

# Did You Forkget It?

Detecting One-Day Vulnerabilities in Open-source  
Forks With Global History Analysis

19/02/2026

*Work under peer review*

Romain  
Lefeuvre

University of Rennes  
IRISA

Charly  
Reux

Inria  
IRISA

Stefano  
Zachirolli

LTCI, Télécom Paris,  
Polytechnic Institute of Paris

Olivier  
Barais

University of Rennes  
IRISA

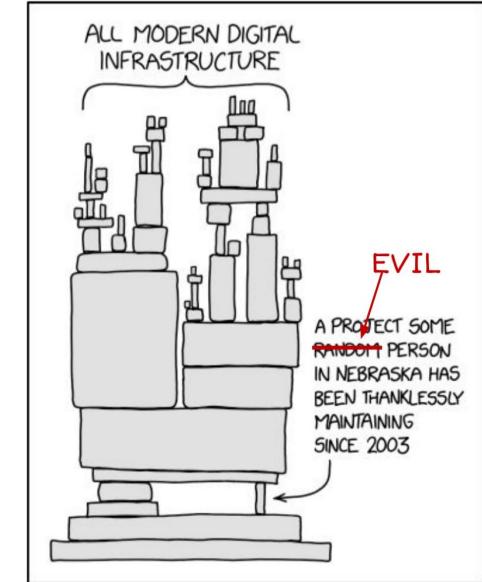
Benoit  
Combemale

Inria  
IRISA

# OSS supply chain security

- OSS are at the **heart** of modern and modular software[1]
- **96%** of studied private software contained **OSS dependency** [1] [2]
- A software **attack surface** includes all its OSS dependencies

**OSS security is critical**



[1] [https://www.linuxfoundation.org/hubfs/Research%20Reports/lfr\\_harvard\\_censusII\\_mar2022\\_042824b.pdf?hsLang=en](https://www.linuxfoundation.org/hubfs/Research%20Reports/lfr_harvard_censusII_mar2022_042824b.pdf?hsLang=en)

[2] <https://www.blackduck.com/resources/analyst-reports/open-source-security-risk-analysis.html#introMenu>

# One-day vulnerability

- Definition : Vulnerabilities that are **publicly known, but not fixed yet** in a software you use
- Challenge : identify them quickly and exhaustively, then apply countermeasure

## CVE-2024-3094 Detail

### Description

Malicious code was discovered in the upstream tarballs of xz, starting with version 5.6.0. Through a series of complex obfuscations, the liblzma build process extracts a prebuilt object file from a disguised test file existing in the source code, which is then used to modify specific functions in the liblzma code. This results in a modified liblzma library that can be used by any software linked against this library, intercepting and modifying the data interaction with this library.

### Metrics

[CVSS Version 4.0](#) [CVSS Version 3.x](#) [CVSS Version 2.0](#)

NVD enrichment offers reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

**CVSS 3.x Severity and Vector Strings:**



CNA: Red Hat, Inc.

Base Score:  10.0 CRITICAL

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

## CVE-2021-44228 Detail

### Description

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

### Metrics

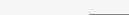
[CVSS Version 4.0](#) [CVSS Version 3.x](#) [CVSS Version 2.0](#)

NVD enrichment offers reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

**CVSS 3.x Severity and Vector Strings:**



NIST: NVD

Base Score:  10.0 CRITICAL

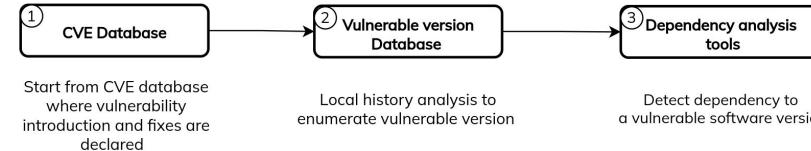
Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

*xz utils backdoor*

*Log4J vulnerability*

# One-day vulnerability in open source via declared dependencies

- Well-known and documented challenges
- Many tools available (OSV scanner, OWASP Dependency check, Snyk, Gitlab Dependency scanning, Dependabot ...)
- Based on CVE Databases ( OSV, NVD, CVE List, etc.)



# One-day vulnerability in open source via forking

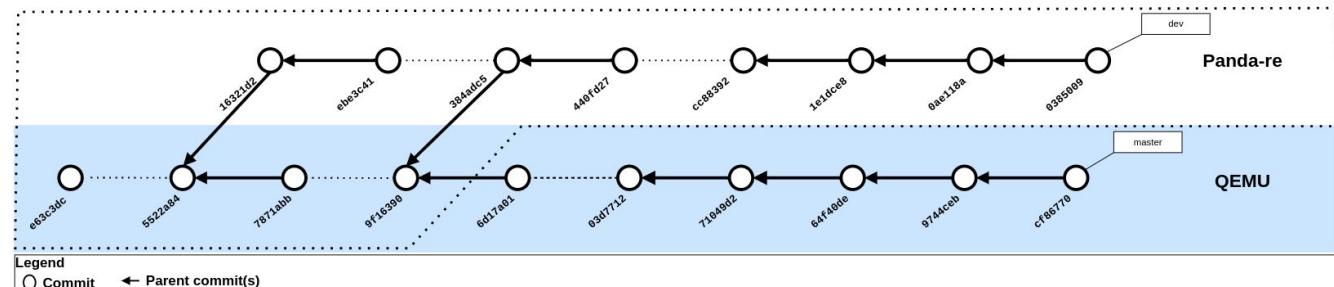
Definition :

- Creating a new repository by copying the source code and the history of an existing project [1]
- Shared commit fork [2] : Two repositories are considered as forks if they **shared a commit**

Typical scenario :

- 1) Start from existing OSS (e.g. QEMU)
- 2) Create your own (e.g. Panda-re)
- 3) Periodically integrate changes

More than 40% of repository hosted on GitHub are forks [3]



Deduplicated git history in a fork ecosystem

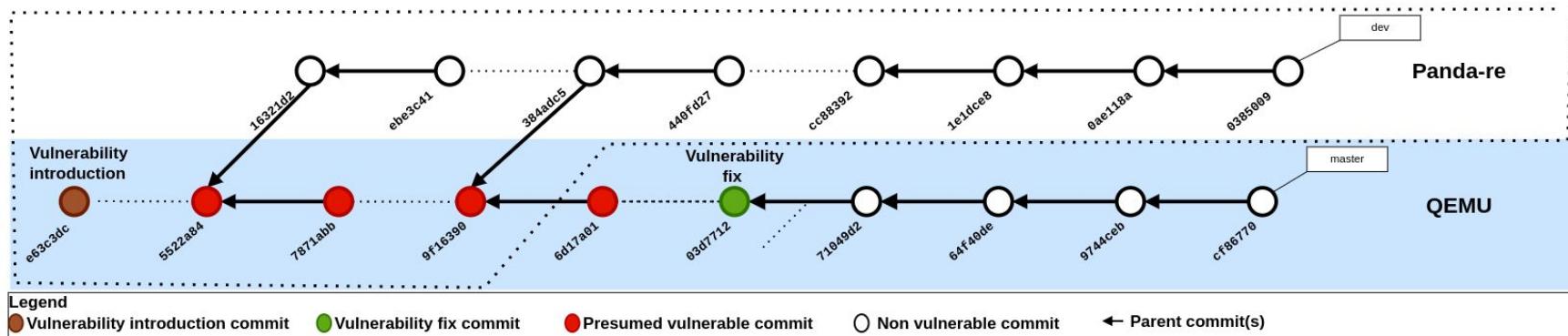
[1] Shurui Zhou, Bogdan Vasilescu, and Christian Kastner. 2020. How has forking changed in the last 20 years? a study of hard forks on GitHub. ICSE '20

[2] Antoine Pietri, Guillaume Rousseau, and Stefano Zucchirolli. 2020. Forking Without Clicking: on How to Identify Software Repository Forks. MSR '20

[3] Analysis based on an export of Software Heritage as of April 3, 2024

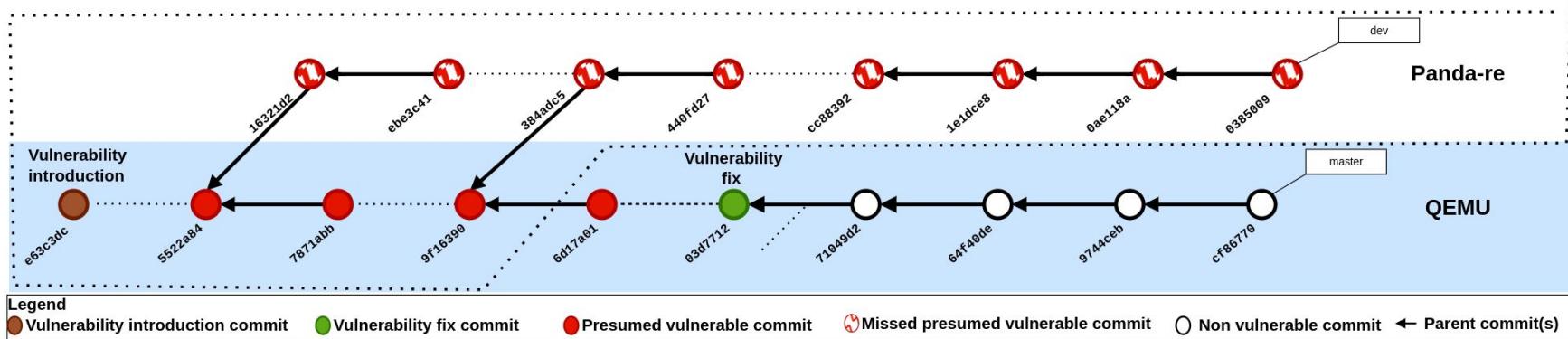
# One-day vulnerability in open source via forking

- Any 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.



# One-day vulnerability in open source via forking

- Any 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.



Current approach failed to handle vulnerable commit in forks

One-day vulnerability related to forking is not studied

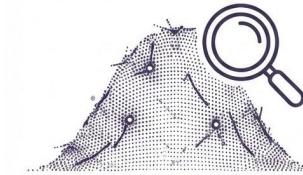
# Detecting One-Day Vulnerabilities in Open-Source Forks

## *Challenges*



### 1. Forks identification

Forks can be hosted on heterogeneous forge.



### 2. Large Scale analysis

## Research question

**RQ1 - Is it feasible to propagate the information about which commits introduce and fix known vulnerabilities to the global commit graph of public code?**

RQ2 - Can we use the global vulnerability propagation to identify unpatched forks?

# The Open Source Vulnerability (OSV) database as input



- Standardization initiative [1]
- Map vulnerability with affected software version and/or vulnerability event on git history
- Aggregate multiple database

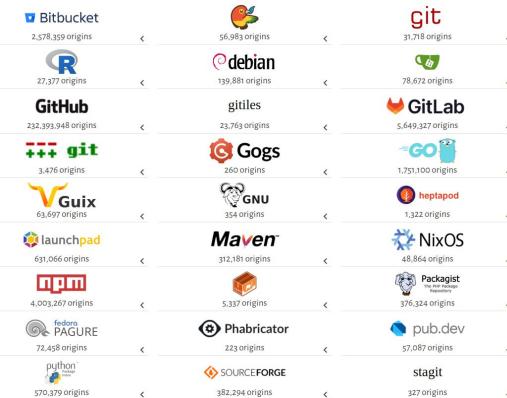
```
"ranges": [ {  
    "type": "GIT",  
    "repo": "https://github.com/owner/repo",  
    "events": [  
        { "introduced": "X" },  
        { "fixed": "Y" }  
    ]  
} ]
```

```
{ "last_affected": "Y" },
```

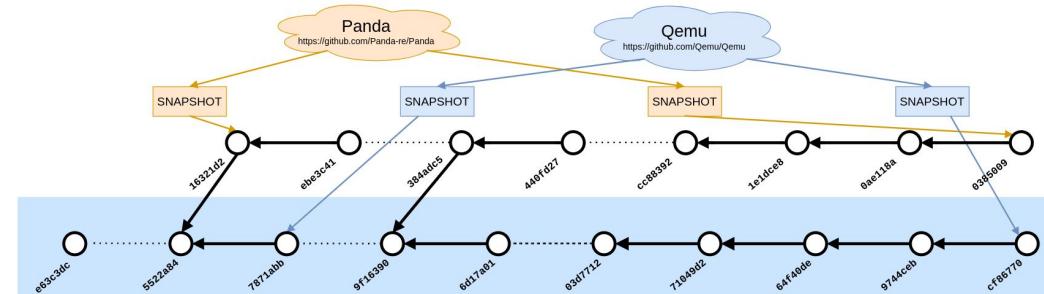
```
{ "limit": "Y" },
```

# Software Heritage, a perfect candidate

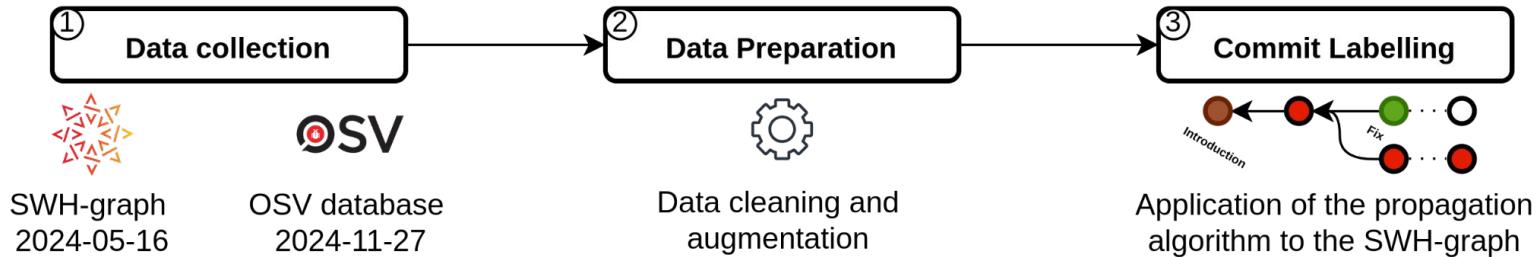
Cross forges



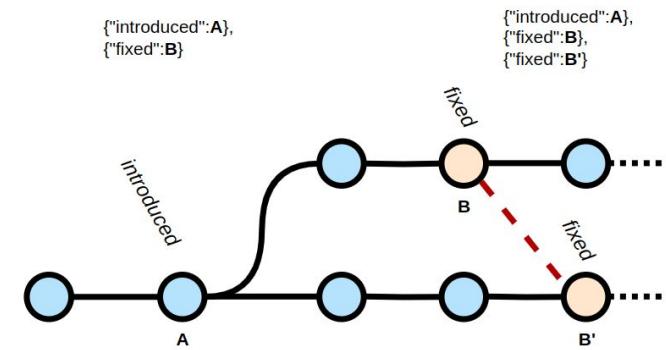
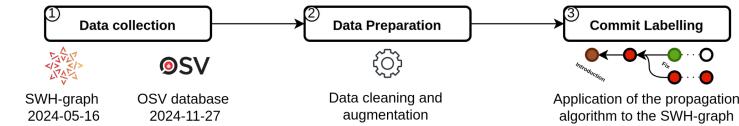
Deduplicated model



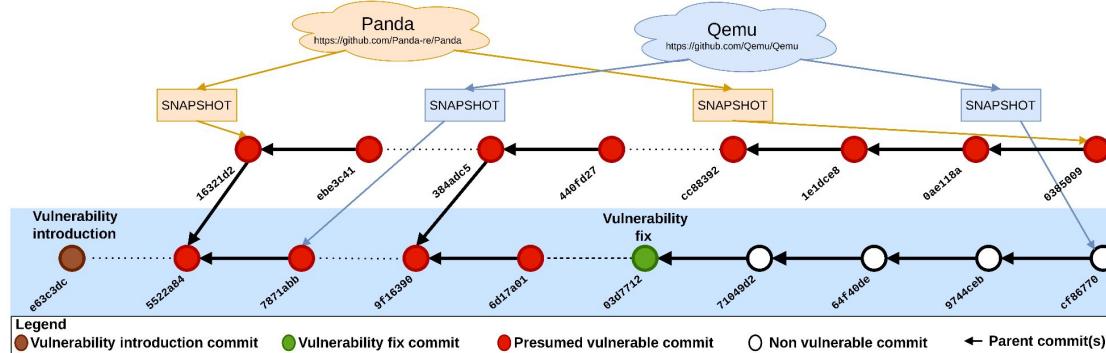
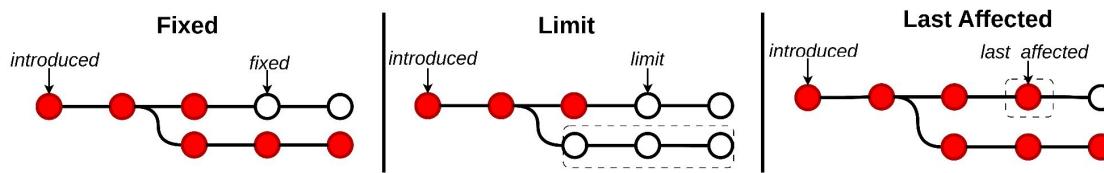
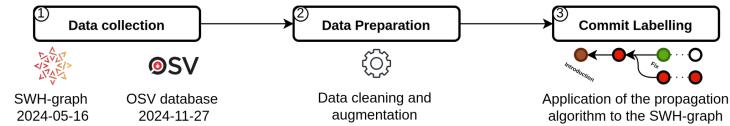
# Experimental Protocol



# Data preparation



# Labelling



Algorithm that label the SWH graph based on OSV event semantic

# Results

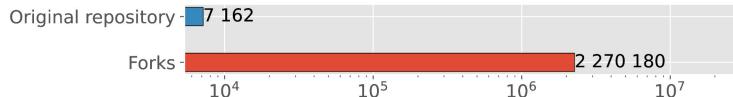


Fig 1. Number of forks having at least one vulnerable commit.

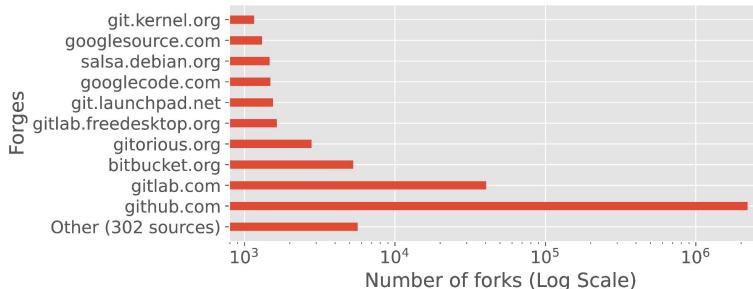
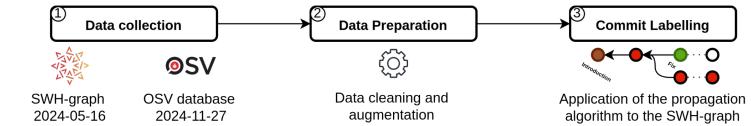
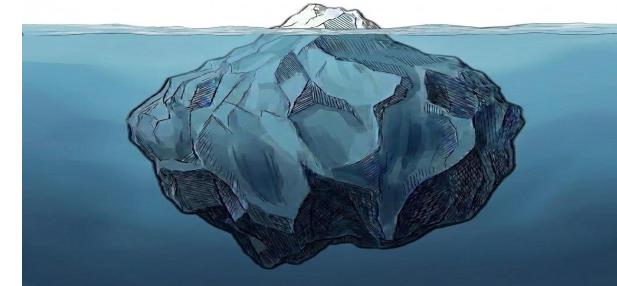


Fig 2. Number of forks having at least one vulnerable commit by forge.



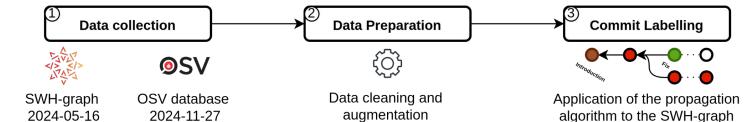
**7 162 OSV referenced repos**



**2.2 M** associated forks with at least a vulnerable commit

From more than **302** forges

# Results



**72 M** commits presumed  
vulnerable on **referenced**  
**repos**



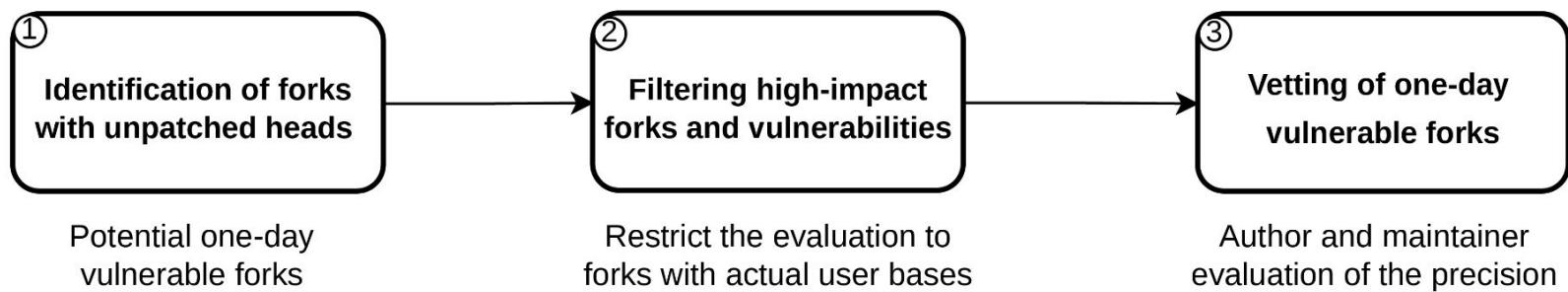
**86 M** commits  
presumed vulnerable on  
**forks**

## Research question

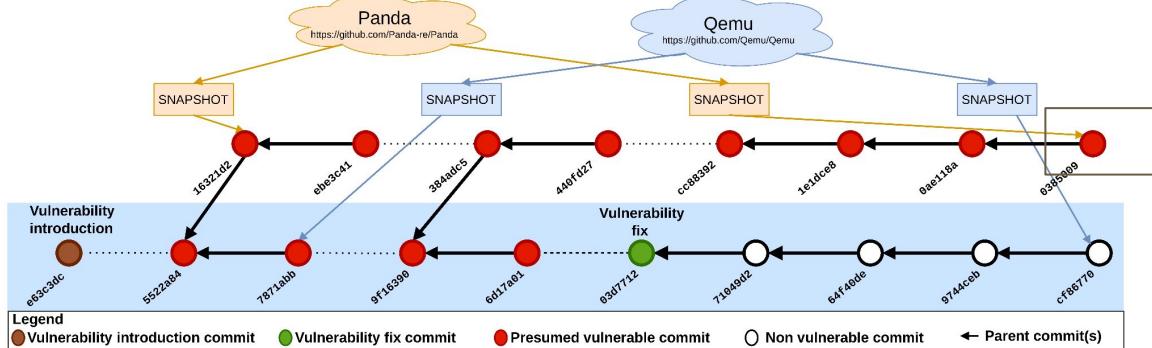
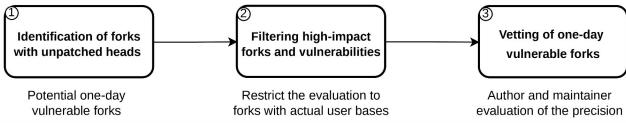
RQ1 - Is it feasible to propagate the information about which commits introduce and fix known vulnerabilities to the global commit graph of public code?

**RQ2: Can we leverage the proposed global history analysis approach to detect at scale one-day vulnerabilities in real-world forked OSS?**

# Experimental protocol

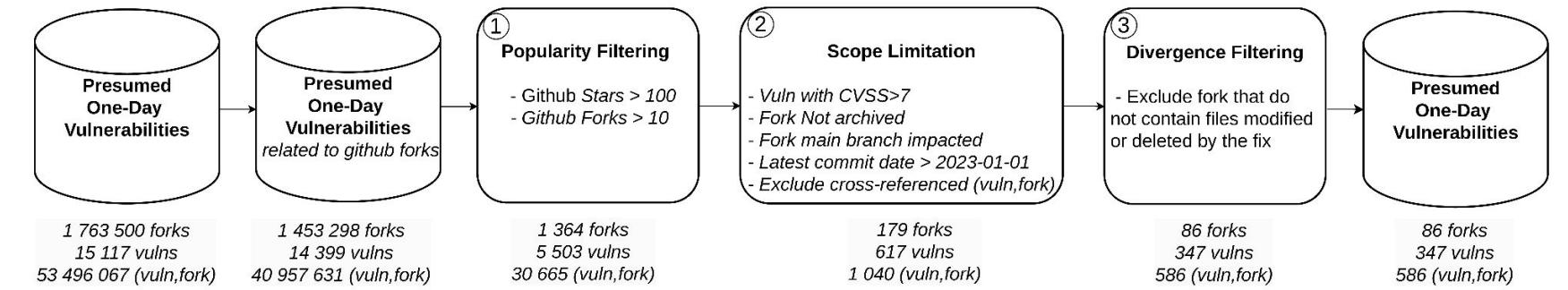
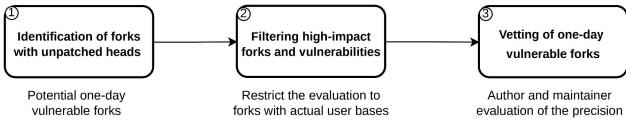


# 1- Identifying unpatched forks



**1.7 M forks with an unpatched head among the 2.2 M**

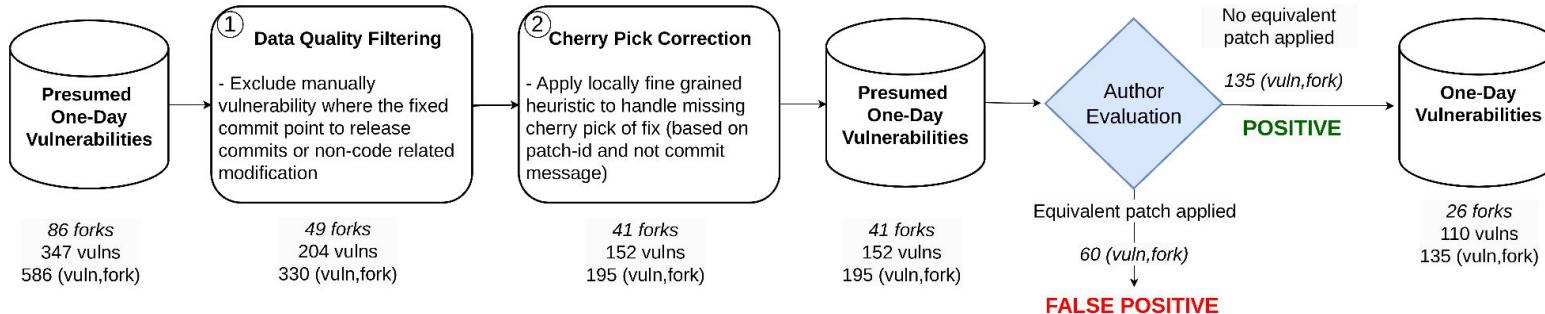
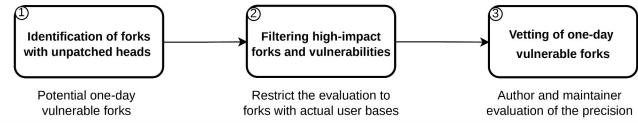
## 2- Filtering high-impact forks and vulnerabilities



**86 forks with 586 (vuln,fork) presumed one-day vulnerabilities**

## 3- Vetting of one-day vulnerable forks

### *Author evaluation*

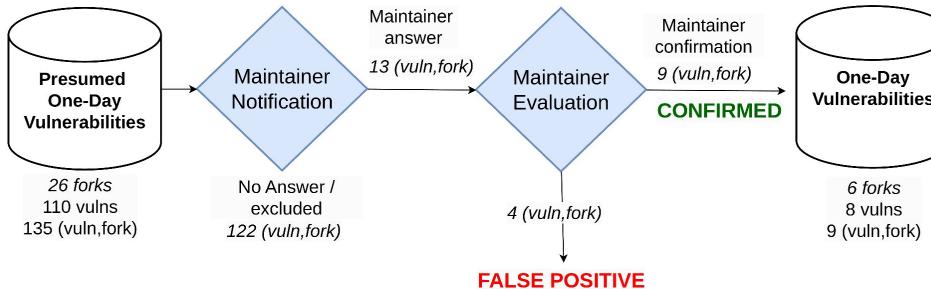
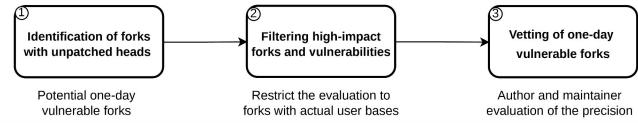


Detecting equivalent patch that could have been applied by fork maintainer

**135** positive and **60** false-positive one-day vulnerability (fork,vulnerability)  
**69 %** of precision

## 3- Vetting of one-day vulnerable forks

### Maintainer evaluation



Maintainer response = the only ground truth

**9** positive and **13** false-positive one-day vulnerability (fork,vulnerability)  
**69 %** of precision

## Confirmed cases



### Panda.re

Fork of QEMU System emulator and virtualizer  
CVE-2019-13164



### sonyxperiadev/kernel

Fork of linux kernel for xperia dev community  
CVE-2021-45485  
CVE-2021-4154



### go-nv/goenv

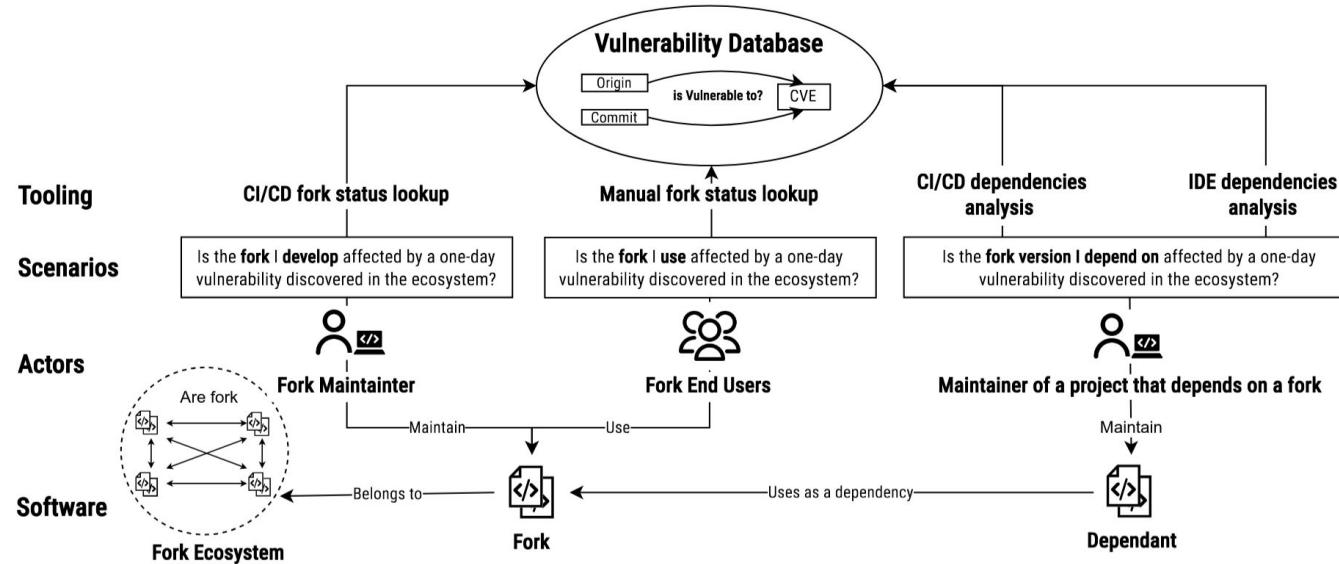
Fork of pyenv for go  
CVE-2022-35861



### bitcoin-sv/bitcoin-sv

Fork of bitcoin  
CVE-2021-37492

# Integration in software development processes



# Prototype : Enabling fork status lookup



Database lookup  
didyouforkgetit.dev

**Did you forkget it?**

Global history analysis to detect one-day vulnerabilities in open source forks

Search over more than **2.2M forks** and **158M commits**

<https://github.com/panda-re/panda> [Repository URL](#)

**Search**

**Filter Results**

Showing 2 of 546 results

CVE Identifier	Branch Name
CVE-2019-13164	Filter by branch...
	<a href="#">Clear Filters</a>

Severity Level

- Critical
- High
- Medium
- Low
- None

Show branches that do not contain refs/heads

**Found 1 distinct vulnerability**

Across 2 branches

# refs/heads/dev 1 vuln

**CVE-2019-13164 >**

**HIGH 7.8**

Revision ID: [swih:1:rev:cc@e43e0347918d74c1dfef9e3eb360432ec843](#)

Origin: <https://github.com/panda-re/panda>

Vulnerable head  
commit identifier  
(SWHID)

**CVE-2019-13164**

osv-output/CVE-2019-13164.json

**Summary**

qemu-bridge-helper.c in QEMU 3.1 and 4.0.0 does not ensure that a network interface name (obtained from bridge.conf or a --br=bridge option) is limited to the IFNAMSIZ size, which can lead to an ACL bypass.

**Details**

qemu-bridge-helper.c in QEMU 3.1 and 4.0.0 does not ensure that a network interface name (obtained from bridge.conf or a --br=bridge option) is limited to the IFNAMSIZ size, which can lead to an ACL bypass.

**Severity**

CVSS\_V3: **7.80** **High**

[Show CVSS Vector](#)

**Affected Packages**

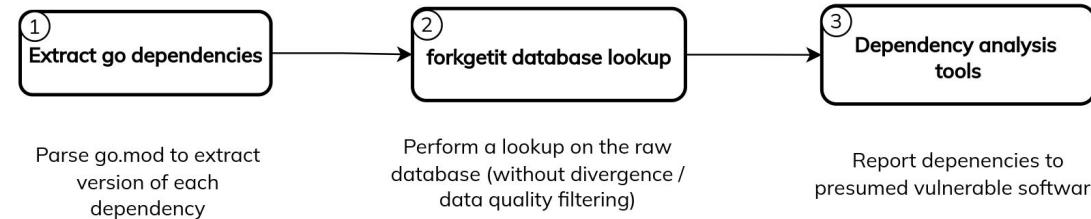
Version Ranges:

Type: GIT  
Repo: <https://github.com/qemu/qemu>  
Introduced: 0  
Fixed: [03d7712b4bcd47bfe0fe14ba2fffa87e111fa086](#)

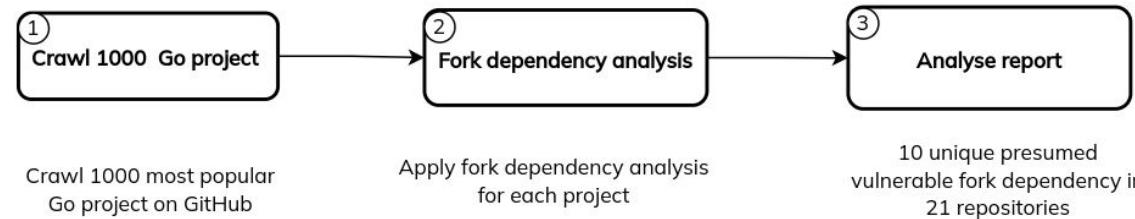
Associated fixed  
commit

# Prototype : Go dependency analysis

## Approach



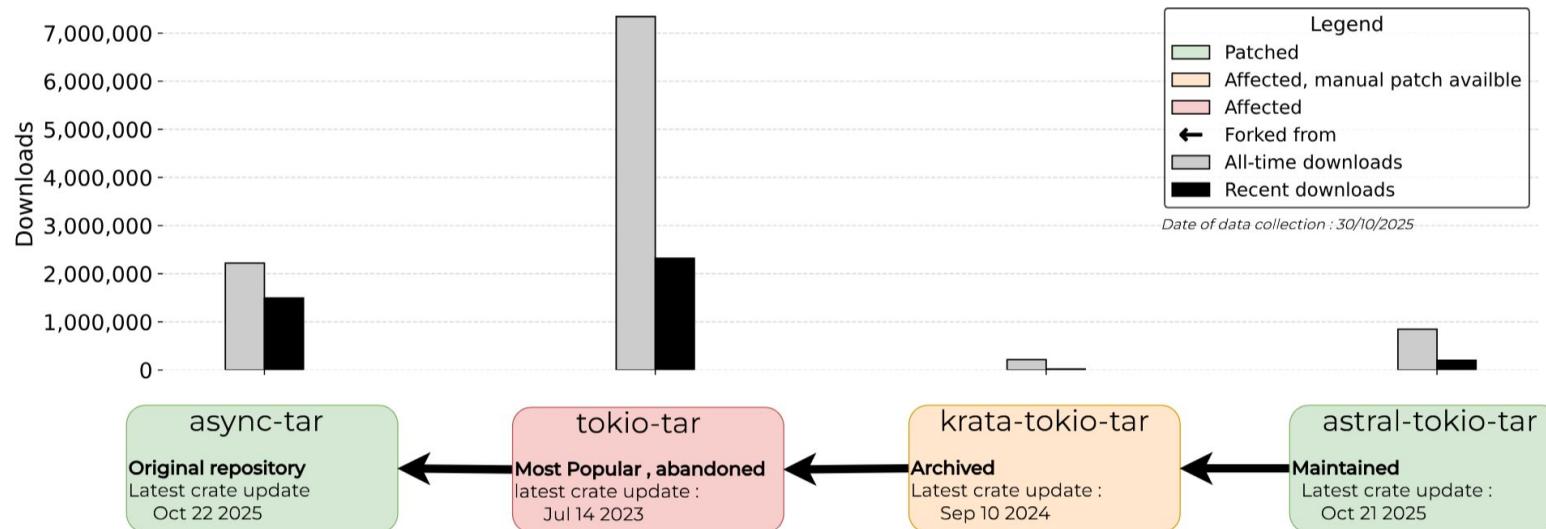
## Case study



**21 repositories** reported as dependent to presumed vulnerable forks

# Responsible disclosure

## The case of CVE-2025-62518 called “Tarmagedon”



# Conclusion

RQ1 - Labelling global commit graph with vulnerability information

- Global history analysis based on SWH
- **2.2 M forks** with presumed vulnerable commit

RQ2 - Detecting one-day vulnerabilities in real-world forked OSS

- Author evaluation : **135** positive one-day with 69 % of precision
- Maintainer evaluation : **9 confirmed positive** over 13 response

Perspective : Integration in software development processes



Thanks !

