



Workshop around Provenance, Telemetry and Security Applications

NICE: A Framework for Declarative and Machine-Checkable Vulnerability Reproduction

Minh-Luân NGUYEN

Authors: Olivier Levillain, Julien Malka, Stefano Zacchiroli, Théo Zimmermann



Contents

1 Research problem

Why vulnerability reproduction matters and what information is usually missing.

2 Technical basis

Functional Package Management, Nix, and NixOS.

3 NICE Framework

What is NICE and how does it works?

4 Evaluation

Performance, discussion and conclusion.

“It works on my machine.”

“Great. Can I borrow your machine from 2014?”

A PoC alone does not preserve dependencies, services, packages or exploit conditions.



Why vulnerability reproduction matters

95.1% reports in Mu et al. missed at least one required field [1]

54.9% of 368 memory-error vulnerabilities were reproduced, despite 3,600+ analyst-hours. [1]

➔ Vulnerability reproduction is often fragile, time-consuming and often not easy.

Understanding the Reproducibility of Crowd-reported Security Vulnerabilities

^{†‡}Dongliang Mu*, [‡]Alejandro Cuevas, [§]Limin Yang, [§]Hang Hu
[‡]Xinyu Xing, [†]Bing Mao, [§]Gang Wang

[†]National Key Laboratory for Novel Software Technology, Nanjing University, China
[‡]College of Information Sciences and Technology, The Pennsylvania State University, USA

[§]Department of Computer Science, Virginia Tech, USA
dzm77@ist.psu.edu, aledancuevas@psu.edu, {liminyang, hanghu}@vt.edu,
xxing@ist.psu.edu, maobing@nju.edu.cn, gangwang@vt.edu

Abstract

Today's software systems are increasingly relying on the "power of the crowd" to identify new security vulnerabilities. And yet, it is not well understood how *reproducible* the crowd-reported vulnerabilities are. In this paper, we perform the first empirical analysis on a wide range of real-world security vulnerabilities (368 in total) with the goal of quantifying their reproducibility. Following a carefully controlled workflow, we organize a focused group of security analysts to carry out reproduction experiments. With 3600 man-hours spent, we obtain quantitative evidence on the prevalence of missing information in vulnerability reports and the low reproducibility of the vulnerabilities. We find that relying on a single vulnerability report from a popular security forum is generally difficult to succeed due to the incomplete information. By widely crowdsourcing the information gathering, security analysts could increase the reproduction success rate, but still face key challenges to troubleshoot the non-reproducible cases. To further explore solutions, we surveyed hackers, researchers, and engineers who have extensive domain expertise in software security (N=43). Going beyond Internet-scale crowdsourcing, we find that, security professionals heavily rely on manual debugging and speculative guessing to infer the missed information. Our result suggests that there is not only a necessity to overhaul the way a security forum collects vulnerability reports, but also a need for automated mechanisms to collect information commonly missing in a report.

*Work was done while visiting The Pennsylvania State University.

1 Introduction

Security vulnerabilities in software systems are posing a serious threat to users, organizations and even nations. In 2017, unpatched vulnerabilities allowed the WannaCry ransomware cryptoworm to shutdown more than 300,000 computers around the globe [29]. Around the same time, another vulnerability in Equifax's Apache servers led to a devastating data breach that exposed half of the American population's Social Security Numbers [48].

Identifying security vulnerabilities has been increasingly challenging. Due to the high complexity of modern software, it is no longer feasible for in-house teams to identify all possible vulnerabilities before a software release. Consequently, an increasing number of software vendors have begun to rely on "the power of the crowd" for vulnerability identification. Today, anyone on the Internet (e.g., white hat hackers, security analysts, and even regular software users) can identify and report a vulnerability. Companies such as Google and Microsoft are spending millions of dollars on their "bug bounty" programs to reward vulnerability reporters [38, 54, 41]. To further raise community awareness, the reporter may obtain a Common Vulnerabilities and Exposures (CVE) ID, and archive the entry in various online vulnerability databases. As of December 2017, the CVE website has archived more than 95,000 security vulnerabilities.

Despite the large number of crowd-reported vulnerabilities, there is still a major gap between vulnerability reporting and vulnerability patching. Recent measurements show that it takes a long time, sometimes multiple years, for a vulnerability to be patched after the initial report [43]. In addition to the lack of awareness, anecdotal evidence also asserts the poor quality of crowd-sourced reports. For example, a Facebook user once identified a vulnerability that allowed attackers to post

Functional Package Manager & Nix



Install old packages →
dependency conflict → fix
another library → repeat.



Build outputs derive from
declared inputs and
dependency graphs.

Functional Package Manager & Nix

Nixpkgs

Nixpkgs is currently the largest general-purpose collection of open-source software.

NixOS

Specify the whole system: packages, services, users, network and files.

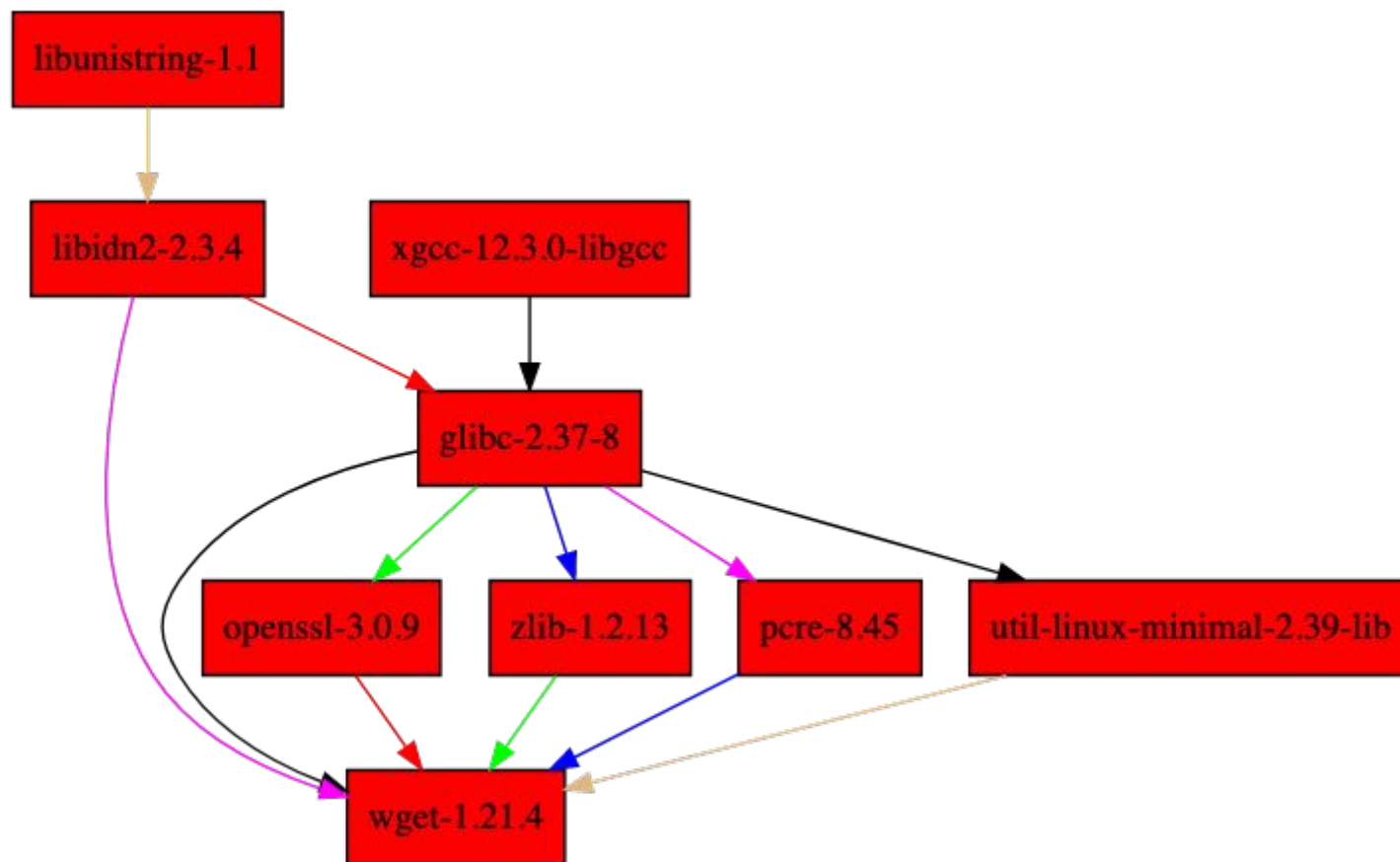
FPM

A functional package manager is a software deployment tool that treats the package build and installation process as a pure function in programming.



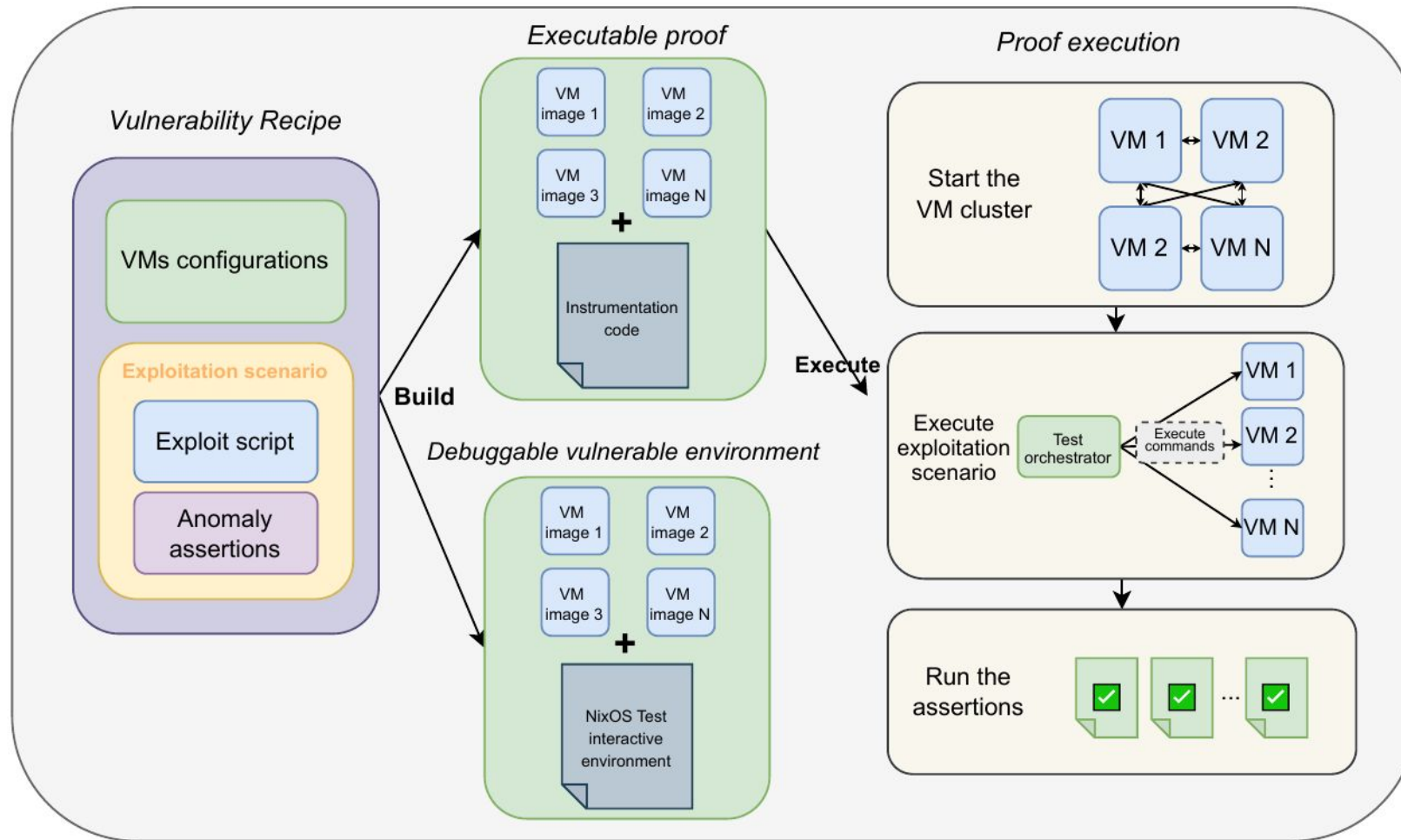
```
/nix/store/0warqn2vkvwy4vwdrxyryaccqgqi3y5p-hello-2.12.1
├── /nix/store/cg9s562sa33k78m63njfn1rw47dp9z0i-glibc-2.40-66
│   ├── /nix/store/6nkqdqzpa75514lhglgnjs5k4dklw4sb-libidn2-2.3.8
│   │   └── /nix/store/yypqcvqhmv8y4zpicgxdigp3giq81gzb-libunistring-1.3
│   │       └── /nix/store/yypqcvqhmv8y4zpicgxdigp3giq81gzb-libunistring-1.3 [...]
```

Functional Package Manager & Nix



<https://skalski.dev/understanding-nix/>

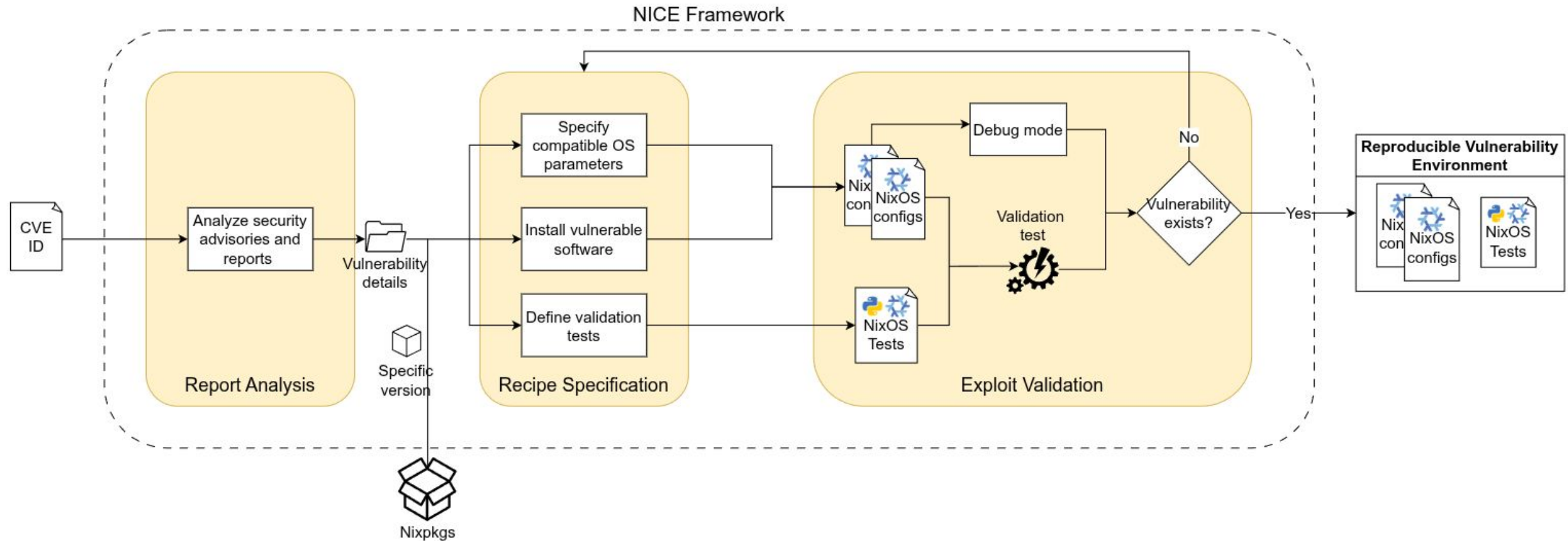
NICE: The Nix CVE reproduction framework



What becomes explicit?

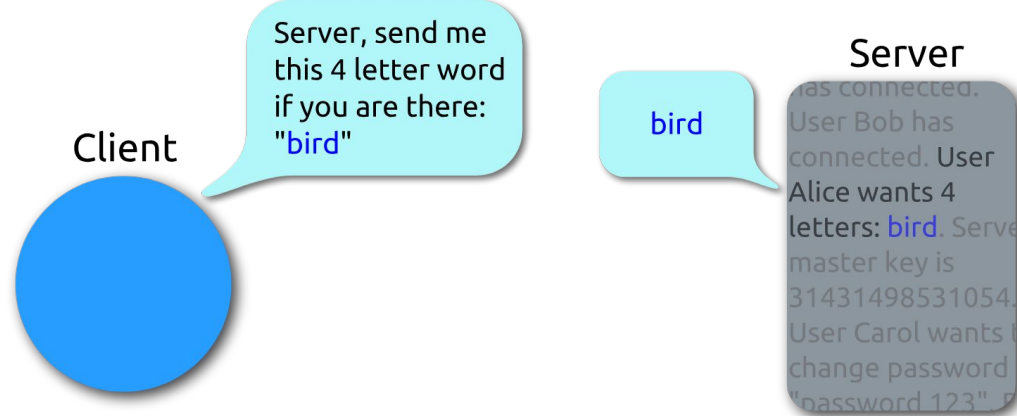
The vulnerable environment + the procedure required to demonstrate exploitability. The result is an executable experiment.

NICE: A framework for declarative and machine-checkable vulnerability reproduction

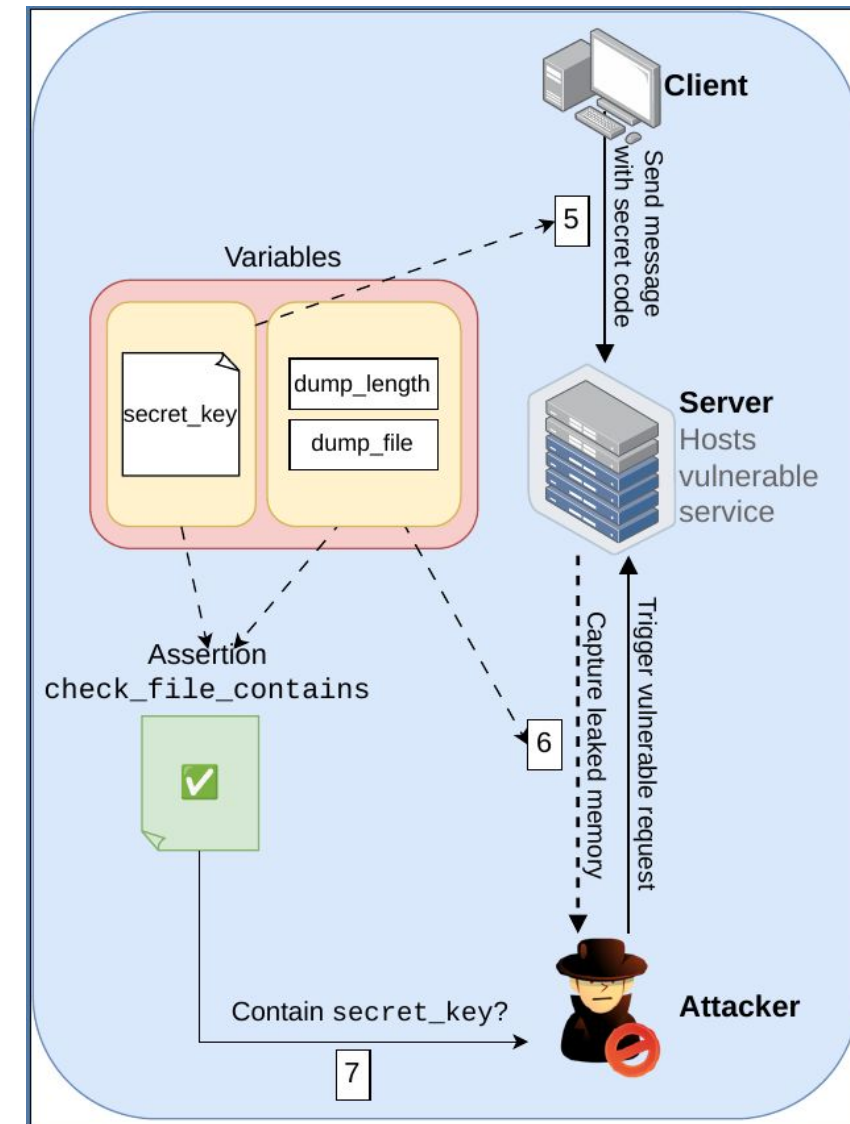
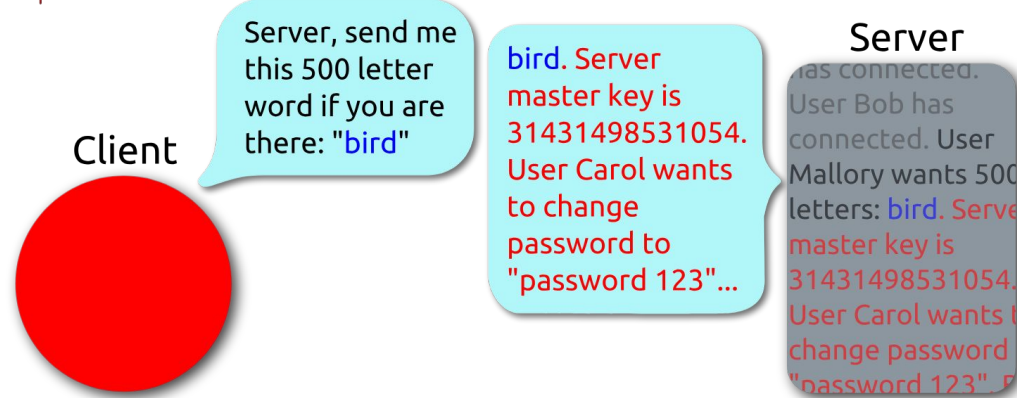


NICE: A framework for declarative and machine-checkable vulnerability reproduction

Heartbeat – Normal usage

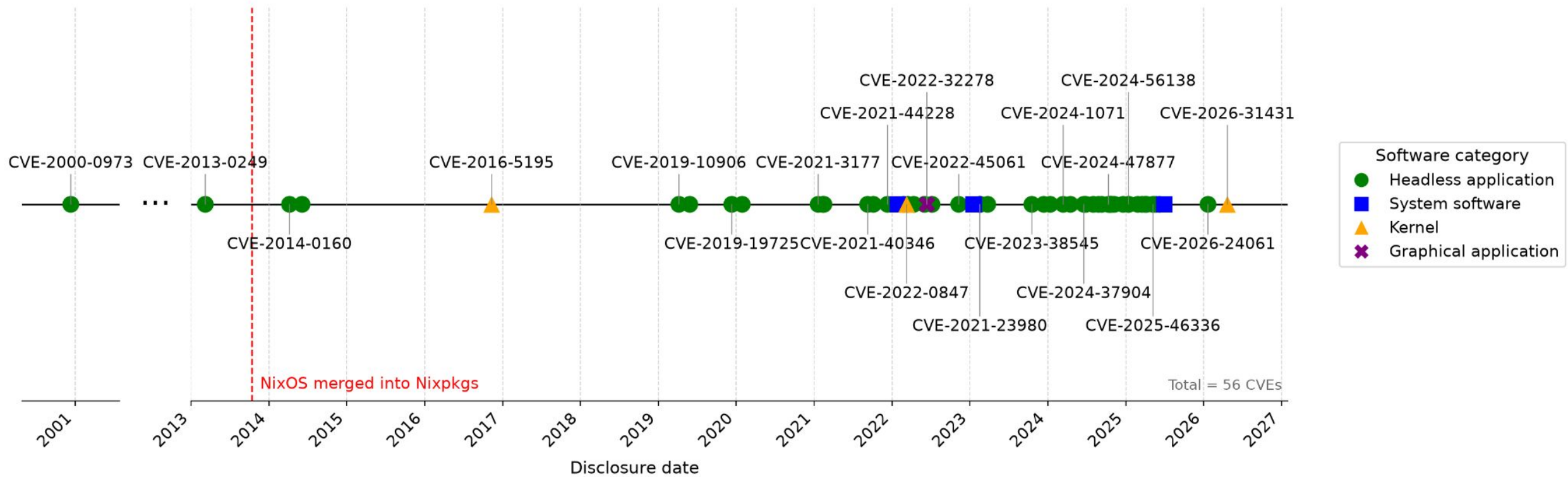


Heartbeat – Malicious usage



NICE: A framework for declarative and machine-checkable vulnerability reproduction





LLM Evaluation

54.5%

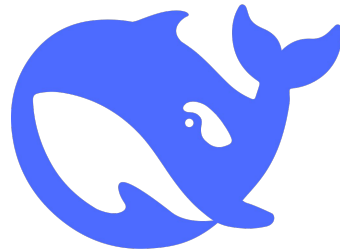
60 / 110 recipes accepted by the separate evaluator.

\$0.73

Average LLM cost per CVE attempt.

62 min

Average runtime per reproduction task.



What did we learn?

- The evaluator agent is pretty good.
- The LLM reproducer agents are very enthusiastic in writing recipes.
- Suitable for AI-assisted vulnerable environment packaging

**LLM is good but still
have limitations**

Lightweight sharing

Share a textual recipe, not a large binary VM snapshot.

Push-button validation

Rebuild and verify with minimal manual work.

Ease of review

Explicit, machine-checkable evidence of exploitation.

Reproducibility over time

The recipe survives ecosystem drift over time.

Limitations

Exploit availability

In practice, PoCs are not always publicly available, and when they are, they often lack information

Scope boundaries

Non-Linux systems are out of scope, so does the physical vulnerabilities



EDUCATION

Hands-on teaching labs and long-lived cyber ranges.



RESEARCH

Stable vulnerability benchmarks for security tools.



TRIAGE

Executable reports, audit-friendly evidence and patch validation.





Conclusion

- NICE Framework: A framework for Declarative and Machine-Checkable Vulnerability Reproduction
- NICE Archive: A collection of reproducible vulnerability recipe.
- Evaluation of NICE using LLM.

Thank you for your attention!



github.com/minhluannguyen/nice-archive

Reproducible reporting for curl

Motivation

Low-quality AI-assisted reports can consume maintainer time when claims do not reproduce.

Prototype

A NICE-style curl report contains an advisory, an executable recipe, and anomaly assertions.

Claim → executable evidence → maintainer review



CURL AND LIBCURL

DEATH BY A THOUSAND SLOPS

🕒 JULY 14, 2025 👤 DANIEL STENBERG 💬 58 COMMENTS

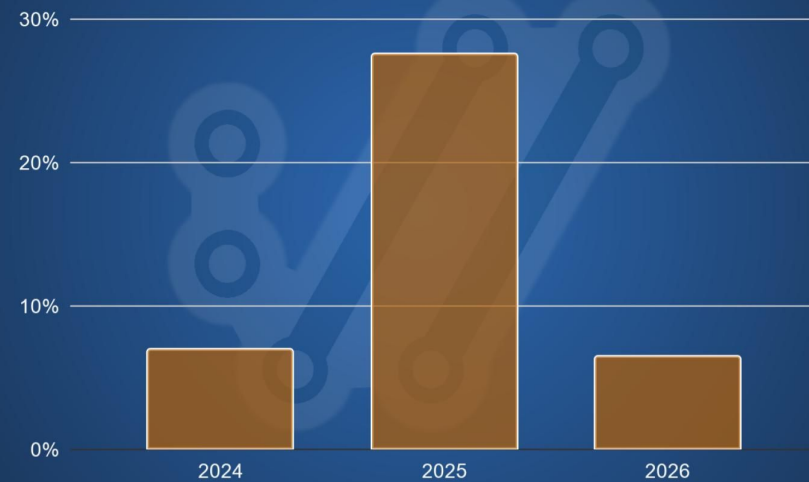
I have previously blogged about the relatively new trend of [AI slop](#) in vulnerability reports submitted to curl and how it hurts and exhausts us.

Reproducible reporting for curl

Current outputs

- nice-curl-vuln-reporter project
- Contact the maintainer of curl several time and exchange ideas.

AI slop rate



Daniel Stenberg