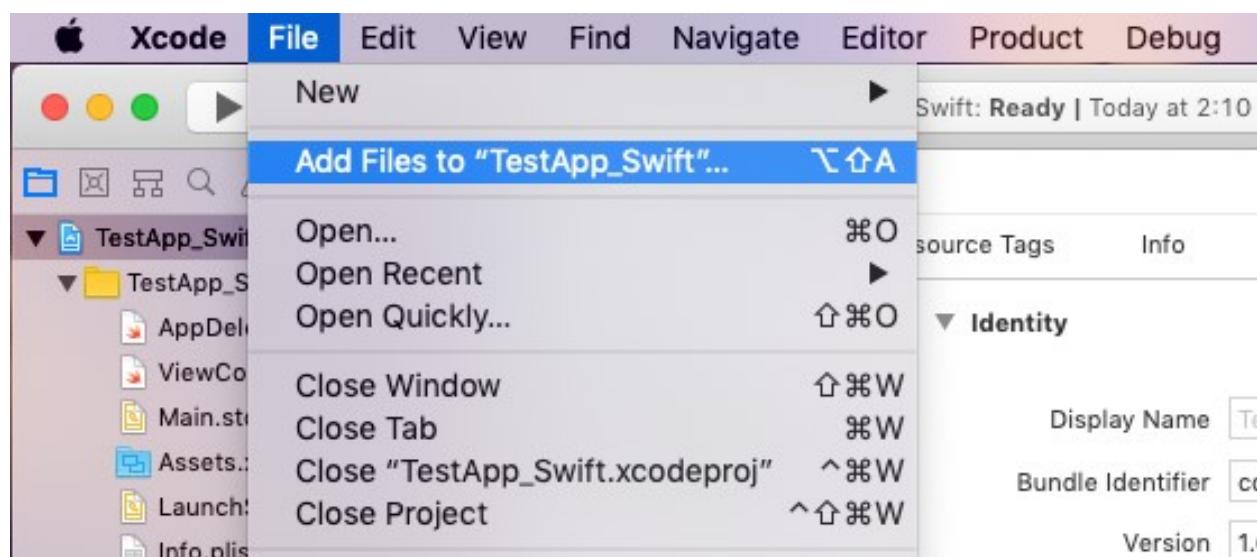


## Part 2. 프로젝트에 AppSealing 파일 추가

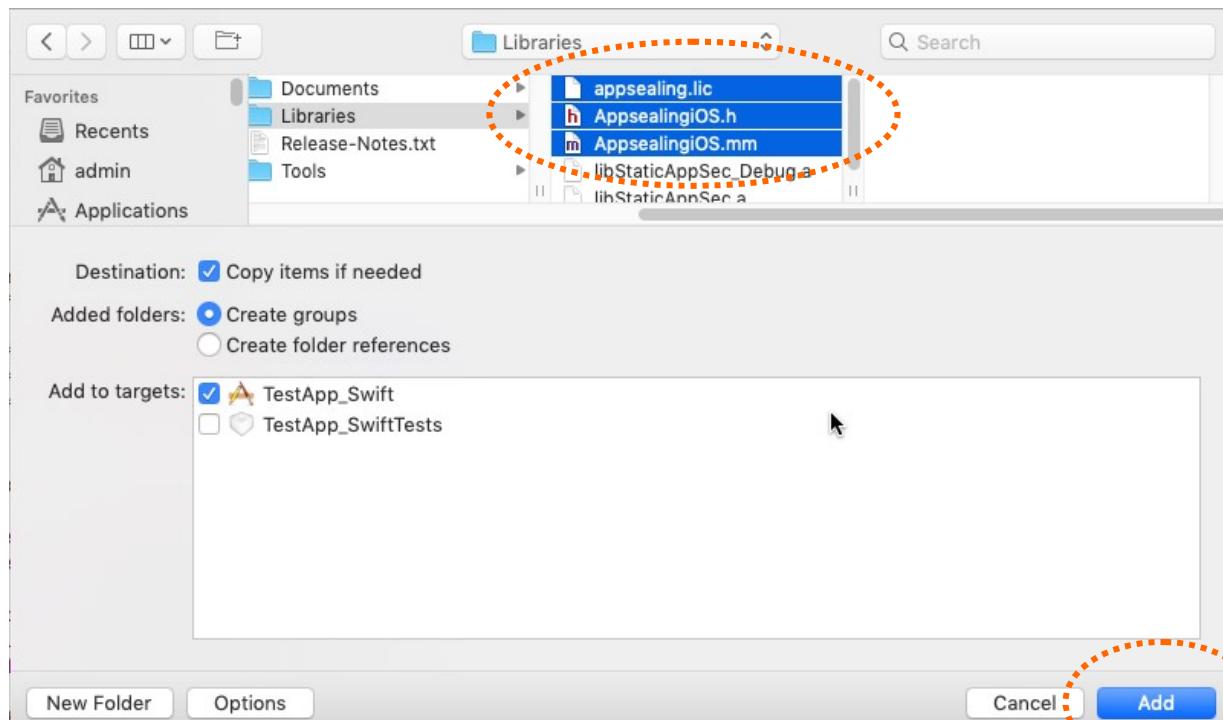
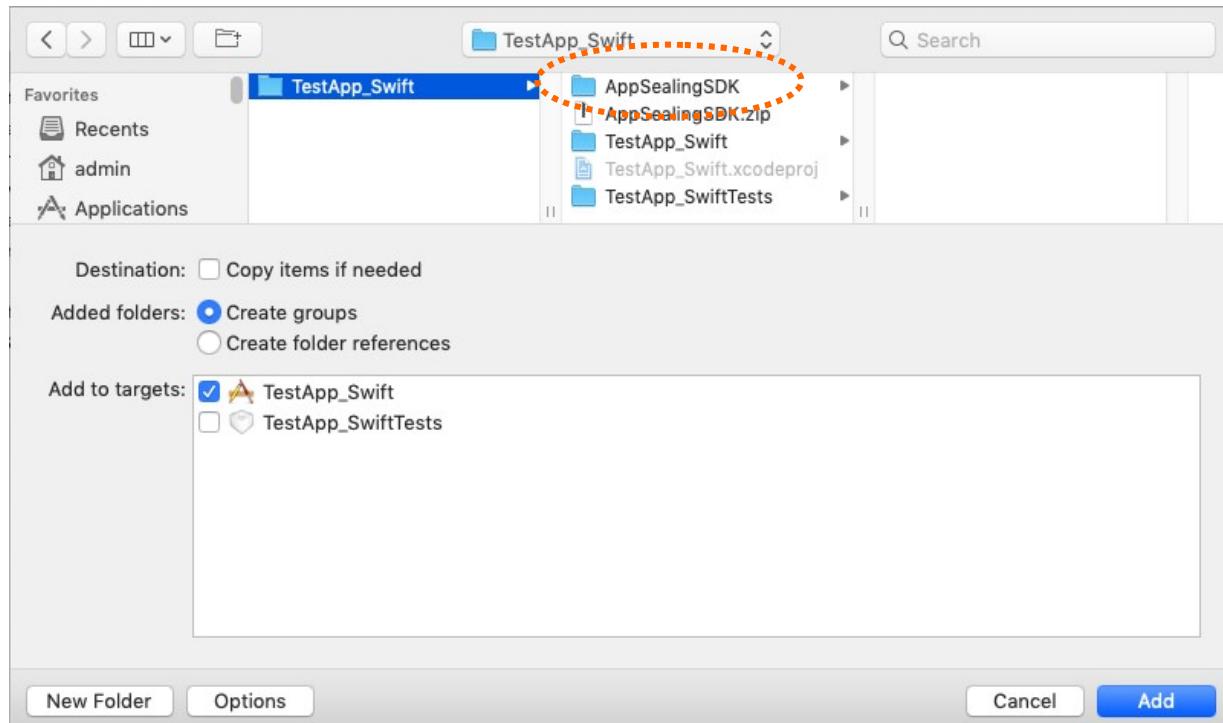
### 2-1 Xcode Project 열기



### 2-2 'File >> Add Files to "TestApp\_Swift"...' 를 이용하여 파일 추가



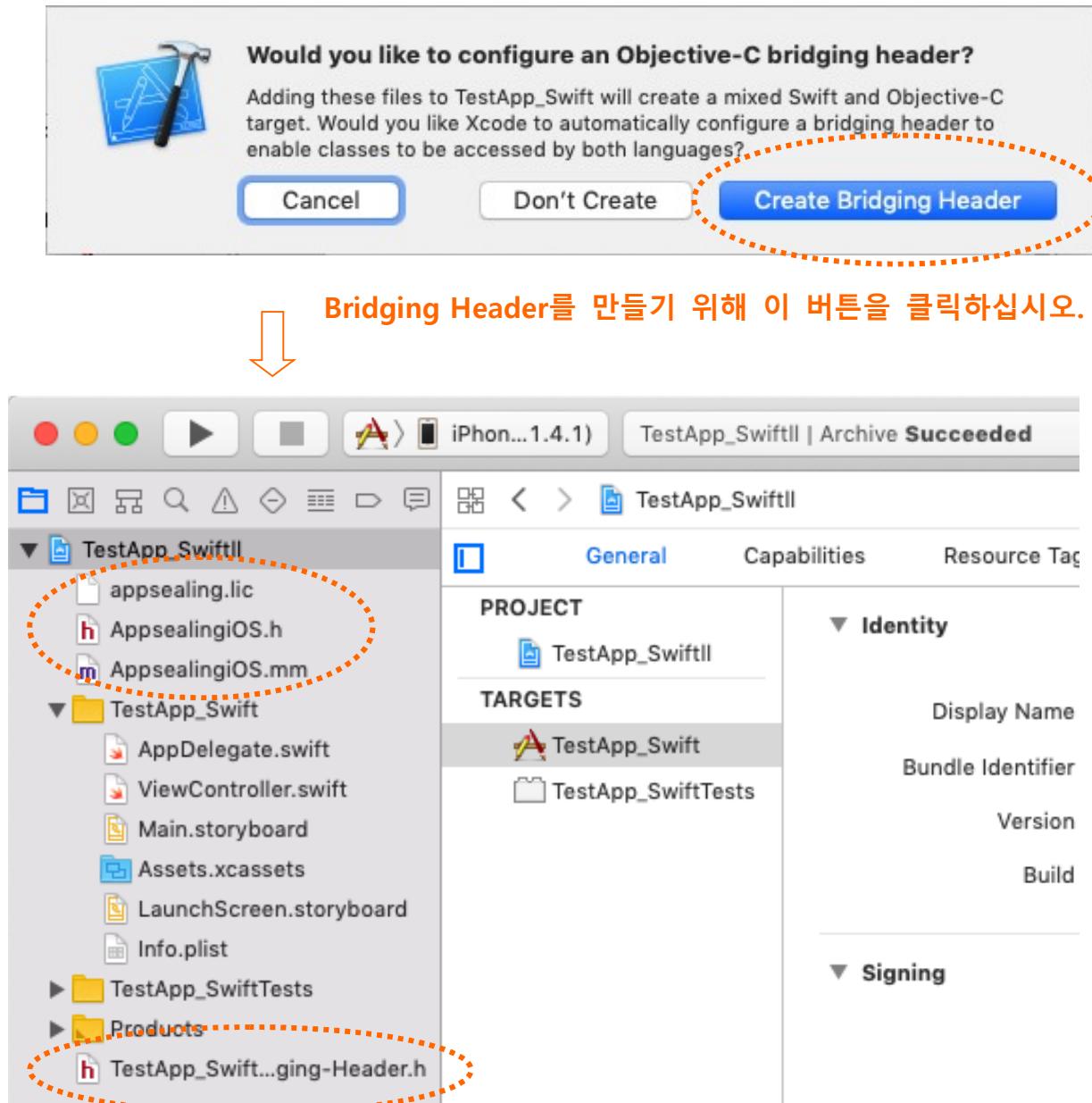
대화 상자에서 "AppSealingSDK/Library/" 폴더의 **"appsealing.lic"**, **"AppsealingiOS.h"** and **"AppsealingiOS.mm"**를 선택하고 "Add" 버튼을 눌러 프로젝트에 추가하십시오.



파일 추가를 위해 버튼을 누르십시오

**프로젝트가 Swift 기반인 경우에만 이 부분을 진행하십시오.**  
**(Objective-C 기반 프로젝트인 경우 이 페이지와 다음 페이지를 참조하지 마십시오.)**

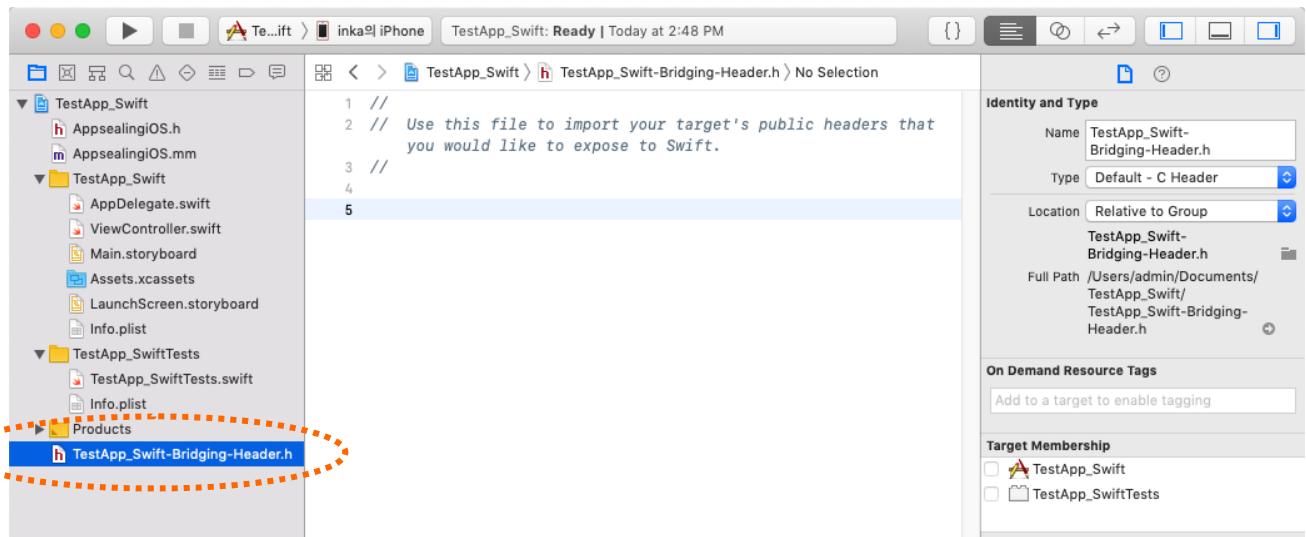
프로젝트가 swift 기반에서 프로젝트에 bridging header가 없는 경우 bridging header를 생성하라는 대화 상자가 나타날 수 있습니다. "Create Bridging Header"을 클릭하여 bridging header를 만드십시오.



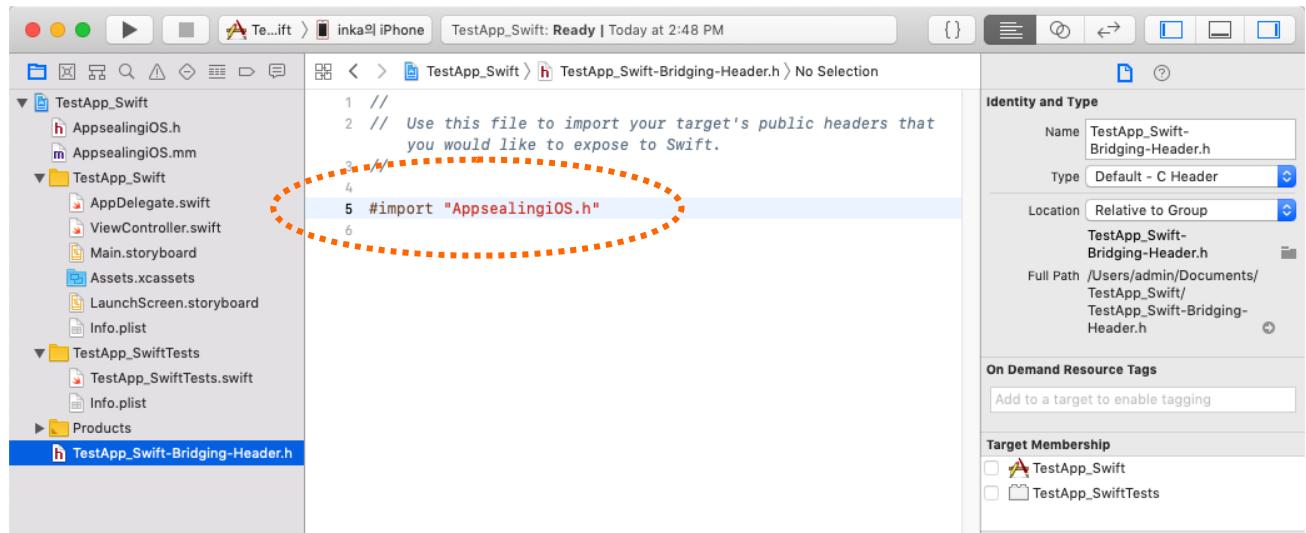
\* 프로젝트에 이미 bridging header가 있는 경우, bridging header 생성 대화상자가 나타나지 않고 기존 bridging header 파일을 사용할 수 있습니다.

\*\* 프로젝트가 Objective-C 기반인 경우 bridging header 파일은 사용되지 않습니다.

### 2-3 Bridging header 파일에 AppSealing header 추가 (Swift-project)



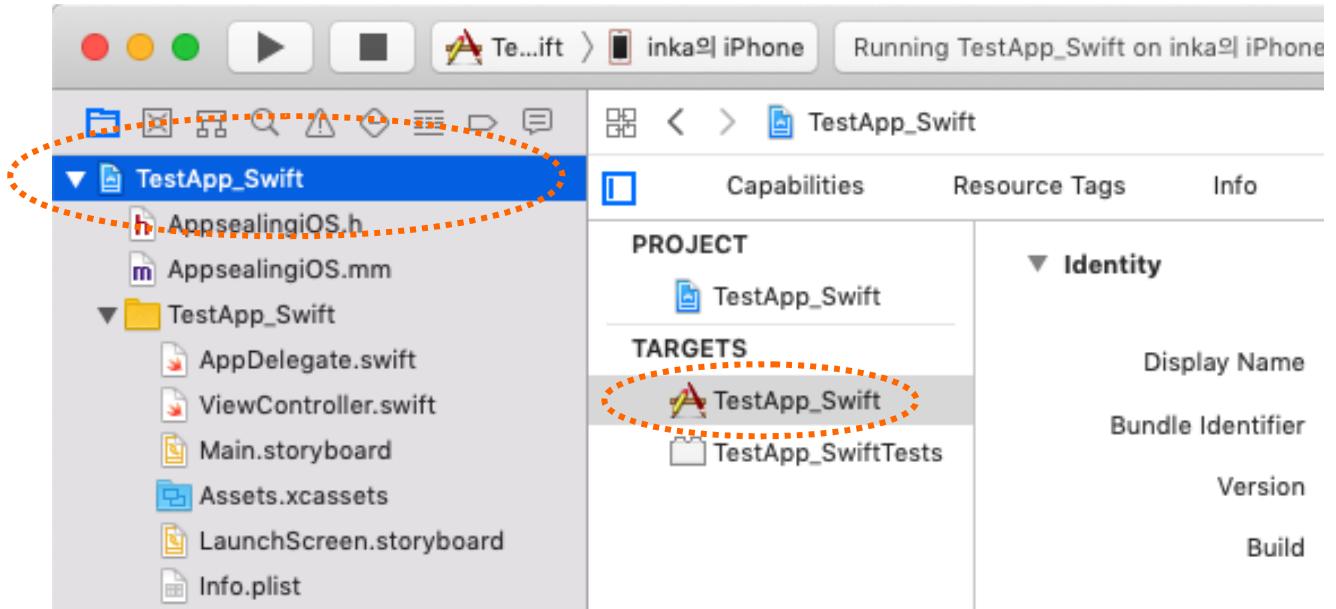
새로 생성된 "TestApp\_Swift-Bridging-Header.h" 파일을 선택하고 `#import "AppsealingiOS.h"`를 문서 하단에 추가하십시오.



## Part 3. 프로젝트 "Build Settings" 설정

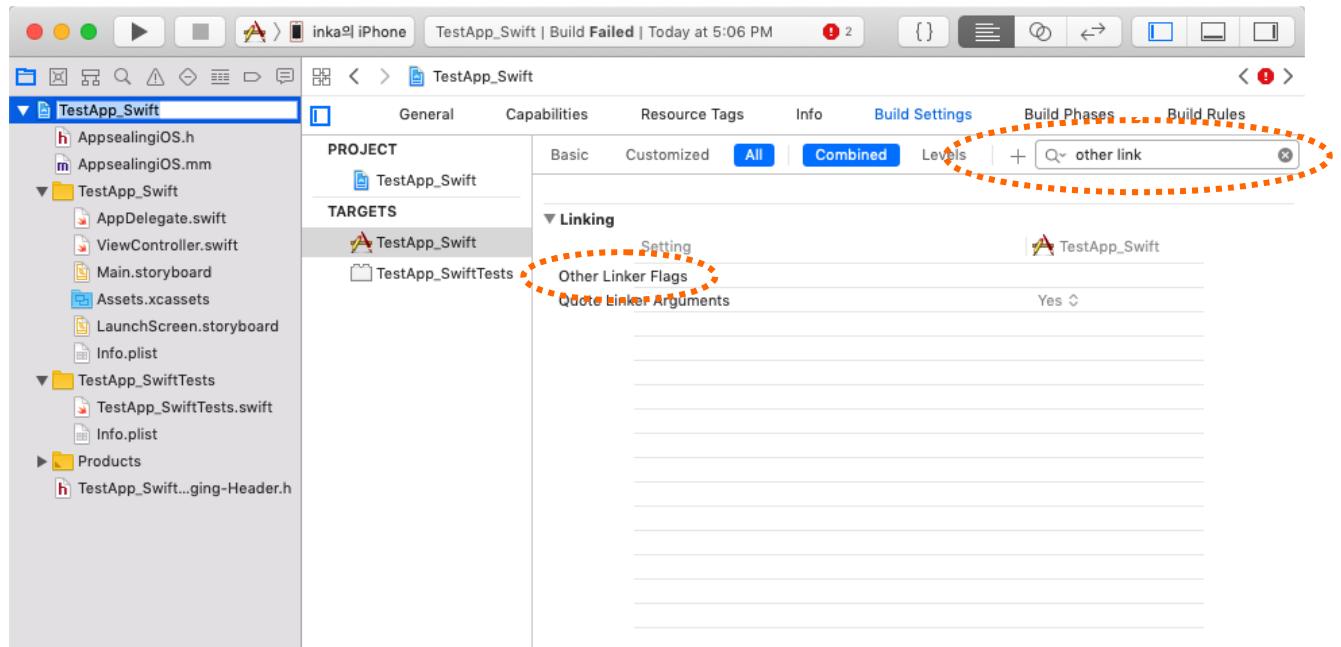
### 3-1 프로젝트 메인 target 선택

프로젝트 네비게이터에서 프로젝트를 선택하고 TARGETS에서 메인 타깃인 "TestApp\_Swift"를 선택합니다.



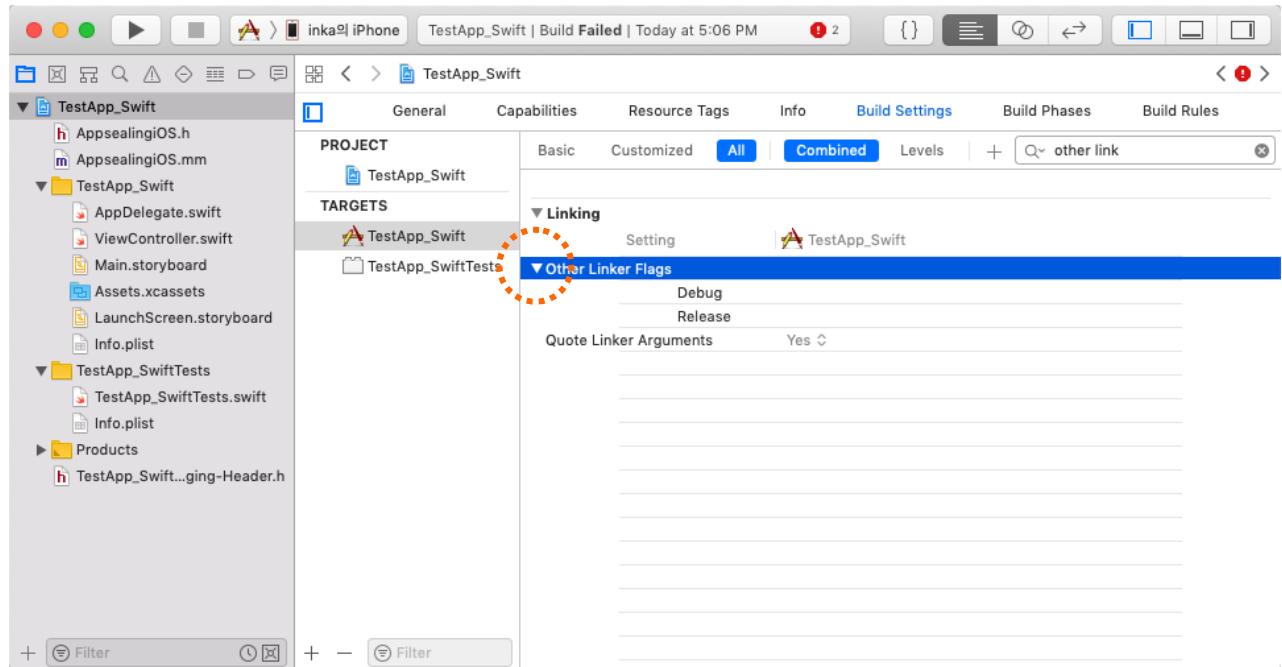
### 3-2 Build Settings "Other Linker Flags" 설정

프로젝트 선택 후 Build Settings에서 other link를 검색하십시오.



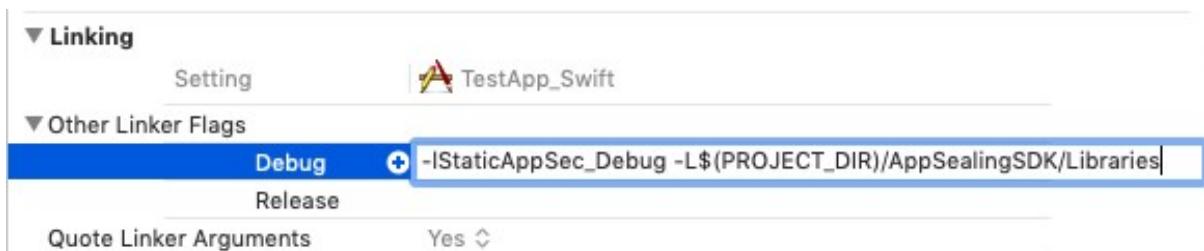
"Other Linker Flags" 필드가 나타나면 다음 단계를 수행하여 값을 설정하십시오.

1) "Other Linker Flags" 왼쪽 삼각형 아이콘을 눌러 "Other Linker Flags"를 확장하십시오.



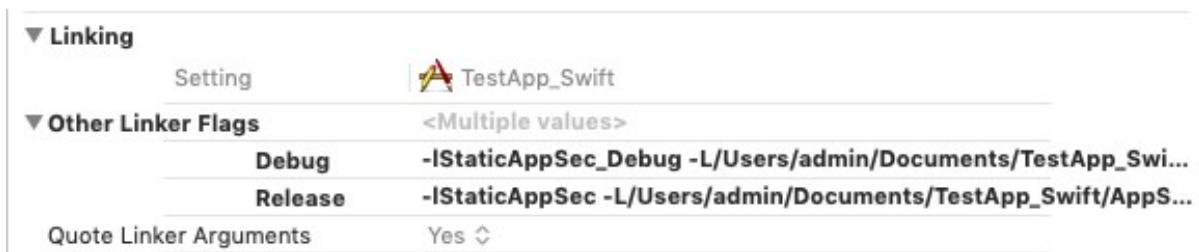
2) "Debug" 아이템을 선택하고 클릭하여 아래 텍스트를 입력하십시오.

*-IStaticAppSec\_Debug -L\$(PROJECT\_DIR)/AppSealingSDK/Libraries*



3) "Release" 아이템을 선택하고 클릭하여 아래 텍스트를 입력하십시오.

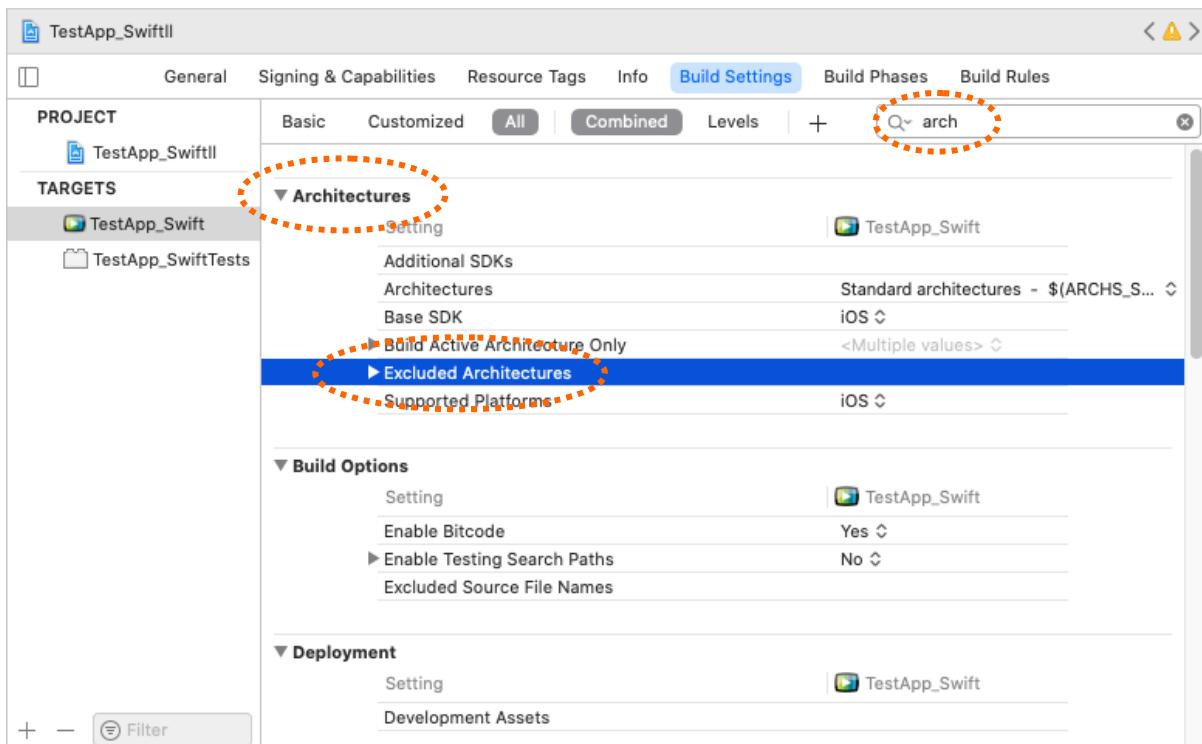
*-IStaticAppSec -L\$(PROJECT\_DIR)/AppSealingSDK/Libraries*



다음은 Simulator에서 Release 빌드 실행이 필요한 경우를 위한 설정 내용입니다. Simulator에서 Debug 빌드만 실행하고 Release 빌드를 실행할 상황이 없다면 아래 단계는 진행하지 않아도 됩니다.

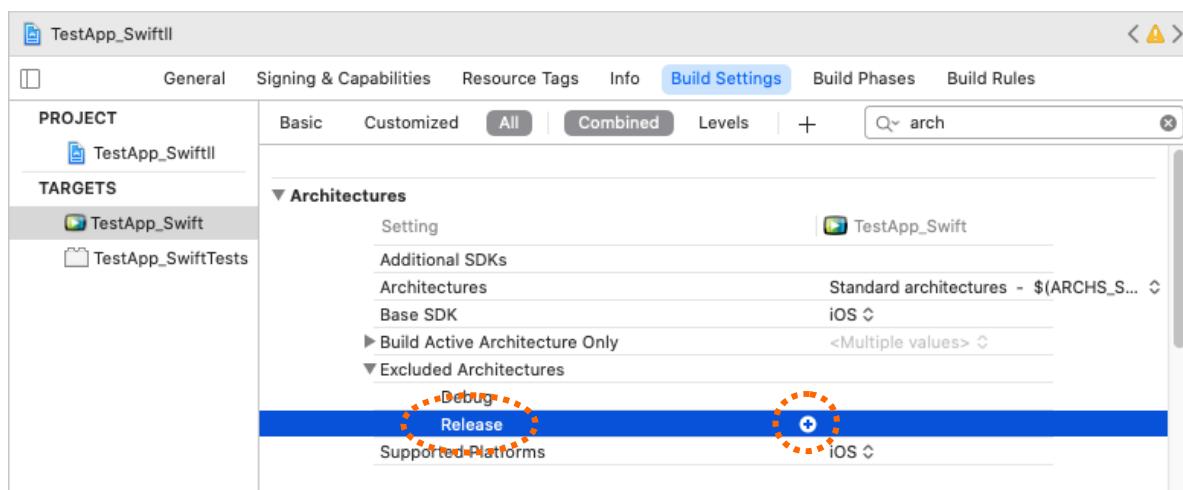
### 3-3 Build Settings "Architectures – Excluded Architectures" 설정

먼저 Build Settings에서 arch를 검색하십시오.

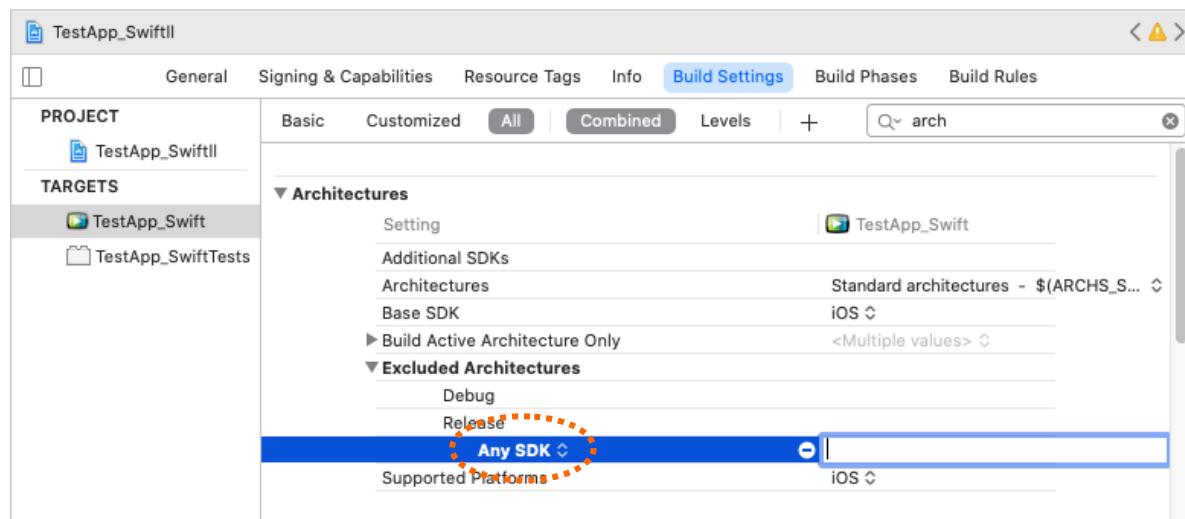


"Architectures" 필드가 나타나면 다음 단계를 수행하여 값을 설정하십시오.

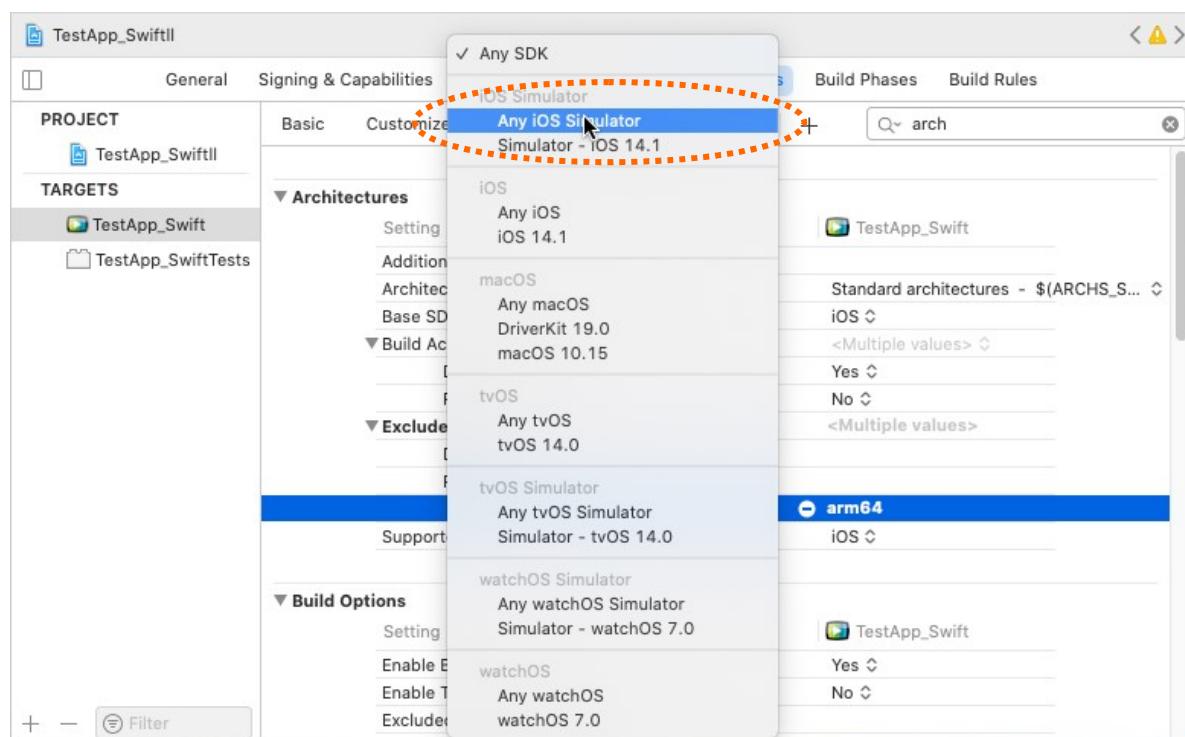
- 1) "Excluded Architectures" 왼쪽 삼각형 아이콘을 눌러 확장하십시오.



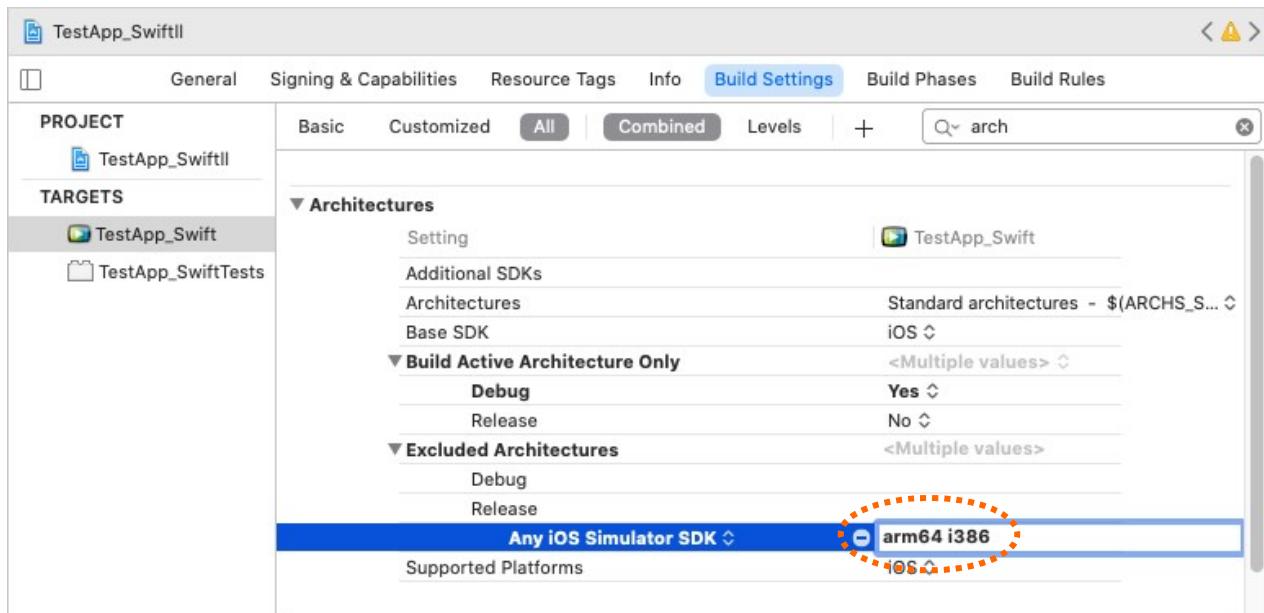
2) "Release" 항목을 선택하고 오른쪽의 "+" 아이콘을 클릭하십시오.



3) "Any SDK" 항목이 추가되면 클릭하여 팝업 메뉴를 띄우십시오.



4) 팝업 메뉴의 항목 중에서 “Any iOS Simulator”를 선택합니다.



5) “Any iOS Simulator SDK” 우측의 입력란에 “**arm64 i386**” 값을 입력합니다. (arm64 와 i386 사이는 공백이 들어갑니다)

이제 AppSealing 보안 기능이 프로젝트에 추가되었습니다. 평소와 같이 'Build', 'Run' 또는 'Archive'를 수행하십시오.

### 3-4 앱 빌드 모드에 따른 주의 사항

\* AppSealing은 디버그 모드와 릴리스 모드에서 다르게 동작합니다.

디버그 모드로 빌드 하는 경우 다음의 AppSealing 보안 기능은 원활한 개발을 위해서 비활성화됩니다:

- 탈옥 감지 : Jailbreak detection
- 안티 디버깅 : Anti-debugging
- 실행 파일 암호화 감지 : Not encrypted executable file detection
- 앱 무결성 감지 : App-Integrity corruption detection
- 앱 재서명 감지 : Re-signing detection

이 기능들은 릴리스 모드로 빌드 하면 활성화됩니다.

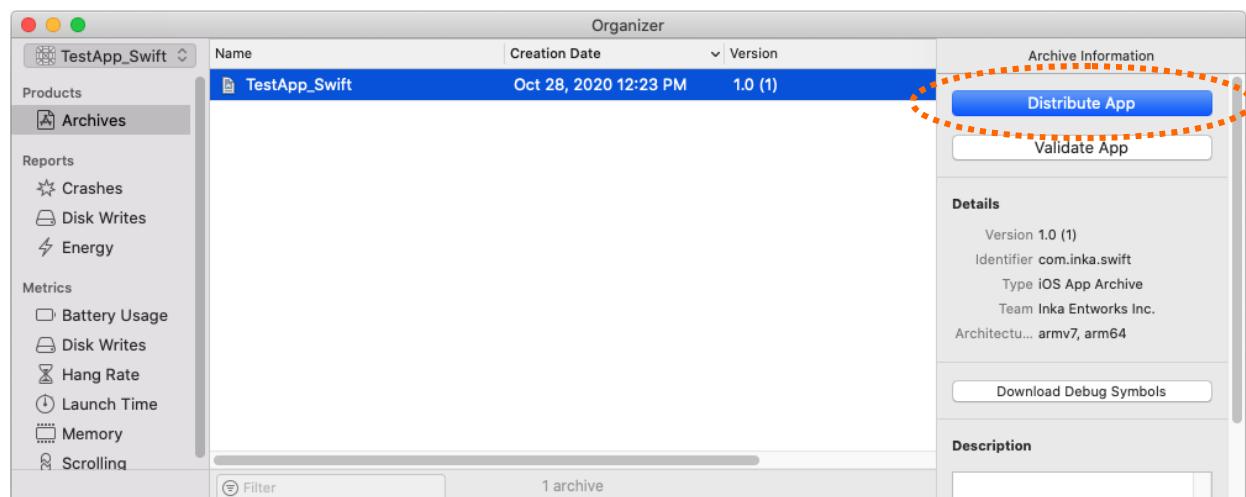
앱 스토어에 배포할 때 앱을 릴리스 모드로 빌드 하십시오. 릴리스 모드로 AppSealing 을 테스트 할 경우 앱스토어 또는 TestFlight로 앱을 배포해야 합니다. 그렇지 않으면 실행 파일이 암호화되지 않은 것이 탐지되거나 앱이 변조된 것으로 탐지되어 일정 시간 후 앱이 종료됩니다.

### 3-5 앱 무결성 및 인증서 검증 정보 생성

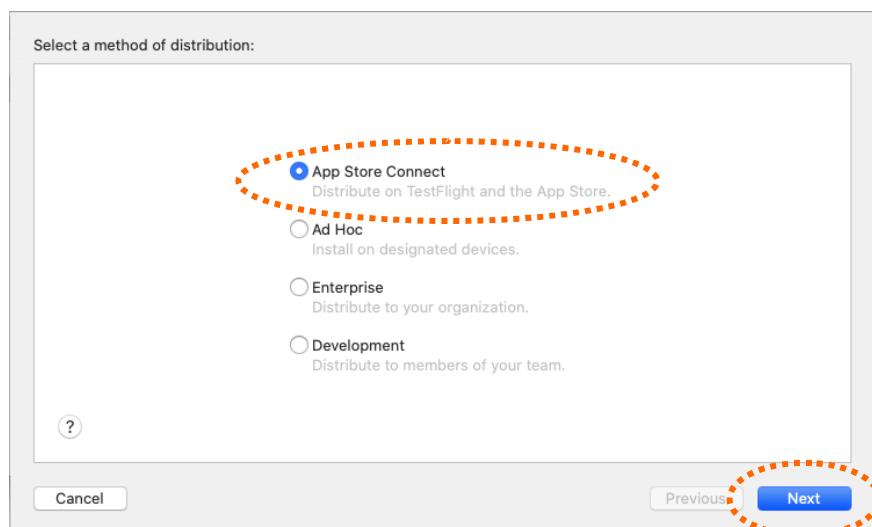
애플 테스트하거나 스토어에 배포할 경우 앱의 무결성 검증을 위한 추가 작업을 진행해야 합니다. 이 단계를 건너뛸 경우 단말에서 앱을 실행하면 무결성 검증에 실패하여 일정 시간 후 앱이 종료됩니다.

Release 모드로 빌드된 앱을 Development나 Ad Hoc으로 배포할 경우는 실행파일 암호화 미적용으로 앱 실행 시 보안 체크 로직에 의해 일정 시간 후 앱이 종료되기 때문에 이 단계를 생략해도 결과는 동일하지만 **TestFlight나 App Store 배포 시에는 반드시 이 단계를 수행해야 합니다.**

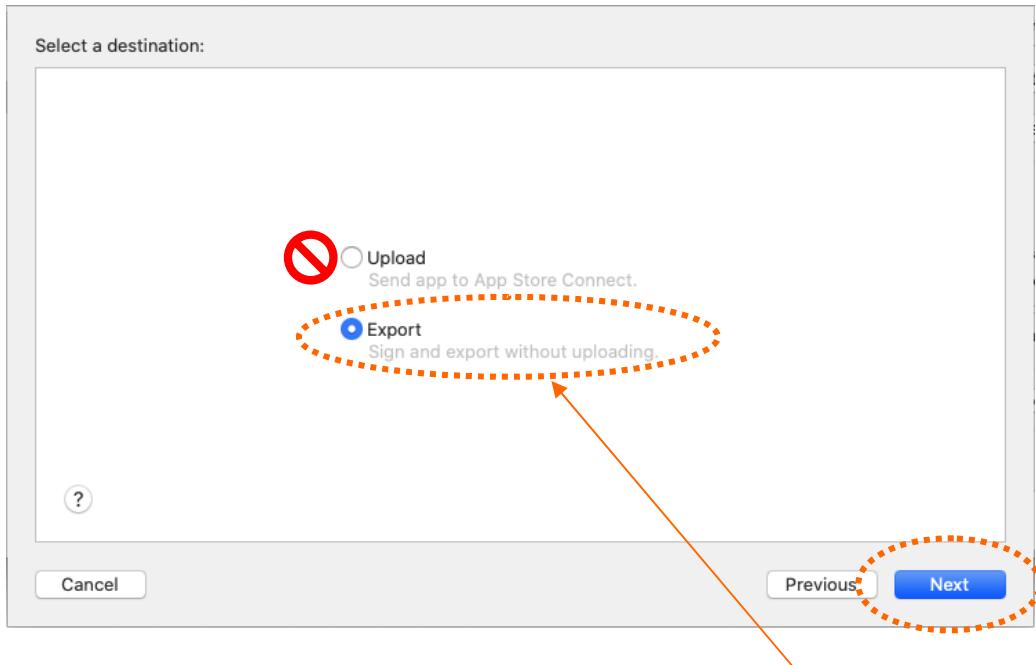
애플 스토어나 TestFlight에 앱을 업로드하기 위한 과정을 단계 별로 살펴 보겠습니다. 아래는 Archive가 완료된 상태의 Organizer 화면입니다.



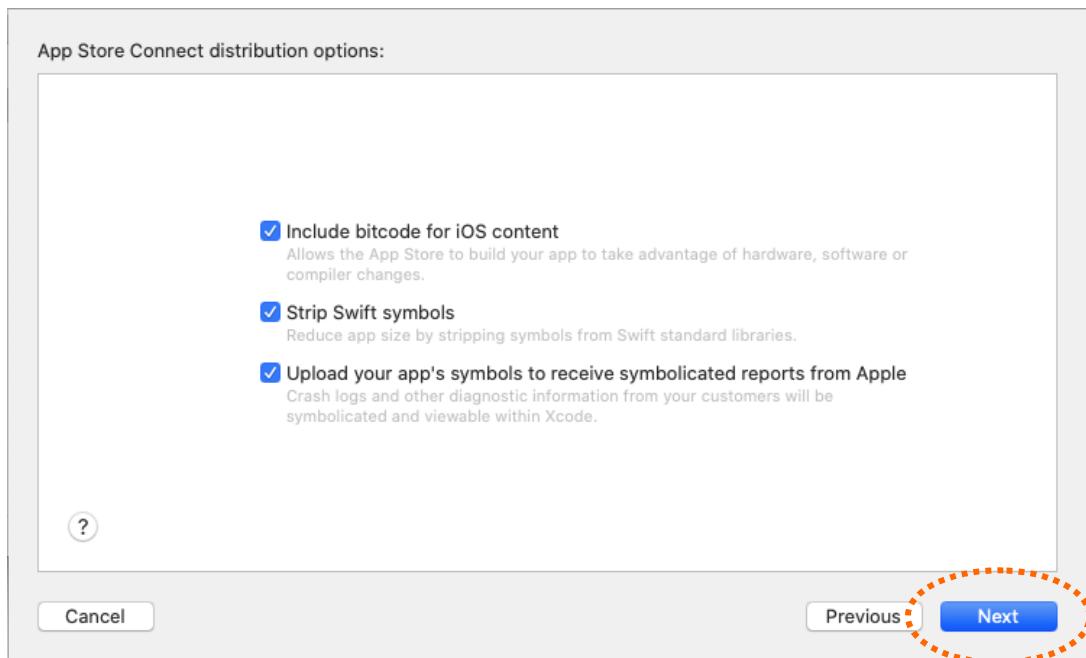
이 상태에서 업로드 할 IPA를 생성하기 위해 "Distribute App" 버튼을 클릭합니다.



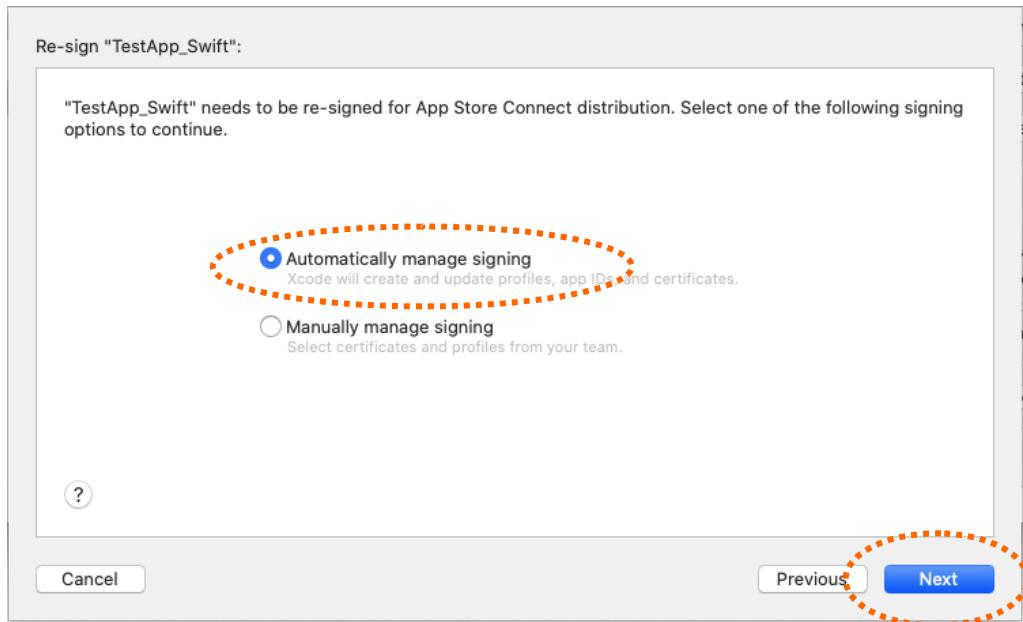
"App Store Connect"가 선택된 상태에서 "Next" 버튼을 클릭합니다.



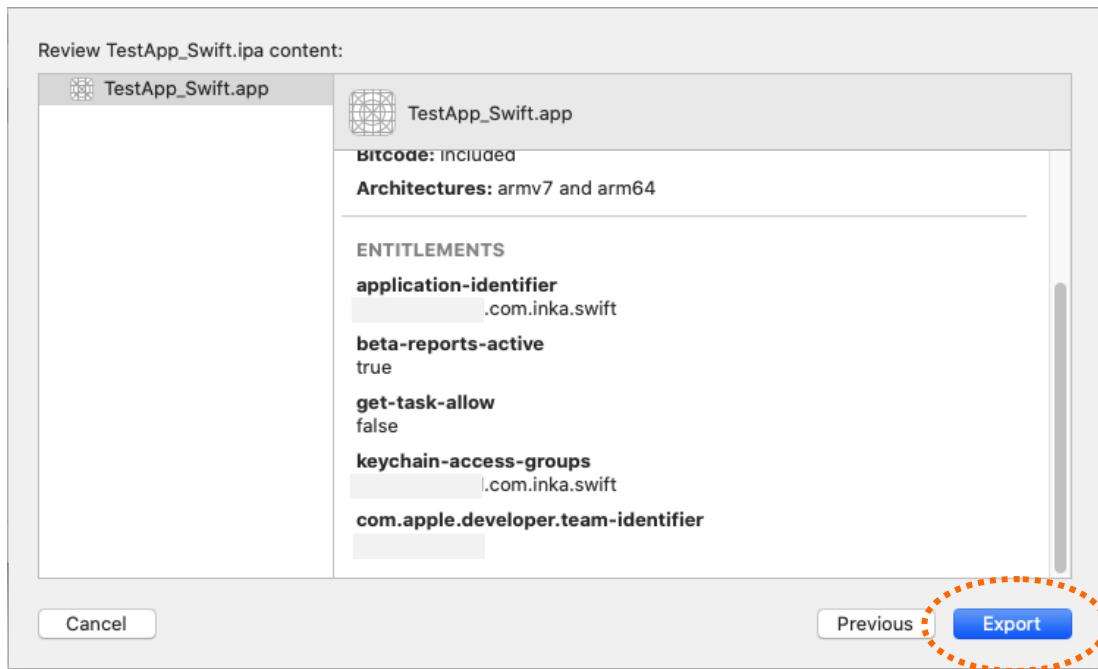
기존에는 "Upload"를 선택하는 경우가 대부분이지만 **AppSealing 적용을 위해서는 반드시 "Export"를 선택해야 합니다.** 이는 업로드 전 무결성 및 인증서 스냅샷 처리 작업이 필요하기 때문으로 **이 과정이 잘못되거나 생략되면 앱이 정상적으로 실행되지 않으므로 유의해야 합니다.** "Export"가 선택된 상태에서 "Next" 버튼을 클릭합니다.



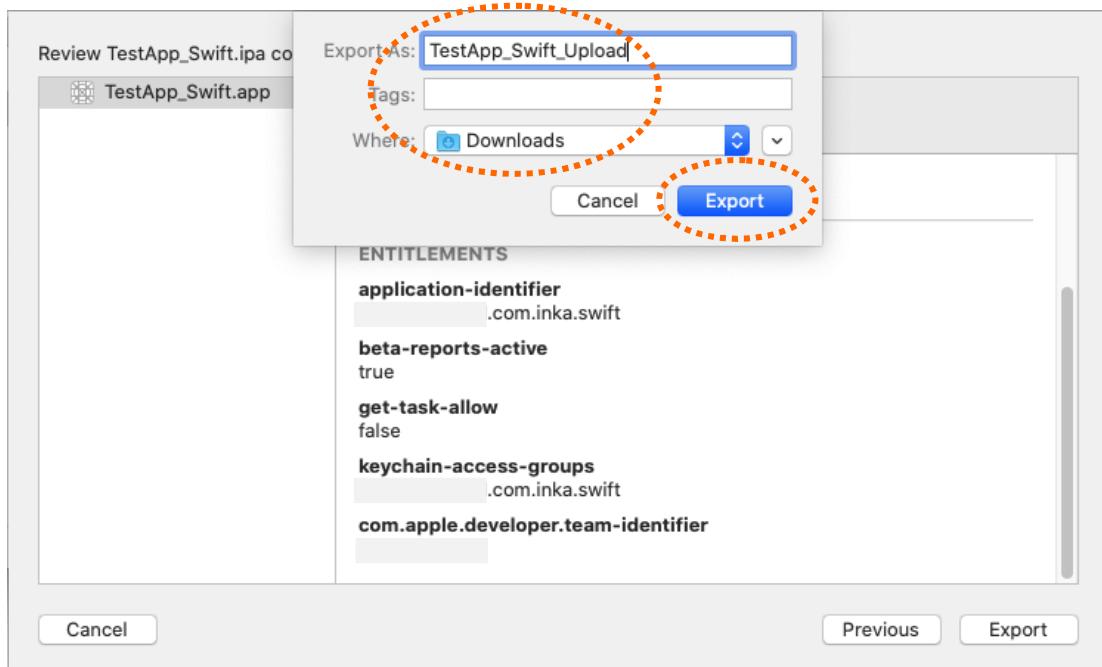
옵션 선택은 그대로 둔 상태로 "Next" 버튼을 클릭합니다.



옵션 선택은 그대로 둔 상태로 "Next" 버튼을 클릭합니다. 이제 아래와 같이 IPA 파일로 export 할 수 있는 창이 나타납니다.

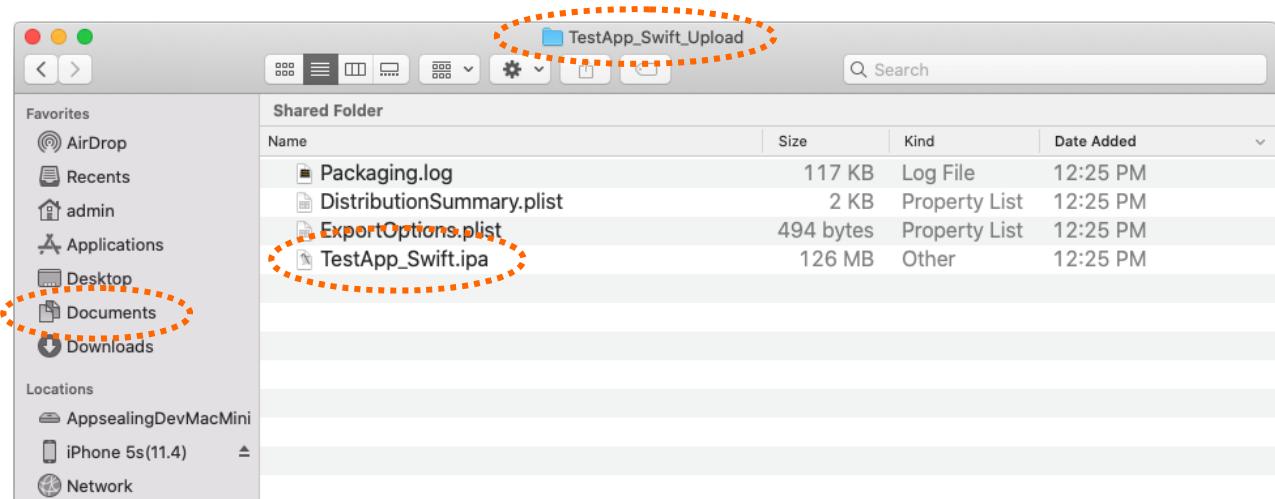


내용을 확인한 후 "Export" 버튼을 클릭합니다.

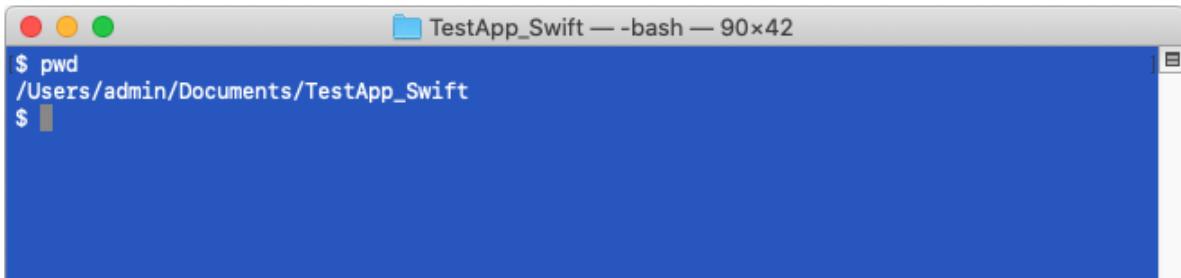


Export 위치를 지정하는 창이 표시되면 저장 위치를 선택하고 “Export”를 클릭합니다. 여기서는 ~/Downloads/TestApp\_Swift\_Upload 라는 이름의 폴더를 사용했습니다.

“Export”를 클릭하면 해당 폴더에 IPA 파일이 생성됩니다. 아래와 같이 Finder에서 확인할 수 있습니다. 이제 이 폴더의 위치를 잘 기억하거나 또는 Finder에서 해당 폴더로 이동한 후 창을 열어 놓습니다.



이제 Export된 IPA 파일을 가지고 작업을 진행해야 합니다. 터미널 앱을 실행하고 Xcode 프로젝트 폴더로 이동합니다.



```
$ pwd
/Users/admin/Documents/TestApp_Swift
```

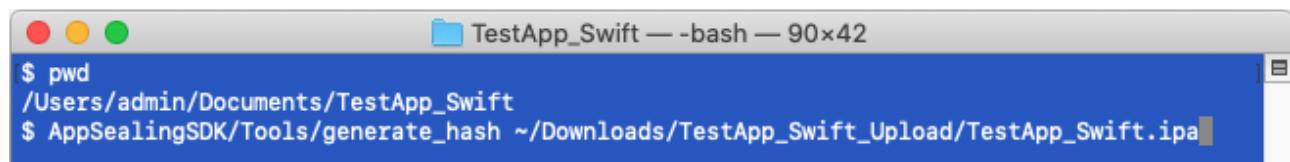
이 문서에서는 Xcode 프로젝트 경로로 ~/Documents/TestApp\_Swift 를 사용했습니다. 위 그림처럼 해당 위치로 이동해서 pwd를 실행하면 경로가 출력되는 것을 확인할 수 있습니다.

generate\_hash 파일에 실행 권한을 부여합니다.

```
$ chmod +x AppSealingSDK/Tools/generate_hash
```

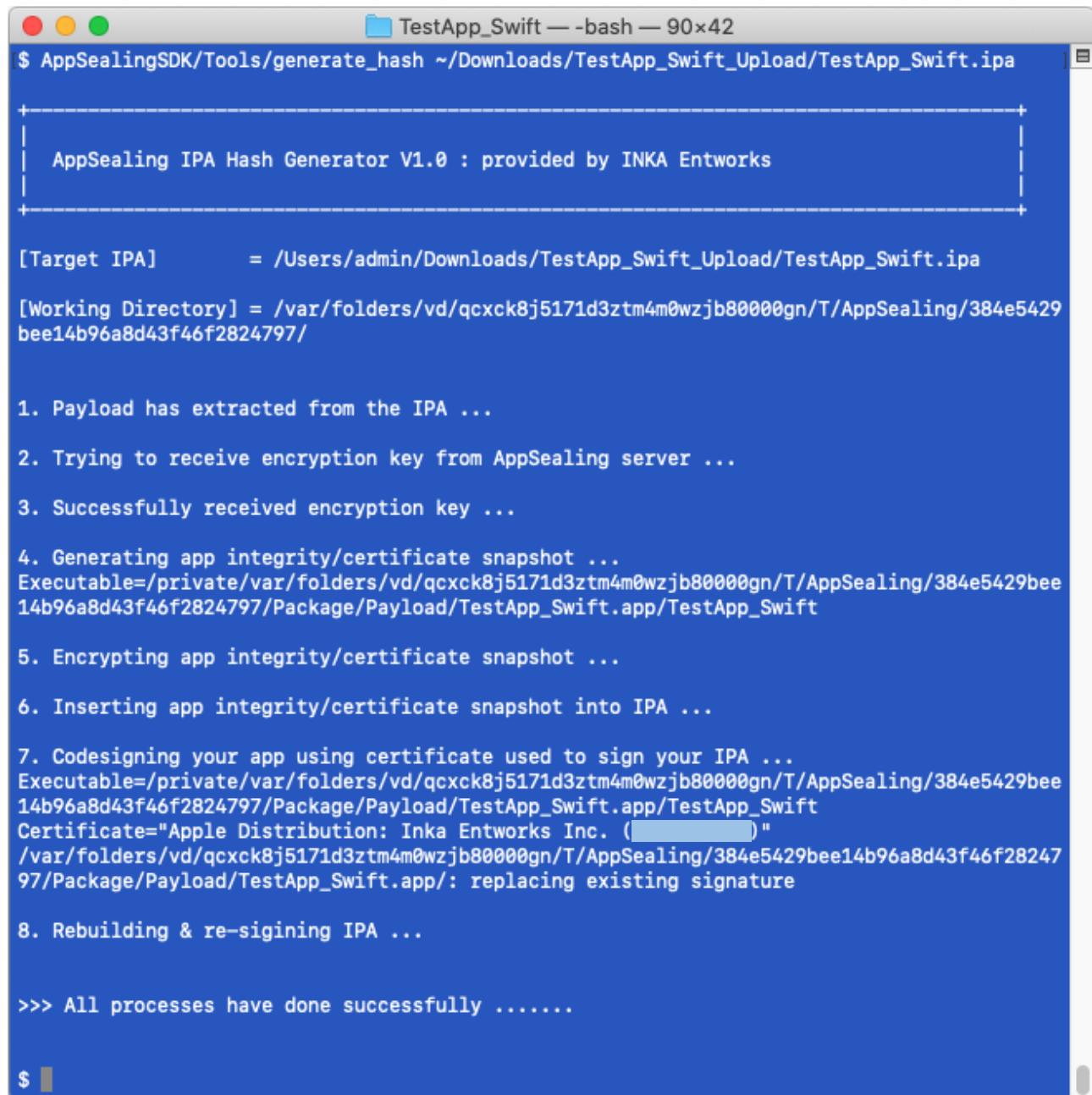
이제 아래와 같이 generate\_hash 스크립트를 실행합니다. 파라미터는 이전 단계에서 Export 했던 IPA 파일의 경로입니다. 경로를 직접 타이핑 하지 않고 열어 두었던 Finder 창에서 TestApp\_Swift.ipa 파일을 터미널 창으로 드래그&드롭 해도 됩니다.

```
$ AppSealingSDK/Tools/generate_hash ~/Downloads/TestApp_Swift_Upload/TestApp_Swift.ipa
```



```
$ pwd
/Users/admin/Documents/TestApp_Swift
$ AppSealingSDK/Tools/generate_hash ~/Downloads/TestApp_Swift_Upload/TestApp_Swift.ipa
```

스크립트를 실행하면 아래 화면과 같이 진행 상황이 표시되면서 지정한 IPA에 무결성 검증 정보와 인증서 검증 정보를 추가하게 됩니다.



```
TestApp_Swift — -bash — 90x42
$ AppSealingSDK/Tools/generate_hash ~/Downloads/TestApp_Swift_Upload/TestApp_Swift.ipa

AppSealing IPA Hash Generator V1.0 : provided by INKA Entworks

[Target IPA]      = /Users/admin/Downloads/TestApp_Swift_Upload/TestApp_Swift.ipa
[Working Directory] = /var/folders/vd/qcxck8j5171d3ztm4m0wzjb80000gn/T/AppSealing/384e5429bee14b96a8d43f46f2824797/

1. Payload has extracted from the IPA ...
2. Trying to receive encryption key from AppSealing server ...
3. Successfully received encryption key ...
4. Generating app integrity/certificate snapshot ...
Executable=/private/var/folders/vd/qcxck8j5171d3ztm4m0wzjb80000gn/T/AppSealing/384e5429bee14b96a8d43f46f2824797/Package/Payload/TestApp_Swift.app/TestApp_Swift
5. Encrypting app integrity/certificate snapshot ...
6. Inserting app integrity/certificate snapshot into IPA ...
7. Codesigning your app using certificate used to sign your IPA ...
Executable=/private/var/folders/vd/qcxck8j5171d3ztm4m0wzjb80000gn/T/AppSealing/384e5429bee14b96a8d43f46f2824797/Package/Payload/TestApp_Swift.app/TestApp_Swift
Certificate="Apple Distribution: Inka Entworks Inc. ([REDACTED])"
/var/folders/vd/qcxck8j5171d3ztm4m0wzjb80000gn/T/AppSealing/384e5429bee14b96a8d43f46f2824797/Package/Payload/TestApp_Swift.app/: replacing existing signature
8. Rebuilding & re-signing IPA ...

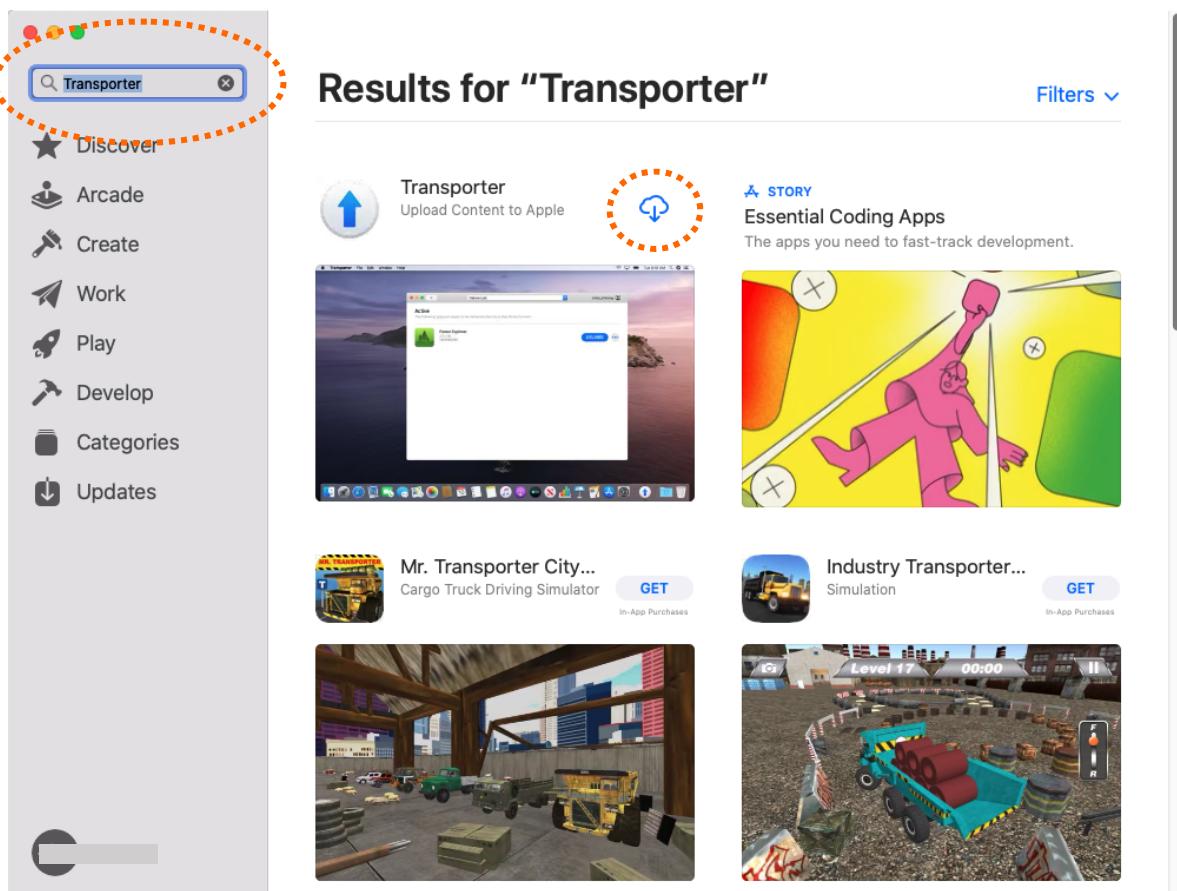
>>> All processes have done successfully ......

$
```

이 과정은 "Ad Hoc", "Enterprise", "Development"로 배포할 경우에도 동일하게 적용됩니다.

### 3-6 재서명 앱 App Store Connect 업로드

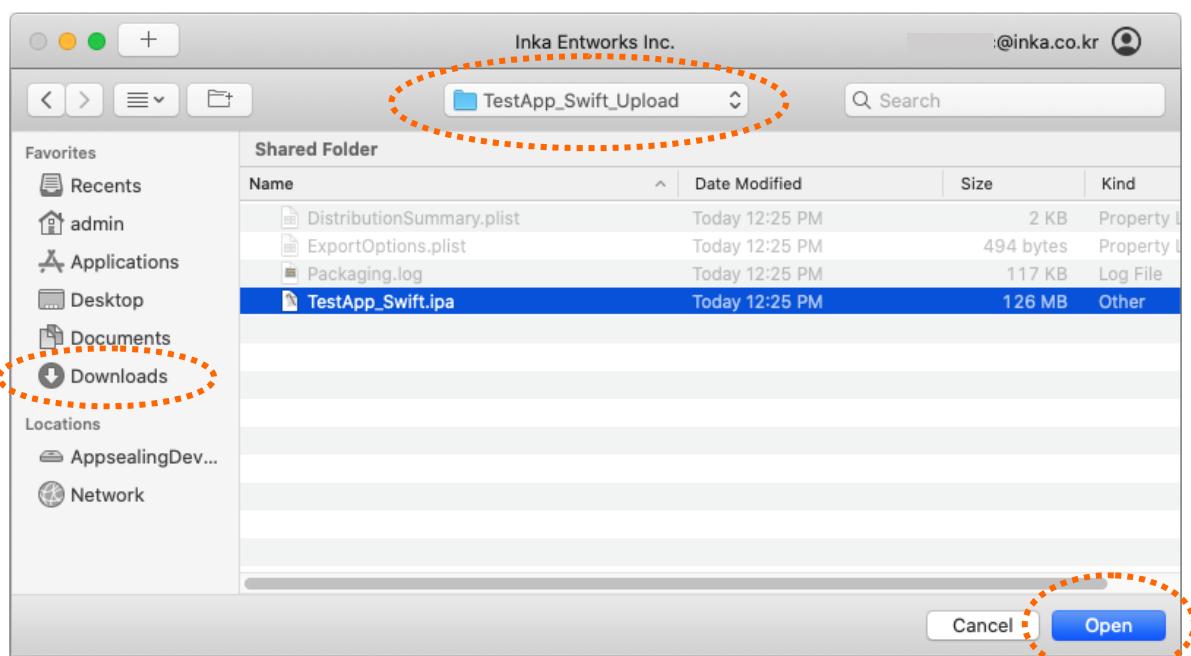
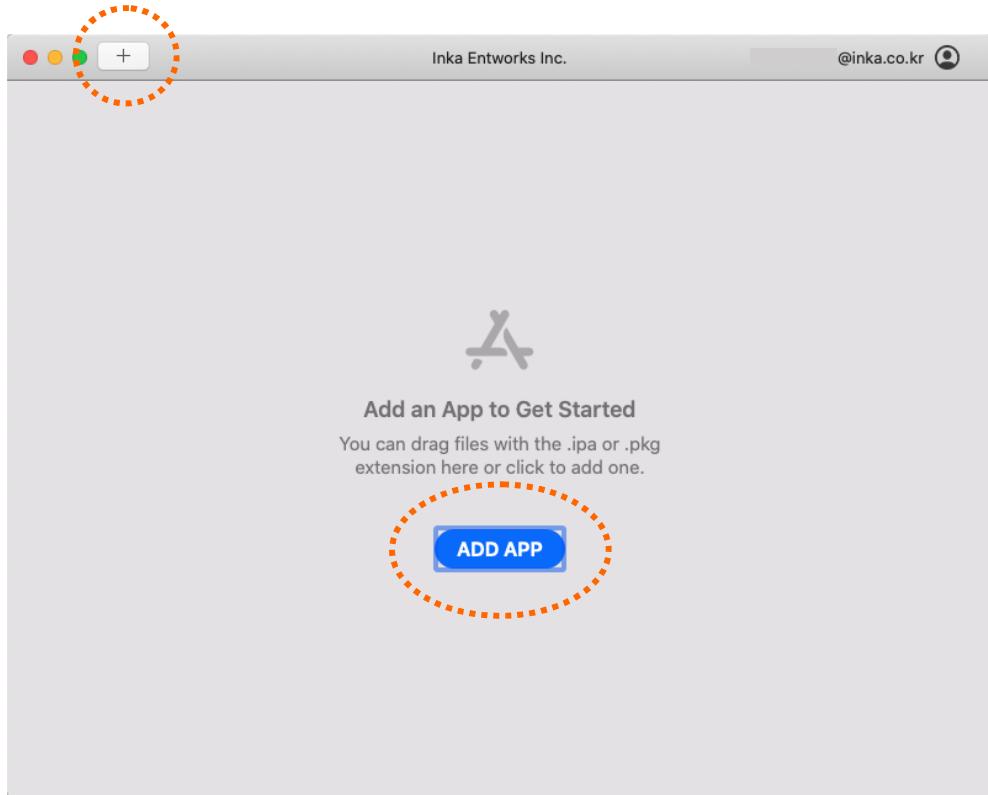
이제 재서명이 완료된 IPA를 App Store Connect에 업로드 합니다. 간편한 업로드를 위해 Transporter 앱을 사용하겠습니다. 맥 OS에 Transporter가 설치되어 있지 않은 경우 맥 AppStore를 실행하고 “Transporter”를 검색하여 설치합니다.



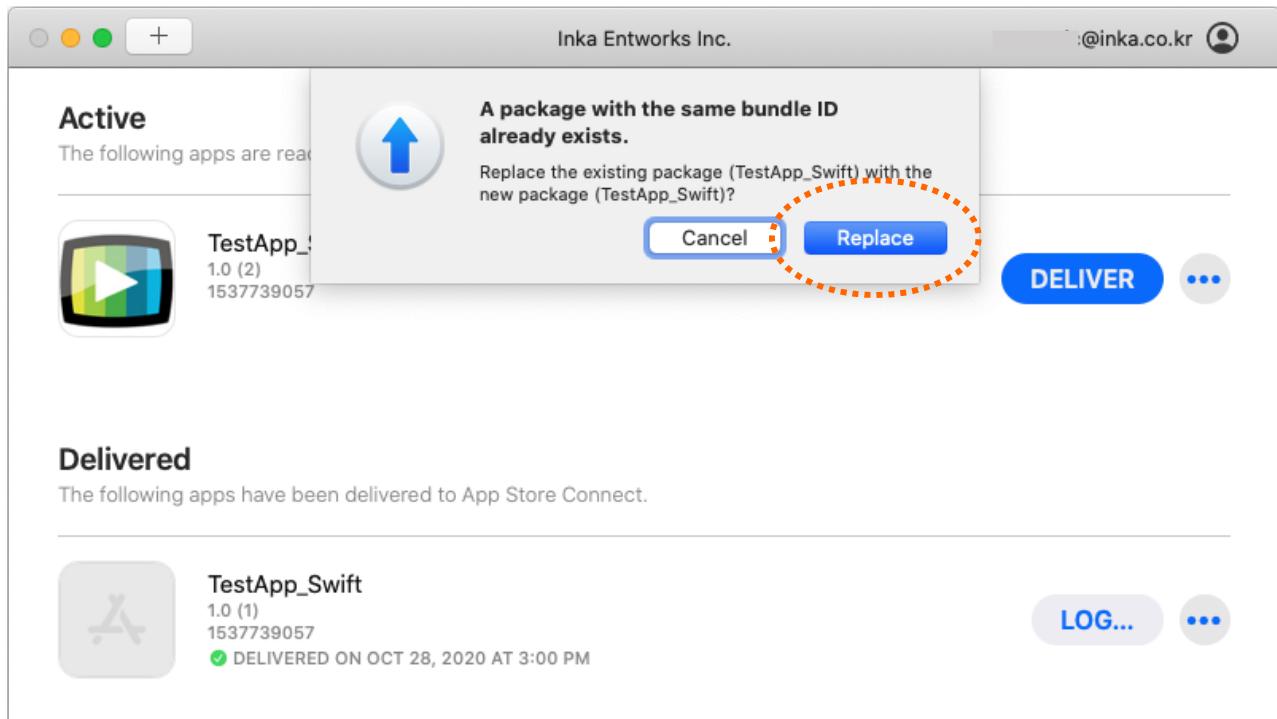
설치를 완료하고 실행하면 아래와 같이 Apple ID를 입력하는 창이 나타납니다. Apple 계정 ID와 비밀번호를 입력합니다. (이 과정은 최초 1회만 수행하면 됩니다)



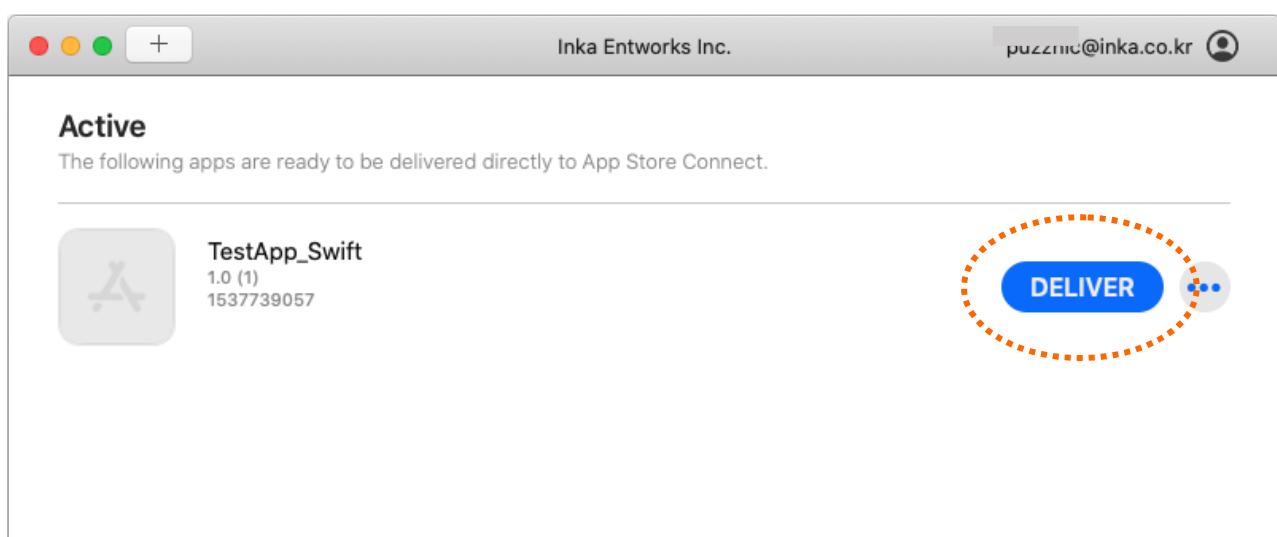
ID와 비밀번호를 입력하고 로그인 하면 아래와 같은 창이 나타나며 여기서 업로드 할 IPA를 선택하기 위해 좌측 상단의 + 버튼 또는 창 중앙의 "ADD APP" 버튼을 클릭하고 이전 단계에서 재설명이 완료된 IPA 파일을 선택합니다.

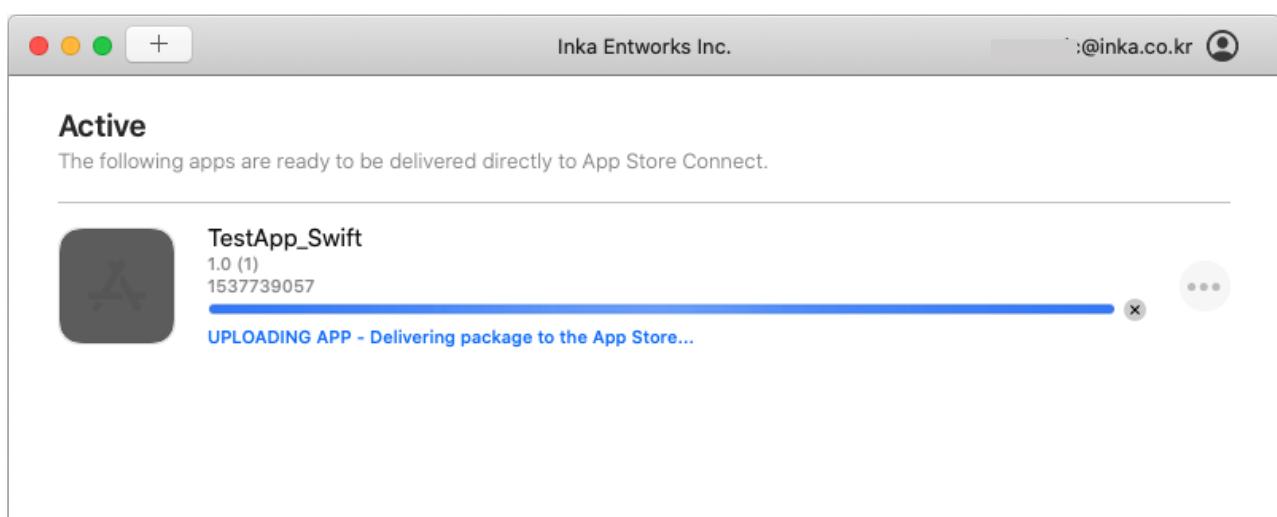
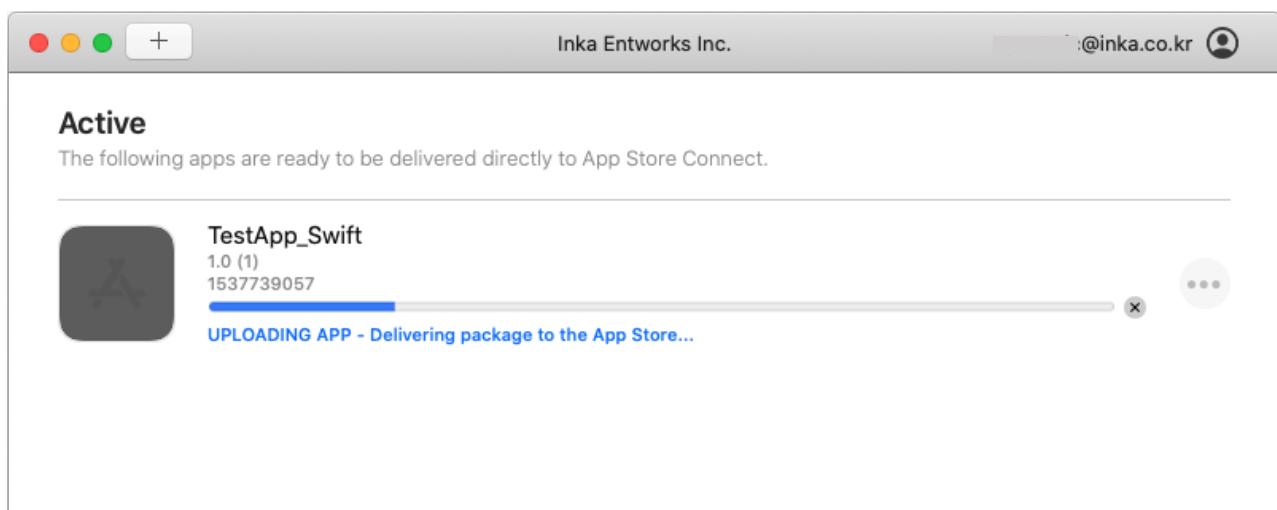
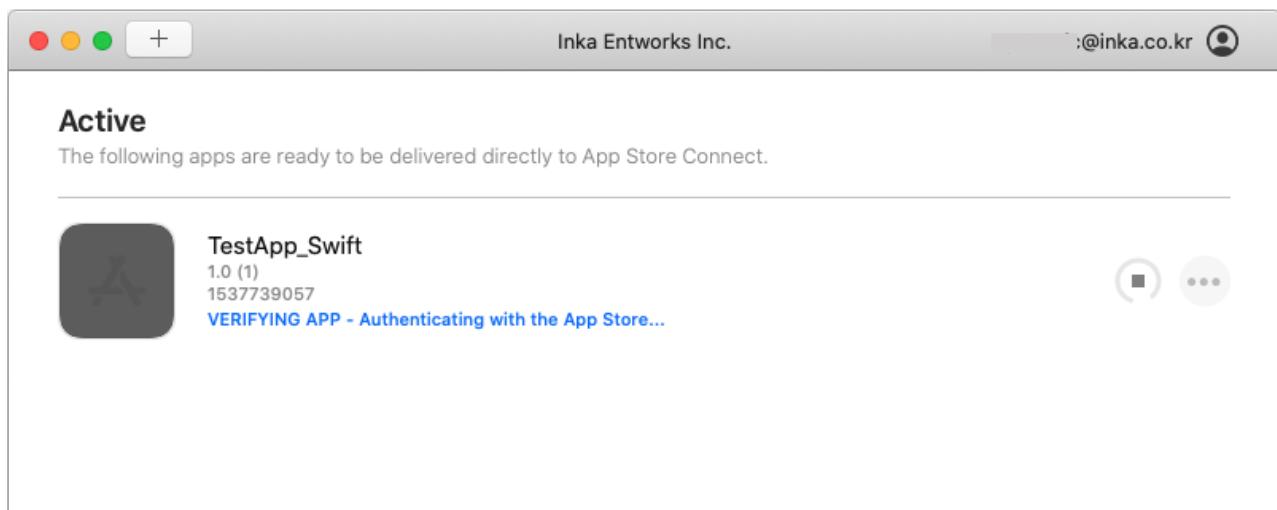


앱 최초 전송 이후 앱의 버전이나 빌드 번호를 증가시키고 새로 업로드 하려고 할 경우 IPA 파일을 추가하면 번들 ID가 동일하기 때문에 아래와 같은 확인 창이 나타날 수 있습니다. 이 경우 “Replace” 버튼을 클릭합니다.

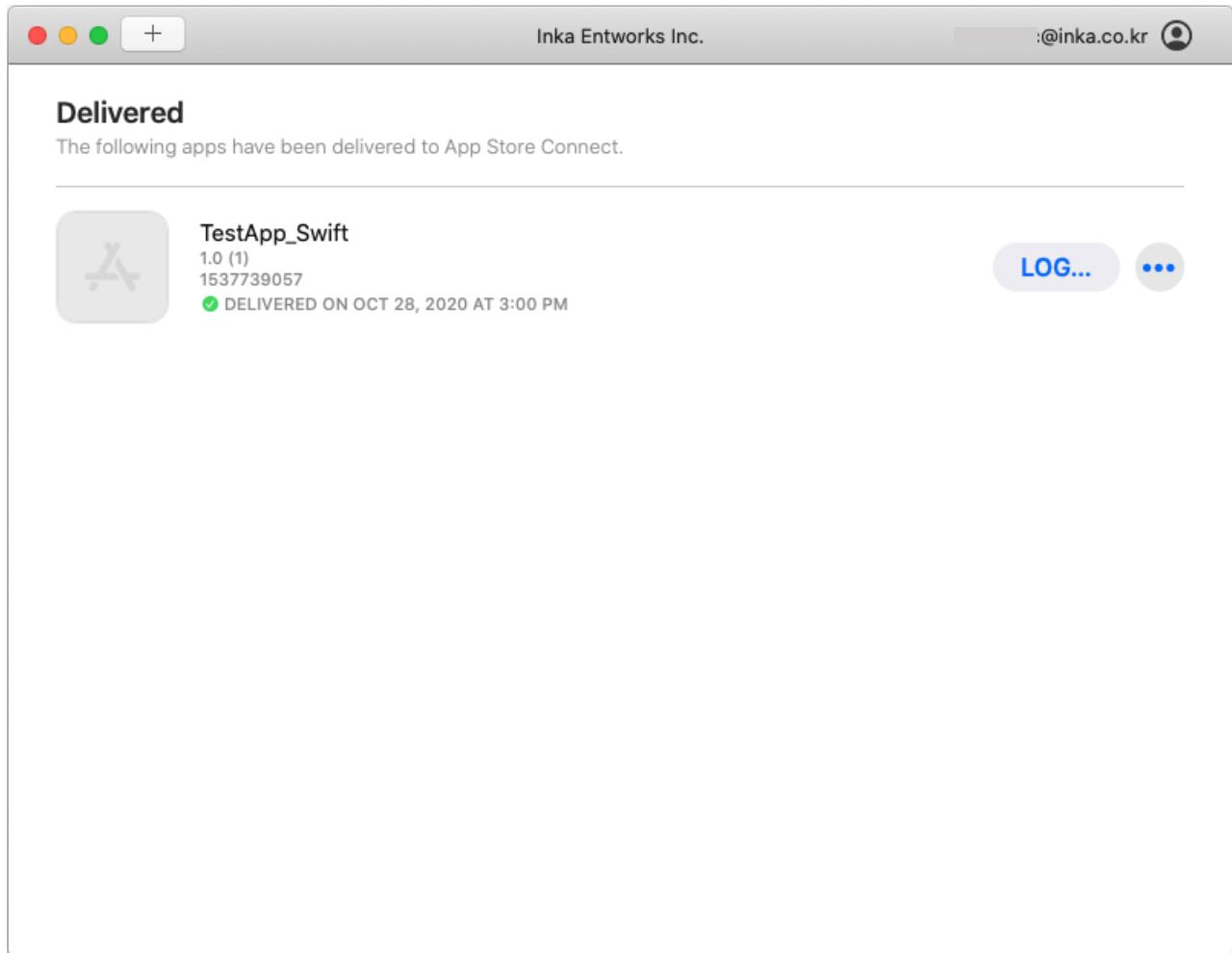


IPA가 추가된 상태에서 “DELIVER” 버튼을 클릭하면 앱 검증과 App Store Connect 업로드 과정이 진행됩니다.





아래 화면과 같이 전송이 완료되면 App Store Connect에서 심사를 위한 빌드 제출이나 TestFlight용 빌드 배포를 진행할 수 있습니다.



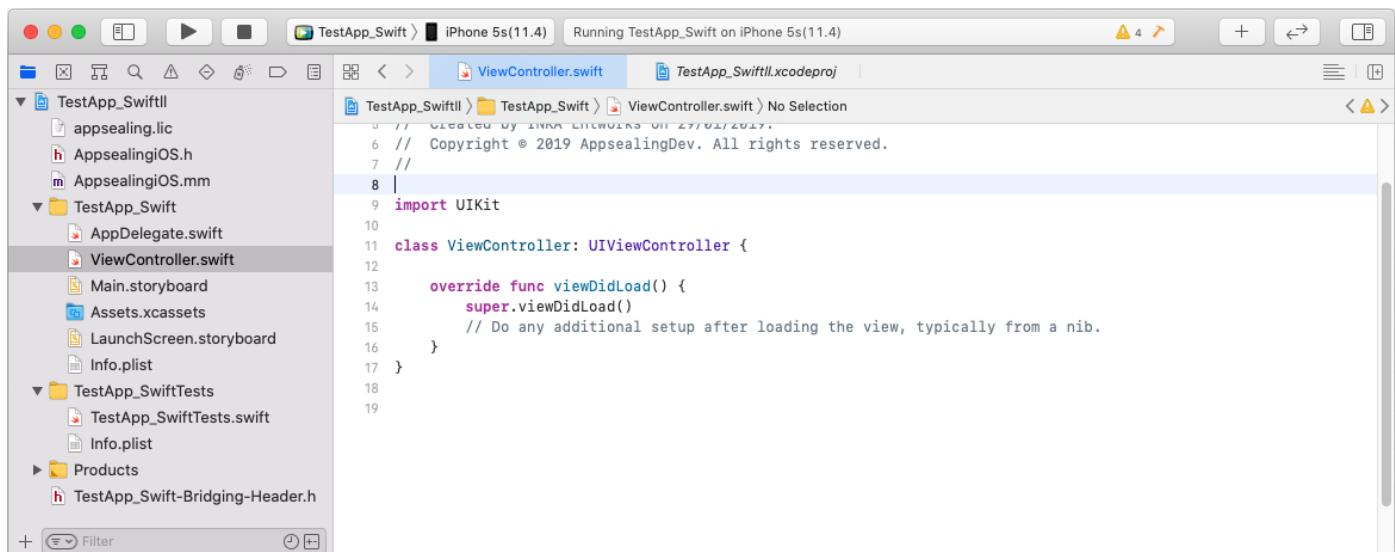
## Part 4. 보안 위협 안내를 위한 GUI

AppSealing 라이브러리는 앱이 실행될 때 자동으로 활성화됩니다. AppSealing 라이브러리가 비정상적인 환경 (탈옥된 장치, 실행 파일의 복호화 된 상태, 디버거 연결)을 감지한 경우 사용자 조작과 관계없이 20초 후에 앱이 닫히도록 설계되어 있습니다. 앱 탐지 결과를 사용자에게 통지하고 사용자가 일부 잘못된 환경에 있음을 인식할 수 있도록 적절한 메시지 상자를 표시하십시오.

앱에서 해당 대화 상자를 표시하려면 아래 과정을 참고하여 "ViewController.swift" 파일에 코드를 추가하십시오. (프로젝트가 Objective-C 기반인 경우 ViewController.mm 수정)

### 4-1 앱에서 UIAlertController 출력

먼저 Xcode project를 열고 왼쪽 프로젝트 패널에서 ViewController.swift" 파일을 선택합니다.



ViewController.swift 파일을 연 후 해당 파일에 다음 코드를 삽입하십시오. 만약 ViewController.swift 파일에 'viewDidAppear' 메서드가 이미 포함되어 있다면 'super.viewDidAppear( animated );' 줄 바로 아래 부분의 코드를 삽입하십시오.

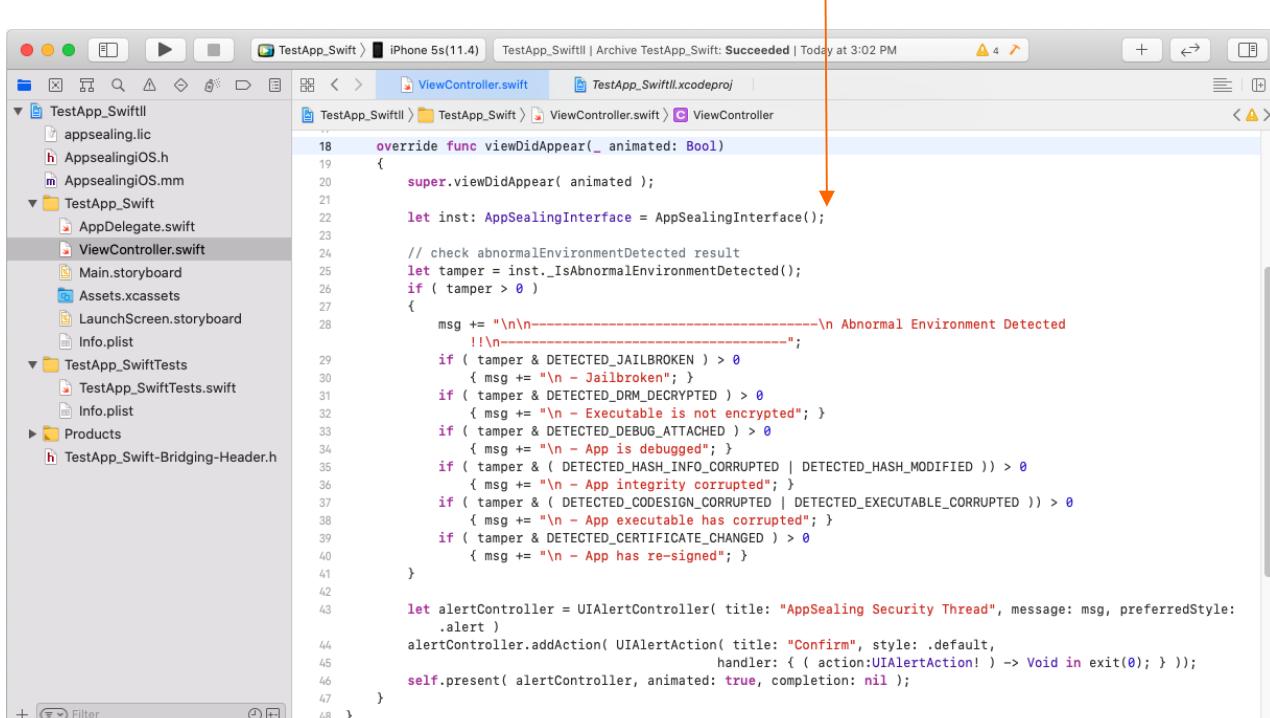
## Simple UI code into 'ViewController.swift' for Swift project

```

override func viewDidAppear(_ animated: Bool)
{
    super.viewDidAppear( animated );

    let inst: AppSealingInterface = AppSealingInterface();
    let tamper: Int32 = inst._IsAbnormalEnvironmentDetected();
    if ( tamper > 0 )
    {
        var msg = "Abnormal Environment Detected !!";
        if ( tamper & DETECTED_JAILBROKEN ) > 0
            { msg += "\n - Jailbroken"; }
        if ( tamper & DETECTED_DRM_DECRYPTED ) > 0
            { msg += "\n - Executable is not encrypted"; }
        if ( tamper & DETECTED_DEBUG_ATTACHED ) > 0
            { msg += "\n - App is debugged"; }
        if ( tamper & ( DETECTED_HASH_INFO_CORRUPTED | DETECTED_HASH_MODIFIED ) ) > 0
            { msg += "\n - App integrity corrupted"; }
        if ( tamper & ( DETECTED_CODESIGN_CORRUPTED | DETECTED_EXECUTABLE_CORRUPTED ) ) > 0
            { msg += "\n - App executable has corrupted"; }
        if ( tamper & DETECTED_CERTIFICATE_CHANGED ) > 0
            { msg += "\n - App has re-signed"; }
        let alertController = UIAlertController(title: "AppSealing",
                                              message: msg, preferredStyle: .alert );
        alertController.addAction(UIAlertAction(title: "Confirm", style: .default,
                                              handler: { (action:UIAlertAction!) -> Void in exit(0); } ));
        self.present(alertController, animated: true, completion: nil);
    }
}

```



위의 샘플 UI 코드는 "AppsealingiOS.mm" 파일에도 포함되어 있습니다. 코드를 복사하여 붙여 넣을 수 있습니다.

프로젝트가 Objective-C 기반인 경우 다음 코드를 사용하여 UI를 표시할 수 있습니다.

#### Simple UI code into 'ViewController.mm' for Objective-C project

```
#include "AppsealingiOS.h"

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];

    int tamper = ObjC_IsAbnormalEnvironmentDetected();
    if ( tamper > 0 )
    {
        NSString* msg = @"Abnormal Environment Detected !!";
        if (( tamper & DETECTED_JAILBROKEN ) > 0 )
            msg = [msg stringByAppendingString:@"\n - Jailbroken"];
        if (( tamper & DETECTED_DRM_DECRYPTED ) > 0 )
            msg = [msg stringByAppendingString:@"\n - Executable is not encrypted"];
        if (( tamper & DETECTED_DEBUG_ATTACHED ) > 0 )
            msg = [msg stringByAppendingString:@"\n - App is debugged"];
        if ( tamper & ( DETECTED_HASH_INFO_CORRUPTED | DETECTED_HASH_MODIFIED ) ) > 0
            msg = [msg stringByAppendingString:@"\n - App integrity corrupted"];
        if ( tamper & ( DETECTED_CODESIGN_CORRUPTED | DETECTED_EXECUTABLE_CORRUPTED ) ) > 0
            msg = [msg stringByAppendingString:@"\n - App executable has corrupted"];
        if ( tamper & DETECTED_CERTIFICATE_CHANGED ) > 0
            msg = [msg stringByAppendingString:@"\n - App has re-signed"];

        UIAlertController *alert = [UIAlertController alertControllerWithTitle:@"AppSealing"
                                                               message:msg
                                                               preferredStyle:UIAlertControllerStyleAlert];
        UIAlertAction *confirm = [UIAlertAction actionWithTitle:@"Confirm"
                                                       style:UIAlertActionStyleDefault
                                                       handler:^(UIAlertAction * _Nonnull action) { exit(0); }];
        [alert addAction:confirm];
        [self presentViewController:alert animated:YES completion:nil];
    }
}
```

Xcode나 gdb를 사용하여 앱을 디버깅하거나 비정상적인 장치에서 앱을 실행하면 앱에서 아래와 같은 간단한 경고 상자가 표시됩니다. 이런 상황에서 사용자 조치와 상관없이 앱은 20초 후에 자동으로 종료됩니다.

iOS 기기에서 보안 침입이 감지되면 경고창이 출력되고 20초 후 또는 사용자가 "Confirm" 버튼을 누르면 앱이 닫힙니다.



## Part 5. AppSealing 기기 고유 ID 획득

AppSealing SDK는 기기 별로 고유 ID를 생성하고 관리합니다. AppSealing SDK를 사용하는 고객은 필요한 경우 AppSealing에서 제공하는 인터페이스를 이용하여 기기 고유 ID를 확인할 수 있습니다. 확인된 고유 ID는 AppSealing에서 제공하는 해킹 데이터 서비스와 연계하여 비즈니스에 활용할 수 있습니다.

### 5-1 기기 고유 ID 확인

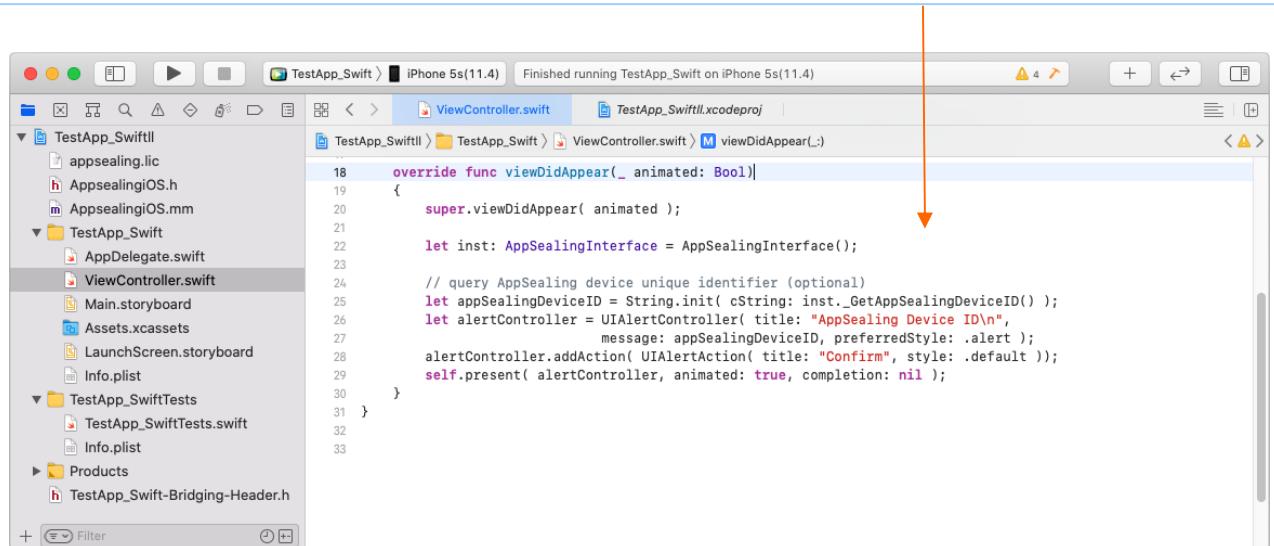
프로젝트의 "ViewController.swift" 파일을 열고 다음 코드를 삽입하십시오. 만약 ViewController.swift 파일에 'viewDidAppear' 메서드가 이미 있다면 'super.viewDidAppear( animated );' 줄 아래에 다음 코드를 삽입하십시오.)

#### Simple UI code into 'ViewController.swift' for Swift project

```
override func viewDidAppear(_ animated: Bool)
{
    super.viewDidAppear( animated );

    let inst: AppSealingInterface = AppSealingInterface();
    let appSealingDeviceID = String.init( cString: inst._GetAppSealingDeviceID() );
    let alertController = UIAlertController( title: "AppSealing DeviceID",
                                            message: appSealingDeviceID, preferredStyle: .alert );

    alertController.addAction( UIAlertAction( title: "Confirm", style: .default ) );
    self.present( alertController, animated: true, completion: nil );
}
```



샘플 UI 코드는 “AppSealingiOS.mm” 파일에도 포함되어 있습니다. 코드를 복사하여 붙여 넣을 수 있습니다.

프로젝트가 Objective-C 기반인 경우 다음 코드를 사용하여 UI를 표시할 수 있습니다.

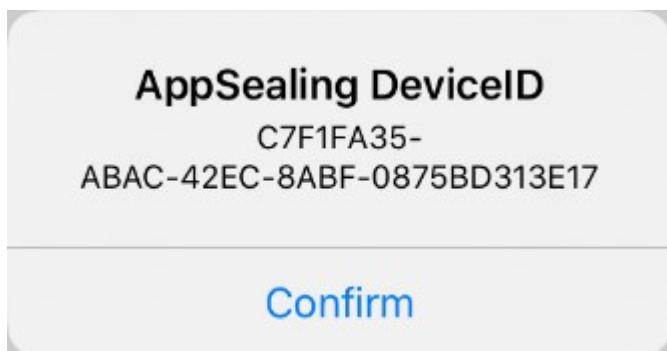
Simple UI code into ‘ViewController.mm’ for **Objective-C** project

```
#include "AppSealingiOS.h"

char _appSealingDeviceID[64];
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];

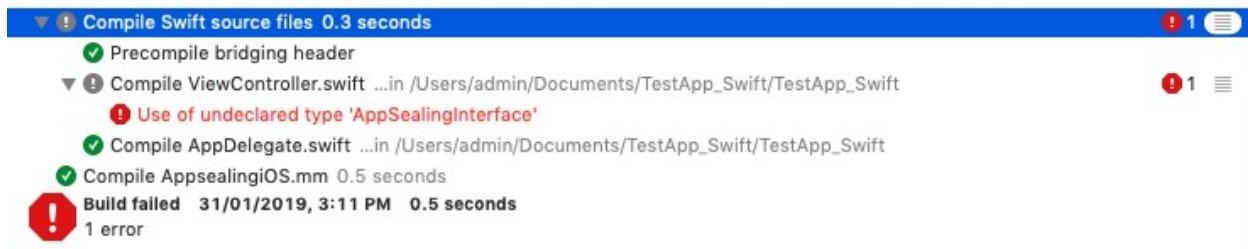
    if ( ObjC_GetAppSealingDeviceID( _appSealingDeviceID ) == 0 )
    {
        UIAlertController *alert = [UIAlertController alertControllerWithTitle:@"AppSealing DeviceID"
                                                               message:[NSString alloc] initWithUTF8String:_appSealingDeviceID]
                                                               preferredStyle:UIAlertControllerStyleAlert];
        UIAlertAction *confirm = [UIAlertAction actionWithTitle:@"Confirm"
                                                       style:UIAlertActionStyleDefault
                                                       handler:^(UIAlertAction * _Nonnull action) { }];
        [alert addAction:confirm];
        [self presentViewController:alert animated:YES completion:nil];
    }
}
```

앱을 실행하면 아래와 같이 메시지 박스에서 기기 고유 ID를 확인할 수 있습니다.

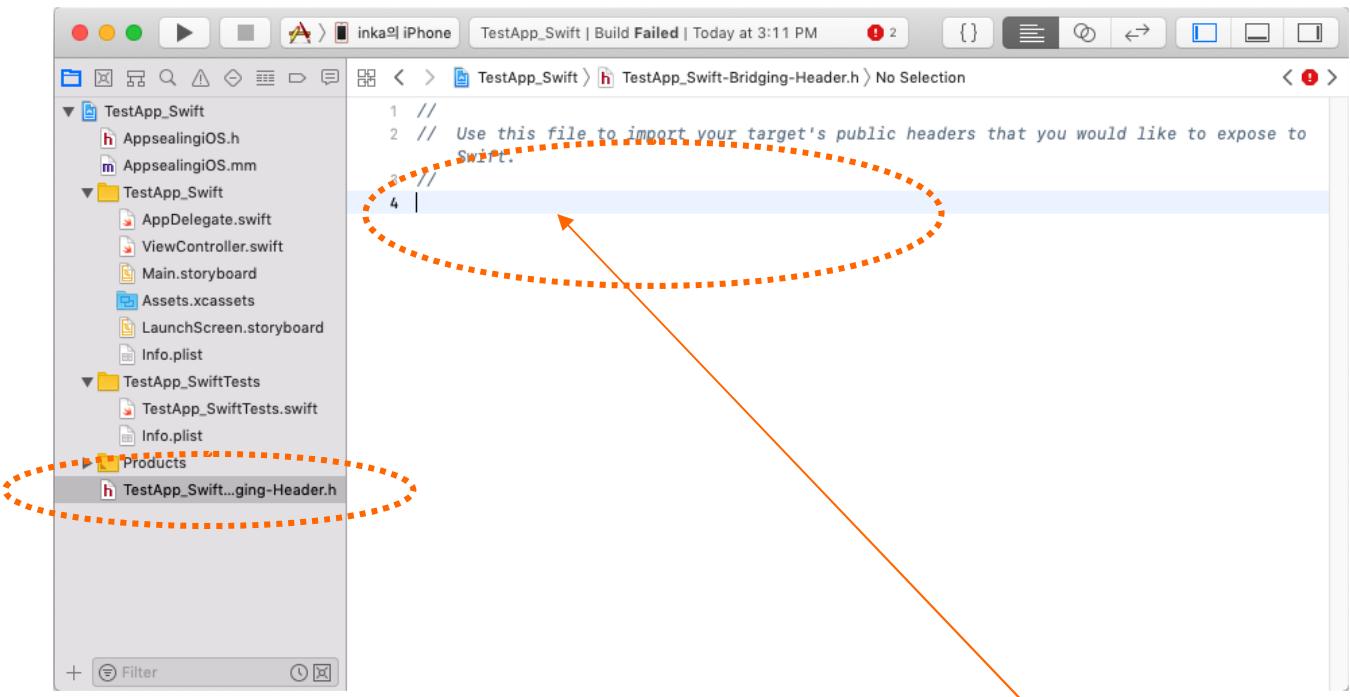


## Part 6. 문제 해결

### 6-1 Xcode Build error (1) Use of undeclared type 'AppSealingInterface'



Bridging header 파일에 '#import "AppsealingiOS.h"' 가 생략되면 위와 같은 오류가 발생합니다.



"TestApp\_Swift-Bridging-Header.h"를 선택하고 '#import "AppsealingiOS.h"'를 추가하십시오. "AppSealingiOS.h" 파일이 지정된 경로에 해당 파일이 존재하는지 확인하십시오. (Xcode project/AppSealingSDK/Libraries/AppsealingiOS.h)

또한 "AppSealingiOS.h" 파일은 "AppSealingiOS.mm" 파일과 함께 프로젝트에 포함되어 있어야 합니다.

## 6-2 Xcode Build error (2) Undefined symbols for architecture arm64: "\_OBJC\_CLASS\_\$\_AppSealingInterface"

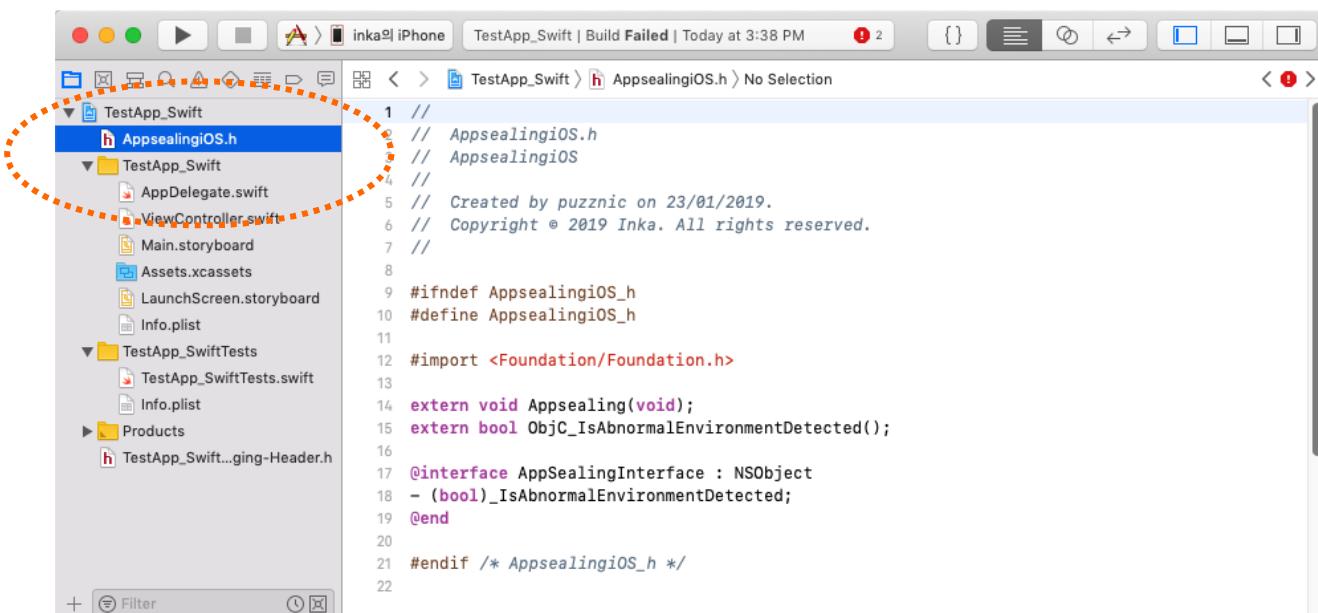
```

Ld /Users/admin/Desktop/Build/Products/Debug-iphoneos/TestApp_Swift.app/TestApp_Swift 0.1 seconds
(in target: TestApp_Swift)
cd /Users/admin/Documents/TestApp_Swift
export IPHONEOS_DEPLOYMENT_TARGET=12.1
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang -
arch arm64 -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/
Developer/SDKs/iPhoneOS12.1.sdk -L/Users/admin/Desktop/Build/Products/Debug-iphoneos -F/Users/
admin/Desktop/Build/Products/Debug-iphoneos -filelist /Users/admin/Desktop/Build/
Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/Objects-normal/arm64-
TestApp_Swift.LinkFileList -Xlinker -rpath -Xlinker @executable_path/Frameworks -miphoneos-
version-min=12.1 -dead_strip -Xlinker -object_path_lto -Xlinker /Users/admin/Desktop/Build/
Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/Objects-normal/arm64-
TestApp_Swift_lto.o -Xlinker -export_dynamic -Xlinker -no_deduplicate -fembed-bitcode-marker -
fobjc-link-runtime -L/Applications/Xcode.app/Contents/Developer/Toolchains/
XcodeDefault.xctoolchain/usr/lib/swift/iphoneos -Xlinker -add_ast_path -Xlinker /Users/admin/
Desktop/Build/Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/
Objects-normal/arm64/TestApp_Swift.swiftmodule -lStaticAppSec_Debug -L/Users/admin/Documents/
TestApp_Swift/AppSealingSDK/Libraries -Xlinker -dependency_info -Xlinker /Users/admin/Desktop/
Build/Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/Objects-normal/
arm64/TestApp_Swift_dependency_info.dat -o /Users/admin/Desktop/Build/Products/Debug-iphoneos/
TestApp_Swift.app/TestApp_Swift

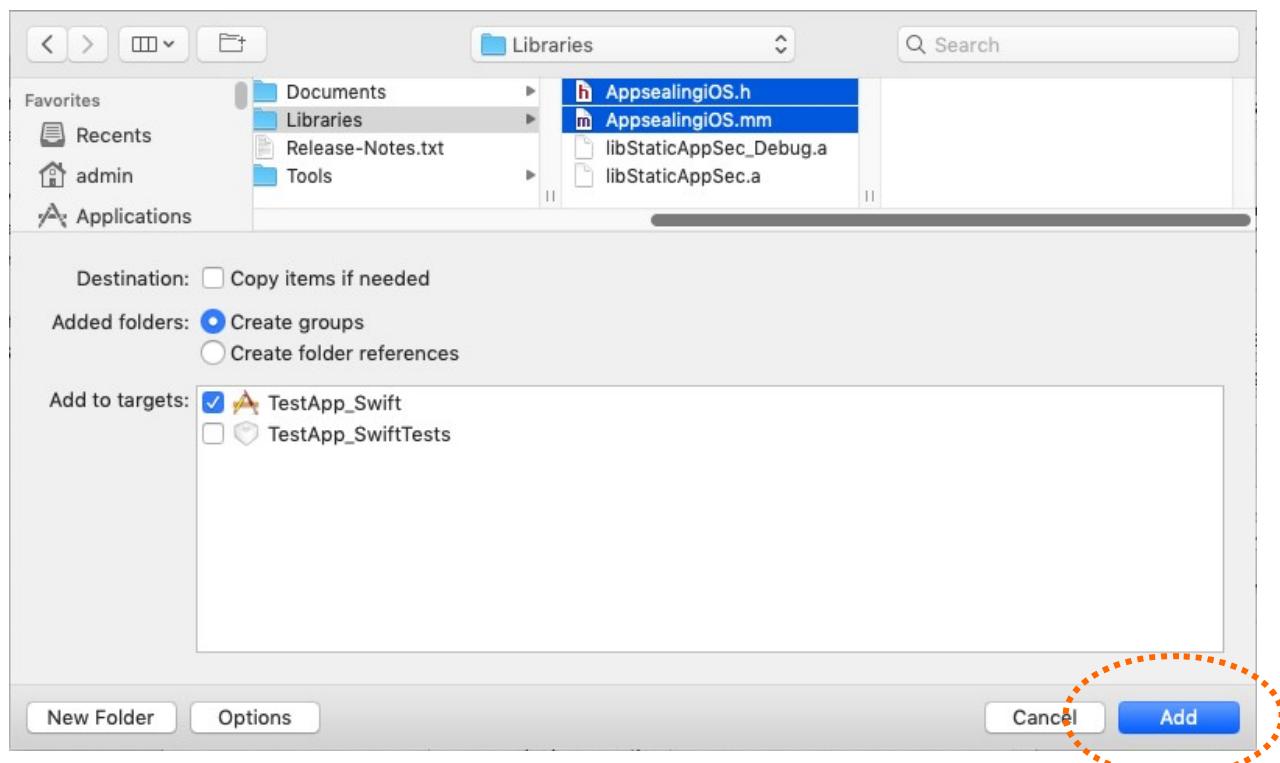
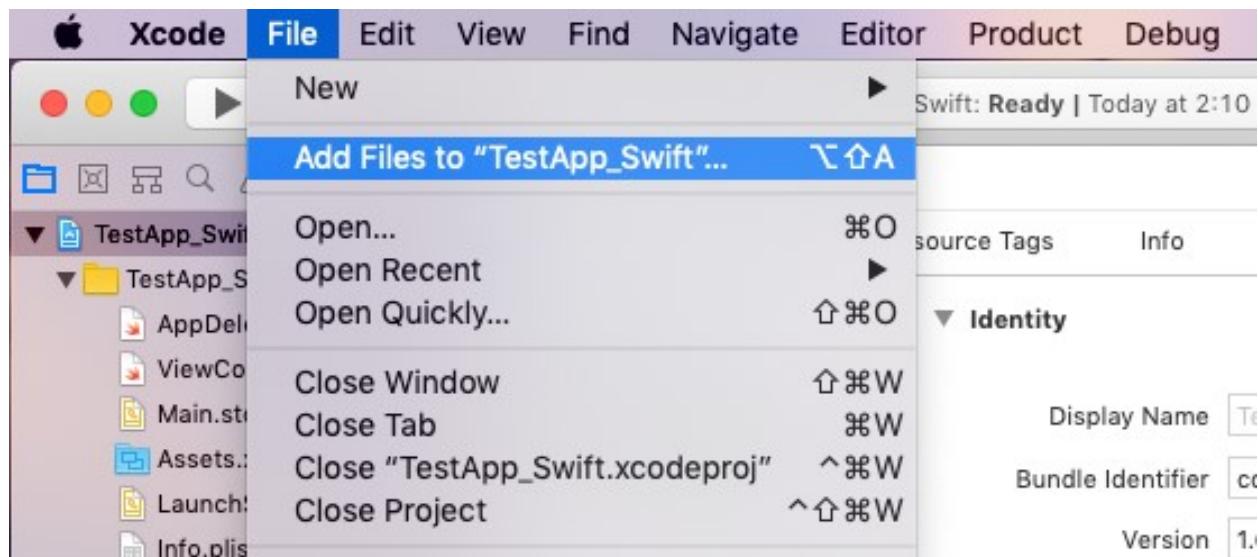
Undefined symbols for architecture arm64:
  "_OBJC_CLASS_$_AppSealingInterface", referenced from:
    objc-class-ref in ViewController.o
ld: symbol(s) not found for architecture arm64
clang: error: linker command failed with exit code 1 (use -v to see invocation)

symbol(s) not found for architecture arm64
  linker command failed with exit code 1 (use -v to see invocation)
✓ Copy TestApp_Swift.swiftmodule 0.1 seconds
✓ Copy TestApp_Swift.swiftdoc 0.1 seconds
Build failed 31/01/2019, 3:38 PM 0.9 seconds
1 error
!
```

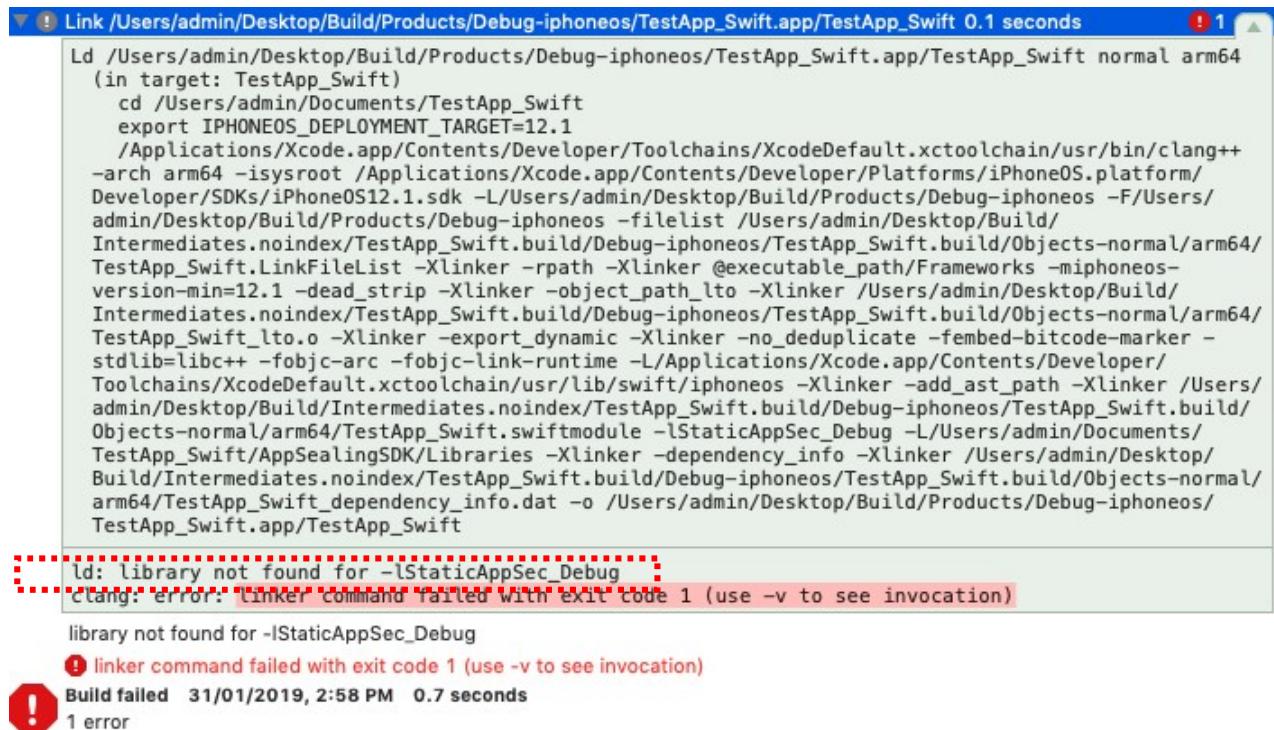
이 링커 오류는 "AppSealingiOS.mm"이 프로젝트에 포함되어 있지 않을 때 발생합니다. 프로젝트 트리에서 해당 파일이 추가되어 있는지 확인하십시오.



"AppSealingiOS.mm" 파일이 프로젝트에 포함되어 있지 않은 경우 "File > Add Files to "TestApp\_Swift" ..." 메뉴를 실행하고 AppSealing SDK 경로에서 파일을 선택하여 프로젝트에 포함 시키십시오.



### 6-3 Xcode Build error (3) **ld: library not found for -lStaticAppSec\_Debug**



```
Ld /Users/admin/Desktop/Build/Products/Debug-iphoneos/TestApp_Swift.app/TestApp_Swift normal arm64
(in target: TestApp_Swift)
cd /Users/admin/Documents/TestApp_Swift
export IPHONEOS_DEPLOYMENT_TARGET=12.1
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang++
-arch arm64 -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/
Developer/SDKs/iPhoneOS12.1.sdk -L/Users/admin/Desktop/Build/Products/Debug-iphoneos -F/Users/
admin/Desktop/Build/Products/Debug-iphoneos -filelist /Users/admin/Desktop/Build/
Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/Objects-normal/arm64-
TestApp_Swift.LinkFileList -Xlinker -rpath -Xlinker @executable_path/Frameworks -miphoneos-
version-min=12.1 -dead_strip -Xlinker -object_path_lto -Xlinker /Users/admin/Desktop/Build/
Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/Objects-normal/arm64-
TestApp_Swift_lto.o -Xlinker -export_dynamic -Xlinker -no_deduplicate -fembed-bitcode-marker -
stdlib=libc++ -fobjc-arc -fobjc-link-runtime -L/Applications/Xcode.app/Contents/Developer/
Toolchains/XcodeDefault.xctoolchain/usr/lib/swift/iphoneos -Xlinker -add_ast_path -Xlinker /Users/
admin/Desktop/Build/Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/
Objects-normal/arm64/TestApp_Swift.swiftmodule -lStaticAppSec_Debug -L/Users/admin/Documents/
TestApp_Swift/AppSealingSDK/Libraries -Xlinker -dependency_info -Xlinker /Users/admin/Desktop/
Build/Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/Objects-normal/
arm64/TestApp_Swift_dependency_info.dat -o /Users/admin/Desktop/Build/Products/Debug-iphoneos/
TestApp_Swift.app/TestApp_Swift

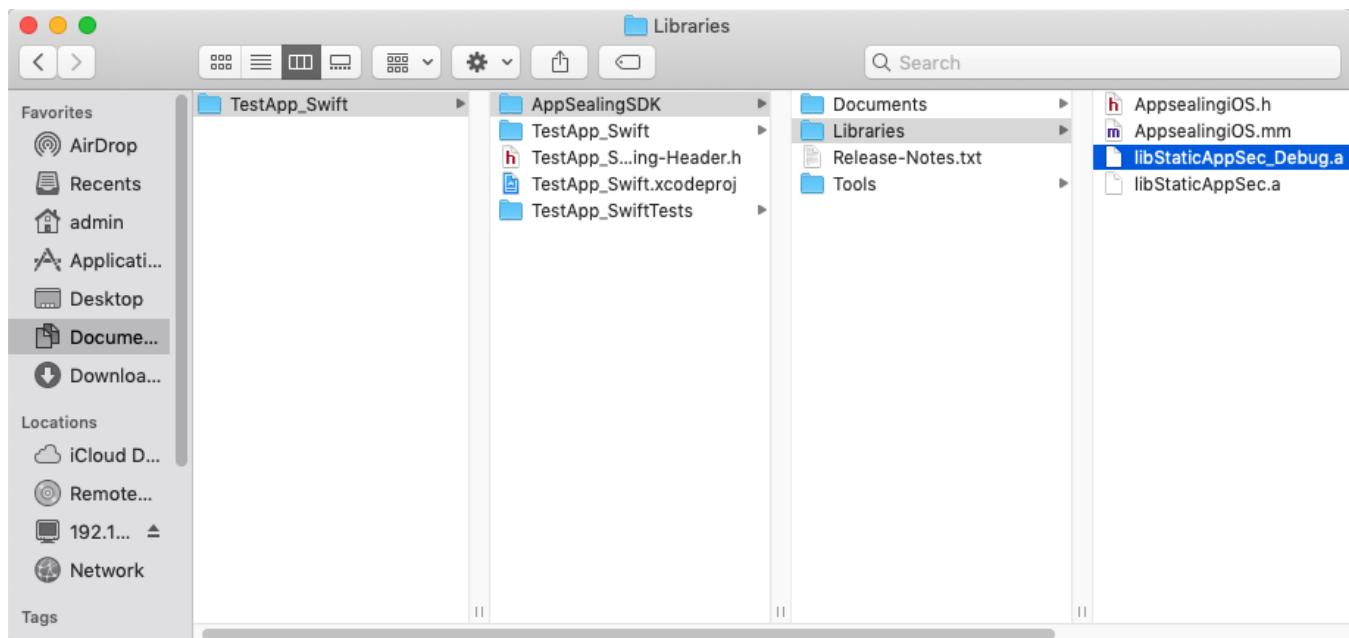
ld: library not found for -lStaticAppSec_Debug
clang: error: linker command failed with exit code 1 (use -v to see invocation)

library not found for -lStaticAppSec_Debug
① linker command failed with exit code 1 (use -v to see invocation)
Build failed 31/01/2019, 2:58 PM 0.7 seconds
1 error
```

The screenshot shows the Xcode build log window. A red box highlights the error message "ld: library not found for -lStaticAppSec\_Debug". Below the log, a red exclamation mark icon indicates a build failure. The text "Build failed 31/01/2019, 2:58 PM 0.7 seconds" and "1 error" are displayed.

이 링커 오류는 "libStaticAppSec\_Debug.a" 파일이 지정된 폴더에 없거나 손상된 경우에 발생합니다. "libStaticAppSec\_Debug.a" 파일은 아래 경로에 존재해야 합니다.

*"TestApp\_Swift (Project folder) > AppSealingSDK > Libraries > libStaticAppSec\_Debug.a"*

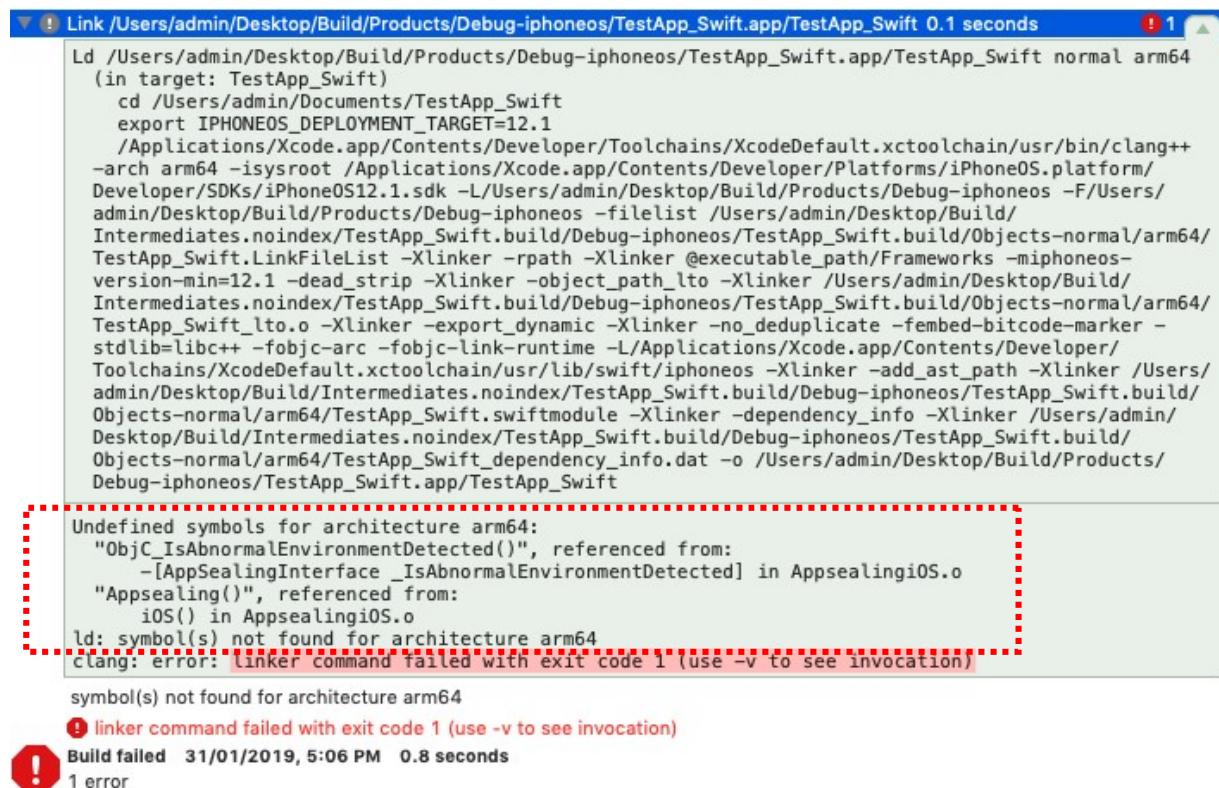


파일이 누락되었거나 손상되었을 경우 AppSealing SDK zip 파일을 다시 압축 해제하시거나 새로 다운로드하십시오.

이 링커 오류는 release 빌드 시 “lStaticAppSec” 옵션을 추가했을 때에도 발생할 수 있습니다. Release 빌드를 위해서 “libStaticAppSec.a” 파일이 적절한 위치에 존재해야 합니다.

*"TestApp\_Swift (Project folder) > AppSealingSDK > Libraries > libStaticAppSec.a"*

## 6-4 Xcode Build error (4) Undefined symbols for architecture arm64: "ObjC\_IsAbnormalEnvironmentDetected()", "Appsealing()"



```

Link /Users/admin/Desktop/Build/Products/Debug-iphoneos/TestApp_Swift.app/TestApp_Swift 0.1 seconds
! 1
Ld /Users/admin/Desktop/Build/Products/Debug-iphoneos/TestApp_Swift.app/TestApp_Swift normal arm64
(in target: TestApp_Swift)
cd /Users/admin/Documents/TestApp_Swift
export IPHONEOS_DEPLOYMENT_TARGET=12.1
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/clang++
-arch arm64 -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/
Developer/SDKs/iPhoneOS12.1.sdk -L/Users/admin/Desktop/Build/Products/Debug-iphoneos -F/Users/
admin/Desktop/Build/Products/Debug-iphoneos -filelist /Users/admin/Desktop/Build/
Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/Objects-normal/arm64/
TestApp_Swift.LinkFileList -Xlinker -rpath -Xlinker @executable_path/Frameworks -miphoneos-
version-min=12.1 -dead_strip -Xlinker -object_path_lto -Xlinker /Users/admin/Desktop/Build/
Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/Objects-normal/arm64/
TestApp_Swift_lto.o -Xlinker -export_dynamic -Xlinker -no_deduplicate -fembed-bitcode-marker -
stdlib=libc++ -fobjc-arc -fobjc-link-runtime -L/Applications/Xcode.app/Contents/Developer/
Toolchains/XcodeDefault.xctoolchain/usr/lib/swift/iphoneos -Xlinker -add_ast_path -Xlinker /Users/
admin/Desktop/Build/Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/
Objects-normal/arm64/TestApp_Swift.swiftmodule -Xlinker -dependency_info -Xlinker /Users/admin/
/Desktop/Build/Intermediates.noindex/TestApp_Swift.build/Debug-iphoneos/TestApp_Swift.build/
Objects-normal/arm64/TestApp_Swift_dependency_info.dat -o /Users/admin/Desktop/Build/Products/
Debug-iphoneos/TestApp_Swift.app/TestApp_Swift

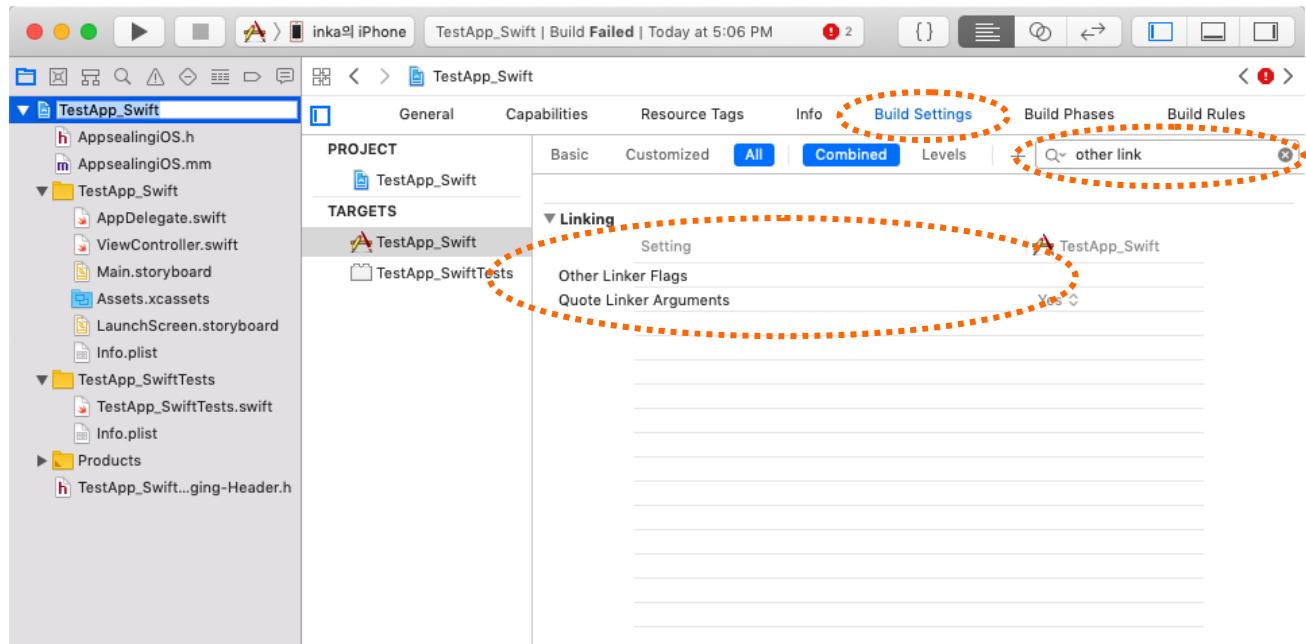
Undefined symbols for architecture arm64:
"ObjC_IsAbnormalEnvironmentDetected()", referenced from:
-[AppSealingInterface _IsAbnormalEnvironmentDetected] in AppsealingiOS.o
"Appsealing()", referenced from:
iOS() in AppsealingiOS.o
ld: symbol(s) not found for architecture arm64
clang: error: linker command failed with exit code 1 (use -v to see invocation)

symbol(s) not found for architecture arm64
! 1 linker command failed with exit code 1 (use -v to see invocation)
Build failed 31/01/2019, 5:06 PM 0.8 seconds
1 error

```

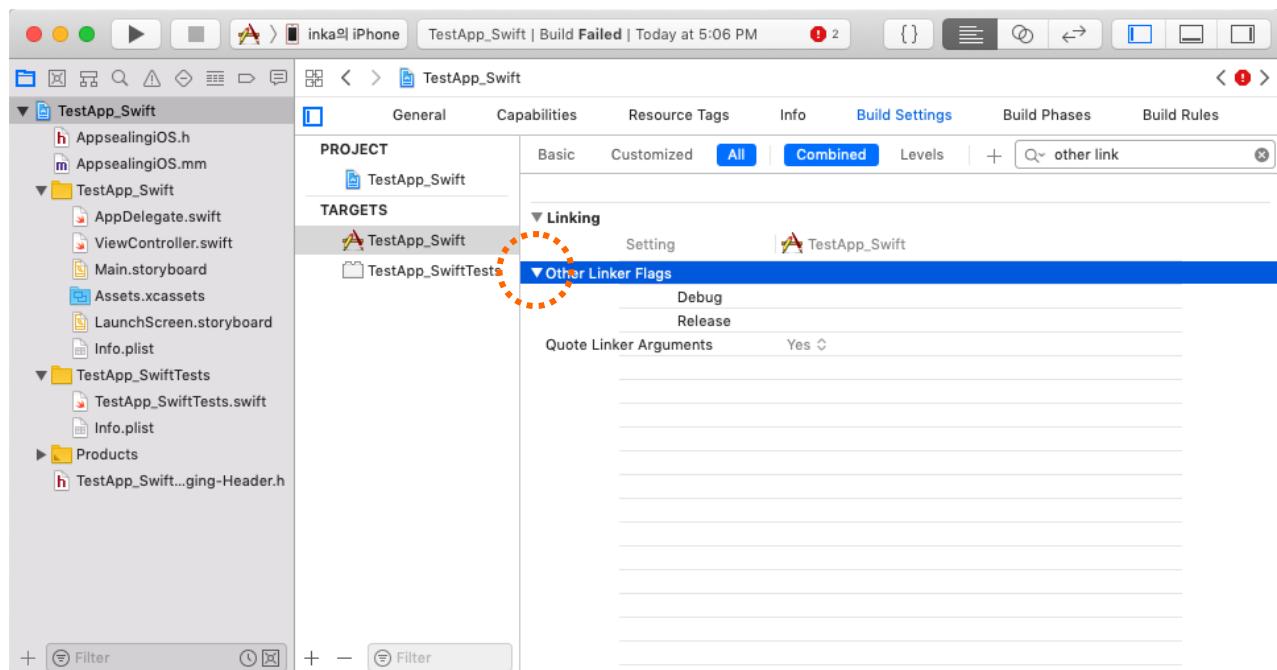
이 링커 오류는 "Build Settings"의 "Other Linker Flags" 필드 값이 없거나 잘못된 경우 발생합니다.

먼저, "Build Settings"의 **"Other Linker Flags"** 필드 값을 확인해보십시오.



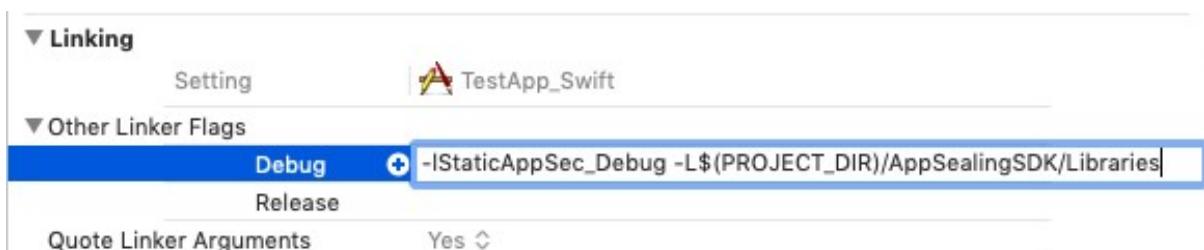
다음으로 "Other Linker Flags" 설정을 복원하십시오.

- 1) 설정값 삭제 : "Other Linker Flags" 줄을 선택하고 'Delete' 키를 눌러 삭제합니다.
- 2) "Other Linker Flags"의 왼쪽 삼각형 아이콘을 클릭하여 "Other Linker Flags"를 확장하십시오.



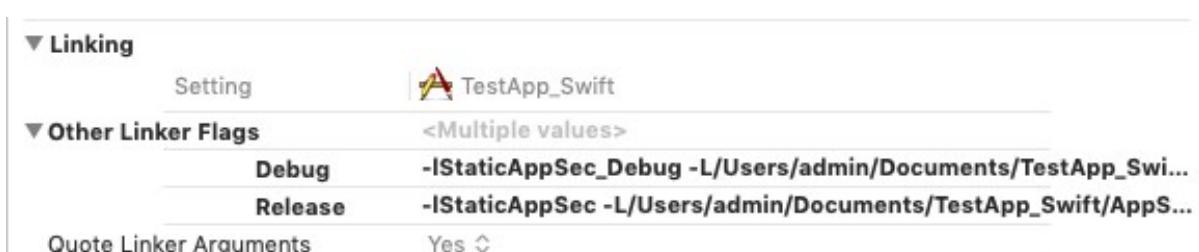
- 3) "Debug" 항목을 선택하고 클릭하여 아래 텍스트를 입력하십시오.

*-IStaticAppSec\_Debug -L\$(PROJECT\_DIR)/AppSealingSDK/Libraries*



- 4) "Release" 항목을 선택하고 클릭하여 아래 텍스트를 입력하십시오.

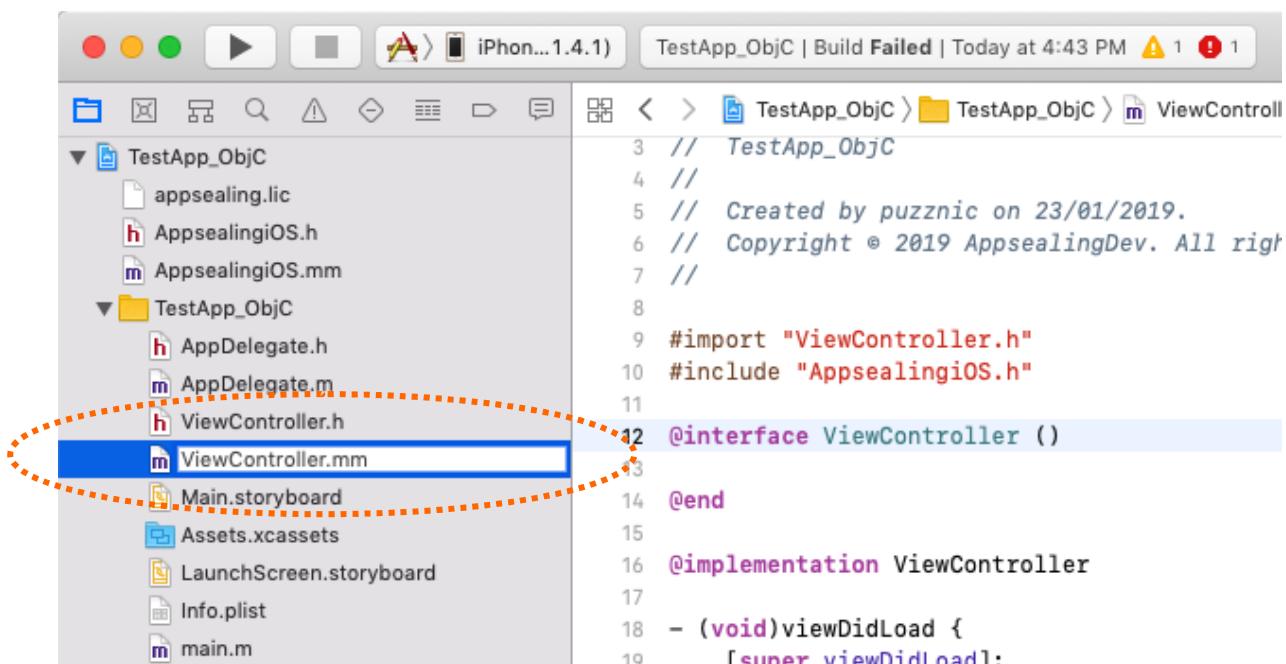
*-IStaticAppSec -L\$(PROJECT\_DIR)/AppSealingSDK/Libraries*



6-5 Xcode Build error (5) **Undefined symbols for architecture arm64:**  
"ObjC\_IsAbnormalEnvironmentDetected()" (Objective-C 프로젝트)

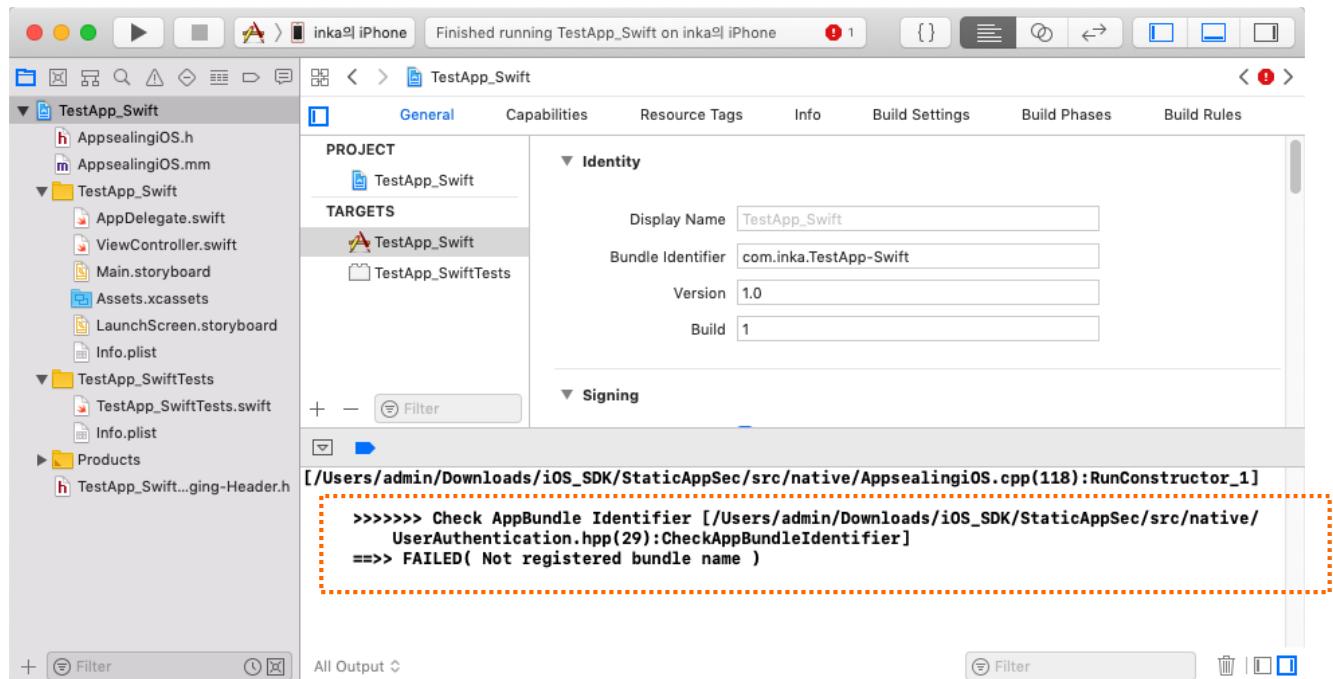
```
! Link /Users/admin/Desktop/Build/Intermediates.noindex/TestApp_ObjC.build/Release-iphoneos/TestApp_Obj... ! 1
Ld /Users/admin/Desktop/Build/Intermediates.noindex/TestApp_ObjC.build/Release-iphoneos/
TestApp_ObjC.build/Objects-normal/armv7/TestApp_ObjC normal armv7 (in target:
TestApp_ObjC)
cd /Users/admin/Documents/TestApp_ObjC
export IPHONEOS_DEPLOYMENT_TARGET=9.0
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/
clang++ -arch armv7 -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/
iPhoneOS.platform/Developer/SDKs/iPhoneOS12.1.sdk -L/Users/admin/Desktop/Build/Products/
Release-iphoneos -F/Users/admin/Desktop/Build/Products/Release-iphoneos -filelist /Users/
admin/Desktop/Build/Intermediates.noindex/TestApp_ObjC.build/Release-iphoneos/
TestApp_ObjC.build/Objects-normal/armv7/TestApp_ObjC.LinkFileList -Xlinker -rpath -
-Xlinker @executable_path/Frameworks -miphoneos-version-min=9.0 -dead_strip -Xlinker -
object_path_lto -Xlinker /Users/admin/Desktop/Build/Intermediates.noindex/
TestApp_ObjC.build/Release-iphoneos/TestApp_ObjC.build/Objects-normal/armv7/
TestApp_ObjC_lto.o -fembed-bitcode-marker -stdlib=libc++ -fobjc-arc -fobjc-link-runtime -
lStaticAppSec -L/Users/admin/Documents/TestApp_ObjC/AppSealingSDK/Libraries -Xlinker -
dependency_info -Xlinker /Users/admin/Desktop/Build/Intermediates.noindex/
TestApp_ObjC.build/Release-iphoneos/TestApp_ObjC.build/Objects-normal/armv7/
TestApp_ObjC_dependency_info.dat -o /Users/admin/Desktop/Build/Intermediates.noindex/
TestApp_ObjC.build/Release-iphoneos/TestApp_ObjC.build/Objects-normal/armv7/TestApp_ObjC
```

이 링커 오류는 프로젝트가 Objective-C 기반일 때 ViewController 코드의 확장자가 ".m"인 경우 발생합니다. 파일 확장자를 ".mm"으로 변경하십시오.

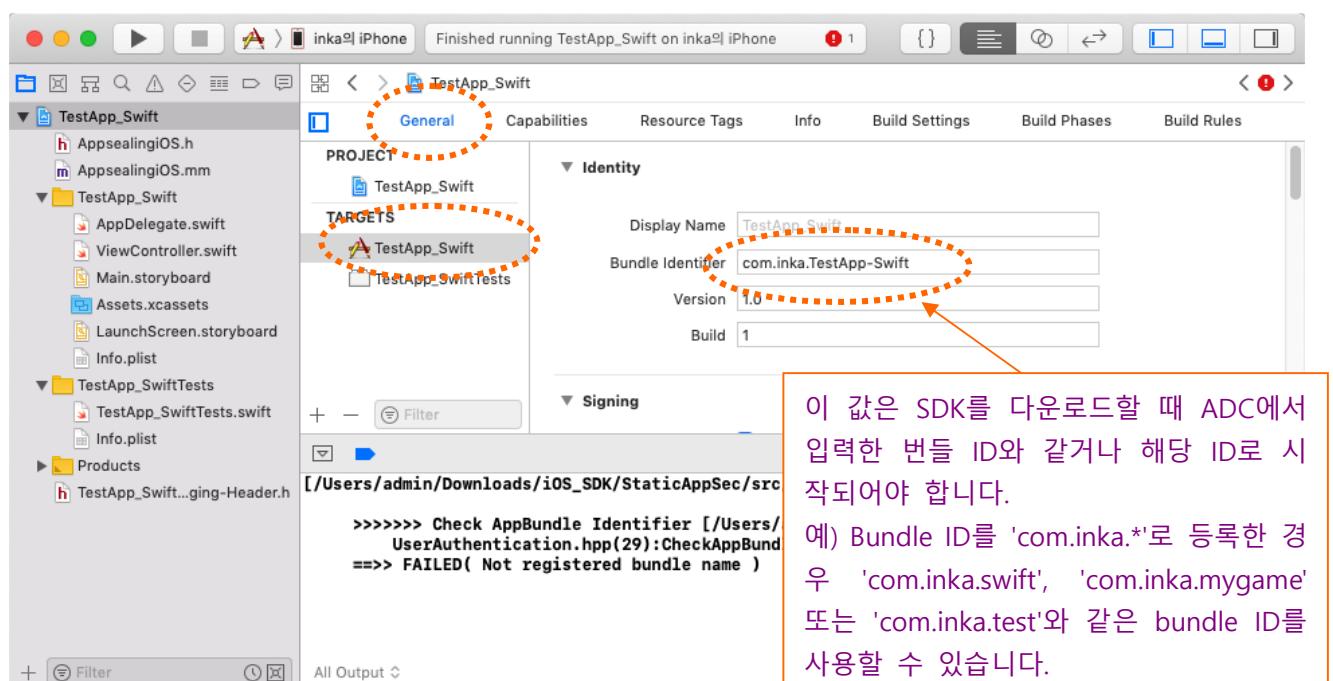


## 6-6 Execution error (!) 앱이 실행되고 곧 바로 종료됩니다.

기기에서 앱이 실행 직후 종료된 경우 bundle id가 유효한지 실행 로그 메시지를 확인하십시오.

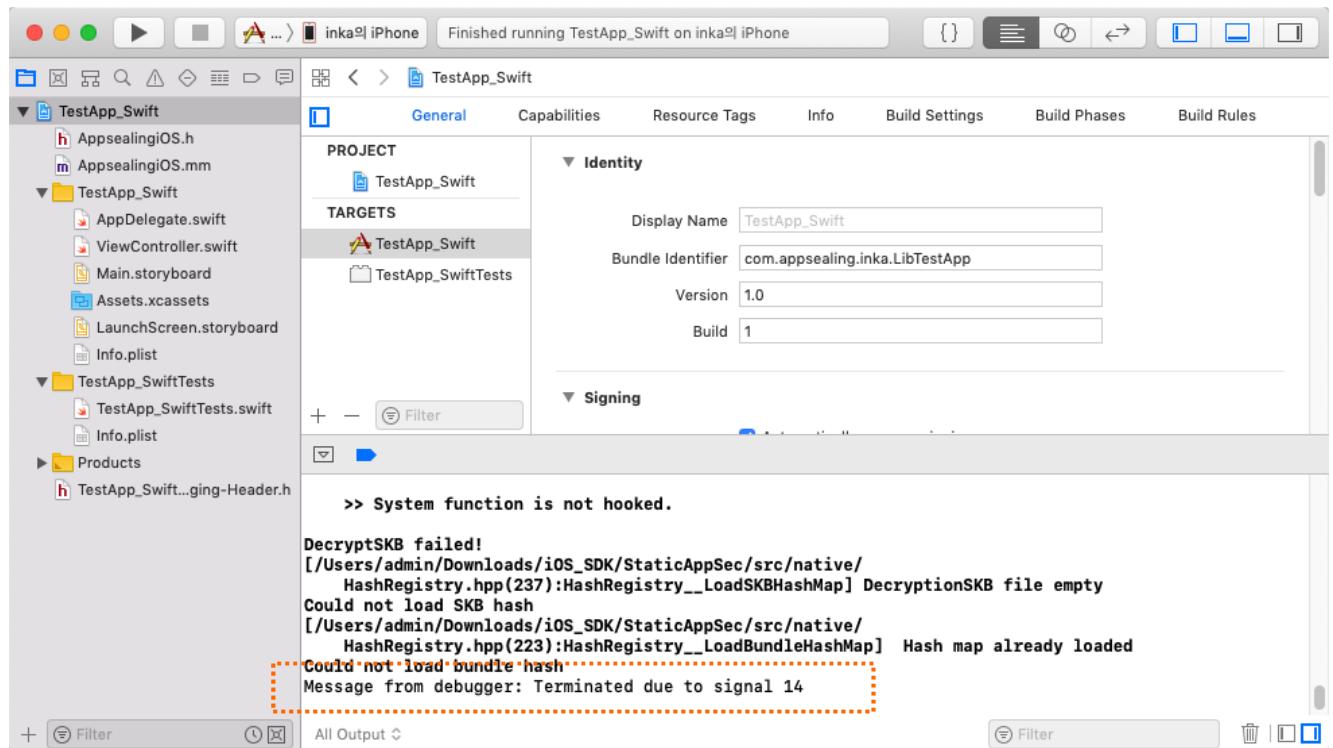


로그 메시지에 "`==> FAILED( Not registered bundle name )`" 와 같은 문자열이 포함되어 있으면 현재 프로젝트의 bundle id가 AppSealing SDK를 다운로드하는 동안 AppSealing Developer Center (ADC)에 제대로 등록되어 있지 않다는 의미입니다. ADC에서 SDK를 다운로드할 때 입력한 값과 같은 bundle id를 사용하십시오.



## 6-7 Execution error (II) 앱이 실행 중 갑자기 종료됩니다.

실행 후 약 20초 후에 갑자기 앱이 종료된 경우, 실행 로그를 확인하여 앱이 릴리스로 실행되었는지 확인할 수 있습니다.

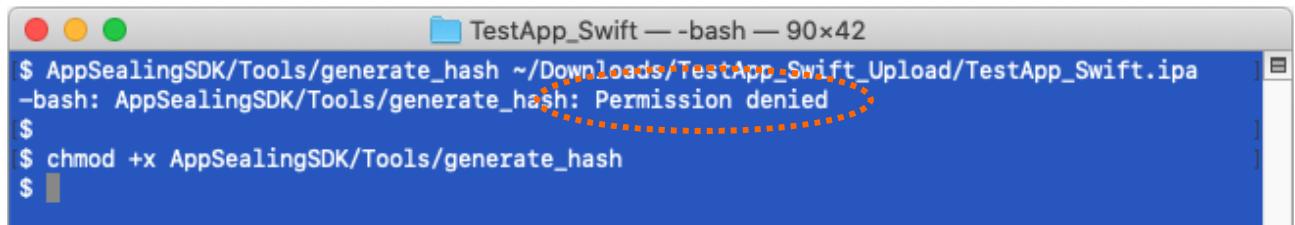


로그 메시지에 "Message from debugger: Terminated due to signal 14" 와 같은 문자열이 포함되어 있는 경우 앱이 release로 빌드, 실행되었고 AppSealing 보안 로직이 활성화되었음을 알 수 있습니다. Release 모드의 AppSealing 보안 로직은 탈옥된 장치, 실행 파일의 복호화 된 상태, 디버거 연결과 같은 보안 침입을 확인하며, 어떤 문제가 있을 경우 사용자의 조치나 기타 상황에 관계없이 20초 후에 앱을 닫습니다.

Release 모드에서는 AppSealing 로직이 활성화되고 앱이 종료되므로 AppStore 또는 TestFlight에 배포하기 전까지 디버그 모드로 앱을 테스트하십시오. AppSealing 로직은 release 모드로 빌드 시 Xcode 장치에서 앱을 실행하면 디버깅 환경으로 탐지하거나 실행 파일이 비정상적으로 해독되어 있음을 감지하고 20초 후에 앱을 종료합니다.

### 6-8 무결성/인증서 스냅샷 생성 스크립트가 실행되지 않습니다.

3-5 단계의 무결성/인증서 스냅샷 정보 생성 단계에서 아래와 같이 스크립트 실행이 실패하고 “권한 없음” 오류 메시지가 출력되는 경우가 발생할 수 있습니다.



```
$ AppSealingSDK/Tools/generate_hash ~/Downloads/TestApp_Swift_Upload/TestApp_Swift.ipa
-bash: AppSealingSDK/Tools/generate_hash: Permission denied
$ chmod +x AppSealingSDK/Tools/generate_hash
$
```

이 경우 아래와 같이 스크립트 파일에 실행 권한을 추가하는 명령을 실행하고 다시 시도해 보십시오.

```
$ chmod +x AppSealingSDK/Tools/generate_hash
```

### 6-9 Continuous Integration Tools에서 AppSealing SDK 사용

AppSealing SDK는 “Jenkins” 또는 “Buddy”와 같은 CI (Continuous Integration) 도구를 이용하여 통합될 수 있습니다. AppSealing SDK가 Xcode 프로젝트에 적용되면 평소와 같이 명령 줄 쉘 스크립트를 사용하여 프로젝트를 아카이브하고 내보낼 수 있습니다.