

Signet 1.0b2 for Mojave and High Sierra

Release Notes

Howard Oakley <https://eclecticlight.co>



These release notes introduce **Signet**, a free tool for checking the signatures of apps and other bundles in macOS.

Normally, the signatures of apps and other bundles installed on your Mac are checked thoroughly just once, when they are first run after being downloaded and installed. If you keep migrating from one Mac to the next, chances are that your current Mac contains a lot of executable code which has no signatures at all, and possibly some for which the signatures have been revoked.

Checking signatures is a very good way of discovering old software left over from the past which has never been cleared away properly. It can also reveal apps and bundles which, because of their signature weakness or absence, could be exploited or abused. While there's an excellent tool for checking individual signatures, Objective-See's [What's Your Sign](#), I don't know of any which will check multiple bundles in this way.

When I was developing Signet, I discovered a Microsoft app whose signature had been revoked, which macOS Mojave was quite happy to open despite that, a lot of properly signed apps which contained unsigned apps inside them, and a great many apps and other bundles (some quite recent) which have no signature at all.

What you need

- A Mac running Mojave or High Sierra. This release has been built to be fully compatible with Mojave (APFS, Light and Dark modes) and High Sierra, but doesn't support any previous release of macOS.
- A copy of the latest release of Signet from <https://eclecticlight.co/downloads/> (This is delivered by secure HTTPS download.)

Getting started

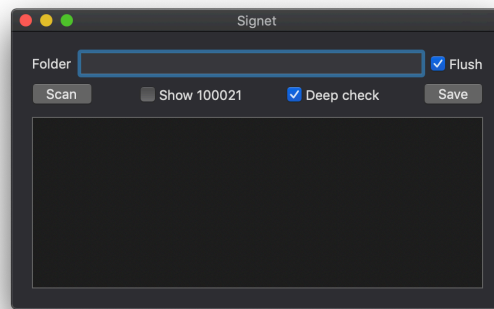
Signet comes compressed as a Zip file, which you should decompress, and move the app to your preferred folder, such as /Applications. It is not fussy where it is run from, though. It not only uses my developer code signature, but is notarized too. If you see any abnormal warning when opening it the first time, or at any time thereafter, please contact me immediately.

Use

Signet lets you check bundles, including apps, on any storage accessible from your Mac. These are special types of folder with set contents and structure, and are standard for apps, various types of plug-in, extensions, frameworks, and more. Most contain executable code, which should mean that they are signed, even when nested inside a signed app.

Signet lets you specify the folder to be searched in any of three ways:

- If you type the path to a folder in the box next to the word **Folder**, when you click on the **Scan** button, Signet uses that as the folder in which to search. If that folder does not exist, it will do nothing. You can start this folder path with the special character ~ as shorthand to indicate your Home folder, e.g. ~/Applications is the Applications folder in your Home folder.
- If you have *no* text in the box next to the word **Folder** when you click on the **Scan** button, Signet displays a normal Open File dialog in which you should select the folder you wish to search, then click on the **Open** button.
- You can also drag and drop any folder from a Finder window into the box next to the word **Folder**, then click on the **Scan** button to scan that folder.



Normally, when you insert the path to a folder in the **Folder** textbox, that path is cleared when you perform the next scan. If you want the path to remain there, so that you can modify it, perhaps, ensure that you uncheck the **Flush** checkbox. When it is ticked, the textbox will be cleared of its contents each time that you scan.

Signet then inspects every item within your chosen folder to determine if it is a bundle, and whether it contains executable code. If it contains executable code, Signet then asks macOS to validate its signature. Any errors reported are then compiled to form Signet's output. When it has checked any bundle, it then looks inside that for nested bundles, checking those as appropriate.

There are currently two checkboxes in Signet's windows. The first, labelled **Show 100021**, determines whether its check should also report those bundles which return an error type 100021 indicating that trying to access the bundle failed because of an 'illegal operation on a directory'. The second, labelled **Deep check**, determines the thoroughness of signature checking. By default, it is ticked, and checks for and reports most if not all errors; if you

Signet 1.0b2 for Mojave and High Sierra

Release Notes

Howard Oakley <https://eclecticlight.co>

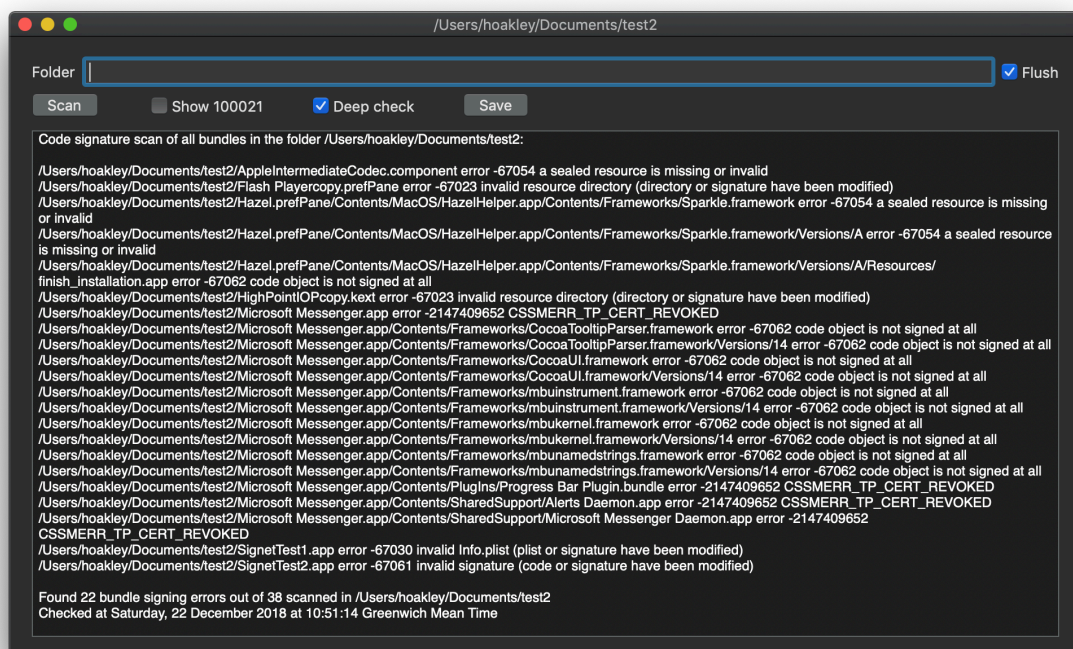
uncheck it, tests will be quicker, but only detect unsigned bundles, those with revoked certificates, and more significant errors.

Inspecting large and deep folders can take many minutes. Signet runs its scans in the background, displaying a ‘busy’ spinner next to the Scan button to indicate that it is still working hard to complete the scan. It also uses a lot of memory during very large scans: macOS should cope fine with this, but you should run scans on smaller folders such as / Applications rather than scanning the root /.

Once its search is complete, Signet will summarise its results into the lower scrolling text view. These include the total number of executable bundles found, out of the total number of executable bundles found. To save the report, click on the **Save** button and select an appropriate or new file.

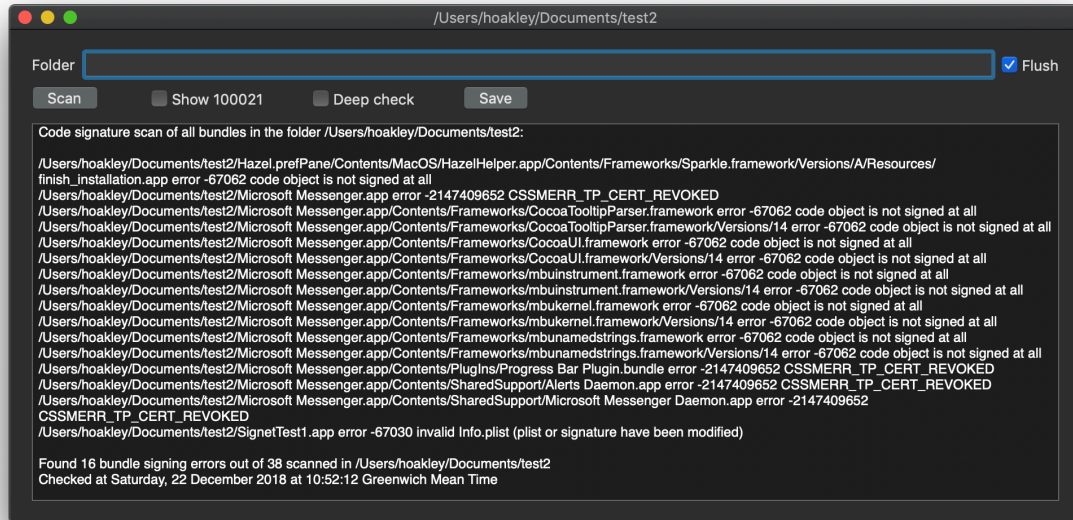
Unlike my 32-bitCheck app, Signet can open multiple windows. When you’re done with the app, use the **Quit** command to quit it. In the **Help** menu the **Browse updates** item opens a list of software updates at the Eclectic Light Company blog in your default browser.

⚠ Signet does not have the ability to scan any folders containing private data, including your photos, email, address book, etc. Those folders shouldn’t contain executable code, so this app shouldn’t be going anywhere near them. However, if you want to include those protected folders in its scans, add it to the **Full Disk Access** list in the **Privacy** tab of the Security & Privacy pane.



Signet 1.0b2 for Mojave and High Sierra
Release Notes
Howard Oakley <https://eclecticlight.co>

When run with the **Deep check** box ticked, Signet checks for and reports all the errors which can readily be detected in bundle signatures. In my folder of test samples, it finds 22 bundles which have signing errors, including one notarized app in which the code has been altered in two bytes.



With the **Deep check** box unchecked, the scan is significantly quicker, but it only detects and reports three types of error: unsigned bundles, those with revoked certificates, and one in which the Info.plist has been altered, a total of 16 signing errors. It doesn't detect another in which the code has been altered.

How it works

Signet performs a deep traversal of all items within the chosen path. It checks each URL within the directory to determine whether it is a Finder alias; if it is, it does not do anything further, but if it isn't a Finder alias, it then checks whether it is an executable bundle, i.e. a bundle containing executable code. Errors occurring at this stage are included in Signet's listing only when the **Show 100021** checkbox is ticked/checked.

If it is an executable bundle, Signet treats the bundle as containing 'static code' and asks macOS to perform a validation on it according to the **Deep check** setting. If that is ticked (default), the checks are performed using `SecCSFlags` of `kSecCSStrictValidate`; if that checkbox is unchecked, then `SecCSFlags` are set to `kSecCSBasicValidateOnly`.

There are circumstances in which even strict checking may miss a signature problem, but those are most unlikely to occur except in malware deliberately trying to cheat signature validation. As Signet is not intended for use to detect malware, the speed gain in not implementing a full strict validation is preferred over that completeness.

The two depth settings do not affect the depth of scanning for bundles in any way.

Signet 1.0b2 for Mojave and High Sierra
Release Notes
Howard Oakley <https://eclecticlight.co>

Viewed in the unified log, such checks typically appear as:

```
07:26:00.808115 Unable to parse ticket.
07:26:00.808116 error registering ticket: -1
07:26:00.808141 com.apple.securityd MacOS error: -1
07:26:00.808200 com.apple.securityd Error registering stapled ticket: [hex]
07:26:00.808635 SecTrustEvaluateIfNecessary
07:26:00.808944 com.apple.securityd cert[2]: WeakSignature =(leaf)[]> 0
07:26:00.808951 com.apple.securityd cert[2]: WeakSignature =(leaf)[]> 0
07:26:00.809584 com.apple.securityd cert[2]: WeakSignature =(leaf)[]> 0
07:26:00.809588 com.apple.securityd cert[2]: MissingIntermediate =(leaf)[force]> 0
07:26:00.809593 com.apple.securityd cert[2]: WeakSignature =(leaf)[]> 0
07:26:00.809603 com.apple.securityd cert[2]: WeakSignature =(path)[]> 0
07:26:00.809619 com.apple.securityd cert[2]: BlackListedKey =(path)[force]> 0
07:26:00.809766 com.apple.securityd Trust evaluate failure: [root BlackListedKey
MissingIntermediate WeakSignature]
07:26:00.810094 SecStaticCode: verification failed (trust result 6, error
-2147409652)
07:26:00.810098 com.apple.securityd MacOS error: -2147409652
07:26:00.810134 com.apple.securityd MacOS error: -2147409652
```

which is then reported by Signet as:

```
/Users/hoakley/Documents/test2/Microsoft Messenger.app error -2147409652
CSSMERR_TP_CERT_REVOKED
```

At the end of each report, Signet gives the total number of bundles which it checked, and the number of those for which signature validation return an error.

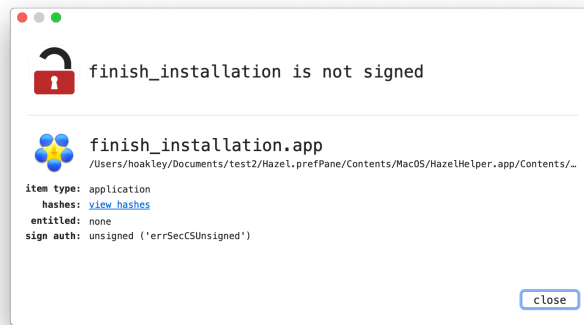
Interpreting results

Scans performed with the **Show 100021** checkbox ticked are likely to return large numbers of errors of type 100021, kPOSIXErrorEISDIR, an 'illegal operation on a directory'. These appear benign and irrelevant. Error codes from signature validation are explained as meaningfully as macOS permits. Common ones include:

- -2147409652 CSSMERR_TP_CERT_REVOKED, the certificate has been revoked
- -67007 resource envelope is obsolete (version 1 signature)
- -67008 unsealed contents present in the root directory of an embedded framework
- -67013 resource envelope is obsolete (custom omit rules)
- -67021 nested code is modified or invalid
- -67023 invalid resource directory (directory or signature have been modified)
- -67030 invalid Info.plist (plist or signature have been modified)
- -67054 a sealed resource is missing or invalid
- -67056 code has no resources but signature indicates they must be present
- -67061 invalid signature (code or signature have been modified)
- -67062 code object is not signed at all, which is by far the most common.

Many third-party apps which are distributed independently of the App Store use the Sparkle mechanism to offer and install updates. This hasn't been without its vulnerabilities in the past, and one odd feature about many (but not all) apps which currently use Sparkle is that one app nested within them is unsigned: Autoupdate.app or finish_installation.app, which actually runs the update process. This is normally found in the path within the app's bundle of / Frameworks/Sparkle.framework/Versions/A/Resources/Autoupdate.app.

Signet 1.0b2 for Mojave and High Sierra
Release Notes
Howard Oakley <https://eclecticlight.co>



Many older apps such as Apple's Aperture and the Bento database used now-obsolete signature formats, and return errors of -67013 “resource envelope is obsolete (custom omit rules)”.

Even current App Store apps may have signature errors in nested components: for example, I have an audio app purchased from the App Store which contains three plug-ins in its app bundle which fail Signet's strict validation, one of which reports that “nested code is modified or invalid”.

Many accessory apps from major vendors such as Adobe and Microsoft are unsigned or return signature errors. For example, Adobe Acrobat 2015 contains two nested apps which lack any signature, and no less than 12 of the support apps and bundles for old Adobe products which are still installed on my Mac have had their certificates revoked.

Current Microsoft Office apps (16.20) contain large numbers of proofing tools which contain executable code, but are completely unsigned.

Very many frameworks and plug-ins are unsigned or return signing errors. Some of the most serious include Apple's Compressor.framework and Qmaster.framework stored in /Library/Frameworks, which use an ‘obsolete resource envelope’ from an old signature format, two StuffIt frameworks, Microsoft's SilverLight Internet Plug-In which is completely unsigned, the Java Virtual Machines stored in /Library/Java, in which even the JDK 10.0.2 and its preference pane are unsigned. Most older QuickTime components are reported with signature errors, with many lacking any signatures.

So what should you do with a bundle for which errors are reported? That depends on what the bundle is for and what the error is. Some reputable software developers consider that what Signet does is completely worthless. They argue that signatures are only there for Gatekeeper, and as Gatekeeper is only there to check apps when they are first run, it doesn't matter once the app has passed that check.

Others consider that checking signatures is “security theatre”, and unless those checks were accompanied by databases containing ‘true’ signature references, they are worthless.

Signet 1.0b2 for Mojave and High Sierra

Release Notes

Howard Oakley <https://eclecticlight.co>

I built Signet for myself. If you want to use it, you're very welcome. I use it to discover old and potentially vulnerable software so that I can look for more modern replacements, or simply get rid of them as expired. I think that this is all part of good hygiene, or housekeeping if you prefer, and the information which Signet provides about signatures is a useful input to my decisions about what to keep and what to get rid of. In the course of doing that, I think that I am making my Mac significantly more secure, particularly when I can replace an old unsigned app with one from the App Store or which is notarized. I'd like you to have the same choice: if you don't feel that it helps, then by all means trash it, but please don't ask for your money back or complain that it wasn't worth using.

Warnings

As mentioned above, there is a known vulnerability in signature checking which means that crafted signatures can sometimes cheat the validation system used by Signet. Signet is not intended to detect malware, but to scan non-malicious apps and bundles for signature errors. If you want to detect malware, use one of the several excellent anti-malware apps such as Malwarebytes.

When Signet scans bundles, it does so exhaustively, which takes significant time and memory. The checks it performs may require online validation of certificates, which can also slow them down. If you ask Signet to check any folder containing a very large number of bundles, that will be protracted, involve many signature checks, a lot of CPU time, and lots of memory too. Don't be tempted to try it on the root folder of your startup volume, or it could take many hours. With a little patience, it can be used on individual top-level folders such as /System/Library (dull but important), /Library (rich and very important), /Applications (often rich and surprising), and ~/Library (rich and important).

Signet 1.0b2 for Mojave and High Sierra

Release Notes

Howard Oakley <https://eclecticlight.co>

Change list

Signet 1.0b2:

- added less deep check option
- hopefully built for compatibility with High Sierra
- improving timing of busy spinner when scanning.

Signet 1.0b1:

- initial release.

22 December 2018.