Quarantine and the quarantine flag

Published: 2020-10-29 · Archived: 2026-04-05 14:31:11 UTC

We all know that a ‘quarantine flag’ is attached to files which are downloaded from the Internet, using most but not all apps, and determines whether an app needs to undergo a full first run check by Gatekeeper. In fact, there’s a great deal more to quarantine and its extended attribute than that. For a start, the majority of items on your Mac which carry a quarantine flag aren’t apps at all, but non-executable documents. And in most cases, macOS doesn’t even know why they are there.

Quarantine and the `com.apple.quarantine` extended attribute (xattr) originated in macOS 10.5 in 2007, although Gatekeeper didn’t appear until 10.7 in 2011-12, at around the same time that sandboxing was introduced.

Entering quarantine

All files which are downloaded from the Internet, using HTTPS or HTTP, in email messages, over AirDrop, and by other means, can have a quarantine flag attached to them by the app which performs the downloading. Custom app download-installers and most updaters either don’t set the flag at all, or, when one is set, remove it (for example, Sparkle-based updaters).

The quarantine flag is an opt-in system, not one imposed by macOS itself. Any developer, including malware authors, can download files from the Internet without setting the flag on them, and any app on your Mac can change or strip the quarantine flag on any item to which it has write permission. The use of these flags in security is very much a gentleman’s agreement, which is easily broken when software doesn’t behave like a gentleman.

Setting the quarantine flag is normally determined the Info.plist property list which every app is required to contain. The entry there which controls flag behaviour is named LSFileQuarantineEnabled, and you can inspect this in each app to check what should happen when that app creates a new file, for example when downloading something from the Internet. When this is set to true, every new file created by that app should have the quarantine flag set; when false, they won’t unless macOS overrides that behaviour. If an unsandboxed app’s Info.plist doesn’t set LSFileQuarantineEnabled explicitly, then the default is not to set the quarantine flag.

macOS also provides a set of overrides to what appears in the Info.plist of many apps, listed in the Additions item in /System/Library/CoreServices/CoreTypes.bundle/Contents/Resources/Exceptions.plist.

The Exceptions.plist property list contains five dictionaries:

- **Additions**, which assigns a lot of app categories, sets Java version requirements, and determines default settings for quarantine on documents created by apps.
- **AppNapOverrides**, which sets App Nap behaviours.
- **HighResolutionOverrides**, which overrides High Res options for apps.
- **LaunchOverrides**, which can disable specific version ranges of apps from being launched; these prevent many older apps from being run.
- **MergeDocumentTypes**, which merges some document types such as *doc* and *docx* for specific apps.

- **Overrides**, which can override other settings.

For example, the entry in the Additions dictionary for the popular BitTorrent client Transmission reads:

```
<key>org.m0k.transmission</key>
<dict>
<key>LSApplicationCategoryType</key>
<string>public-category.internet</string>
<key>LSFileQuarantineEnabled</key>
<true/>
</dict>
```

Referring to the app by its ID of `org.m0k.transmission`, that first assigns the app to an app category of `public-category.internet`, and then sets the app to set the quarantine flag on all documents that it creates, including everything that it downloads.

Among the existing overrides in Catalina, for example, are org.pythonmac.unspecified.BitTorrent and org.xlife.Xtorrent, which ensures that Transmission, Xtorrent and PythonMac BitTorrent clients should write quarantine flags to all their downloaded files. Although this Exceptions property list doesn't cover every client, it should ensure that most do protect their downloads with quarantine flags. However, there's no method by which you can add or modify these, and they don't appear to apply to command tools such as `curl`, which is often used to bypass quarantine flag attachment.

Behaviour – apps

The quarantine flag is among the stickiest of all xattrs. When you unZip an archive which has been flagged, the xattr is normally propagated to all items which are saved from that, a behaviour which ensures that compressed apps retain their flag when uncompressed, for example. This isn't, though, imposed by macOS, and some tools and utilities which can decompress archives may not follow this behaviour; the bundled Archive Utility does, though.

In macOS Mojave and later, a typical quarantine xattr consists of a Unicode string of the form

```
0083;5991b778;Safari.app;BC4DFC58-0D26-460D-9688-81D119298642
```

with the components:

1. the quarantine value in hexadecimal,
2. the time at which the xattr was attached, in hexadecimal,
3. the app or agent responsible for creating the xattr (normally the downloading app too),
4. a UUID referring to the entry for this quarantine flag in the QuarantineEvents database

separated by semi-colons.

The QuarantineEvents database is an SQLite database at ~/Library/Preferences/com.apple.LaunchServices.QuarantineEventsV2. There appears to be no system-level equivalent, so each user is only able to access details of their own quarantine events, not those of other users. It's not known whether spent events are ever removed from there, nor whether maintenance is performed on it.

When an app or other executable is run from the GUI, its quarantine xattr is checked before opening. If the executable code hasn't been cleared previously by a full Gatekeeper check, that is deemed to be its first run, and a full check is performed. If that's successful, the quarantine value on all checked executable code is changed from (for example)

`00000000 10000011 = 83`

to

`00000000 11000011 = C3`

Subsequent attempts to run that code are then no longer blocked for the first run Gatekeeper check to be performed. Successful completion of that first run check doesn't alter the quarantine xattrs attached to non-executable files within an app bundle, though.

Earlier versions of macOS have used other bits in the quarantine value too. For example, in Sierra and earlier an app which has passed first run and been successfully opened could end up with a value of

`00000000 11100011 = E3`

the lower-order bit signifying that the app had also been run. These appear to have fallen into disuse in Mojave.

The QuarantineEvents database contains and retains additional information for these flags, which includes the reason for their attachment in the LSQuarantineType value, which includes: LSQuarantineTypeWebDownload, LSQuarantineTypeEmailAttachment, LSQuarantineTypeOtherDownload, LSQuarantineTypeInstantMessageAttachment, LSQuarantineTypeCalendarEventAttachment, LSQuarantineTypeOtherAttachment, and LSQuarantineTypeSandboxed, which is only attached to documents.

In normal circumstances, quarantine xattrs which are attached to apps and other executables remain in place until that app is removed.

Developers who ship apps which are signed or notarized need to check that those will successfully pass through Gatekeeper first run checks before distributing those apps. I have discussed ways of doing this here.

Executable scripts with quarantine flags may not undergo Gatekeeper checks, because most are unsigned. The presence of a quarantine flag on a script therefore normally results in macOS blocking it from running until that flag has been removed.

Behaviour – documents

macOS has been attaching quarantine flags to documents for as long as it has been to apps, as part of the the same process. If a webpage or other file is downloaded from the Internet and saved on your Mac by an app which adds quarantine flags, then a normal quarantine xattr will be added to it. When you decompress a flagged Zip archive, quarantine flags are automatically attached to all the files extracted from it.

These non-app flags differ as follows:

- Opening a quarantined document or non-executable file doesn't trigger a Gatekeeper check, which would in any case be meaningless.
- Nothing appears to change or remove a flag, unless you use a utility or command to do so. The sole exception to this is with flags attached by sandboxed apps, which can replace one another.

- macOS also adds its own quarantine flags to documents which haven't been downloaded from the Internet. Oddly, while quarantine of apps is an opt-in behaviour, you can't opt out of this behaviour, as it's built into the macOS sandbox.
- Performing certain operations with quarantined documents may be forbidden. For example, a flagged shell script can't normally be executed. Thus using flagged files can result in errors.

Document quarantine xattrs attached following download from the Internet have identical content to those attached to apps, including the UUID of an entry in the QuarantineEvents database. Those attached to files which haven't been downloaded differ, as they aren't associated with entries in the QuarantineEvents database, so lack the UUID. Quarantine values you're likely to encounter include:

- `00000000 10000001 = 81`
- `00000000 10000010 = 82`
- `00000000 10000011 = 83`

each of which has the high-order bit set to indicate that the file is still in quarantine, and are normally given with an LSQuarantineType of LSQuarantineTypeSandboxed.

The addition of quarantine flags to files which have never been downloaded from the Internet appears to be a relatively recent behaviour, but has now been seen to occur in macOS Sierra and later. Apple doesn't appear to have documented this for users, and references to this behaviour are buried deep in Apple's now outdated but not replaced Entitlement Key Reference.

Sandboxed apps, which includes many of those bundled with macOS and all delivered by the App Store, will attach quarantine flags to files which Apple describes as executable unless the app has the com.apple.security.files.user-selected.executable entitlement. Apple explains:
"By default, when writing executable files in sandboxed apps, the files are quarantined. Gatekeeper prevents quarantined executable files and other similar files (shell scripts, web archives, and so on) from opening or executing unless the user explicitly launches them from Finder.
If those executables are tools that are intended to run from the command line, such as shell scripts, this presents a problem."

Note that this refers to a sandboxed app writing executable files, which should only account for a very small number of files having a quarantine flag attached.

The role and purpose of these quarantine flags added by sandboxed apps remains obscure, beyond being used to prevent the execution of shell scripts, web archives, etc. Not only that, but sandboxed apps write them, when permissions allow, to all documents which they open, even though the document may not be formally saved by that app.

Access

As extended attributes, you can view and edit quarantine flags using standard tools including the `xattr` command and my free utility xattred. These flags are easier to work with than many xattrs as they are UTF-8 text, not encoded binary.

As a developer, you can opt to access them as xattrs, which is easiest using extensions to the URL class in Swift, for instance. They're also accessible more directly as NSURL Resources, using code such as

```
var theQuarFlag: AnyObject? = nil
try theNSURL.getResourceValue(&theQuarFlag, forKey: kCFURLQuarantinePropertiesKey as URLResourceKey)
```

which, if `theQuarFlag` is non-nil, is a dictionary containing key-value pairs for all known fields in the quarantine flag. These not only include the data in the xattr, but where there's an extant entry in the QuarantineEvents database, this also returns information from that. Keys are listed in LSQuarantine.h, and include:

- LSQuarantineAgentNameKey, the name of the downloading agent;
- LSQuarantineAgentBundleIdentifierKey, the bundle ID of the downloading agent;
- LSQuarantineTimeStampKey, a CFDateRef to the date and time that the quarantine flag was attached;
- LSQuarantineTypeKey, one of the values listed above;
- LSQuarantineOriginURLKey, a CFURLRef to the original location of the file;
- LSQuarantineDataURLKey, a CFURLRef to the data source of the file.

Removing a quarantine flag simply requires setting it to nil, as in

```
try theNSURL.setResourceValue(nil, forKey: kCFURLQuarantinePropertiesKey as URLResourceKey)
```
