

Source Code Tracking at Software Heritage Scale

for compliance, open science, and security

Stefano Zacchioli

Software Heritage – zack@upsilon.cc, [@zacchiro](https://twitter.com/zacchiro)

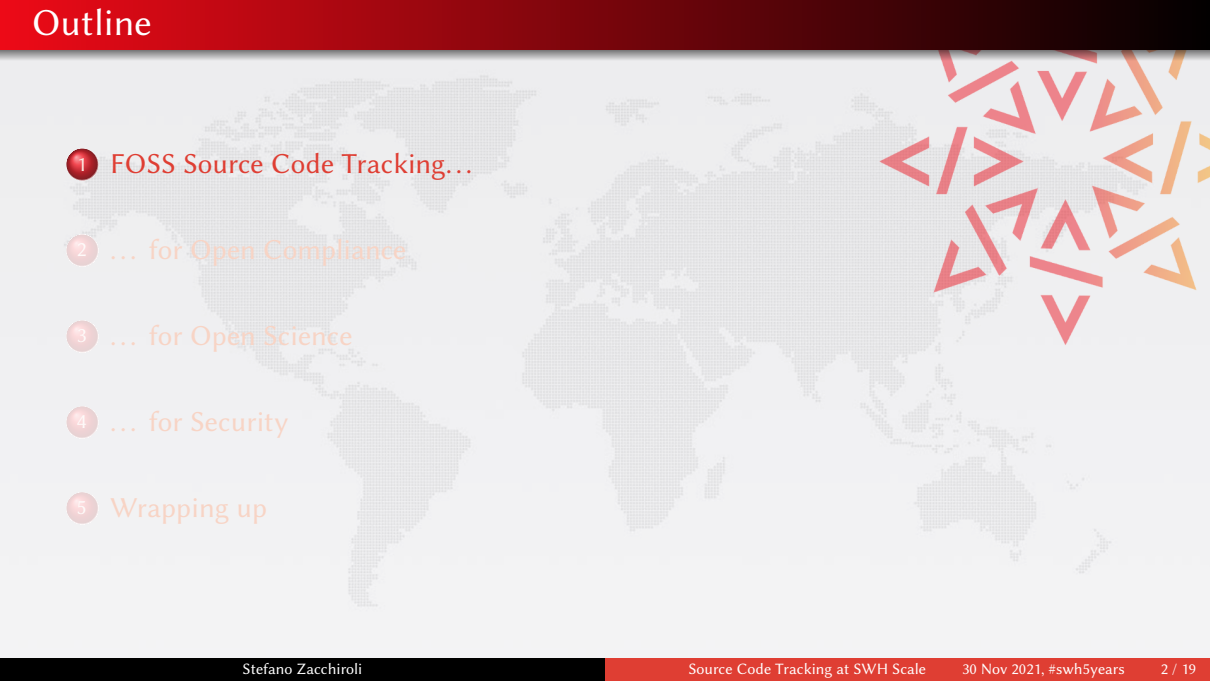
30 Nov 2021

#swh5years — Sponsors meeting
UNESCO



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE



1 FOSS Source Code Tracking...

2 ... for Open Compliance

3 ... for Open Science

4 ... for Security

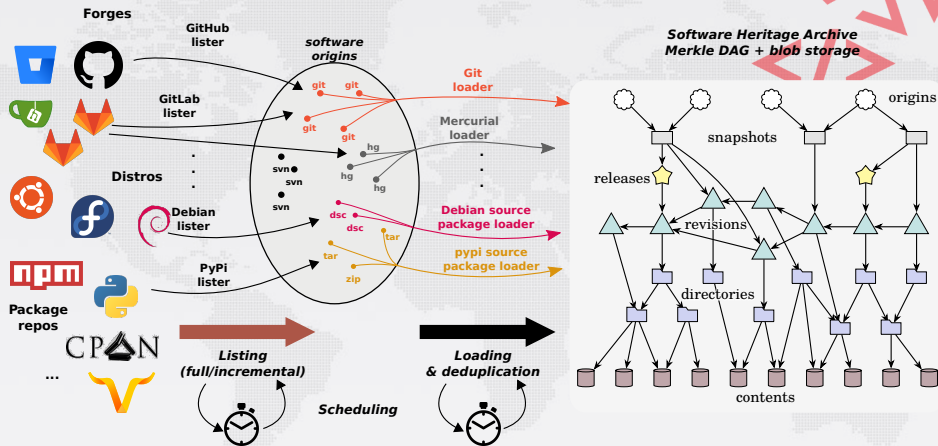
5 Wrapping up

The largest free/open source software archive



The largest public source code archive in the world (and growing!)

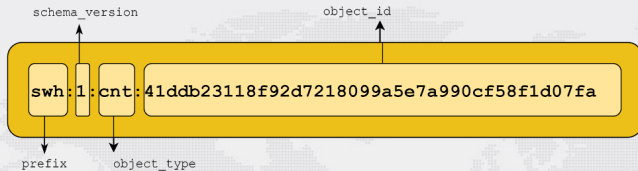
Automation and storage



Full development history permanently archived in a uniform data model.

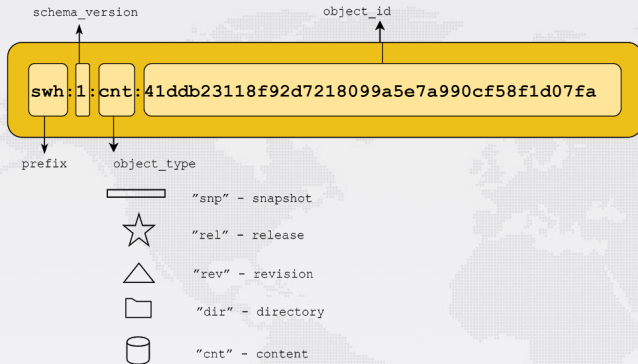
Meet the Software Heritage Identifiers (SWHIDs)

(full spec)



Meet the Software Heritage Identifiers (SWHIDs)

(full spec)







An emerging standard

- in Linux Foundation's [SPDX 2.2](#)
- IANA-registered "swh:" URI prefix
- WikiData property [P6138](#)



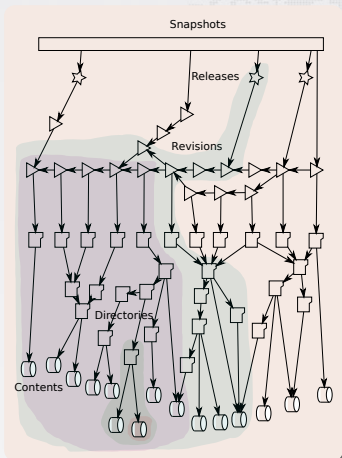
An emerging standard

- in Linux Foundation's [SPDX 2.2](#)
- IANA-registered "swh:" URI prefix
- WikiData property [P6138](#)

Examples

- [Apollo 11 AGC excerpt](#)
- [Quake III rsqrt](#)

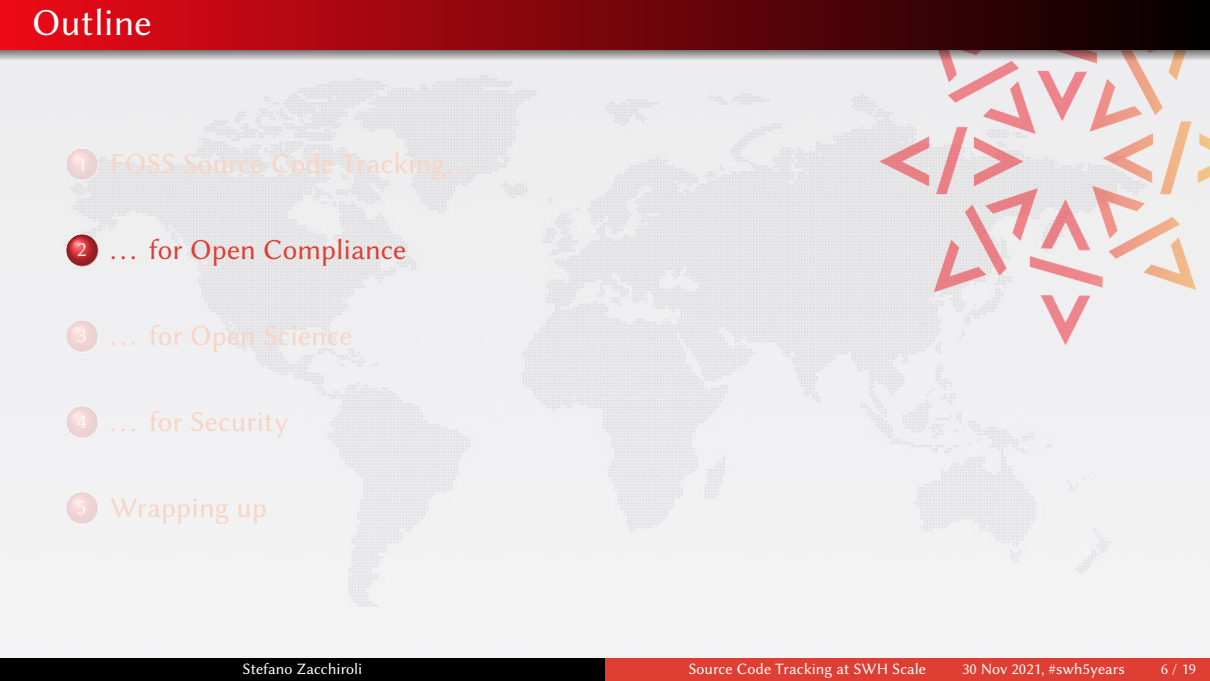
"It's Turtles SWHIDs all the way down"



Reference **any source code artifact** that has ever been shared—source code file, tree, commit, release, repository state—using the same, standard identifier.

Try it out:

```
$ pip install swh.model[cli]
$ swh identify /srv/src/linux/kernel/
swh:1:dir:b770a2aed8db52df737f88f18ca6bf39a1582240
```



1 FOSS Source Code Tracking...

2 ... for Open Compliance

3 ... for Open Science

4 ... for Security

5 Wrapping up

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.

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.

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.

We still lack a source code scanning tool that is compliant with Open Compliance principles and addresses industry practical needs.

Can we build one on top of Software Heritage?

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.

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 (of the current prototype)

- query the Software Heritage archive as source of truth about public code
- leverages the Merkle DAG model and SWHIDs for maximum scanning efficiency
 - e.g., no need to query the back-end for files contained in a known directory
- file-level granularity
- output: source tree partition into known (= published before) v. unknown

Code: forge.softwareheritage.org/source/swh-scanner (GPL 3+)

Package: pypi.org/project/swh.scanner

swh-scanner — Example

```
$ pip install swh.scanner

$ swh scanner scan -f json /srv/src/linux/kernel
{
  [...]
  "/srv/src/linux/kernel/auditsc.c": {
    "known": true,
    "swhid": "swh:1:cnt:814406a35db163080bbf937524d63690861ff750" },
  "/srv/src/linux/kernel/backtracetest.c": {
    "known": true,
    "swhid": "swh:1:cnt:a2a97fa3071b1c7ee6595d61a172f7ccc73ea40b" },
  "/srv/src/linux/kernel/bounds.c": {
    "known": true,
    "swhid": "swh:1:cnt:9795d75b09b2323306ad6a058a6350a87a251443" },
  "/srv/src/linux/kernel/bpf": {
    "known": true,
    "swhid": "swh:1:dir:fcd9987804d26274fee1eb6711fac38036ccaee7" },
  "/srv/src/linux/kernel/capability.c": {
    "known": true,
    "swhid": "swh:1:cnt:1444f3954d750ba685b9423e94522e0243175f90" },
  [...]
}
0,53s user 0,61s system 145% cpu 1,867 total
```

swh-scanner — Example (cont.)

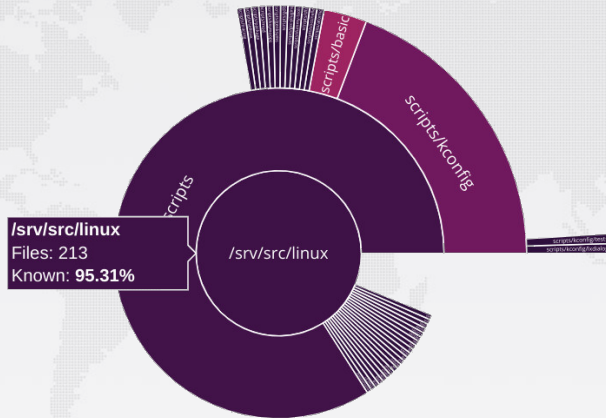
```
$ du -sh --exclude=.git /srv/src/linux
1,1G /srv/src/linux

$ time swH scanner scan -f json -x *.git /srv/src/linux
{
  [...]
  "/srv/src/linux/arch": {
    "known": true,
    "swHid": "swH:1:dir:590c329d3548b7d552fc913a51965353f01c9e2f" },
  [...]
  "/srv/src/linux/scripts/kallsyms.c": {
    "known": true,
    "swHid": "swH:1:cnt:0096cd9653327584fe62ce56ba158c68875c5067" },
  "/srv/src/linux/scripts/kconfig": {
    "known": false,
    "swHid": "swH:1:dir:548afc93bd01d2fba0dfcc0fd8c69f4b082ab8c6" },
  "/srv/src/linux/scripts/kconfig/.conf.o.cmd": {
    "known": false,
    "swHid": "swH:1:cnt:0d8be19e430c082ece6a3803923ad6ecb9e7d413" },
  [...]
}
20,84s user 1,52s system 103% cpu 21,540 total
```

swh-scanner — Example (cont.)

Interactive mode to drill-down and inspect unknown files:

```
$ swh scanner scan -f sunburst -x *.git /srv/src/linux
```



swh-scanner — Going further

swh-scanner shows that *it is possible* to create a source code scanner that is both open source and backed by the most comprehensive open data FOSS archive.

swh-scanner — Going further

swh-scanner shows that *it is possible* to create a source code scanner that is both open source and backed by the most comprehensive open data FOSS archive.

Roadmap

swh-scanner is *not a production-ready scanner*. The following features are still missing:

- license information → in-house scanning + ClearlyDefined
- provenance information → Software Heritage crawling info
- increase granularity to snippet/SLOC

Some of these are low-hanging fruits, some require substantial R&D investments.

swh-scanner shows that *it is possible* to create a source code scanner that is both open source and backed by the most comprehensive open data FOSS archive.

Roadmap

swh-scanner is *not a production-ready scanner*. The following features are still missing:

- license information → in-house scanning + ClearlyDefined
- provenance information → Software Heritage crawling info
- increase granularity to snippet/SLOC

Some of these are low-hanging fruits, some require substantial R&D investments.

Feedback welcome

- feel free to play with swh-scanner, feedback is very welcome!
- caveat: intensive use will result in hitting the API rate-limit



1 FOSS Source Code Tracking...

2 ... for Open Compliance

3 ... for Open Science

4 ... for Security

5 Wrapping up

Use case 1: Researcher (Prior Art Search)

Verify that the novelty status of the replication package of a paper under submission matches expectations:

- Original code written for the experiment should be novel
- Reused 3rd-party FOSS components should not be novel (verifying this also helps with spotting undesirable local patches)

Use case 2: Open Science Publisher (Plagiarism Detection)

- Verify that the source code part of papers, submitted as original work by the authors, are in fact original.
- This is already standard publisher procedure for the *textual part* of submitted papers, but it isn't yet for software source code.

Prior Art Search & Plagiarism Detection — Example

Let's verify that in the replication package of our ICSE 2021 paper about sw-h-fuse we have used a public, archived version of the package:

```
$ sw-h scanner scan -f ndjson replication-package/sw-h-fuse/  
{"." : {"sw-hid": "sw-h:1:dir:3d4f903b[...]", "known": true}}  
...
```

Let's check that the rest of the replication package is novel (at submission time, it will be archived in Software Heritage at camera-ready time):

```
$ sw-h scanner scan -f ndjson replication-package  
{"." : {"sw-hid": "sw-h:1:dir:14ecd6[...]", "known": false}}  
...
```

- **Researchers** can integrate these checks in their pre-submission checklists
- **Publishers** can integrate these checks into existing plagiarism detection pipelines, making results available to scientific editors and reviewers



1 FOSS Source Code Tracking...

2 ... for Open Compliance

3 ... for Open Science

4 ... for Security

5 Wrapping up

Tracking unfixed vulnerabilities

OpenSSL (prior to 1.0.1i) has vulnerabilities fixed on 2015-01-01

let's find software that still uses unpatched versions after 2015-01-01

Tracking unfixed vulnerabilities

OpenSSL (prior to 1.0.1i) has vulnerabilities fixed on 2015-01-01

let's find software that still uses unpatched versions after 2015-01-01

Obtain SWHIDs of known vulnerable version of d1_both.c:

```
$ for f in openssl-1.0.1*/ssl/d1_both.c; do swh-identify $f; done
```

Tracking unfixed vulnerabilities

OpenSSL (prior to 1.0.1i) has vulnerabilities fixed on 2015-01-01

let's find software that still uses unpatched versions after 2015-01-01

Obtain SWHIDs of known vulnerable version of d1_both.c:

```
$ for f in openssl-1.0.1*/ssl/d1_both.c; do swh-identify $f; done
```

swh:1:cnt:0a84f957118afa9804451add380eca4719a9765e	openssl-1.0.1-beta1/ssl/d1_both.c
swh:1:cnt:7a5596a6b373aeabbd6d8d674f0e20b1618c5012	openssl-1.0.1f/ssl/d1_both.c
swh:1:cnt:2e8cf681ed0976e2b16460170fda27c77cfec6cc	openssl-1.0.1g/ssl/d1_both.c
swh:1:cnt:04aa23107ec53c184505e98091306c7391091bb5	openssl-1.0.1h/ssl/d1_both.c
swh:1:cnt:de8bab873f2cf114d0d1b3e49acfa09bb9d0e4f7	openssl-1.0.1/ssl/d1_both.c

Tracking unfixed vulnerabilities

OpenSSL (prior to 1.0.1i) has vulnerabilities fixed on 2015-01-01

let's find software that still uses unpatched versions after 2015-01-01

Obtain SWHIDs of known vulnerable version of d1_both.c:

```
$ for f in openssl-1.0.1*/ssl/d1_both.c; do swh-identify $f; done
```

swh:1:cnt:0a84f957118afa9804451add380eca4719a9765e	openssl-1.0.1-beta1/ssl/d1_both.c
swh:1:cnt:7a5596a6b373aeabbd6d8d674f0e20b1618c5012	openssl-1.0.1f/ssl/d1_both.c
swh:1:cnt:2e8cf681ed0976e2b16460170fda27c77cfec6cc	openssl-1.0.1g/ssl/d1_both.c
swh:1:cnt:04aa23107ec53c184505e98091306c7391091bb5	openssl-1.0.1h/ssl/d1_both.c
swh:1:cnt:de8bab873f2cf114d0d1b3e49acfa09bb9d0e4f7	openssl-1.0.1/ssl/d1_both.c

What repositories still ship a vulnerable version of OpenSSL after the fix?

Example: vulnerable software

[resources](#) / [openssl-1.0.1f](#) / [ssl](#) / [d1_both.c](#)

The screenshot shows a GitHub repository page. At the top, the repository path is `resources / openssl-1.0.1f / ssl / d1_both.c`. Below the path, there are tabs for `Code`, `Issues` (0), `Pull requests` (0), `Projects` (0), `Wiki`, and `Insights`. The `Code` tab is selected. Below the tabs, the text "History for `resources / openssl-1.0.1f / ssl / d1_both.c`" is displayed. Underneath, there is a section titled "Commits on Jun 11, 2015". A single commit is listed with the message "First Check-in" and the author "a[redacted] committed on 11 Jun 2015". To the right of the commit message are icons for the repository, the author, and the commit itself. At the bottom of the commit list, there are buttons for "Newer" and "Older".



Commits on Jun 11, 2015

How do we go from file/dir SWHIDs back to repositories?

At the scale of Software Heritage, querying efficiently *all* the places where a given SWHID can be found in is a challenging R&D problem, that nobody has ever tackled in its full generality (see Rousseau et al., Empir. Softw. Eng. (2020) for the scientific details).

- We are working on a **complete software provenance index** for the archive
- Meanwhile we have building blocks in place that address specific parts of the problem, based on a **compressed graph** representation of the archive (see Boldi et al., SANER (2020) for sci. details). They come with an API you can play with.

Code: forge.softwareheritage.org/source/swh-graph (GPL 3+)

Doc: docs.softwareheritage.org/devel/swh-graph

API: docs.softwareheritage.org/devel/swh-graph/api.html

(All) repositories containing vulnerable versions of `d1_both.c`:

```
$ swh-whereis swh:1:cnt:0a84f957118afa9804451add380eca4719a9765e
https://github.com/mathiassamuelson/openssl-dane-ms
https://gitorious.org/baserock-morphs/openssl.git
https://github.com/tack/openssl_tack
https://gitorious.org/myopenssl/myopenssl.git
https://bitbucket.org/xreach/android-external-openssl.git
[...]
```

Where `swh-whereis` is a trivial wrapper around the `swh-graph` API:

```
curl --silent --fail --location \
  "${API_URL}/graph/leaves/${SWHID}?direction=backward&resolve_origins=true"
```

Caveats

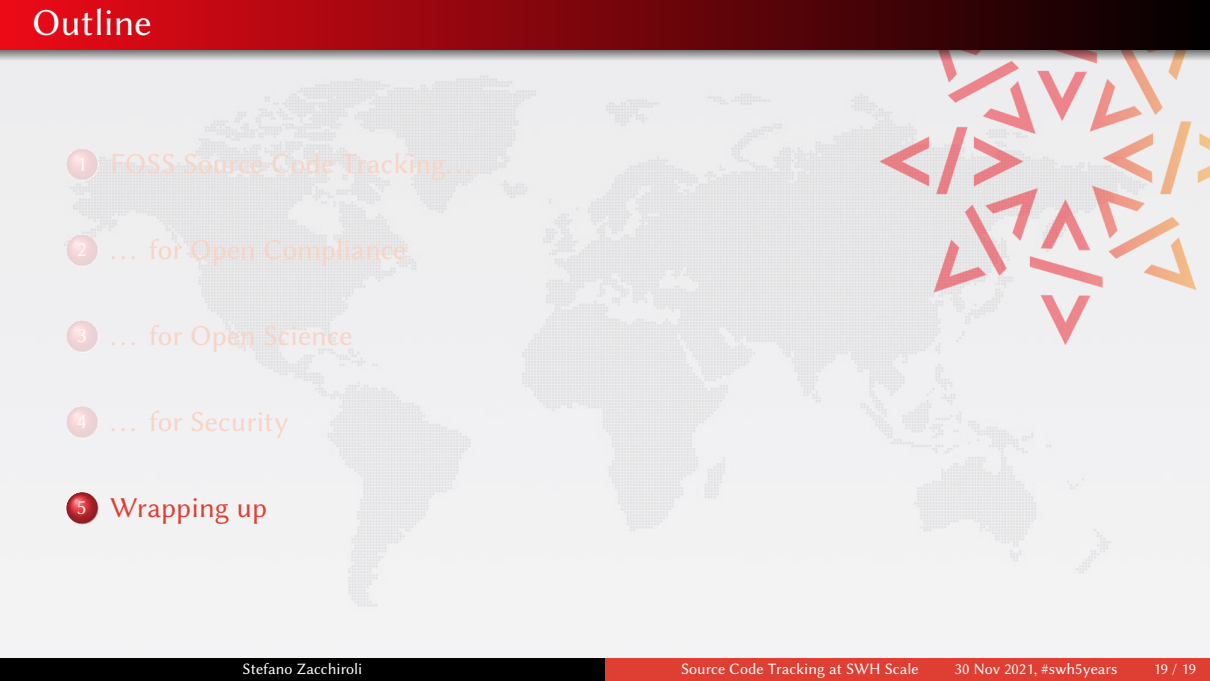
- no filtering/sorting on commit timestamps (yet)
- no filtering on path (e.g., OpenSSL should be included in a sub-dir, etc)

Same API, compliance use case

Find software that has been licensed under the original 1.0 version of the Affero GPL license (SWHID: swh:1:cnt:8f3209754390bbc58953d49701ed45c9d4a1a47f).

```
$ curl https://archive.softwareheritage.org/api/1/graph/randomwalk/\
    swh:1:cnt:8f3209754390bbc58953d49701ed45c9d4a1a47f/ori/\
    ?direction=backward&limit=-1&resolve_origins=true
https://github.com/uwsampa/grappa/
```

- note the random walk, for spot checks, examples, etc.
- (Grappa has since been re-released under the BSD 3-Clause license)



1 FOSS Source Code Tracking...

2 ... for Open Compliance

3 ... for Open Science

4 ... for Security

5 Wrapping up



Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

www.softwareheritage.org

[@swheritage](https://twitter.com/swheritage)

- thanks to its data model, Software Heritage provides a **global view** on the largest public collection of software source code artifacts
- this global view enables **source code tracking** of public code (FOSS, and more) at an unprecedented scale
- global source code tracking is a key building block for addressing use cases in domains such as **license compliance**, **open science**, and **security**
- we are laying the foundations for addressing these use cases

Contacts

Stefano Zacchiroli / zack@upsilon.cc / [@zacchiro](https://twitter.com/zacchiro) / [@zacchiro@mastodon.xyz](https://mastodon.xyz/@zacchiro)

Web API — Integrate your tools with the Software Heritage archive

RESTful API to programmatically access the Software Heritage archive

<https://archive.softwareheritage.org/api/>

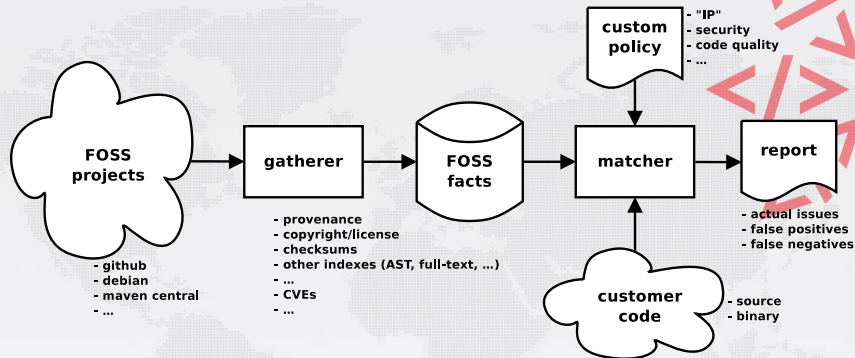
Features

- pointwise **browsing** of the archive
 - ... snapshots → revisions → directories → contents ...
- full access to the **metadata** of archived objects
- **crawling** information
 - *when have you last visited this Git repository I care about?*
 - *where were its branches/tags pointing to at the time?*

Endpoint index

<https://archive.softwareheritage.org/api/1/>

Anatomy of a KYSW toolchain



source: *A Community Take on the License Compliance Industry*, Stefano Zacchiroli, FOSDEM 2016, Legal and Policy Issues devroom,

<https://upsilon.cc/~zack/talks/2016/2016-01-31-fosdem-compliance.pdf>

A source **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.