Building a Safer Open Source Supply Chain

with Software Heritage, the Great Library of Source Code

Stefano Zacchiroli

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

10 Oct 2025
8th Madrilenian Seminar on Software Research (MADSESE)
Madrid, Spain

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Outline

B_IGNOREHEADING

About the speaker

- Professor of Computer Science, Télécom Paris, Polytechnic Institute of 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 & CSO

Outline

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

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
```

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
```

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 and technical* 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)

```
P63SPOT3        CA      BIT6            # IS THE LR ANTENNA IN POSITION 1 YET
                EXTEND
                RAND    CHAN33
                EXTEND
                BZF     P63SPOT4        # BRANCH IF ANTENNA ALREADY IN POSITION 1

                CAF     CODE500         # ASTRONAUT:    PLEASE CRANK THE
                TC      BANKCALL        #               SILLY THING AROUND
                CADR    GOPERF1
                TCF     GOTOPOOH        # TERMINATE
                TCF     P63SPOT3        # PROCEED       SEE IF HE'S LYING

P63SPOT4        TC      BANKCALL        # ENTER         INITIALIZE LANDING RADAR
                CADR    SETPOS1

                TC      POSTJUMP        # OFF TO SEE THE WIZARD ...
                CADR    BURNBABY
```

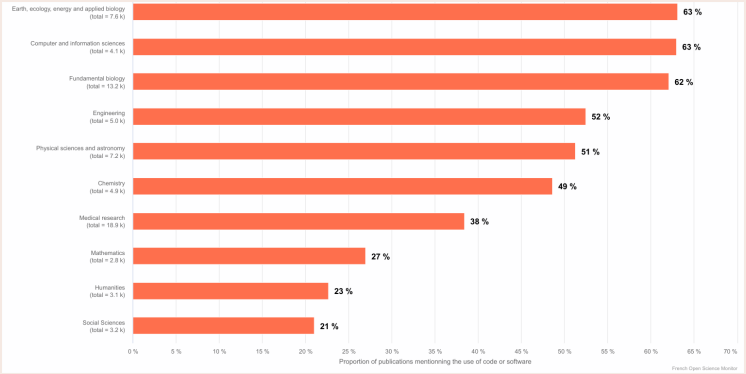
Quake III source code (excerpt)

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y; // evil floating point bit level hacking
    i = 0x5f3759df - ( i >> 1 ); // what the fuck?
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
// y = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this
can be removed

    return y;
}
```

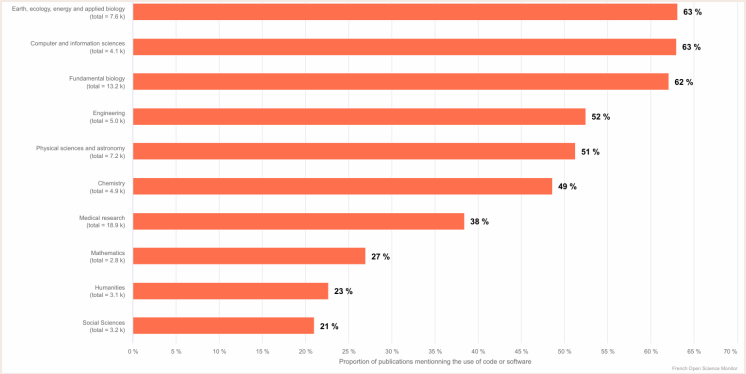
Software source code is precious *scientific* knowledge



Software powers modern research: 20%+ articles use software, all disciplines.

2023 French Open Science Monitor

Software source code is precious *scientific* knowledge



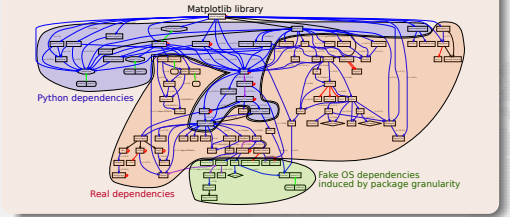
Software powers modern research: 20%+ articles use software, all disciplines.

2023 French Open Science Monitor

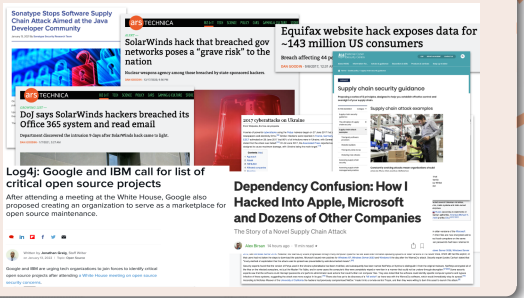
We need a *dedicated infrastructure* to preserve and share *all* this knowledge!

Enhancing software Reuse, Security and Transparency

Software complexity is growing…

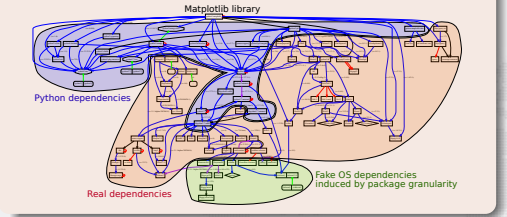


…you need to Know Your Software (KYSW)

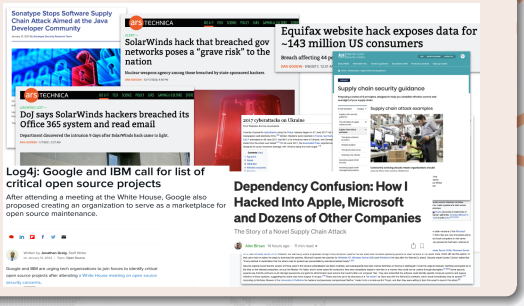


Enhancing software Reuse, Security and Transparency

Software complexity is growing…



…you need to Know Your Software (KYSW)



Sec. 4. Enhancing Software Supply Chain Security

ensuring and attesting […] to the integrity and provenance of open source software

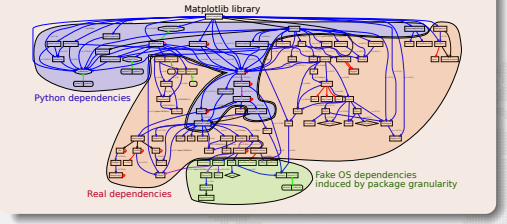
May 2021 POTUS Executive Order

EU Cyber Resilience Act (2024/2847)

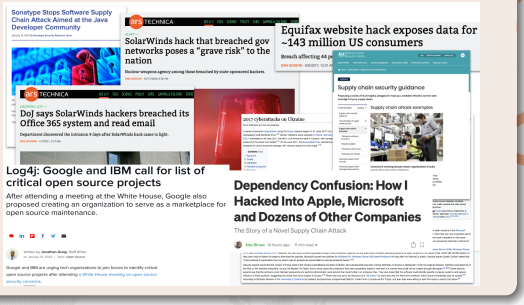
Regulation aims to […] ensuring […] software products […] with fewer vulnerabilities.

Enhancing software Reuse, Security and Transparency

Software complexity is growing…



…you need to Know Your Software (KYSW)



Sec. 4. Enhancing Software Supply Chain Security

ensuring and attesting […] to the integrity and provenance of open source software

May 2021 POTUS Executive Order

EU Cyber Resilience Act (2024/2847)

Regulation aims to […] ensuring […] software products […] with fewer vulnerabilities.

We need a *trusted* knowledge base providing *software integrity and provenance*!

Software source code is fragile

Endangered source code …

- *link rot*
- *data rot*
- *platform consolidation*
  - 2015 Google Code and Gitorious.org shutdown: ~1M
  - 2019 Bitbucket mercurial phase out: ~250.000
  - 2022 GitLab.com: remove inactive projects?

Software source code is fragile

Endangered source code …

- *link rot*
- *data rot*
- *platform consolidation*
  - 2015 Google Code and Gitorious.org shutdown: ~1M
  - 2019 Bitbucket mercurial phase out: ~250.000
  - 2022 GitLab.com: remove inactive projects?

… is endangered knowledge!

broken links and missing pieces in the *web of knowledge* of humankind

Software source code is fragile

Endangered source code …

- *link rot*
- *data rot*
- *platform consolidation*
  - 2015 Google Code and Gitorious.org shutdown: ~1M
  - 2019 Bitbucket mercurial phase out: ~250.000
  - 2022 GitLab.com: remove inactive projects?

… is endangered knowledge!

broken links and missing pieces in the *web of knowledge* of humankind

Bottomline: we need a global, long term effort

to build a *universal archive* of *all software source code*

Outline

Unveiled in 2016

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

Unveiled in 2016

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

Unveiled in 2016

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

Unveiled in 2016

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

Today: a *universal* software archive, as a shared infrastructure

One infrastructure
open and shared

Today: a *universal* software archive, as a shared infrastructure

One infrastructure
open and shared

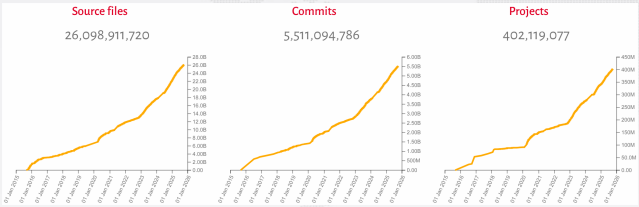
The largest archive ever built

Today: a *universal* software archive, as a shared infrastructure

One infrastructure
open and shared

inria unesco

The largest archive ever built



Bitbucket 2,818,626 origins	 56,983 origins	git 33,472 origins
 29,046 origins	debian 142,984 origins	 90,755 origins
GitHub 266,235,919 origins	gitiles 24,614 origins	GitLab 5,803,610 origins
git 3,824 origins	Gogs 422 origins	 1,966,688 origins
Guix 56,248 origins	GNU 354 origins	heptapod 1,358 origins
launchpad 654,759 origins	Maven 425,606 origins	NixOS 34,420 origins
npm 4,070,945 origins	 4,941 origins	Packagist 380,156 origins
PAGURE 72,459 origins	Phabricator 198 origins	pub.dev 63,003 origins
python 621,919 origins	SOURCEFORGE 382,368 origins	stagit 343 origins

figures as of January 18 2025

A revolutionary infrastructure

The *graph* of public software development

All software development in a single graph …

- enable traceability

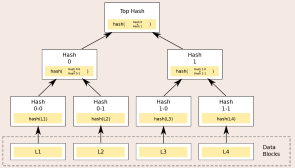
A revolutionary infrastructure

The *graph* of public software development

All software development in a single graph …

- enable traceability

The *global ledger* of public code



… a Merkle graph

- ensure integrity

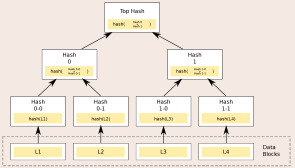
A revolutionary infrastructure

The *graph* of public software development

All software development in a single graph …

- enable traceability

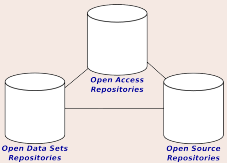
The *global ledger* of public code



… a Merkle graph

- ensure integrity

A *pillar* of Open Science



Reference archive of Research Software

- reproducibility
- reference

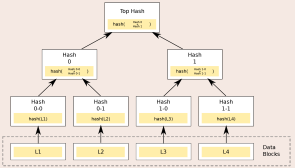
A revolutionary infrastructure

The *graph* of public software development

All software development in a single graph …

- enable traceability

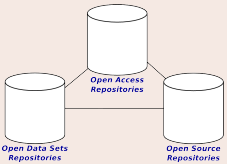
The *global ledger* of public code



… a Merkle graph

- ensure integrity

A *pillar* of Open Science



Reference archive of Research Software

- reproducibility
- reference

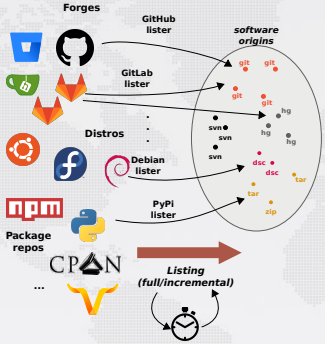
Reference platform for *Big Code*

uniform data structure

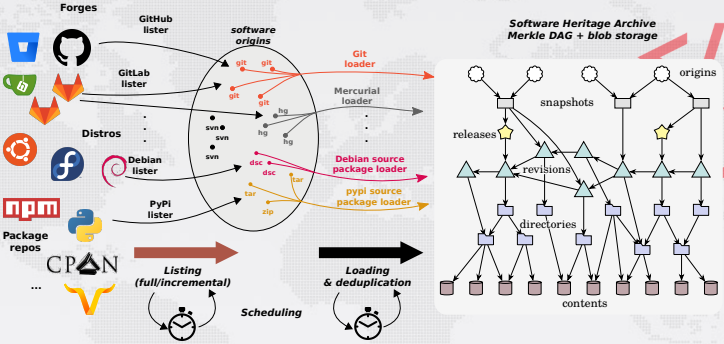
- large scale studies
- cybersecurity
- machine learning, AI, …

(more later)

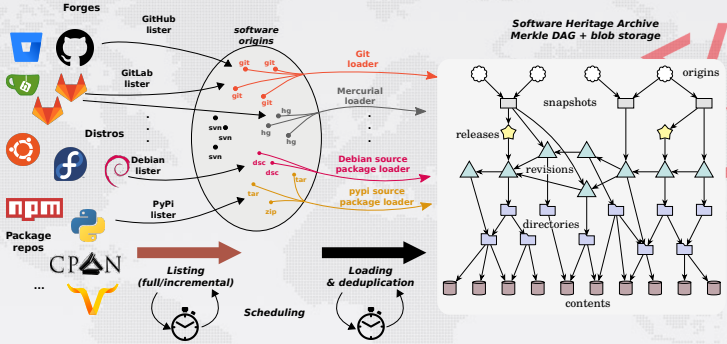
A peek under the hood: a universal archive



A peek under the hood: a universal archive



A peek under the hood: a universal archive



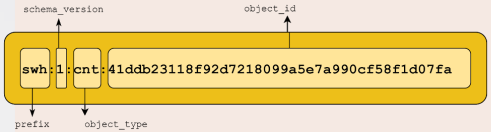
Global development history permanently archived in a uniform data model

- over 24 billion unique source files from over 375 million software projects
- ~2PB (compressed) blobs, ~50 B nodes, ~900 B edges

Referencing all source code artifacts with SWHIDs

Software Heritage Identifiers (SWHID)

see swhid.org

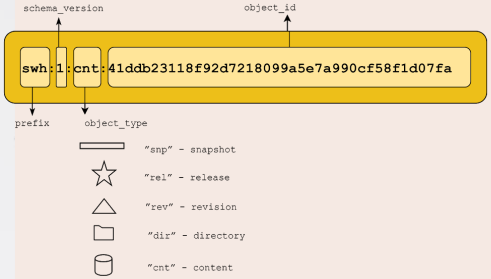


50+B
intrinsic,
decentralised,
cryptographic

Referencing all source code artifacts with SWHIDs

Software Heritage Identifiers (SWHID)

see swhid.org

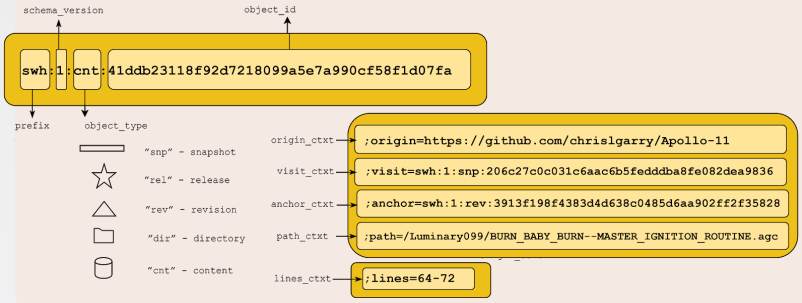


50+B
intrinsic,
decentralised,
cryptographic

Referencing all source code artifacts with SWHIDs

Software Heritage Identifiers (SWHID)

see swhid.org

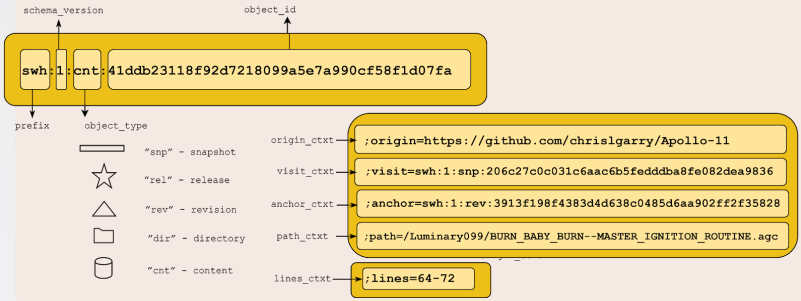


50+B
intrinsic,
decentralised,
cryptographic

Referencing all source code artifacts with SWHIDs

Software Heritage Identifiers (SWHID)

see swhid.org



50+B
intrinsic,
decentralised,
cryptographic

Full fledged *source code references* for traceability, integrity and reproducibility

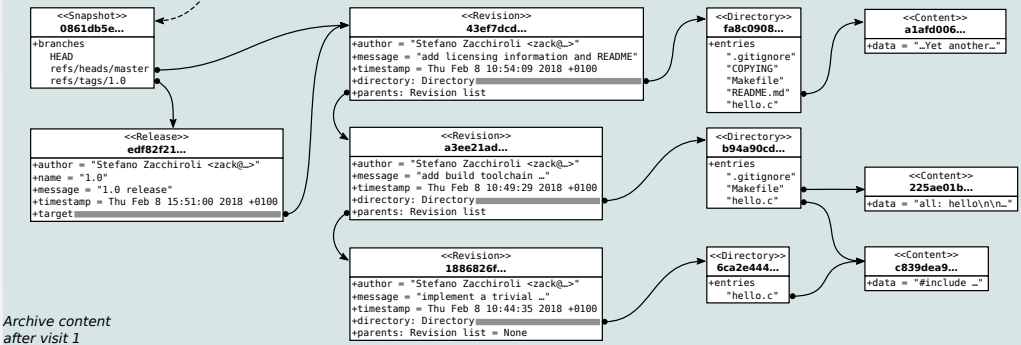
- Linux Foundation SPDX 2.2
- IANA-registered "swh:"
- WikiData property P6138

Examples: Apollo 11 AGC excerpt, Quake III rsqrt
Guidelines available, see the HOWTO

ISO/IEC 18670, see swhid.org

The archive: a (giant) Merkle DAG

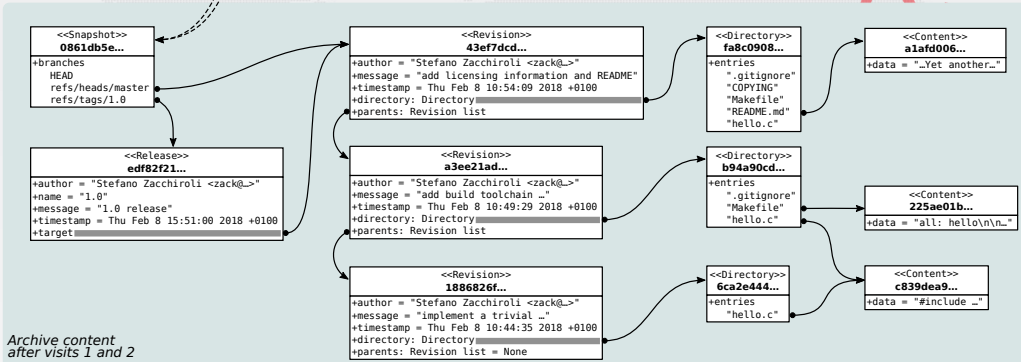
origin	visit	snapshot	timestamp
https://forge.softwareheritage.org/source/helloworld.git	1	0861db5e…	Fri Feb 9 12:38:45 2018 +0100



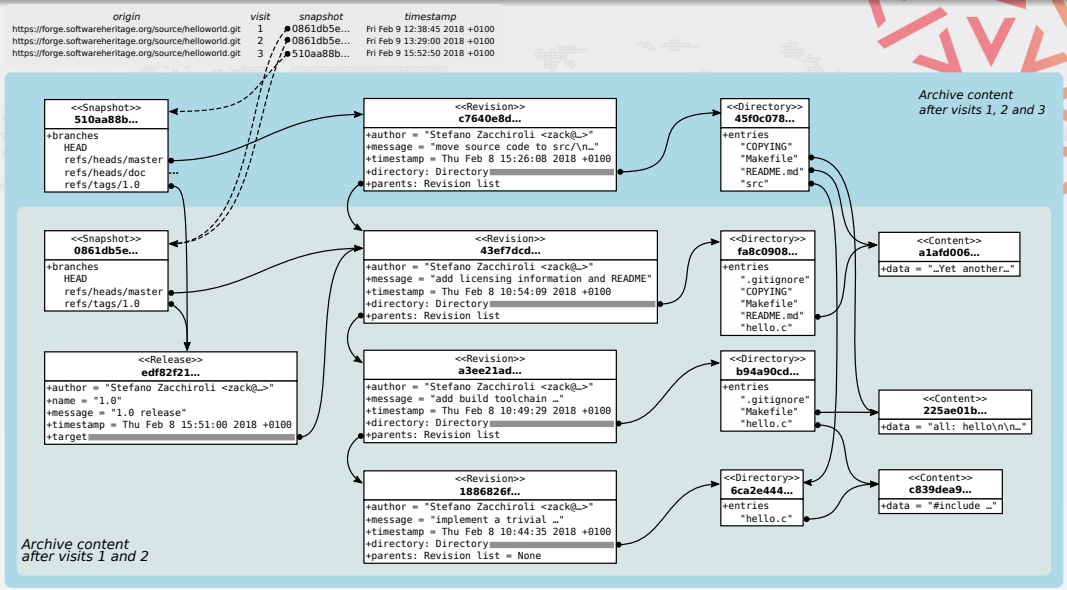
Archive content after visit 1

The archive: a (giant) Merkle DAG

origin	visit	snapshot	timestamp
https://forge.softwareheritage.org/source/helloworld.git	1	0861db5e…	Fri Feb 9 12:38:45 2018 +0100
https://forge.softwareheritage.org/source/helloworld.git	2	0861db5e…	Fri Feb 9 13:29:00 2018 +0100



The archive: a (giant) Merkle DAG



A walkthrough

General

- Browse the archive, get and use SWHIDs, e.g. Apollo 11 excerpt, Parcoursup excerpt
- Trigger archival with the browser extension or webhook forge integration

Open Science

- Curated deposit via HAL, e.g.: LinBox, SLALOM, Givaro, SumGra, Coq proof, …
- Cite software with the biblatex-software style, e.g.: article from IPOL

History of software: rescuing landmark legacy software

see SWHAP process, Software Stories, and SWHAP Days 2022

Public code

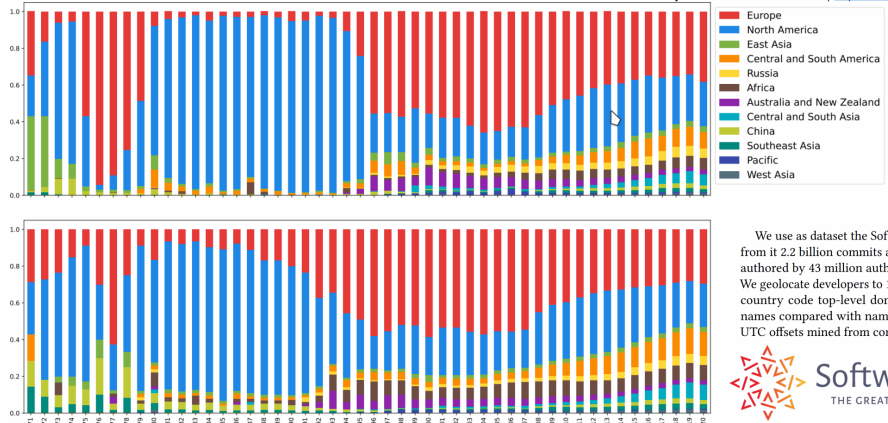
Archived source code from code.gouv.fr

Outline

(Open) Source Code comes from all over the world…

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

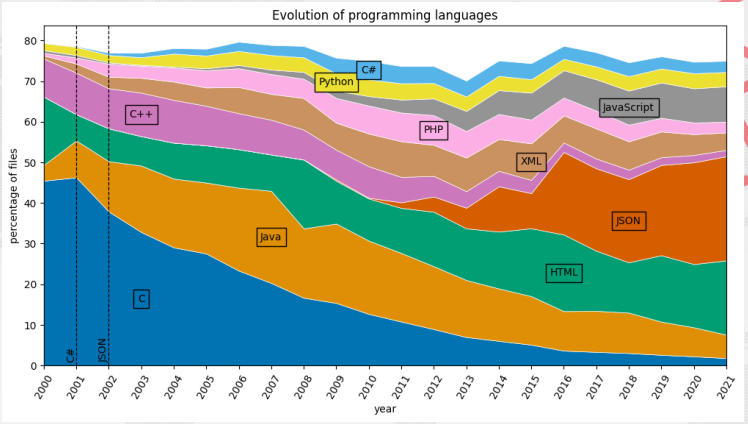


We use as dataset the Software Heritage archive [3] and analyze from it 2.2 billion commits archived from 160 million projects and authored by 43 million authors during the 1971–2021 time period. We geolocate developers to 12 world regions, using as signals email country code top-level domains (ccTLDs) 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

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

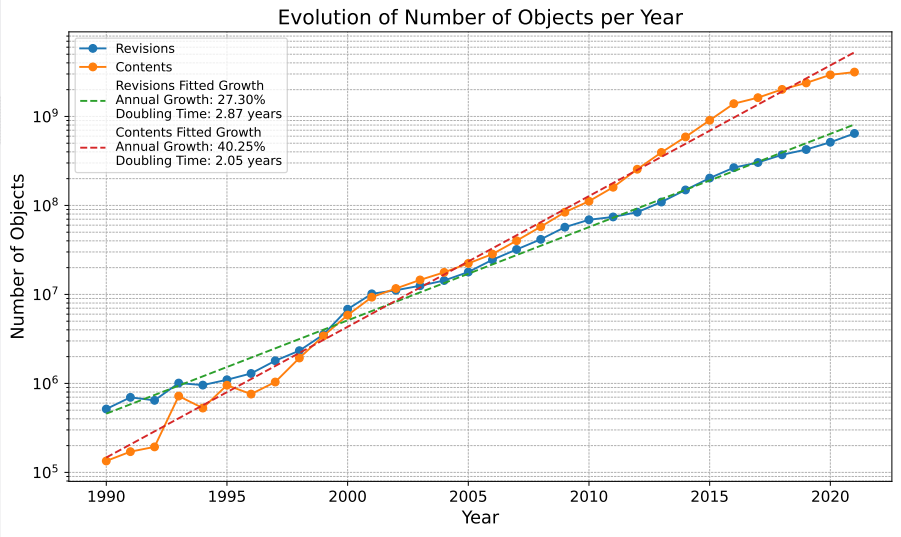
… it is written in many (programming) languages …



Evolution of the activity for programming, markup, and data languages from 2000 to 2021.

A. Desmazières, R. Di Cosmo, V. Lorentz
50 years of programming language evolution through the Software Heritage Looking Glass
MSR 2025. To appear.

… and it grows at an exponential rate



Rousseau, Di Cosmo, Zacchiroli: Software provenance tracking at the scale of public source code. Empir. Softw. Eng. 25(4): 2930-2959 (2020)

Outline

Open source security

Open source software can be freely used, copied, and modified.

Open Source Software (OSS) is everywhere

- Huge boost for innovation! (e.g., reduced time to market)
- 96% of (non-open) software products depend on open source (2022).
- Open source is at the heart of the global digital infrastructure.

Open source security

Open source software can be freely used, copied, and modified.

Open Source Software (OSS) is everywhere

- Huge boost for innovation! (e.g., reduced time to market)
- 96% of (non-open) software products depend on open source (2022).
- Open source is at the heart of the global digital infrastructure.

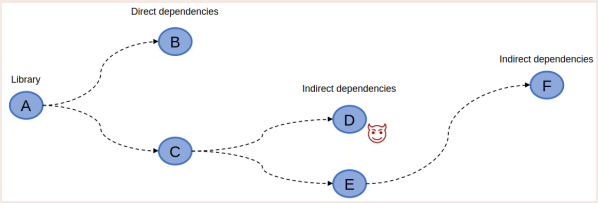
With great exposure comes great scrutiny…

- …by both good and bad actors.
- OSS is more and more targeted by attackers.
- Increased policy attention to secure OSS, e.g.:
  - US: Biden's executive orders (2022, Jan 2025!)
  - EU: CRA, progressively coming into effect

Software supply chain attacks

Reusing OSS via dependencies

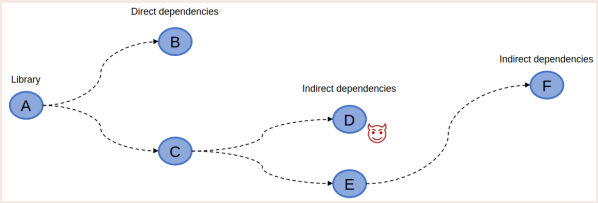
- Software dependencies: a popular way of reusing open source software.
- Software product A uses functionalities implemented in OSS product B . . . and so on.

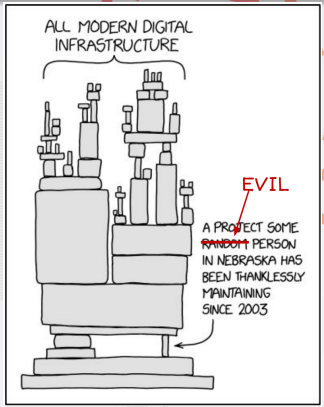


Software supply chain attacks

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), ...

What does Software Heritage bring to the table?

- The largest archive that guarantees the:
  1. availability,
  2. integrity, and
  3. traceability of (OSS) source code.

Securing open source with Software Heritage

What does Software Heritage bring to the table?

- The largest archive that guarantees the:
  1. availability,
  2. integrity, and
  3. traceability of (OSS) source code.
- A universal, open knowledge base of *facts* about open source software…
- …that can be leveraged by everyone (not only the big players) to secure OSS.

Securing open source with Software Heritage

What does Software Heritage bring to the table?

- The largest archive that guarantees the:
  1. availability,
  2. integrity, and
  3. traceability of (OSS) source code.
- A universal, open knowledge base of *facts* about open source software…
- …that can be leveraged by everyone (not only the big players) to secure OSS.

SWHSec project

swhsec.github.io

- 2023–2027 R&D project, funded by French national CampusCyber
- 8 research teams, including SWH core

Axes: (1) extending SWH with security info + (2) code analysis, dependency analysis, vulnerability tracking, automatic vulnerability fixing, … at SWH scale.

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*.

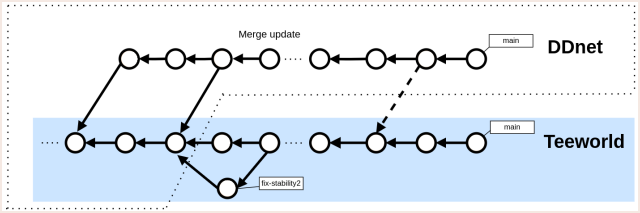
One-day vulnerabilities in open source

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*.

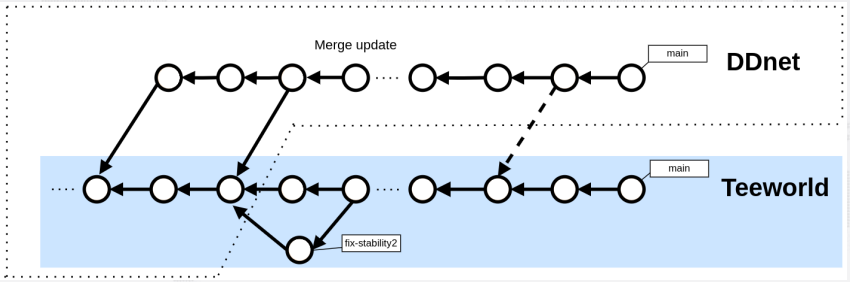
Reusing OSS via forks

Open source is also reused via forking: (1) start from existing OSS (e.g., Teeworlds game), (2) create your own (e.g., DDnet), (3) periodically integrate changes.



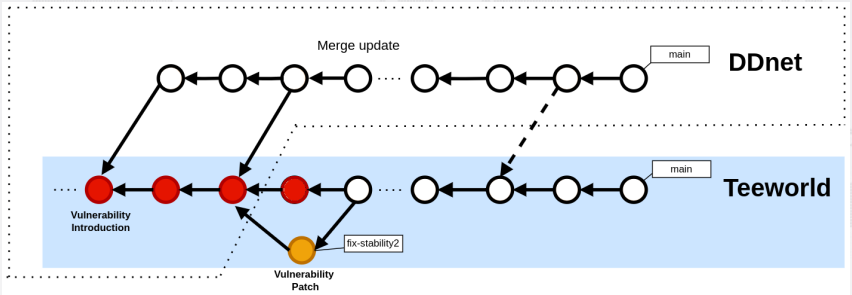
Vulnerability propagation through forks

- 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.



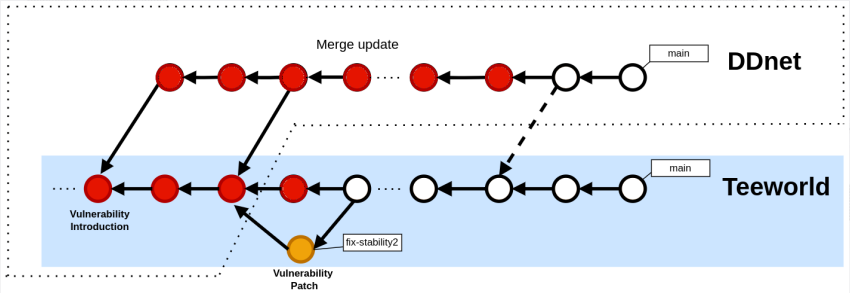
Vulnerability propagation through forks

- 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.



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

1. Start from a public DB of vuln. introduced/fixed in public commits (e.g., OSV.dev).
2. "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.
3. Inform maintainers of vulnerable forks. (After validation.)

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

Approach

1. Start from a public DB of vuln. introduced/fixed in public commits (e.g., OSV.dev).
2. "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.
3. Inform maintainers of vulnerable forks. (After validation.)

Early results

- Starting from 7126 projects, identified 2.2 M (million) forks of repositories referenced by OSV.dev, containing vulnerable commits; 1.7 M forks vulnerable in their most recent commit.
- 86.6 M vulnerable commits were specific to forks and span 312 development platforms, not findable with current tools.
- Among 65 manually vetted cases in popular forks, 3 confirmed crit. vulnerabilities.

R. Lefeuvre, C. Reux, S. Zacchiroli, O. Barais, B. Combemale. *One-day Vulnerabilities Across Open Source Forks*. To appear, 2025.

Outline

Software [License] Compliance

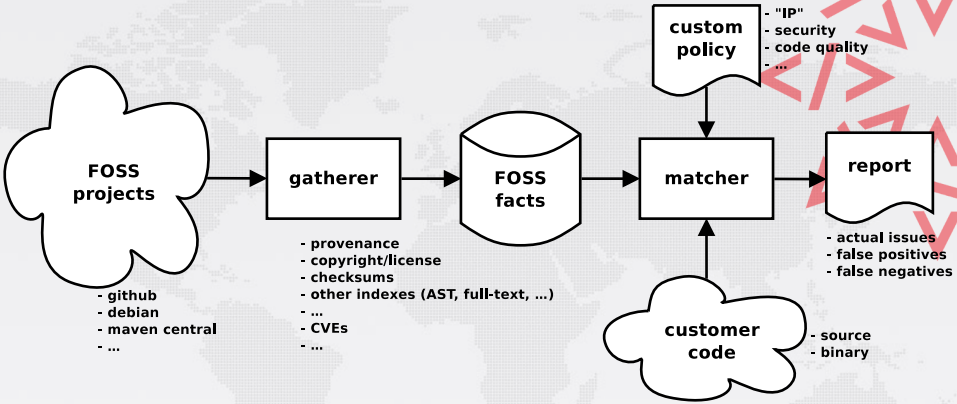
What Is Software License Compliance?

Software license compliance *is the process of ensuring that your company is only using software it is authorized to use. The most fundamental part of this process is comparing the way an organization is using software to the software licenses that the organization has purchased. This involves tracking installations and usage, keeping meticulous records, and understanding the terms of software licenses.* — Thales, 2025

- Particularly relevant for open source software, due to abundance of use.
- Common more general interpretation: not only about licenses, but all relevant characteristics (e.g., security: known vulnerabilities, historical track record, etc.) of all software components present in your software supply chain.
- KYSW: Know Your SoftWare (again!)

S. Phipps, S. Zacchiroli. *Continuous Open Source License Compliance.* IEEE Computer 53(12): 115-119 (2020).

Anatomy of a KYSW toolchain



A code scanner is the key ingredient of all KYSW toolchains: it scans a local source code base and compares it to a FOSS knowledge base, summarizing findings.

Open compliance vs Source code scanning

Definition (Open Compliance)

The pursuit of compliance with *license obligations* and other *best practices* for the management of open source software components, using only open technologies such as: open source software, open data information, and open access documentation.

Why

Reduced lock-in risks, lower total cost of ownership (TCO), crowdsourcing, alignment with FOSS community ethos.

Open compliance vs Source code scanning

Definition (Open Compliance)

The pursuit of compliance with *license obligations* and other *best practices* for the management of open source software components, using only open technologies such as: open source software, open data information, and open access documentation.

Why

Reduced lock-in risks, lower total cost of ownership (TCO), crowdsourcing, alignment with FOSS community ethos.

Q: Can we build an industry-grade source code scanning tool, compliant with Open Compliance principles, on top of Software Heritage?

SWH Scanner

Vision

`swh-scanner` is an open source and open data source code scanner for open compliance workflows, backed by the largest public archive of FOSS source code.

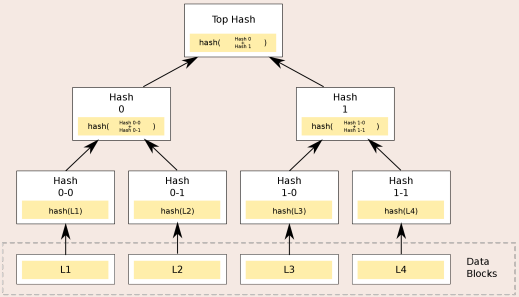
Design

- Partition a source tree into known (= published before) v. unknown
- Provide provenance information on demand
- Software Heritage Archive as ground truth for public code
- Merkle DAG model and SWHIDs for maximum efficiency
- File-level granularity

Code: gitlab.softwareheritage.org/swh/devel/swh-scanner (GPL 3+)
Package: pypi.org/project/swh.scanner

Leveraging the Software Heritage data model for efficient scanning

Merkle DAG



Efficient scanning

- If a node (e.g., the root directory of a project) is known to the Software Heritage archive, all contained files and directories are known as well → no need to query for them!
- If a node is not known, we recurse to children and stop querying when reaching known nodes (e.g., embedded copies of 3rd party FOSS code or previous versions)

Daniele Serafini, Stefano Zacchiroli
Efficient Prior Publication Identification for Open Source Code
OSS+OpenSym 2022. ACM 2022.
https://hal.science/hal-03735961/

Demo

Setup

```
$ pip install swh-scanner

$ swh scanner setup
$ swh scanner scan $PROJECT_PATH
```

Setup

```
$ pip install swh-scanner

$ swh scanner setup
$ swh scanner scan $PROJECT_PATH
```

Demo

swh-scanner demo — Efficiency

```
$ du -sh --exclude=.git /srv/src/linux/git
4,1G /srv/src/linux/git
```

```
$ time swh scanner scan /srv/src/linux
Files:                  78277
           known:       78267 ( 99%)
directories:             5085
      fully-known:       5081 ( 99%)
  partially-known:          4 (  0%)

38,65s user 4,71s system 81% cpu 53,127 total
```

```
$ swh scanner scan --output-format ndjson /srv/src/linux/git | grep false
...
{"scripts/kconfig/symbol.o": {"swhid": "swh:1:cnt:874f19...", "known": false}}
...
```

Outline

Can I train my AI on Software Heritage?

Yes, but only in a principled way.

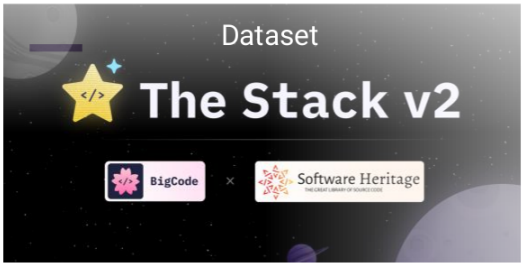
October 19, 2023

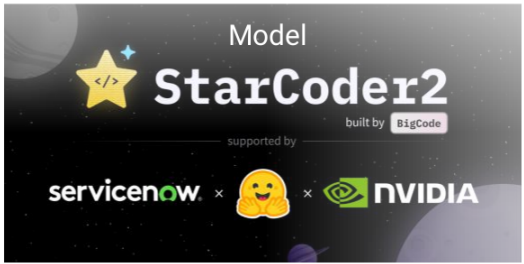
Software Heritage Statement on Large Language Models for Code

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

Can we build a state-of-the-art code-LLM that way?





Released February 28th 2024

Yes one can build **the best open LLM for code available** while fully adhering to the Software Heritage principles for responsible LLMs, …

and even more: the full training pipeline is made public too!

Anton Lozhkov et al. *StarCoder 2 and The Stack v2: The Next Generation.* arXiv:2402.19173, 2024

Code Commons: A digital commons for responsible AI

CodeCommons

Open, responsible, and transparent AI: Our shared goal

CodeCommons is an ambitious project to create the world's most comprehensive digital commons for code

Building on the existing foundation of Software Heritage, the largest publicly available source code archive, CodeCommons aims to bring into one place all the **critical** and **qualified** information needed to create **smaller**, **better** datasets for the next generation of AI tools.

At its core, the project prioritizes transparency and traceability, enabling model builders and users to **respect creators' rights** while promoting **sovereign** and **sustainable** AI.

codecommons.org

Outline

Conclusion

- Software Heritage is the largest archive in the world of public code and its development history.
- It is a gold mine of research leads in software engineering, software security, …
- It is also a vast knowledge base providing software integrity and provenance of public code, for industry applications related to the software supply chain.

Research on Software Heritage

www.softwareheritage.org/publications

Questions?

- Ask me!

Contact

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

Appendix

An international, non profit initiative for the long term

Sharing the vision

UNESCO
United Nations
Educational, Scientific and
Cultural Organization

And many more ...

www.softwareheritage.org/support/testimonials

An international, non profit initiative for the long term

Sharing the vision

UNESCO
United Nations Educational, Scientific and Cultural Organization

And many more ...
www.softwareheritage.org/support/testimonials

Donors, members, sponsors

Inria

Diamond sponsors: cea, IBM, Microsoft

Platinum sponsors: cnrs, Ministère de l'Enseignement supérieur et de la Recherche, Infosys, HUAWEI

Gold sponsors: aws, Google, openinventionnetwork, Université Paris Cité

Silver sponsors: AdaCore, Ministère des Armées, DGA, Premier Ministre, GitHub, Sorbonne Université, Università di Pisa

Bronze sponsors: cnes, Inria, Sant'Anna, Scuola Normale Superiore, Université de Lorraine, université PARIS-SACLAY

www.softwareheritage.org/support/sponsors

we are all concerned, anyone can join and help