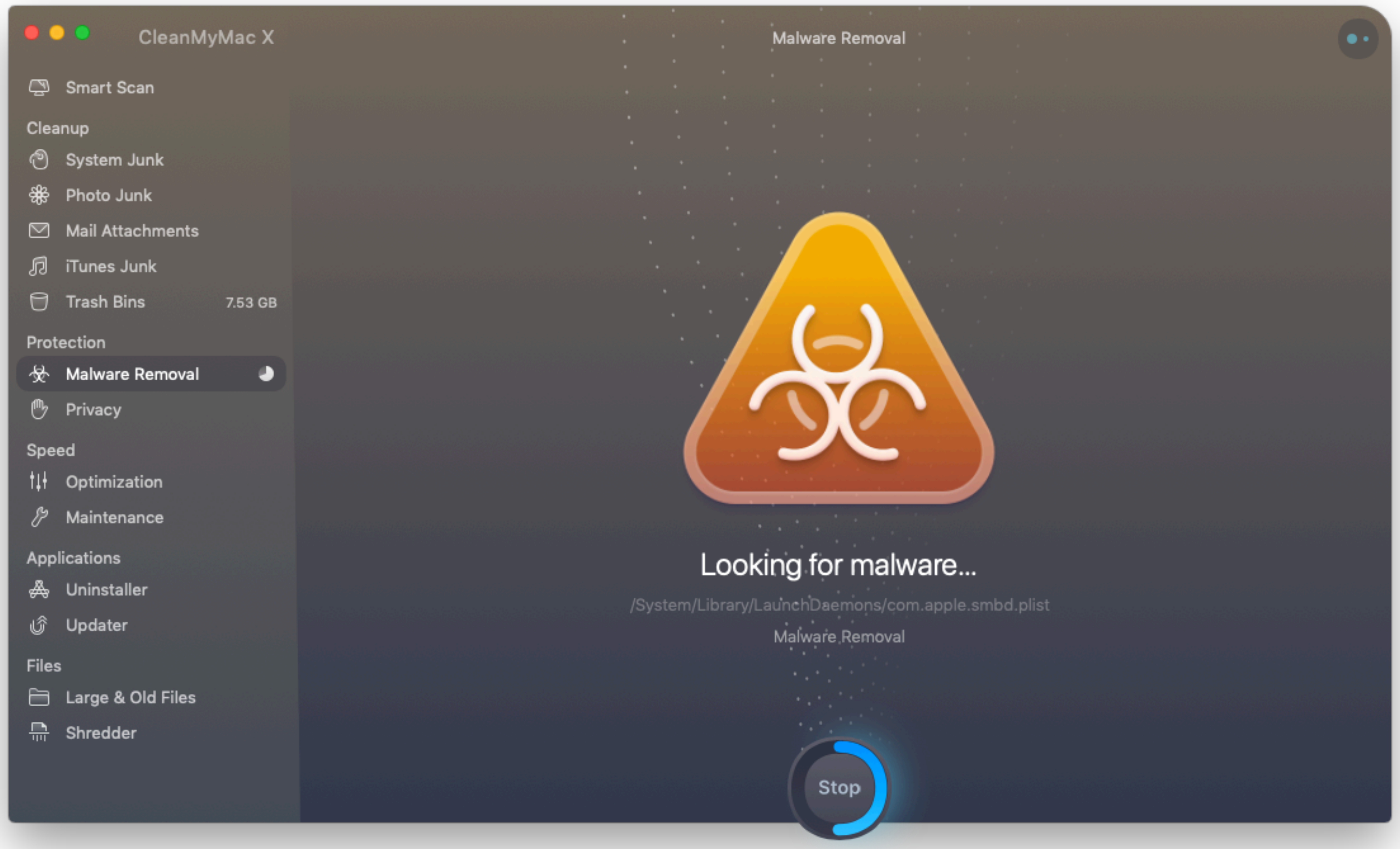
Job(s) Bless Us!
Privileged Operations on macOS

@aronskaya 🇺🇦

MacPaw Software Engineer, Anti-malware team, Triage team



WOMEN WHO CODE KYIV

WWC Kyiv macOS Chapter Lead

iaronskaya

Agenda

- Intro to privileged operations API on macOS
- First CleanMyMac's security issue, reported by Talos
- CleanMyMac on hackerone
- Comparison of privileged operations implementation on and SETAPP
- Summary & Takeaways

Intro to privileged operations API on macOS

High-level APIs

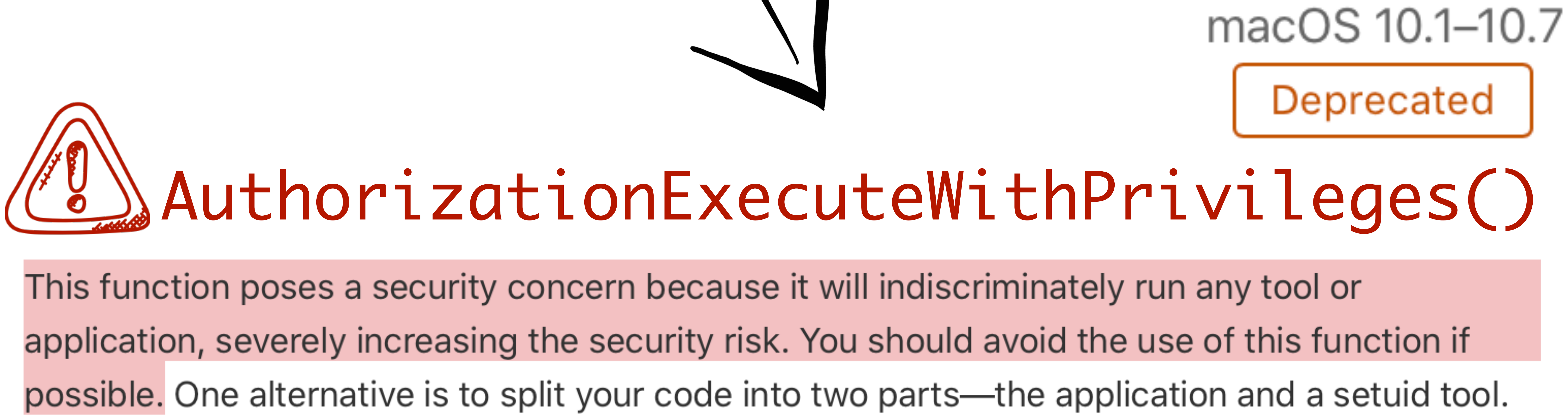
SMJobBless()

AuthorizationExecuteWithPrivileges()

High-level APIs

SMJobBless()

AuthorizationExecuteWithPrivileges()



macOS 10.1–10.7

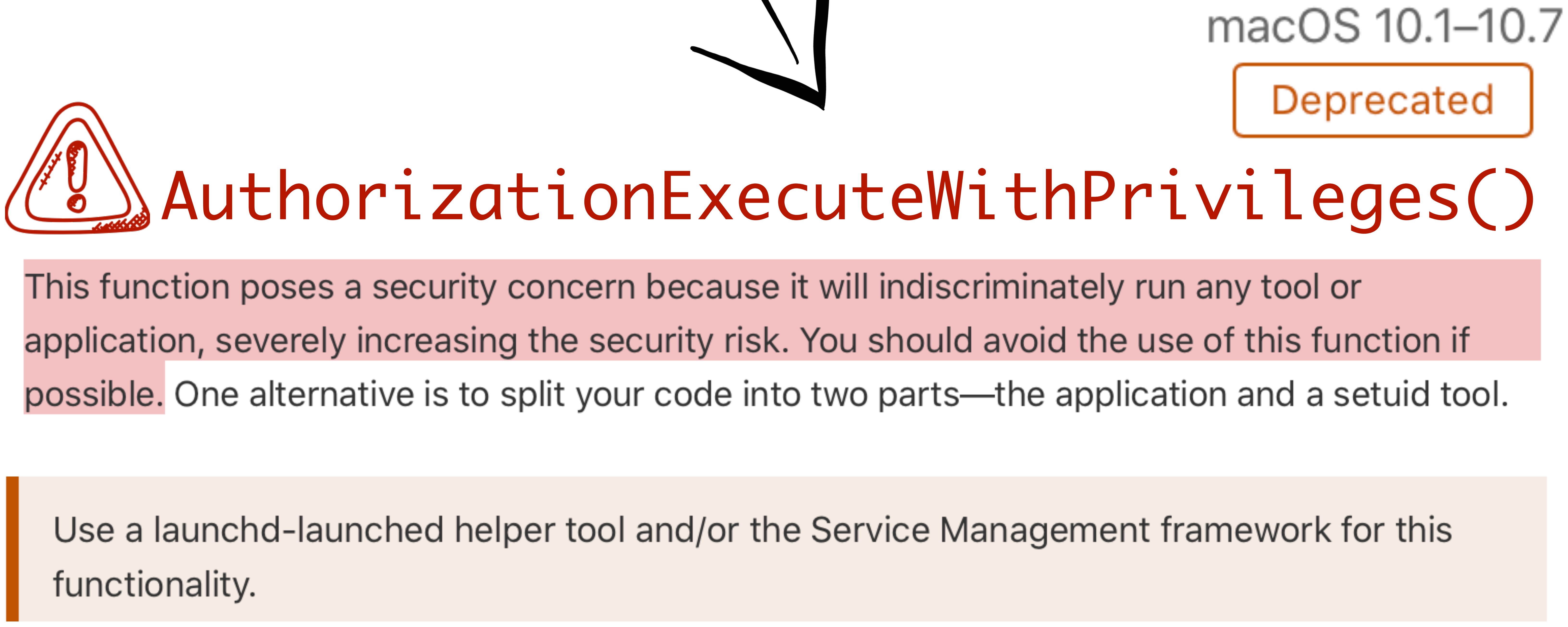
Deprecated

This function poses a security concern because it will indiscriminately run any tool or application, severely increasing the security risk. You should avoid the use of this function if possible. One alternative is to split your code into two parts—the application and a setuid tool.

High-level APIs

SMJobBless()

AuthorizationExecuteWithPrivileges()



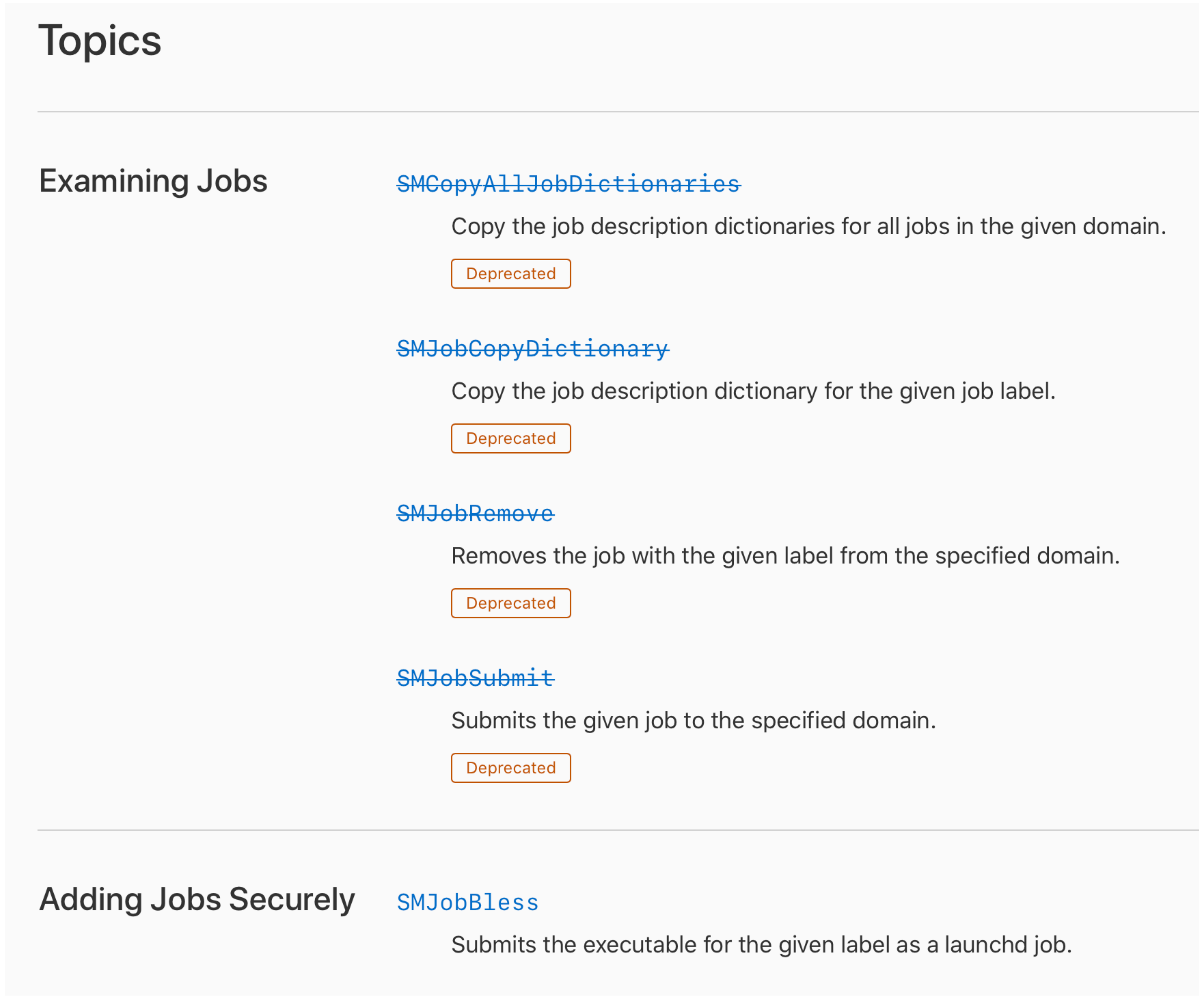
macOS 10.1–10.7

Deprecated

This function poses a security concern because it will indiscriminately run any tool or application, severely increasing the security risk. You should avoid the use of this function if possible. One alternative is to split your code into two parts—the application and a setuid tool.

Use a launchd-launched helper tool and/or the Service Management framework for this functionality.

There is no 'UnBless' 😔



Topics

Examining Jobs

~~SMCopyAllJobDictionaries~~

Copy the job description dictionaries for all jobs in the given domain.

Deprecated

~~SMJobCopyDictionary~~

Copy the job description dictionary for the given job label.

Deprecated

~~SMJobRemove~~

Removes the job with the given label from the specified domain.

Deprecated

~~SMJobSubmit~~

Submits the given job to the specified domain.

Deprecated

Adding Jobs Securely

SMJobBless

Submits the executable for the given label as a launchd job.

Signing requirements

Client

Client has requirements for Helper(s)

$\longrightarrow$

$\longleftarrow$

Helper has requirements for Client(s)

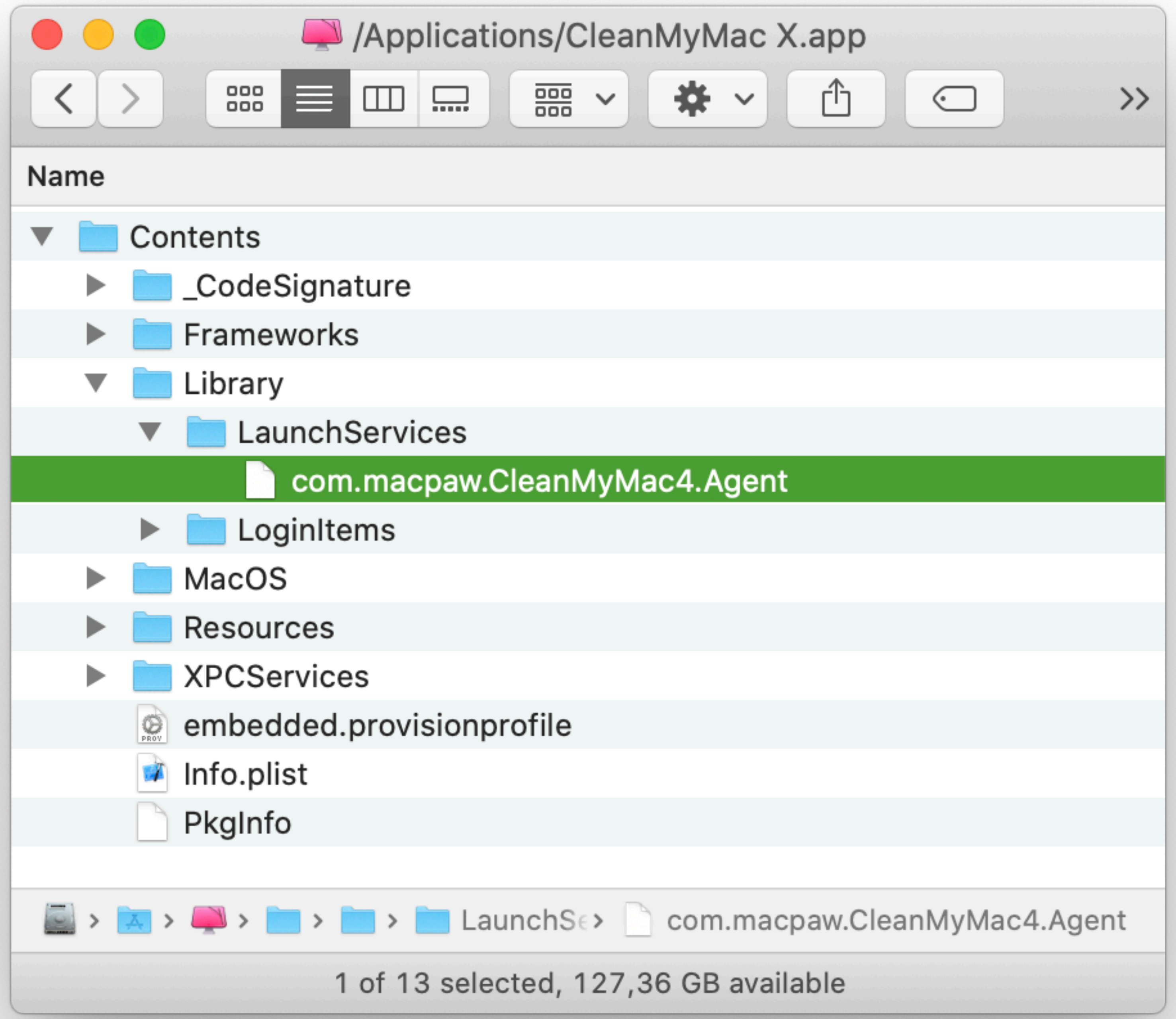
Privileged Helper

OS performs validation of the requirements ONLY on install & update of the Helper

⚠ NO validation is performed on establishing XPC connection

SMJobBless()

1. Client has the Privileged Helper executable in the bundle



/Applications/CleanMyMac X.app

Name

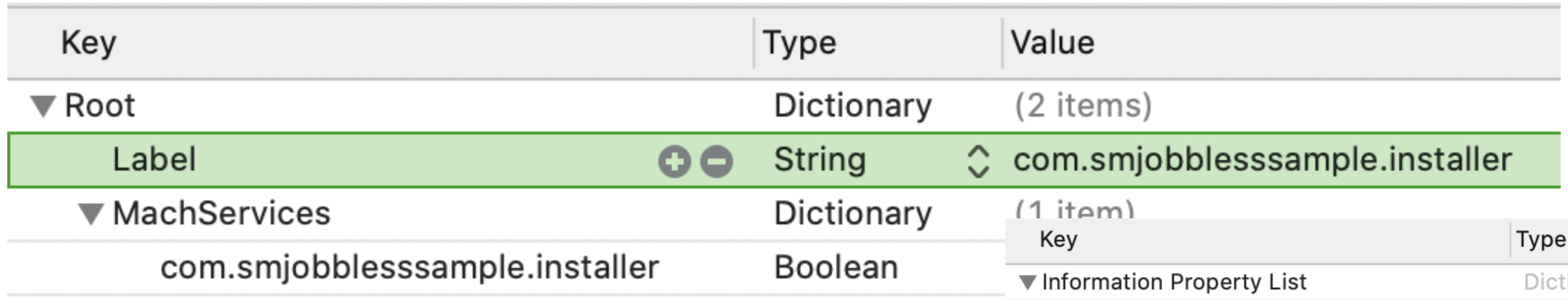
- Contents
  - _CodeSignature
  - Frameworks
  - Library
    - LaunchServices
      - com.macpaw.CleanMyMac4.Agent
    - LoginItems
  - MacOS
  - Resources
  - XPCServices
  - embedded.provisionprofile
  - Info.plist
  - PkgInfo

LaunchS› com.macpaw.CleanMyMac4.Agent

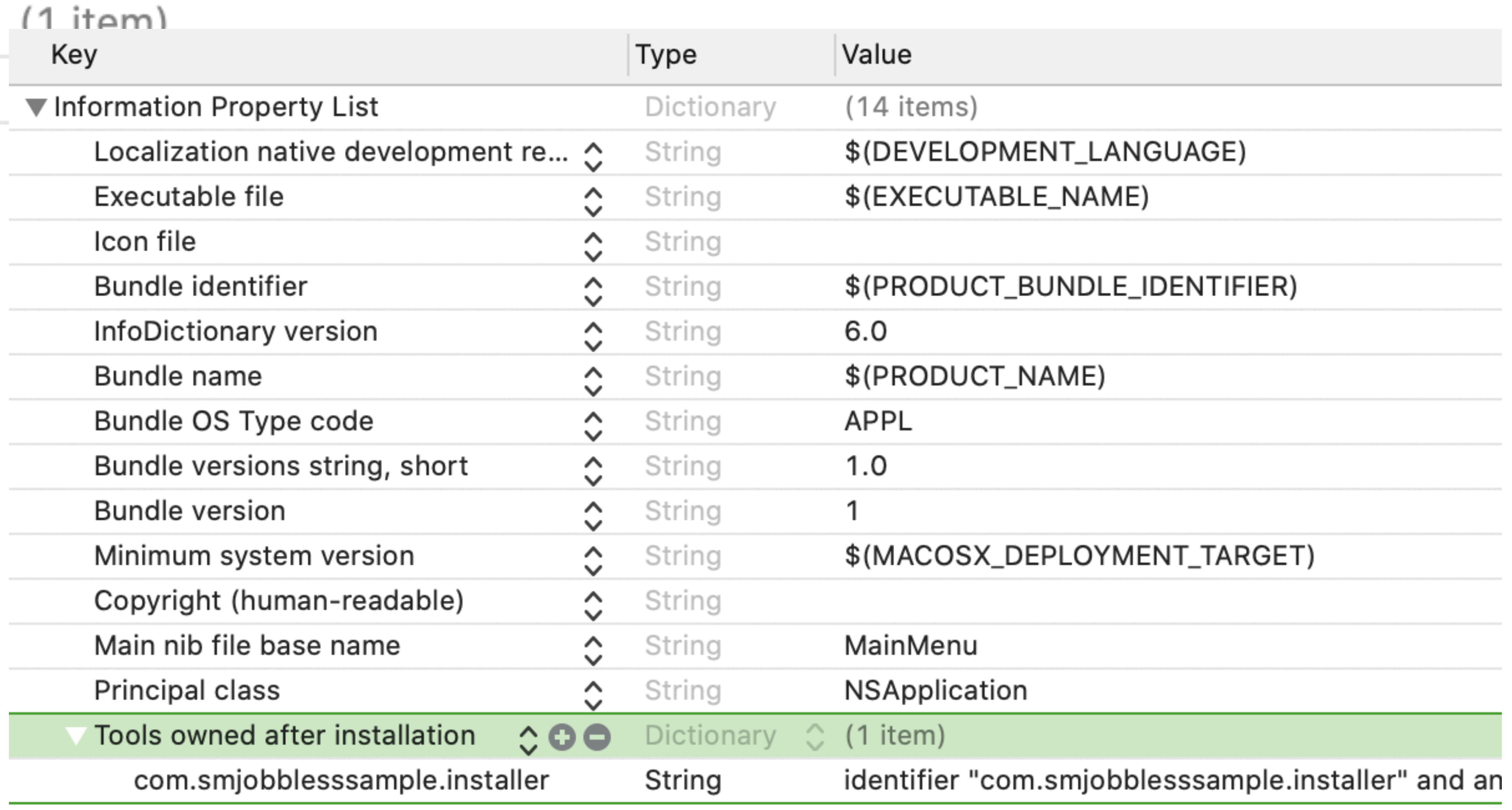
1 of 13 selected, 127,36 GB available

2. Signing requirements are met

- Both client and Helper are signed
- Privileged Helper has a plist file for launchd embedded into __TEXT section
- Privileged Helper has Info.plist embedded
- Client has signing requirements listed in its Info.plist



Key	Type	Value
▼ Root	Dictionary	(2 items)
Label	String	com.smjobblesssample.installer
▼ MachServices	Dictionary	(1 item)
com.smjobblesssample.installer	Boolean	

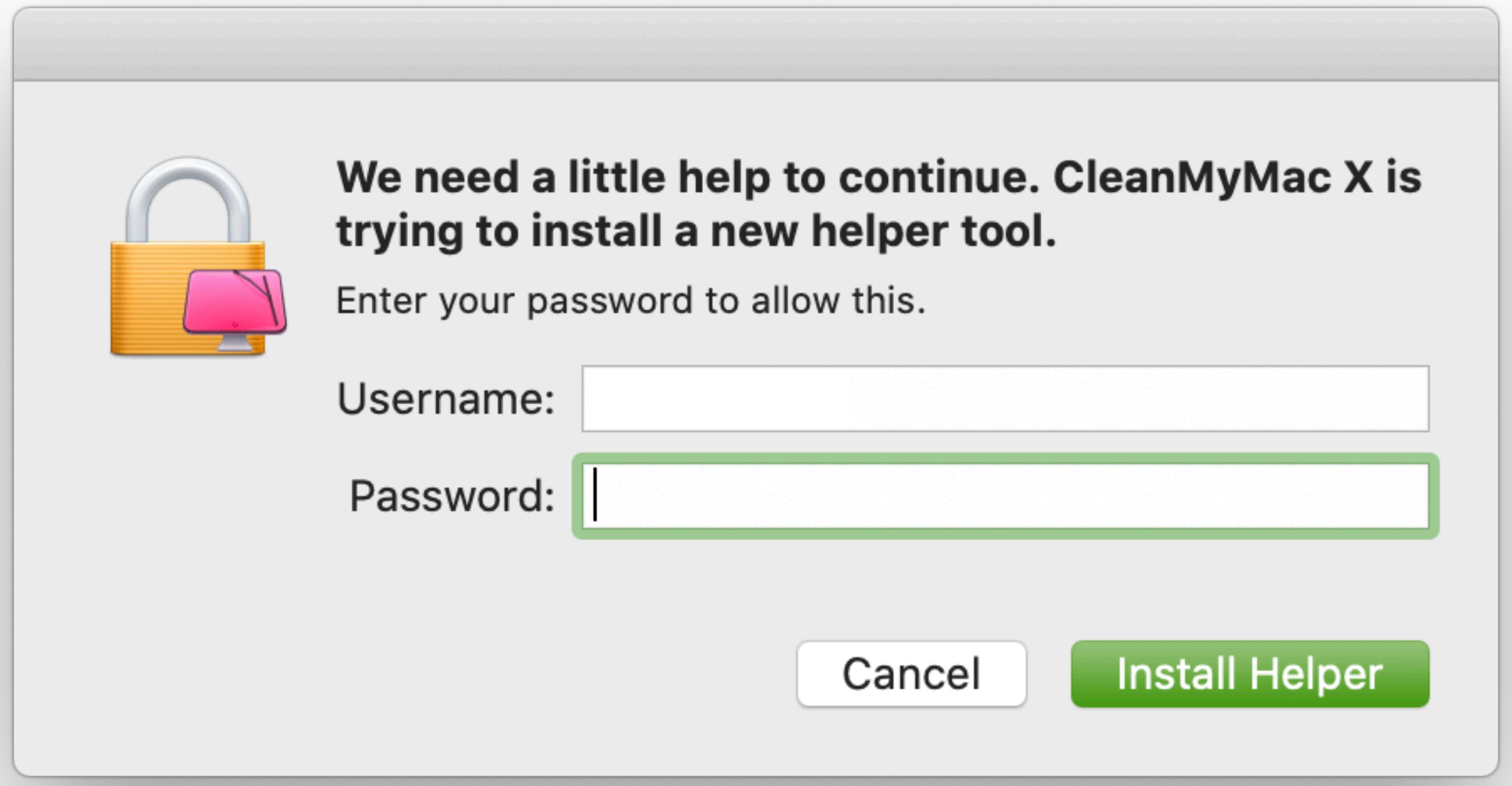


Key	Type	Value
▼ Information Property List	Dictionary	(14 items)
Localization native development re...	String	$(DEVELOPMENT_LANGUAGE)
Executable file	String	$(EXECUTABLE_NAME)
Icon file	String	
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle version	String	1
Minimum system version	String	$(MACOSX_DEPLOYMENT_TARGET)
Copyright (human-readable)	String	
Main nib file base name	String	MainMenu
Principal class	String	NSApplication
▼ Tools owned after installation	Dictionary	(1 item)
com.smjobblesssample.installer	String	identifier "com.smjobblesssample.installer" and an

SMJobBless()

3. Obtain Authorization object:
call AuthorizationCreate()

```
    const AuthorizationRights *kNoRightsSpecified = NULL;
    AuthorizationFlags flags          =    kAuthorizationFlagDefaults            |
                                          kAuthorizationFlagInteractionAllowed    |
                                          kAuthorizationFlagPreAuthorize          |
                                          kAuthorizationFlagExtendRights;

    self.lastErrorCode = AuthorizationCreate(kNoRightsSpecified,
        kAuthorizationEmptyEnvironment, flags, &_authRef);
```


We need a little help to continue. CleanMyMac X is trying to install a new helper tool.

Enter your password to allow this.

Username:

Password:

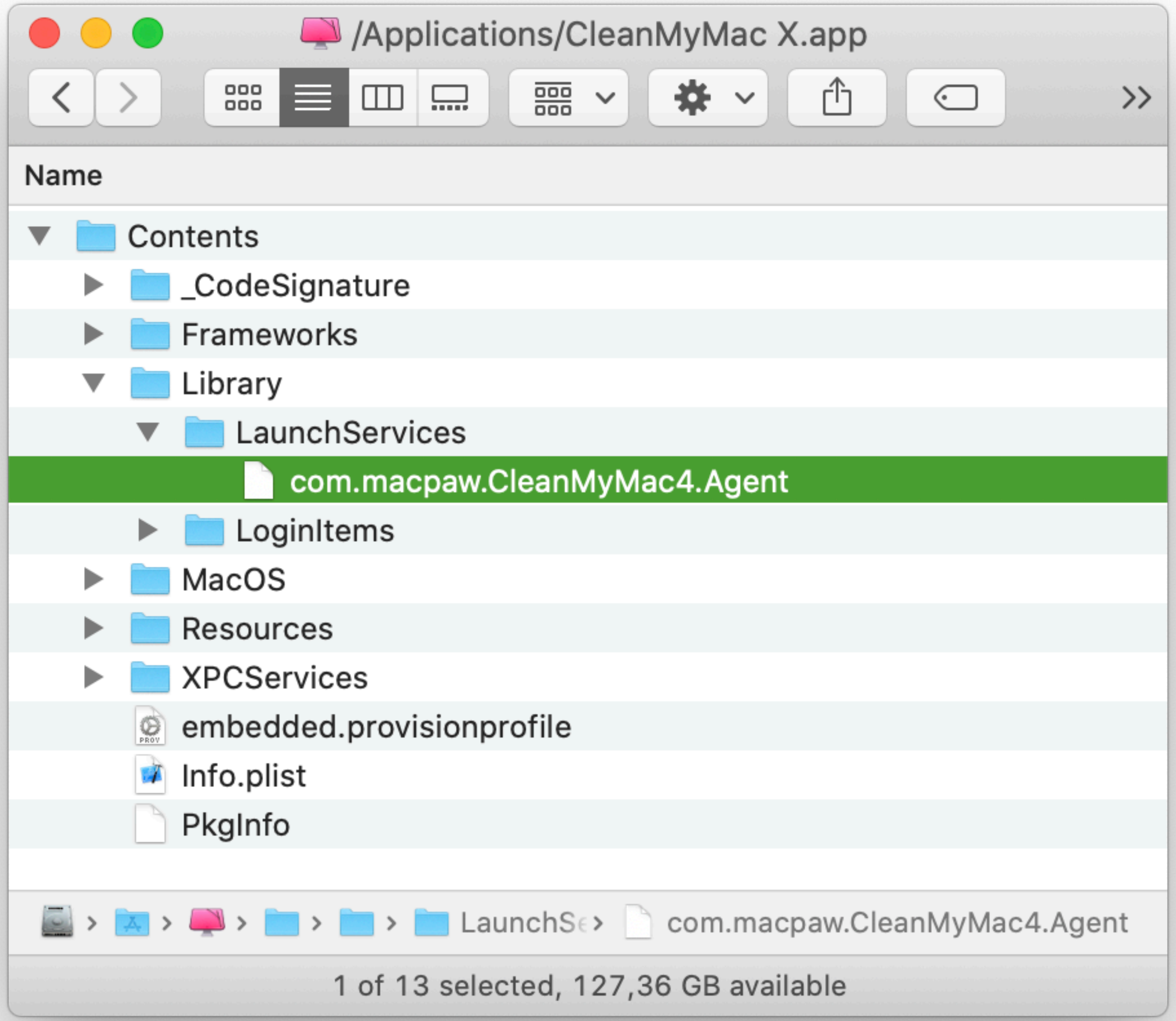
Cancel | Install Helper

4. Call SMJobBless() with acquired
Authorization object

```
Boolean res = SMJobBless(kSMDomainSystemLaunchd,
                         label,
                         authRef,
                         &error);
```

SMJobBless()

5. OS validates code signing requirements in client and helper's Info.plist and copies the executable from the bundle to **/Library/PrivilegedHelperTools**

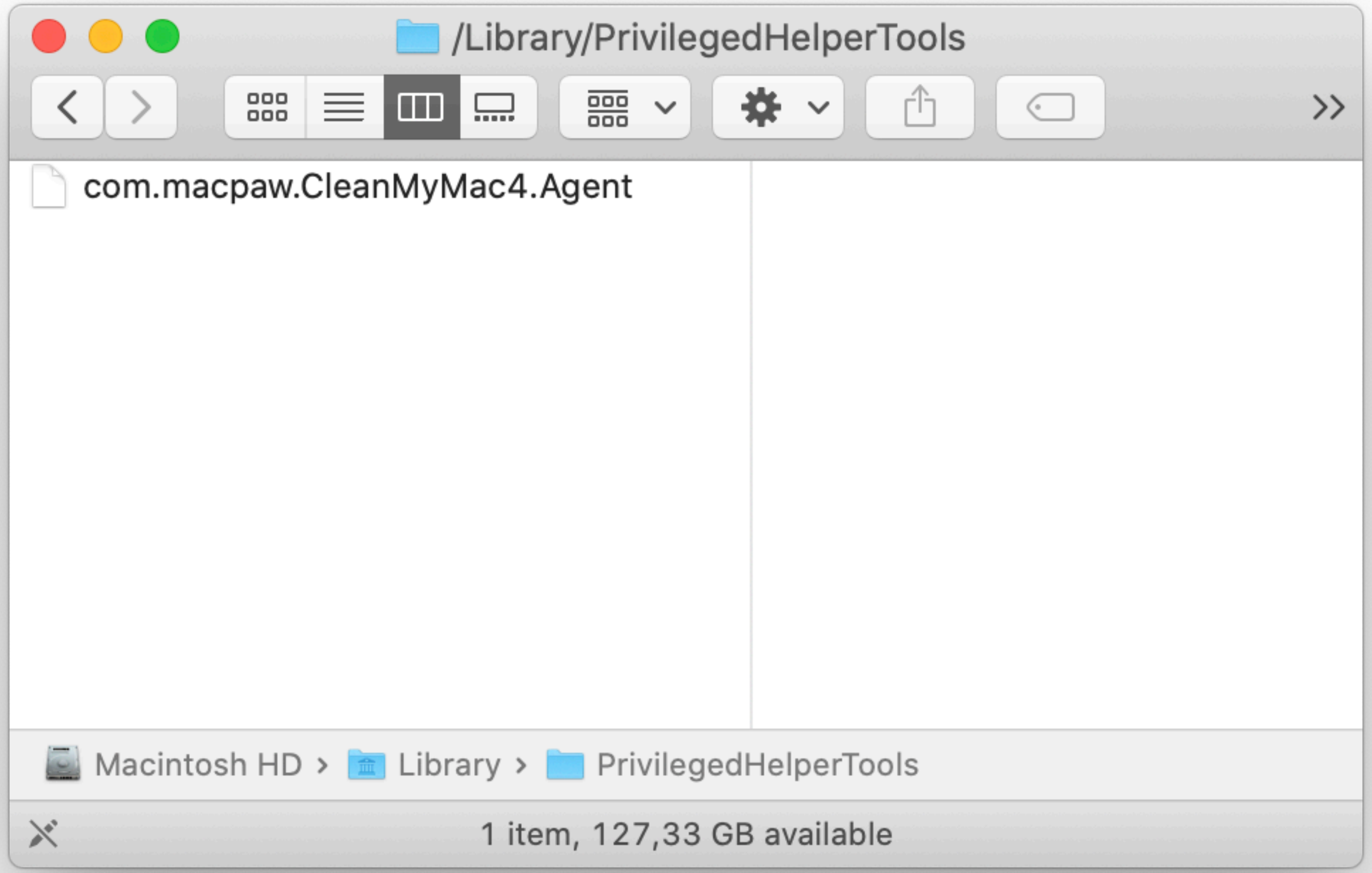


/Applications/CleanMyMac X.app

```
Name
▼ Contents
  ▶ _CodeSignature
  ▶ Frameworks
  ▼ Library
    ▼ LaunchServices
        com.macpaw.CleanMyMac4.Agent
    ▶ LoginItems
  ▶ MacOS
  ▶ Resources
  ▶ XPCServices
    embedded.provisionprofile
    Info.plist
    PkgInfo
```

› › › › › LaunchSe› com.macpaw.CleanMyMac4.Agent

1 of 13 selected, 127,36 GB available



/Library/PrivilegedHelperTools

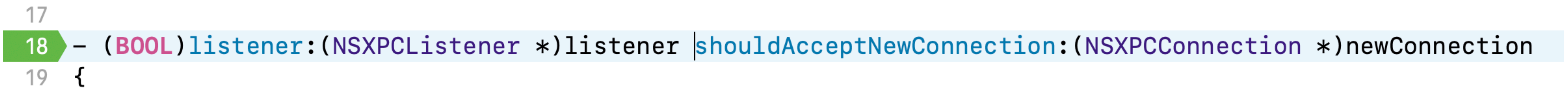
com.macpaw.CleanMyMac4.Agent

Macintosh HD › Library › PrivilegedHelperTools

1 item, 127,33 GB available

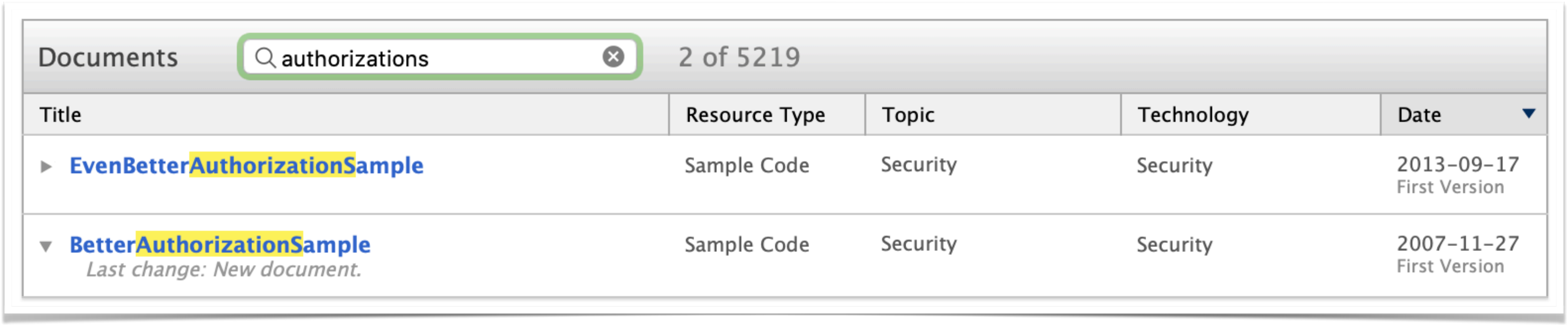
SMJobBless()

5. Client can establish XPC connection to the Privileged Helper



```
- (BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection *)newConnection
{
```

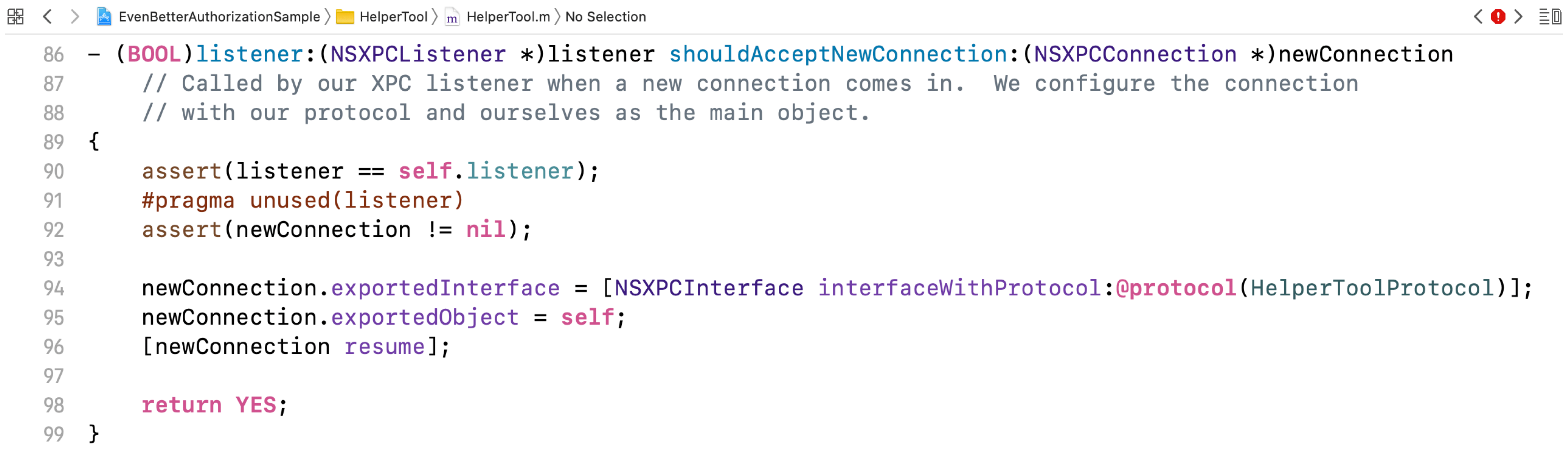
Apple's Sample Code



Documents authorizations 2 of 5219

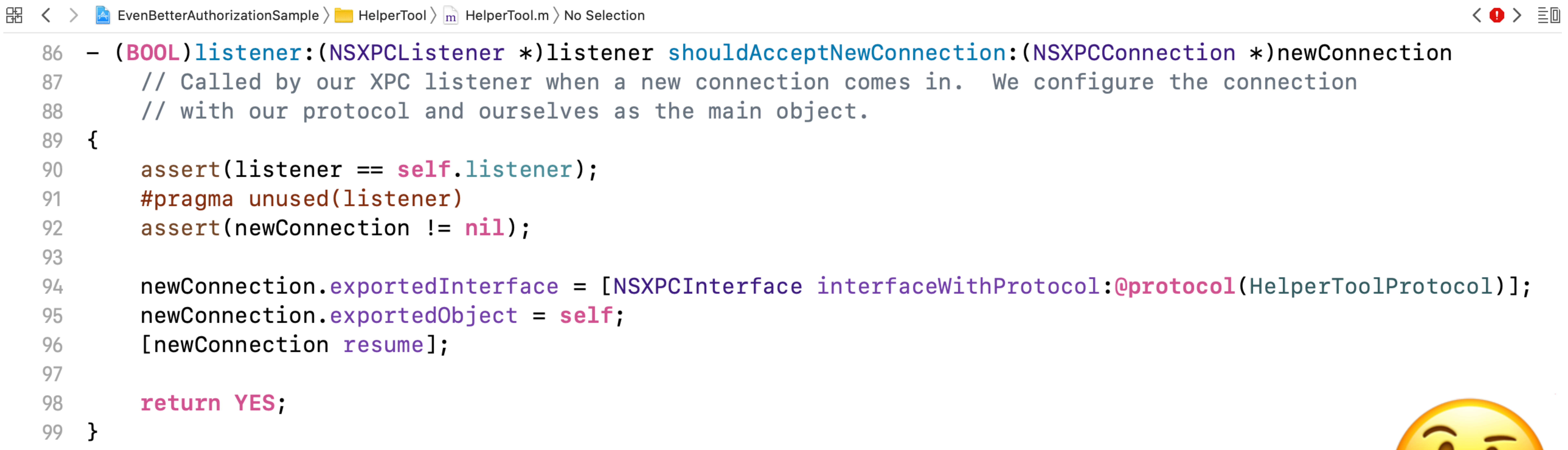
Title	Resource Type	Topic	Technology	Date
▸ EvenBetterAuthorizationSample	Sample Code	Security	Security	2013-09-17 First Version
▾ BetterAuthorizationSample *Last change: New document.*	Sample Code	Security	Security	2007-11-27 First Version

Apple's Sample Code


EvenBetterAuthorizationSample 〉 HelperTool 〉 HelperTool.m 〉 No Selection

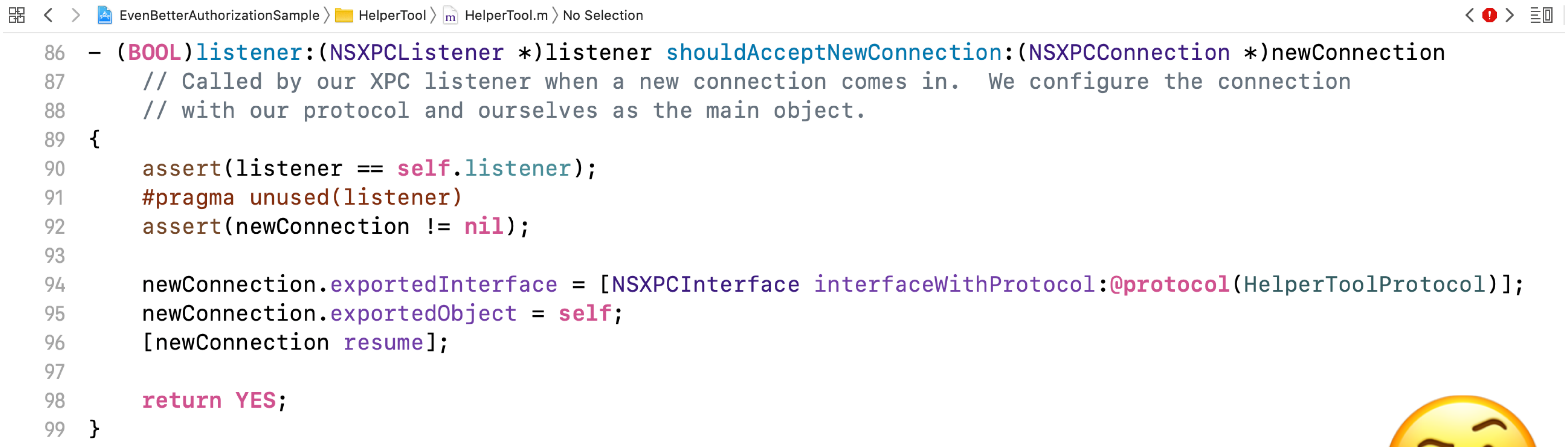
```
- (BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection *)newConnection
    // Called by our XPC listener when a new connection comes in.  We configure the connection
    // with our protocol and ourselves as the main object.
{
    assert(listener == self.listener);
    #pragma unused(listener)
    assert(newConnection != nil);

    newConnection.exportedInterface = [NSXPCInterface interfaceWithProtocol:@protocol(HelperToolProtocol)];
    newConnection.exportedObject = self;
    [newConnection resume];

    return YES;
}
```

Apple’s Sample Code


EvenBetterAuthorizationSample 〉 HelperTool 〉 HelperTool.m 〉 No Selection

```
- (BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection *)newConnection
    // Called by our XPC listener when a new connection comes in.  We configure the connection
    // with our protocol and ourselves as the main object.
{
    assert(listener == self.listener);
    #pragma unused(listener)
    assert(newConnection != nil);

    newConnection.exportedInterface = [NSXPCInterface interfaceWithProtocol:@protocol(HelperToolProtocol)];
    newConnection.exportedObject = self;
    [newConnection resume];

    return YES;
}
```

Issue #1

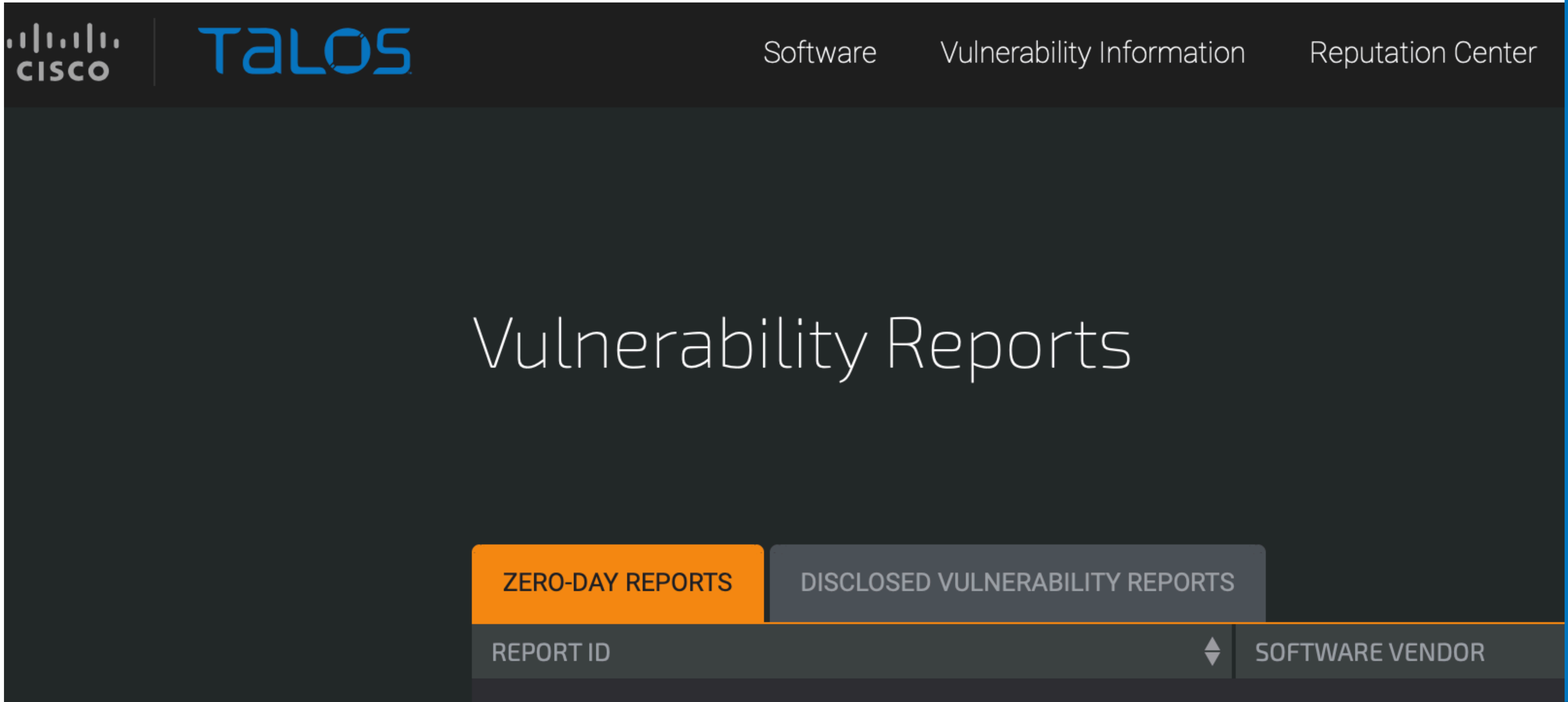

EvenBetterAuthorizationSample 〉 HelperTool 〉 HelperTool.m 〉 No Selection

```
- (BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection *)newConnection
    // Called by our XPC listener when a new connection comes in.  We configure the connection
    // with our protocol and ourselves as the main object.
{
    assert(listener == self.listener);
    #pragma unused(listener)
    assert(newConnection != nil);

    newConnection.exportedInterface = [NSXPCInterface interfaceWithProtocol:@protocol(HelperToolProtocol)];
    newConnection.exportedObject = self;
    [newConnection resume];

    return YES;
}
```

First security issue, reported by TALOS

Zero-Day Reports

- November 2018



Talos

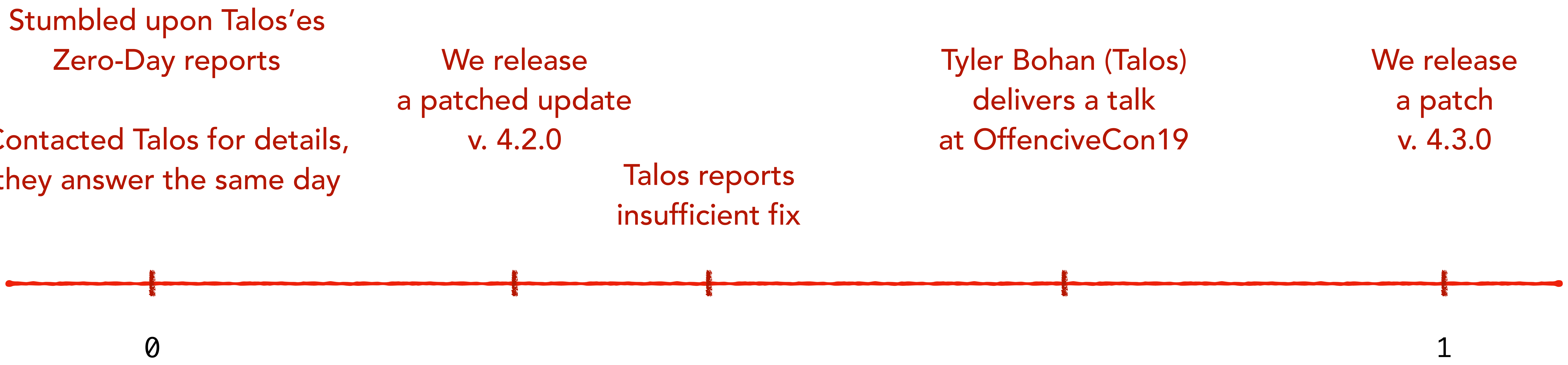
ZERO-DAY REPORTS | DISCLOSED VULNERABILITY REPORTS

CleanMyMac

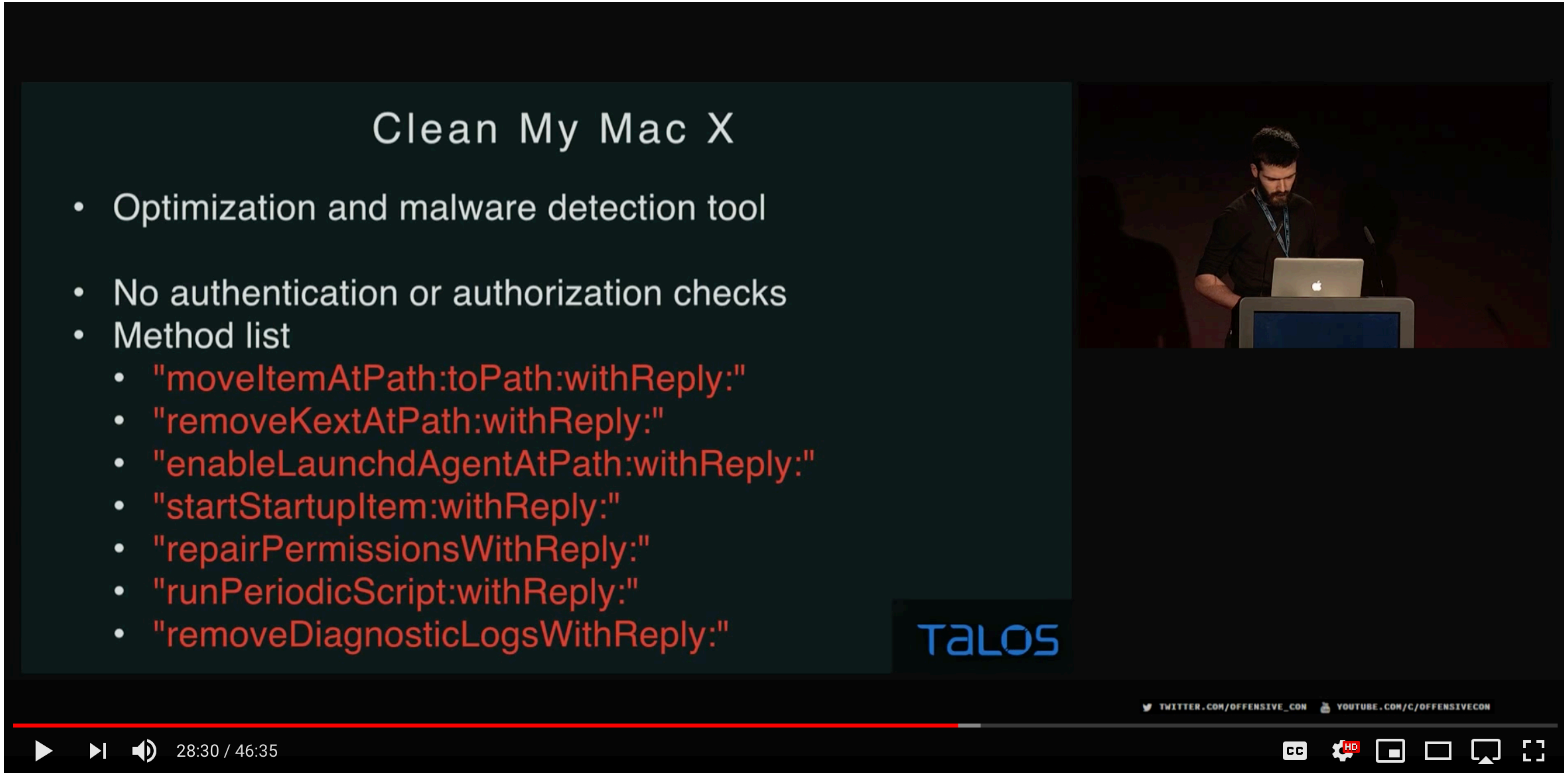
REPORT ID	TITLE	REPORT DATE	CVE NUMBER	CVSS SCORE
TALOS-2019-0759	CleanMyMac X incomplete update patch privilege escalation vulnerability	2019-03-11	CVE-2019-5011	7.1
TALOS-2018-0705	CleanMyMac X moveItemAtPath privilege escalation vulnerability	2019-01-02	CVE-2018-4032	7.1
TALOS-2018-0708	CleanMyMac X truncateFileAtPath Privilege Escalation Vulnerability	2019-01-02	CVE-2018-4035	7.1
TALOS-2018-0707	CleanMyMac X removeItemAtPath Privilege Escalation Vulnerability	2019-01-02	CVE-2018-4034	7.1
TALOS-2018-0706	CleanMyMac X moveToTrashItemAtPath privilege escalation vulnerability	2019-01-02	CVE-2018-4033	7.1
TALOS-2018-0709	CleanMyMac X removeKextAtPath privilege escalation vulnerability	2019-01-02	CVE-2018-4036	7.1
TALOS-2018-0710	CleanMyMac X removeDiagnosticLogs privilege escalation vulnerability	2019-01-02	CVE-2018-4037	7.1

Showing 1 to 7 of 7 results

Timeline


Stumbled upon Talos'es Zero-Day reports
Contacted Talos for details, they answer the same day
We release a patched update v. 4.2.0
Talos reports insufficient fix
Tyler Bohan (Talos) delivers a talk at OffenciveCon19
We release a patch v. 4.3.0
0
1

Tyler Bohan: 'OSX XPC Revisited - 3rd Party Application Flaws' at OffensiveCon19

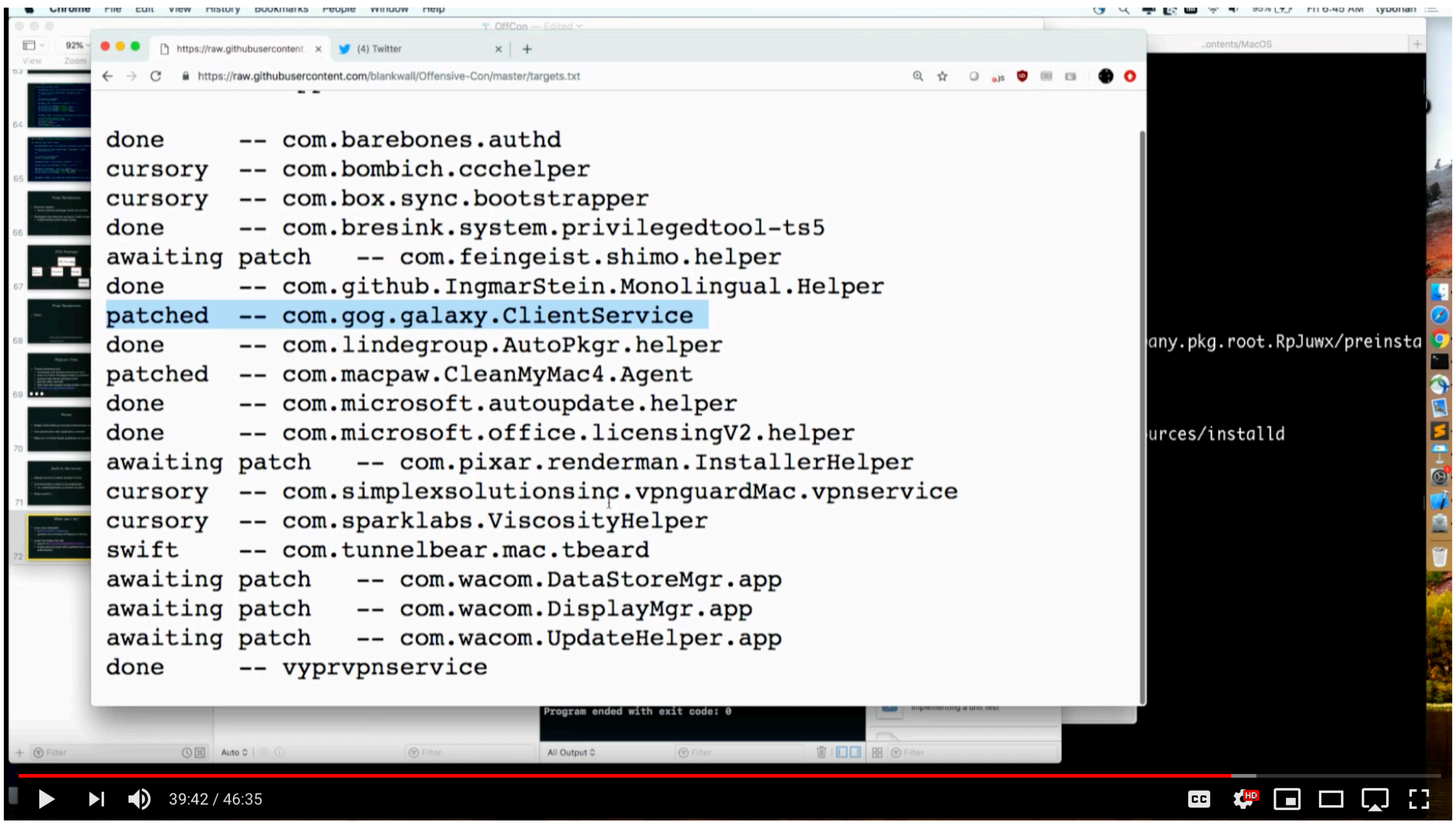


Clean My Mac X

- Optimization and malware detection tool
- No authentication or authorization checks
- Method list
  - "moveItemAtPath:toPath:withReply:"
  - "removeKextAtPath:withReply:"
  - "enableLaunchdAgentAtPath:withReply:"
  - "startStartupItem:withReply:"
  - "repairPermissionsWithReply:"
  - "runPeriodicScript:withReply:"
  - "removeDiagnosticLogsWithReply:"

https://www.youtube.com/watch?v=KPzhTqwf0bA

Tyler Bohan: ‘OSX XPC Revisited - 3rd Party Application Flaws’ at OffensiveCon19



```
done      -- com.barebones.authd
cursory   -- com.bombich.ccchelper
cursory   -- com.box.sync.bootstrapper
done      -- com.bresink.system.privilegedtool-ts5
awaiting patch   -- com.feingeist.shimo.helper
done      -- com.github.IngmarStein.Monolingual.Helper
patched   -- com.gog.galaxy.ClientService
done      -- com.lindegroup.AutoPkgr.helper
patched   -- com.macpaw.CleanMyMac4.Agent
done      -- com.microsoft.autoupdate.helper
done      -- com.microsoft.office.licensingV2.helper
awaiting patch   -- com.pixar.renderman.InstallerHelper
cursory   -- com.simplexsolutionsinc.vpnguardMac.vpnservice
cursory   -- com.sparklabs.ViscosityHelper
swift     -- com.tunnelbear.mac.tbeard
awaiting patch   -- com.wacom.DataStoreMgr.app
awaiting patch   -- com.wacom.DisplayMgr.app
awaiting patch   -- com.wacom.UpdateHelper.app
done      -- vyprvpnservice
```

https://raw.githubusercontent.com/blankwall/Offensive-Con/master/targets.txt
(4) Twitter
any.pkg.root.RpJuwx/preinsta
urces/installd
Program ended with exit code: 0
39:42 / 46:35

https://www.youtube.com/watch?v=KPzhTqwf0bA

Fix

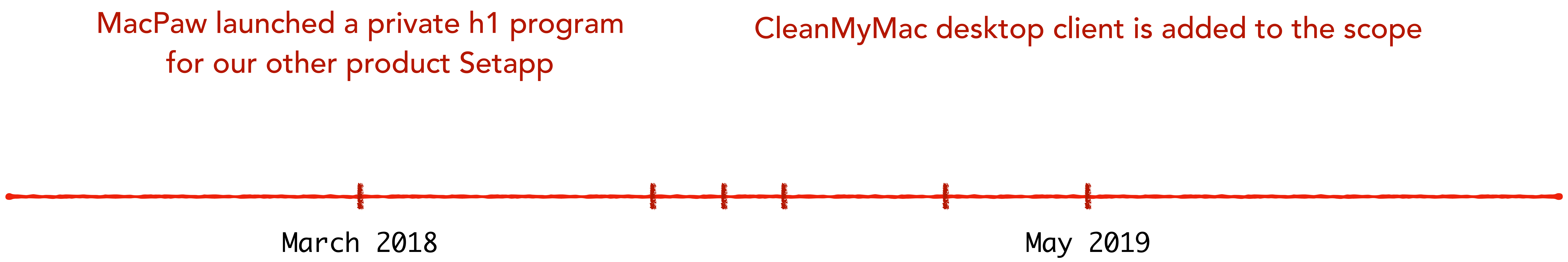
```objc
- (BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection *)newConnection
{
    CFErrorRef errors = NULL;
    SecCodeRef code = NULL;
    SecRequirementRef requirement = NULL;

    NSDictionary *attributes = @{ (__bridge NSString *)kSecGuestAttributePid: @(connection.processIdentifier) };
    SecCodeCopyGuestWithAttributes(0, (__bridge CFDictionaryRef)attributes, kSecCSDefaultFlags, &code);

    NSString *entitlement = @"anchor trusted and certificate leaf [subject.CN] = \
    \"Developer ID Application: MacPaw Inc. (AAAAAAAAAA)\"";

    SecRequirementCreateWithStringAndErrors((__bridge CFStringRef)entitlement, kSecCSDefaultFlags, &errors, &requirement);

    OSStatus status = SecCodeCheckValidity(code, kSecCSDefaultFlags, requirement);

    if (errSecSuccess != status)
    {
        return NO;
    }
```

Fix #1

```objc
- (BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection
    *)newConnection
{
    CFErrorRef errors = NULL;
    SecCodeRef code = NULL;
    SecRequirementRef requirement = NULL;

    NSDictionary *attributes = @{ (__bridge NSString *)kSecGuestAttributePid:
        @(connection.processIdentifier) };
    SecCodeCopyGuestWithAttributes(0, (__bridge CFDictionaryRef)attributes,
        kSecCSDefaultFlags, &code);

    NSString *entitlement = @"anchor trusted and certificate leaf [subject.CN] = \
    \"Developer ID Application: MacPaw Inc. (AAAAAAAAAA)\"";

    SecRequirementCreateWithStringAndErrors((__bridge CFStringRef)entitlement,
        kSecCSDefaultFlags, &errors, &requirement);

    OSStatus status = SecCodeCheckValidity(code, kSecCSDefaultFlags, requirement);

    if (errSecSuccess != status)
    {
        return NO;
    }
```

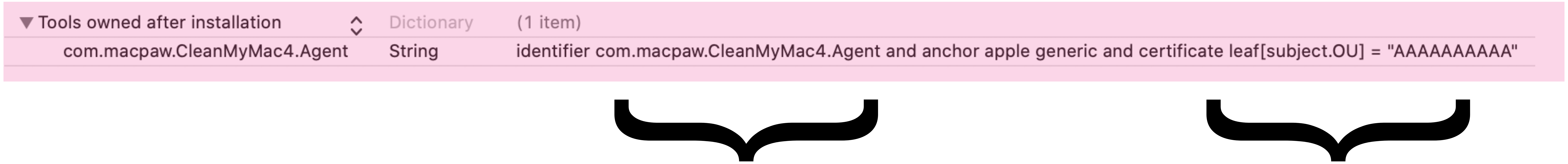
on hackerone

Timeline


MacPaw launched a private h1 program for our other product Setapp
CleanMyMac desktop client is added to the scope
March 2018
May 2019

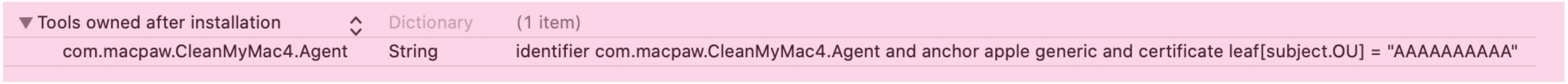
h1

Client's requirements



▼ Tools owned after installation	Dictionary	(1 item)
com.macpaw.CleanMyMac4.Agent	String	identifier com.macpaw.CleanMyMac4.Agent and anchor apple generic and certificate leaf[subject.OU] = "AAAAAAAAAA"

Client's requirements

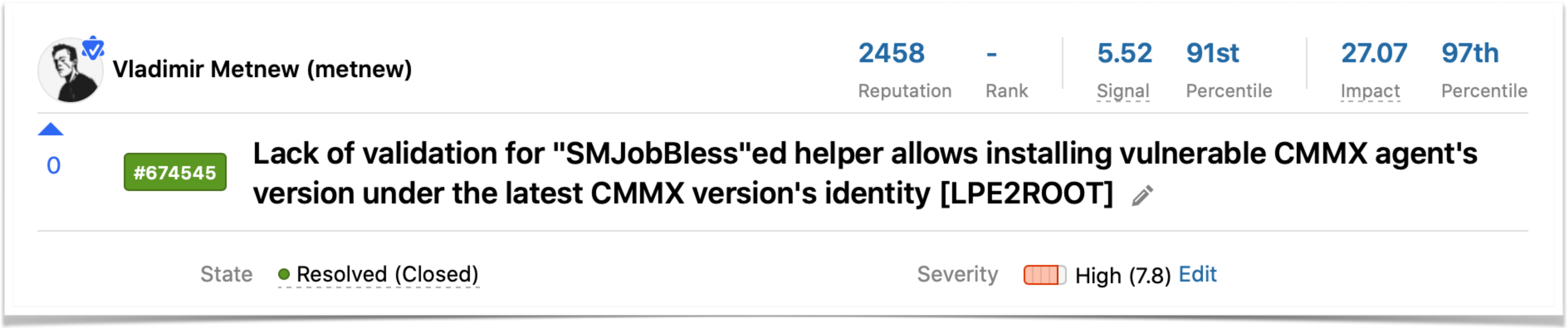


Tools owned after installation	Dictionary	(1 item)
com.macpaw.CleanMyMac4.Agent	String	identifier com.macpaw.CleanMyMac4.Agent and anchor apple generic and certificate leaf[subject.OU] = "AAAAAAAAAA"

Bundle identifier

Signing identity (team id)

Privileged helper's executable can be replaced with old version



Vladimir Metnew (metnew)

2458	-	5.52	91st	27.07	97th
Reputation	Rank	Signal	Percentile	Impact	Percentile

0 #674545 Lack of validation for "SMJobBless"ed helper allows installing vulnerable CMMX agent's version under the latest CMMX version's identity [LPE2ROOT]

State ● Resolved (Closed)

Severity High (7.8) Edit

h1

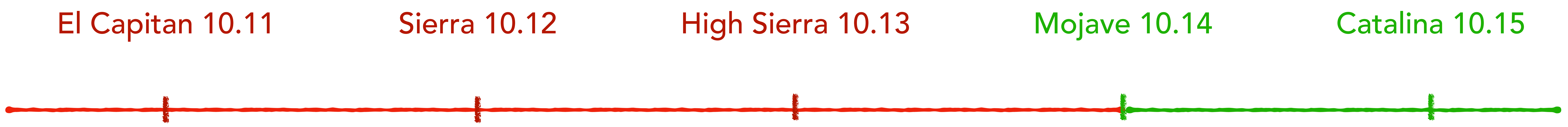
What’s the fuss about old versions?

What’s the fuss about old versions?

Hardened Runtime introduced in Mojave:

- libraries signing validation == protect from dylib injection
- remove get-task-allow from entitlements == protect from attaching with debugger

(and other things)



Issue #2: steps

Preconditions: Privileged Helper is not authorized yet. A malicious executable is present on the user's computer.

1. Download an app version, vulnerable to dylib injection
2. Replace the Privileged Helper executable in the installed app with the vulnerable one
3. User authorizes the Helper
4. Perform a dylib injection into the Helper—it is run as root!

What about code signing?

Replacing the Privileged Helper in the signed bundle doesn’t change anything, because

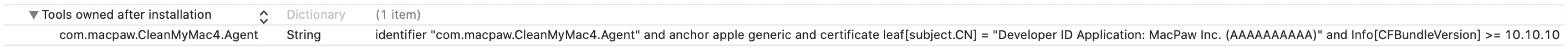
<u>OS validates the signature only when app is quarantined</u>

After the first launch <u>no signature validation</u> is performed on Mojave.

Time-to-time signature checks were announced in Catalina.

h1

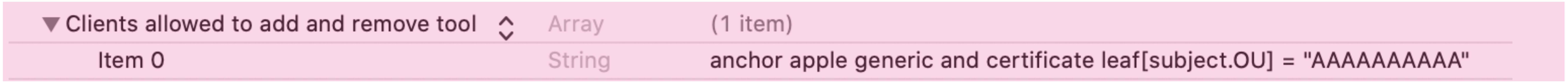
Fix #2



▼ Tools owned after installation	Dictionary	(1 item)
com.macpaw.CleanMyMac4.Agent	String	identifier "com.macpaw.CleanMyMac4.Agent" and anchor apple generic and certificate leaf[subject.CN] = "Developer ID Application: MacPaw Inc. (AAAAAAAAAA)" and Info[CFBundleVersion] >= 10.10.10

$\underbrace{\text{Info[CFBundleVersion] >= 10.10.10}}_{\text{Version check}}$

Privileged Helper's requirements

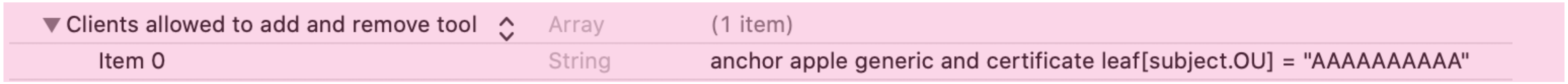


▼ Clients allowed to add and remove tool	Array	(1 item)
Item 0	String	anchor apple generic and certificate leaf[subject.OU] = "AAAAAAAAAA"

$\underbrace{\text{"AAAAAAAAAA"}}_{\text{Signing identity (team id)}}$

h1

Privileged Helper's requirements

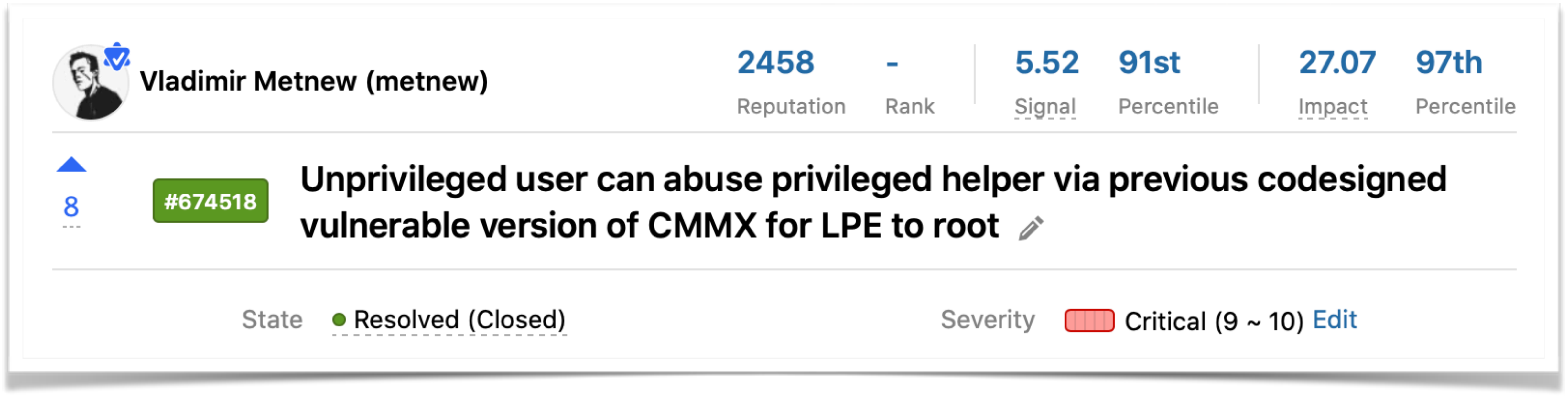


▼ Clients allowed to add and remove tool	Array	(1 item)
Item 0	String	anchor apple generic and certificate leaf[subject.OU] = "AAAAAAAAAA"

$\underbrace{}$

Signing identity (team id)

old 🐞 client versions can connect



Vladimir Metnew (metnew)

2458	-	5.52	91st	27.07	97th
Reputation	Rank	Signal	Percentile	Impact	Percentile

8 #674518 **Unprivileged user can abuse privileged helper via previous codesigned vulnerable version of CMMX for LPE to root**

State ● Resolved (Closed) Severity 🟥 Critical (9 ~ 10) Edit

Issue #3: steps

Preconditions: Privileged Helper is authorized. A malicious executable is present on the user's computer.

- Download an old app version, vulnerable to dylib injection
- Launch client executable with a dylib injection
- Call privileged helper's methods from the injected code
  - In our case it leads to LPE to root

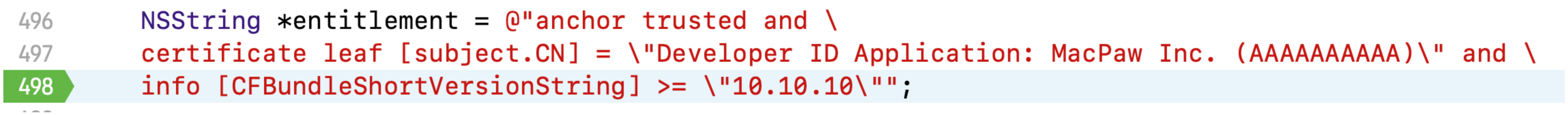
Issue #3: steps

Preconditions: Privileged Helper is authorized. A malicious executable is present on the user's computer.

- Download an old app version, vulnerable to dylib injection
- Launch client executable with a dylib injection
- Call privileged helper's methods from the injected code
  - In our case it leads to LPE to root

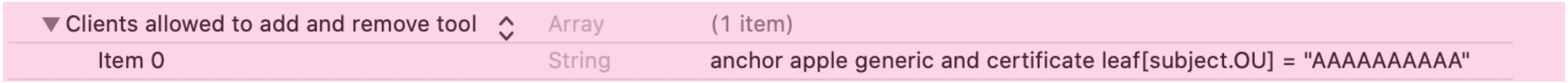
Takeaway: Dylib injection does NOT break the code signature

Fix #3



```
NSString *entitlement = @"anchor trusted and \
certificate leaf [subject.CN] = \"Developer ID Application: MacPaw Inc. (AAAAAAAAAA)\" and \
info [CFBundleShortVersionString] >= \"10.10.10\"";
```

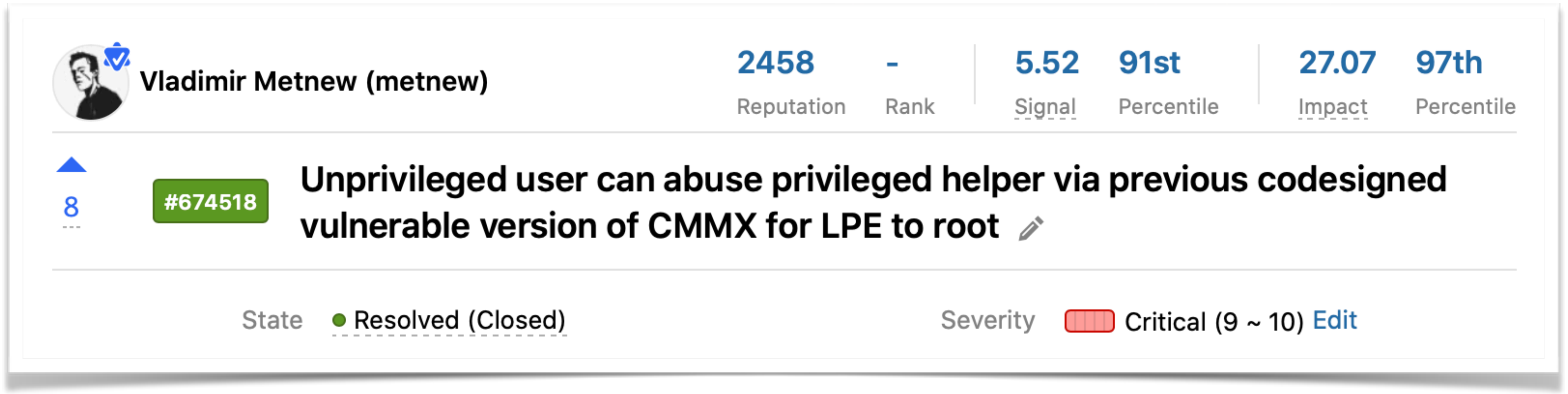
Privileged Helper’s requirements



▼ Clients allowed to add and remove tool	Array	(1 item)
Item 0	String	anchor apple generic and certificate leaf[subject.OU] = "AAAAAAAAAA"

Signing identity (team id)

old 🐞 client versions can connect



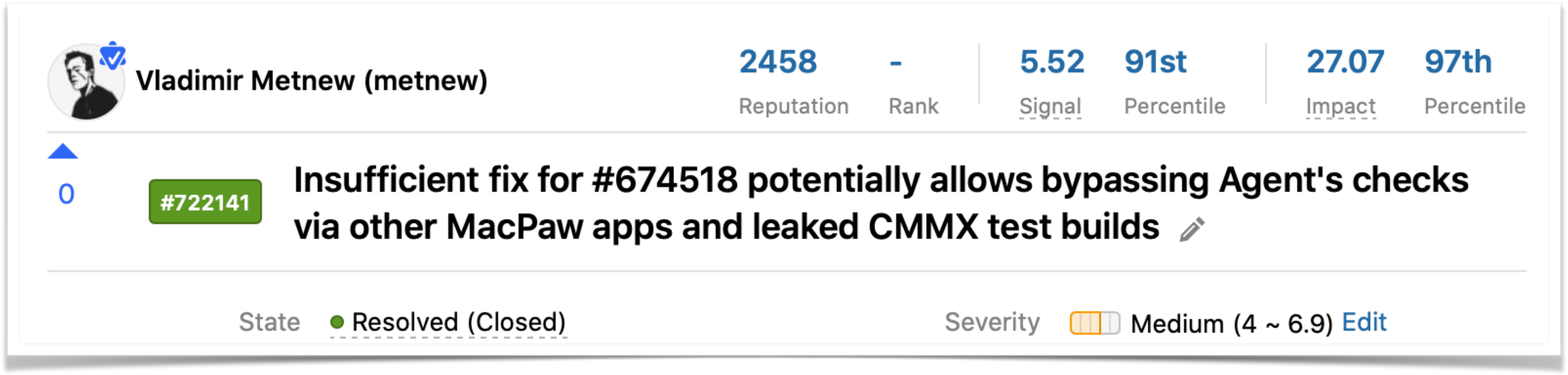
Vladimir Metnew (metnew)

2458	-	5.52	91st	27.07	97th
Reputation	Rank	Signal	Percentile	Impact	Percentile

8 #674518 **Unprivileged user can abuse privileged helper via previous codesigned vulnerable version of CMMX for LPE to root**

State ● Resolved (Closed) Severity Critical (9 ~ 10) Edit

other apps of the same vendor can connect



Vladimir Metnew (metnew)

2458	-	5.52	91st	27.07	97th
Reputation	Rank	Signal	Percentile	Impact	Percentile

0 #722141 **Insufficient fix for #674518 potentially allows bypassing Agent's checks via other MacPaw apps and leaked CMMX test builds**

State ● Resolved (Closed) Severity Medium (4 ~ 6.9) Edit

Fix #4

```
    NSString *entitlement = @"anchor trusted and \
    certificate leaf [subject.CN] = \"Developer ID Application: MacPaw Inc. (AAAAAAAAAA)\" and \
    info [CFBundleShortVersionString] >= \"10.10.10\" and \
    identifier \"com.macpaw.CleanMyMac4\"";
```

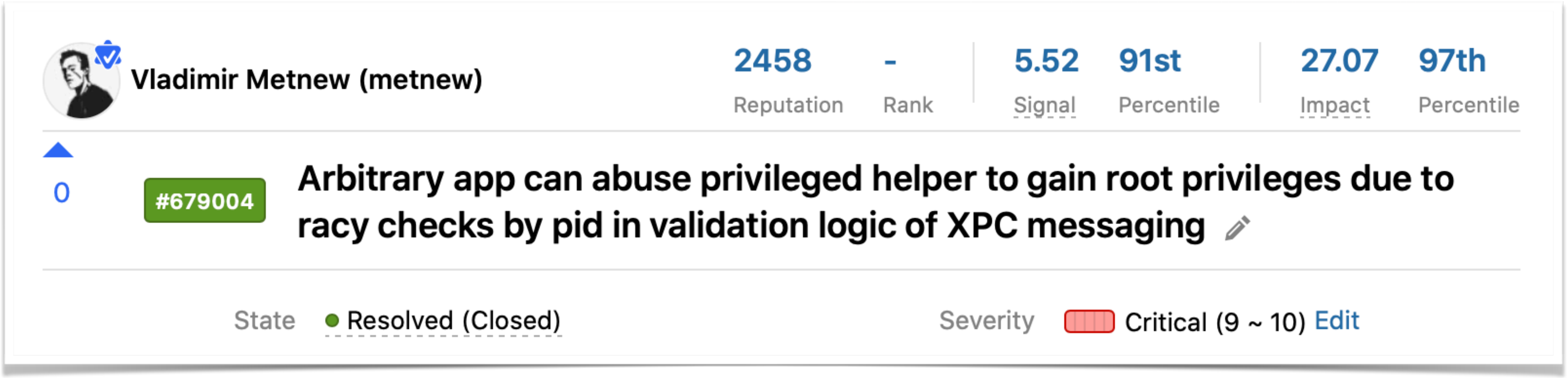
Privileged Helper’s code

```
SecCodeRef code = NULL;
NSDictionary *attributes = @{

    kSecGuestAttributePid: @(connection.processIdentifier) };

SecCodeCopyGuestWithAttributes(0, attributes, kSecCSDefaultFlags, &code);
```

Privileged Helper’s code

```
    SecCodeRef code = NULL;
    NSDictionary *attributes = @{

        kSecGuestAttributePid: @(connection.processIdentifier) };

    SecCodeCopyGuestWithAttributes(0, attributes, kSecCSDefaultFlags, &code);
```

anyone can impersonate the client due to pid checks logic performed by OS



Vladimir Metnew (metnew)

2458	-	5.52	91st	27.07	97th
Reputation	Rank	Signal	Percentile	Impact	Percentile

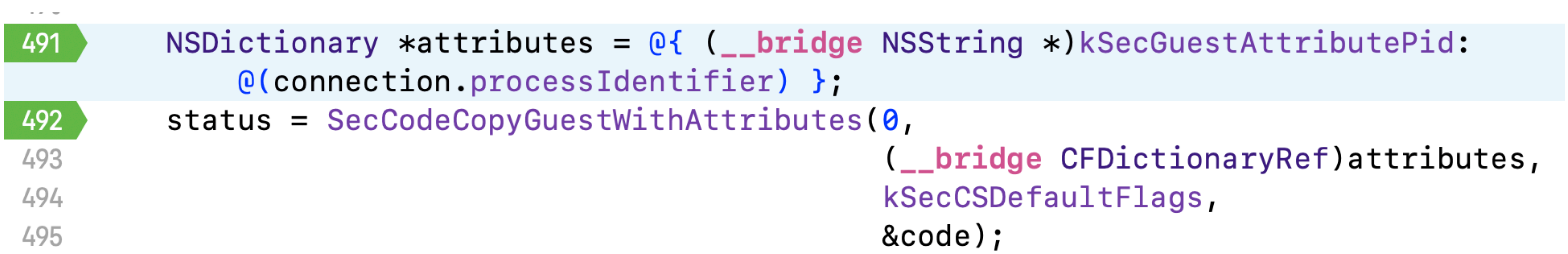
0

#679004 **Arbitrary app can abuse privileged helper to gain root privileges due to racy checks by pid in validation logic of XPC messaging**

State ● Resolved (Closed)

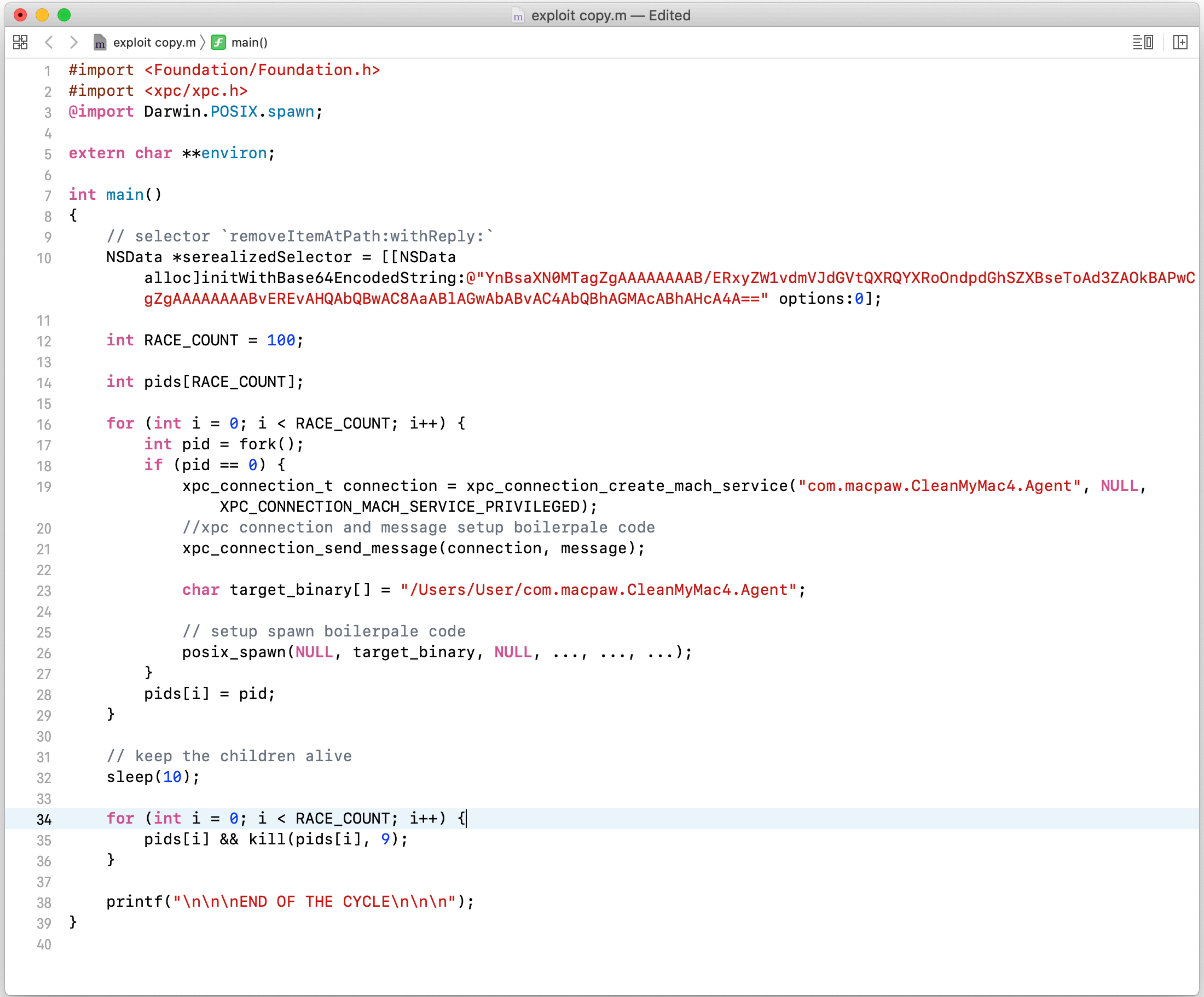
Severity 🟥 Critical (9 ~ 10) Edit

Issue #5: anyone can impersonate the client due to 🐞 racy 🐞 pid checks performed by OS



```
    NSDictionary *attributes = @{ (__bridge NSString *)kSecGuestAttributePid:
        @(connection.processIdentifier) };
    status = SecCodeCopyGuestWithAttributes(0,
                                            (__bridge CFDictionaryRef)attributes,
                                            kSecCSDefaultFlags,
                                            &code);
```

h1

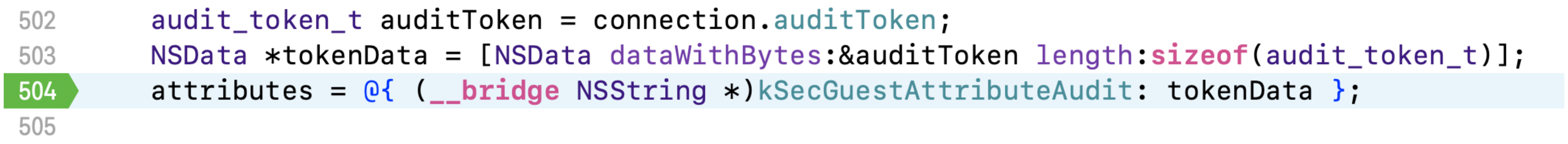


exploit copy.m — Edited

exploit copy.m 〉 main()

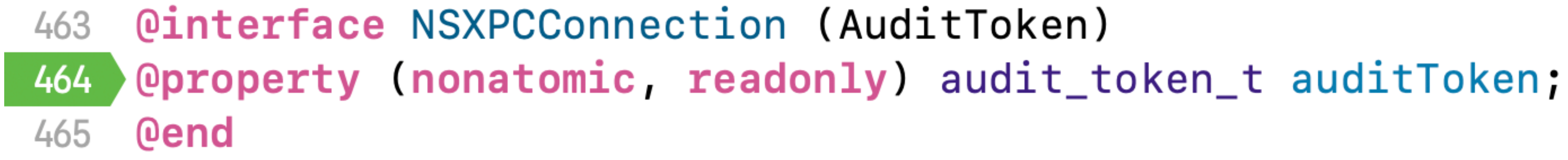
```objc
#import <Foundation/Foundation.h>
#import <xpc/xpc.h>
@import Darwin.POSIX.spawn;

extern char **environ;

int main()
{
    // selector `removeItemAtPath:withReply:`
    NSData *serealizedSelector = [[NSData
        alloc]initWithBase64EncodedString:@"YnBsaXN0MTagZgAAAAAAAB/ERxyZW1vdmVJdGVtQXRQYXRoOndpdGhSZXBseToAd3ZAOkBAPwC
        gZgAAAAAAABvEREvAHQAbQBwAC8AaABlAGwAbABvAC4AbQBhAGMAcABhAHcA4A==" options:0];

    int RACE_COUNT = 100;

    int pids[RACE_COUNT];

    for (int i = 0; i < RACE_COUNT; i++) {
        int pid = fork();
        if (pid == 0) {
            xpc_connection_t connection = xpc_connection_create_mach_service("com.macpaw.CleanMyMac4.Agent", NULL,
                XPC_CONNECTION_MACH_SERVICE_PRIVILEGED);
            //xpc connection and message setup boilerpale code
            xpc_connection_send_message(connection, message);

            char target_binary[] = "/Users/User/com.macpaw.CleanMyMac4.Agent";

            // setup spawn boilerpale code
            posix_spawn(NULL, target_binary, NULL, ..., ..., ...);
        }
        pids[i] = pid;
    }

    // keep the children alive
    sleep(10);

    for (int i = 0; i < RACE_COUNT; i++) {
        pids[i] && kill(pids[i], 9);
    }

    printf("\n\n\nEND OF THE CYCLE\n\n\n");
}
```

Fix #5

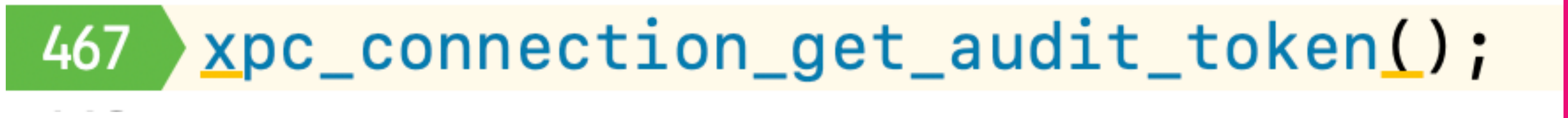


```
audit_token_t auditToken = connection.auditToken;
NSData *tokenData = [NSData dataWithBytes:&auditToken length:sizeof(audit_token_t)];
attributes = @{ (__bridge NSString *)kSecGuestAttributeAudit: tokenData };
```

The APIs are private 😞



```
@interface NSXPCConnection (AuditToken)
@property (nonatomic, readonly) audit_token_t auditToken;
@end
```



```
xpc_connection_get_audit_token();
```

Privileged operations implementation on and SETAPP

Application		SETAPP
API	SMJobBless()	AuthorizationExecuteWithPrivileges()
# bugs reported*	5	

* as for March 2020

Application		SETAPP
API	SMJobBless()	AuthorizationExecuteWithPrivileges()
# bugs reported*	5	0

* as for March 2020

Summary & Takeaways

Takeaways for developers

1. Think about security in your project/company. A good start is creating a security@yourcompany.com email handle.

2. Have one source of truth for Client's signing requirements and one for Privileged Helper's, e.g. put them in Preprocessor Macros and use it in:

   ℹ️ **Info.plist** file

   👑 **listener:shouldAcceptNewConnection:**

3. In signing requirements check at least for:

   🧑‍💻 signing identity

   🆔 bundle identifier

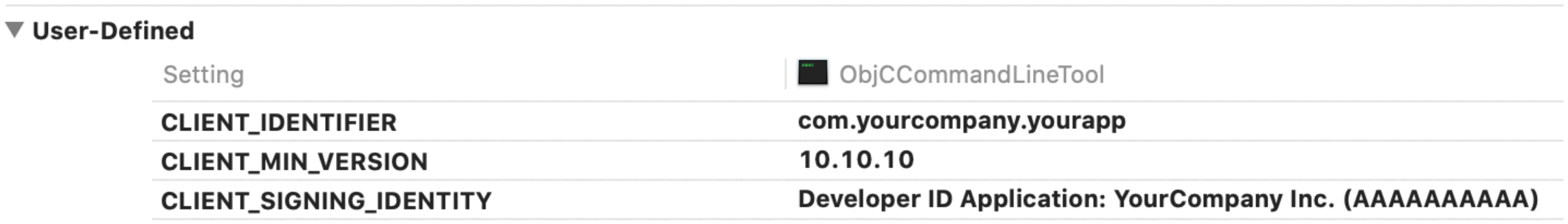
   #️⃣ minimum version

4. In `SecCodeCopyGuestWithAttributes` use 🏷️ audit token to obtain code reference for signature validation, not the pid

5. In order to be a good citizen remember to unregister the Privileged Helper via **launchctl** or **SMJobRemove** API, remove the executable from **/Library/PrivilegedHelperTools** and the auto generated .plist from **/Library/LaunchDaemons**

Example set up requirements for Privileged Helper

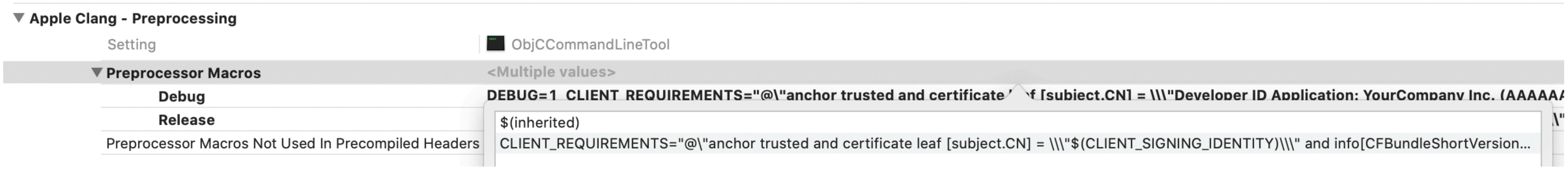
1. Add User-Defined Build Settings:



User-Defined

Setting	ObjCCommandLineTool
CLIENT_IDENTIFIER	com.yourcompany.yourapp
CLIENT_MIN_VERSION	10.10.10
CLIENT_SIGNING_IDENTITY	Developer ID Application: YourCompany Inc. (AAAAAAAAAA)

2. Use them to create a macro definition



Apple Clang - Preprocessing

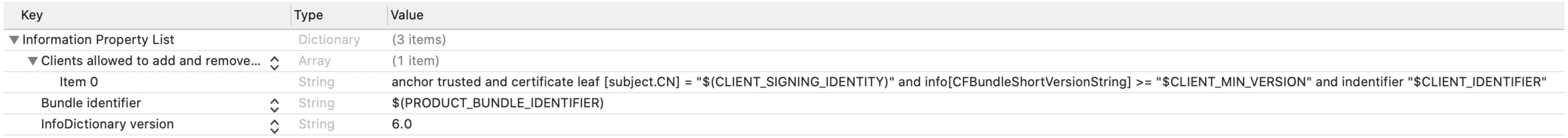
Setting	ObjCCommandLineTool
Preprocessor Macros	<Multiple values>
Debug	DEBUG=1 CLIENT_REQUIREMENTS="@\"anchor trusted and certificate f [subject.CN] = \\\"Developer ID Application: YourCompany Inc. (AAAAAA
Release	\"
Preprocessor Macros Not Used In Precompiled Headers	

```
$(inherited)
CLIENT_REQUIREMENTS="@\"anchor trusted and certificate leaf [subject.CN] = \\\"$(CLIENT_SIGNING_IDENTITY)\\\" and info[CFBundleShortVersion...
```

```
CLIENT_REQUIREMENTS="@\"anchor trusted and certificate leaf
[subject.CN] = \\\"$(CLIENT_SIGNING_IDENTITY)\\\" and
info[CFBundleShortVersionString] >= \\\"$CLIENT_MIN_VERSION\\\"
and identifier \\\"$CLIENT_IDENTIFIER\\\"\""
```

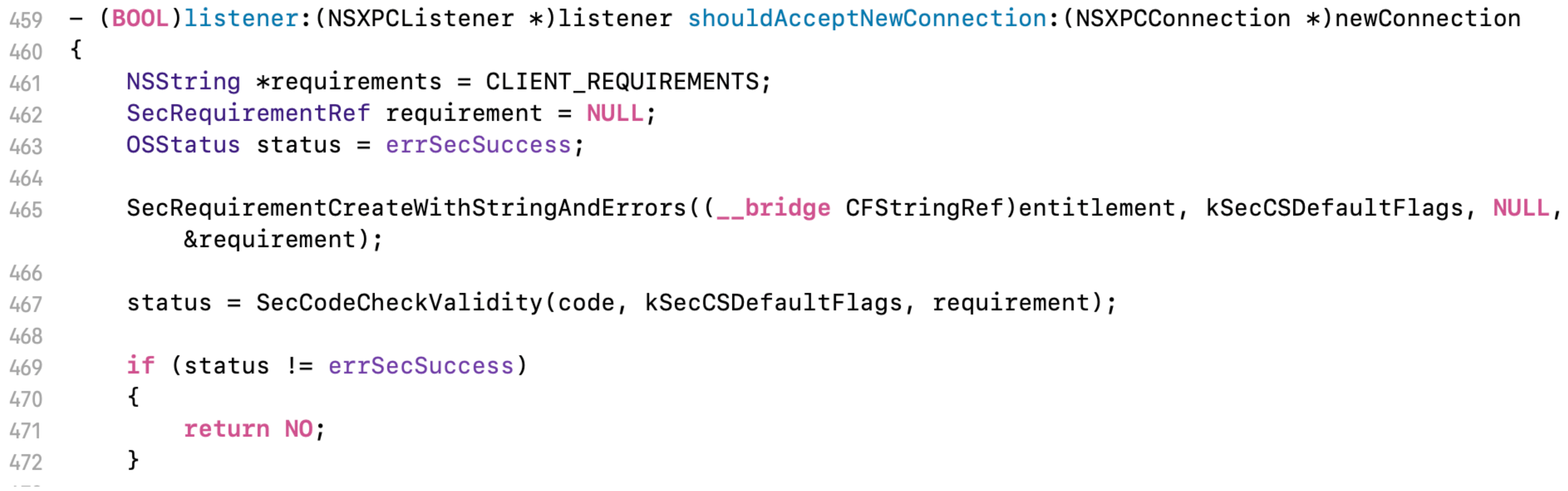
Example set up requirements for Privileged Helper

3. Use your Build Settings in Info.plist client requirements:



Key	Type	Value
▼ Information Property List	Dictionary	(3 items)
▼ Clients allowed to add and remove...	Array	(1 item)
Item 0	String	anchor trusted and certificate leaf [subject.CN] = "$(CLIENT_SIGNING_IDENTITY)" and info[CFBundleShortVersionString] >= "$CLIENT_MIN_VERSION" and indentifier "$CLIENT_IDENTIFIER"
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0

4. Use the Macro Definition from 2. in code to validate incoming connection:



```
- (BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection *)newConnection
{
    NSString *requirements = CLIENT_REQUIREMENTS;
    SecRequirementRef requirement = NULL;
    OSStatus status = errSecSuccess;

    SecRequirementCreateWithStringAndErrors((__bridge CFStringRef)entitlement, kSecCSDefaultFlags, NULL,
        &requirement);

    status = SecCodeCheckValidity(code, kSecCSDefaultFlags, requirement);

    if (status != errSecSuccess)
    {
        return NO;
    }
```

📝 Summary/Wishlist

Summary/Wishlist

1. We need the documentation

There is no easily available Apple's documentation about securing XPC connection with Privileged Helpers

2. We need Code Samples

Apple's code samples are not secure

3. Using pid to check the signature of a process is not secure. It should be clearly stated in docs

Checks by pid are racy by nature

4. Audit token should not be private

It is the most secure way, but it is not available to 3rd party developers

5. There should be some Uninstallation API

When the app is being removed, the Helpers are usually forgotten in **/Library/PrivilegedHelperTools**

📖 Further Reading

1. project-zero ‘Issue 1223: MacOS/iOS userspace entitlement checking is racy’ by Ian Beer

2. OffensiveCon19 'OSX XPC Revisited - 3rd Party Application Flaws' by Tyler Bohan

3. Apple Developer Forums 'XPC restricted to processes with the same code signing?'

4. Objective Development ‘The Story Behind CVE-2019-13013’ by Christian from Little Snitch

5. ‘No Privileged Helper Tool Left Behind’ by Erik Berglund

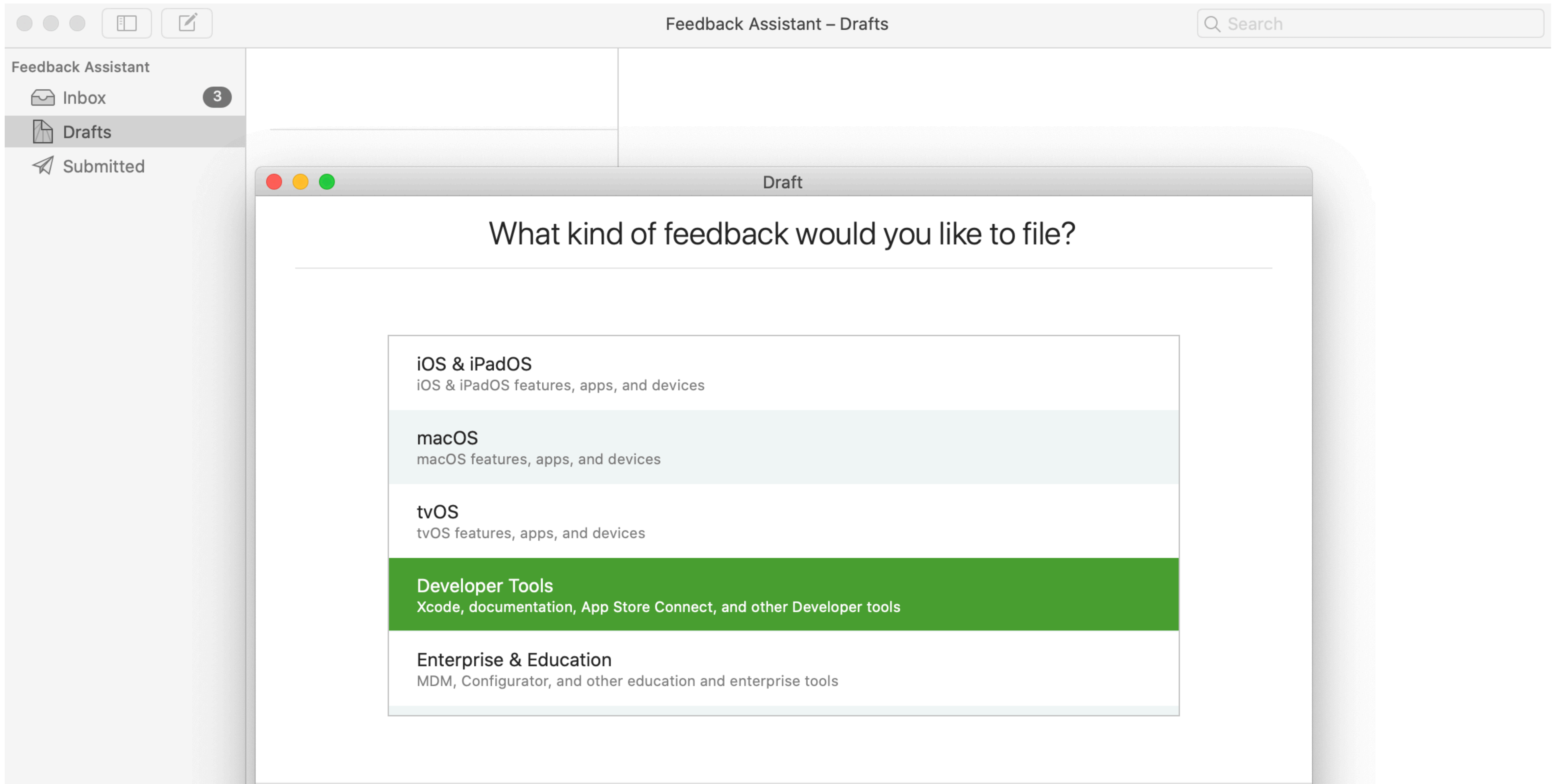
Call to Action 🧞

If I could ask you to do 1 thing, let it be:

Call to Action 🧞

If I could ask you to do 1 thing, let it be:

reporting to Apple, that audit tokens should be made available for 3rd party developers:



Feedback Assistant – Drafts

Feedback Assistant
- Inbox 3
- Drafts
- Submitted

Draft

What kind of feedback would you like to file?

iOS & iPadOS
iOS & iPadOS features, apps, and devices

macOS
macOS features, apps, and devices

tvOS
tvOS features, apps, and devices

Developer Tools
Xcode, documentation, App Store Connect, and other Developer tools

Enterprise & Education
MDM, Configurator, and other education and enterprise tools

Thank you! 🤓

iaronskaya