RFC 358: Private code data philosophy

Editor: ericbm@sourcegraph.com
Status: APPROVED
Requested reviewers (please review by EOD 2021-04-22): Bill Creager
Approvals: Joe Chen Dan Adler Christina Forney
Team: Core application

Background

Private code is coming to Sourcegraph Cloud!

Problem

Sourcegraph pivoted towards on-prem solutions in part for organizations to feel secure in connecting our product to their code base. In this blog post explaining the switch, we noted "many of our customers have high security requirements and don't feel comfortable uploading their private code to an external cloud software provider. They prefer a solution that can run on their own infrastructure, under their own security terms and authorization mechanisms, with full control over who has access".

While customers and prospects likely acknowledge we'll collect more data on their product usage on the Cloud product (this is a common practice, but I have no specific evidence orgs expect this from us specifically), we *do not* want to collect any information about specifics of a company's code base. Among many reasons we take this approach are to maintain the trust of customers, prospects and the ecosystem at large that we don't see their sensitive data and apply best security practices across the board to limit the risk of hosting private code on Cloud.

We *do*, however, want to collect more data than we currently get from on-prem deployments to make better decisions. And decoupling the information we want from what we don't want requires a strategy for what to collect, and then code to be written to implement it!

Proposal

We collect event/org/user-level data from individuals and organizations with private code on Cloud, but ***never*** any sensitive data. This is generally aligned with other cloud tools, such as GitHub, GitLab, Atlassian and others. Any data point or metadata of a data point that could reveal information about private code, such as a search or a URL that an action took place on, will never be collected.

For reference, the pings philosophy for self hosted instances is the data needs to be anonymous, aggregated (e.g. the number of users that churned from one month to the next,

instead of the individual users that churned) and non-specific (e.g. no repo names, no usernames, no file names, no specific search queries, etc.).

The Sourcegraph Cloud data philosophy is **only non-specific**. It will not be anonymous or aggregated. This applies to both what we want to ingest into our analytics pipeline and the user data that Sourcegraph teammates should have access to.

The following is a mostly exhaustive list of the different data types.

Data type	What we want to ingest in our analytics pipeline (**table names are in bold**)	What it's used for
Events	**event_logs**, but exclude all metadata (argument, URL) that would potentially expose private information. This means we would collect timestamp, user_id, name, source	Retention, activation, usage patterns, feature usage, search success percentages, etc…(so much opportunity with this data)
Users	**users** (id, username, created_at, deleted_at, updated_it, site_admin) and **user_emails** (user_id, email)	Correlation of cohorts and different user characteristics to usage patterns, emails for onboarding and drip email campaigns
Org	**org**, **org_invitations**, **org_members**, **external_services** without any sensitive information	Org expansion, future referral program(?), code host connections
Features	**saved_searches**, **batchchanges** (and any other feature tables) without any sensitive information	Feature usage and correlations with adoption/usage
Pings	It would be great if we could still collect pings from organizations on Cloud so we don't have to maintain two different sources of aggregated data	Aggregated information we care about

Definition of success

This proposal does not address any implementation of Cloud data implementation or security best practices around how the sensitive data should be handled; these details will be left up to our amazing engineering teams :)

The associated work (TBD) with the execution of this RFC is completed in Milestone 5 of the Core applications team (as Tomás Senart noted in Slack).