Outline

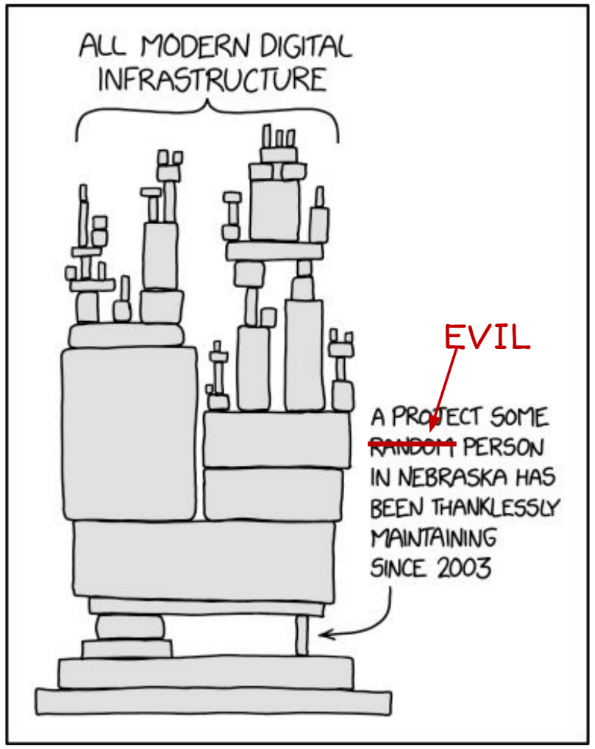
Cybersecurity

Stefano Zacchiroli

Co-founder and
Chief Scientific Officer

Context: OSS supply chain security & Software Heritage (redux)

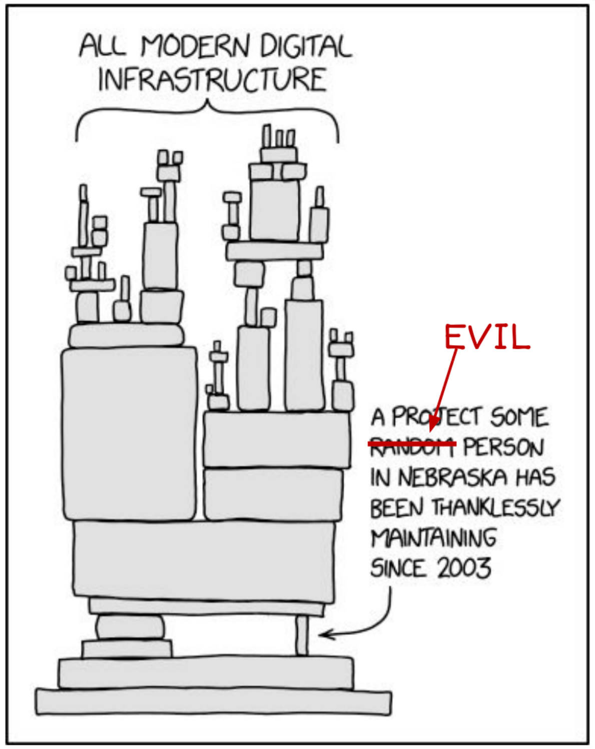
- Open Source Software (OSS) is at the heart of the global digital infrastructure.
- This attracts bad actors who leverage the software supply chain to target victims.
- Your attack surface includes all your OSS dependencies.


ALL MODERN DIGITAL INFRASTRUCTURE
EVIL
A PROJECT SOME ~~RANDOM~~ PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

based on xkcd.com/2347

Context: OSS supply chain security & Software Heritage (redux)

- Open Source Software (OSS) is at the heart of the global digital infrastructure.
- This attracts bad actors who leverage the software supply chain to target victims.
- Your attack surface includes all your OSS dependencies.


ALL MODERN DIGITAL INFRASTRUCTURE
EVIL
A PROJECT SOME ~~RANDOM~~ PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

based on xkcd.com/2347

What does Software Heritage (SWH) bring to the table?

- Availability, integrity, and traceability of all OSS source code.
- A universal, open knowledge base of *facts* about open source software.

Coming up: 2 concrete R&D examples of how to leverage SWH to increase OSS security.

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.

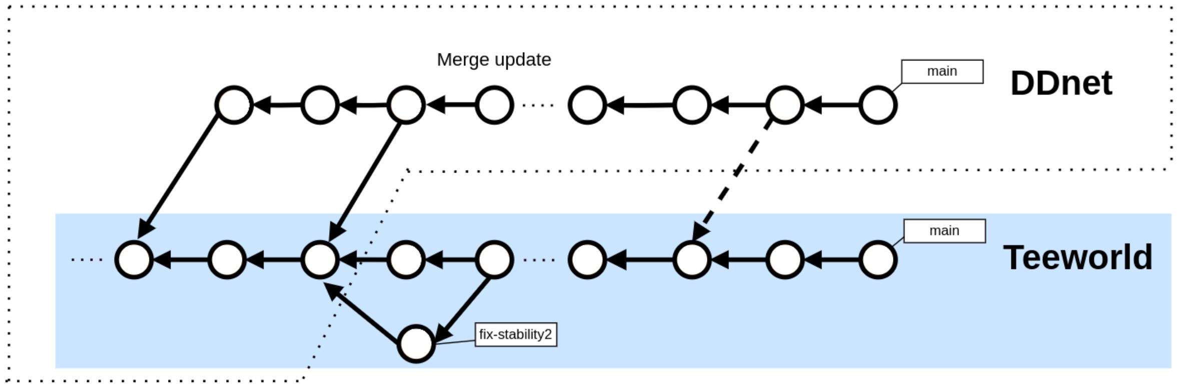
One-day vulnerabilities in open source

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.

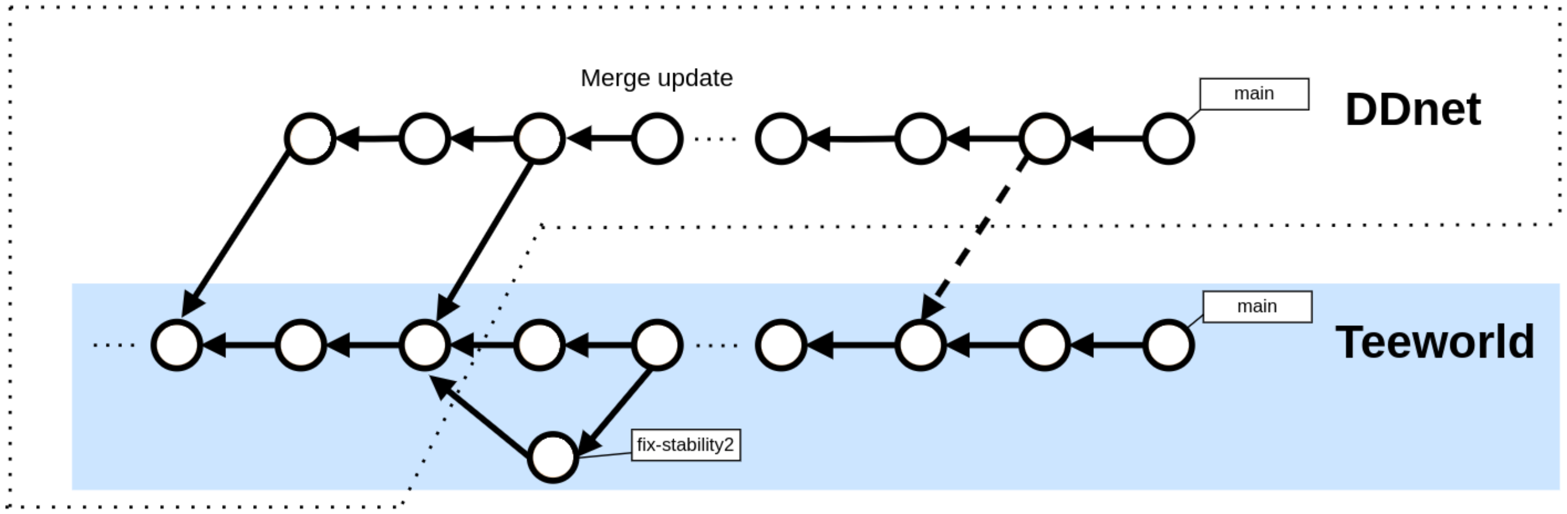
Reusing OSS via forks

…but OSS 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.



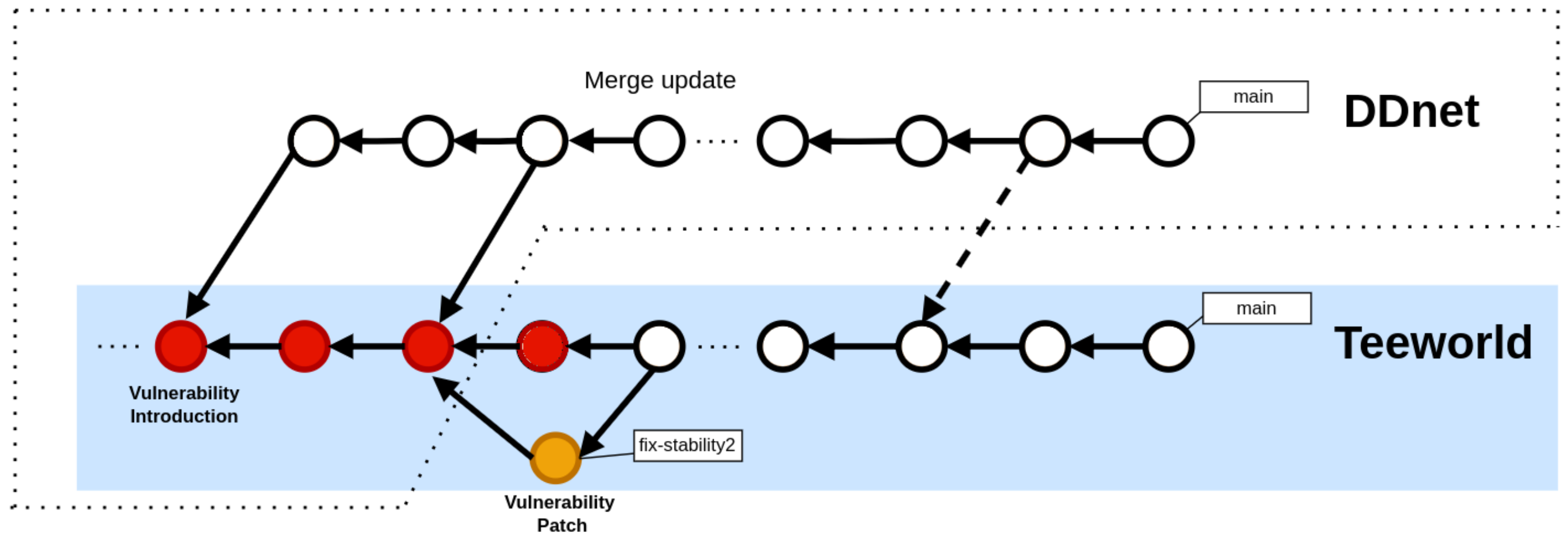
Vulnerability propagation through forks

- 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.



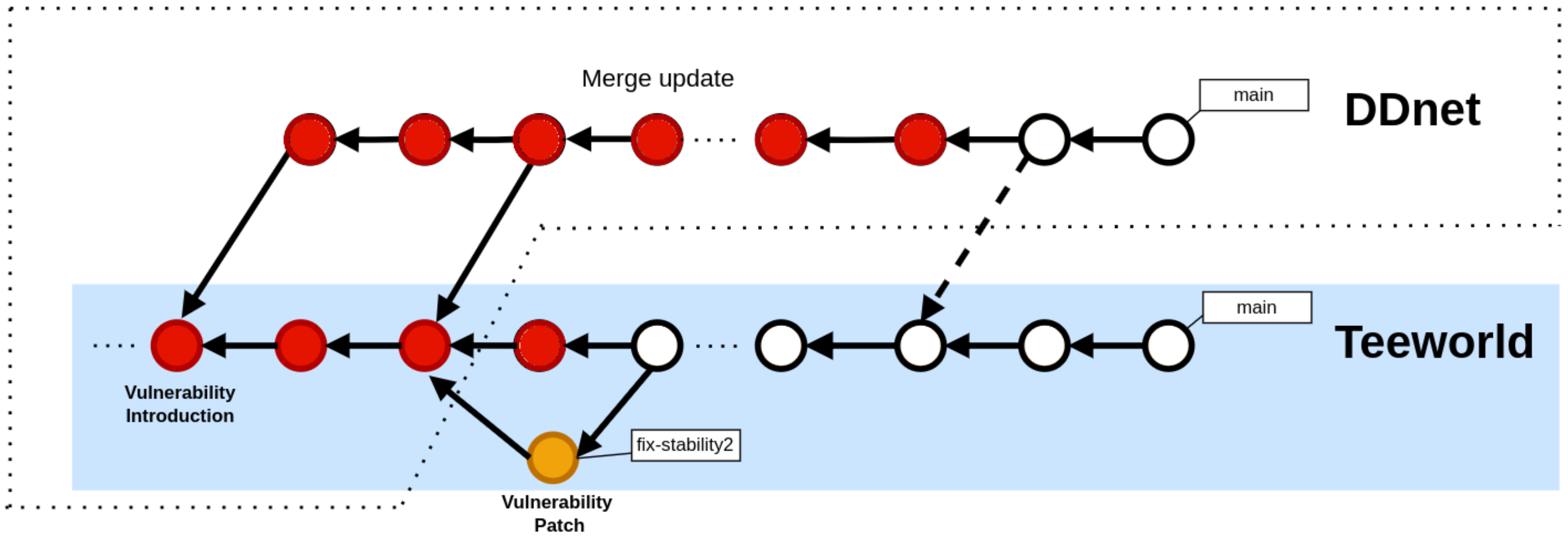
Vulnerability propagation through forks

- 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.)

Results

- Starting from 7162 repos in OSV, we identified 1.7 M forks potentially vulnerable in their most recent commit.
  - 86.6 M vulnerable commits were 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.

Romain Lefeuvre, Charly Reux, Stefano Zacchiroli, Olivier Barais, Benoit Combemale
Chasing One-day Vulnerabilities Across Open Source Forks
https://arxiv.org/abs/2511.05097, Nov 2025.

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?

Git repository alterations

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

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

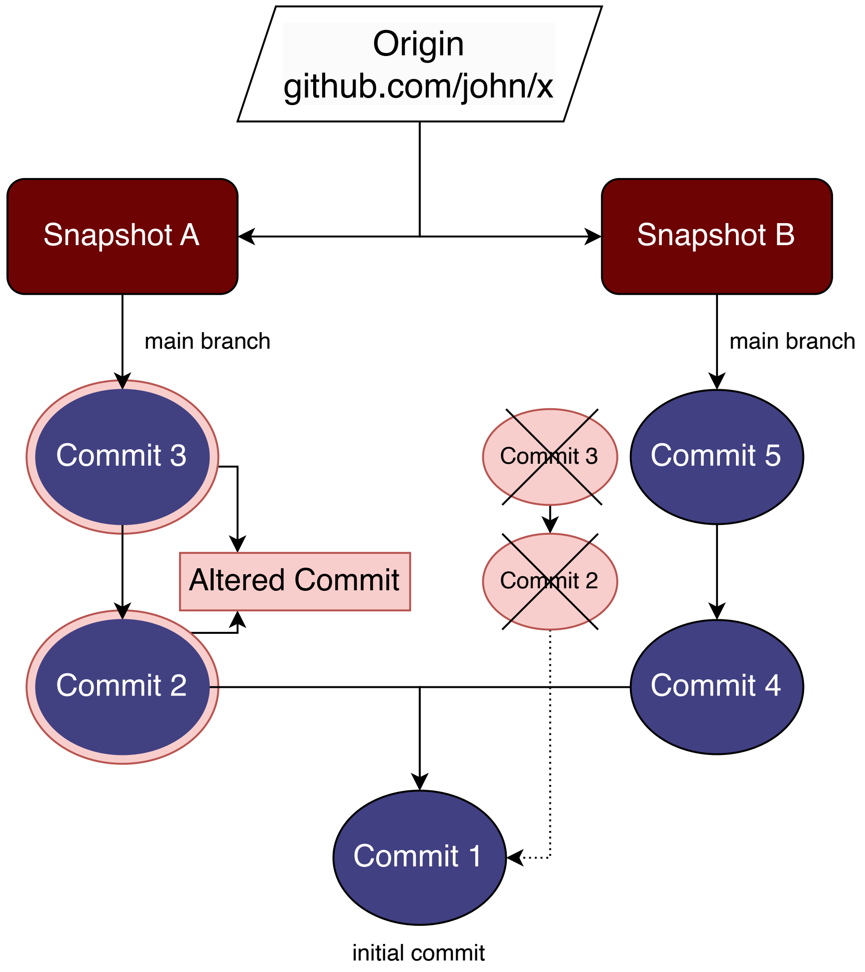
- Useful feature! E.g., to clean your code before sharing it
- Also annoying and risky on public branches
  - Hinders reproducibility and voids availability of specific Git objects
  - Supply chain concerns: what was altered and why?

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.

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



Key findings on destructive repository alterations

How often?

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

Where?

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

What?

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

Case studies

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
[...]
```

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

Current status and next steps, together

- These were preliminary research results that pave the road for next steps.
- We welcome your feedback, inputs, and help to pursue them together!

Current status and next steps, together

- These were preliminary research results that pave the road for next steps.
- We welcome your feedback, inputs, and help to pursue them together!

Detection of one-day vulnerabilities at the scale of public code

- We can deploy this at SWH scale, and make results accessible via a public API.
- Should we?

Detection of history alterations in Git repositories of interest

- Prototype stage, with good potential for supply chain security.
- Should we put more effort on this line of work?

Responsible disclosure

- If we discover novel security-relevant information at SWH scale, how can we enact responsible disclosure?

Current status and next steps, together

- These were preliminary research results that pave the road for next steps.
- We welcome your feedback, inputs, and help to pursue them together!

Detection of one-day vulnerabilities at the scale of public code

- We can deploy this at SWH scale, and make results accessible via a public API.
- Should we?

Detection of history alterations in Git repositories of interest

- Prototype stage, with good potential for supply chain security.
- Should we put more effort on this line of work?

Responsible disclosure

- If we discover novel security-relevant information at SWH scale, how can we enact responsible disclosure?

Contact

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