Destructive Repository Alterations: a Threat to The Open Source Software Supply Chain

Stefano Zacchiroli

Institut Polytechnique de Paris & Software Heritage
stefano.zacchiroli@telecom-paris.fr

17 November 2026 — European Cyber Week, Rennes



Software Heritage
THE GREAT LIBRARY OF SOURCE CODE



Outline

1. Introduction
2. Rewritten Git histories
3. Mutated Git tags
4. Conclusion



About the speaker

- Professor of Computer Science, Télécom Paris, Institut Polytechnique de Paris
- Software Heritage co-founder & Chief Scientific Officer (CSO)
- Co-Principal Investigator (Co-PI), Software Heritage Security (SWHSec) project
- Free/Open Source Software activist (20+ years); Debian Developer & former 3x Debian Project Leader; former OSI director

Today's agenda

1. Open source is a strategic dependency, and its supply chain is under attack
2. Rewritten Git histories: how often, where, what (ASE 2025)
3. Mutated Git tags: the "immutable" release anchors that move (ACM REP 2026)

Open source security, policy

With great exposure comes great scrutiny…

… by both good and bad actors

- OSS is more and more targeted by attackers: *supply chain attacks*
  - event-stream (2018) · Log4Shell (2021) · node-ipc (2022) · XZ utils (2024)
- Increased policy attention to secure OSS, e.g.:
  - US: Biden's executive orders (2022, Jan 2025!)
  - EU: Cyber Resilience Act (CRA), progressively coming into effect
    - open source compliance work organised in the ORC WG



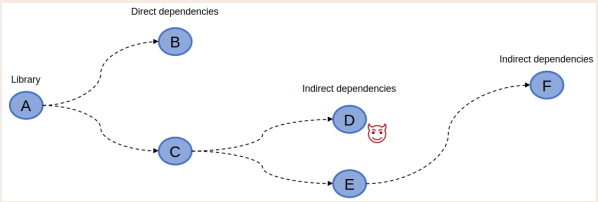


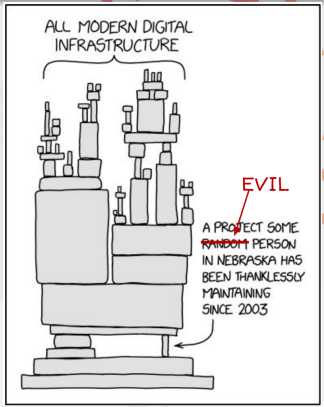


Software supply chain attacks

Reusing OSS via dependencies

- Software dependencies: a popular way of reusing open source software.
- Software product A uses functionalities implemented in OSS product B … and so on.





based on xkcd.com/2347

Attacking the software supply chain

- Attacking undermaintained "leaf" packages (e.g., D) → efficient attack strategy
- Many documented attacks: event-stream (2018), node-ipc (2022), XZ utils (2024), …

Outline



1. Introduction
2. Rewritten Git histories
3. Mutated Git tags
4. Conclusion

Git repository alterations

Git allows rewriting history (of the version control kind!)

```
$ git rebase --interactive <...>
$ git push --force
```



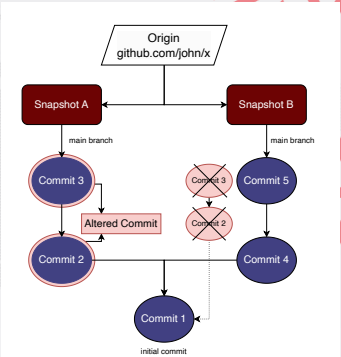
- Useful feature! E.g., to clean your code before sharing it
- Also annoying and risky on public branches
  - Hinders reproducibility and voids availability of specific Git objects
  - Supply chain concerns: what was altered and why?

1. How often are public Git histories altered?
2. When this happens, what is changed and why?

Note: forges do not keep the history of these modifications. SWH is the only place where they can be analyzed globally.

A large-scale analysis of Git repository alterations

1. Retrieve from Software Heritage 111 M Git repositories (1) archived at least twice, (2) with different states ("snapshots")
2. For each repository, compare snapshots 2-by-2 to detect altered histories and the root cause of each alteration
3. Classify altered commits by what changed before/after alteration



Solal Rapaport, Laurent Pautet, Samuel Tardieu, Stefano Zacchiroli
Altered Histories in Version Control System Repositories: Evidence from the Trenches
ASE 2025 https://arxiv.org/abs/2509.09294

Key findings on destructive repository alterations

How often?

- 1.22M repositories contain altered histories (~1.1%)
- 8.7M altered commits → Pro tip: make sure your important commits are in SWH!

Where?

- Pull request branches: 37.6% → Might be OK, dependening on the workflow
- main/master branches: 11.4% → Concerning!

What?

- Commit metadata (13.3%): author, date, message, … → Risky for provenance & IP
- File/dir. changes (76.8%): *retroactive* file modifications and/or deletions

Case studies: licenses and secrets

Case study #1: License changes

- ~800K retroactively altered license files on main branches
- Spanning 32k repositories (76 with 1000+ stars)
- 79% version updates (e.g., GPL 2→3)
- 14% full changes (e.g., MIT→GPL)
- Serious concern: retroactive changes may *de facto* suppress previously granted rights (without an archival copy!)

Case study #2: Removing secrets

- 13M file removals involved files/paths referring to "secrets"
  - Examples: private keys, certificates, passwords
- Spanning 75k repositories
- Issue: History alteration $\not\Rightarrow$ security (archived copies persist)
- Keys must be rotated, not only purged from Git
- Might indicate poor security practices.

GitHistorian prototype

- Imagine you would like to avoid repositories with a track record of history alterations, or at least be alerted about them, for vetting purposes. How can you?
- For demonstration purposes only, we developed GitHistorian, a prototype OSS tool that leverages SWH data to address this need.

```
$ git-historian check https://github.com/example/project --branch main --verbose
Connected to the Software Heritage database!
Found 2 altered history records for 'https://github.com/example/project'

Record #1:
  Branch Name: refs/heads/master
  Altered Commit: swh:1:rev:a1b2c3d4e5f6789...
  File Path: assets/private/id_rsa
  Status: Removed
[...]
```

Outline



1. Introduction
2. Rewritten Git histories
3. Mutated Git tags
4. Conclusion

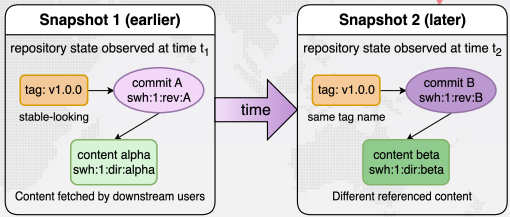
Mutating the "immutable": Git tag alterations

Tags as release anchors

- Releases are often *referenced by tag* (`v1.2.3`): Go modules, Git dependencies in npm, Cargo, pip, GitHub Actions, CI pipelines, package recipes, …
- Commonly assumed to be immutable

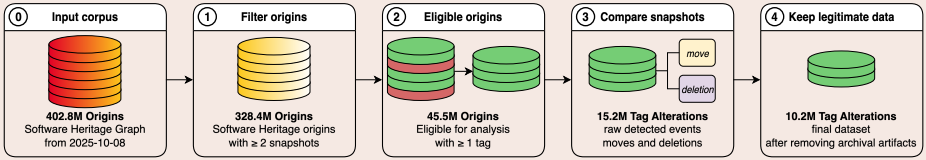
…but Git tags are *mutable*

- Tags can be deleted, or moved (force-pushed) to point to a different commit
- Git docs: "people MUST be able to trust their tag-names"
- Forges keep no record of what a tag used to point to



A large-scale analysis of Git tag alterations

1. Retrieve from Software Heritage repos archived ≥ 2 times (328.4 M, 2016–2025), keep Git ones with tags (45.5 M)
2. Compare consecutive snapshots: did a tag disappear, or point to a different target (SWHID)?
3. Classify: deletions vs. moves (different code, commit metadata, or release)



Solal Rapaport, Laurent Pautet, Samuel Tardieu, Stefano Zacchiroli, Théo Zimmermann
Mutating the "Immutable": A Large-Scale Study of Git Tag Alterations
ACM REP 2026 https://doi.org/10.1145/3820002.3828585

Key findings on tag alterations

How often?

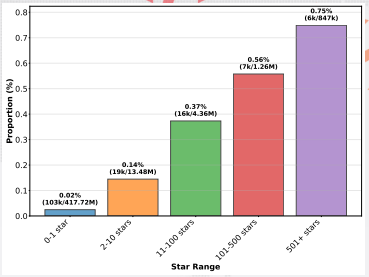
- 10.2 M tag alterations in 189 k repos, growing steadily

How?

- Deletions (93.8%) → availability: builds break
- Moves (6.2%, ~626k): same name, new target; 68% point to different code → integrity!

Where?

- 74% of alterations are in repos with 0–1 GitHub stars, but…
- …the more popular a repository, the more likely it has altered tags



repositories with ≥ 1 alteration, by star count

Tag alterations in the wild: attacks and broken builds

Supply-chain attack: `tj-actions/changed-files` (March 2025)

- Widely used GitHub Action; attackers moved its version tags to a malicious commit
- Payload dumped CI runner memory (secrets, tokens, credentials) into workflow logs
- Every workflow pinned to a version tag (`@v39`, `@v47`, …) ran it on the next run
- Captured by Software Heritage in a fork (`zendesk/changed-files`): 346 tags moved to the same malicious commit on 14 March 2025

Broken builds: Nixpkgs cross-analysis

- 32 packages reference altered tags; 7 fail to rebuild without the binary cache (4: tag deleted, 3: hash mismatch)

```
$ nix build github:NixOS/nixpkgs/08ffd21[...]#libv3270.src --rebuild
source> trying https://[...]/refs/tags/5.5.0.tar.gz
error: hash mismatch in fixed-output derivation '/nix/[...]':
         specified: sha256-Cn/to1/[...]=
            got:    sha256-tW3CPga[...]=
```

What to do about tag alterations — recommendations

Build tools & package managers

- Tags are fine as *human-friendly names*, but pin dependencies to cryptographic identifiers (commit hashes, SWHIDs)
- Record both: tag name + hash it resolved to (as, e.g., Nixpkgs and Go already do)

Forges

- Expose audit logs of tag mutations (creation, move, deletion)
- GitHub immutable releases (Aug 2025): good step, but opt-in, GitHub-only, and for releases rather than Git tags

Everyone

- Monitor tag alterations as a standard supply-chain security practice
- Software Heritage keeps every snapshot it has seen → past tag states remain auditable

Outline



1. Introduction
2. Rewritten Git histories
3. Mutated Git tags
4. Conclusion

Repository alterations as a supply chain threat

Availability

- Code you depend on silently vanishes: deleted tags, rewritten branches → broken builds, irreproducible artifacts

Integrity

- Same name, different code: moved tags → vector for supply-chain attacks (`tj-actions/changed-files`, 2025)

Provenance & accountability

- Retroactive edits to authorship, history, licenses → what was actually shipped, by whom, under which terms?
- Purged secrets are not revoked secrets: copies persist

Forges show *the present* of the open source supply chain. An independent memory of *the past*, like Software Heritage, provides its audit log.

Conclusion

- Git histories and tags are mutable, and forges keep no record of what changed
- Software Heritage keeps every snapshot it has seen → alterations become auditable at scale: 1.22 M rewritten histories, 10.2 M tag alterations

Recommendations

- Archive what you depend on: otherwise it can silently change or vanish
- Pin dependencies to intrinsic identifiers (commit hashes, SWHIDs), not tags
- Monitor alterations of your dependencies as standard supply chain practice

softwareheritage.org — Stefano Zacchiroli / stefano.zacchiroli@telecom-paris.fr / @zacchiro@mastodon.xyz


Solal Rapaport, Laurent Pautet, Samuel Tardieu, Stefano Zacchiroli
Altered Histories in Version Control System Repositories: Evidence from the Trenches
ASE 2025. https://arxiv.org/abs/2509.09294


Solal Rapaport, Laurent Pautet, Samuel Tardieu, Stefano Zacchiroli, Théo Zimmermann
Mutating the "Immutable": A Large-Scale Study of Git Tag Alterations
ACM REP 2026. https://doi.org/10.1145/3820002.3828585



Appendix

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all

Reference catalog



find and reference all software source code

Universal archive



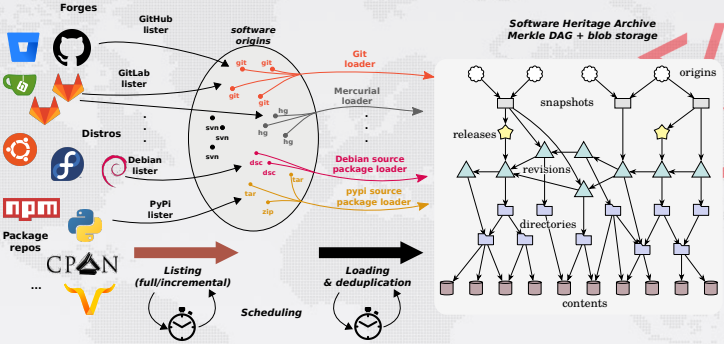
preserve and share all software source code

Research infrastructure



enable analysis of all software source code

A peek under the hood: a universal archive



Global development history permanently archived in a uniform data model

- over 30 billion unique source files from over 445 million software projects
- ~3PB (compressed) blobs, ~60 B nodes, over 1 trillion edges

The largest software archive, a shared infrastructure

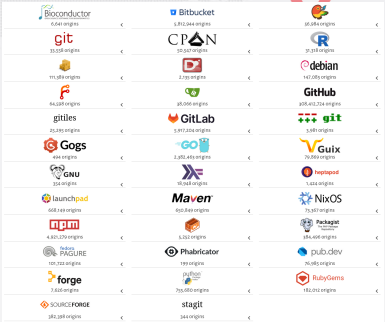
One infrastructure
open and shared



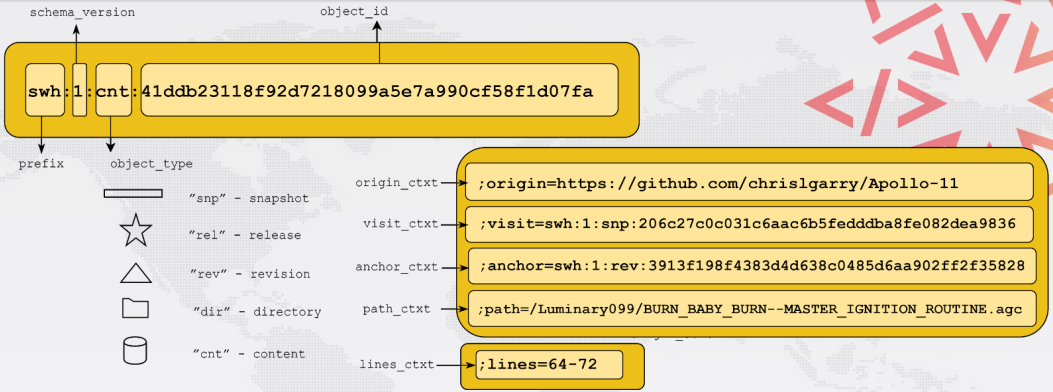
Inria unesco

The largest archive ever built





Software Heritage Identifiers (SWHIDs) (swhid.org)



An emerging standard

- Adoption: SPDX 2.2, IANA-registered "swh:" URI prefix, WikiData P6138, …
- Breaking news: standard ISO/IEC 18670:2025

Examples

- Apollo 11 AGC excerpt
- Quake III rsqrt