

AuthorizationPlug-in: Programming

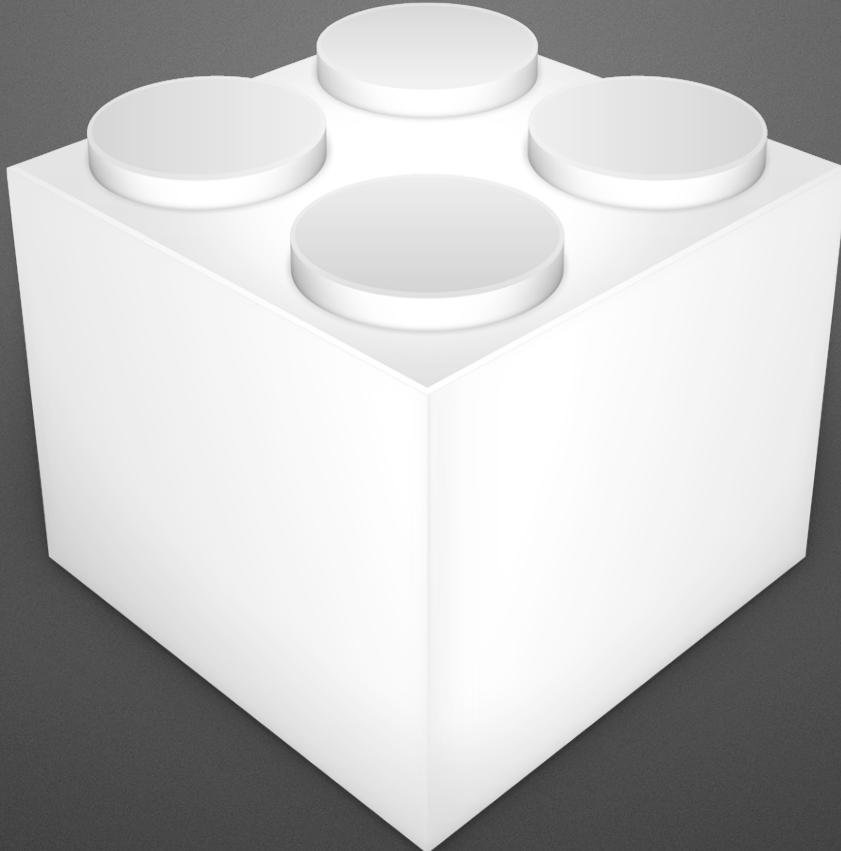
whoami: TomBurgin

Slack: tburgin

Github: tburgin

Twitter: @tomjburgin

Work: (NIH) SRA International



Authorization : Plug-in?

Plug-ins: GitHub



[MagerValp/LoginScriptPlugin](#)

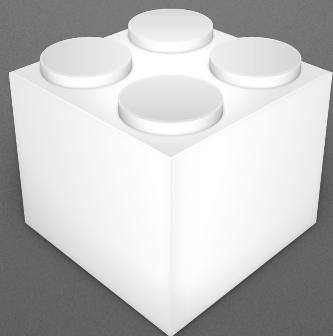


[google/macops/tree/master/keychainminder](#)



[ygini/Unlock](#)

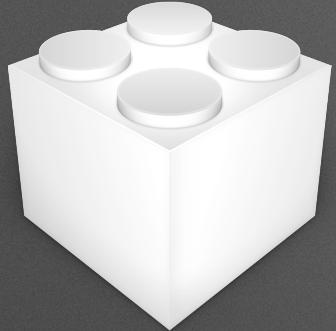
Plug-in : pieces



<string>.bundle on the file system</string>



<string>entry in the authorizationdb</string>



/Library/Security/SecurityAgentPlugins/



```
security authorizationdb read system.login.console  
security authorizationdb write system.login.console  
security authorizationdb read authenticate  
security authorizationdb write authenticate
```

Today: functionality



! Today : Plug-inCore

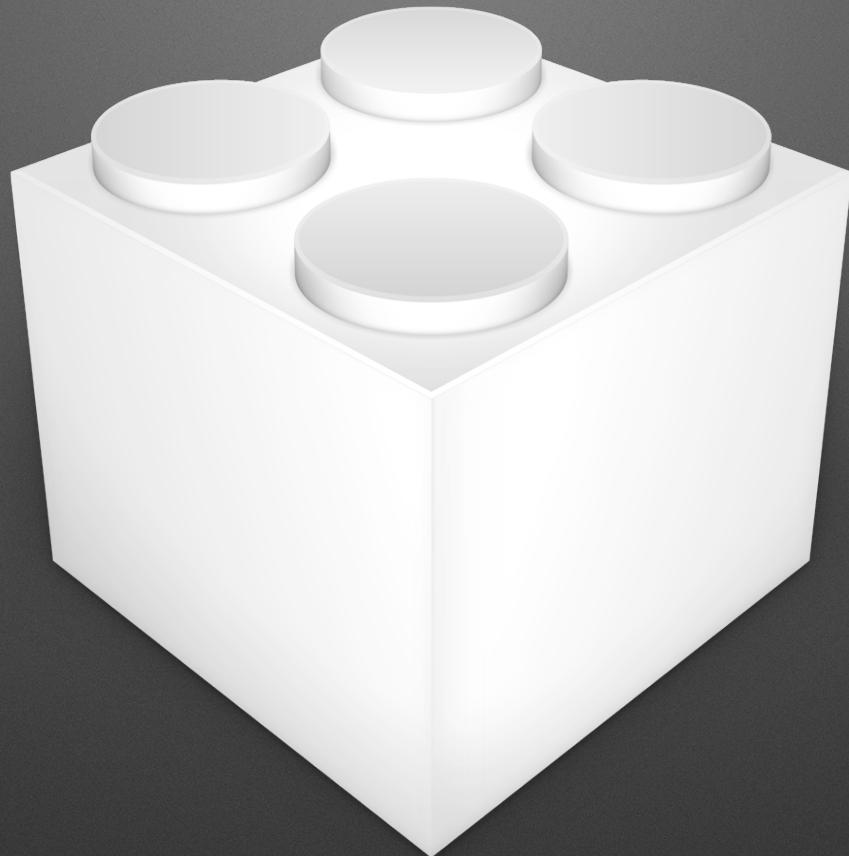
*Let's make some decisions: Authorization Plugin
Programming – Tom Burgin & Jeremy Baker*

<http://macadmins.psu.edu/conference/resources/>

VerifyAuthPlugin : Goals

```
<goal01>Create 2 Mechanisms</goal01>
<goal02>Share Data between Mechanisms</goal02>
    <goal03>Prompt user for PIN</goal03>
<goal04>Make an authorization decision</goal04>
```

VerifyAuthPlugin : Mechs



```
<string>VerifyAuthPlugin:MachinePIN,privileged</string>
<string>VerifyAuthPlugin:Verify</string>
```

```
security authorizationdb read system.login.console
```

```
<string>builtin:policy-banner</string>
<string>loginwindow:login</string>
<string>builtin:login-begin</string>
<string>builtin:reset-password,privileged</string>
<string>builtin:forward-login,privileged</string>
<string>builtin:auto-login,privileged</string>
<string>builtin:authenticate,privileged</string>
<string>VerifyAuthPlugin:MachinePIN,privileged</string>
<string>VerifyAuthPlugin:Verify</string>
<string>PKINITMechanism:auth,privileged</string>
<string>builtin:login-success</string>
<string>loginwindow:success</string>
<string>loginwindow:FDESupport,privileged</string>
<string>HomeDirMechanism:login,privileged</string>
<string>HomeDirMechanism:status</string>
<string>MCXMechanism:login</string>
<string>loginwindow:done</string>
```

```
<string>builtin:policy-banner</string>
<string>loginwindow:login</string>
<string>builtin:login-begin</string>
<string>builtin:reset-password,privileged</string>
<string>builtin:forward-login,privileged</string>
<string>builtin:auto-login,privileged</string>
<string>builtin:authenticate,privileged</string>

<string>VerifyAuthPlugin:MachinePIN,privileged</string>
<string>VerifyAuthPlugin:Verify</string>

<string>PKINITMechanism:auth,privileged</string>
<string>builtin:login-success</string>
<string>loginwindow:success</string>
<string>loginwindow:FDESupport,privileged</string>
<string>HomeDirMechanism:login,privileged</string>
<string>HomeDirMechanism:status</string>
<string>MCXMechanism:login</string>
<string>loginwindow:done</string>
```



tom



Tom Admin

→ <string>loginwindow:login</string> ←
<string>builtin:authenticate,privileged</string>
<string>VerifyAuthPlugin:MachinePIN,privileged</string>
<string>VerifyAuthPlugin:Verify</string>
<string>loginwindow:done</string>



Tom Admin



<string>loginwindow:login</string>

→<string>builtin:authenticate,privileged</string>←

<string>VerifyAuthPlugin:MachinePIN,privileged</string>

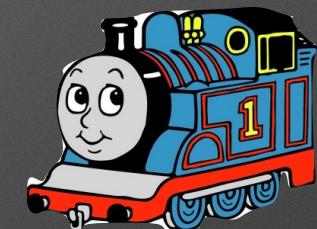
<string>VerifyAuthPlugin:Verify</string>

<string>loginwindow:done</string>

```
<string>loginwindow:login</string>
<string>builtin:authenticate,privileged</string>
```



PIN



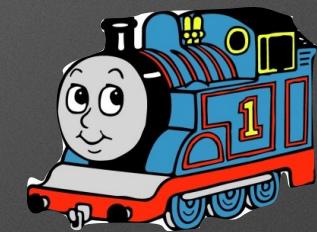
```
→ <string>VerifyAuthPlugin:MachinePIN,privileged</string> ←
    <string>VerifyAuthPlugin:Verify</string>
    <string>loginwindow:done</string>
```

```
<string>loginwindow:login</string>
<string>builtin:authenticate,privileged</string>
```

Verify

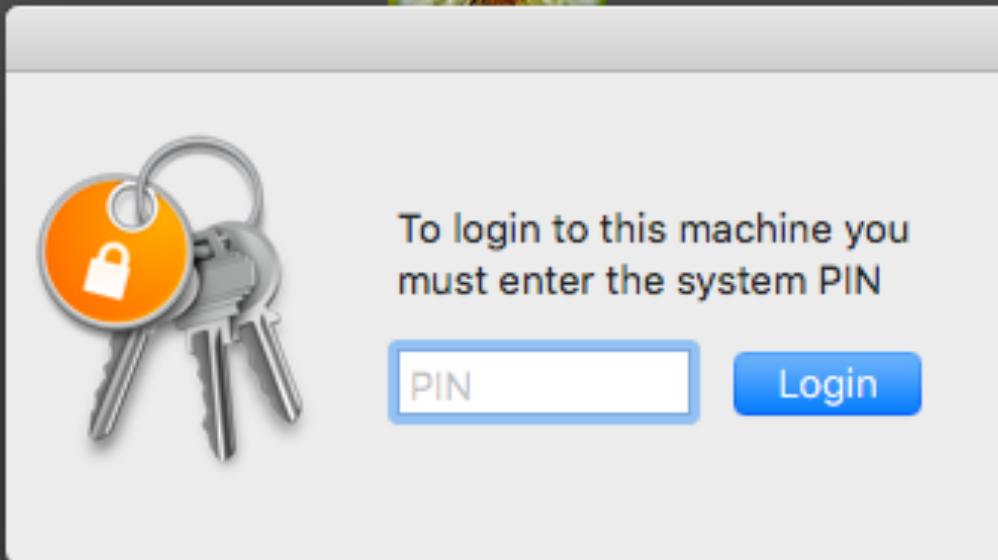


PIN



```
→ <string>VerifyAuthPlugin:MachinePIN,privileged</string>
    <string>VerifyAuthPlugin:Verify</string> ←
        <string>loginwindow:done</string>
```

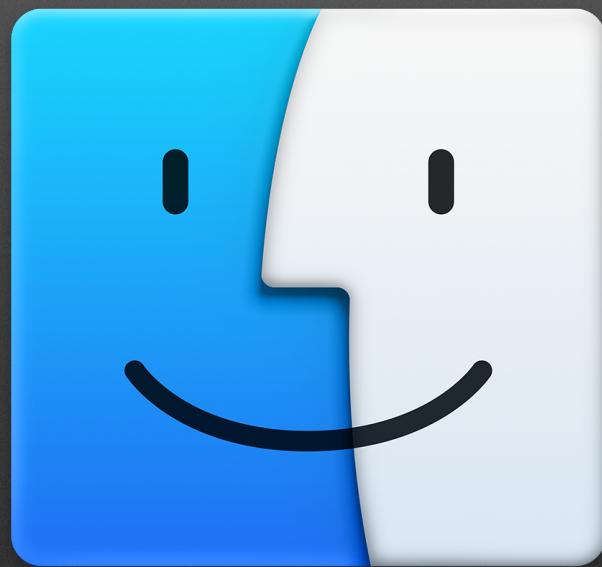
```
<string>loginwindow:login</string>
<string>builtin:authenticate,privileged</string>
```



```
→ <string>VerifyAuthPlugin:MachinePIN,privileged</string>
    <string>VerifyAuthPlugin:Verify</string> ←
    <string>loginwindow:done</string>
```

```
<string>loginwindow:login</string>
<string>builtin:authenticate,privileged</string>
<string>VerifyAuthPlugin:MachinePIN,privileged</string>
    <string>VerifyAuthPlugin:Verify</string>
```

→ <string>loginwindow:done</string> ←



Download



[github.com/tburgin/
MacTech_2015](https://github.com/tburgin/MacTech_2015)

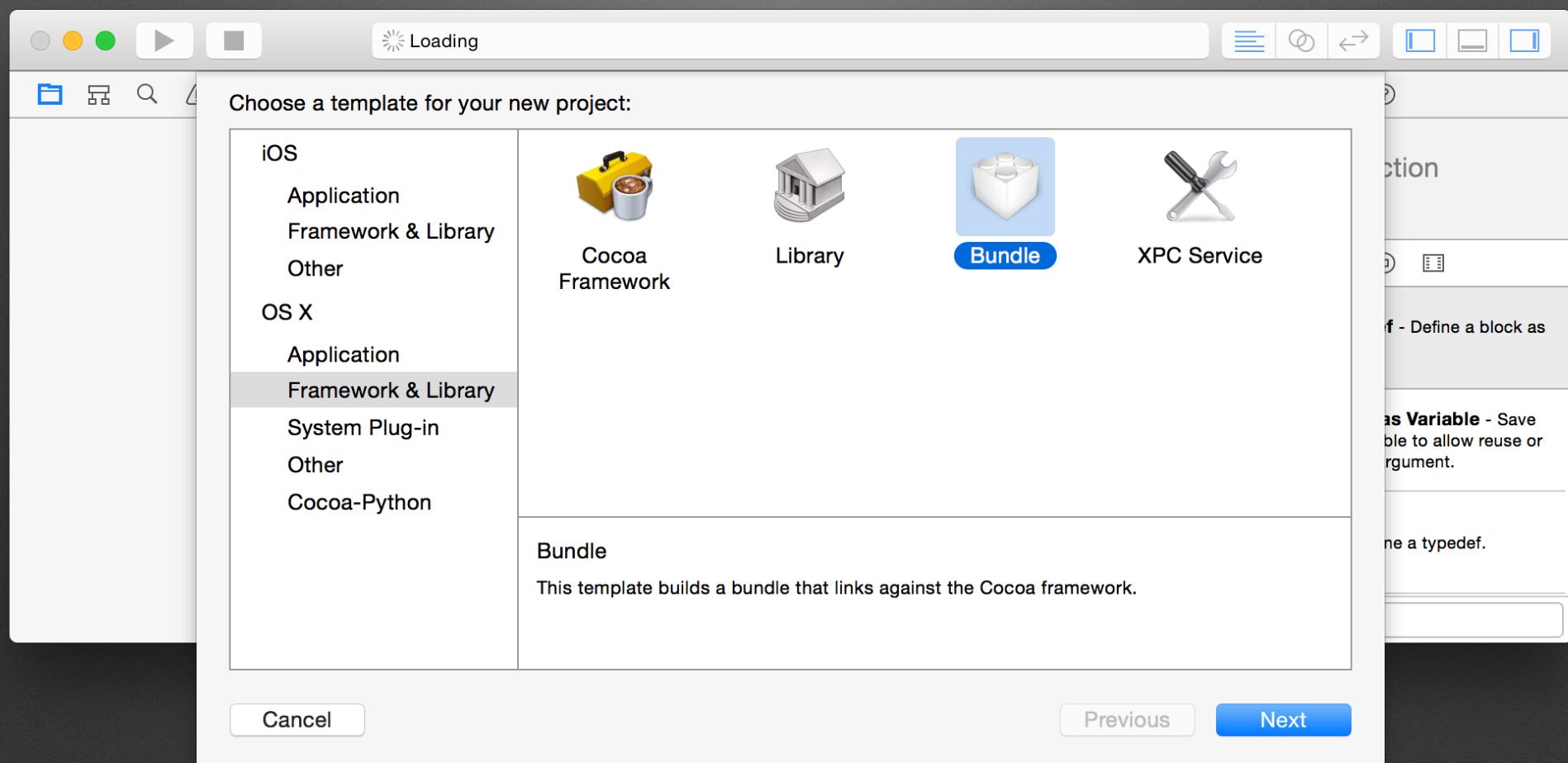


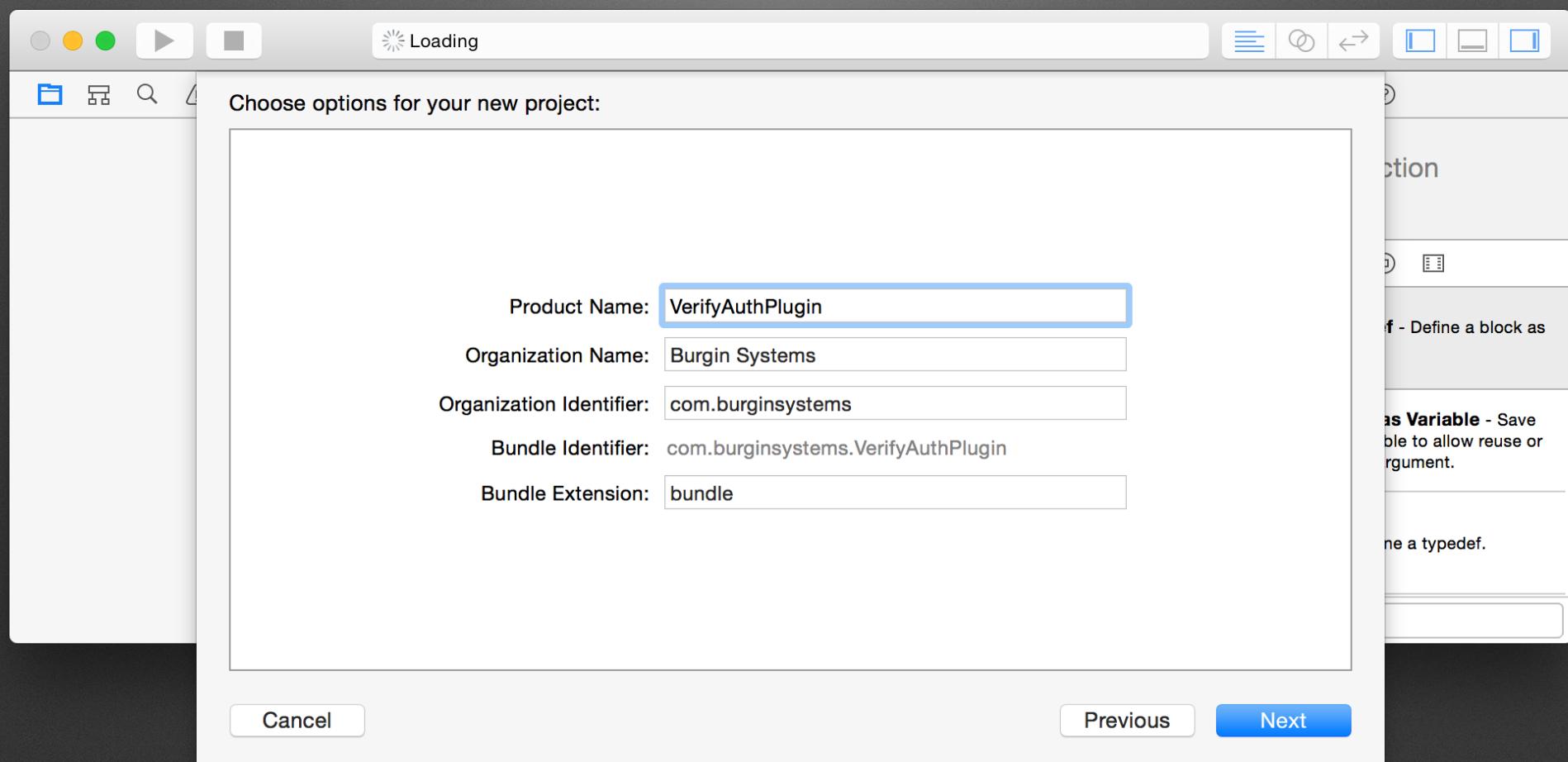
Code:Snippets

m



SWIFT





Show in Finder
Open with External Editor
Open As ►
Show File Inspector

New File...
New Project...
Add Files to “VerifyAuthPlugin”...

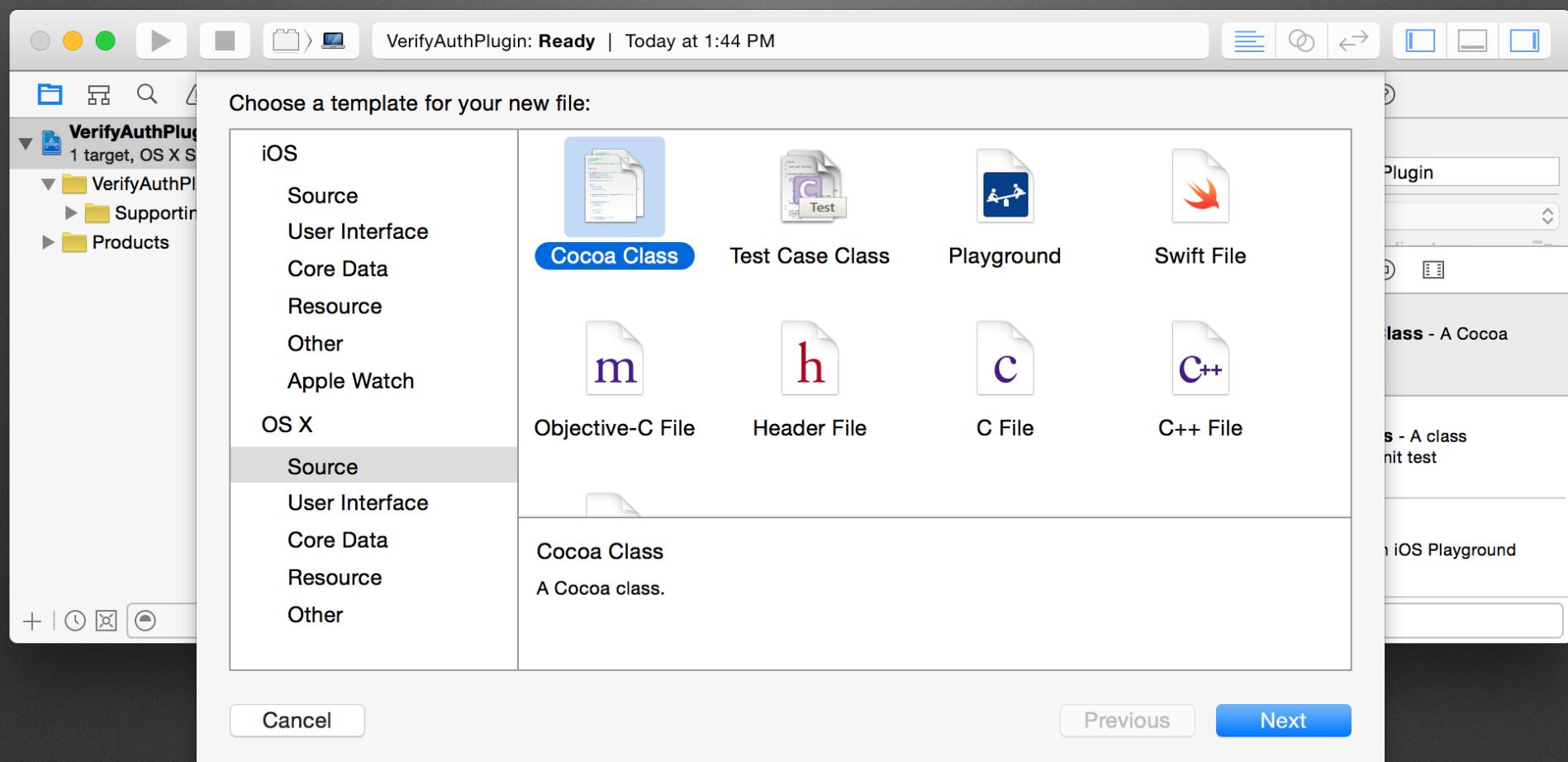
Delete

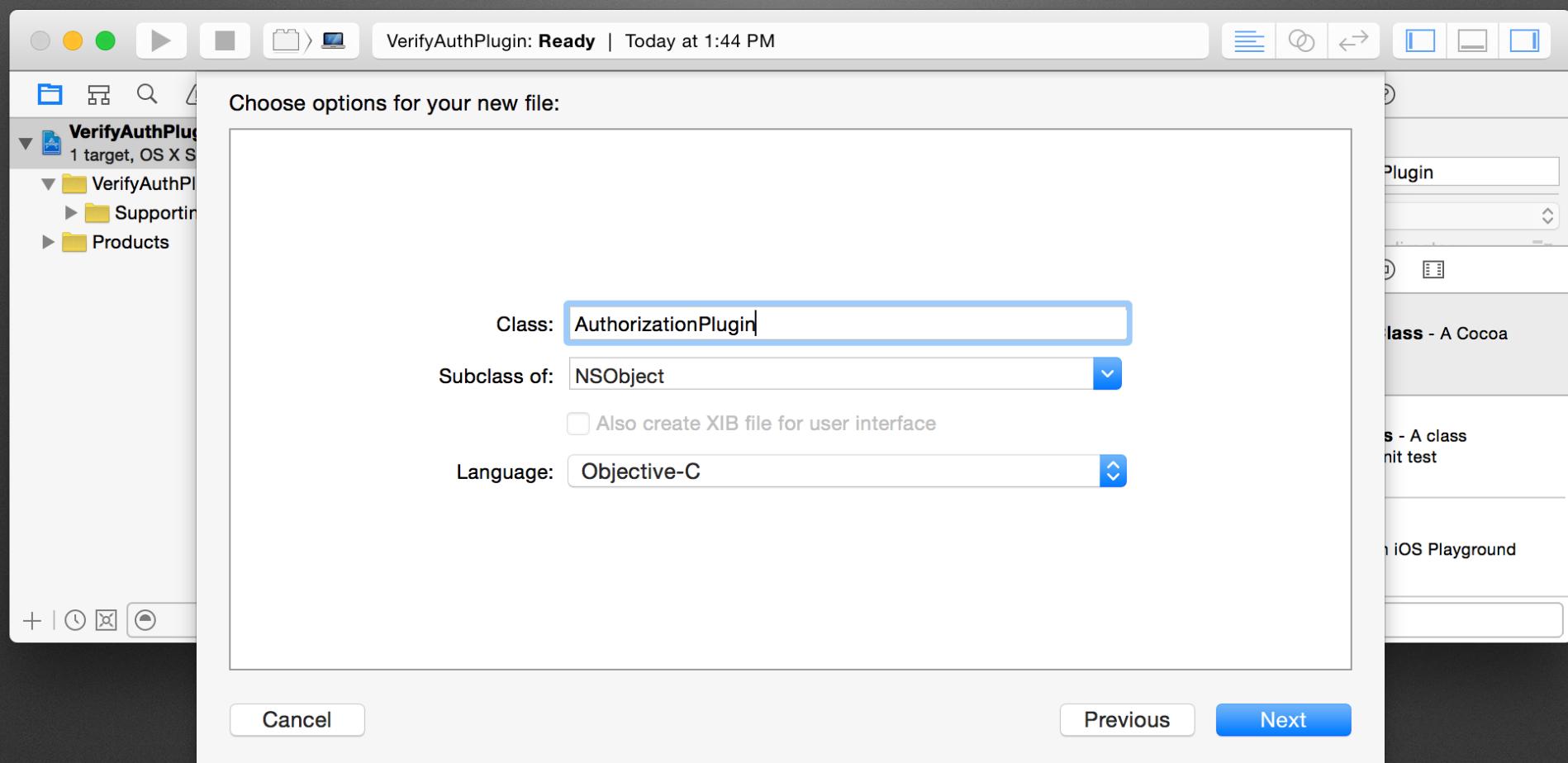
New Group
New Group from Selection

Sort by Name
Sort by Type

Find in Selected Groups...

Source Control ►
Project Navigator Help ►





Code:AuthorizationPlugin

Core: Done

*Let's make some decisions: Authorization Plugin
Programming – Tom Burgin & Jeremy Baker*

<http://macadmins.psu.edu/conference/resources/>

Mechanism: MachinePIN

- Show in Finder
- Open with External Editor
- Open As ►
- Show File Inspector

- New File...
- Add Files to “VerifyAuthPlugin”...

- Delete

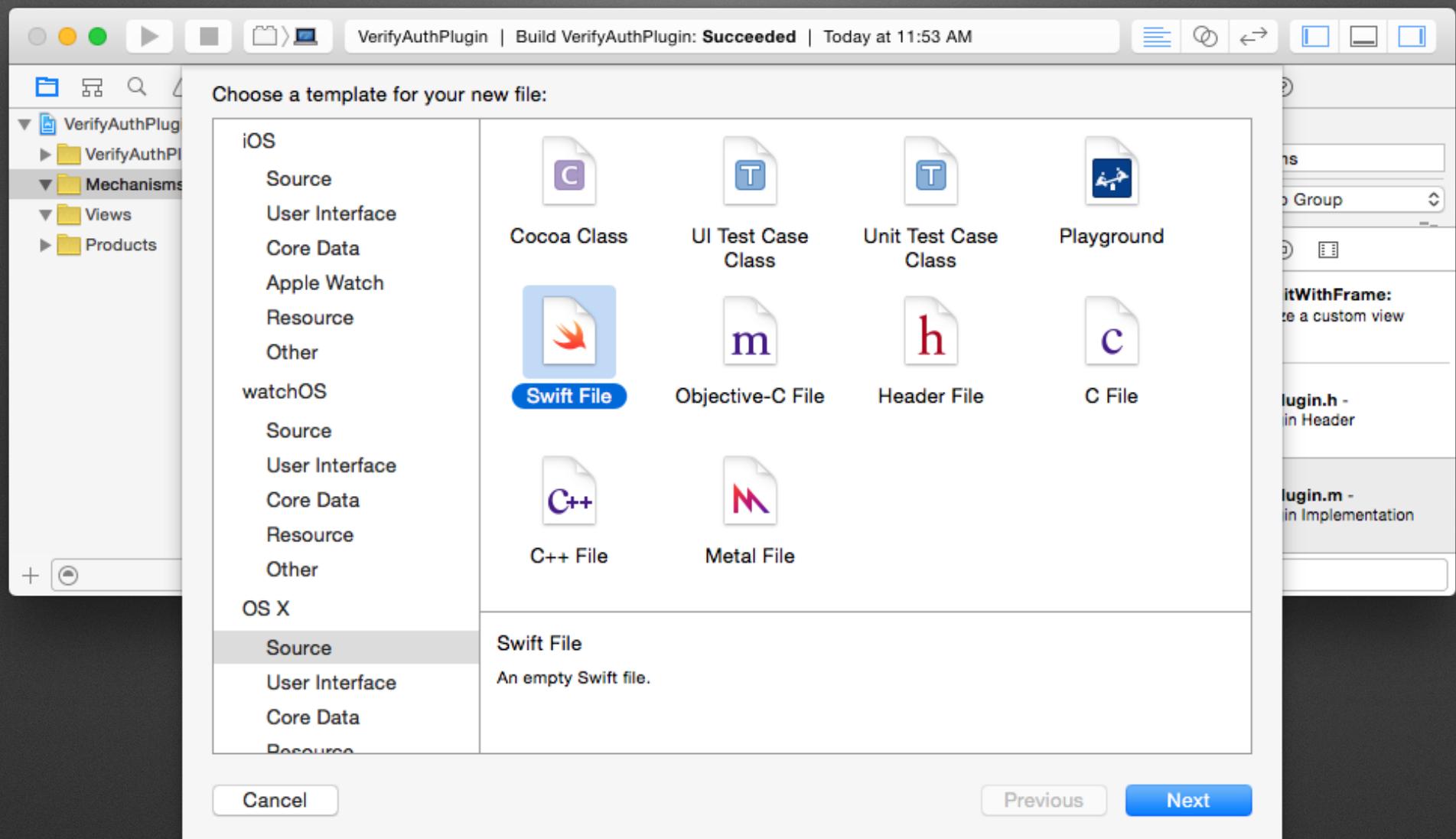
- New Group
- New Group from Selection

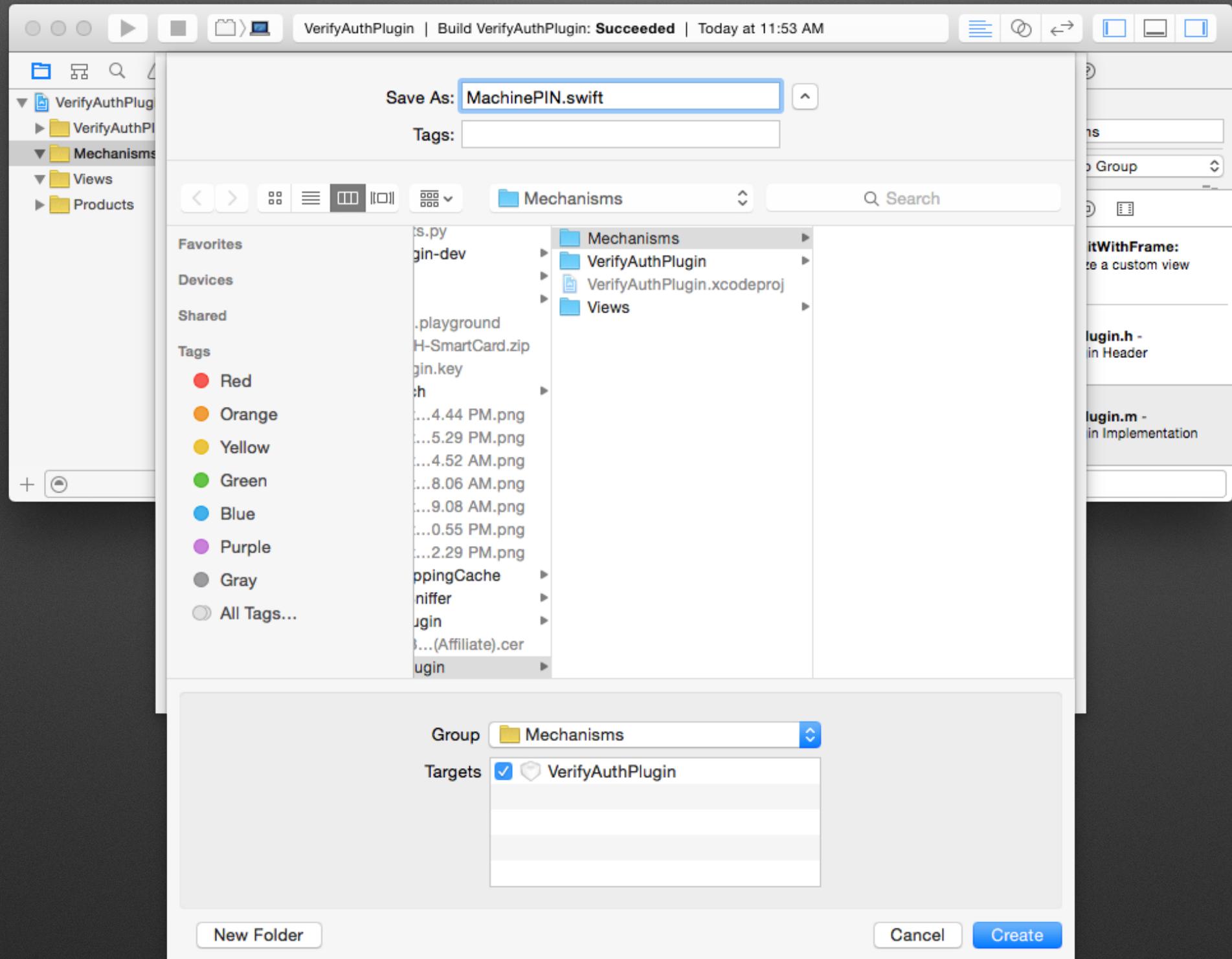
- Sort by Name
- Sort by Type

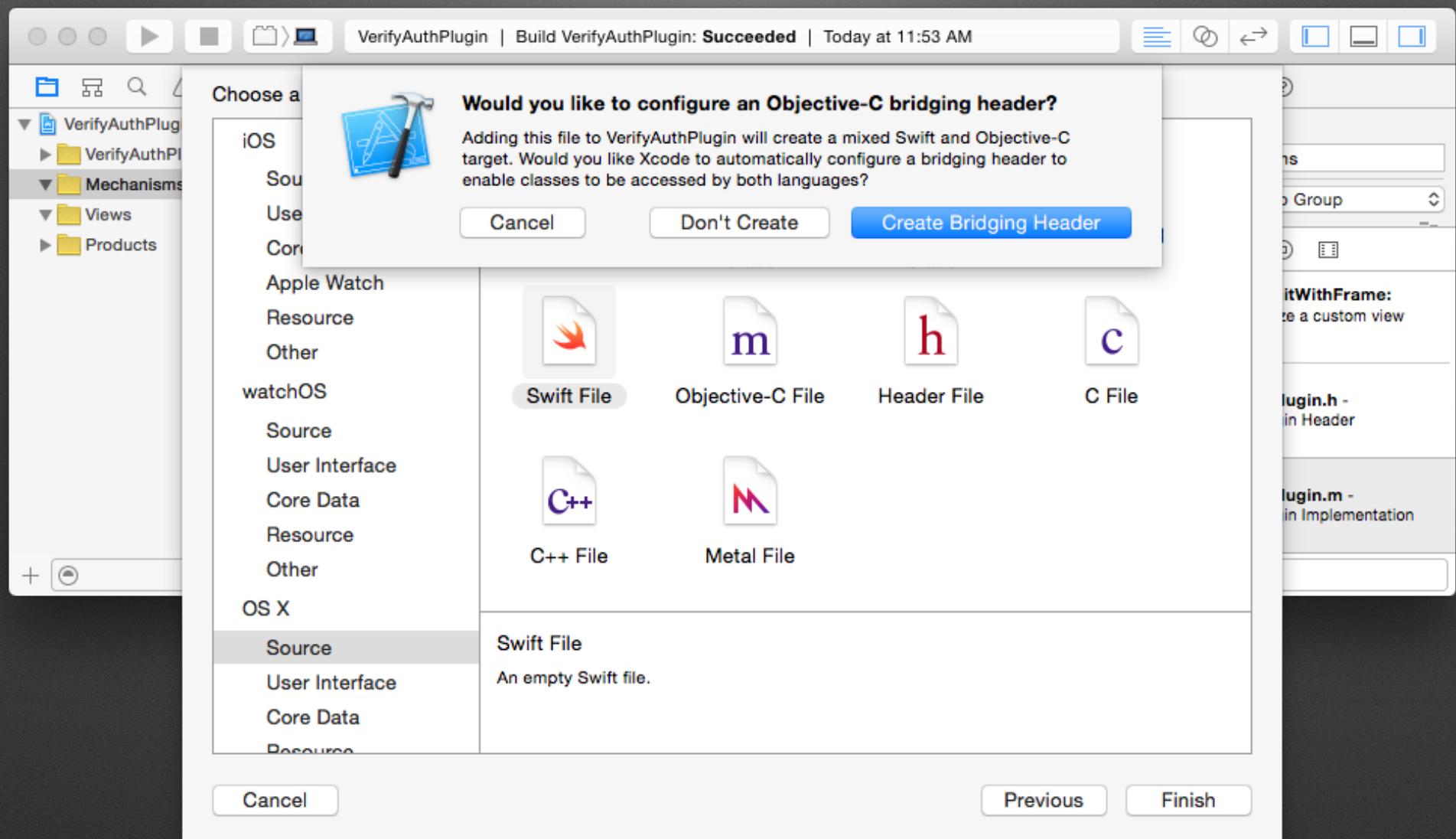
- Find in Selected Groups...

- Source Control ►

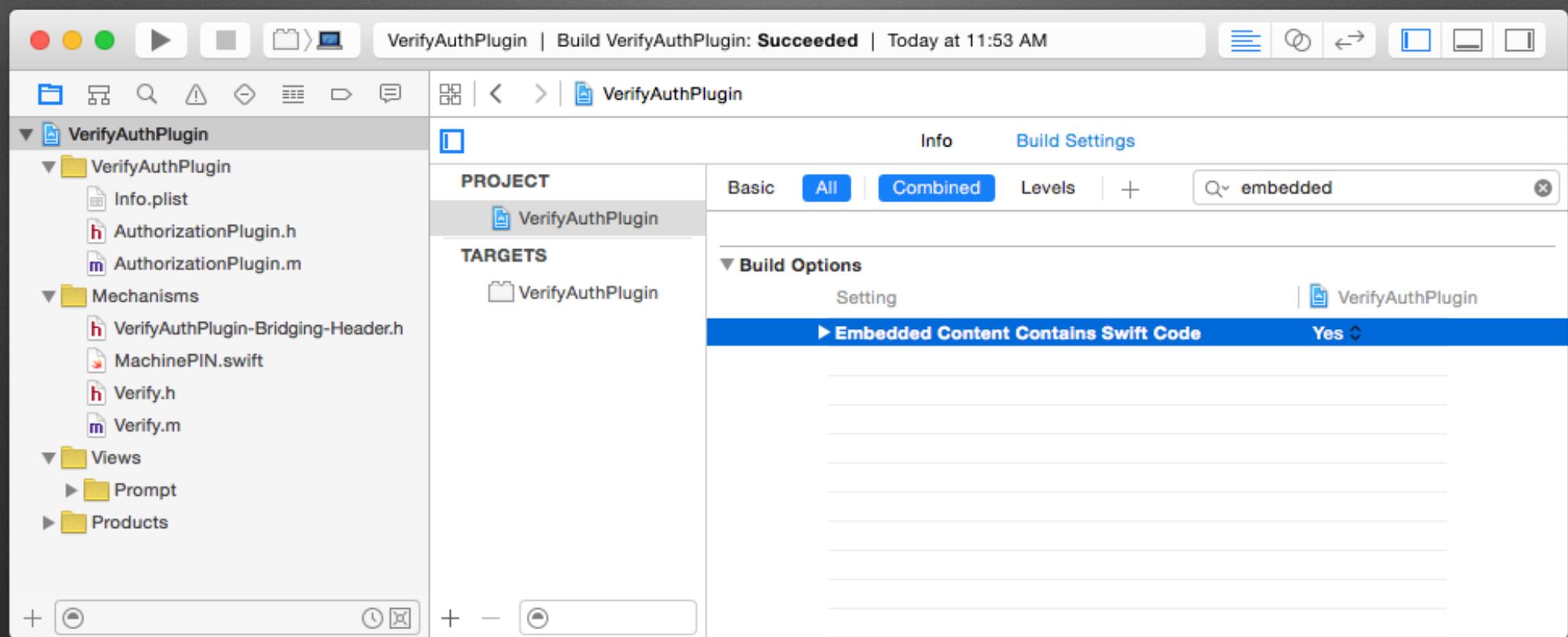
- Project Navigator Help ►







```
//  
// This is how we set the inter-mechanism context data  
private func setHintValue(pin : NSString?) -> Bool {  
  
    // Try and unwrap the optional NSString  
    guard let pin = pin  
    else {  
        NSLog([+] Failed to unwrap inPin");  
        return false  
    }  
  
    // Try and unwrap the optional NSData returned from archivedDataWithRootObject  
    // This can be decoded on the other side with unarchiveObjectWithData  
    guard let data : NSData = NSKeyedArchiver.archivedDataWithRootObject(pin)  
    else {  
        NSLog([+] Failed to unwrap archivedDataWithRootObject");  
        return false  
    }  
  
    // Fill the AuthorizationValue struct with our data  
    var value = AuthorizationValue(length: data.length,  
                                    data: UnsafeMutablePointer<Void>(data.bytes))  
  
    // Use the MechanismRecord SetHintValue callback to set the  
    // inter-mechanism context data  
    let err : OSStatus = self.mechanism.memory.fPlugin.memory.fCallbacks.memory.  
        SetHintValue(mechanism.memory.fEngine, contextPINDomain.UTF8String, &value)  
  
    return (err == errSecSuccess) ? true : false  
}
```



Code : MachinePIN

Mechanism: Verify

- Show in Finder
- Open with External Editor
- Open As ►
- Show File Inspector

- New File...
- Add Files to “VerifyAuthPlugin”...

- Delete

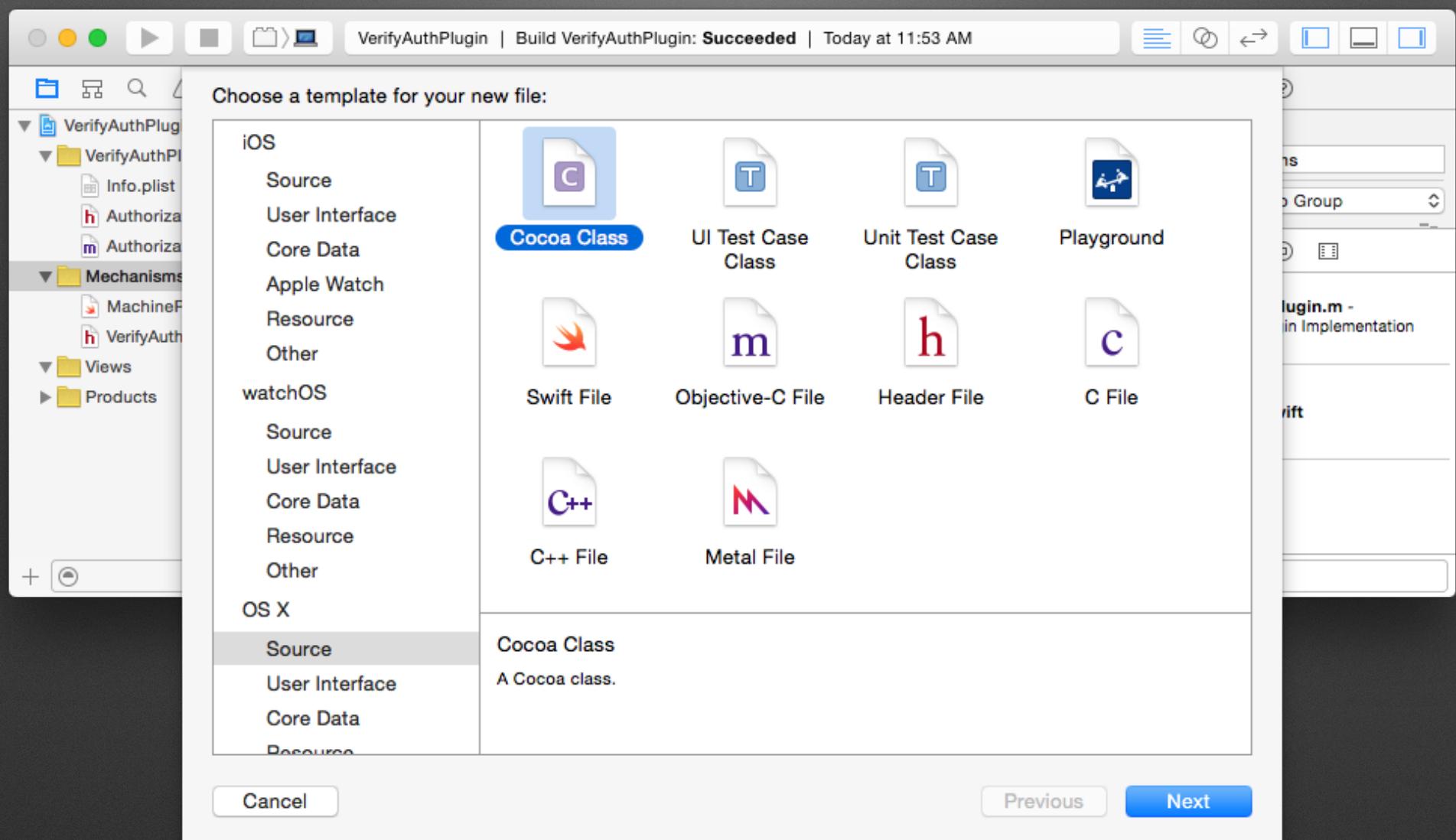
- New Group
- New Group from Selection

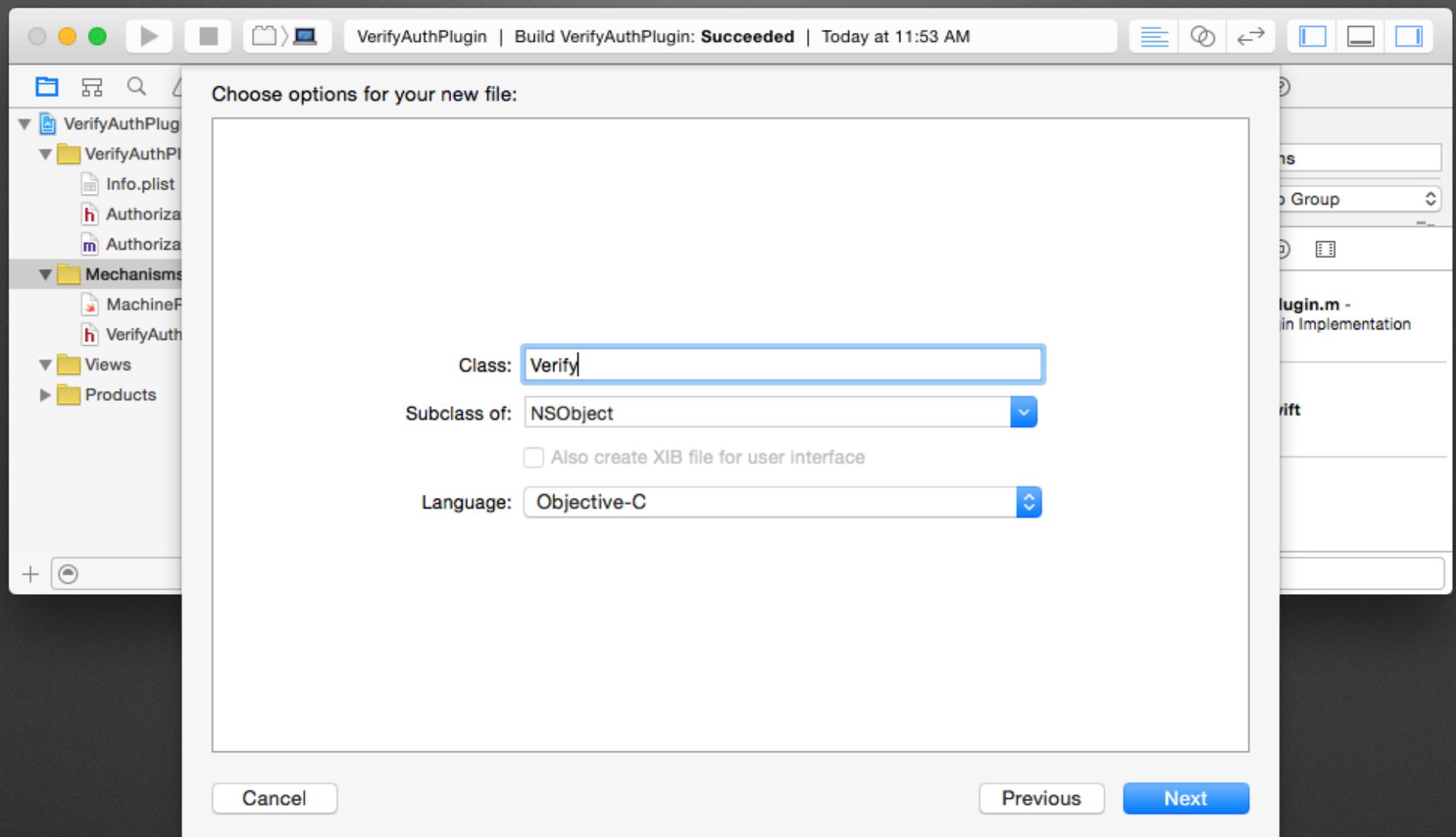
- Sort by Name
- Sort by Type

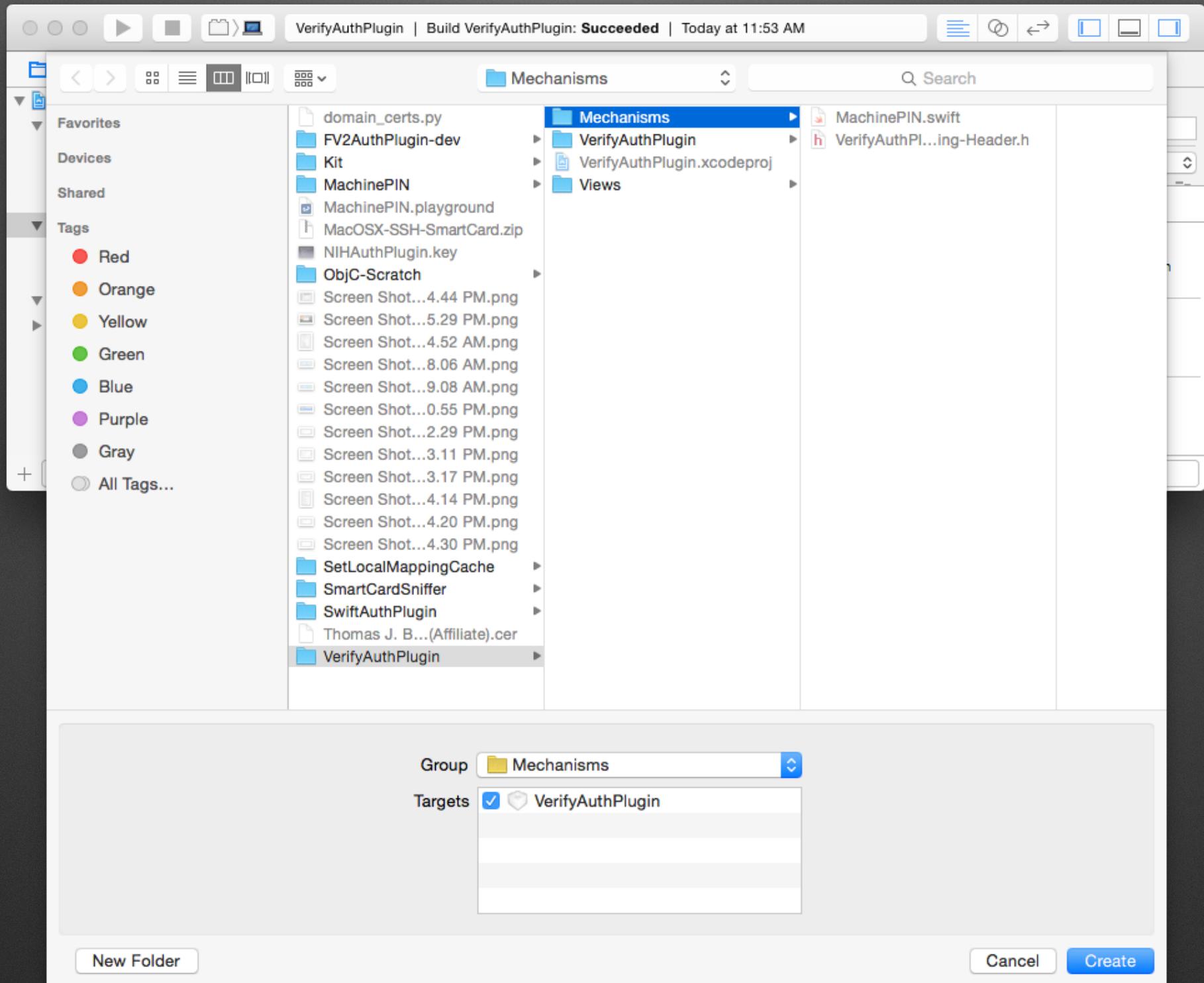
- Find in Selected Groups...

- Source Control ►

- Project Navigator Help ►







```
- (NSString *)getPIN {

    // Setup method variables
    NSString *pin;
    const AuthorizationValue *value;

    // This NSString will be used as the domain for the inter-mechanism context data
    NSString *contextPINDomain = @"com.burginsystems.pin";

    // Use the MechanismRecord GetHintValue callback to get the
    // inter-mechanism context data
    NSLog(@"[+] Attempting to read %@", contextPINDomain);
    if (_mechanism->fPlugin->fCallbacks->GetHintValue(_mechanism->fEngine,
                                                       [contextPINDomain UTF8String],
                                                       &value) == errAuthorizationSuccess) {

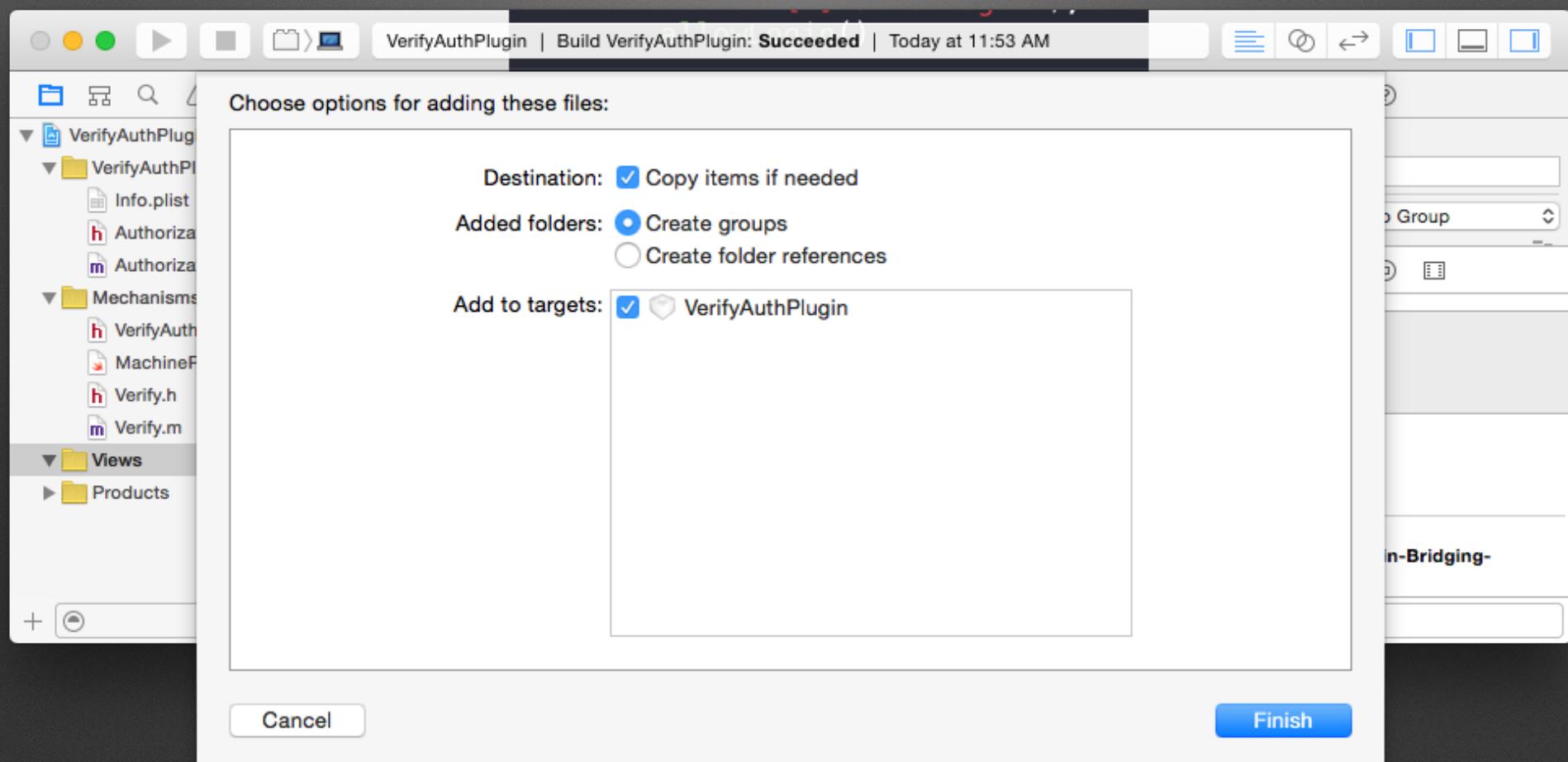
        NSData *pinData = [[NSData alloc] initWithBytes:value->data length:value->length];
        id ret = [NSKeyedUnarchiver unarchiveObjectWithData:pinData];
        pin = (NSString *)ret;

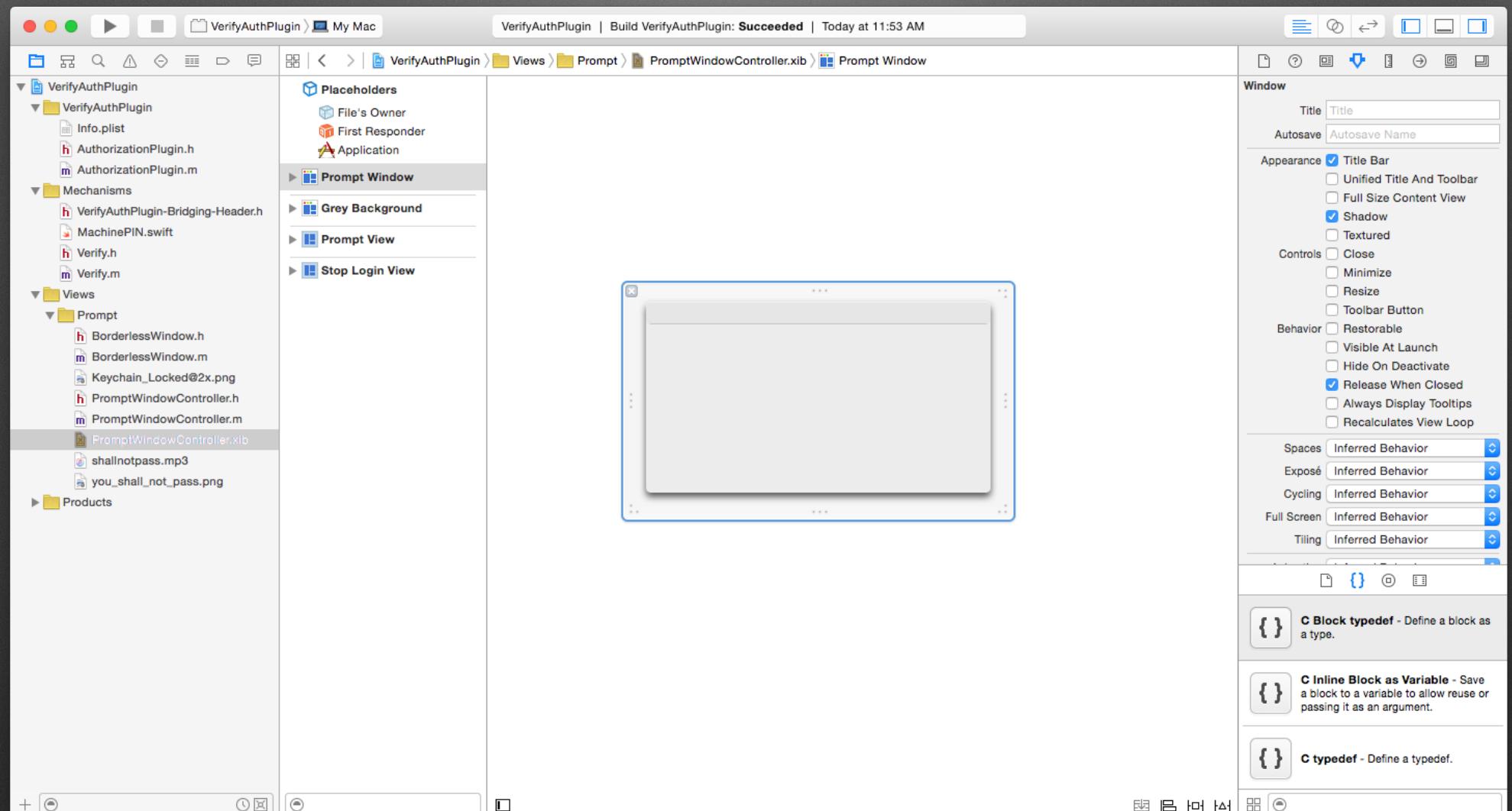
    } else {
        NSLog(@"[!] Failed to read %@", contextPINDomain);
    }

    return pin;
}
```

Code:Verify

Add : theView





VerifyAuthPlugin : Goals

```
<done>Create 2 Mechanisms</done>
<done>Share Data between Mechanisms</done>
    <done>Prompt user for PIN</done>
<done>Make an authorization decision</done>
```

Plug : in

Verify.m

```
#import "PromptWindowController.h"  
// Create an instance of the PromptWindowController  
    &&
```

Code

AuthorizationPlugin.m

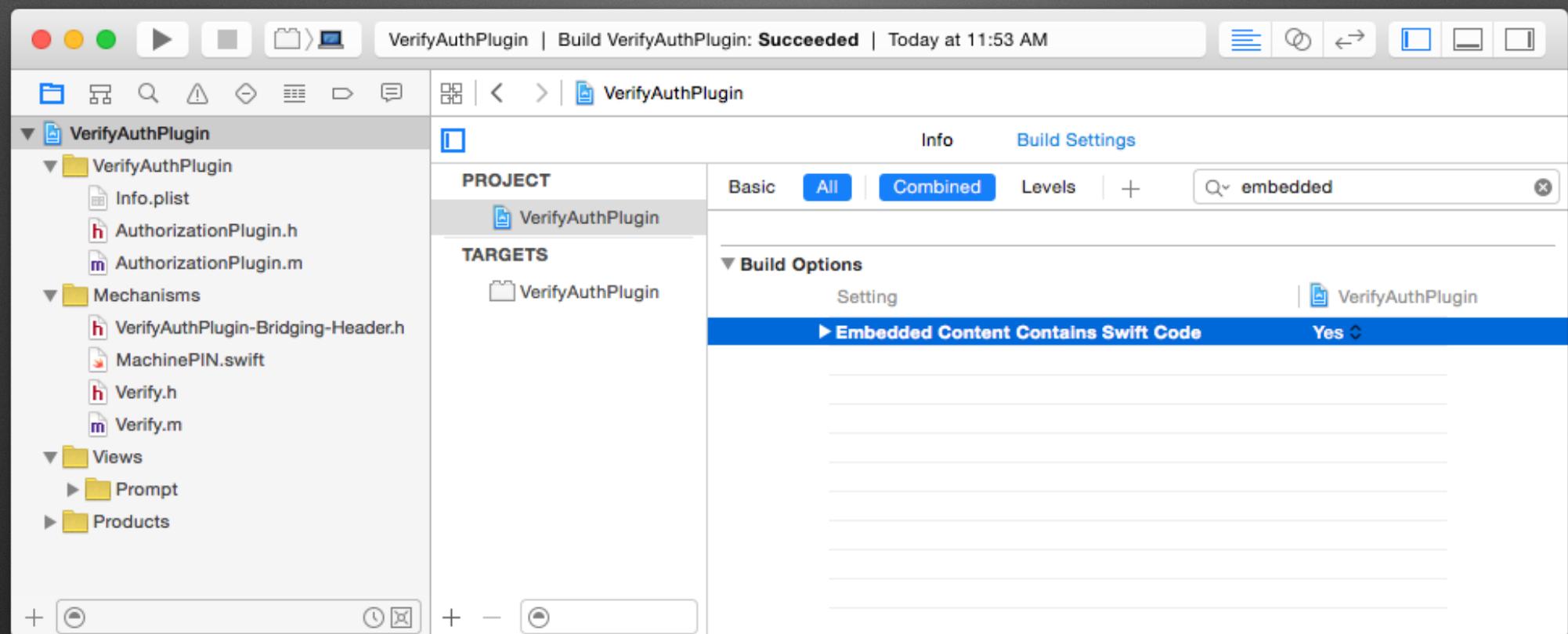
```
// Special auto-generated header. It makes the Swift
// classes available to ObjC
#import "VerifyAuthPlugin-Swift.h"

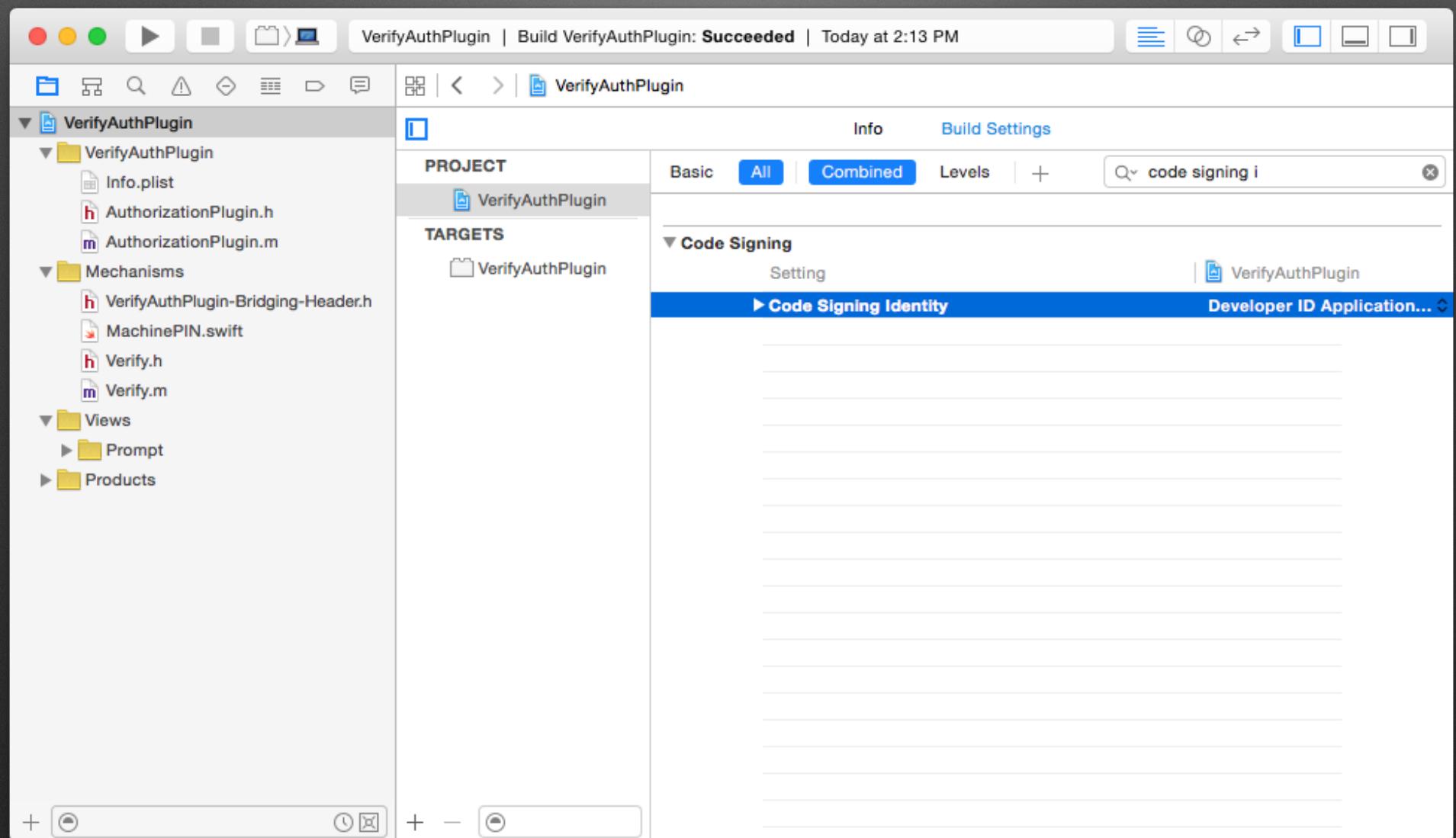
// Gains access to the ObjC Verify Mech
#import "Verify.h"

// Call the MachinePIN mechanism
// Call the Verify mechanism
```

Code

Don't Forget!







Build Succeeded

Install: Test

```
<step01>Copy:bundle</step01>
<step02>Add:authorizationdb</step02>
<step03>Keychain:com.burginsystems.pin</step03>
```

Lessons: Learned

- Privileged and non-privileged mechanisms run on different processes.
 - Use hints to share data between address spaces. Use objects to share data between like mechanisms.
- Check for null terminated strings within the context and hint values.
 - Different session callers fill the data inconsistently.