

On the Long-Term Reproducibility of Vulnerable Environments

Olivier Levillain

Nicolas Dejon

Clément Parssegny

Sylvie Laniepce



ISC 2026, Rennes (France)
Wednesday, October 28th 2026

Motivation

The Need for Reliable Vulnerable Reproduction

Software Vulnerabilities Reproduction Applications

- ▶ Study and better understand actual flaws
- ▶ Evaluate security tools
- ▶ Teach students how to find and patch security issues
- ▶ Mount training exercises

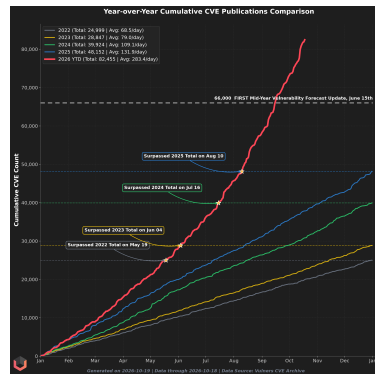
A Mountain of CVEs

Many CVEs...

A Mountain of CVEs

Many CVEs...

- ... at an increasing rate



Source: https://alukashenkov.github.io/vulnpocalypse_2026/

Source: https://alukashenkov.github.io/vulnpocalypse_2026/

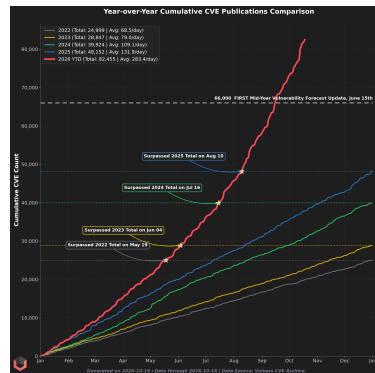
A Mountain of CVEs

Many CVEs...

- ... at an increasing rate

Many questions...

- How many can we reproduce?
- How does the reproducibility evolve in time?
- Are there reproduction success/failure factors?
- How can operational security teams anticipate or react?



Source: https://alukashenkov.github.io/vulnpocalypse_2026/

Source: https://alukashenkov.github.io/vulnpocalypse_2026/

Design Choices

Focus of this Work and Technological Choices

Our focus

- ▶ Vulnerabilities affecting Linux systems
- ▶ Services that can be deployed in the cloud
- ▶ Surrounding tools
- ▶ Exclusions: client-side software, kernel and low-level tools

Focus of this Work and Technological Choices

Our focus

- ▶ Vulnerabilities affecting Linux systems
- ▶ Services that can be deployed in the cloud
- ▶ Surrounding tools
- ▶ Exclusions: client-side software, kernel and low-level tools

Choices

- ▶ A popular Linux distribution, Debian
 - ▶ DSAs (Debian Security Advisories)
 - ▶ Snapshot archives

Focus of this Work and Technological Choices

Our focus

- ▶ Vulnerabilities affecting Linux systems
- ▶ Services that can be deployed in the cloud
- ▶ Surrounding tools
- ▶ Exclusions: client-side software, kernel and low-level tools

Choices

- ▶ A popular Linux distribution, Debian
 - ▶ DSAs (Debian Security Advisories)
 - ▶ Snapshot archives
- ▶ A container solution to package our work, Docker

Only Public Information Sources

We only rely on public databases for the vulnerability metadata

- ▶ The NVD for the enriched CVE database
- ▶ Debian Security Advisories (DSAs) and associated metadata

We reuse existing exploits

- ▶ Exploit-DB (<https://www.exploit-db.com/>)
- ▶ Files attached to relevant tickets in the bug trackers
- ▶ Blogs and GitHub Repositories

Methodology

Step 1: Data Collection

- ▶ Starting point: a CVE id
- ▶ Clear link between CVEs and DSAs
- ▶ The Debian Security Tracker identifies the vulnerable component (package and version)
- ▶ We also get a timestamp to select a base image

Step 1: Data Collection

- ▶ Starting point: a CVE id
- ▶ Clear link between CVEs and DSAs
- ▶ The Debian Security Tracker identifies the vulnerable component (package and version)
- ▶ We also get a timestamp to select a base image

Pain points

- ▶ Choosing the right release is sometimes tricky
- ▶ Selecting the relevant *binary* package(s) to install from the source package

Step 1: Data Collection

- ▶ Starting point: a CVE id
- ▶ Clear link between CVEs and DSAs
- ▶ The Debian Security Tracker identifies the vulnerable component (package and version)
- ▶ We also get a timestamp to select a base image

Pain points

- ▶ Choosing the right release is sometimes tricky
- ▶ Selecting the relevant *binary* package(s) to install from the source package

Example: CVE-2026-24061

- ▶ An access control bypass in `telnetd`
- ▶ DSA-6106
- ▶ Source package: `inetutils`
- ▶ Version: `2:2.4-2`, `2:2.6-3`
- ▶ Timestamp: December 2025
- ▶ Release: `bookworm`, `trixie`
- ▶ Package: `inetutils-telnetd`

Step 1: Data Collection

- ▶ Starting point: a CVE id
- ▶ Clear link between CVEs and DSAs
- ▶ The Debian Security Tracker identifies the vulnerable component (package and version)
- ▶ We also get a timestamp to select a base image

Pain points

- ▶ Choosing the right release is sometimes tricky
- ▶ Selecting the relevant *binary* package(s) to install from the source package

Example: CVE-2026-24061

- ▶ An access control bypass in `telnetd`
- ▶ DSA-6106
- ▶ Source package: `inetutils`
- ▶ Version: 2:2.4-2, 2:2.6-3
- ▶ Timestamp: December 2025
- ▶ Release: `bookworm`, `trixie`
- ▶ Package: `inetutils-telnetd`

Step 2: Environment Creation and Setup (1/2)

Environment Creation and Setup

- ▶ Start from a base image (usually from the past)
- ▶ Configure the package system to fetch old packages
- ▶ Install the vulnerable version

Step 2: Environment Creation and Setup (1/2)

Environment Creation and Setup

- ▶ Start from a base image (usually from the past)
- ▶ Configure the package system to fetch old packages
- ▶ Install the vulnerable version
- ▶ Configure the target
- ▶ If needed, expose the service

Step 2: Environment Creation and Setup (2/2)

Example: CVE-2026-24061

```
FROM debian:trixie-20251229-slim
```

```
RUN rm /etc/apt/sources.list.d/debian.sources
```

```
RUN echo 'deb http://snapshot.debian.org/[TIMESTAMP]/' stable main \  
    >> /etc/apt/sources.list.d/snapshot.list
```

```
RUN apt update
```

```
RUN apt install -y inetutils-telnetd
```

```
RUN echo 'telnet stream tcp nowait root [telnetd]' > /etc/inetd.conf
```

```
EXPOSE 23
```

```
CMD inetutils-telnetd -d
```

Step 3: Exploit Search and Reproduction Validation

Exploit Search

- ▶ Look for PoCs
 - ▶ following NVD links
 - ▶ on Exploit-DB or on CVE-named GitHub repositories
 - ▶ on the bug tracker of the affected project
 - ▶ on interesting mailing lists
- ▶ Review and adapt the PoC when necessary

Step 3: Exploit Search and Reproduction Validation

Exploit Search

- ▶ Look for PoCs
 - ▶ following NVD links
 - ▶ on Exploit-DB or on CVE-named GitHub repositories
 - ▶ on the bug tracker of the affected project
 - ▶ on interesting mailing lists
- ▶ Review and adapt the PoC when necessary

Reproduction Validation

- ▶ If the expected effect is reproduced, great
- ▶ Sometimes, only partial validation
- ▶ In the absence of a proper exploitation, we cannot guarantee that the environment is actually vulnerable

Demo for CVE-2026-24061

Output of our Study

For each CVE, we record

- ▶ **CVE enhancement:** detailed description, impact and publication dates, information quality, patch timeline

Output of our Study

For each CVE, we record

- ▶ **CVE enhancement:** detailed description, impact and publication dates, information quality, patch timeline
- ▶ **Vulnerable Environment Specification:** Debian release, vulnerable package/version, installation method and environment complexity

Output of our Study

For each CVE, we record

- ▶ **CVE enhancement:** detailed description, impact and publication dates, information quality, patch timeline
- ▶ **Vulnerable Environment Specification:** Debian release, vulnerable package/version, installation method and environment complexity
- ▶ **Exploitation Specification:** exploit process, exploit source, public release date, exploit configuration complexity

Output of our Study

For each CVE, we record

- ▶ **CVE enhancement:** detailed description, impact and publication dates, information quality, patch timeline
- ▶ **Vulnerable Environment Specification:** Debian release, vulnerable package/version, installation method and environment complexity
- ▶ **Exploitation Specification:** exploit process, exploit source, public release date, exploit configuration complexity
- ▶ **Reproduction Status:** Success/Partial Success/Failure, reason of reproduction failure if any

Output of our Study

For each CVE, we record

- ▶ **CVE enhancement:** detailed description, impact and publication dates, information quality, patch timeline
- ▶ **Vulnerable Environment Specification:** Debian release, vulnerable package/version, installation method and environment complexity
- ▶ **Exploitation Specification:** exploit process, exploit source, public release date, exploit configuration complexity
- ▶ **Reproduction Status:** Success/Partial Success/Failure, reason of reproduction failure if any

Manual study by two analysts who cross-checked the artifacts

Results

Research Questions

- ▶ RQ1: To what extent is it possible to systematically reproduce a vulnerable environment with publicly available information?
- ▶ RQ2: What are the primary challenges that hinder the reproduction of vulnerable environments?
- ▶ RQ3: What critical gaps exist between public vulnerability intelligence and the operational realities of security teams?

Corpus Selection: 4 Complementary Corporuses

Corpus	Period	Selection	#CVEs
10Y-50	2014–2023	Curated for variety in severity, package, and date; includes major vulnerabilities (Heartbleed, ImageTragick, Baron Samedit, PwnKit)	50
Recent-50	2020–2023	Same criteria as 10Y-50	50
Pop-2023	2023	CVEs in Debian's top 1,000 packages by popularity	74
Cage4Deno ¹	2016-2022	External research corpus; CVEs affecting various programs (e.g. FFmpeg or ImageMagick) to test a kernel-level sandbox, no provided vulnerable environments	10

142 unique *relevant* CVEs, corpuses overlap

relevant=Debian-based, userland vulnerability, container-oriented

¹Abbadini et al., *Cage4Deno: A Fine-Grained Sandbox for Deno Subprocesses*, ASIA CCS 2023

RQ1: To what extent is it possible to systematically reproduce a vulnerable environment with publicly available information?

Corpus	Relevant CVEs	Reproduction Success (incl. partial)
10Y-50	50	44% (50%)
Recent-50	50	44% (52%)
Pop-2023	74	34% (57%)
Cage4Deno (2016-2022)	10	80% (100%)
Total	142	41% (57%)

58/142 fully reproduced (41%)
81/142 fully or partially reproduced (57%)

CVE-2020-7247: LPE or RCE via OpenSMTPD shell injection

DSA-4611-1

Name	DSA-4611-1
Description	opensmtpd - security update
Source	Debian
References	CVE-2020-7247

The information below is based on the following data on fixed versions.

Package	Type	Release	Fixed Version
opensmtpd	source	stretch	6.0.2p1-2+deb9u2
opensmtpd	source	buster	6.0.3p1-5+deb10u3

Figure: DSA: opensmtpd v6.0 on Debian 9-10.

NATIONAL VULNERABILITY DATABASE

NIST
NATIONAL VULNERABILITY
DATABASE
NVD

VULNERABILITIES

🌸 CVE-2020-7247 Detail

Description

smtp_mailaddr in smtp_session.c in OpenSMTPD 6.6, as used in OpenBSD 6.6 and other products, allows remote attackers to execute arbitrary commands as root via a crafted SMTP session, as demonstrated by shell metacharacters in a MAIL FROM field. This affects the "uncommented" default configuration. The issue exists because of an incorrect return value upon failure of input validation.

QUICK INFO

CVE Dictionary Entry:

[CVE-2020-7247](#)

NVD Published Date:

Jan 29, 2020

Figure: NVD: opensmtpd v6.6 on Debian 11.

CVE-2020-7247: LPE or RCE via OpenSMTPD shell injection

DSA-4611-1

Name	DSA-4611-1
Description	opensmtpd - security update
Source	Debian
References	CVE-2020-7247

The information below is based on the following data on fixed versions.

Package	Type	Release	Fixed Version
opensmtpd	source	stretch	6.0.2p1-2+deb9u2
opensmtpd	source	buster	6.0.3p1-5+deb10u3

Figure: DSA: opensmtpd v6.0 on Debian 9-10.

NATIONAL VULNERABILITY DATABASE

NIST
NATIONAL VULNERABILITY
DATABASE
NVD

VULNERABILITIES

🌸 CVE-2020-7247 Detail

Description

smtp_mailaddr in smtp_session.c in OpenSMTPD 6.6, as used in OpenBSD 6.6 and other products, allows remote attackers to execute arbitrary commands as root via a crafted SMTP session, as demonstrated by shell metacharacters in a MAIL FROM field. This affects the "uncommented" default configuration. The issue exists because of an incorrect return value upon failure of input validation.

QUICK INFO

CVE Dictionary Entry:

[CVE-2020-7247](#)

NVD Published Date:

Jan 29, 2020

Figure: NVD: opensmtpd v6.6 on Debian 11.

Metadata are not a proof of exploitability
Reproduction success correlates with information quality

RQ2: What are the primary challenges that hinder the reproduction of vulnerable environments?

Table: Non Reproduction Reasons.

Reason for Failure	Not Reproduced	Partially Reproduced
Exploit Not Working	9	0
Incomplete Exploit	0	6
Memory Sanitizer Only	0	15
No Exploit Available	46	1
Non Functional Container	2	0
Missing Information	4	1
Total	61 $\approx 43\%$	23 $\approx 16\%$

Challenges:

- Exploitation: Public availability of exploits, vulnerability exposure (crash, memory overflows..) through ASan (Address Sanitizer) compilation only, incomplete exploit
- Construction: lack of adequate base image, failed configuration

RQ3: What critical gaps exist between public vulnerability intelligence and the operational realities of security teams?

1. Official Patch → NVD Disclosure

- ▶ delay: **33%** disclosed 30+ days *after* the official patch
- ▶ early: **7%** disclosed *before* an official patch existed

RQ3: What critical gaps exist between public vulnerability intelligence and the operational realities of security teams?

1. Official Patch → NVD Disclosure

- ▶ delay: **33%** disclosed 30+ days *after* the official patch
- ▶ early: **7%** disclosed *before* an official patch existed

2. Official Patch → Debian Fixed Package

- ▶ **39%** lagged beyond 30 days
- ▶ **CVSS-patch speed correlation:** Critical: **3 days** High: **12 days** Medium: **34 days**

RQ3: What critical gaps exist between public vulnerability intelligence and the operational realities of security teams?

1. Official Patch → NVD Disclosure

- ▶ delay: **33%** disclosed 30+ days *after* the official patch
- ▶ early: **7%** disclosed *before* an official patch existed

2. Official Patch → Debian Fixed Package

- ▶ **39%** lagged beyond 30 days
- ▶ **CVSS-patch speed correlation:** Critical: **3 days** High: **12 days** Medium: **34 days**

3. NVD Publication → Functional PoC

- ▶ **42%** of PoCs available *before or the same day* as NVD publication
- ▶ **No correlation CVSS-exploit availability**

Container Scanner Reliability (Trivy/Grype)

Corpus	Relevant CVEs	Fully or Partially Reproduced	Detected by Trivy/Grype
10Y-50	50	22	16 (73%)
Recent-50	50	26	18 (69%)
Pop-2023	74	42	32 (76%)
Cage4Deno	10	10	9 (90%)
Total	142	81	62 (76%)

- ▶ 24% of validated vulnerable environments **NOT** detected by any scanner
- ▶ Scanners disagreed on 'CVE-2014-8129' (flagged by Grype but not Trivy)

Discussions

Revisiting Our Choices

- ▶ Standard containers prevent us from studying kernel vulnerabilities
 - ▶ Switch to a VM-based system?
 - ▶ Or use a different container runtime?

Revisiting Our Choices

- ▶ Standard containers prevent us from studying kernel vulnerabilities
 - ▶ Switch to a VM-based system?
 - ▶ Or use a different container runtime?
- ▶ Linux is not Debian
 - ▶ Could we extend our work to other distributions?
 - ▶ What about non-Linux CVEs?

Revisiting Our Choices

- ▶ Standard containers prevent us from studying kernel vulnerabilities
 - ▶ Switch to a VM-based system?
 - ▶ Or use a different container runtime?
- ▶ Linux is not Debian
 - ▶ Could we extend our work to other distributions?
 - ▶ What about non-Linux CVEs?
- ▶ How to find the missing exploits?
 - ▶ Since we started our journey, generating exploits seems easier...
 - ▶ But it raises *interesting* ethical considerations

Automation

- ▶ Our methodology is partly automated thanks to DECRET²
 - ▶ it was presented in Rennes some time ago (SSTIC 2023)³
- ▶ What about LLMs?

²Source code: <https://github.com/Orange-OpenSource/decret>

³https://www.sstic.org/2023/presentation/reproduction_automatisee_de_vulnerabilites_logicielles_dans_un_environnement_conteneurise_decret/

Conclusion and Perspectives

- ▶ Vulnerability Open Knowledge (VOK) database:
 - ▶ 142 unique CVEs,
 - ▶ 10-year period (2014-2023),
 - ▶ 41% fully reproduced, 57% fully or partially,
 - ▶ documented reproduction (Dockerfile, exploit source, reproduction description),
 - ▶ and other metadata...
- ▶ Ground truth for operational blind spots (24% detection failure of container scanners), CTF basis, vulnerability research and training
- ▶ What about long-term reproducibility and the volatility of currently available metadata?

You are welcome to contribute!

Questions ?

Thank you for your attention! Merci !



The VOK (Vulnerability Open Knowledge) database

<https://github.com/Orange-OpenSource/vulnerability-open-knowledge>



The DECRET vulnerable environment reproduction tool

<https://github.com/Orange-OpenSource/decret>



Backup slides

Exploit Rate by Package Category and Exploit Source Distribution

Package Category	Typical Packages	Exploit Rate
Parsing / Media / Directory	zlib, librsvg, ghostscript	75%–100%
Top 10 Most Affected	Popular Linux packages	41%
Network Services / DNS	bind9, ntp	12%
Database Engines	postgresql	0%

GitHub Repositories (45%)

GitLab / Vendor (12%)

Technical Blogs (10%)

Security Advisories (9%)

Bug Trackers (7%)

Exploit-DB (7%)

Academic / Other (10%)