A Guide to Responsible Usage Data Collection In Open Source

•••

Avi Press
avi@scarf.sh
OSA Con, December 14, 2023

Quickly introducing myself

- In the past, I have found myself needing basic usage metrics on my own OSS projects.
- Today, I specifically work on OSS analytics as Founder & CEO of Scarf (https://scarf.sh)
- Most of the lessons in this talk are things I learned the hard way.
- Co-Host of the Hacking Open Source Business Podcast

Before we begin

- Please jump in with questions as we go!
- ***Nothing in this talk is legal advice. Talk to your legal team about your data collection policies.***
- Analytics in OSS is a (decreasingly) divisive subject. This talk aims to raise an overview considerations for those who find themselves *requiring* some form of analytics, and wanting to do it as responsibly as possible. Keep an open mind!
- Quick audience poll

Why would we do this in the first place?

How does your business answer hard questions about your OSS community?

- Where in the world are our users & community located geographically?
- How do our users find us?
- What does version adoption actually look like?
- Which companies are using our OSS? Which ones could we upsell this quarter?

Your OSS community is
your most important asset.

Your community is more than you might think

Traditional community interacts with you directly:

- Project collaborators
- Chat & forum discussions
- Events

Many people are missed this way

- Most importantly, your users!
  - Most of your OSS users will never engage with you directly.
- People reading your content, discussions, code without visibly engaging.

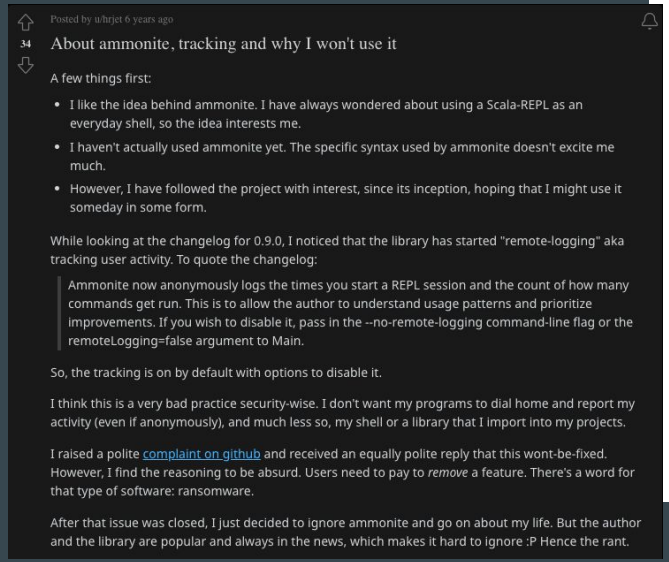
Surveys are not enough

- Most of your users will not fill one out.
- It’s hard to write a good survey. Bias creeps in everywhere.
- Stated preferences vs revealed preferences

Why is it hard to just track everything we might want to measure?

OSS users don’t want to be tracked

The general guidance for how to collect usage analytics in OSS: **Don’t**



Posted by u/hrjet 6 years ago

34

About ammonite, tracking and why I won't use it

A few things first:

- I like the idea behind ammonite. I have always wondered about using a Scala-REPL as an everyday shell, so the idea interests me.
- I haven't actually used ammonite yet. The specific syntax used by ammonite doesn't excite me much.
- However, I have followed the project with interest, since its inception, hoping that I might use it someday in some form.

While looking at the changelog for 0.9.0, I noticed that the library has started "remote-logging" aka tracking user activity. To quote the changelog:

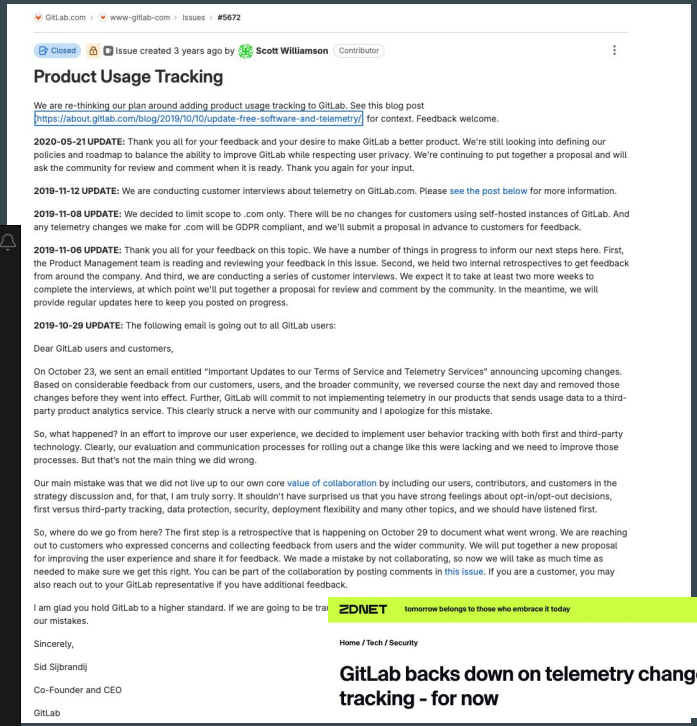
> Ammonite now anonymously logs the times you start a REPL session and the count of how many commands get run. This is to allow the author to understand usage patterns and prioritize improvements. If you wish to disable it, pass in the --no-remote-logging command-line flag or the remoteLogging=false argument to Main.

So, the tracking is on by default with options to disable it.

I think this is a very bad practice security-wise. I don't want my programs to dial home and report my activity (even if anonymously), and much less so, my shell or a library that I import into my projects.

I raised a polite complaint on github and received an equally polite reply that this wont-be-fixed. However, I find the reasoning to be absurd. Users need to pay to *remove* a feature. There's a word for that type of software: ransomware.

After that issue was closed, I just decided to ignore ammonite and go on about my life. But the author and the library are popular and always in the news, which makes it hard to ignore :P Hence the rant.



GitLab.com › www-gitlab-com › Issues › #5672

Closed · Issue created 3 years ago by **Scott Williamson** Contributor

Product Usage Tracking

We are re-thinking our plan around adding product usage tracking to GitLab. See this blog post https://about.gitlab.com/blog/2019/10/10/update-free-software-and-telemetry/ for context. Feedback welcome.

2020-05-21 UPDATE: Thank you all for your feedback and your desire to make GitLab a better product. We're still looking into defining our policies and roadmap to balance the ability to improve GitLab while respecting user privacy. We're continuing to put together a proposal and will ask the community for review and comment when it is ready. Thank you again for your input.

2019-11-12 UPDATE: We are conducting customer interviews about telemetry on GitLab.com. Please see the post below for more information.

2019-11-08 UPDATE: We decided to limit scope to .com only. There will be no changes for customers using self-hosted instances of GitLab. And any telemetry changes we make for .com will be GDPR compliant, and we'll submit a proposal in advance to customers for feedback.

2019-11-06 UPDATE: Thank you all for your feedback on this topic. We have a number of things in progress to inform our next steps here. First, the Product Management team is reading and reviewing your feedback in this issue. Second, we held two internal retrospectives to get feedback from around the company. And third, we are conducting a series of customer interviews. We expect it to take at least two more weeks to complete the interviews, at which point we'll put together a proposal for review and comment by the community. In the meantime, we will provide regular updates here to keep you posted on progress.

2019-10-29 UPDATE: The following email is going out to all GitLab users:

Dear GitLab users and customers,

On October 23, we sent an email entitled “Important Updates to our Terms of Service and Telemetry Services” announcing upcoming changes. Based on considerable feedback from our customers, users, and the broader community, we reversed course the next day and removed those changes before they went into effect. Further, GitLab will commit to not implementing telemetry in our products that sends usage data to a third-party product analytics service. This clearly struck a nerve with our community and I apologize for this mistake.

So, what happened? In an effort to improve our user experience, we decided to implement user behavior tracking with both first and third-party technology. Clearly, our evaluation and communication processes for rolling out a change like this were lacking and we need to improve those processes. But that's not the main thing we did wrong.

Our main mistake was that we did not live up to our own core value of collaboration by including our users, contributors, and customers in the strategy discussion and, for that, I am truly sorry. It shouldn't have surprised us that you have strong feelings about opt-in/opt-out decisions, first versus third-party tracking, data protection, security, deployment flexibility and many other topics, and we should have listened first.

So, where do we go from here? The first step is a retrospective that is happening on October 29 to document what went wrong. We are reaching out to customers who expressed concerns and collecting feedback from users and the wider community. We will put together a new proposal for improving the user experience and share it for feedback. We made a mistake by not collaborating, so now we will take as much time as needed to make sure we get this right. You can be part of the collaboration by posting comments in this issue. If you are a customer, you may also reach out to your GitLab representative if you have additional feedback.

I am glad you hold GitLab to a higher standard. If we are going to be tra
our mistakes.

Sincerely,

Sid Sijbrandij

Co-Founder and CEO

GitLab

ZDNET tomorrow belongs to those who embrace it today

Home / Tech / Security

GitLab backs down on telemetry changes and forced tracking - for now

But sometimes it’s fine?

Safety & Security

Personally Identifiable Information (PII)

“Any representation of information that permits the identity of an individual to whom the information applies to be reasonably inferred by either direct or indirect means.”

- *NIST, https://csrc.nist.gov/glossary/term/PII*

Primary question to answer: Are you going to be handling PII?

Storage of PII

- Avoiding storage of PII helps you avoid a variety of complications
  - You can still get a lot of value without PII! Anonymize wherever you can.
- The surface area matters. Minimize it! Don’t collect more than you need.
  - Storing less PII is better.
  - Storing it only temporarily is better.
- Encrypt at rest and in transit.
  - If you are tracking on, eg an IP address, salt your hashes

Compliance

- Abiding by GDPR is non trivial but it may be more achievable than you might think
  - You probably have a *legitimate basis* for collecting PII from OSS usage pertaining to your business (talk to your legal team)
  - If you use cookies, you need a cookie notice.
- Be thoughtful about 1st party vs 3rd party analytics.
  - Liability vs control tradeoffs.
  - Ensure any sub processors make the guarantees you need.
  - Data processor vs data controller

Consent

- Just allowing opt out may not be enough, you may need *multiple* ways to opt out.
  - Not every environment can easily add environment variables just to opt out.
  - Not every environment can easily make modifications to their package manifest just to opt out.
  - Not every environment is being operated by a human.
- Opt-in analytics sidestep the issue ***but sacrifice most of the value.***

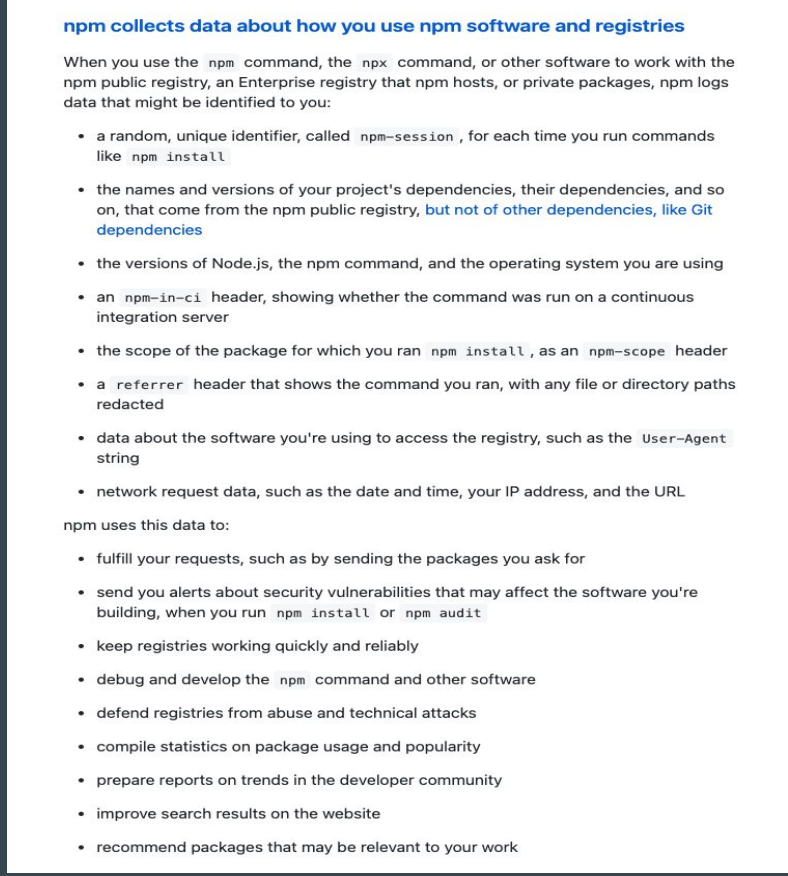
Ethics and Usage Data

The Common Question...

Is OSS usage data ethical to collect?

… Is No Longer a Practical Question

Much of this data is already being collected by *someone* (eg., artifact registries)



npm collects data about how you use npm software and registries

When you use the `npm` command, the `npx` command, or other software to work with the npm public registry, an Enterprise registry that npm hosts, or private packages, npm logs data that might be identified to you:

- a random, unique identifier, called `npm-session`, for each time you run commands like `npm install`
- the names and versions of your project's dependencies, their dependencies, and so on, that come from the npm public registry, but not of other dependencies, like Git dependencies
- the versions of Node.js, the npm command, and the operating system you are using
- an `npm-in-ci` header, showing whether the command was run on a continuous integration server
- the scope of the package for which you ran `npm install`, as an `npm-scope` header
- a `referrer` header that shows the command you ran, with any file or directory paths redacted
- data about the software you're using to access the registry, such as the `User-Agent` string
- network request data, such as the date and time, your IP address, and the URL

npm uses this data to:

- fulfill your requests, such as by sending the packages you ask for
- send you alerts about security vulnerabilities that may affect the software you're building, when you run `npm install` or `npm audit`
- keep registries working quickly and reliably
- debug and develop the `npm` command and other software
- defend registries from abuse and technical attacks
- compile statistics on package usage and popularity
- prepare reports on trends in the developer community
- improve search results on the website
- recommend packages that may be relevant to your work

npm collects personally identifiable information every time you use it.

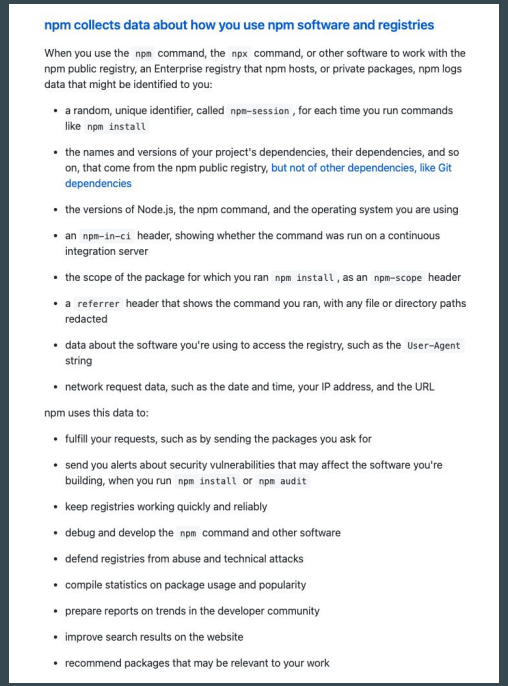
The More Modern Questions

Given that OSS usage data is already being collected, in general:

- Who should have access to this data?
- How should this data be used?
- How should it be stored?
- What compliance implications will this have downstream?
- How is consent managed?

People's expectations matter. A lot.

People are more tolerant of analytics that collect the data in a manner that matches their expectations.
Expectation Fit of analytics modalities will affect community response, even when the data being collected is the same.



npm collects data about how you use npm software and registries

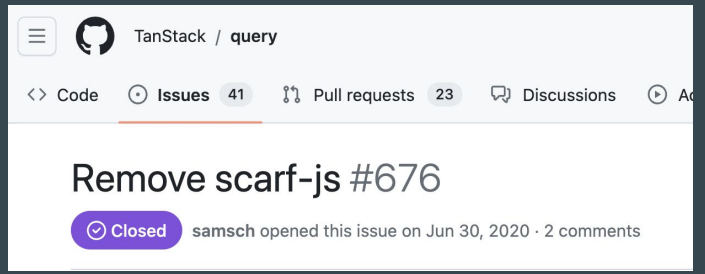
When you use the `npm` command, the `npx` command, or other software to work with the npm public registry, an Enterprise registry that npm hosts, or private packages, npm logs data that might be identified to you:

- a random, unique identifier, called `npm-session`, for each time you run commands like `npm install`
- the names and versions of your project's dependencies, their dependencies, and so on, that come from the npm public registry, but not of other dependencies, like Git dependencies
- the versions of Node.js, the npm command, and the operating system you are using
- an `npm-in-ci` header, showing whether the command was run on a continuous integration server
- the scope of the package for which you ran `npm install`, as an `npm-scope` header
- a `referrer` header that shows the command you ran, with any file or directory paths redacted
- data about the software you're using to access the registry, such as the `User-Agent` string
- network request data, such as the date and time, your IP address, and the URL

npm uses this data to:

- fulfill your requests, such as by sending the packages you ask for
- send you alerts about security vulnerabilities that may affect the software you're building, when you run `npm install` or `npm audit`
- keep registries working quickly and reliably
- debug and develop the `npm` command and other software
- defend registries from abuse and technical attacks
- compile statistics on package usage and popularity
- prepare reports on trends in the developer community
- improve search results on the website
- recommend packages that may be relevant to your work

High expectation fit:
first-party data collection
by a platform / product.



TanStack / query

Code | Issues 41 | Pull requests 23 | Discussions | Ac

Remove scarf-js #676

Closed **samsch** opened this issue on Jun 30, 2020 · 2 comments

Low expectation fit:
postInstall hook from
package

SIGN IN / UP — The Register

SOFTWARE 57

Fedora Project mulls 'privacy preserving' usage telemetry

Presumably Red Hat feels it hasn't alienated enough people recently

Liam Proven — Mon 10 Jul 2023 / 19:00 UTC

Low expectation fit:
Violating solidified
project norms

Set your community's expectations

- Suddenly adding lots of analytics into a project that previously didn't have it can be problematic.
  - But not necessarily!
- Being clear about your project goals align your community and prevent surprises.
- There's nothing wrong with aiming to commercialize an OSS project! Be upfront, manage expectations and avoid bad surprises.
- Expectations are a moving target.

Methods of data collection

- Server access logs. Your users are fetching your software from *somewhere.*
  - Package downloads
  - Image / pixel downloads
  - *High expectation fit*
- Telemetry
  - Version checks / heartbeats
  - Event logging
  - Tracing
  - *Variable expectation fit*

Transparency is good but not enough

- Be clear in what you collect and and why you collect it.
  - Be clear on **when** you collect it, as this directly impacts your expectation fit.
- Align your business with your community. How does being informed on usage help your users?
- People don't want to be spammed, have their info sold, etc.
- Transparency does not ensure a positive response (eg, GitLab, Fedora).

Access

- Consider what data, if any, you are going to publish publicly.
- Sharing your insights back with your community is a great way to build trust.

Wrapping Up

- OSS usage analytics *can* be done safely and ethically when done intention and care against many different considerations.
- Understand, manage, and fit into your users' expectations. Analytics with **high expectation fit** will be met with **significantly better reception** than with **lower expectation fit**

Thank you!

Avi Press

avi@scarf.sh

https://scarf.sh

GitHub: @aviaviavi

Twitter: @avi_press

Podcast: Hacking Open Source Business