Opting In to Transparent Telemetry

Transparent Telemetry, Part 4

Russ Cox
February 24, 2023

research.swtch.com/telemetry-opt-in

Earlier this month I posted "Transparent Telemetry for Open-Source Projects*", making the case that open-source software projects need to find an open-source-friendly way to collect basic usage and performance information about their software, to help maintainers understand how their software is used and prioritize their work. I invited feedback on a GitHub discussion and over email.

In general, the feedback was mostly constructive, and mostly positive. In the GitHub discussion, there were some unconstructive trolls with no connection to Go who showed up for a while, but they were the exception rather than the rule: most people seemed to be engaging in good faith. I also saw some good discussion on Twitter and Mastodon, and I received some good feedback via email.

People who read the posts seemed to agree that there's a real problem here for open-source maintainers and that transparent telemetry or something like it is an appropriately minimal amount of collection. By far the most common suggestion was to make the system opt-in (default off) instead of opt-out (default on). I have revised the design to do that.

The rest of this post discusses the reasons for the change to opt-in as well as the effects on the rest of the design.

Opt-in vs Opt-out

Many people were fine with this telemetry being on by default, precisely because it is carefully designed to be so minimal. On the other hand, multiple people told me things like "this is the first system I've seen that I'd actually opt in to, but I'd still turn it off if it was on by default." A few people were concerned about a kind of reidentification attack on the published records: if you knew that company X was on record as being the only user of a given obscure configuration, you might be able to find their records and learn other things about which Go features they use. My imagination is not good enough to figure out how an attacker could exploit data like that, but even so it does seem not worth dismissing entirely.

Another concern I have with an opt-out system is that my goal is to find a path that works for many open source projects, not just Go. Although we've carefully architected the system not to collect sensitive data, if Go's telemetry were opt-out, that might be used to justify other opt-out systems that are not as careful about what they collect. For example, despite the blog posts discussing at length the distinction between everything .NET collects and the much more limited data I am suggesting that Go collect, too many people still seemed to equate them.

For all these reasons, I've revised the design to be opt-in (default off). Doing so does introduce at least two costs: first, we must run an ongoing campaign to educate users about opting in is a good choice for them, and second, with fewer systems reporting, the telemetry cost imposed on any particular user is higher.

The Campaign Cost of Opt-In

The first new cost in an opt-in system is the ongoing overhead of asking users to opt in, with a clear explanation of what is and is not collected. There are a few obvious times when that makes sense:

– During a graphical install of Go, with two different buttons for the two choices (no click-through default).

– In blog posts and release notes for new Go releases.

– During our traditional user surveys.

– The first time VS Code Go is invoked on Go code.

We may think of others ways too, of course, including giving talks about this topic, explaining why we designed the system as we did, and encouraging users to opt in.

One person suggested making the system opt-in and then collecting tons more information, like the details of every command invocation, geolocation, and more. I think that would be a serious mistake. I would be happy to stand in front of a crowd and explain the current system and why users should opt in. I would be embarrassed to try to do the same for the more detailed telemetry this person suggested, because it seems indefensible: we simply don't need all that information to inform our decisions.

The Privacy Cost of Opt-In

Even with an ongoing campaign to have users opt in, we should expect that fewer systems will end up with telemetry enabled than if it were opt-out. I don't have a good sense of what level of opting in we should expect from an effective opt-in campaign. In part, this is because most systems can't run such a campaign (see the previous paragraph). However, in the past I have heard some people say that typical opt-in rates can be as low as a few percent.

I think we would be happy to get 10% and thrilled to get 20%, but I don't know whether that's possible. We don't even have a precise way to measure the opt-in rate, since by definition we can't see anything about people who have opted out. We have estimated the number of Go developers* at a few million, so seeing a few hundred thousand systems opted in would, I think, be a success.

Transparent telemetry uses sampling to reduce the privacy cost to any one system: we only need around 16,000 reports in a given week for 1% accuracy at a 99% confidence level. If there are a million systems reporting, which seemed plausible in the opt-out system, then any given system only needs to report 1.6% of the time, or less than once a year. On the other hand, if we'd be happy to get 100,000 systems opted in, then we need each system to report 16% of the time, or about once every month and a half. The opt-in system is fundamentally more invasive to any given installation than the opt-out system. Of course, by design it is still not terribly invasive, since uploaded reports do not contain any identifying information or any strings not already known to the collection system. But still, individual systems carry a larger burden when there are fewer, as there will be in an opt-in system.

Can We Still Make Good Decisions?

Thinking about even lower opt-in rates, will the system still able to provide data to help us making decisions? I think it will be, although the sampling rates will be higher.

A common mistake when thinking about polls and other sampled data collection is to confuse the fraction of samples (surveying, say, 1% of a population) with the accuracy of the result (making a claim with 1% accuracy). Those are different percentages. As we saw in "The Magic of Sampling, and its Limitations*", if there are a billion systems and we only sample 16,000 of them (0.0016%), we can still make claims with 1% accuracy, despite ignoring 99.9984% of the systems. Of course, there are not billions of Go installations (yet!). Assuming there are three million Go installations, as long as there is no correlation between whether a system is opted in and the value we want to measure, having even a 1% opt-in rate (30,000 systems available to report) should give us fairly accurate data. And of course as more systems opt in, each individual system will be sampled less often.

The "as long as there is no correlation" caveat deserves more scrutiny. What might be correlated with opting in to Go telemetry? Certainly the opt-in campaign methods will skew who opts in:

– People using the Go graphical installers will be more likely to opt in.
– People who read our blog and release notes will be more likely to opt in.
– People who take our surveys will be more likely to opt in.
– People who use VS Code Go will be more likely to opt in.

The next question is whether any of these would be correlated with and therefore skew metrics we are interested in, and how we would know. One obvious blind spot would be people installing Go through a package manager like `apt-get` or Homebrew. Perhaps the telemetry would undercount Linux and Mac. It might also overcount VS Code Go users. We can check many of these high-level machine demographics against the Go developer survey and others. These are biases to be aware of when interpreting the data, but I've been clear from the start that data is one input to the decision making process, not the determining factor. Some data is almost always better than no data: if the 10,000 systems you can see are all working properly, it's incredibly unlikely that there's a problem affecting even as many as 5% of all systems.

No Telemetry At All?

Before ending the post, I want to look at a couple other common suggestions. The first was to not have any telemetry at all: just use bug reports and surveys. Of all the people I saw make this comment, not one of them acknowledged reading the explanation of why those don't work* at the start of the first post.

Some people made a "slippery slope" argument that having any telemetry at all would make it easier to add more invasive telemetry: good telemetry opens the door to bad telemetry. When you examine this argument more closely, it doesn't hold up. The process for adding "bad telemetry" to Go would be the same whether or not there's already "good telemetry": modify the source code, get it checked in, and wait for it to go out in a new release. The defense strategy in either case is that Go is open source: anyone can inspect its source code, including recompiling it and verifying that the distributed binaries match that source code. Also, anyone can watch and participate in its development. If you are concerned about bad actors making changes, watch the development of Go, or trust others to watch it. If you don't believe enough people are watching, then you probably shouldn't use Go at all. You might reconsider other open-source software you use* too.

Cryptographic Approaches

The most interesting suggestion was to use a system like Prio* or Prochlo*. Prio in particular is being productionized by ISRG (the same group that runs Let's Encrypt) as Divvi Up*.

Those systems are not yet widely available, and perhaps it would make sense to use them in the future. Both depend on having two separate entities to run the system, with users trusting that the two will not collude to unmask their privacy. This kind of scheme provides enough privacy to allow collecting far more sensitive data, such as fine-grained location data or web page URLs. Luckily, for Go we have no need of anything that sensitive.

Another problem with these systems is that they are difficult to explain and inspect. The math behind them is solid, and if you're collecting that kind of sensitive data, you absolutely need something like them. But if, like in the Go design, you're not sending anything sensitive, then it could be better to use a simpler system that is easier to explain, so that people understand exactly what they are opting in to, and that is easier to watch, so that people can see exactly what information is being sent. People might well be more comfortable and more likely to opt in to a system they can more easily understand and verify.

Adopting a cryptographic schemes would therefore require significantly more effort and explanation for a relatively small privacy improvement; For now, it doesn't seem worth using one of those. If in the future one of those systems became a standard, widely accepted mechanism for open source telemetry, it would make sense to reexamine that decision.

Another possibility would be for operating system vendors to serve the role of collectors in one of these systems, taking care of collection from many programs running on the system instead. If done right, this could ensure greater privacy (provided users trust their operating system vendors!) instead of each system inventing a new wheel. If privacy-respecting telemetry collection became a standard operating system functionality, then it would absolutely make sense for Go to use that. Some day, perhaps.

Next Steps

Next week I intend to submit an actual Go proposal to add *opt-in* transparent telemetry to the Go toolchain. I have also added notes to the original blog posts mentioning the design change and pointing to this one.

It probably makes sense to prototype the system in `gopls`, the LSP server used by VS Code Go and other editors, to allow faster iteration. Once we are happy with the system, then it would make sense to add it to the standard Go toolchain as well, to enable the many use cases* in the earlier post.

Thanks to everyone who took the time to write constructive, helpful feedback. Those discussions are open source at its best.

* Asterisks mark hyperlinked text.