Confusion about access groups and iOS extensions.

I have an app that's been out for awhile and uses the Keychain to store a token. Now, I'm adding an extension to my app, and so I'm using the methods described in the "App Extension Best Practices" session from 2015 to share a secret between my app and my extension using an access group.

5.9k

My question is about the automatic search behavior for the query API. It's said in the session that update/delete/matching all "do the right thing" with "accessible" keychains. What keychains are supposed to be accessible to the extension?

What I'm finding is that for existing app users (people who had saved the secret before I was using access groups), my extension has access to the keychain item despite it not having an access group. What I anticipated was that that wouldn't work, and I'd have to do some kind of migration step to move my non-access group item into an access group one. But it looks like I don't have to do that, at least when I'm debugging with Xcode. Am I missing something here or is that the expected behavior?

Security

Answer this Question

Asked 4 years ago by bzapaatheskmm

Add a Comment

Answers

I presume you're working on iOS (or watchOS or tvOS) here; if you're developing for the Mac, things are more complex.

You can think of the keychain as a database table, where rows in the table are keychain items and columns are various keychain attributes. One of those attributes is

`kSecAttrAccessGroup`

, the **access group** of the item. Here's how things break down by API:

- For the query APIs (like

  `SecItemCopyMatching`

  ), if you don't supply

  `kSecAttrAccessGroup`

  in the query then it does a wildcard search, returning all items you have access to, that is, whose access group is contained in your app's **effective keychain access group list**.
- OTOH, when your query contains a value for

  `kSecAttrAccessGroup`

  then:
  - that value must be listed in your app's effective keychain access group list
  - items are returned if their access group matches that value
- When you create an item (

  `SecItemAdd`

  ), if you don't supply a value for

  `kSecAttrAccessGroup`

  then its access group is set to the **default keychain access group**.
- OTOH, if you do supply a value for

  `kSecAttrAccessGroup`

  then that value is used as the access group of the new item. Obviously this must be in the app's effective keychain access group list

Note If your new item is not unique the add will fail with

`errSecDuplicateItem`

. To understand uniqueness in the keychain, read this post from the old DevForums. Your app's effective keychain access group list is defined by its entitlements (which, in turn, must be whitelisted by its provisioning profile). Specifically, it is formed by concatenating, in order, all of the following:

- the list from the

  `keychain-access-groups`

  entitlement (if any)
- the value of the

  `application-identifier`

  entitlement
- the list from the

  `com.apple.security.application-groups`

  entitlement (if any)

The first item in this list is the default keychain access group.

So, to debug keychain access group problems you must:

1. dump the entitlements of all parties involved
2. dump the access group of the items involved
3. evaluate the results based on the discussion above

You can dump entitlements like this:

```
$ codesign -d --entitlements :- /path/to/your.app
```

if you have an

`.appex`

inside your app, you must also dump its entitlements. To do this, pass in the path to that

`.appex`

. Make sure you do the one inside the app, not the one at the top level of the build folder, to avoid any problems with the entitlements being changed as the

`.appex`

is copied into your

`.app`

.

If you want to figure out the entitlements for your app that's currently in the store, you can download it via iTunes on the Mac, unpack it (the

`.ipa`

is actually a

`.zip`

), and dump entitlements from there.
Share and Enjoy
—
Quinn "The Eskimo!"
Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Posted 4 years ago by eskimo

Add a Comment

> So, to debug keychain access group problems you must:
> 1. dump the entitlements of all parties involved

Quinn, please describe what the values should look like.

I see in the provisioning profile:

```
<key>keychain-access-groups</key>                    <-- #1
<array>
    <string>Axxxxxx2.*</string>
</array>
```

And in the entitlements file:

```
<key>keychain-access-groups</key>                    <-- #1
<array>
    <string>Axxxxxx2.com.company.appname.apps.shared</string>
</array>
```

(and am receiving the -34018 errSecMissingEntitlement error, yet the keychain clearly works when NOT including the kSecAccessGroup key and forgo the use of kSecAttrSynchronizable)

Posted 4 years ago by davidcc

Add a Comment

> And in the entitlements file

Is that the entitlements file (that is, the

`.entitlements`

file you see in your Xcode project)? Or the entitlements dumped from the built binary? You really need to start with the latter whilst debugging entitlement problems.
My Debugging Entitlement Issues explains how I go about this, and has a link to our official doc on the subject (TN2415).
Share and Enjoy
—
Quinn "The Eskimo!"
Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Posted 4 years ago by eskimo

Add a Comment

@eskimo thank you very much for the detailed explanation.

I did small tests around what you wrote:

OTOH, when your query contains a value for

`kSecAttrAccessGroup`

then:

- that value must be listed in your app's effective keychain access group list
- items are returned if their access group matches that value

From what I can see, Keychain Access Group might not be contained in Access Group and the application can still read values from the shared keychain. See the part of my entitlements dump:

```
<key>application-identifier</key>
<string>ABCDEF1234.com.example.MyApp</string>
<key>com.apple.developer.team-identifier</key>
<string>ABCDEF1234</string>
<key>com.apple.security.application-groups</key>
<array>
<string>group.com.example.MyApp</string>
</array>
<key>keychain-access-groups</key>
<array>
<string>ABCDEF1234.com.example.MyApp</string>
<string>ABCDEF1234.com.whatever</string>
</array>
```

The application still can access values stored with "ABCDEF1234.com.whatever" from other application (same Team ID and Keychain Sharing Group). Is it expected or I'm missing something here?

Posted 3 years ago by playability

Add a Comment

> The application still can access values stored with "ABCDEF1234.com.whatever" from other application (same Team ID and Keychain Sharing Group). Is it expected or I'm missing something here?

I'm not sure I'm reading this correctly but:

1. If the item is stored in

   `ABCDEF1234.com.whatever`

   , and
2. Your

   `keychain-access-groups`

   entitlement contains

   `ABCDEF1234.com.whatever`

   (which it does in the example you posted), then

3. It makes sense that the item is accessible to you

Share and Enjoy
—
Quinn "The Eskimo!"
Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Posted 3 years ago by eskimo

Add a Comment

I wonder, why

`ABCDEF1234.com.whatever`

should _not_ be contained in the com.apple.security.application-groups in a form of

```
<key>com.apple.security.application-groups</key>
<array>
<string>group.com.example.MyApp</string>
<string>group.com.whatever</string>
</array>
```

?

Posted 3 years ago by playability

Add a Comment

App Groups (

`com.apple.security.application-groups`

) and keychain access groups (

`keychain-access-groups`

).
I are not the same thing. If you just want to share keychain items, you should add a keychain access group. If you want to share other stuff, add an App Group. That also gives you access to a matching keychain access group (by the third point in the list from my 4 Nov post). That access is enabled regardless of whether you put the App Group name in the keychain access group entitlement.
Share and Enjoy
—
Quinn "The Eskimo!"
Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Posted 3 years ago by eskimo

Add a Comment

Thank you! This makes perfect sense to me now.

Posted 3 years ago by playability

Add a Comment

Your Answer

Enter your answer here

Live Preview

Submit