



Christian Burkert, Johanna Ansohn McDougall, Hannes Federrath

Data Minimisation Potential for Timestamps in Git: An Empirical Analysis of User Configurations

Agenda

Timestamp Data Minimisation (in Git)

- Why do we need it?
- What is necessary timestamp data?
- What is the impact of minimisation on user privacy?

You have probably used Git



```
configitour — christian@MacBook-Pro
..er/configitour
→ configitour git:(master) ✗ git add paper.tex
→ configitour git:(master) ✗ git commit -m "Finalize camera ready"
```

Git is by far the most popular SCM/VCS

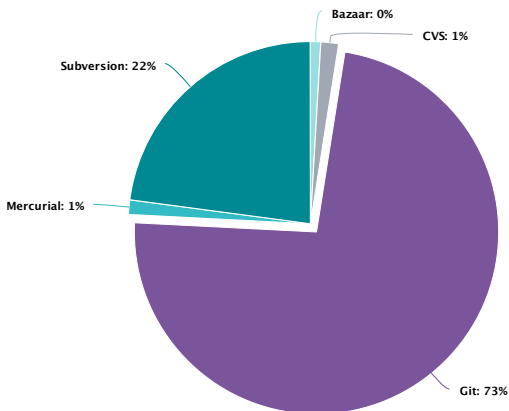


Figure: Git repos make up 73% of the projects indexed by OpenHub

Source: <https://www.openhub.net/repositories/compare>

Timestamp in Git

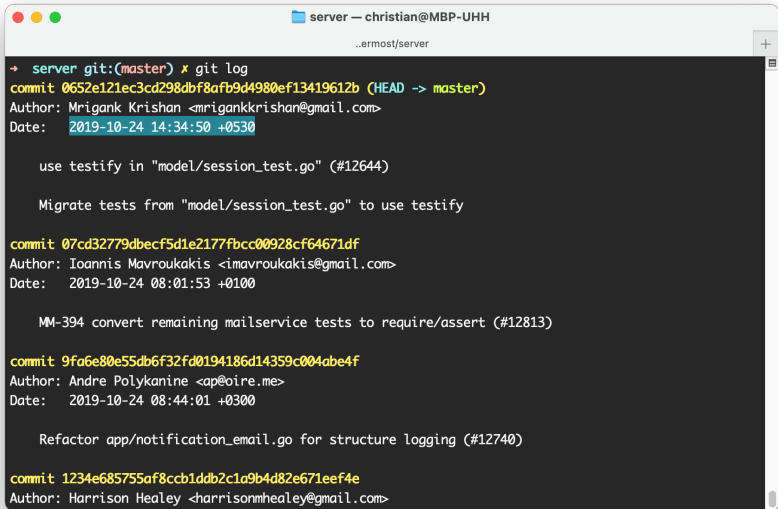
```
1 # git cat-file -p 4a3c5d8119c25804ef94521688fa277a83e0beec
2 tree 0601d4dd701fbefdcdf69c9a9cf71a3b61cfc28
3 parent 36d60dfe2c66dfdb09767e11511c9d7e6483202d
4 author Chris Lattner <clattner@apple.com> 1485549958 -0800
5 committer Chris Lattner <clattner@apple.com> 1485549958 -0800
6
7 Joe is taking over as the overall maintainer of Swift, thanks.
```

Timestamp as User Info in Git

The screenshot shows the GitHub interface for the repository `EMPRI-DEVOPS / git-privacy`. The commit history is displayed under the `master` branch. Several commit entries are visible, with their timestamps highlighted in red boxes:

- Commits on **Apr 12, 2021**
 - `Fix clone tests' environment and typos` by `cburkert` committed 19 days ago ✓ (SHA: `c6c5c40`)
 - `Handle changed stdin input for pre-push hook. No check when forced.` by `cburkert` committed 19 days ago ✗ (SHA: `8bc1de9`)
- Commits on **Apr 11, 2021**
 - `Unittest with Github Action` by `cburkert` committed 20 days ago ✓ (SHA: `aa7ccaa`)
- Commits on **Aug 30, 2020**
 - `Code cleanup` by `cburkert` committed on 31 Aug 2020 ✓ (SHA: `788485b`)

Timestamp as User Info in Git



```
server — christian@MBP-UHH
..ermost/server

→ server git:(master) * git log
commit 0652e121ec3cd298dbf8afb9d4980ef13419612b (HEAD -> master)
Author: Mrigank Krishan <mrigankkrishan@gmail.com>
Date: 2019-10-24 14:34:50 +0530

    use testify in "model/session_test.go" (#12644)

    Migrate tests from "model/session_test.go" to use testify

commit 07cd32779dbecf5d1e2177bcc00928cf64671df
Author: Ioannis Mavroukakis <imavroukakis@gmail.com>
Date: 2019-10-24 08:01:53 +0100

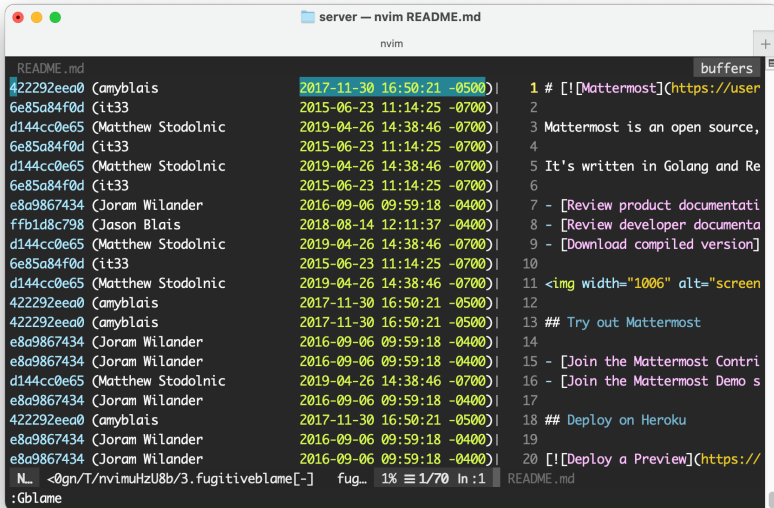
    MM-394 convert remaining mailservice tests to require/assert (#12813)

commit 9fa6e80e55db6f32fd0194186d14359c004abe4f
Author: Andre Polykanine <ap@oire.me>
Date: 2019-10-24 08:44:01 +0300

    Refactor app/notification_email.go for structure logging (#12740)

commit 1234e685755af8ccb1ddb2c1a9b4d82e671eef4e
Author: Harrison Healey <harrisonmhealey@gmail.com>
```

Timestamp as User Info in Git



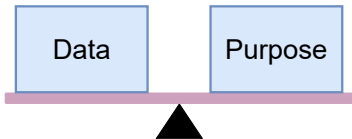
The screenshot shows a vim editor window titled "server - nvim README.md". The editor is displaying a git log for the file README.md. The log entries are listed in a table-like format with columns for commit hash, author, date, time, and offset. The date and time fields are highlighted in yellow. The log content is as follows:

```
README.md buffers
422292eea0 (amyblais 2017-11-30 16:50:21 -0500) | 1 # [![Mattermost](https://user
6e85a84f0d (it33 2015-06-23 11:14:25 -0700) | 2
d144cc0e65 (Matthew Stodolnic 2019-04-26 14:38:46 -0700) | 3 Mattermost is an open source,
6e85a84f0d (it33 2015-06-23 11:14:25 -0700) | 4
d144cc0e65 (Matthew Stodolnic 2019-04-26 14:38:46 -0700) | 5 It's written in Golang and Re
6e85a84f0d (it33 2015-06-23 11:14:25 -0700) | 6
e8a9867434 (Joram Wilander 2016-09-06 09:59:18 -0400) | 7 - [Review product documentati
ffb1d8c798 (Jason Blais 2018-08-14 12:11:37 -0400) | 8 - [Review developer documenta
d144cc0e65 (Matthew Stodolnic 2019-04-26 14:38:46 -0700) | 9 - [Download compiled version]
6e85a84f0d (it33 2015-06-23 11:14:25 -0700) | 10
d144cc0e65 (Matthew Stodolnic 2019-04-26 14:38:46 -0700) | 11 <img width="1006" alt="screen
422292eea0 (amyblais 2017-11-30 16:50:21 -0500) | 12
422292eea0 (amyblais 2017-11-30 16:50:21 -0500) | 13 ## Try out Mattermost
e8a9867434 (Joram Wilander 2016-09-06 09:59:18 -0400) | 14
e8a9867434 (Joram Wilander 2016-09-06 09:59:18 -0400) | 15 - [Join the Mattermost Contri
d144cc0e65 (Matthew Stodolnic 2019-04-26 14:38:46 -0700) | 16 - [Join the Mattermost Demo s
e8a9867434 (Joram Wilander 2016-09-06 09:59:18 -0400) | 17
422292eea0 (amyblais 2017-11-30 16:50:21 -0500) | 18 ## Deploy on Heroku
e8a9867434 (Joram Wilander 2016-09-06 09:59:18 -0400) | 19
e8a9867434 (Joram Wilander 2016-09-06 09:59:18 -0400) | 20 [![Deploy a Preview](https://
N_ <0gn/T/nvimHzU8b/3.fugitiveblame[-] fug... 1% 1/70 In :1 README.md
:Gblame
```


Background: Data Minimisation

Definition (Data Minimisation)

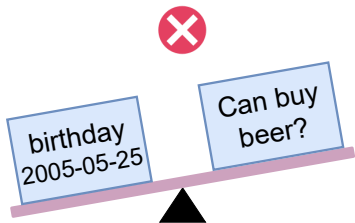
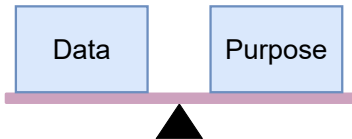
Personal data shall be adequate, relevant and limited to what is **necessary in relation to the purposes** for which they are processed. (Article 5 GDPR)



Background: Data Minimisation

Definition (Data Minimisation)

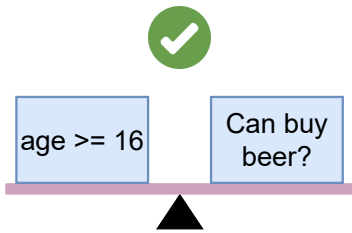
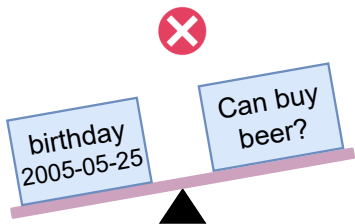
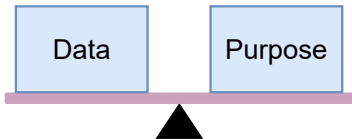
Personal data shall be adequate, relevant and limited to what is **necessary in relation to the purposes** for which they are processed. (Article 5 GDPR)



Background: Data Minimisation

Definition (Data Minimisation)

Personal data shall be adequate, relevant and limited to what is **necessary in relation to the purposes** for which they are processed. (Article 5 GDPR)



Goal: Understand Potential for Data Minimisation



```
1 # git cat-file -p 4a3c5d8119c25804ef94521688fa277a83e0beec
2 tree 0601d4dd701fbefdcdf69c9a9cf71a3b61cfcfd28
3 parent 36d60dfe2c66dfdb09767e11511c9d7e6483202d
4 author Chris Lattner <clattner@apple.com> 1485549958 -0800
5 committer Chris Lattner <clattner@apple.com> 1485549958 -0800
6
7 Joe is taking over as the overall maintainer of Swift, thanks.
```

Background: Customisability of Git's Date Handling

Git's Presentation is Customisable

```
server — christian@MBP-UHH
..ermost/server

→ server git:(master) x git log
commit 0652e121ec3cd298dbf8afb9d4980ef13419612b (HEAD -> master)
Author: Mrigank Krishan <mrigankkrishan@gmail.com>
Date: 2019-10-24 14:34:50 +0530

    use testify in "model/session_test.go" (#12644)

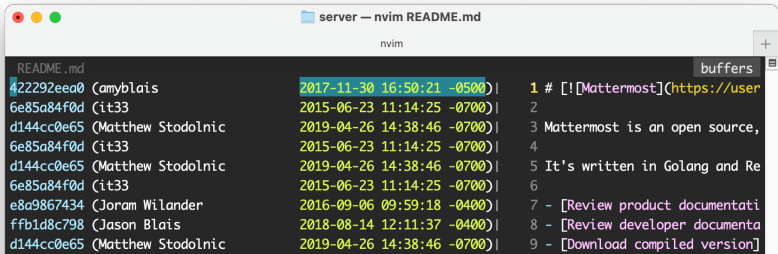
    Migrate tests from "model/session_test.go" to use testify

commit 07cd32779dbecf5d1e2177fbcc00928cf64671df
```

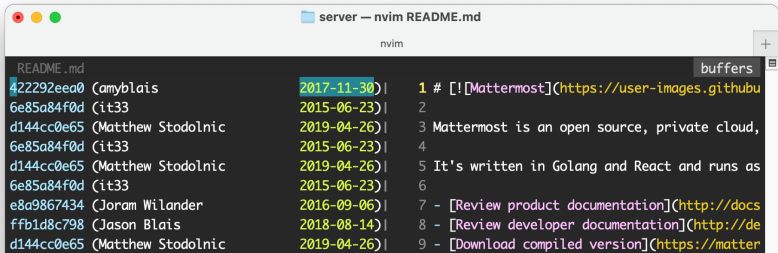
```
server — christian@MBP-UHH
..ermost/server

→ server git:(master) x git log --pretty="%h (%cr) %cn%n%w(80,10,10)%s"
0652e121e (1 year, 7 months ago) George Goldberg
    use testify in "model/session_test.go" (#12644)
07cd32779 (1 year, 7 months ago) Ben Schumacher
    MM-394 convert remaining mailservice tests to require/assert (#12813)
9fa6e80e5 (1 year, 7 months ago) Ben Schumacher
    Refactor app/notification_email.go for structure logging (#12740)
1234e6857 (1 year, 7 months ago) GitHub
    MM-19574 Don't return metadata for deleted posts (#12878)
23aa193f2 (1 year, 7 months ago) Ben Schumacher
```

Git's Presentation is Customisable



```
server -- nvim README.md
nvim
buffers
README.md
422292eea0 (amyblais 2017-11-30 16:50:21 -0500) | 1 # [!][Mattermost](https://user
6e85a84f0d (it33 2015-06-23 11:14:25 -0700) | 2
d144cc0e65 (Matthew Stodolnic 2019-04-26 14:38:46 -0700) | 3 Mattermost is an open source,
6e85a84f0d (it33 2015-06-23 11:14:25 -0700) | 4
d144cc0e65 (Matthew Stodolnic 2019-04-26 14:38:46 -0700) | 5 It's written in Golang and Re
6e85a84f0d (it33 2015-06-23 11:14:25 -0700) | 6
e8a9867434 (Joram Wilander 2016-09-06 09:59:18 -0400) | 7 - [Review product documentati
ffb1d8c798 (Jason Blais 2018-08-14 12:11:37 -0400) | 8 - [Review developer documenta
d144cc0e65 (Matthew Stodolnic 2019-04-26 14:38:46 -0700) | 9 - [Download compiled version]
```



```
server -- nvim README.md
nvim
buffers
README.md
422292eea0 (amyblais 2017-11-30) | 1 # [!][Mattermost](https://user-images.githubu
6e85a84f0d (it33 2015-06-23) | 2
d144cc0e65 (Matthew Stodolnic 2019-04-26) | 3 Mattermost is an open source, private cloud,
6e85a84f0d (it33 2015-06-23) | 4
d144cc0e65 (Matthew Stodolnic 2019-04-26) | 5 It's written in Golang and React and runs as
6e85a84f0d (it33 2015-06-23) | 6
e8a9867434 (Joram Wilander 2016-09-06) | 7 - [Review product documentation](http://docs
ffb1d8c798 (Jason Blais 2018-08-14) | 8 - [Review developer documentation](http://de
d144cc0e65 (Matthew Stodolnic 2019-04-26) | 9 - [Download compiled version](https://matter
```

Background: Git Options relating Timestamps

Table: Date-related features in Git.

Option	Description
<code>--pretty [=<format>]</code>	Custom object output format
<code>--date=<format></code>	Change date output format
<code>--until=<date></code>	Only show objects until date
<code>--since=<date></code>	Only show objects since date

Background: Git Options relating Timestamps

Table: Date-related features in Git.

Option	Description
<code>--pretty[=<format>]</code>	Custom object output format
<code>--date=<format></code>	Change date output format
<code>--until=<date></code>	Only show objects until date
<code>--since=<date></code>	Only show objects since date

Table: Support for date-related features in Git subcommands.

	annotate	blame	log	shortlog	show	...
<code>--date</code>	✓ [†]	✓	✓		✓*	
<code>--pretty</code>			✓		✓	
<code>--since/--until</code>		✓	✓	✓	✓*	

*=undocumented, †=disfunctional

Background: Git's Date Formatting Options (`--date`)

Customisation of the date formatting can also influence precision.

Name	Suffix	Precision	Example(s)
default	-	sec	Wed Sep 22 14:57:31 2021 +0200
human	-	day to	Sep 21 2021
		sec	7 seconds ago
iso	i/l	sec	2021-09-22 14:57:31 +0200
raw	-	sec	1632315451 +0200
relative	r	year to	7 years ago
		sec	7 seconds ago
rfc	D	sec	Wed, 22 Sep 2021 14:57:31 +0200
short	s	day	2021-10-04
unix	t	sec	1632315451

Background: Git's Pretty Formatting Options (`--pretty`)

Allows customisation of

- included date types (author, commit, both, none)
- date formatting (like with `--date`).

Background: Git's Pretty Formatting Options (`--pretty`)

Allows customisation of

- included date types (author, commit, both, none)
- date formatting (like with `--date`).

```
1 $ git log --pretty=fuller
2 commit e71a7ef2090484177c75726b4e92a5d8945f0fb2
3 Author:      Christian Burkert <burkert@example.com>
4 AuthorDate:  2021-05-20 12:00:01 +0200
5 Commit:      John Doe <doe@example.com>
6 CommitDate:  2021-05-22 00:00:00 +0500
7
8     I'm a commit message headline
9
```

Background: Git's Pretty Formatting Options (`--pretty`)

Allows customisation of

- included date types (author, commit, both, none)
- date formatting (like with `--date`).

```
1 $ git log --pretty=fuller
2 commit e71a7ef2090484177c75726b4e92a5d8945f0fb2
3 Author:      Christian Burkert <burkert@example.com>
4 AuthorDate:  2021-05-20 12:00:01 +0200
5 Commit:      John Doe <doe@example.com>
6 CommitDate:  2021-05-22 00:00:00 +0500
7
8     I'm a commit message headline
9
10 $ git log --pretty="%h <%an> (%cr)%n%w(80,8,8)%s"
11 e71a7ef <Christian Burkert> (5 days ago)
12     I'm a commit message headline
```

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example: 1 hour 30 minutes ago | yesterday 5pm

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example:

1 hour 30 minutes ago
hour minute

yesterday 5pm

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example:

1 hour 30 minutes ago
hour minute

yesterday 5pm
day hour

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example: 1 hour 30 minutes ago
 hour minute

Precisions: hour < minute

yesterday 5pm
 day hour

day < hour

Background: Git's Output Filtering Options (`--until` / `--since`)

Filtering allows temporal limitation of output to

- objects since/until a given filter date
- specified in a natural (very lenient) format

Filter dates convey information about date precision demands.

Example: 1 hour 30 minutes ago
 hour minute

Precisions: hour < minute

Highest: minute

yesterday 5pm
 day hour

day < hour

hour

Background: Git Configuration Files

Customisations can also be made in config files (e.g., ~/.gitconfig).

```
1 [alias]
2 lf = log --pretty=fuller --date=iso
3 hi = log --pretty="%h <%an>%d (%cr)%n%w(80,8,8)%s"
4 bs = blame --date=short
5 sl = shortlog --since="1 day ago"
6
```

Background: Git Configuration Files

Customisations can also be made in config files (e.g., ~/.gitconfig).

```
1 [alias]
2 lf = log --pretty=fuller --date=iso
3 hi = log --pretty="%h <%an>%d (%cr)%n%w(80,8,8)%s"
4 bs = blame --date=short
5 sl = shortlog --since="1 day ago"
6
7 [pretty]
8 my = %h %an (%ar)
9
```

Background: Git Configuration Files

Customisations can also be made in config files (e.g., `~/.gitconfig`).

```
1 [alias]
2   lf = log --pretty=fuller --date=iso
3   hi = log --pretty="%h <%an>%d (%cr)%n%w(80,8,8)%s"
4   bs = blame --date=short
5   sl = shortlog --since="1 day ago"
6
7 [pretty]
8   my = %h %an (%ar)
9
10 [format]
11   pretty = my
12 [log]
13   date = iso
14 [blame]
15   date = short
```

Evaluation

Empirical Analysis of Git Config Files

Guiding Are second-precision timestamps in Git necessary?

Empirical Analysis of Git Config Files

Guiding Are second-precision timestamps in Git necessary?

Dataset GitHub hosts over 20 000 config files that contain alias definitions

Empirical Analysis of Git Config Files

Guiding Are second-precision timestamps in Git necessary?

Dataset GitHub hosts over 20 000 config files that contain alias definitions

RQ 1. What precision demands result from user customisations?

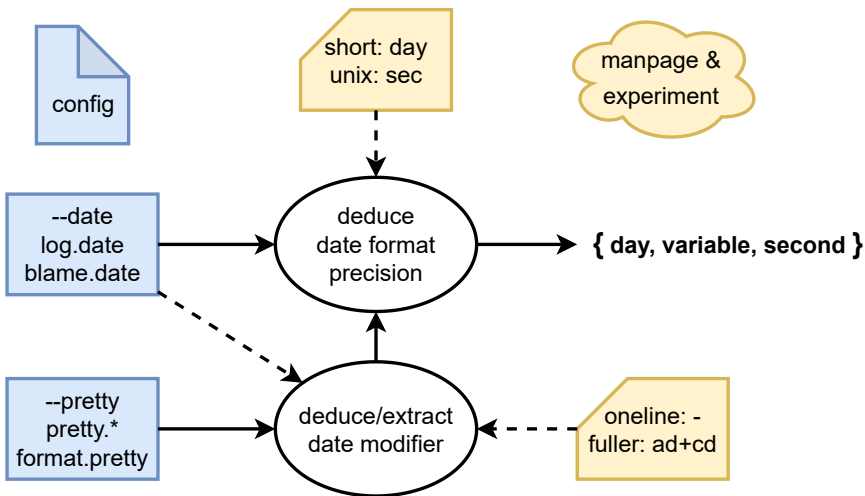
Empirical Analysis of Git Config Files

Guiding Are second-precision timestamps in Git necessary?

Dataset GitHub hosts over 20 000 config files that contain alias definitions

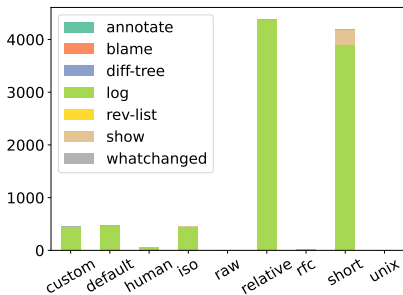
- RQ**
1. What precision demands result from user customisations?
 2. Would a small to moderate precision reduction have an impact on user privacy?

Process of Precision Evaluation

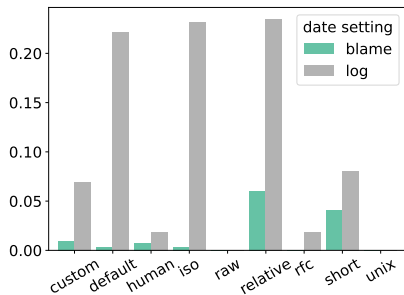


Results for Date Formatting (--date)

- Popularity of date formats depends on context (see *short*)
- Format *relative* popular in all contexts



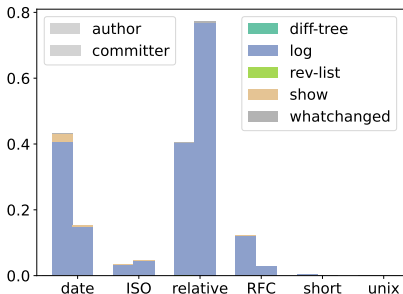
(a) used with `--date` in command aliases



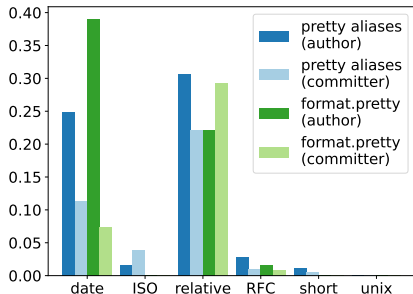
(b) used for config settings

Results for Pretty Formatting (`--pretty`)

- ~40% of formats use neither date type
- otherwise *date* and *relative* are popular

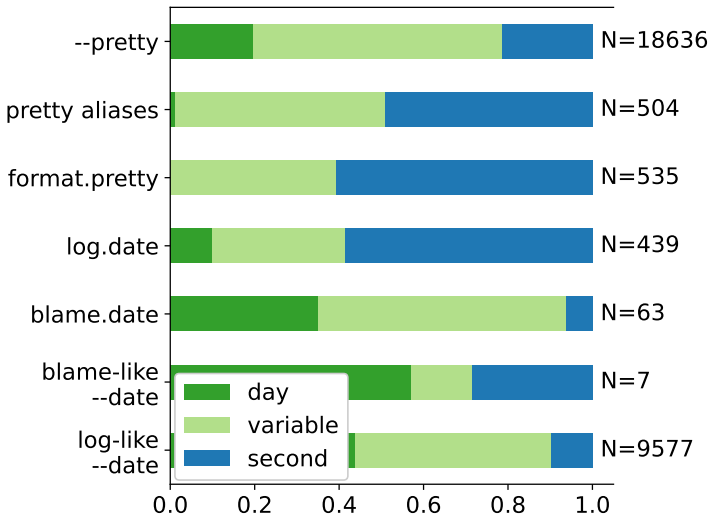


(a) custom formats in subcommand aliases

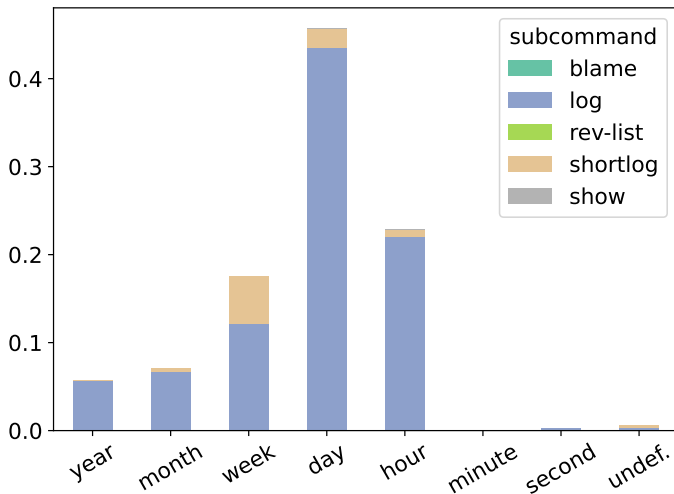


(b) format aliases and `format.pretty`

Results: Date Precision in Output Formatting (Overview)



Results: Precision in Filters



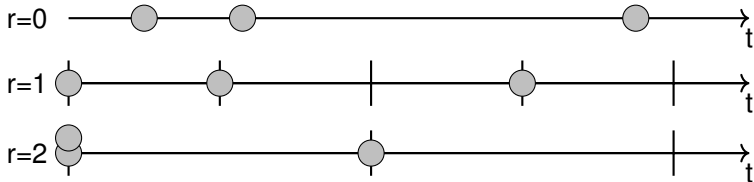
RQ 2: Assessing the Privacy Impact of Precision Reduction

RQ Would a small to moderate precision reduction have an impact on user privacy?

RQ 2: Assessing the Privacy Impact of Precision Reduction

RQ Would a small to moderate precision reduction have an impact on user privacy?

Assumption Making chronologically adjacent activities temporally indistinguishable reduces privacy impact

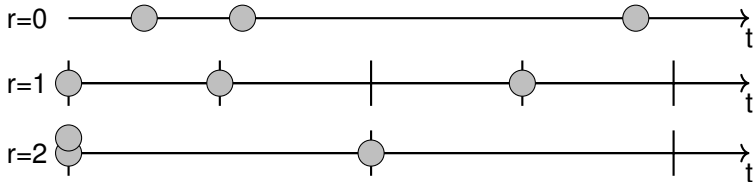


RQ 2: Assessing the Privacy Impact of Precision Reduction

RQ Would a small to moderate precision reduction have an impact on user privacy?

Assumption Making chronologically adjacent activities temporally indistinguishable reduces privacy impact

Dataset GitHub snapshot with 360 million commits of medium-active users (1k to 10k total commits)



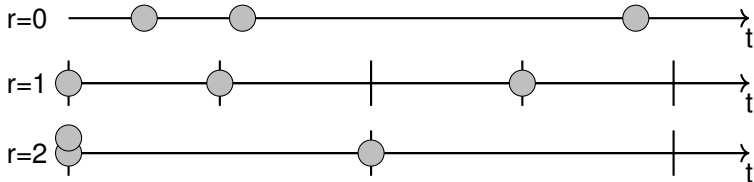
RQ 2: Assessing the Privacy Impact of Precision Reduction

RQ Would a small to moderate precision reduction have an impact on user privacy?

Assumption Making chronologically adjacent activities temporally indistinguishable reduces privacy impact

Dataset GitHub snapshot with 360 million commits of medium-active users (1k to 10k total commits)

Variable Ratio of a user's temporally distinguishable activities depending on precision reduction level



RQ 2: Assessing the Privacy Impact of Precision Reduction

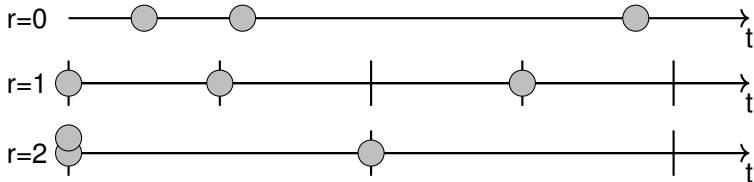
RQ Would a small to moderate precision reduction have an impact on user privacy?

Assumption Making chronologically adjacent activities temporally indistinguishable reduces privacy impact

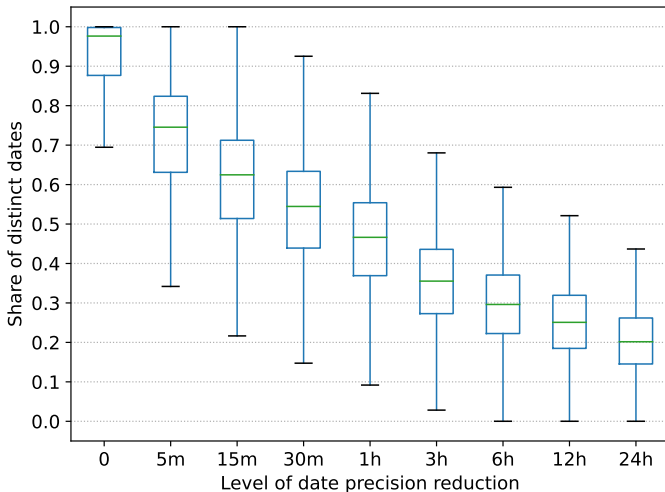
Dataset GitHub snapshot with 360 million commits of medium-active users (1k to 10k total commits)

Variable Ratio of a user's temporally distinguishable activities depending on precision reduction level

Result Ratio of distinguishable activities declines fast:
Precision 1h: <50% distinguishable (median)



RQ 2: Assessing the Privacy Impact of Precision Reduction II



Limitations

- Git configs as indicators for precision demand
 - Biased towards desire for non-default behaviour?

Limitations

- Git configs as indicators for precision demand
 - Biased towards desire for non-default behaviour?
 - Biased towards more experienced users (discoverability)?

Limitations

- Git configs as indicators for precision demand
 - Biased towards desire for non-default behaviour?
 - Biased towards more experienced users (discoverability)?
 - Format might be chosen counter to demand (e. g., ISO)

Limitations

- Git configs as indicators for precision demand
 - Biased towards desire for non-default behaviour?
 - Biased towards more experienced users (discoverability)?
 - Format might be chosen counter to demand (e. g., ISO)
- Privacy impact assessment
 - Ratio of distinguishable time points only considers risks of direct interval inference,
i. e., how long did it take between X and Y?

Summary: Findings & Contribution

We contributed empirical evidence on

- timestamp precision demand (in Git)
 - analysed >20 000 public Git configs
 - <10% of `--date` in aliases use seconds
 - ~10% of `--pretty` in aliases use sec
 - ~0.5% of filters use min or sec
 - published dataset of config features

Summary: Findings & Contribution

We contributed empirical evidence on

- timestamp precision demand (in Git)
 - analysed >20 000 public Git configs
 - <10% of `--date` in aliases use seconds
 - ~10% of `--pretty` in aliases use sec
 - ~0.5% of filters use min or sec
 - published dataset of config features
- privacy impact of precision reduction
 - analysed >360 million commits
 - distinguishable: 75% at 5min, 50% at 1h

Summary: Findings & Contribution

We contributed empirical evidence on

- timestamp precision demand (in Git)
 - analysed >20 000 public Git configs
 - <10% of `--date` in aliases use seconds
 - ~10% of `--pretty` in aliases use sec
 - ~0.5% of filters use min or sec
 - published dataset of config features
- privacy impact of precision reduction
 - analysed >360 million commits
 - distinguishable: 75% at 5min, 50% at 1h



[https://github.com/
EMPRI-DEVOPS/
git-privacy](https://github.com/EMPRI-DEVOPS/git-privacy)

Developed `git-privacy` for timestamp redaction

Contact

Christian Burkert

Tel. +49 40 42883-2406

Mail christian.burkert@uni-hamburg.de

I'd be happy to hear from you!



OpenPGP Fingerprint:

9B97 CC4B 5FF4 7BA3 EF7B 1966 A5FB 6E0B 41AC CDFB