Can We Trust LLM Security Research? A Reproducibility Study of Large Language Model Security Papers in Tier-1 Security Conferences

Zeyuan Chen[†], Ye Leng[†], Chi Cui[†], Mengfei Liang[†], Bo Shao[†],
Xinyu Zhang[†], Yun Shen[‡], Michael Backes[†], Xinyue Shen[†§], Yang Zhang[†*]
[†]*CISPA Helmholtz Center for Information Security,* [‡]*HPE,* [§]*University of Waterloo*
{*zeyuan.chen, ye.leng, chi.cui, mengfei.liang, bo.shao, xinyu.zhang, zhang*}*@cispa.de*
mbackes1978@gmail.com, yun.shen@hpe.com, xinyue.shen@uwaterloo.ca

***Abstract*—As large language models (LLMs) are increasingly deployed in real-world systems, research on their security vulnerabilities and defenses has grown rapidly. Despite this growth, the practical reproducibility of LLM security research remains underexplored. In this paper, we present a systematic measurement study of 106 LLM security papers published at CCS, USENIX Security, IEEE S&P, and NDSS between 2023 and 2025. Among these, 84 papers release executable artifacts, which two independent experts attempt to reproduce under a standardized, time-bounded reproduction protocol. We successfully reproduce 33 artifacts, yielding an overall success rate of 39.29%. Beyond artifact availability and executability, we examine result fidelity. Among the 33 reproduced artifacts, 8 meet the primary reproduction criterion but show partial discrepancies in broader reported results. Our analysis identifies key obstacles to reproducibility, including environment drift, restricted access to datasets or model checkpoints, API changes, and excessive computational requirements. We also identify LLM-specific sources of result drift, such as evolving model behavior, deprecated closed-source model versions, prompt and decoding sensitivity, and post-publication safety updates. These factors can cause reproduced results to diverge from the results reported in the original papers even when artifacts remain executable. Within USENIX Security, AE badge participation is associated with a higher observed reproduction rate (66.67% vs. 40.43%). Based on these findings, we provide concrete recommendations to improve artifact quality, reporting practices, and long-term verifiability in LLM security research.**

1. Introduction

The rapid adoption of Large Language Models (LLMs) [17], [26], [37], [38], [40], [47], [56] has introduced new security risks. Researchers have demonstrated that attackers can circumvent safety guardrails through jailbreak prompts, hijack execution flow via prompt injection, or compromise model integrity through training-time data poisoning and backdoors [19], [25], [50], [52]. For this reason, growing numbers of security researchers focus on LLM security, aiming to improve model robustness and safety. But validating their findings faces a challenge unique to LLM-based systems: they do not stand still.

Prior work has examined reproducibility in machine learning (ML) security research [33], primarily in settings with fixed model architectures and static evaluation pipelines. However, directly reusing experimental protocols from prior ML security reproducibility studies is insufficient for LLM-based systems due to several fundamental differences. LLMs are frequently updated, especially when accessed through proprietary APIs [11], [17], their outputs depend heavily on prompts and interaction protocols [45], [49]. As such, even minor implementation changes can substantially alter evaluation outcomes, making reproducibility fragile. In addition, reproducing LLM-based attacks and defenses often incurs higher computational and financial costs [3], [48], and LLM security introduces new attack surfaces, such as jailbreaks and prompt injection, that have no direct analogues in traditional ML systems. These differences mean that reproducibility in LLM security cannot be judged only by whether an artifact is available or executable. An artifact may still run, yet produce results that no longer support the original security claim due to model/API drift, prompt sensitivity, or post-publication safety updates. We therefore treat reproducibility as a multi-stage process covering artifact availability, executability, and result fidelity.

Here, we examine trust through one specific dimension: practical reproducibility. For research findings to be actionable, independent researchers must be able to rerun experiments to validate proposed attacks and defenses. Without such verification, it becomes difficult to distinguish real vulnerabilities from artifacts of fragile experimental setups, a concern particularly acute in LLM security, where empirical results increasingly influence deployment decisions and mitigation strategies [6], [18]. Yet reproducibility remains challenging in practice. Studies often depend on external model APIs or gated datasets and checkpoints that are difficult to archive [30], [54], [55], and missing documentation or assets can render independent verification effectively impossible. These challenges raise a practical question for the community: to what extent are recent LLM security results actually reproducible? To answer this question, we conduct a systematic reproducibility study of LLM attack

* *Corresponding Author.*

and defense papers from tier-1 security conferences. We identify and analyze common barriers to reproduction, highlight practices enabling successful reproduction, and provide recommendations to improve the reproducibility of LLM security research.

Contributions. In this paper, we answer a simple yet critical question: to what extent can the empirical results reported in prior LLM security papers be reproduced today? We reproduce LLM security papers published from 2023 to 2025 from four tier-1 security conferences: ACM CCS, USENIX Security, IEEE S&P, and NDSS. Specifically, we seek to answer the following research questions. **RQ1:** What is the current state of reproducibility in LLM attack and defense research? **RQ2:** What are the barriers preventing successful reproducibility? **RQ3:** What practices enable reproducible LLM security research? To address these questions, we systematically collect 106 papers on LLM security from tier-1 security conferences published between 2023 and 2025. Among them, 84 papers release artifacts and serve as the primary target of our reproducibility analysis. We evaluate the availability and quality of four critical components required for reproducibility: method descriptions, experimental specifications, datasets, and code artifacts. For papers providing sufficient resources, we conduct reproduction attempts following standardized protocols, allocating up to 12 hours for execution. Throughout this process, we record encountered obstacles, categorize failure modes, and analyze factors associated with successful reproduction. Based on this systematic study, our contributions are as follows:

- A systematic reproducibility study of LLM security research in top-tier security conferences. We evaluate artifact availability, executability, and result fidelity across 84 LLM security papers and observe an overall reproduction success rate of 39.29%, with a clear gap between running released code and reproducing the original reported results.
- A systematic characterization of reproducibility challenges in LLM security. We uncover and classify recurring failure patterns, including general barriers shared with prior ML reproducibility studies such as unstable environments and unavailable artifacts, and barriers unique to LLM security, including API and model version changes, post-publication safety updates, prompt sensitivity, and excessive inference costs.
- Drawing on empirical evidence from reproduction outcomes, failure mode analysis, and success case studies, we propose actionable recommendations for authors, reviewers, and publication conferences.
- A descriptive comparison of reproducibility across conferences and artifact evaluation participation. We report conference-level differences descriptively and find that, within USENIX Security, AE badge participation is associated with a higher reproduction success rate (66.67% vs. 40.43%). The study involved over 3,900 person-hours of manual reproduction across 84 papers.

2. Background and Related Work

LLM Security Research. Recent work has shown that LLMs can introduce security risks beyond traditional ML systems [29], [59], including prompt injection and jailbreak bypasses [5], [25], [50], [53], training-time poisoning and backdoors [20], [52], [58], and privacy threats such as model extraction and memorization [19], [27], [48]. Compared to prior ML security settings, LLM security evaluations are highly sensitive to implementation details, including prompt, decoding parameters, and model or API versions [51], [57]. Existing defenses span inference-time safeguards, training-time mitigations, and provenance mechanisms [9], [12], [36], [44], [52], [58], [60], [61]. However, these methods are often evaluated under paper-specific threat models and evolving evaluation pipelines, making cross-study comparison and independent validation difficult without complete, executable artifacts.

Open Science and Artifact Evaluation in Security. In recent years, leading security conferences have increasingly formalized reproducibility and transparency requirements through explicit open science and artifact evaluation (AE) policies. USENIX Security requires authors to disclose artifact availability and commit to sharing them upon acceptance, subject to verification [42], [43], while CCS mandates an open science appendix enumerating code, datasets, and configuration artifacts and requires justification for any omissions starting with the 2026 submission cycle [41]. Other conferences, including IEEE S&P, DSN, and Euro S&P, have adopted similar requirements or optional disclosures as part of the submission process [21], [22], [24]. Beyond submission-time disclosure, many security conferences operate formal AE processes after paper acceptance. Under AE, artifact evaluation committees assess submitted code, datasets, experimental scripts, and models against venue-specific criteria such as public availability, documentation quality, and the ability to exercise key results [14], [34]. Conceptually, AE and reproducibility span multiple levels, from artifact release to successful execution and result agreement with the paper's claims. Since early adoption by ACSAC and WiSec [2], [32], AE has become standard at major conferences including IEEE S&P, NDSS, and DSN, often accompanied by artifact badges indicating availability or functional reproducibility [21], [23], [31]. While these policies improve artifact availability and transparency, they evaluate artifact presence and basic usability rather than whether reported claims can be reproduced in practice. As a result, artifacts may be available yet non-executable, incomplete, or coupled to unstable dependencies and external APIs. This discrepancy between nominal compliance and practical reproducibility is especially evident in LLM security research, underscoring the need for systematic evaluations of reproduction outcomes under realistic constraints.

Reproducibility Measurement Research. Our work builds on prior reproducibility studies in computer systems and security that use standardized, time-bounded protocols to assess whether published results can be independently re-

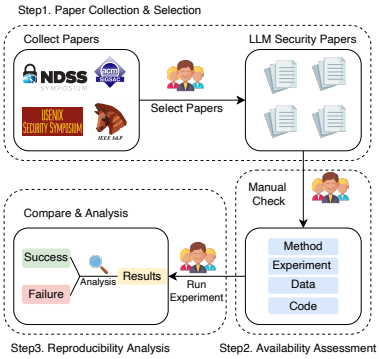


Figure 1: Overview of our reproducibility study methodology. We first collect and select LLM security papers from ACM CCS, USENIX Security, IEEE S&P, and NDSS. We then assess the availability of methods, experiments, data, and code. Finally, we reproduce their results and analyze the outcomes to identify patterns of success and failure.

produced and to systematically diagnose failure causes [10], [33]. We adopt a similar evaluation framework but apply it to LLM security, a setting where reproducibility is complicated by model and API drift, prompt- and decoding-sensitive behavior, gated or proprietary model assets, and evaluation pipelines that frequently depend on external services. Methodologically, we draw on prior discussions distinguishing repeatability from reproducibility and on recent tier-1 security measurement studies that emphasize systematic reproduction over anecdotal case-by-case efforts [8], [46].

Close Work. Evertz et al. [15] identify methodological pitfalls in LLM security research through a review of 72 papers, while our study complements them by directly reproducing 84 released artifacts and measuring whether reported claims can be independently verified. More details are in Appendix B.

3. Study Design

Workflow. To answer the research questions, we adopt an evaluation workflow (Figure 1) covering paper selection, artifact availability assessment, and reproducibility analysis.

Step 1: Paper Collection & Selection. We construct our dataset by applying a multi-step filtering process to papers published at four tier-1 security conferences: ACM CCS, USENIX Security, IEEE S&P, and NDSS. The study covers a three-year period from 2023 to 2025, beginning shortly after the public release of ChatGPT in November 2022 [35]. We focus on these conferences for comparability across LLM security papers and artifact-evaluation practices, so our results should not be interpreted as a census of all LLM security research.

Automated Retrieval and Keyword Filtering. We screen approximately 3,287 main-conference papers accepted by the four conferences during 2023–2025 using an automated retrieval pipeline. Candidate papers are identified through keyword matching over broad LLM-related terms, including *"large language model," "language model," "LLM," "agent," "RAG," "AI," "generative AI"* and *"GPT."* Paper metadata are obtained from DBLP and official conference-maintained sources. This screening yields 169 candidate papers.

Expert Annotation. The study focuses on papers in which LLM behavior is central to the security claim, rather than papers that merely use LLMs as tools for other security tasks. We employ *five domain experts* [1] to independently evaluate each paper against three inclusion criteria: (1) the paper's core contribution involves attacking or defending LLMs; (2) the paper focuses on security-relevant research, including safety, robustness, and privacy; and (3) the paper presents empirical evaluation with quantitative results.

Consensus. We resolve annotation disagreements through majority voting among the five experts [16], [39]. A paper is included in the final dataset only if at least three experts independently consider it satisfies the above inclusion criteria. This process results in a final dataset of 106 papers. Paper screening and selection involve five experts, while reproduction involves six experts.

Step 2: Availability Assessment. We conduct a systematic audit of resources provided by authors, following the methodology of Olszewski et al. [33]. Selected papers are evaluated along four dimensions using predefined criteria: (1) whether algorithmic descriptions specify the steps and implementation details needed for reproduction; (2) whether evaluation configurations specify the key hyperparameters, prompts, and environment settings needed to reconstruct the evaluation; (3) whether required datasets are publicly available and usable; and (4) whether implementation code or execution scripts are available.

Inter-Rater Reliability. To mitigate subjectivity in the assessment process, each paper is independently coded by five domain experts. We measure inter-rater reliability using the κ statistic [28], a robust statistic that accounts for chance agreement. The agreement score is $\kappa = 0.82$. This indicates a strong level of agreement among evaluators.

Availability Distribution. Across the 106 papers, 65 focus on attacks and 41 on defenses. Among these papers, 84 (79.25%) make their code publicly available, 7 (6.60%) include code repository links, but the repositories are incomplete, and 15 (14.15%) papers only present method descriptions, experiments, and data specifications without releasing any code artifacts from their papers.

Step 3: Reproducibility Analysis. We attempt to *reproduce key empirical claims from the 84 papers with publicly accessible code*. Given practical constraints, we focus on each paper's primary claims: attack success rates for attack papers and claimed robustness against representative attacks for defense papers. We compare reproduced outcomes with original claims to quantify result stability and evaluate asso-

1. These domain experts are graduate-level or above researchers with prior experience in LLM security, model safety, and reproducibility evaluation.

ciations between artifact evaluation badges and reproducibility rates.

Resource Constraints and Failure Conditions. To standardize our evaluation of practical reproducibility and ensure comparability with prior measurement studies [7], [33], we adopt the time-boxed methodology established by Collberg et al. [7] and Olszewski et al. [33]. We impose a standardized resource budget on each artifact:

- Setup and Debug (4 hours): We allocate up to 2 hours for environment setup and installation using provided documentation and up to an additional 2 hours to resolve installation failures, missing dependencies, or runtime errors. Troubleshooting combines manual debugging with then-state-of-the-art LLM tools, including GPT-5.2.
- Execution (12 hours): We allocate up to 12 hours for experiment execution, including model training, fine-tuning, and evaluation where required.

This budget is intended to measure practical reproducibility under standardized constraints, so artifacts requiring unreleased checkpoints, retraining from scratch, or substantially longer execution are treated as exposing reproducibility barriers. A reproduction attempt is classified as a failure under this protocol if (1) the environment cannot be configured within the setup budget; (2) technical errors or missing dependencies cannot be resolved within the debugging budget; or (3) execution exceeds the 12-hour limit without yielding results. To assess cutoff sensitivity, we continued four runtime failures beyond 48 hours (results are reported in Section 4.1).

Mitigating Model Deprecation. Reproducing LLM security research is complicated by frequent updates and deprecations of proprietary models. When specific closed-source model versions referenced in papers are unavailable, exact reproduction is infeasible. For closed-source models, we prioritize OpenAI experiments given institutional API access, which did not affect paper inclusion. We substitute the closest available model version when the original version is deprecated. For example, if `gpt-4o-2024-08-06` is unavailable, we use the most similar currently available `gpt-4o` variant. We treat such substitutions as feasible approximations, not exact reproduction, and flag them separately because they may affect result fidelity.

Review Protocol and Scoring. To reduce subjectivity, each reproduction attempt is independently conducted by *two domain experts* [2]. Following Olszewski et al. [33], we report the better-performing reproduction outcome across the two attempts as an estimate of an upper bound under reasonable evaluator effort. Results are matched if the primary metric differs by at most $\pm$ 5.00%. This accounts for stochastic variation in LLM outputs while preserving result-level fidelity as a pragmatic threshold. Otherwise, they are mismatched. Overall success rates use 95% Wilson confidence intervals; we avoid population-level generalizations beyond the evaluated artifacts. Reproduced results are compared against original claims to assess result stability.

2. Both reviewers are graduate-level researchers with at least 1.5 years of experience in LLM security.

TABLE 1: Reproducibility Results across Study Dimensions. "Total" denotes the number of papers evaluated; "Reproduced" denotes the number of successful reproductions.

Dimension	Category	Total	Reproduced	Success Rate
Paper Type	Attack	54	19	35.19%
	Defense	30	14	46.67%
Publication Year	2023*	1	1	100.00%
	2024	26	9	34.62%
	2025	57	23	40.35%
Conference	USENIX Security	56	25	44.64%
	ACM CCS	13	5	38.46%
	IEEE S&P	9	2	22.22%
	NDSS	6	1	16.67%
USENIX AE Status (Conditional)	w/ AE Badge**	14	9	66.67%
	w/o AE Badge	42	17	40.43%

*Limited sample size reflecting early-stage LLM adoption in security conferences.
**AE badge indicates passing all three evaluation stages.

We incorporate checklist-based verification of experimental setup determinism [1], [4], with details of the checklist in Appendix D.

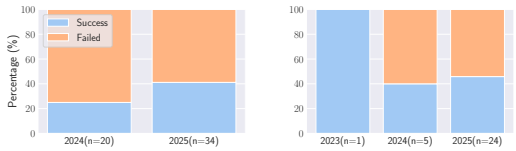
Time and Resource Consumption. We log clock time, GPU utilization, and memory usage for each reproduction attempt to quantify the resource burden of reproducing prior LLM security research. Experiments are conducted with OpenAI API access to avoid rate limiting and are executed on an NVIDIA DGX A100 system with eight A100 GPUs, as well as RTX 3090 and NVIDIA B200 GPUs, supporting multi-GPU parallelism. Across two independent attempts per artifact, the estimated equivalent experimental cost was at least $25,958 ($24,118 compute/server; $1,840 OpenAI API), including repeated compute use during debugging, trial-and-error, and execution. Successful and failed artifacts used about 460 and 350 GPU-hours, respectively (13.9 vs. 6.9 per artifact). This corresponds to about $309 per evaluated artifact across two attempts ($155 per attempt). The reproduction experiments are carried out by six experts. One expert could not join paper selection due to time constraints, so five conducted it. Each expert contributes approximately 2–3 working days per week, for a total estimated effort of over 3,900 person-hours. To support reproducibility of our study, the full list of the 106 papers and an anonymized artifact-level dataset for the 84 evaluated artifacts are publicly available (see Appendix A).

4. Overall Reproducibility Results

We begin by presenting the distribution and overall success rates of the reproduction. We then examine reproducibility from three critical dimensions: usability of provided artifacts, consistency of reproduced results with reported findings, and practical resource requirements, i.e., to answer **RQ1**.

4.1. Overall Statistics

Building upon Section 3, we summarize the distribution of collected papers and their reproducibility results. Unless



(a) Attack Methods (b) Defense Methods

Figure 2: Overall temporal trends in reproducibility success and failure rates across LLM attack and defense. n represents the total number of papers analyzed in each year.

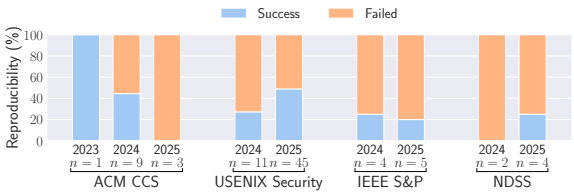


Figure 3: Reproduction outcomes by conference and publication year. n denotes the number of evaluated artifacts in each subgroup.

stated otherwise, rates use the 84 executable-artifact papers as the denominator and report the better of two attempts as an optimistic upper bound; individual attempts succeeded in 61/168 cases (36.31%), with 79/84 outcome-level agreement (94.05%). Table 1 breaks down outcomes by paper type, publication year, venue, and AE status. Since several subgroup cells are small, especially for IEEE S&P and NDSS, we interpret subgroup differences descriptively rather than as population-level conclusions.

Time-Budget Sensitivity. Seven setup failures in our reproduction records received an additional 2 hours of troubleshooting. These cases were identified retrospectively from the reproduction records rather than selected in advance by random sampling. None became executable. Four runtime failures were continued beyond 48 hours without producing results, leaving reproduction unchanged at 33/84. For example, one repository required 300 epochs of training from scratch, estimated at over 100 hours. These extended checks did not change the reproduction outcomes for the tested cases. However, our time-bounded protocol may still underestimate reproducibility for resource-intensive artifacts. We therefore interpret these failures as reflecting practical reproducibility under our evaluation protocol rather than inherent irreproducibility.

Reproducibility Trends for Attack and Defense Methods. We first analyze reproducibility trends over time, separating attacks and defenses to identify their specific failure patterns.

Attack Methods. Our analysis reveals a persistent fragility in attack artifacts, despite a marginal upward trend in reproducibility rates (Figure 2a). In 2024, only 25.00% (5 out of 20) of attack methods can be successfully reproduced. We find that many failures are related to environment setup and API availability issues, such as missing dependencies or deprecated LLM APIs. While the success rate improves to 41.18% (14 out of 34) in 2025, the majority of attack methods remain irreproducible and suffer from the same failure patterns. Our results indicate that adversarial LLM research continues to suffer from fragile experimental setups that degrade faster than they can be validated.

Defense Methods. Defense artifacts demonstrate better reproducibility relative to attacks, yet remain inconsistent (Figure 2b). In 2024, 40.00% (2 out of 5) of defenses are reproducible. Unlike attacks, which often fail due to missing or broken code dependencies and API volatility, defense failures are often associated with dependencies on proprietary services, private datasets, or unreleased training checkpoints (as detailed in Section 5.2). In 2025, the success rate improves to 45.83% (11 out of 24), indicating a shift toward better artifact reproducibility. Note that results from 2023 should be interpreted with caution due to the extremely small sample size (n=1), which limits the generalizability of conclusions for that year. We only include it for completeness purposes.

Comparative Assessment. Across the evaluated period, defense methods exhibit a consistently higher reproduction rate (46.67%) compared to attack methods (35.19%). However, neither category currently achieves the level of experimental robustness required for long-term independent validation. These findings highlight systemic reproducibility challenges in LLM security research. Addressing these challenges requires stronger artifact evaluation and more rigorous reporting norms to improve the reliability of published attack and defense methods.

Security-Specific Categories. Table 5 in Appendix C shows that reproducibility varies across LLM-security categories: prompt injection reproduced in 1/10 cases (10.00%), adversarial prompt attacks in 0/4, jailbreak in 5/14 (35.71%), memorization in 6/14 (42.86%), while backdoor and model extraction each reproduced in 3/5 cases (60.00%). Descriptively, prompt injection and adversarial prompt attacks show the lowest reproduction rates, while backdoor and model extraction show the highest among these categories. Given small category sizes, these patterns are descriptive.

Reproducibility Across Conferences. We then compare overall reproducibility success rates across the four tier-1 security conferences. As shown in Figure 3, we compare reproduction outcomes across the four tier-1 security conferences descriptively. In our evaluated sample, the observed reproduction rates were 44.64% (25 of 56) for USENIX Security, 38.46% (5 out of 13) for ACM CCS, 22.22% (2 out of 9) for IEEE S&P, and 16.67% (1 out of 6) for NDSS. Given the small and uneven subgroup sizes, these comparisons are descriptive and should not be interpreted as population-level differences across venues.

ACM CCS. The reproducibility success rate is 44.44% (4 out of 9) in 2024 and 0% (0 out of 3) in 2025. The observed rates differ between 2024 and 2025, although the 2025 subgroup is very small.

USENIX Security. In our evaluated sample, USENIX Security had the highest observed reproducibility rate among the

TABLE 2: Layered reproducibility results for USENIX Security papers with and without AE badges. Success rates are computed over papers reaching each stage; passing reproduction implies passing availability and functionality.

Stage	Status	# Success	# Failure	Success Rate
Availability	w/ badge	20	22	47.62%
	w/o badge	5	9	35.71%
Functionality	w/ badge	8	6	57.14%
	w/o badge	17	25	40.48%
Reproduction	w/ badge	6	3	66.67%
	w/o badge	19	28	40.43%

four conferences, with a higher observed rate in 2025 than in 2024. Papers published in 2024 exhibit a reproduction success rate of 27.27% (3 out of 11). Papers from 2025 achieve a higher reproduction rate of 48.89% (22 out of 45).

IEEE S&P. Reproducibility remained low at 25.00% (1 out of 4) and 20.00% (1 out of 5) in 2024 and 2025, respectively. The two observed rates are similar, with only a small difference between the 2024 and 2025 outcomes in our sample.

NDSS. Among the two papers published in 2024 that meet our criteria, none were reproduced, while 1 of 4 (25.00%) was reproduced in 2025.

AE and Reproducibility. We examine how AE badge participation relates to reproducibility outcomes for USENIX Security papers across successive reproduction stages. Following the three-stage pipeline from Section 2: (1) Availability, which measures if artifacts are released at all; (2) Functionality, which assesses if the code executes correctly given availability; and (3) Reproduction, which determines if the results match the paper's claims, conditional on successful execution. As shown in Table 2, availability rates are nearly identical for badged and non-badged papers, suggesting that in this dataset AE badges are not strongly associated with artifact release. In contrast, differences appear after artifacts are available. Badged papers have a higher functionality success rate (57.14% vs. 40.48%), and among papers that execute successfully, a higher reproduction success rate (66.67% vs. 40.43%). These comparisons are correlational and based on limited, uneven subgroups, which constrains the strength of any inference. AE participation may co-occur with other factors, such as author incentives or engineering effort, that also affect executability and result matching. Thus, within our evaluated USENIX Security sample, AE badge participation is associated with higher downstream success rates among released artifacts, while reproduction failures remain present.

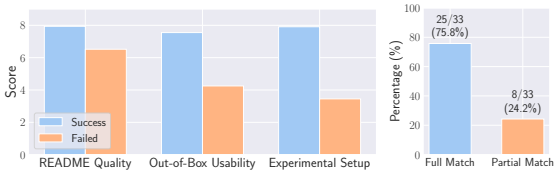
Overall Reproducibility Results. We successfully reproduced 33 out of 84 papers (39.29%), with a 95% confidence interval of [28.80%, 50.55%]. As shown in Figure 7 from Appendix C, successful reproduction does not always imply faithful result matching, with a substantial fraction of reproduced papers exhibiting only partial or mismatched outputs. Observed differences across attack/defense types, publication years, and conferences are descriptive and should not be interpreted as population-level differences. These observations characterize artifact reproducibility outcomes under our evaluation setting. These findings motivate our analysis of specific barriers to reproducibility (Section 5) and inform our recommendations for improving reproducible research in this domain (Section 7).

4.2. Usability Analysis

We evaluate artifact usability as a factor associated with the reproducibility of LLM security. Unlike traditional ML artifacts, LLM security artifacts often rely on evolving model APIs, external datasets, GPU-specific environments, and rapidly changing software dependencies, introducing additional execution complexity. We assess usability along three dimensions: (1) README quality, (2) out-of-the-box usability, and (3) experimental setup determinism, reflecting common friction points encountered during reproduction. Usability scores are checklist-based ordinal scores (1–10), with higher scores indicating better usability. The predefined scoring anchors are detailed in Appendix D. Figure 4a shows that artifacts with successful reproduction consistently exhibit higher usability scores across all three dimensions.

README Quality. High-quality documentation is associated with reproduction success: artifacts that reproduce successfully achieve higher README quality scores at 7.64 compared to 6.25 for failed artifacts. Artifacts that provide dependency specifications, explicit environment setup instructions, version-pinned libraries, and end-to-end execution workflows are more likely to reproduce reported results. In contrast, many failed artifacts provide incomplete or outdated README files, omit critical configuration steps, or reference deprecated repositories and only have abstracts. We also observe that LLM-specific artifacts frequently suffer from documentation drift. For example, instructions written for older model versions, e.g., earlier GPT or open-source checkpoint releases, may no longer produce comparable results due to silent model updates or distribution changes. This highlights the importance of precise documentation that accounts for model versioning and external service volatility.

Out-of-Box Usability. Out-of-the-box usability shows the largest observed usability gap between reproduction outcomes: successfully reproduced artifacts demonstrate usability scores of 7.55, higher than the 4.25 observed for failed artifacts. Successfully reproduced artifacts provide runnable code with minimal manual modification, automated setup scripts, and accessible dependencies, including datasets, pretrained checkpoints, and inference APIs. These artifacts allow evaluators to execute experiments with limited troubleshooting or reverse engineering. By contrast, failed artifacts frequently require substantial manual intervention, such as patching broken dependencies, reconstructing missing datasets, and resolving version conflicts. In LLM security research, additional friction arises from API rate limits, authentication requirements, unavailable proprietary models, and undocumented inference parameters, all of which can impede faithful reproduction. These findings suggest that



(a) Usability (b) Result Match

Figure 4: Checklist-based evaluation of artifact usability and result fidelity in LLM security research. Usability is measured through README quality, out-of-the-box usability, and experimental setup, while result fidelity captures whether reproduced artifacts exhibit partial discrepancies in broader reported results.

reproducibility in LLM security is tightly coupled with infrastructure readiness, as practical execution barriers can prevent independent verification even when methodologies are conceptually sound.

Experimental Setup Determinism. Experimental setup determinism is associated with artifact reproducibility: successfully reproduced artifacts achieve higher determinism scores at 7.91 compared to 3.45 for failed artifacts. Artifacts that control randomness through fixed seeds, pinned software versions, and containerized execution environments tend to produce more stable and repeatable results. Deterministic setup is particularly important for LLM-based experiments, where nondeterminism can arise from sampling randomness, model backend updates, or distributed execution variance. In contrast, many artifacts relying on loosely specified environments or unpinned dependencies often exhibit run-to-run variability or diverge from reported results, complicating faithful reproduction.

We observe a consistent usability gap across all three dimensions. Artifacts that successfully reproduce results exhibit systematically higher usability. Together, these results indicate a robust association between artifact usability and reproducibility in evaluations of LLM attacks and defenses. This suggests that usability-related factors represent a meaningful practical challenge for independent reproducibility in LLM security research. In Section 5, we further analyze failure cases to understand whether these breakdowns stem from methodological limitations or practical barriers.

4.3. Result Matching Analysis

Beyond usability, we evaluate whether reproduced experiments yield results that match those reported in the original LLM attack and defense papers. While an artifact may execute successfully, faithful reproduction requires that reproduced performance metrics, success rates, or attack effectiveness align with reported results. Figure 4b summarizes result fidelity among the 33 successfully reproduced artifacts. Among the 33 successfully reproduced artifacts, 25 show no broader discrepancy, while 8 (24.24%) meet the primary reproduction criterion but exhibit partial discrepancies in other reported results. Failed artifacts are not treated as an independent fidelity comparison because many produce no comparable output. Varying the matching threshold from $\pm 4\%$ to $\pm 6\%$ changes matched artifacts only from 26/33 to 27/33, indicating limited threshold sensitivity. Even when code executes successfully, reproduced results may differ due to changes in model behavior, API updates, dependency drift, or undocumented evaluation settings. These partial discrepancies can occur even when the primary metric remains within our $\pm 5\%$ reproduction criterion, showing that reproduction success does not necessarily imply complete fidelity across all reported results. For example, certain jailbreak attacks effective against deployed LLMs at the time of publication may show reduced success when reproduced later, following safety filter updates. This fidelity risk is particularly salient for prompt-based attacks, such as jailbreak and prompt injection, whose outcomes can be sensitive to prompt wording, decoding parameters, model/API updates, and evolving safety behavior.

Taken together, these results indicate that reproducibility in the LLM security domain should be assessed not only in terms of execution success but also in terms of result fidelity. When reproduced outputs deviate from reported findings, running released artifacts alone may not provide complete validation of security claims. This discrepancy suggests that result mismatches in LLM security experiments may stem from multiple factors, which are further analyzed in terms of underlying technical causes in Section 5. These observations highlight the importance of controlling evaluation conditions, documenting model and API versions, and reporting statistical variability when assessing LLM attack and defense methods.

Lesson: Under our evaluation protocol, fewer than half of the evaluated LLM security artifacts can be successfully reproduced, and execution success does not guarantee result fidelity. We face unique challenges in reproducing LLM security, where models, APIs, and safety filters change after publication. Attack artifacts are especially fragile, as they target live systems that are patched, causing reproduced results to decay even when code remains functional. Observed reproduction rates vary across venues and years in our sample. Given the small subgroup sizes, these differences are descriptive. Within our evaluated USENIX Security sample, AE badge participation is associated with a higher observed reproduction rate (66.67% vs. 40.43%), without implying a causal effect.

5. Barriers to Reproducibility

Despite our reproduction efforts, 60.71% (51 out of 84) of papers failed to reproduce successfully (95% CI: [49.45%, 71.20%]). In this section, we examine why reproductions fail, which subsequently addresses **RQ2** by

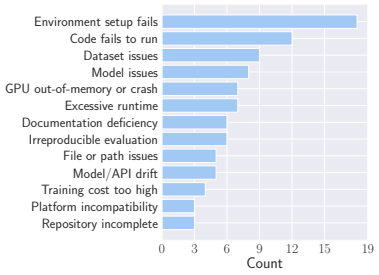


Figure 5: Breakdown of reproducibility failure causes across LLM security artifacts. Counts reflect the best-performing attempt per artifact; categories are non-exclusive.

identifying and analyzing the barriers that prevent successful reproduction.

5.1. Failure Analysis

During the reproduction process, domain experts attempted to reproduce each artifact and recorded every failure case. We define a taxonomy of 13 distinct failure modes to track these failures. Detailed analysis of these failure modes and their underlying causes is provided in Section 5.2. Note that a single artifact often leads to multiple failure modes. Our experts record a total of 93 failure instances across these 13 failure modes based on the best-performing reproduction attempt per artifact (see Figure 5). Each failure mode corresponds to an observable breakdown during reproduction, and a single artifact may be assigned multiple failure modes if multiple independent breakdowns are observed. For each failure mode, an artifact contributes at most one instance, regardless of how many times the same issue occurs during execution. Across security categories, the most frequent non-exclusive failure modes largely overlap, with no clear category-specific pattern (Table 6 in Appendix C).

Environment and Execution as Primary Failure Surfaces. Among these 13 failure modes, *environment setup failure* was the single most frequent mode ($n = 18$). Our experts find that many artifacts cannot be executed due to installation errors, unresolved dependencies, incompatible CUDA or driver configurations, or mismatches between expected and available software stacks. These failures often occur before any experiment can be executed, representing a primary barrier to reproducing artifacts. A closely related failure mode is *code fails to run*. That is, even when dependencies were successfully installed, numerous artifacts encountered runtime errors, crashed during execution, or failed to complete the intended pipeline. These failures, 12 instances in total, stem from broken scripts, misconfigured execution commands, and incomplete experiment workflows, all of which result in early termination before evaluation can proceed. Together, environment- and execution-related failures account for 30 out of 93 observed reproducibility failures, highlighting underlying software environment issues as the dominant failure modes.

Data, Model, and Resource-Related Failure Modes. We also observe a reasonable number of failure instances attributed to *dataset issues* and *model issues* with 9 and 8 failures, respectively. In these cases, reproduction attempts are blocked because the required datasets are missing, incomplete, or incompatible with the provided code, or because referenced model checkpoints cannot be obtained or loaded successfully. Computational bottlenecks also present significant hurdles. Failures due to *GPU out-of-memory or crash* and *excessive runtime* together account for 14 failures, indicating that computational constraints represent a common reproducibility bottleneck. In addition, 4 failures attributed to *training cost too high* reflect cases where reproduction is technically possible but impractical under our setting due to substantial computational requirements. Taken together, data-, model-, and resource-related failures account for 35 failure instances.

Evaluation, Documentation, and Interface Failures. Even when the code runs successfully, reproduction may fail due to *irreproducible evaluation*, which accounts for 6 failures. In these cases, experiments are complete, yet reported results cannot be replicated due to unclear evaluation protocols, inconsistent scoring outputs, or mismatches between expected and reproduced metrics. We also record 6 failures due to *documentation deficiency*, which impedes correct experiment reproduction by leaving uncertainty about configuration choices, preprocessing steps, or runtime parameters. In addition, 13 failures from interface and compatibility issues, including *Model/API drift*, *file or path issues*, and *platform incompatibility*. These cases demonstrate that successful execution is a prerequisite, not a guarantee, of successful reproduction.

Repository Completeness and Artifact Integrity. Finally, *repository incomplete* represents a small but tangible barrier, accounting for 3 failures. In these cases, essential components, such as core scripts, configuration files, model checkpoints, or download links, are missing from the public release. These failures indicate a breakdown in artifact packaging and archival, blocking independent reproduction from the released artifact without additional materials. These failure categories describe how reproduction attempts break down, but not the underlying technical causes. Multiple failure modes may stem from the same root cause, and a single artifact may exhibit multiple breakdowns. Failure modes frequently co-occur, particularly environment conflicts combined with code execution errors. Each failure mode is defined by a concrete operational criterion describing when the corresponding breakdown is recorded, with representative examples discussed in Section 5.2.

5.2. Root Cause Analysis

While the previous section characterizes observable reproduction failures, we analyze their underlying technical root causes. We observe that failure modes are not independent. In particular, environment setup failures frequently co-occur with code execution failures, indicating that de-

TABLE 3: Failure modes and root causes, grouped by generic and LLM-related barriers. Counts are non-exclusive.

Type	Failure Mode	#	Root Cause Category	Underlying Reason
Generic artifact failures	Environment setup fails	18	Dependency/environment conflict	Library version conflicts and CUDA incompatibilities
	Code fails to run	12	Non-functional or broken code	Incomplete README, setup instructions, or prompt templates
	Documentation deficiency	6	Missing or unclear documentation	Incomplete README or missing setup instructions
	File or path issues	5	File or path errors	Missing files, folders, or invalid paths
	Platform incompatibility	3	OS or platform constraints	Unsupported OS, hardware, or CPU architecture
	Repository incomplete	3	Missing core code or artifacts	Core components not released or removed
LLM-amplified or LLM-specific failures	Dataset issue	9	Dataset access or missing data	Dead links, gated access, or missing test data
	Model issue	8	Model access or missing checkpoints	Gated models, missing checkpoints, or unavailable model versions
	GPU out-of-memory or crash	7	Compute or GPU limitations	Model size exceeds GPU memory capacity
	Excessive runtime	7	Runtime or timeout	Execution stalls or exceeds practical time limits
	Irreproducible evaluation	6	Evaluation not reproducible	Private tools, missing evaluators, manual or prompt-sensitive scoring
	Model/API drift	5	API/model drift	Deprecated models, API changes, or post-publication patches
	Training cost too high	4	Excessive training cost	Training requires prohibitive compute resources

pendency and environment drift often propagate into downstream runtime errors. Table 3 summarizes the mapping from surface barriers to root-cause categories and technical explanations. Across these failures, most barriers are generic or mixed, while closed-model deprecation and API/model drift affect 5-7 artifacts (5.95%-8.33%) and represent the clearest LLM-specific result-fidelity risks.

Dependency and Environment Conflicts. Categorized as *dependency or environment conflict*, it underlies many environment setup failures. These issues frequently stem from incompatible library versions, CUDA mismatches, deprecated frameworks, or undocumented system requirements under default settings. LLM artifacts are especially sensitive to such conflicts because they often depend on rapidly evolving inference libraries, GPU runtimes, and model-serving stacks that change more frequently than traditional ML toolchains. For example, we observed repositories that pinned `torch < 2.0` while relying on features introduced in `torch >= 2.1`, as well as artifacts that broke due to incompatible updates in PEFT that rendered LoRA adapters unusable. In other cases, missing or insufficient `requirements.txt` files forced manual environment reconstruction, substantially increasing setup friction. Separately, LLM-assisted debugging identified a root/sudo privilege requirement in one setup failure, but the artifact remained non-executable. Given the rapid evolution of LLM or ML-based software stacks, artifacts that lack environment pinning or containerization become increasingly fragile over time, making reproduction highly sensitive to platform drift.

Non-functional or Broken Code. Another major root cause, *non-functional or broken code*, corresponds to artifacts where execution fails despite successful environment setup. These failures involve missing modules, syntax errors, broken pipelines, or incomplete implementations. For instance, we observe artifacts with missing or non-importable modules (e.g., import failures in `transformers` and `huggingface_hub`), missing symbols such as `split_torch_state_dict_into_shards`, and LangChain execution errors that caused pipelines to crash at runtime, preventing any evaluation output. We also encounter data and configuration issues, such as missing dataset paths (`FileNotFoundError`), invalid argument parsing (`HfArgumentParser ValueError`), incorrect task settings, and malformed model metadata (e.g., missing `config.json`), which render execution pipelines non-functional. In other cases, repositories contained incomplete experiment pipelines, where core scripts or execution stages were absent, leading to early termination before producing results. In such cases, reproduction fails not due to conceptual flaws but because the released artifact does not constitute fully executable research. In LLM security artifacts, such failures are often compounded by complex multi-stage pipelines involving model querying, prompt generation, post-processing, and evaluation, increasing the likelihood of silent breakage.

Missing or Inaccessible Data and Models. We identify *dataset access or missing data* and *model access or missing checkpoints* as recurring root causes of reproduction failures. Many artifacts depend on datasets that are gated, relocated, or no longer publicly available, including expired Google Drive links, partially released datasets on Zenodo, or datasets not provided, while others reference model checkpoints that are missing, restricted, or incompatible with current frameworks, such as invalid DeepSpeech2 checkpoints. This problem is particularly acute in LLM security, where evaluation frequently relies on proprietary APIs, gated model weights, or restricted benchmarks that cannot be redistributed. For example, some artifacts referenced datasets hosted on inactive or invalid links or depended on gated Hugging Face models or invalid checkpoint identifiers, preventing access to required evaluation resources. In other cases, artifacts referenced invalid model identifiers (e.g., `meta-llama/Llama-2-70b`) or failed to release pretrained weights, preventing access to required model checkpoints and rendering reproduction infeasible even when code is otherwise functional. In LLM security research, where evaluation relies on proprietary APIs or gated model weights, the absence of stable asset archiving significantly undermines long-term reproducibility.

Compute and Resource Constraints. We identify *compute or GPU limitations* as a major root cause of reproduction failures, including GPU memory exhaustion and hardware resource shortages. These limitations manifest as CUDA out-of-memory errors on A100, RTX 3090, and B200 GPUs under recommended configurations, device mismatch errors and failures involving incompatible or offloaded weights (e.g., `.to()` applied to offloaded parameters). Some ar-

tifacts fail due to *runtime or timeout* constraints, where execution stalls or exceeds practical time budgets, such as jobs timing out after 12 hours or producing no results after 48 hours. In more extreme cases, failures arise from *excessive training cost*, making full reproduction impractical under our setting even when the technical pipeline is otherwise functional. Four artifacts are classified as *training cost too high*. Table 7 in Appendix C details their training requirements and observed or estimated resource burdens. These cases illustrate how substantial training requirements can become a practical reproduction barrier. For instance, `huggyllama/llama-65b` exhibits out-of-memory errors on A100 and B200 GPUs. These root causes highlight the growing role of resource scalability as a structural barrier to reproducible LLM experimentation.

Documentation and Evaluation Deficiencies. Beyond technical infrastructure, we identify *missing or unclear documentation* as a recurring root cause of reproduction failures. Incomplete `README.txt`, absent setup instructions, and underspecified execution parameters prevent faithful reconstruction of experiments. For example, some artifacts omitted key evaluation parameters such as prompt templates or scoring scripts, making it impossible to match reported metrics. Since LLM security evaluations often depend on subtle prompt wording, decoding parameters, and black-box model behaviors, incomplete documentation can have outsized effects on reproducibility. Relatedly, *evaluation not reproducible* emerges as a distinct root cause, as evaluation is often non-automatable, relying on private harnesses, manual scoring, or single-sample testing that cannot be externally replicated. *Irreproducible evaluation* is assigned if the artifact executes successfully, but the evaluation procedure cannot be reproduced due to missing evaluation scripts, reliance on private or unavailable tooling, or non-automatable manual scoring without a released rubric.

Platform, Interface, and Temporal Instability. We identify *API/model drift* as a recurring root cause of reproducibility failures when artifacts depend on external services or evolving interfaces. Breaking API changes, deprecated endpoints, or modified response formats can silently invalidate previously functional code. We observed failures from deprecated API endpoints or SDK updates that altered response formats, breaking previously functional pipelines. Another category of failures arises from *file or path errors*, where artifacts assume specific directory layouts, hard-coded absolute paths, or unavailable filesystem locations. For example, we encounter nonexistent dataset paths, inconsistencies between audio files and metadata, hard-coded local paths in scripts (e.g., `download_model.sh`), and missing utility files that break automated setup. Such assumptions frequently break when reproductions are conducted on different machines, operating systems, or storage environments. We also observe that certain LLM-based attacks, such as jailbreak and prompt injection, exhibit temporal fragility, as providers rapidly patch model behavior or safety filters, rendering previously effective exploits obsolete. In addition, *OS or platform constraints* restrict reproducibility when artifacts assume unsupported hardware, operating systems, or execution environments. For example, we encountered CUDA/GPU incompatibilities, Conda build failures (e.g., `flash-attn`), tightly coupled library stacks (e.g., `langchain` with `llava==1.0.1`), and missing credentials for gated models such as LLaMA-2 that prevented execution.

Artifact Completeness and Release Practices. Some failures, accounting for 3.23% of observed failure instances, originate from *missing core code or artifacts*, including absent model checkpoints, incomplete repositories, or removed auxiliary files. For instance, some repositories lacked pretrained checkpoints, runnable pipelines, or essential scripts and core components, rendering the artifacts unusable out of the box. In other cases, authors released only base model code without pretrained weights, requiring extensive manual modification before execution or forcing reproducers to retrain models from scratch, often incurring multi-day training time and substantial computational cost. These cases point to broader challenges in artifact release practices, where essential components are not preserved or distributed alongside the publication.

Taken together, our root cause analysis shows that reproducibility failures in LLM security research arise from multiple interacting systemic weaknesses rather than a single dominant factor. Many of these issues are exacerbated by LLM-specific characteristics such as reliance on proprietary APIs, rapidly evolving model behavior, and sensitivity to prompt and decoding configurations.

Paper-Artifact Consistency. To examine paper-artifact consistency, four annotators assessed 11 predefined items across 24 pairs (12 failed and 12 reproduced; κ=0.77). We identify 51 discrepancies among the 264 checks (19.3%). Checkpoint mismatches occur in 6/12 failed pairs versus 0/12 reproduced pairs, and inference-parameter mismatches in 5/12 versus 1/12. Checkpoint and inference-parameter discrepancies are more prevalent among failed reproductions, although the small sample limits stronger inference. The full item-level results are reported in Table 8 in Appendix C.

Lesson: Reproducibility in LLM security faces barriers absent from prior ML reproducibility studies. Beyond general failures such as broken environments and missing artifacts, LLM-specific barriers, including model and API deprecation, post-publication safety patches, prompt and decoding sensitivity, and prohibitive inference costs, can invalidate results even when artifacts remain executable. These are not implementation errors; they are inherent properties of LLM-based systems that make reproducibility a moving target.

6. Practices Enabling Reproducibility

To answer **RQ3**, we analyze the 33 successfully reproduced papers (39.29% of evaluated artifacts) to identify

TABLE 4: Practices associated with successful reproduction. Percentages indicate artifacts satisfying each criterion among successful and failed outcomes, based on rubric-binarized checklist scores ($\geq$8). Differences are tested with Fisher's exact test.

Dimension	Successful Artifacts	Failed Artifacts	Gap
Well-Documented Instructions	63.64%	41.18%	22.46%
Out-of-the-Box Usability	63.64%	15.69%	47.95%
Setup Time Within 2hrs	87.88%	70.59%	17.29%
Clear Experiment Setup	66.67%	1.96%	64.71%

practices that enable reproducibility. We investigate what makes these papers reproducible, identify factors associated with successful reproduction, and showcase examples of best practices.

6.1. Success vs. Failure Comparison

We analyze reproduced artifacts to identify patterns correlating with reproducibility success in LLM security. This highlights characteristics, such as documentation completeness and execution usability. Table 4 summarizes these characteristics across successful and failed artifacts, revealing systematic differences that inform best practices for artifact release. To quantify these differences, we use BH-corrected Fisher's exact tests on binarized checklist indicators and report odds ratios (95% CI; +0.5 to avoid infinite ORs), with q-values denoting BH-adjusted p-values where $OR > 1$ success is favored [13] (see Section E). Artifacts may exhibit multiple reproducibility-enabling characteristics.

Well-Documented Instructions. We observe a strong association between high README quality and reproduction success. 63.64% of successfully reproduced artifacts exhibit high README quality, compared to 41.18% among failed artifacts, a 22.46% gap (OR=2.77, 95% CI 1.11-6.94, $q = 0.04$). This difference suggests that artifacts with clearer documentation are substantially more likely to support successful independent reproduction. Across evaluated artifacts, high README quality is substantially more prevalent among successfully reproduced cases than among failed ones. Although documentation alone does not guarantee reproducibility, its consistently higher occurrence in successful artifacts indicates that README quality is a robust empirical correlate of reproducibility in LLM security.

Out-of-the-Box Usability. High usability shows one of the largest disparities between successful and failed artifacts. Successfully reproduced artifacts demonstrate a high usability high-quality artifact rate of 63.64%, compared to only 15.69% for failed artifacts, representing a substantial gap of 47.95% (OR=10.23; $q = 1.24 \times 10^{-5}$). This pronounced disparity indicates that usability-related characteristics are consistently more prevalent in successfully reproduced artifacts than in unsuccessful ones. Across evaluated artifacts, usability stands out as a distinguishing artifact-level property between successful and failed reproduction cases.

Result Fidelity. Result matching is meaningful only for artifacts that produce comparable outputs. We therefore do not treat fidelity as an independent factor associated with reproduction success. Instead, it serves as a post-reproduction assessment of whether successfully reproduced artifacts remain consistent with the broader results reported in the paper. As discussed in Section 4.3, successful reproduction can still coexist with partial discrepancies, highlighting the distinction between executability, primary-claim reproduction, and broader result fidelity.

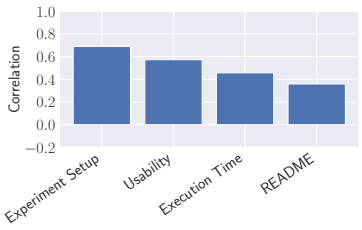


Figure 6: Correlation between artifact properties and reproduction success. Bars show Pearson correlation coefficients, and higher values indicate stronger association with reproduction success.

Setup Time Within 2hrs. Setup efficiency shows a more moderate but consistent difference between successful and failed artifacts. A total of 87.88% of successful artifacts can be set up within two hours, compared to 70.59% of failed artifacts, corresponding to a 17.29% gap (OR=4.04, 95% CI 1.07-15.22, $q = 0.04$). Although the separation is smaller than that observed for other dimensions, faster setup times occur more frequently among successfully reproduced artifacts. This pattern suggests that setup efficiency represents a secondary, yet observable, empirical distinction between successful and unsuccessful reproduction outcomes.

Clear Experiment Setup. Clarity of experimental setup constitutes another notable differentiation between successful and failed artifacts. Our analysis reveals that 66.67% of successful artifacts include clearly specified experimental configurations, compared to 1.96% of failed artifacts, a 64.71% gap (OR=109.20; $q = 6.33 \times 10^{-11}$). This sizable gap indicates that explicit configuration disclosure is substantially more prevalent among artifacts that achieve successful reproduction. In comparative terms, experiment setup clarity emerges as a salient empirical characteristic distinguishing successful reproduction cases from failures.

Across all four evaluated dimensions, successful artifacts consistently exhibit higher prevalence of high README quality, usability, setup efficiency, and experiment setup clarity than failed artifacts.

6.2. Correlation of Reproducibility

To complement the analysis in Section 6.1, we analyze artifact properties that are associated with, rather than causally determine, reproduction success. Figure 6 summarizes the associations: setup ($r = 0.68$), usability ($r = 0.59$), execution time ($r = 0.49$), and README quality ($r = 0.38$). All correlations are computed at the

artifact level using binary indicators and reflect association rather than causation.

Experiment Setup. It shows the strongest observed association among these artifact properties ($r = 0.68$, $p = 1.11 \times 10^{-12}$). Artifacts that explicitly expose experimental configurations, including model versions, hyperparameters, prompt formats, and evaluation scripts, reduce ambiguity in reproduction and limit configuration drift across environments. For example, successful artifacts specify structured attack or defense pipelines, such as graph-based modeling with CFG/AST/DFG construction, bandit-based vulnerability scanning, and iterative prompt optimization. They also expose experimental semantics via configuration files, including model configurations, decoding settings, prompt templates, query strategies, and LLM provider mappings, and provide explicit data generation pipelines, such as `dataset.py` for prompt loading, query policies, train/test construction, and automated JSONL dataset creation. This is likely because LLM-based attacks and defenses are highly sensitive to implementation details, where even minor configuration mismatches can lead to substantially different behaviors or evaluation outcomes. Given the sensitivity of LLM security evaluations to small configuration changes, transparent setup specifications play a critical role in supporting auditability and experimental integrity.

Usability. This factor exhibits a higher point estimate with reproduction success ($r = 0.59$, $p = 4.91 \times 10^{-9}$). Artifacts that offer coherent execution workflows, automated setup steps, and accessible dependencies reduce the operational burden on evaluators and minimize the risk of silent execution failures. For example, successful artifacts provide reproducible Python + Conda environments with pinned dependencies, comprehensive requirement specifications, and unified installation manifests (e.g., `jailbreak.yml`), enabling low-friction and consistent execution across platforms. This is particularly important in LLM security research, where reproduction often involves multi-stage pipelines, such as prompt construction, model querying, and response parsing, that are prone to breakage from minor runtime or environment inconsistencies. More broadly, this factor reflects a principle of operational reliability: reproducible security research benefits from pipelines that can be executed end-to-end without ad hoc intervention or undocumented workarounds.

Execution Time. A moderate positive correlation is observed for this factor ($r = 0.49$, $p = 2.50 \times 10^{-6}$). Faster-running artifacts are more likely to complete full evaluation cycles and to be iteratively tested by independent researchers, particularly in resource-intensive LLM security settings. For example, most successfully reproduced artifacts in our dataset complete within approximately 2.25 hours per experimental run, while a heavy-tailed subset of pipelines extends beyond 10 hours, making them highly sensitive to available compute budgets and more prone to incomplete or abandoned attempts. This is especially relevant for LLM-based evaluations, where high inference costs, API rate limits, and long multi-query pipelines can discourage repeated runs or lead to premature termination of experiments. Artifacts that release cached results or intermediate outputs help amortize execution cost and reduce recomputation overhead, improving the feasibility of repeated validation and systematic reproduction. While execution speed does not directly determine correctness, it influences the practical feasibility of repeated validation and systematic reproduction efforts.

README Quality. Reproduction success correlates moderately with this factor ($r = 0.38$, $p = 3.54 \times 10^{-4}$). High-quality documentation, often in README files, that clearly describes environment setup, dependency requirements, and execution procedures helps reduce interpretation errors and improves transparency in artifact release. For instance, effective `README.txt` clearly defines tool identity and provides quick start instructions, CLI usage examples, and demo notebooks, enabling evaluators to correctly understand and invoke the artifact. This is particularly important in LLM security research, where reproduction often depends on subtle prompt formatting choices, API usage details, and evaluation protocols that are difficult to infer without explicit written guidance. Although documentation alone cannot ensure successful reproduction, it increases the probability that evaluators can correctly configure and execute the artifact.

Overall, successful reproduction in LLM security is primarily characterized by artifact properties that support consistent evaluation, reliable execution, and low-friction experimentation. This suggests that LLM security reproducibility depends more on robust, clear, and practically executable evaluation pipelines than on isolated technical correctness. Exploratory stratification by AE badge status, publication year, and paper type yields broadly similar association patterns across subgroups. However, given the small and uneven subgroup sizes, these results should be interpreted as descriptive robustness checks rather than evidence of subgroup-specific effects.

Lesson: Reproducibility in LLM security builds on, but extends beyond, the practices identified in prior ML reproducibility studies. While documentation and usability matter in any setting, LLM security introduces additional requirements: explicit prompt templates and decoding configurations, specific model versions and API provider mappings, and cached outputs to offset prohibitive inference costs. Without these, even minor configuration drift in LLM-based evaluation pipelines can silently invalidate results entirely.

7. Recommendations

Lessons Learned. Our reproduction efforts show that the main barrier to scientific verification is not simply the absence of artifacts, but a combination of incomplete experimental specifications, inaccessible assets (e.g., datasets and model checkpoints), fragile execution environments, and evaluation pipelines. Reproducing LLM security research

typically involves complex, multi-stage workflows spanning data preprocessing, model inference, and evaluation, where these components are tightly coupled, and failures at any stage can halt the entire reproduction process. Based on these observations, we distill the following core lessons:

- **Availability $\neq$ Reproducibility.** Publicly available artifacts do not guarantee reproducibility. Even accessible code, underspecified configurations, implicit assumptions, and brittle dependencies can render an artifact irreproducible. Reproducibility depends less on availability, more on stability and clarity of end-to-end workflow.
- **Environment Setup as a Main Barrier.** A substantial number of reproduction failures ($n = 30$) occur during environment setup and execution. Our findings show that many releases lack details that ensure environmental portability. Dependencies on local-only configurations or undocumented system constraints effectively prevent independent validation by the broader research community.
- **Fragility of the LLM Evaluation Pipeline.** LLM security research operates within a rapidly evolving ecosystem. Reproducibility is therefore affected by shifting API versions, model updates, large external datasets, and significant computational requirements, especially on GPUs. As a result, evaluation pipelines become highly sensitive to even minor implementation details. Without implementation-level specifications (e.g., exact system prompts, model checkpoints, or preprocessing steps), the evaluation pipeline often breaks, leading to failed reproduction attempts.

These findings indicate that code availability alone is insufficient for independent verification. Artifact evaluation criteria emphasizing executability, environmental portability, and complete workflow specification may better promote reproducible research.

Implications for Authors. To facilitate reproducibility, authors should prioritize making experimental workflows complete and executable. This includes providing implementation-level descriptions that specify threat models, prompts, model interaction patterns, decision logic, and experimental configurations, including exact model checkpoints, inference parameters, system prompts, hyperparameters, and scoring rules. Checkpoint discrepancies occur in 6/12 failed versus 0/12 reproduced pairs, and inference-parameter discrepancies in 5/12 versus 1/12. Durable access to datasets and model checkpoints should be maintained through stable hosting and integrity checks. Authors should release fully executable, end-to-end code artifacts that automate data preparation, execution, and evaluation, with pinned dependencies, containers, and fixed seeds where possible, while reporting variability when stochastic LLM behavior affects results. When experiments rely on proprietary APIs or evolving model services, authors should document exact model/API versions, prompts, decoding settings, and evaluation dates, provide fallback and access guidance, and maintain released artifacts for long-term executability. To support practical reproduction under realistic resource constraints, authors should report approximate compute and runtime costs and provide cached intermediates or pretrained checkpoints when feasible.

Implications for Publication Practices. Our findings motivate publication practices that emphasize executable artifacts, complete configurations, and result fidelity. Papers that clearly disclose the four core components examined in this study, namely method descriptions, experimental configurations, datasets, and code artifacts, better support independent reproduction. We further observe that artifact availability alone is insufficient for faithful result verification, as successful execution does not necessarily imply result consistency. Artifact evaluation can therefore prioritize executability, exact model/checkpoint and inference configurations, and agreement with key reported results rather than availability alone. Such practices may support independent verification of LLM security claims.

8. Discussion & Limitation

Discussion. Our findings show that LLM-security reproducibility is not only about artifact failure but also result fidelity: executable artifacts may no longer validate the original claim as models, APIs, prompts, or safety policies change. This matters because LLM security results inform real-world decisions about model safety and deployment. A key insight is that reproducibility in LLM security is not a binary but a multi-stage process, spanning artifact release, successful execution, and result reproduction. In practice, successful execution alone is often insufficient to validate reported claims, particularly given the sensitivity of LLM evaluations to prompts, decoding settings, and model backends. Our results further highlight two structural challenges. First, practical feasibility under realistic compute and time budgets is a critical but underappreciated constraint on reproducibility. Second, model version drift, including evolving model APIs and deprecated versions, complicates long-term reproducibility of security claims, especially in closed-source models. Our conclusions apply to practical reproducibility in this 2023-2025 four-conference corpus under our protocol, not to all LLM-assisted security research. Together, these findings suggest that improving reproducibility in LLM security will require coordinated advances in research practices, tooling robustness, and publication norms rather than isolated technical fixes.

Limitation. Our study has several limitations. First, we focus on LLM attack and defense papers published at four tier-1 security conferences between 2023 and 2025, so our findings may not generalize to other conferences, workshops, preprints, or broader areas of ML or LLM security. Second, reproduction attempts are limited to papers that release executable artifacts. Papers without executable artifacts are excluded, which may cause our reported success rates to overestimate reproducibility in the broader literature. Third, our time-bounded protocol may underestimate reproducibility for resource-intensive artifacts. Extended checks did not change outcomes for the tested cases, but they do not rule out successful reproduction under substantially

larger resource budgets. Thus, these failures reflect practical reproducibility under our protocol rather than inherent irreproducibility. Fourth, our conference- and AE-level analyses are based on relatively small and uneven subgroups. These comparisons are therefore descriptive and should be interpreted cautiously rather than as population-level differences or causal effects. Fifth, some results depend on proprietary APIs and evolving model versions; when exact versions were unavailable, we used the closest alternatives as feasible approximations rather than exact reproductions, which may affect result comparability. Lastly, despite two independent attempts and structured criteria, result matching and failure diagnosis remain judgment-based. LLM assistance lacks a no-LLM control, and authors were not contacted to preserve independent reproducibility.

9. Conclusion

This paper presents a systematic study of reproducibility in recent LLM security research. We find that reproducibility remains limited in practice, with many released artifacts failing to execute or reproduce reported results due to fragile environments, missing assets, evolving model dependencies, high computational costs, and evaluation sensitivity. Our results indicate that successful reproduction depends on preserving a complete chain of experimental evidence, including method descriptions, experimental configurations, datasets, and executable artifacts. Papers providing these components achieve higher reproduction success, and within our evaluated USENIX Security sample, AE participation is associated with higher downstream success rates. Overall, our findings suggest that improving reproducibility in LLM security requires experimental transparency, artifact robustness, and long-term verifiability beyond merely releasing code.

References

[1] AAAI. Reproducibility Checklist. https://aaai.org/conference/aaai/aaai-23/reproducibility-checklist/, 2023.

[2] ACSAC. ACSAC Artifacts. https://www.acsac.org/2017/artifacts/, 2017.

[3] Daniel Ayzenshteyn, Roy Weiss, and Yisroel Mirsky. Cloak, Honey, Trap: Proactive Defenses Against LLM Agents. In *USENIX Security Symposium (USENIX Security)*, pages 8095–8114. USENIX, 2025.

[4] Anya Belz. QRA++: Quantified Reproducibility Assessment for Common Types of Results in Natural Language Processing. *CoRR abs/2505.17043*, 2025.

[5] Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. JailbreakRadar: Comprehensive Assessment of Jailbreak Attacks Against LLMs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 21538–21566. ACL, 2025.

[6] Zhixuan Chu, Yan Wang, Longfei Li, Zhibo Wang, Zhan Qin, and Kui Ren. A Causal Explainable Guardrails for Large Language Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1136–1150. ACM, 2024.

[7] Christian Collberg, Todd Proebsting, and Alex M. Warren. Repeatability and Benefaction in Computer Systems Research. *University of Arizona Technical Report*, 2015.

[8] Christian Collberg and Todd A. Proebsting. Repeatability in Computer Systems Research. *Communications of the ACM*, 2016.

[9] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, Ilia Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. Scalable Watermarking for Identifying Large Language Model Outputs. *Nature*, 2024.

[10] Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann. Reproducibility and Replicability of Web Measurement Studies. In *The Web Conference (WWW)*, pages 533–544. ACM, 2022.

[11] Tian Dong, Minhui Xue, Guoxing Chen, Rayne Holland, Yan Meng, Shaofeng Li, Zhen Liu, and Haojin Zhu. The Philosopher's Stone: Trojaning Plugins of Large Language Models. In *Network and Distributed System Security Symposium (NDSS)*. ISOC, 2025.

[12] Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. Flocks of Stochastic Parrots: Differentially Private Prompt Learning for Large Language Models. *CoRR abs/2305.15594*, 2023.

[13] Bradley Efron and Trevor Hastie. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Cambridge University Press, 2021.

[14] ESEC/FSE. Call for Artifact Evaluation. http://2011.esec-fse.org/cfp-artifact-evaluation, 2017.

[15] Jonathan Evertz, Niklas Risse, Nicolai Neuer, Andreas Müller, Philipp Normann, Gaetano Sapia, Srishti Gupta, David Pape, Soumya Shaw, Devansh Srivastav, Christian Wressnegger, Erwin Quiring, Thorsten Eisenhofer, Daniel Arp, and Lea Schönherr. Chasing Shadows: Pitfalls in LLM Security Research. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2026.

[16] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. Guidelines for Including Grey Literature and Conducting Multivocal Literature Reviews in Software Engineering. *Information and Software Technology*, 2019.

[17] Xiangming Gu, Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Ye Wang, Jing Jiang, and Min Lin. Agent Smith: A Single Image Can Jailbreak One Million Multimodal LLM Agents Exponentially Fast. In *International Conference on Machine Learning (ICML)*. PMLR, 2024.

[18] Jingxuan He and Martin Vechev. Large Language Models for Code: Security Hardening and Adversarial Testing. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1865–1879. ACM, 2023.

[19] Yu He, Boheng Li, Liu Liu, Zhongjie Ba, Wei Dong, Yiming Li, Zhan Qin, Kui Ren, and Chun Chen. Towards Label-Only Membership Inference Attack against Pre-Trained Large Language Models. In *USENIX Security Symposium (USENIX Security)*, pages 1609–1628. USENIX, 2025.

[20] Yujin Huang, Terry Yue Zhuo, Qiongkai Xu, Han Hu, Xingliang Yuan, and Chunyang Chen. Training-free Lexical Backdoor Attacks on Language Models. In *The Web Conference (WWW)*, pages 2198–2208. ACM, 2023.

[21] IEEE. DSN-2026: Research Track Call For Contributions. https://dsn2026.github.io/cfpapers.html, 2026.

[22] IEEE. IEEE European S&P'26 Call for Papers. https://eurosp2026.ieee-security.org/cfp.html, 2026.

[23] IEEE. IEEE S&P'26 Call for Artifacts. https://sp2026.ieee-security.org/cfartifacts.html, 2026.

[24] IEEE. IEEE S&P'26 Call for Papers. https://sp2026.ieee-security.org/cfpapers.html, 2026.

[25] Andrey Labunets, Nishit V. Pandya, Ashish Hooda, Xiaohan Fu, and Earlence Fernandes. Fun-tuning: Characterizing the Vulnerability of Proprietary LLMs to Optimization-based Prompt Injection Attacks via the Fine-Tuning Interface. In *IEEE Symposium on Security and Privacy (S&P)*, pages 411–429. IEEE, 2025.

[26] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2023.

[27] Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership Inference Attacks against Language Models via Neighbourhood Comparison. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 11330–11343. ACL, 2023.

[28] Mary L McHugh. Interrater Reliability: the Kappa Statistic. *Biochemia Medica*, 2012.

[29] Matthieu Meeus, Shubham Jain, Marek Rei, and Yves-Alexandre de Montjoye. Did the Neurons Read your Book? Document-level Membership Inference for Large Language Models. In *USENIX Security Symposium (USENIX Security)*. USENIX, 2024.

[30] Mahmoud Nazzal, Issa Khalil, Abdallah Khreishah, and NhatHai Phan. Promsec: Prompt Optimization for Secure Generation of Functional Source Code with Large Language Models (LLMs). In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2266–2280. ACM, 2024.

[31] NDSS. NDSS Symposium 2026 Call for Artifacts. https://www.ndss-symposium.org/ndss2026/submissions/call-for-artifacts/, 2026.

[32] Guevara Noubir. Reproducibility in Wireless Experimentation: Need, Challenges, and Approaches. In *Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization (WiNTECH)*, pages 1–1. ACM, 2016.

[33] Daniel Olszewski, Allison Lu, Carson Stillman, Kevin Warren, Cole Kitroser, Alejandro Pascual, Divyajyoti Ukirde, Kevin Butler, and Patrick Traynor. "Get in Researchers; We're Measuring Reproducibility": A Reproducibility Study of Machine Learning Papers in Tier 1 Security Conferences. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 3433–3459. ACM, 2023.

[34] Daniel Olszewski, Tyler Tucker, Kevin R. B. Butler, and Patrick Traynor. SoK: Towards a Unified Approach to Applied Replicability for Computer Security. In *USENIX Security Symposium (USENIX Security)*, pages 469–488. USENIX, 2025.

[35] OpenAI. Introducing ChatGPT. https://openai.com/index/chatgpt/, 2022.

[36] Ashwinee Panda, Tong Wu, Jiachen T. Wang, and Prateek Mittal. Differentially Private In-Context Learning. *CoRR abs/2305.01639*, 2023.

[37] Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 2:1–2:22. ACM, 2023.

[38] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In *IEEE Symposium on Security and Privacy (S&P)*, pages 754–768. IEEE, 2022.

[39] Joshua R. Polanin, Terri D. Pigott, Dorothy L. Espelage, and Jennifer K. Grotpeter. Best Practice Guidelines for Abstract Screening Large-evidence Systematic Reviews and Meta-analyses. *Research Synthesis Methods*, 2019.

[40] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In *International Conference on Learning Representations (ICLR)*, 2024.

[41] ACM SIGSAC. ACM CCS '26 Call for papers. https://www.sigsac.org/ccs/CCS2026/call-for/call-for-papers.html, 2026.

[42] USENIX Security Symposium. USENIX Security '25 Call for Papers. https://www.usenix.org/conference/usenixsecurity25/call-for-papers, 2025.

[43] USENIX Security Symposium. USENIX Security '26 Call for Papers. https://www.usenix.org/conference/usenixsecurity26/call-for-papers, 2026.

[44] Xinyu Tang, Richard Shin, Huseyin A. Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. Privacy-Preserving In-Context Learning with Differentially Private Few-Shot Generation. *CoRR abs/2309.11765*, 2023.

[45] Saiteja Utpala, Sara Hooker, and Pin-Yu Chen. Locally Differentially Private Document Generation Using Zero Shot Prompting. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 8442–8457. ACL, 2023.

[46] Jan Vitek and Tomas Kalibera. Repeatability, Reproducibility, and Rigor in Systems Research. In *International Conference on Embedded Software (EMSOFT)*, pages 33–38. ACM, 2011.

[47] Yue Wang, Hung Le, Akhilesh Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. CodeT5+: Open Code Large Language Models for Code Understanding and Generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1069–1088. ACL, 2023.

[48] Rui Wen, Zheng Li, Michael Backes, and Yang Zhang. Membership Inference Attacks Against In-Context Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2024.

[49] Guanlong Wu, Zheng Zhang, Yao Zhang, Weili Wang, Jianyu Niu, Ye Wu, and Yinqian Zhang. I Know What You Asked: Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving. In *Network and Distributed System Security Symposium (NDSS)*. ISOC, 2025.

[50] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. SafeDecoding: Defending against Jailbreak Attacks via Safety-Aware Decoding. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5587–5605. ACL, 2024.

[51] Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. A Comprehensive Study of Jailbreak Attack versus Defense for Large Language Models. *CoRR abs/2402.13457*, 2024.

[52] Nan Yan, Yuqing Li, Xiong Wang, Jing Chen, Kun He, and Bo Li. {EmbedX}: Embedding-Based Cross-Trigger Backdoor Attack Against Large Language Models. In *USENIX Security Symposium (USENIX Security)*, pages 241–257. USENIX, 2025.

[53] Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models. *CoRR abs/2312.14197*, 2023.

[54] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts. *CoRR abs/2309.10253*, 2023.

[55] Yaman Yu, Yiren Liu, Yuqi Zhang, Yun Huang, and Yang Wang. YouthSafe: A Youth-Centric Safety Benchmark and Safeguard Model for Large Language Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 4349–4363. ACM, 2025.

[56] Zhiyuan Yu, Yuhao Wu, Ning Zhang, Chenguang Wang, Yevgeniy Vorobeychik, and Chaowei Xiao. CodeIPPrompt: Intellectual Property Infringement Assessment of Code Language Models. In *International Conference on Machine Learning (ICML)*. JMLR, 2023.

[57] Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. AutoDefense: Multi-Agent LLM Defense against Jailbreak Attacks. In *NeurIPS Safe Generative AI Workshop (SafeGenAi)*, 2024.

[58] Shengfang Zhai, Yinpeng Dong, Qingni Shen, Shi Pu, Yuejian Fang, and Hang Su. Text-to-Image Diffusion Models can be Easily Backdoored through Multimodal Data Poisoning. In *ACM International Conference on Multimedia (MM)*, pages 1577–1587. ACM, 2023.

[59] Zhexin Zhang, Leqi Lei, Lindong Wu, Rui Sun, Yongkang Huang, Chong Long, Xiao Liu, Xuanyu Lei, Jie Tang, and Minlie Huang. SafetyBench: Evaluating the Safety of Large Language Models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 15537–15553. ACL, 2024.

[60] Shuai Zhao, Jinming Wen, Anh Tuan Luu, Junbo Zhao, and Jie Fu. Prompt as Triggers for Backdoor Attack: Examining the Vulnerability in Language Models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 12303–12317. ACL, 2023.

[61] Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. Provable Robust Watermarking for AI-Generated Text. In *International Conference on Learning Representations (ICLR)*. ICLR, 2024.

Appendix A. Open Science

The materials supporting this study are publicly available at https://github.com/TrustAIRLab/LLM_Reproducibility. The repository contains the full list of the 106 papers included in our study, together with publicly available artifact/repository links where available. It also provides an anonymized artifact-level dataset covering the 84 evaluated artifacts, including reproduction outcomes, failure classifications, and checklist scores.

Appendix B. Compare with Close Works

Research Objective. Evertz et al. [15] address a validity question: do LLM security papers contain methodological pitfalls that could undermine their claims? We instead address a reproducibility question: can the empirical results reported in LLM security papers be independently verified today? These questions are related but distinct. A study may be methodologically sound yet fail to reproduce. Conversely, reproducible results may still be affected by methodological limitations.

Methodology. Evertz et al. [15] conduct a static, review-based study of pitfall prevalence, complemented by four targeted case studies intended to illustrate pitfalls rather than reproduce original experimental results. Their analysis relies on 15 researchers labeling pitfall presence across 72 papers. We instead perform direct artifact reproduction, executing 84 artifacts end-to-end under a standardized, time-bounded protocol and comparing observed outcomes against reported claims in the original papers. This distinction is important. While Evertz et al. [15] demonstrate that methodological pitfalls can affect results under controlled conditions, we evaluate whether published results hold under independent execution of the original artifacts.

Scope. Evertz et al. [15] cover 72 papers across eight venues spanning software engineering (ICSE, FSE, ISSTA, ASE) and security (CCS, USENIX Security, IEEE S&P, NDSS).

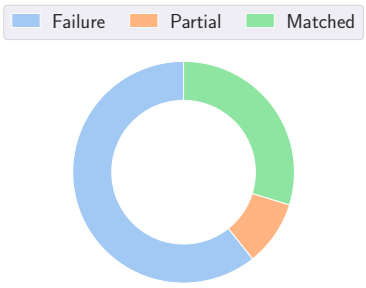


Figure 7: Overall distribution of reproduction outcomes: 60.70% failure, 9.52% partial, 29.76% matched. Papers are categorized as "failure", "partial" (code runs but reported results deviate from the original paper), or "matched" (reproduced results closely match the original results).

Our study focuses exclusively on LLM attack and defense papers from the four major security conferences (CCS, USENIX Security, IEEE S&P, NDSS). Our scope enables descriptive comparison across venues and AE participation within the security community.

LLM-Specific Failure Modes. Evertz et al. [15] identify pitfalls that threaten validity at publication time, including prompt sensitivity, model ambiguity, and data leakage. We study a complementary class of failure modes that emerge after publication and directly affect reproducibility. These include post-publication safety patches that invalidate previously effective attacks, silent API or model deprecations that break execution pipelines, and prompt or decoding sensitivity that causes result drift without any code changes, among others. Importantly, these failures can arise in methodologically sound papers since they reflect properties of the evolving LLM ecosystem rather than flaws in individual research methodology.

Appendix C. Additional Results

This appendix provides additional analyses of reproduction outcomes, including the overall outcome distribution (Figure 7), security-category reproducibility (Table 5), category-level failure modes (Table 6), resource-intensive training cases (Table 7), and paper-artifact consistency (Table 8).

Appendix D. Details for Reproducibility Checklist

To standardize the recording of reproduction attempts across evaluated artifacts, we use a structured checklist that is completed during each reproduction process. The checklist is applied uniformly by multiple domain experts to document observable properties, execution outcomes, and failure conditions encountered during reproduction. Below,

TABLE 5: Reproduction outcomes across ten LLM-security categories. For each category, we report the total number of evaluated artifacts, the number successfully reproduced, the number failed, and the corresponding reproduction rate. "Other" groups heterogeneous, low-frequency topics such as hate speech, prompt stealing, content alignment, and LLM-assisted security applications.

Category	#	Failed	Reproduced	Rate
Jailbreak	14	9	5	35.71%
Adversarial prompt attack	4	4	0	0.00%
Backdoor	5	2	3	60.00%
Data poisoning	6	4	2	33.33%
Evasion of guardrails	4	2	2	50.00%
Hallucination	3	2	1	33.33%
Prompt injection	10	9	1	10.00%
Memorization	14	8	6	42.86%
Model extraction	5	2	3	60.00%
Other	19	9	10	52.63%

TABLE 6: Most frequent non-exclusive failure modes across LLM-security categories. For each category, we report the two most frequent failure modes, retaining ties. A failed artifact may exhibit multiple failure modes.

Category	Failed N	Most frequent failure modes
Jailbreak	9	Irreproducible evaluation (4) Model/API drift (3) Repository incomplete (3)
Adversarial prompt attack	4	Excessive runtime (2) Documentation deficiency (2) Dataset issue (2)
Backdoor	2	Code fails to run (2) Documentation deficiency (2)
Data poisoning	4	Code fails to run (2) Environment setup fails (2) Platform incompatibility (2)
Evasion of guardrails	2	Code fails to run (2) Environment setup fails (2)
Hallucination	2	Environment setup fails (2) Documentation deficiency (2)
Prompt injection	9	Code fails to run (3) Dataset issue (3)
Memorization	8	Environment setup fails (5) File or path issues (3)
Model extraction	2	Code fails to run (2) Model issue (2)
Other	9	Environment setup fails (5) GPU out-of-memory or crash (3) Model issue (3)

TABLE 7: Resource-intensive training cases classified as "Training cost too high," with the corresponding training requirements and observed or estimated resource burden.

Case	Training requirement	Evidence of resource burden
1	Train from scratch for 300 epochs samples	Estimated runtime $\geq$100 h; No pretrained model released
2	LLaVA instruction tuning	Observed runtime $\geq$24 h
3	Train model from scratch	No trained model released; Full training required before evaluation
4	Iterative training over 10 rounds with 800 samples	10 training iterations x 800 samples

we describe the checklist items and how each field was recorded during evaluation.

Artifact Identification. For each paper, evaluators record basic identifying information, including

- Title of the paper.
- Public link to the paper PDF.
- Public links to code artifacts, such as GitHub, Zenodo, and Hugging Face.
- Paper category (attack or defense) and finer-grained security types (e.g., jailbreak, watermarking).

This information is recorded for traceability and bookkeeping purposes.

Code Availability and Execution Protocol. To assess whether reproduction is feasible in principle, experts record the following binary indicators:

- Code acquisition. Whether the implementation code required for reproduction is publicly accessible and could be downloaded (yes/no).
- Execution protocol availability. Whether the artifact provides an explicit execution protocol, like README, that can be followed (yes/no).

If either the code or an executable protocol is absent at the time of reproduction, this should be recorded as "no."

Model Dependencies. To characterize the dependency surface of each artifact, evaluators recorded:

- Base models, including open-source LLMs, like LLaMA-family models, and GPT-family APIs.
- Additional model dependencies, including other closed-source LLMs, like the Claude-family APIs.

This information is used to contextualize reproduction difficulty, particularly with respect to gated models, deprecated APIs, or unavailable checkpoints.

Resource Requirements and Time Budget. To standardize reproduction efforts and ensure comparability across artifacts, evaluators recorded the following under a fixed resource budget:

- Setup time: Maximum of 2 hours for environment configuration and dependency installation.
- Debugging time: Maximum of 2 hours for resolving installation or runtime errors using the provided documentation.
- Execution time: Maximum of 12 hours to complete the primary experiment or evaluation.

Artifacts exceeding these limits without producing results are marked accordingly.

TABLE 8: Paper–artifact consistency across 24 stratified paper–artifact pairs. Four annotators evaluated 11 predefined consistency items per pair (κ = 0.77). The table reports discrepancies for failed and successful reproductions.

Consistency item	Failed (n=12)	Successful (n=12)
D1 Model version	2/12 (16.67%)	3/12 (25.00%)
D2 Inference/invocation parameters	5/12 (41.67%)	1/12 (8.33%)
D3 Model scope	2/12 (16.67%)	3/12 (25.00%)
D4 Checkpoint	**6/12 (50.00%)**	**0/12 (0.00%)**
D5 Dataset identity	0/12 (0.00%)	2/12 (16.67%)
D6 Dataset size/split	3/12 (25.00%)	0/12 (0.00%)
D7 Preprocessing	2/12 (16.67%)	1/12 (8.33%)
D8 Evaluation metrics	1/12 (8.33%)	3/12 (25.00%)
D9 Judge prompt	1/12 (8.33%)	1/12 (8.33%)
D10 Prompt template	2/12 (16.67%)	4/12 (33.33%)
D11 Hyperparameters	5/12 (41.67%)	4/12 (33.33%)

Failure Classification. When a reproduction attempt does not complete successfully, evaluators assign one or more failure classifications from a predefined taxonomy:

- Non-functional (setup $>$ 2 hours). The artifact cannot be set up within the 2-hour setup budget due to installation, dependency, or environment issues. Non-functional (debugging $>$ 2 hours). Execution has failed, and errors cannot be resolved within the 2-hour debugging budget.
- Timeout (execution $>$ 12 hours). Execution has not completed within the 12-hour time limit or has become stalled.
- Other: failures not covered by the above categories. A brief explanation is required in the comments field.

A single artifact can be assigned multiple failure classifications. This taxonomy underlies the empirical failure analysis in Section 5.

Reproduction Results and Recording. For artifacts that have executed successfully, evaluators recorded both reproduced results and the corresponding results reported in the original paper for direct comparison.

- Reproduced attack or defense success rate, obtained by running the released artifact under the specified execution protocol.
- Official attack or defense success rate, as reported in the original paper.
- Additional performance metrics, when applicable, such as attack efficiency, query counts, or other resource-related measures reported by the authors.

These values were recorded verbatim to support subsequent aggregation and comparison across artifacts. The comparison is displayed in Section 4 and Section 6.

Documentation and Artifact Quality Indicators. To capture factors related to practical reproducibility, evaluators assign ordinal scores to the following checklist items based on predefined criteria. Each item is scored on a discrete 1–10 scale.

- README quality. Evaluates the completeness and clarity of the provided documentation, including environment requirements, target descriptions, and execution instructions needed for independent reproduction.
- Out-of-the-box usability. Assesses the extent to which the artifact can be installed and executed by following the provided instructions, considering dependency completeness, installation difficulty, and required manual intervention.
- Result matching. Measures the degree to which reproduced results align with the results reported in the original paper, considering both quantitative success rates and qualitative similarity of outputs.
- Experimental setup determinism. Evaluates whether the artifact explicitly controls sources of randomness, such as random seeds, data splits, and model initialization, to support consistent and repeatable execution.

These scores are recorded for each artifact and later thresholded to derive the binary indicators used in our analysis. We thresholded ordinal scores at $\geq$8 to distinguish artifacts that meet basic reproducibility requirements from those exhibiting strong reproducibility characteristics, consistent with the predefined qualitative anchors of the checklist scale. These thresholds are determined prior to analysis and applied uniformly across all artifacts.

Scoring Anchors. The 1-10 checklist scores use predefined anchors. For README quality and out-of-box usability, scores of 1, 3, 6, 8, and 10 correspond to terrible, poor, average, good, and excellent quality levels, respectively, based on environment information, dependency completeness, data accessibility, and execution guidance. For result matching, scores 10, 8, 6, 3, and 1 correspond to result discrepancies of $\leq$5%, $\leq$10%, $\leq$30%, approximately 50%, and $>$50% or no comparable reproduced result, respectively. Experimental setup determinism uses the same qualitative anchors, ranging from an unreproducible setup with uncontrolled conditions (1) to well-defined splits, comprehensive seed control, and consistent configurations (10).

Appendix E. Statistical Testing

For each criterion, we form a 2×2 contingency table (successful vs. failed $\times$ criterion satisfied vs. not satisfied) using the binarized checklist indicator (score ≥ 8). We apply two-sided Fisher's exact tests and correct for multiple comparisons across the evaluated criteria using the Benjamini-Hochberg procedure; we report the resulting adjusted p-values as q-values. Effect sizes are reported as odds ratios (OR) with 95% confidence intervals; when a cell count is zero, OR/CIs are computed with Haldane-Anscombe correction ($+0.5$ to each cell) [13].

Appendix F. Ethics Considerations

This work is a measurement study of reproducibility in LLM security research. We do not collect data from human subjects and analyze only publicly available papers and artifacts. During reproduction attempts, we follow a responsible reproduction principle: experiments run in controlled research environments, use minimum functionality to evaluate primary claims, and avoid releasing new attack payloads, prompts, or exploit configurations beyond what is already public. When using proprietary model APIs, we follow providers' terms of service and avoid bypassing access controls. We use a time-bounded protocol and report resource constraints explicitly, emphasizing practical reproducibility rather than unconstrained reproduction.

Appendix G. Meta-Review

The following meta-review was prepared by the program committee for the 2027 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

G.1. Summary

The reviewers thank the authors for submitting their paper. The paper presents a large-scale empirical study of reproducibility in LLM security research, examining 106 papers from major security venues (USENIX Security, CCS, IEEE S&P, and NDSS) published between 2023 and 2025 and conducting hands-on reproduction attempts for 84 papers with released artifacts. The main finding is that only a minority of evaluated papers could be reproduced under the authors' protocol, with failures often attributable to factors particularly relevant in LLM-based research, including API evolution, model drift, dependency issues, and computational constraints. The study further proposes recommendations for improving reproducibility practices in this rapidly evolving area.

G.2. Scientific Contributions

- Addresses a Long-Known Issue
- Provides a Valuable Step Forward in an Established Field
- Establishes a New Research Direction
- Independent Confirmation of Important Results with Limited Prior Research

G.3. Reasons for Acceptance

1) The paper addresses a long-known issue and provides a valuable step forward in an established field by empirically demonstrating serious challenges with reproducibility among artifacts from top security research venues. This constitutes a significant empirical effort and contribution and offers practical recommendations likely to benefit the broader research community.
2) The paper establishes a new research direction by identifying key LLM-specific barriers to reproducibility, such as model drift, API evolution, dependency issues, and computational costs.
3) The paper provides independent confirmation that just because artifacts can be executed, it does not imply reproducibility of the original scientific claims.

G.4. Noteworthy Concerns

1) The fixed reproduction budget may inhibit reproducibility for LLM-centric artifacts that require substantial computation, long execution times, or complex environments, potentially resulting in underestimation of reproducibility. Section 8 discusses this specific concern.
2) Some conclusions (e.g., comparisons across conferences and artifact evaluation (AE) status) are drawn from relatively small subgroup analyses. These analyses can reveal interesting trends, but should be interpreted cautiously when drawing conclusions. Section 8 discusses this specific concern.