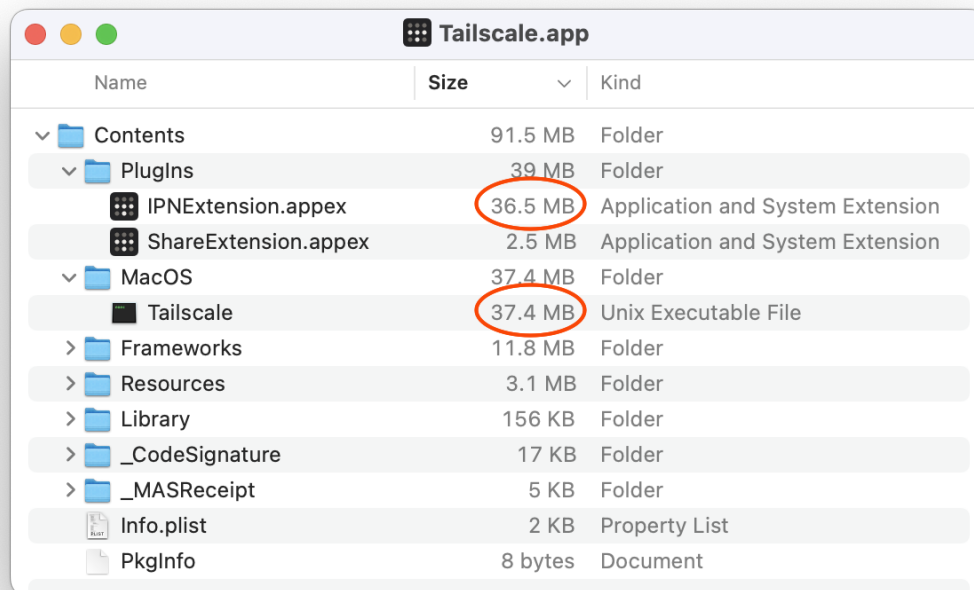


Reducing Tailscale's binary size on macOS

Mihai Parparita on February 16, 2023



A few months ago I was poking around the package contents of Tailscale's macOS app when I happened to notice that the main `Tailscale` binary was surprisingly large — more than 37 megabytes. Some of the size is explained by it being a [universal app](#) — it includes both `x86_64` (for Intel-based Macs) and `arm64` slices (for Apple silicon ones). However, even when accounting for that, it seemed too large for what is mostly a UI wrapper — the core logic lives in a [network extension](#). In fact, looking at that extension (`IPNExtension.appex`), I was surprised to find that it was only slightly smaller (36.5 MB).



Name	Size	Kind
Contents	91.5 MB	Folder
Plugins	39 MB	Folder
IPNExtension.appex	36.5 MB	Application and System Extension
ShareExtension.appex	2.5 MB	Application and System Extension
MacOS	37.4 MB	Folder
Tailscale	37.4 MB	Unix Executable File
Frameworks	11.8 MB	Folder
Resources	3.1 MB	Folder
Library	156 KB	Folder
_CodeSignature	17 KB	Folder
_MASReceipt	5 KB	Folder
Info.plist	2 KB	Property List
PkgInfo	8 bytes	Document

Inspecting the main executable with [Hopper](#) showed a lot of unlabeled functions, which was not immediately helpful. Running `strings` on it showed a lot of Go package names, which was the clue that I needed: we were including Tailscale's open source [Go core](#) not just in the extension (expected), but in the main app itself. This also explained the very similar binary sizes.

I thought this would be a quick size win, so I removed the static library that we generate from our Go code from the main app's dependencies... and promptly got a linker error: "`ActLikeCLI` is not defined." I now had my explanation for why we included the Go code in the main app: Tailscale [has a CLI interface](#), and on macOS it's invoked by [running the regular app](#) from the command line.

Knowing that the app only needed the CLI bits of the Go code, I set about generating a separate static library with just that, figuring it would be a lot smaller. However, as I started to get lost in a maze of twisty little [redo](#) rules, [all alike](#), I began to reconsider. Even if I got this to work, would there be a significant size savings? The CLI static library would still end up including its own copy of the Go runtime and code that is shared with the Tailscale backend. Looking at the Linux version of Tailscale, the `tailscale` binary (which is just the CLI) is still more than half the size of `tailscaled` (the backend daemon) — 14 MB vs. 24 MB.

I decided to approach the CLI invocation from first principles. All the Mac app really needs to do is invoke the `Run` function from Tailscale's `cli` Go package. Most of the time the Go code is already running (in the network extension), and there's a [local HTTP server](#) that the app uses. We could add a `/localapi/v0/cli` endpoint and have the app invoke it. However, I hesitated to do this. This was partly because it would add yet more states and edge cases: what if the backend is not running, or what if a local firewall blocks the request? We also knew that these local endpoints are scrutinized for security reasons, and are wary of adding such powerful ones.

Letting my mind wander a bit, I was reminded of two tidbits:

- 1 Tailscale can be built in a mode where it [combines the backend and CLI into one binary](#), for extra size savings.
- 2 Some programs (most memorably [Chrome](#)) just have a very small shell binary, with the bulk of the logic being in a separate library that is loaded and invoked at startup.

I wondered if this approach could be taken here: the network extension already has the combined binary, including the CLI logic — so what if we treated it as a shared library that we could load and call functions from, when the app needs to run in CLI mode? Some playing around with `dlopen` and `dlsym` showed that it could work. Here's a sketch of what I ended up with (all of the error-handling has been omitted):

```
// Generate path to the app extension binary.
let plugInsURL = Bundle.main.builtInPlugInsURL
let extensionURL = plugInsURL.appendingPathComponent(
    "IPNExtension.appex/Contents/MacOS/IPNExtension")

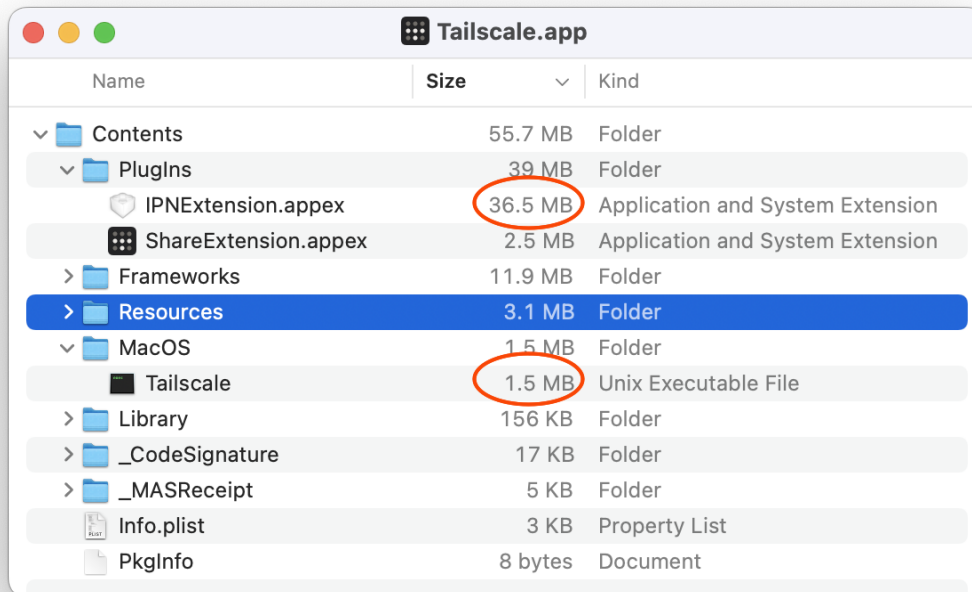
// Open the extension binary as a shared library and find
// the CLI entrypoint symbol.
let extensionHandle = dlopen(extensionURL.path, RTLD_LOCAL)
let goBeCLISymbol = dlsym(extensionHandle, "goBeCLI")

// Invoke the goBeCLI symbol.
 typealias goBeCLIType = @convention(c) (UnsafePointer<Int8>) -> Void
let dir = ... // URL to the app sandbox directory
dir.path.withCString {
    goBeCLI($0)
```

```
}
```

```
// Unreachable beyond this point, Go does an os.Exit.
```

The real code ended up being slightly more complicated, because it needs to handle Tailscale's [standalone variant](#), which has the network extension at a different path. The only other gotcha was that we had to make sure that `goBeCLI` was in [our exported symbols list](#); otherwise, the linker would remove it when doing dead code stripping.



This change shipped with Tailscale v1.36, and it was very satisfying to check the binary size for it — 35 MB smaller than v1.34. In addition to scratching my optimization itch, it should result in faster download and update times for everyone.

Share via [Twitter](#) [LinkedIn](#) [Reddit](#) [Email](#) [Link](#)

[← Back to index](#)

Subscribe for monthly updates

Product updates, blog posts, company news, and more.

Subscribe

Too much email?  [RSS](#)  [Twitter](#)

LEARN

- [SSH Keys](#)
- [Docker SSH](#)
- [DevSecOps](#)
- [Multicloud](#)
- [NAT Traversal](#)
- [MagicDNS](#)
- [PAM](#)
- [PoLP](#)
- [All articles](#)

GET STARTED

- [Overview](#)
- [Pricing](#)
- [Downloads](#)
- [Documentation](#)
- [How It Works](#)
- [Compare Tailscale](#)
- [Customers](#)
- [Changelog](#)
- [Use Tailscale Free](#)

COMPANY

- [Company](#)
- [Newsletter](#)
- [Press Kit](#)
- [Blog](#)
- [Careers](#)
- [Contact Sales](#)
- [Contact Support](#)
- [Community Forum](#)
- [Security](#)
- [Status](#)
- [Twitter](#)
- [GitHub](#)



WireGuard is a registered trademark of Jason A. Donenfeld.

© 2023 Tailscale Inc.

[Privacy & Terms](#)