

What Public Software Can Teach Us

Findings from Software Heritage

Stefano Zacchioli

Télécom Paris, Institut Polytechnique de Paris
`stefano.zacchioli@telecom-paris.fr`

8 October 2026 — Télécom Paris



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

- 
- 1 Software as knowledge
 - 2 Software Heritage
 - 3 From the archive: Public code evolution
 - 4 From the archive: Who writes public code?
 - 5 From the archive: Art in public code
 - 6 From the archive: Securing the software supply chain
 - 7 Conclusion

About the speaker

- Professor of Computer Science, Télécom Paris, Institut Polytechnique de Paris
- Free/Open Source Software activist (20+ years)
- Debian Developer & Former 3x Debian Project Leader
- Former Open Source Initiative (OSI) director
- Software Heritage co-founder & Chief Scientific Officer (CSO)

Software is dual-form knowledge



"The source code for a work means the preferred form of the work for making modifications to it."

GPL Licence

Hello World

Program (excerpt of binary)

```
4004e6: 55
4004e7: 48 89 e5
4004ea: bf 84 05 40 00
4004ef: b8 00 00 00 00
4004f4: e8 c7 fe ff ff
4004f9: 90
4004fa: 5d
4004fb: c3
```

Program (source code)

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

Software *source code* is precious human knowledge

Harold Abelson, Structure and Interpretation of Computer Programs (1st ed.)

1985

“Programs must be written for people to read, and only incidentally for machines to execute.”

Apollo 11 source code (excerpt)

P63SP0T3	CA	BIT6	# IS THE LR ANTENNA IN POSITION 1 YET
	EXTEND		
	RAND	CHAN33	
	EXTEND		
	BZF	P63SP0T4	# BRANCH IF ANTENNA ALREADY IN POSITION 1
	CAF	CODE500	# ASTRONAUT: PLEASE CRANK THE
	TC	BANKCALL	# SILLY THING AROUND
	CADR	GOPERF1	
	TCF	GOTOP00H	# TERMINATE
	TCF	P63SP0T3	# PROCEED SEE IF HE'S LYING
P63SP0T4	TC	BANKCALL	# ENTER INITIALIZE LANDING RADAR
	CADR	SETPOS1	
	TC	POSTJUMP	# OFF TO SEE THE WIZARD ...
	CADR	BURNBABY	

Parcoursup source code (excerpt)

```
public class AlgoOrdreAppel {  
  
    /* la boucle principale de calcul des ordres d'appels.  
     Renvoie une exception en cas de problème. */  
    public static AlgoOrdreAppelSortie calculerOrdresAppels(AlgoOrdreAppelEntree data) throws VerificationException  
    {  
        VerificationEntreeAlgoOrdreAppel.verifier(data);  
  
        AlgoOrdreAppelSortie resultat = new AlgoOrdreAppelSortie();  
        /* calcul de l'ordre d'appel de chaque groupe de classement */  
        for (GroupeClassement ga : data.groupeClassements) {  
            resultat.ordresAppel.put(ga.cgpCod, ga.calculerOrdreAppel());  
        }  
  
        /* vérification avant retour des résultats */  
        new VerificationsResultatsAlgoOrdreAppel().verifier(data, resultat);  
  
        return resultat;  
    }  
  
    private AlgoOrdreAppel() {  
    }  
}
```

Len Shustek, Computer History Museum

2006

“Source code provides a view into the mind of the designer.”

The software commons

Definition (Commons)

The **commons** is the cultural and natural resources accessible to all members of a society, including natural materials such as air, water, and a habitable earth. These resources are held in common, not owned privately. <https://en.wikipedia.org/wiki/Commons>

Definition (Software Commons)

The **software commons** consists of all computer software which is available at little or no cost and which can be altered and reused with few restrictions. Thus *all open source software and all free software are part of the [software] commons.* [...]

https://en.wikipedia.org/wiki/Software_Commons

The Software Commons is fragile



A word cloud centered on the slide, featuring terms related to digital fragility and security. The words are arranged in a circular pattern, with 'damage' and 'disaster' being the largest. Other prominent words include 'malicious', 'obsolete', 'attack', 'deletion', 'format', 'corruption', 'encryption', 'dependencies', 'aging', 'tear', 'dangling', 'wear', 'reference', 'storage', and 'media'. The words are in various colors including purple, blue, green, and brown.

Like all digital information, FOSS is fragile

- link rot: projects are created, moved around, removed
- business-driven code loss (e.g., Gitorious, Google Code, Bitbucket)
- data rot: physical media with legacy software decay

If a website disappears you go to the Internet Archive...

where do you go if (a repository on) GitHub or GitLab goes away?

Outline

- 
- 1 Software as knowledge
 - 2 Software Heritage
 - 3 From the archive: Public code evolution
 - 4 From the archive: Who writes public code?
 - 5 From the archive: Art in public code
 - 6 From the archive: Securing the software supply chain
 - 7 Conclusion



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

One infrastructure
open and shared

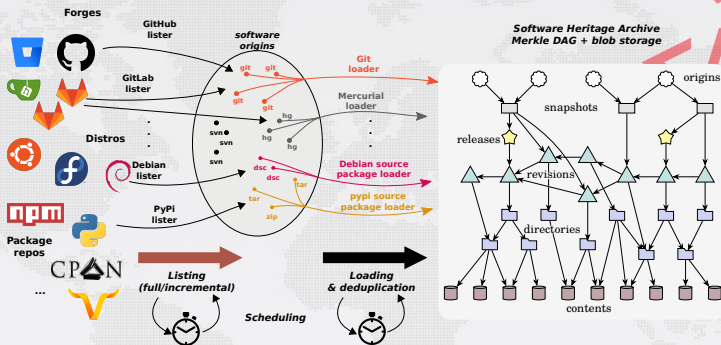


The largest archive ever built



 6,641 origins	 5,812,044 origins	 16,583 origins
 33,538 origins	 30,350 origins	 97,371 origins
 111,388 origins	 2,293 origins	 147,083 origins
 84,535 origins	 35,096 origins	 305,481,724 origins
 25,485 origins	 5,917,324 origins	 3,481 origins
 454 origins	 2,352,486 origins	 79,809 origins
 354 origins	 10,948 origins	 1,424 origins
 658,144 origins	 650,848 origins	 73,537 origins
 4,821,239 origins	 6,322 origins	 184,408 origins
 129,722 origins	 199 origins	 26,085 origins
 7,660 origins	 795,010 origins	 182,012 origins
 382,398 origins	 344 origins	

A peek under the hood: a universal archive



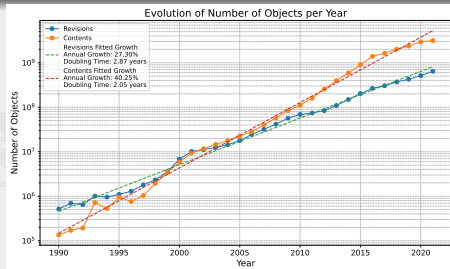
Global development history permanently archived in a uniform data model

- over 29.5 billion unique source files from over 441 million software projects
- ~3PB (compressed) blobs, ~60 B nodes, over 1 trillion edges

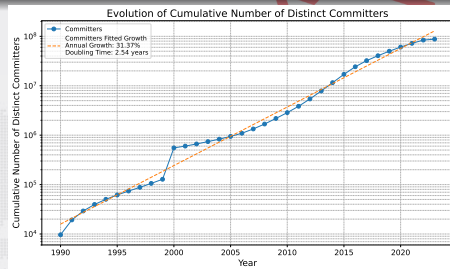
- Browse [the archive](#)
- [Trigger archival](#) of your preferred software in a breeze
- Get and use SWHIDs ([full specification available online](#))
- The [Apollo 11 AGC source code example](#)
- Cite software [with the biblatex-software style](#) from CTAN
- Example use in a research article: compare Fig. 1 and conclusions
 - in [the 2012 version](#)
 - in [the updated version](#) using SWHIDs and Software Heritage
- Example in a journal: [an article from IPOL](#)
- [Curated deposit in SWH via HAL](#), see for example: [LinBox](#), [SLALOM](#), [Givaro](#), [NS2DDV](#), [SumGra](#), [Coq proof](#), ...
- Rescue landmark legacy software, see the [SWHAP process with UNESCO](#)

- 
- 1 Software as knowledge
 - 2 Software Heritage
 - 3 From the archive: Public code evolution**
 - 4 From the archive: Who writes public code?
 - 5 From the archive: Art in public code
 - 6 From the archive: Securing the software supply chain
 - 7 Conclusion

Public code grows exponentially



Source code (unique files and commits)



Contributors (unique commit authors)

- Original commits *double every ~30 months*, original files *every ~22 months*, for decades now
- The number of people who write public code grows exponentially too

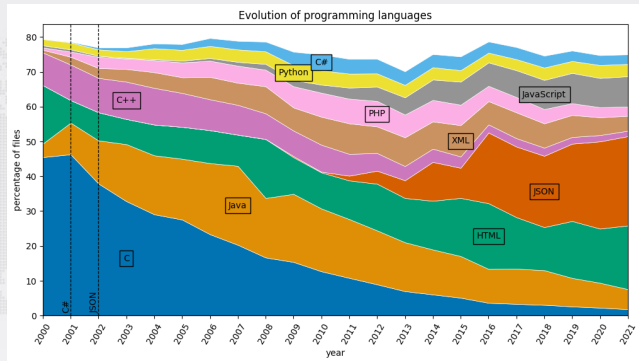


G. Rousseau, R. Di Cosmo, S. Zacchiroli

Software Provenance Tracking at the Scale of Public Source Code

Empirical Software Engineering 25(4): 2930–2959 (2020). [doi:10.1007/s10664-020-09828-5](https://doi.org/10.1007/s10664-020-09828-5) — updated data and code:
gitlab.softwareheritage.org/rdicosmo/archive-growth

Public code is polyglot (and a fashion victim?)



Evolution of coding activity (by file) for programming, markup, and data languages, over decades of public code in Software Heritage



A. Desmazières, R. Di Cosmo, V. Lorentz

50 Years of Programming Language Evolution through the Software Heritage Looking Glass

22nd International Conference on Mining Software Repositories (MSR 2025). [doi:10.1109/MSR66628.2025.00068](https://doi.org/10.1109/MSR66628.2025.00068)

- 
- 1 Software as knowledge
 - 2 Software Heritage
 - 3 From the archive: Public code evolution
 - 4 From the archive: Who writes public code?**
 - 5 From the archive: Art in public code
 - 6 From the archive: Securing the software supply chain
 - 7 Conclusion

MSR '22, May 23–24, 2022, Pittsburgh, PA, USA

Davide Rossi and Stefano Zacchiroli

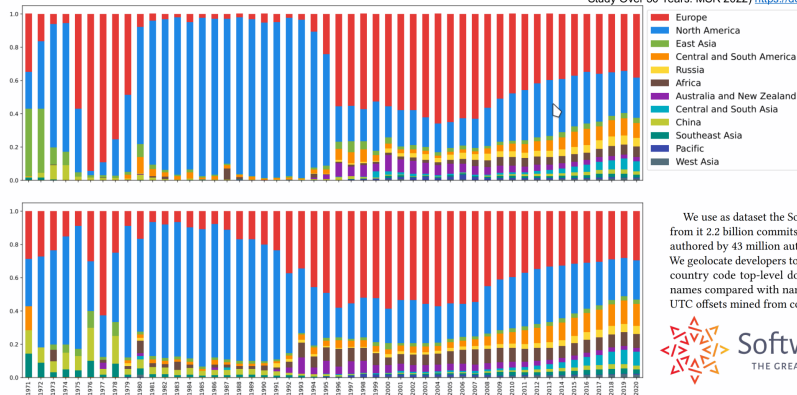
Geographic Diversity in Public Code Contributions: An Exploratory Large-Scale Study Over 50 Years. MSR 2022) <https://doi.org/10.1145/3524842.3528471>

Figure 3: Ratio of commits (above) and active authors (below) by world zone over the 1971–2020 period.

We use as dataset the Software Heritage archive [3] and analyze from it 2.2 billion commits archived from 160 million projects and authored by 43 million authors during the 1971–2021 time period. We geolocate developers to 12 world regions, using as signals email country code top-level domains (ccTLDs) and author (first/last) names compared with name distributions around the world, and UTC offsets mined from commit metadata.



Software Heritage
THE GREAT LIBRARY OF SOURCE CODE



D. Rossi, S. Zacchiroli

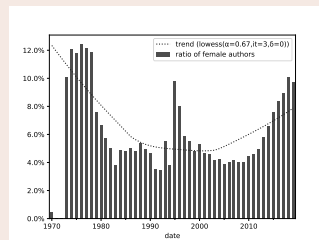
Geographic Diversity in Public Code Contributions: An Exploratory Large-Scale Study Over 50 Years

19th International Conference on Mining Software Repositories (MSR 2022). [doi:10.1145/3524842.3528471](https://doi.org/10.1145/3524842.3528471)

Archived commit metadata contains public information that can be mined to study diversity, equity, inclusion (DEI) traits of the global population of public code authors.

Gender gap — key findings

- Male authors contributed 92% of public code commits up to 2019.
- Female authors (and their commits) have grown stably for 15 years reaching 10% of yearly commits in 2019.
- The COVID-19 pandemic has *caused* a trend inversion (it is not just correlation!)



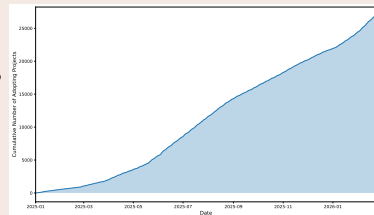
- Zacchiroli. *Gender differences in public code contributions: a 50-year perspective*. IEEE Software, 2021
- Rossi and Zacchiroli. *Worldwide gender differences in public code contributions [...]*. ICSE SEIS, 2022
- Casanueva et al. *The Impact of the COVID-19 Pandemic on Women's Contribution to Public Code*. Empir. Softw. Eng. 30(25), 2025

Coding agents went from nothing to a quarter of GitHub in a year

128k projects mined for traces left by coding agents:

CLAUDE.md, Co-authored-by:, copilot/* branches

- 22–29% of projects in Jan 2025 to Feb 2026
- Broad: every topic, every language, young and old, big and small
- AI commits are $3\times$ larger; $\frac{2}{3}$ are features and bug fixes
- Follow-up replication study: projects *born after August 2025*: 72–76%, Claude Code leading



- Robbes, Matricon, Degueule, Hora, Zacchiroli. *Agentic Much? Adoption of Coding Agents on GitHub*. ACM TOSEM, 2026. Preprint: [arXiv:2601.18341](https://arxiv.org/abs/2601.18341)
- Robbes, Matricon, Degueule, Hora, Zacchiroli. *Agentic Very Much!*. Preprint: [arXiv:2606.07448](https://arxiv.org/abs/2606.07448), 2026
- Robbes, Matricon, Degueule, Hora, Zacchiroli. *Promises, Perils, and (Timely) Heuristics for Mining Coding Agent Activity*. MSR 2026

- 
- 1 Software as knowledge
 - 2 Software Heritage
 - 3 From the archive: Public code evolution
 - 4 From the archive: Who writes public code?
 - 5 From the archive: Art in public code
 - 6 From the archive: Securing the software supply chain
 - 7 Conclusion

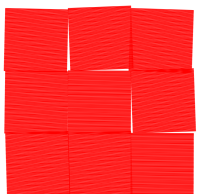
- [fractal-in-processing](#): fractals in Processing, Gitorious, 2010 — *the forge is gone*, the code survives in Software Heritage
 - [carpet.pde](#): a Sierpinski carpet in 30 lines of Processing (a Java dialect)
 - run it again today, translated to p5.js, in the [p5.js web editor](#)
- [Formation Processing, mars 2014](#): a 3-day Processing course (in French!), also rescued from Gitorious
- [junkiyoshi/Insta20240322](#): one openFrameworks artwork a day, since 2016 — see them at [junkiyoshi.com](#)

carpet.pde (2010), translated to p5.js

```
function setup() {
  createCanvas(500, 500);
  noStroke(); background(0);
  carpet(8, 250, 250, 200);
}

function carpet(depth, x, y, w) {
  const c = 120 * depth;
  fill(20 - c, 200 - c, 20 + c);
  rect(x - w/2, y - w/2, w, w);
  if (depth > 1) {
    const s = Math.floor(w / 3);
    const n = [[-1,-1], [-1,1], [1,1],
               [1,-1], [0,-1], [0,1],
               [-1,0], [1,0]];
    for (const [dx, dy] of n)
      carpet(depth-1, x+dx*w, y+dy*w, s);
  }
}
```

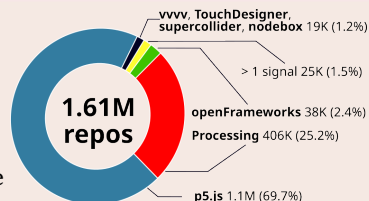
Artists have been coding for 60 years

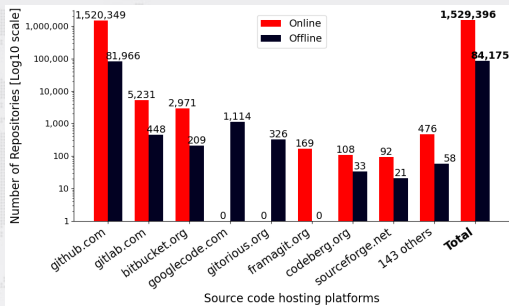


- *Generative art* goes back to Molnár (1975), Sýkora, Cornock, Edmonds
- Artist-led *open source* libraries made it mainstream: [Processing](#), [p5.js](#), [openFrameworks](#), [SuperCollider](#), [TouchDesigner](#), [nodebox](#), [vvvv](#)

Finding art-related code at archive scale

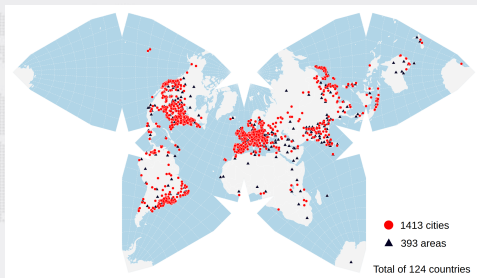
- *Signal*: filenames of 7 art environments (p5 . js, ofApp . cpp, .pde, .scd, .toe, .ndbx, .v4p)
- Swept *every* archived origin, via the compressed graph (2025-05-18)
- **1.61 M repositories**: the first large-scale study of art code





RQ1 — key findings

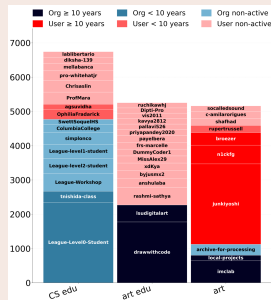
- GitHub holds 99.3% of it; artists, makers and schools also publish on GitLab, Bitbucket, Codeberg, SourceHut and 143 niche forges
- **84,175 repositories (5.5%) are no longer online:** only the archive has them
- Google Code and Gitorious are 100% gone — e.g. 1,114 Processing-era repositories



RQ2 — key findings

- 10% sample: 96,581 main contributors, 14,427 with a resolvable location
- 1,806 distinct places in **124 countries**, on every continent
- USA 4,303, India 1,055, Brazil 934, Canada 898, Japan 586 contributors
- 334 in under-represented regions: 20 African countries, the Middle East, Central Asia, the Caribbean, the Indian Ocean, Iceland

Top 42 accounts, 17,156 repositories: three purposes



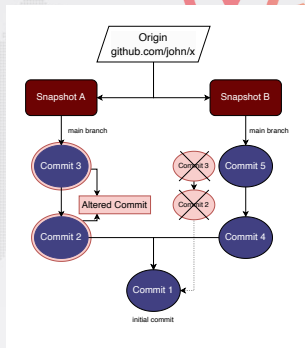
- **CS education**: art as a way into programming — the League of Amazing Programmers alone accounts for 3,309 repositories
- **Art education**: code as a design medium — e.g. drawwithcode, the *Creative Coding* course at Politecnico di Milano
- **Art**: one repository per artwork, installation or tool — e.g. junkiyoshi, one openFrameworks piece a day since 2016 (2,344 repositories)
- Art accounts keep going; educational ones peak in 2020–2022, then stop

Baudry, Gamage, Gonzalez Fernandez, MK, Pena Mendieta, Souza Pinter, Zacchiroli. *Art in Humanity's Code*. IEEE Trans. Softw. Eng., to appear. Open access preprint: [arXiv.org:2608.05174](https://arxiv.org/abs/2608.05174)

- 
- 1 Software as knowledge
 - 2 Software Heritage
 - 3 From the archive: Public code evolution
 - 4 From the archive: Who writes public code?
 - 5 From the archive: Art in public code
 - 6 From the archive: Securing the software supply chain
 - 7 Conclusion

Altered histories in public Git repositories

- Git lets you **rewrite published history** (via force push), and forges keep no trace of it
- SWH compares successive snapshots of **111 M repos: 1.22 M repos** (~1.1%) with altered histories, **8.7 M altered commits**, 11.4% of them on *main/master* branches
- ~800k **license files retroactively altered** on main branches (32k repos)
- 13 M removals of **secrets** (keys, passwords): rewriting history is not security, copies persist
- Prototype tool (*GitHistorian*): vet a repository's track record of history alterations



Solal Rapaport, Laurent Pautet, Samuel Tardieu, Stefano Zacchiroli

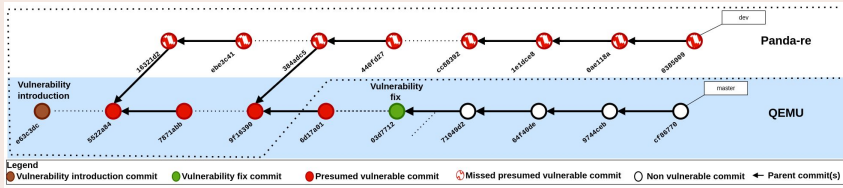
Altered Histories in Version Control System Repositories: Evidence from the Trenches

ASE 2025 <https://arxiv.org/abs/2509.09294>

One-day vulnerabilities across forks

1.7 M forks left behind when a security fix lands

Fork between the *introduction* and the *fix* of a vulnerability, and you inherit it — silently.



- "Color" the graph with [OSV.dev](#) data: 7162 repos → 1.7 M potentially vulnerable forks
- 86.3 M potentially vulnerable commits specific to forks, *invisible to dependency-based tooling*
- 152 cases checked by hand: 135 high-severity vulnerabilities, 9 confirmed by maintainers
- Only doable with a global, deduplicated data model: all forks, all platforms, one graph

- Lefeuvre, Reux, Zacchioli, Barais, Combemale. *Chasing One-day Vulnerabilities Across Open Source Forks*. SCORED 2026. Preprint: [arXiv:2511.05097](https://arxiv.org/abs/2511.05097)

Outline

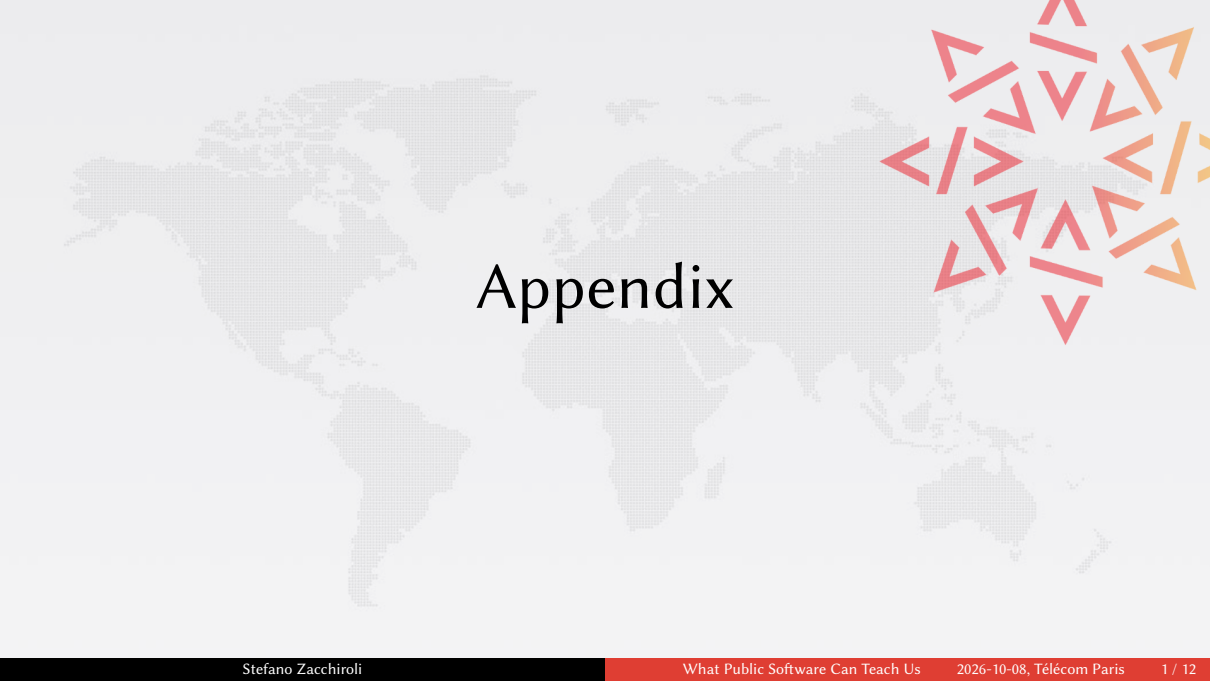
- 
- 1 Software as knowledge
 - 2 Software Heritage
 - 3 From the archive: Public code evolution
 - 4 From the archive: Who writes public code?
 - 5 From the archive: Art in public code
 - 6 From the archive: Securing the software supply chain
 - 7 Conclusion**

Conclusion

- Software source code is **precious human knowledge**: executable *and* readable by humans
- It is also **fragile**: Software Heritage preserves *all* public code and its history, as a shared, non profit infrastructure
- Preserving it made it **observable**: how the commons grows, how code (and bugs) travel, *who* gets to write code, what AI is trained on, what people create with code

Learn more, and get in touch

- Archive: archive.softwareheritage.org
- Papers: softwareheritage.org/publications
- Students: softwareheritage.org/community/students
- [Stefano Zacchioli](#) / stefano.zacchioli@telecom-paris.fr / [@zacchiro@mastodon.xyz](https://mastodon.xyz/@zacchiro)



Appendix

Sharing the vision



www.softwareheritage.org/support/testimonials

Donors, members, sponsors



Diamond sponsors



Platinum sponsors



Gold sponsors



Silver sponsors



Bronze sponsors



www.softwareheritage.org/support/sponsors

Software Hash Identifiers (SWHIDs)



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](#)

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?

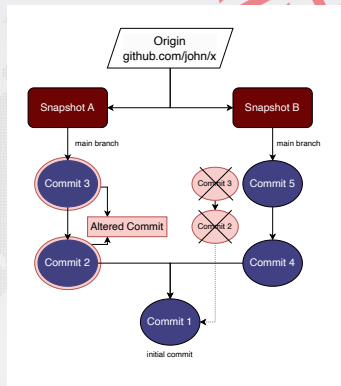


- ❶ How often are public Git histories altered?
- ❷ When this happens, what is changed and why?

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, depending 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 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** $\nrightarrow$ **security** (archived copies persist)
- Keys must be rotated, not only purged from Git
- Might indicate **poor security practices**.

- 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
```

[...]

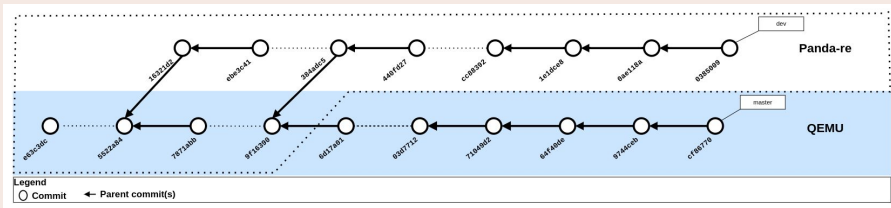
One-day vulnerabilities in open source

One-day vulnerabilities

- Def.: vulnerabilities that are **publicly known, but not fixed yet** in software you use.
- Challenge: **identify them quickly and exhaustively**, then apply countermeasures.
- Many tools available to detect one-day vulnerabilities via declared dependencies.

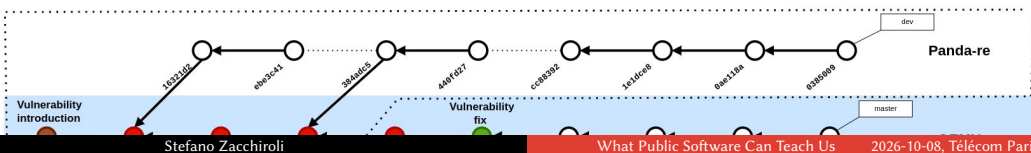
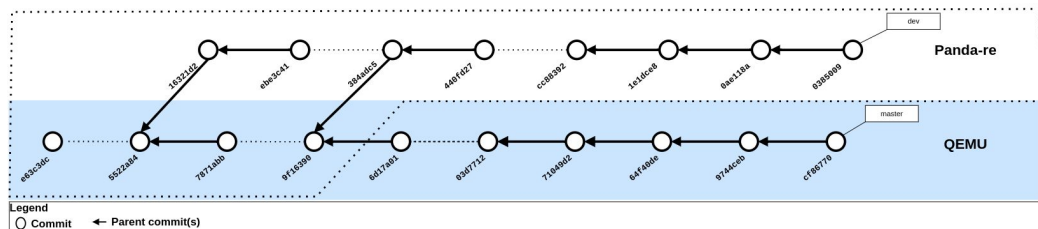
Reusing OSS via forks

...but OSS is also **reused via forking**: (1) start from existing OSS (e.g., the QEMU emulator), (2) create your own (e.g., Panda-re), (3) periodically integrate changes.



Vulnerability propagation through forks

- Any change to a piece of software (*commit*) can **introduce a new vulnerability**.
- Or it can **fix an existing vulnerability**.
- What happens if a project is forked **between introduction and fix** of a vulnerability?
- It inherits the vulnerability, ... until the change with the fix is integrated.



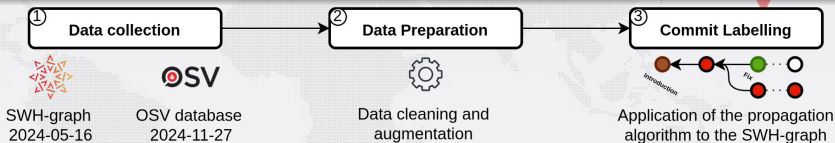
swh-vuln: chasing one-day vulnerabilities across forks... at SWH scale

Approach

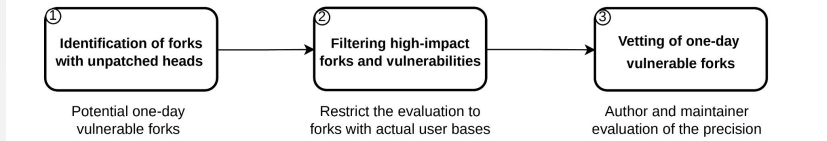
Start from a **public DB of vuln. introduced/fixed** in public commits (e.g., [OSV.dev](#)).

- 1 "Color" the entire graph of public code development history **with vulnerability info.**
 - Software Heritage is the only place where this can be done at the scale of all forks, across all public code, independently of specific development platforms.
- 2 **Inform maintainers** of vulnerable forks. (After validation.)

Phase 1: color the commit graph with vuln. info



Phase 2: find and vet vulnerable forks



Results (highlights)

- Starting from 7162 repos in OSV, we identified **1.7 M forks potentially vulnerable** in their most recent commit.
 - 86.3 M potentially vulnerable commits specific to forks, not findable with current tools.
- We manually verified 152 cases, confirming **135 high-severity vulnerabilities in popular forks**; 9 were further confirmed by maintainers.



panda-re/panda

Fork of QEMU, system emulator and virtualizer

CVE-2019-13164



go-nv/goenv

Fork of pyenv, for Go

CVE-2022-35861

XPERIA
Sony Smartphone

sonyxperiadev/kernel

Fork of Linux, for the Xperia dev community

CVE-2021-45485

CVE-2021-4154

bitcoin-sv/bitcoin-sv

Fork of Bitcoin

CVE-2021-37492



Romain Lefeuvre, Charly Reux, Stefano Zacchiroli, Olivier Barais, Benoit Combemale

Chasing One-day Vulnerabilities Across Open Source Forks

SCORED 2026. Preprint: <https://arxiv.org/abs/2511.05097>