Supporting Artifact Evaluation with LLMs: A Study with Published Security Research Papers

David Heye*,†, Karl Kindermann*,†, Robin Decker*,†, Johannes Lohmöller*,
Anastasiia Belova‡, Sandra Geisler‡, Klaus Wehrle*, Jan Pennekamp*
*Communication and Distributed Systems, RWTH Aachen University, Germany · {lastname}@comsys.rwth-aachen.de
‡Data Stream Management and Analysis, RWTH Aachen University, Germany · {lastname}@dbis.rwth-aachen.de

***Abstract*—Artifact Evaluation (AE) is essential for ensuring the transparency and reliability of research, closing the gap between exploratory work and real-world deployment is particularly important in cybersecurity, particularly in IoT and CPSs, where large-scale, heterogeneous, and privacy-sensitive data meet safety-critical actuation. Yet, manual reproducibility checks are time-consuming and do not scale with growing submission volumes. In this work, we demonstrate that Large Language Models (LLMs) can provide powerful support for AE tasks: (i) text-based reproducibility rating, (ii) autonomous sandboxed execution environment preparation, and (iii) assessment of methodological pitfalls. Our reproducibility-assessment toolkit yields an accuracy of over 72% and autonomously sets up execution environments for 28% of runnable cybersecurity artifacts. Our automated pitfall assessment detects seven prevalent pitfalls with high accuracy ($F_1 > 92\%$). Hence, the toolkit significantly reduces reviewer effort and, when integrated into established AE processes, could incentivize authors to submit higher-quality and more reproducible artifacts. IoT, CPS, and cybersecurity conferences and workshops may integrate the toolkit into their peer-review processes to support reviewers' decisions on awarding artifact badges, improving the overall sustainability of the process.**

***Index Terms*—artificial intelligence; artifact badges; sustainability; large language models**

I. Introduction

The rapid evolution of cyber threats poses a significant challenge for maintaining resilience across many networked domains, including Internet of Things (IoT) and Cyber-Physical System (CPS) deployments, industrial control systems, connected vehicles, and smart cities. Recent reports by the World Economic Forum [1] and the European Union Agency for Cybersecurity [2] demonstrate that adversaries not only refine established attack vectors but also exploit emerging technologies, particularly Artificial Intelligence (AI), to evade traditional defenses. In response, the volume and complexity of security research have grown rapidly [3]. Yet a critical gap persists between proof-of-concept prototypes for research evaluation and solutions that are mature and robust enough for real-world deployment [3]. This gap undermines confidence in published results and obstructs the translation of academic advances into practical solutions.

To foster trust and accelerate the technological transfer, the academic community increasingly adopts reproducibility badges and performs Artifact Evaluation (AE) within the peer-review process [3]–[5]. These processes require authors to submit code, data, and instructions, which independent reviewers use to verify claimed results. However, AE is labor-intensive, depends on volunteers with specialized expertise, and struggles to keep pace with the rising submission rate in cybersecurity conferences and workshops [3].

Large Language Models (LLMs) have demonstrated remarkable natural language understanding, code synthesis, and knowledge extraction capabilities. In cybersecurity contexts, LLMs have been applied to intrusion and anomaly detection [6], secure coding assistance [7], and automated penetration testing [8]. Simultaneously, some researchers are looking into improving conventional peer review with LLMs [9]–[15]. In this paper, we explore a new dimension of their utility: supporting and automating AE for cybersecurity research. In light of the growing number of submissions at cybersecurity venues, we aim to provide automated support for reviewers of scientific contributions to improve the scalability of AE. We introduce an LLM-driven toolkit that analyzes paper texts and accompanying artifacts to (i) extract reproducibility indicators, (ii) detect potential inconsistencies between claims and submitted artifacts, and (iii) identify common pitfalls in experimental design and evaluation. By embedding these capabilities into the peer-review workflow, we aim to improve both the scalability and consistency of the AE process.

Contributions. We propose a three-step LLM-driven toolkit to partially automate reproducibility assessments:

► RATE: Our LLM-based method that conceptualizes reproducibility via concept vectors extracted from the model's hidden states achieves a recall of almost 95%, allowing the automatic discarding of non-reproducible submissions.

► PREPARE: Our LLM-agent framework automatically sets up and runs submitted artifacts in sandboxed environments, preparing nearly 30% of manually reproducible submissions and offering supporting hints for all others.

► ASSESS: By repurposing the concept of the RATE stage, we reliably identify many design and evaluation pitfalls in security contributions, with an accuracy of >90%.

► *Integrated pipeline*: A combination of these stages into a unified AE workflow, which balances computational cost with assessment accuracy, correctly classifies more than 72% of the papers in our dataset regarding their reproducibility.

Open Science. We have published our code on GitHub [16].

† These authors contributed equally to this work.

Organization. The remainder of this paper is structured as follows. Section II provides foundations and introduces recent works on reproducibility and relevant AI techniques, particularly focusing on cybersecurity. Section III details our three-stage LLM-driven pipeline. Section IV describes our implementation and empirical evaluation on a curated dataset of hundreds of security research papers. Section V discusses findings, lessons learned, and directions for future research before we conclude in Section VI.

II. Background and Related Work

AE at security conferences has become vital for ensuring the transparency and reliability of research, fostering a collaborative environment among researchers and experts. Cybersecurity presents unique challenges for AE, often involving rapidly evolving threats, adversarial settings, and complex interactions. Manual AE struggles to scale with growing submission volumes, complex software-hardware stacks, and deeper methodological flaws. Advancements in AI, particularly concerning LLMs, offer promising solutions to automate and enhance certain tasks in this field. In this section, we review current AE practices and their scalability challenges (Section II-A), common pitfalls in AI-driven cybersecurity research (Section II-B), and emerging AI-based automation techniques targeted toward AI and cybersecurity (Section II-C).

A. Artifact Evaluation at Security Conferences

Many top-tier cybersecurity conferences have introduced (currently non-mandatory) AE into their peer-review process. AE requires authors to submit the code, datasets, and documentation (typically including a Readme with setup and execution instructions) for independent reviewers to verify computational reproducibility. Consequently, the reviewers can award reproducibility badges if the code is available, runnable, or provides the claimed results [3], [17].

Several papers point out the importance of artifacts and their evaluation in computer science [18]–[26], and for cybersecurity in particular [27]–[29]. This process promotes transparency, encourages best practices in experiment reporting, and accelerates the adoption of the research code in the community and potentially in production environments [3]. Despite numerous attempts to formalize the requirements for experiment reporting and implementation description [18], [30], [31], many researchers emphasize various challenges with reproducing the results [3], [19], [24], [32], [33].

Despite these benefits, AE often demands extensive manual effort and expertise, especially given the growing number of submissions to cybersecurity conferences [3] and conferences in general [34]. Reviewers must resolve complex dependencies and sometimes accommodate specialized hardware requirements [3]. Double-blind reviewing intensifies these challenges when anonymization forces the removal of identifying parts of the original code or documentation [32].

Further analyses reinforce these difficulties: Liu et al. [19] examine 2196 papers and 1487 corresponding artifacts submitted between 2017 and 2022 to software engineering venues and find no significant improvement in overall artifact quality, noting in particular that the provided Readme files often lack clear instructions and examples. Olszewski et al. [3] systematically inspect 744 AI-focused submissions at top security conferences and find that only 298 include artifacts. Out of the available artifacts, only 57% provide setup instructions, and not all of these instructions lead to the successful execution [3].

Complementing the aforementioned study, we focus on exploring how LLMs can reduce the human workload of AE by providing automated support for key steps and comparing our results against this manually established benchmark.

Issue: *Conventional AE processes no longer scale with rising submission rates and the diversity in utilized software and hardware stacks.*

B. Common Pitfalls in Cybersecurity Research

A rigorous review of a research paper should not only reproduce results but also critically examine the underlying methodology for evaluation and design flaws, complementing AE. Arp et al. [35] identify several recurring pitfalls that undermine the scientific validity of cybersecurity submissions. For example, Sampling Bias or Base Rate Fallacy may lead to overfitting on imbalanced data or inflated detection metrics due to unrealistically high attack rates in the evaluation data [27], [35], [36]. Lab-only evaluations restrict experiments to synthetic environments, failing to capture real-world operational networks' diversity and adaptive strategies [35]. Conventional AE, which focuses primarily on repeating an experiment by rerunning code, often misses these deeper, more foundational issues. However, they remain relevant from the artifact contribution to the community.

In this work, we thus examine how LLMs can be used to detect textual indicators of these flaws and how to integrate the detection into a (semi-)automated AE workflow.

Issue: *Detecting methodological flaws in a study is vital to determine its true contribution; however, these flaws are often hard to detect as part of standard reproducibility checks.*

C. AI-Induced Automation Improvements

LLMs have demonstrated strong code understanding, generation, and document analysis capabilities [37]. In cybersecurity, they are already used for vulnerability detection [38], flagging anomalies or intrusions [6], and for guiding fuzzing campaigns and penetration tests [8]. Parallel efforts apply LLMs to peer review: Numerous authors [9]–[15], [39] introduce various techniques to support peer-review processes at academic conferences with LLMs. While their work provides a foundation for future research on automated academic peer-review systems, they note that some challenges such as susceptibility to adversarial inputs or biases must be resolved before the tools can be widely deployed. Regarding reproducibility, Bhaskar [40] introduces an LLM-based tool to identify reproducibility indicators in AI-related papers and their artifacts, achieving better agreement with human judgments when compared to keyword-based approaches.

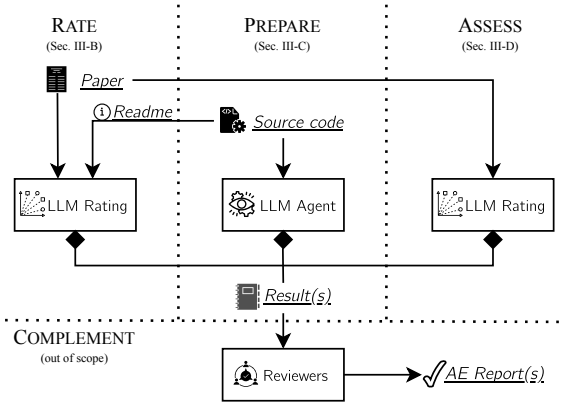

Figure 1. The three pipeline stages require different inputs, and each stage utilizes an LLM. Further, they can be used in combination as desired. Any available results can then be fed into the AE (COMPLEMENT stage).

Despite these advances, a comprehensive automation of AE, including execution environment provisioning, subsequent execution, and detection of methodological pitfalls, remains an open challenge. When complemented with an LLM, such a system can substantially reduce AE experts' manual workload and improve the consistency and reliability of the AE process in cybersecurity research.

Our intuition is that an LLM-driven toolkit that integrates text-based reproducibility screening, automated setup and execution of artifacts, and the detection of common pitfalls may be marketable given the recent advances in AI. Such a system can substantially reduce AE experts' manual workload and improve the consistency and reliability of the AE process in cybersecurity research.

Issue: *The utility of AI for assessing the reproducibility of proposed concepts remains underexplored.*

III. AN LLM-DRIVEN PIPELINE TO AUTOMATE PARTS OF ARTIFACT REPRODUCIBILITY ASSESSMENTS

Having the aforementioned issues in mind and employing recent AI developments, we propose an LLM-driven toolkit that provides automated support for three crucial stages of AE: text-based reproducibility rating (RATE, cf. Section III-B), autonomous execution environment preparation (PREPARE, cf. Section III-C), and methodological-pitfall assessment (ASSESS, cf. Section III-D). Before introducing the design details of the individual steps, we describe how they can be composed into a modular pipeline (Section III-A) to support the manual human peer review.

A. Design Overview

Figure 1 shows the workflow of our pipeline, consisting of three steps that address different parts of AE processes. The steps can be combined as desired for the respective AE process, or they can be used independently. Given this independence, the process can be interrupted at any point, and the generated results can be used or discarded according to the use-case-specific preferences (e.g., to exclude submissions with low reproducibility scores from the review).

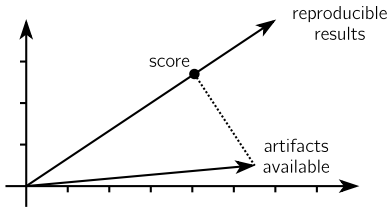

Figure 2. Mapping of concept vectors with a known concept "reproducible results." When measuring a new vector "artifacts available," it is mapped via a projection to the original vector to compute a score.

When using the pipeline in an AE, the process could look as follows: *First*, the RATE stage checks how reproducible the contribution appears based on the paper and the Readme provided along with the source code. If the LLM detects that reproducibility is likely impossible or very challenging, the subsequent stages could, if desired, be canceled.

Second, the PREPARE stage attempts to set up the entire research artifact in a fresh container environment to enable its execution using the provided documentation. The LLM-based agent used in this stage iteratively issues shell commands to clone the repository, install dependencies, and compile and execute code, while parsing the command's outputs in a feedback loop. Suppose that the execution fails and the LLM fails to identify further corrective actions. In that case, the resulting container and a detailed log of commands and errors are archived for further evaluation by an expert, providing them with first insights.

Third, the ASSESS stage focuses on rating the methodological soundness of the submission: Based on the paper submission, it discovers pitfalls that are common in the design and evaluation of contributions in the field. The results could contain valuable insights and can improve the feedback on methodology that reviewers are returning eventually.

Finally, the generated results of all stages, including any created runtime container (PREPARE stage), can be forwarded to the AE reviewers to serve as supplemental material for their "human" expert review. The COMPLEMENT stage is out of scope for this paper, since our goal is to support, streamline, and automate reviewers' work using AI rather than to replace their expert judgment.

B. RATE: Content-Based Reproducibility Ratings

Our toolkit's first step, RATE, quantifies reproducibility as a semantic direction in an LLM's hidden-state space. We adapt Yang et al.'s prior work [41] that extracts concept vectors from LLMs' internal states. By projecting a new text's embedding onto such a concept vector, they quantify how strongly that text represents the respective concept. The authors demonstrate that this approach yields consistent and valid measures for concepts in social science research contexts. In our case, we define the concept as *reproducibility* in cybersecurity research.

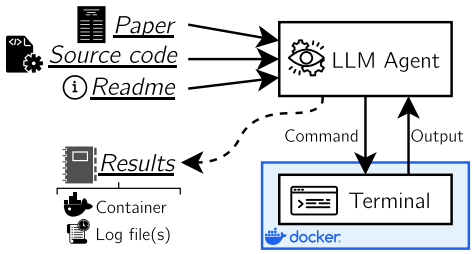

Figure 3. The LLM agent gets access to the paper, the relevant source code and data, as well as a Readme (if available). It then generates commands to execute the code and runs them in a terminal. The outputs are sent back to the agent to determine the next steps.

We begin by crafting two descriptive prompts p^+ and p^- that define the opposite poles of our concept: one characterizing that a paper is "easy to reproduce" and the other describing that a paper is "difficult to reproduce." These prompts instruct the LLM to attend to textual cues such as the clarity of methodological descriptions, the presence and quality of installation and execution instructions, as well as the completeness of supplementary materials.

To extract a reproducibility concept vector, we randomly select a set of n probing texts $t_i, 0 \leq i < n$ and feed each twice into the LLM, once under p^+ and once under p^-. We extract textual cues from each run in the form of embedding vectors from the final layer v_i of the model, yielding pairs (v_i^+, v_i^-). We then compute $v_i^\delta := |v_i^+ - v_i^-|$ for each probe and apply Principal Component Analysis (PCA) to the collection $\{v_i^\delta : 0 \leq i < n\}$. The first principal component serves as our distilled concept vector $\hat{v}$ [41].

To evaluate the reproducibility of a new paper, we obtain its hidden-state embedding v under a neutral prompt and project it onto $\hat{v}$ by computing a dot-product $s := v \cdot \hat{v}/\|\hat{v}\|$. The resulting score s reflects how strongly the paper's text aligns with the distilled reproducibility concept vector constructed from the training dataset. As the method relies only on hidden-state vectors and PCA, it is independent of the specific LLM architecture and can be applied to any model that exposes final-layer embeddings.

C. PREPARE: *Autonomously Setting up Code*

In the PREPARE stage, we deploy an LLM-based agent to automate execution environment setup and code execution within a sandboxed environment, as we detail in Figure 3: Our agent has full access to a shell and is given (i) the paper, (ii) the artifact codebase, and (iii) existing documentation, such as a Readme file. We then prompt it to emit shell commands, which are executed sequentially in a container.

As a first step, we instruct the agent to download any relevant code and datasets required to execute the artifact. After running a command, we capture the output and send it back to the LLM, enabling it to diagnose errors such as missing dependencies, version mismatches, or compilation failures, and to generate follow-up commands to resolve errors. Optionally, the agent may be instructed to output natural-language explanations of each step for human review.

This interactive feedback loop continues until the artifact runs or the agent indicates no further corrective actions are possible. By isolating each artifact in its own container, we ensure (i) reproducibility by starting from a clean system instance, (ii) resource control, such as Graphics Processing Unit (GPU) access, (iii) clean teardown after the execution, and (iv) isolation from other processes running on the host that may otherwise interfere with the execution.

The final deliverable of this stage is either a runnable container image ready for further analysis by an AE expert or a structured error report that pinpoints issues the agent faced. In the latter case, an expert may manually try to fix the detected problems; nonetheless, the LLM agent already completed large parts of trial-and-error setups beforehand.

D. ASSESS: *Identifying Pitfalls in Contributions*

While the previous stages focus on computational reproducibility of the results of a research submission, this stage evaluates the scientific rigor of a submission. Most importantly, it may enhance the quality of reviews issued by AE experts by supporting the detection of otherwise hard-to-notice flaws in the study's methodology and evaluation. We focus on Arp et al.'s [35] taxonomy of ten common pitfalls in AI-driven cybersecurity research; however, our approach is conceptually independent of the specifically analyzed pitfalls. This stage works similarly to the RATE stage by independently extracting a concept vector from the underlying LLM for each of the analyzed pitfalls.

For each of the m analyzed pitfalls, we construct positive and negative prompts that characterize the opposite poles of the respective concept (i.e., pitfall present or pitfall not present). Using the procedure from the RATE stage, we derive a unique concept vector for each pitfall individually using a set of training papers. To assess a new paper, we compute scores $s_i, 0 \leq i < m$ for each pitfall to obtain a feature vector $s := (s_0, \ldots, s_{m-1})$ which we input into a supervised classifier. The classifier outputs which pitfalls are most likely present. The report highlights potential design or evaluation flaws, providing reviewers with insights into the submission's potential methodological strengths and weaknesses.

IV. EVALUATION

To demonstrate the effectiveness of our toolkit on real paper submissions, we measure the accuracy and reliability of our individual steps on two expert-annotated datasets: We employ Olszewski et al.'s [3] dataset of several hundred AI-based cybersecurity papers to benchmark RATE and PREPARE, and Arp et al.'s [35] dataset of 30 papers to assess ASSESS. We begin by introducing the datasets and our experimental setup in Sections IV-A and IV-B, respectively. We then present results for the combined pipeline and its individual components in Sections IV-C and IV-D.

A. Datasets

Reproducibility has no universally accepted quantitative benchmark. To evaluate our pipeline, we, therefore, rely on two expert-annotated datasets: first, Olszewski et al. [3] manually assessed the reproducibility of nearly 750 AI-based security research papers at top-tier conferences. Second, Arp et al. [35] compiled a dataset of 30 papers where they manually record the presence of ten common pitfalls found in studies in cybersecurity. Next, we introduce them and our experimental setup, including the configured LLMs.

1)* OLSZEWSKI-STUDY*: Olszewski et al. [3] invested over eight person-years to manually check the computational reproducibility of artifacts associated with papers on AI in cybersecurity submitted to USENIX Security, ACM CCS, IEEE S&P, and NDSS between 2013 and 2022. They assign discrete reproducibility scores to each submission reflecting, e.g., the effort required to acquire its code and data, to execute its code, and to reproduce the correct results. Furthermore, they document the presence of metadata such as links to code repositories, hyperparameter settings, and dataset splitting.

Most notably, the authors find that out of 744 analyzed submissions, only 298 include artifacts. Of those artifacts, roughly 57% include a Readme file that provides instructions for setting up and executing the corresponding code. The authors only manage to execute 46% of the provided artifacts, while only 20% of the tested code repositories produce the same results as advertised in the original papers.

For our evaluation of RATE and PREPARE, we rely on code repository availability, Readme presence, and manual execution success as ground-truth labels. We only consider the subset of papers where code is available for PREPARE and where, additionally, a Readme is available for RATE.

2)* ARP-STUDY*: Arp et al. [35] manually reviewed 30 papers in cybersecurity submitted to top-tier conferences (2011–2021) to identify ten recurring experimental and design pitfalls. They annotate each paper for the presence or partial presence of each pitfall and whether the authors discuss the flaw in their papers. Notably, they find that sampling bias affects 90% of the analyzed papers, 60% rely on an inappropriate threat model, and that other issues, such as base-rate fallacy and lab-only evaluation scenarios, affect a majority of papers. We rely on the dataset by Arp et al. [35] to evaluate the ASSESS step. While Arp et al. also track whether pitfalls are discussed by authors in the text, we only focus on detecting their presence.

B. Experimental Setup

We now introduce the hardware, models, and procedures used to implement and evaluate our pipeline and its individual stages. All LLM-based components run with fixed prompts and thresholds for binary decisions.

1)* RATE *and* ASSESS*: For both RATE and ASSESS, we run a local instance of LLAMA-3.2-3B-INSTRUCT[1] on a machine equipped with an NVIDIA H-100 Tensor-Core GPU. A prompt template informs the LLM that it has access to the full paper text and, in the case of RATE, a Readme file associated with the submission's code artifact. To derive concept vectors, we fix a random sample of 12 papers for RATE and 10 papers for ASSESS from the respective datasets and run them through the LLM under the positive and negative prompts. The remaining papers form the test set. We compute cutoff scores by optimizing for recall for RATE and via logistic regression for ASSESS.

2)* PREPARE*: Our LLM agent for PREPARE uses OpenAI's GPT-4O-MINI[2] model and interacts with it through the respective web API. Initial experiments with LLAMA-3.2-3B-INSTRUCT reveal that many of the generated commands to run the corresponding artifacts are invalid and that the model quickly runs out of ideas to fix any occurring issues.

For each experiment, the agent spawns a Docker container based on Nvidia's `cuda` image, which in turn uses Ubuntu 22.04 as its base Linux distribution. We host the container on a machine equipped with two Intel Xeon Platinum 8160 CPUs and two NVIDIA Tesla V-100 GPUs. Setting up artifacts that require graphical user interfaces (GUIs) or hardware emulations is, unfortunately, not possible in our setup, leading to failed executions of the corresponding code.

Table I
COMPARISON OF THE OUTPUT OF THE REPRODUCIBILITY PIPELINE WITH THE OLSZEWSKI-STUDY. THE PIPELINE CORRECTLY CLASSIFIES ALMOST THREE-QUARTERS OF THE EXAMINED SUBMISSIONS, PROVIDING EXECUTION ENVIRONMENTS FOR MORE THAN 27% OF ALL SUBMISSIONS MARKED AS RUNNABLE IN THE GROUND-TRUTH.

Total 126		OLSZEWSKI-STUDY runs	OLSZEWSKI-STUDY ¬runs	
PIPELINE	runs	7.14%	8.73%	15.87%
	¬runs	19.05%	65.08%	84.13%
		26.19%	73.81%	

Accuracy: 72.22% Precision: 45.00% Recall: 27.27%

Table II
COMPARISON OF THE OUTPUT OF THE RATE STAGE WITH THE OLSZEWSKI-STUDY. THE APPROACH CORRECTLY CLASSIFIES ALMOST ALL SUBMISSIONS MARKED AS RUNNABLE IN THE GROUND-TRUTH.

Total 130		OLSZEWSKI-STUDY runs	OLSZEWSKI-STUDY ¬runs	
RATE	runs	40.77%	54.62%	95.38%
	¬runs	2.31%	2.31%	4.62%
		43.08%	56.92%	

Accuracy: 43.08% Precision: 42.74% **Recall: 94.64%**

C. Reproducibility Pipeline Evaluation

Table I illustrates the overall performance of our pipeline, i.e., the combination of the RATE and PREPARE stages. We consider the intersection of papers from Olszewski et al.'s [3] dataset in both stages individually. Overall, our system correctly

[1]https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/

[2]https://platform.openai.com/docs/models/o4-mini

Table III
COMPARISON OF THE OUTPUT OF THE PREPARE STAGE WITH THE OLSZEWSKI-STUDY. THE AGENT AUTOMATICALLY SETS UP READY-TO-USE EXECUTION ENVIRONMENTS FOR ALMOST 29% OF ALL SUBMISSIONS MARKED AS RUNNABLE IN THE GROUND-TRUTH.

Total 311		OLSZEWSKI-STUDY		
		runs	¬runs	
PREPARE	runs	7.40%	14.79%	22.19%
	¬runs	18.97%	58.84%	77.81%
		26.37%	73.63%	

Accuracy: **66.24%** Precision: 33.33% Recall: 28.05%

assesses whether an artifact's code can be executed without major effort in more than 72% of cases.

Although only about 7% of *all* attempted artifacts are fully containerized and executed by the pipeline, this performance corresponds to provisioning runnable environments for roughly 28% of the papers that Olszewski et al. [3] manage to execute out of the box, i.e., using only the instructions in the corresponding Readme files. Only about 7% of papers are misclassified as non-runnable when they, in fact, can run out of the box according to Olszewski et al. [3]. These false negatives are often induced by our Docker environment, which cannot emulate special hardware, or by the agent, which cannot fix them without external inputs, e.g., if the link to the sources from the dataset leads to an informative website instead of a Git repository. In any case, the agent provides a reason for failure that human AE experts can use to try to fix the remaining issues manually.

The true negative rate of the pipeline exceeds 85%, meaning that our pipeline reliably filters out non-runnable submissions. These numbers underline that the proposed pipeline can indeed save reviewers from spending valuable time otherwise spent on setting up artifacts using trial-and-error, which is a process that is comparably easy to automate. By prepending the ASSESS stage, submissions deemed unlikely to be reproducible can even be discarded before entering the more costly PREPARE stage, saving valuable computational resources. The results of the pipeline are shared with an AE expert in the COMPLEMENT stage, who can then decide on, e.g., whether to award a reproducibility badge to the given submission.

D. Detailed Evaluation Results

In this section, we give a more detailed overview of the classification results of the individual stages of our pipeline. We highlight how RATE reliably forwards nearly all runnable artifacts to the next stage, how PREPARE autonomously provides numerous execution environments, and how ASSESS detects methodological flaws with high accuracy.

1)* RATE*: This stage aims to find papers whose code is likely not runnable and discard them early, before wasting computational resources and time setting up the code. For the evaluation of this stage, we consider only papers where the code as well as a Readme file are available.

Table II compares this stage's classification results to the OLSZEWSKI-STUDY [3] dataset. The high recall of almost 95% indicates that almost all papers with runnable code are selected to move to the next stage (the false negative rate is just over 6%). In fact, fewer than 3% of all analyzed papers are misclassified as not runnable.

This result makes the step ideal as a first stage for our pipeline: if a paper is deemed not runnable, no computational resources need to be spent to try and execute the respective code. Instead, AE experts are given the result of the stage. If they feel the results can be reproduced after all, the experts can still manually feed the respective code and paper into the PREPARE stage. Given the small number of false negatives, only a few papers are discarded early in the pipeline and not automatically examined for execution.

2)* PREPARE*: Unlike the previous stage, this stage's goal is to automatically set up execution environments to enable experts to quickly run a paper's code and manually assess the validity of the reproduced results. We consider 311 papers from the dataset since our agent does not explicitly require the presence of a Readme file; instead, it can autonomously analyze the code repository structure and, e.g., try to compile and execute relevant files. All papers that have been evaluated in the RATE stage are also analyzed in this stage.

In Table III, we summarize the results of the classification, which show that the agent yields a moderately high accuracy of more than 66%. Notably, this stage alone reliably eliminates the need for experts to manually set up execution environments for papers whose code is not runnable for almost 60% of the analyzed papers. The false negatives are often induced by limitations of our execution environment (cf. Section IV-B): Even though our LLM agent can effectively execute terminal commands, some artifacts require access to graphical desktop environments to, e.g., run Internet browsers, which is out of the scope of our experiments.

3)* ASSESS*: Given the relatively small size of the dataset from the ARP-STUDY [35], i.e., 30 papers, we cannot analyze the pitfalls on *sampling bias* (P1) and *data snooping* (P3). This limitation is due to our training process requiring at least 5 papers per category "pitfall present" and "pitfall not present." For the remaining pitfalls, our evaluation yields promising results. Except for the pitfall on *biased parameters* (P5), the classifier has an accuracy between 90% and 100%. F_1 scores are between 0.92 and 1, and F_2 scores between 0.97 and 1, indicating an accurate response given by our approach. For (P5), our approach performs almost like a random predictor. However, Arp et al. [35] classify most papers as "unclear from text" for this category. We presume that a larger and more representative dataset would fix this problem. The remaining seven pitfalls can be accurately detected using only a small human-annotated dataset. Overall, we conclude that ASSESS is well-suited for detecting common known pitfalls in security-related research papers on AI.

V. Discussion and Future Work

Our results show that an LLM-driven toolkit can reliably filter out non-runnable submissions, autonomously provide execution environments for submitted artifacts, and accurately flag common methodological pitfalls in cybersecurity research. In the following, we discuss our findings in more detail while also addressing limitations of our design, implementation, and evaluation, and proposing directions for future research on the topic. Further, we complement this discussion of findings with a brief overview of lessons learned during our research activities in Section V-C.

A. Individual Findings

Given the overall results of the toolkit, we reflect upon the individual components' strengths and limitations. Furthermore, we outline targeted directions for future enhancements.

1)* RATE*: This stage already yields promising results despite the LLM used for this purpose not being fine-tuned to the given task. Instead, the training data is given to the LLM as a prompt. Future work may evaluate whether fine-tuning an LLM improves the quantification of the concept of artifact reproducibility within the model to generate more precise and consistent concept vectors. However, this change would require a large amount of training data, which is unavailable to us at the time of writing. This training data could, for example, be collected as part of a shadow AE conducted to evaluate the pipeline further, as suggested in Section V-B.

2)* PREPARE*: While this stage automatically creates sandboxed execution environments for many paper artifacts, with the currently used execution environment, we are still unable to handle all submissions correctly. This situation is partly due to technical limitations, e.g., the lack of a desktop environment or specialized hardware required for some evaluations. The former could be solved by adding GUI interaction support to the agent, e.g., using UI-TARS [42]. The latter can be solved by providing a more diverse hardware setup for the stage; this improvement, however, exceeds the scope of this paper, as our goal is to show the general feasibility of the approach.

3)* ASSESS*: Our evaluation of this stage shows that the detection of pitfalls in cybersecurity papers on AI performs very reliably. However, the small size of the evaluation dataset poses limitations to our evaluation. We propose to re-evaluate this stage on a more exhaustive dataset. The creation of such a dataset is, however, infeasible within the scope of this paper.

B. General Findings and Future Directions

Our evaluation shows that our tools, when combined into a pipeline, can provide significant support in the AE process conducted at security conferences. It provides a first step into automating this process by detecting submissions without reproducible artifacts and autonomously preparing their execution to enable AE experts to more quickly assess the validity of the reproduced results. Hence, we provide means to significantly boost the scalability of the process, particularly as we facilitate the tedious task of setting up code environments for performing the evaluations.

1) Open Questions: Despite the potential highlighted in our evaluation, we identify several open questions: (i) Better understanding in detail how "perfect" prompts could look like for the different approaches in our toolkit. (ii) Further comparing different underlying LLMs, as different models may be better in finding and understanding certain concepts or performing certain tasks, in particular, depending on the model size. (iii) Assessing the security risks of applying our pipeline in practice, e.g., regarding the execution of arbitrary code in the artifacts, as well as better understanding the implications for intellectual property fed to closed-source commercial models. Concerning the last question, PREPARE already provides execution environments that are sandboxed in individual Docker containers. However, access to hardware components such as GPUs or other specialized devices may impose additional risks on the system.

2) Integration into Peer-Review: After improving the techniques for automatically assessing artifact reproducibility proposed in this paper, future work may integrate them directly into the review process at cybersecurity conferences. Given more general training data, the process could also be integrated into conferences in other fields. Currently, artifact reproducibility checks are often only performed for accepted papers, i.e., after the review process is completed [4]. Automated reproducibility checks would allow checking a large number of submissions even before issuing an acceptance. While some authors might be concerned with participating in a potentially biased or low-quality AEs, an AI-assisted pipeline may increase their trust in the process and, in turn, improve their willingness to participate. We believe that assessing the usability of the proposed pipeline workflow in the form of shadow AE is a good next step for assessing its maturity.

Integrating our tool into peer review requires addressing manipulation risks, e.g., prompt injection. The attack surface is limited in RATE and ASSESS, as outputs derive from concept vectors extracted from the LLM's hidden states rather than direct generation from paper text. Reviewers read the paper before or in parallel to the partially automated AE, aiding detection of any injections. In PREPARE, injections may occur in Readme files or code comments; however, sandboxed execution and expert assessment of results render their impact negligible. We expect such malicious acts to be rare due to scientific integrity norms and penalized when discovered.

3) Future Evaluations: Finally, future work may expand our research by applying the techniques to papers from different fields. We limit our evaluation to papers in these domains due to the availability of an exhaustive dataset. We believe the approach easily generalizes to other topic areas as it does not directly depend on the contents of the evaluated works. Most importantly, we show that AI is a promising tool to employ in AE with a great potential to complement the process to improve its quality, scalability, and thus sustainability.

C. Lessons Learned

During the development of our pipeline, we noticed several unexpected behaviors across interactive handling, execution,

verification, and contextual reasoning. For example, in the PREPARE stage, the agent reports that it cannot engage with an interactive editor such as `nano` for one experiment. While it could have proposed an alternative non-interactive solution, e.g., using `sed`, the LLM did not have this idea, revealing a gap in its problem-solving repertoire. We suggest evaluating this behavior with more powerful models to assess whether this problem can be resolved.

Furthermore, in one experimental run during our implementation, the agent proposed to comment out an entire program to make it run successfully—returning in an inaccurate assessment. However, this change results in the program not performing any computations or providing any outputs. We worked around this issue by adapting the prompts given to the LLM, highlighting the importance of carefully designed prompts and validation.

In the RATE stage, we notice that the LLM's grasp of the concept of reproducibility is less accurate than that of the different pitfalls analyzed in the ASSESS stage. This result may be induced by the training data of the utilized model: Reproducibility is a niche topic, and today's models are likely not trained on much input that covers this concept.

Simultaneously, even on powerful hardware, local models execute much more slowly than commercial models like GPT-4O-MINI, which are heavily optimized for mass use. In particular, to protect the privacy and confidentiality of submissions during the (confidential) AE process, we propose that future work focuses on optimizing LLMs specifically for the use case of artifact reproducibility assessment.

Overall, we have learned that LLMs constitute a powerful tool that has the potential to substantially complement and improve the AE process at scientific conferences. They can be employed to automate tedious and repetitive tasks while simultaneously streamlining the whole process to help provide more consistent and high-quality feedback to authors.

VI. CONCLUSION

Ensuring the reproducibility of research artifacts in cybersecurity is crucial in science to validate the potential for further use of given experimental and methodological results. It narrows the gap between experiments and simulations, as well as real-world deployments, since stakeholders can better assess the suitability of the approaches for their systems. Currently, some scientific conferences perform time-consuming manual artifact evaluations to assess whether the contributions of the submitted works are reproducible. We propose an LLM-based toolkit that enhances the automation potential of otherwise manual and time-consuming artifact assessments. Our evaluation shows that, when combining the tools into a pipeline, a majority of submissions without runnable artifacts are automatically discarded. At the same time, execution environments are generated for many submissions with runnable code. We propose to integrate such a pipeline into the Artifact Evaluation (AE) process of conferences to incentivize researchers to deliver reproducible results. Furthermore, this change has potential to unburden reviewers by automating a time-consuming part of the review work, with the objective of improving the sustainability of the AE process.

ACKNOWLEDGMENTS

This work was funded by the Federal Ministry of Research, Technology and Space (BMFTR) in Germany under the grant number 16KIS2251 of the SUSTAINET-guardian project. The responsibility for the content of this publication lies with the authors. The authors thank Daniel Arp for supporting the pitfall evaluation (ASSESS stage), which builds upon the survey data by Arp et al. [35].

REFERENCES

[1] World Economic Forum, "Global Cybersecurity Outlook 2025," World Economic Forum, Tech. Rep., 2025.

[2] European Union Agency for Cybersecurity, "ENISA Threat Landscape 2024," European Union Agency for Cybersecurity, Tech. Rep., 2024.

[3] D. Olszewski, A. Lu, C. Stillman *et al.*, ""Get in Researchers; We're Measuring Reproducibility": A Reproducibility Study of Machine Learning Papers in Tier 1 Security Conferences," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM, 2023, pp. 3433–3459.

[4] M. Athanassoulis, P. Triantafillou, R. Appuswamy *et al.*, "Artifacts Availability & Reproducibility (VLDB 2021 Round Table)," *ACM SIGMOD Record*, vol. 51, no. 2, pp. 74–77, 2022.

[5] S. Krishnamurthi, "About Artifact Evaluation," https://artifact-eval.org/about.html, 2014 (accessed November 22, 2025).

[6] H. Zhang, A. Bin Sediq, A. Afana, and M. Erol-Kantarci, "Large Language Models in Wireless Application Design: In-Context Learning-enhanced Automatic Network Intrusion Detection," in *Proceedings of the 2024 IEEE Global Communications Conference (GLOBECOM '24)*. IEEE, 2024, pp. 2479–2484.

[7] G. Sandoval, H. Pearce, T. Nys *et al.*, "Lost at C: A User Study on the Security Implications of Large Language Model Code Assistants," in *Proceedings of the 32nd USENIX Security Symposium (USENIX SEC '23)*. USENIX Association, 2023, pp. 2205–2222.

[8] A. Happe and J. Cito, "Getting pwn'd by AI: Penetration Testing with Large Language Models," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23)*. ACM, 2023, pp. 2082–2086.

[9] L. Cao, L. You, and R&D Team, "CSPaper Review: Fast, Rubric-Faithful Conference Feedback," in *Proceedings of the 18th International Natural Language Generation Conference: System Demonstrations (INLG '25)*. ACL, 2025, pp. 3–7.

[10] I. Kuznetsov, O. M. Afzal, K. Dercksen *et al.*, "What Can Natural Language Processing Do for Peer Review?" arXiv:2405.06563, 2024.

[11] M. Zhu, Y. Weng, L. Yang, and Y. Zhang, "DeepReview: Improving LLM-based Paper Review with Human-like Deep Thinking Process," arXiv:2503.08569, 2025.

[12] R. Ye, X. Pang, J. Chai *et al.*, "Are We There Yet? Revealing the Risks of Utilizing Large Language Models in Scholarly Peer Review," arXiv:2412.01708, 2024.

[13] M. Idahl and Z. Ahmadi, "OpenReviewer: A Specialized Large Language Model for Generating Critical Scientific Paper Reviews," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations) (NAACL '25)*. ACL, 2025, pp. 550–562.

[14] L. Sun, S. Tao, J. Hu, and S. P. Dow, "MetaWriter: Exploring the Potential and Perils of AI Writing Support in Scientific Peer Review," *Proceedings of the ACM on Human-Computer Interaction*, vol. 8, no. CSCW1, 2024.

[15] S. Huang, Q. Wang, W. Lu *et al.*, "PaperEval: A universal, quantitative, and explainable paper evaluation method powered by a multi-agent system," *Information Processing & Management*, vol. 62, no. 6, 2025.

[16] COMSYS, "Artifact Evaluation LLM Support." [Online]. Available: https://github.com/COMSYS/artifact-evaluation-llm-support

[17] ACM, "Artifact Review and Badging - Current," https://www.acm.org/publications/policies/artifact-review-and-badging-current, 2020.

[18] C. S. Timperley, L. Herckis, C. Le Goues, and M. Hilton, "Understanding and Improving Artifact Sharing in Software Engineering Research," *Empirical Software Engineering*, vol. 26, no. 3, 2021.

[19] M. Liu, X. Huang, W. He *et al.*, "Research artifacts in software engineering publications: Status and trends," *Journal of Systems and Software*, vol. 213, 2024.

[20] B. Hermann, S. Winter, and J. Siegmund, "Community Expectations for Research Artifacts and Evaluation Processes," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20)*. ACM, 2020, pp. 469–480.

[21] N. Juristo and S. Vegas, "The role of non-exact replications in software engineering experiments," *Empirical Software Engineering*, vol. 16, no. 3, pp. 295–324, 2011.

[22] O. E. Gundersen and S. Kjensmo, "State of the art: Reproducibility in artificial intelligence," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[23] R. D. Peng, "Reproducible Research in Computational Science," *Science*, vol. 334, no. 6060, pp. 1226–1227, 2011.

[24] S. Winter, C. S. Timperley, B. Hermann *et al.*, "A Retrospective Study of One Decade of Artifact Evaluations," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22)*. ACM, 2022, pp. 145–156.

[25] Q. Scheitle, M. Wählisch, O. Gasser, T. C. Schmidt, and G. Carle, "Towards an Ecosystem for Reproducible Research in Computer Networking," in *Proceedings of the Reproducibility Workshop (Reproducibility '17)*. ACM, 2017, pp. 5–8.

[26] B. Yildiz, H. Hung, J. H. Krijthe *et al.*, "ReproducedPapers.org: Openly Teaching and Structuring Machine Learning Reproducibility," in *Proceedings of the 3rd International Workshop on Reproducible Research in Pattern Recognition (RRPR '21)*, vol. 12636. Springer, 2021, pp. 3–11.

[27] J. Pennekamp, E. Buchholz, M. Dahlmanns *et al.*, "Collaboration is not Evil: A Systematic Look at Security Research for Industrial Use," in *Proceedings of the Workshop on Learning from Authoritative Security Experiment Results (LASER '20)*. ACSAC, 2021.

[28] R. H. Moulton, G. A. McCully, and J. D. Hastings, "Confronting the Reproducibility Crisis: A Case Study of Challenges in Cybersecurity AI," in *Proceedings of the 2024 Cyber Awareness and Research Symposium (CARS '24)*. IEEE, 2024, pp. 1–6.

[29] R. Uetz, C. Hemminghaus, L. Hackländer, P. Schlipper, and M. Henze, "Reproducible and Adaptable Log Data Generation for Sound Cybersecurity Experiments," in *Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC '21)*. ACM, 2021, pp. 690–705.

[30] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard *et al.*, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[31] A. Jedlitschka and D. Pfahl, "Reporting Guidelines for Controlled Experiments in Software Engineering," in *Proceedings of the 2005 International Symposium on Empirical Software Engineering, 2005 (ISESE '05)*. IEEE, 2005, pp. 95–104.

[32] V. Bajpai, M. Kühlewind, J. Ott *et al.*, "Challenges with Reproducibility," in *Proceedings of the Reproducibility Workshop (Reproducibility '17)*. ACM, 2017, pp. 1–4.

[33] C. Collberg, T. Proebsting, and A. M. Warren, "Repeatability and Benefaction in Computer Systems Research: A Study and a Modest Proposal," University of Arizona, Tech. Rep. TR 14-04, 2015.

[34] C. Demetrescu, I. Finocchi, A. Ribichini, and M. Schaerf, "On computer science research and its temporal evolution," *Scientometrics*, vol. 127, no. 8, pp. 4913–4938, 2022.

[35] D. Arp, E. Quiring, F. Pendlebury *et al.*, "Dos and Don'ts of Machine Learning in Computer Security," in *Proceedings of the 31st USENIX Security Symposium (SEC '22)*. USENIX Association, 2022, pp. 3971–3988.

[36] M. A. Lones, "Avoiding Common Machine Learning Pitfalls," *Patterns*, vol. 5, no. 10, 2024.

[37] J. Zhang, H. Bu, H. Wen *et al.*, "When LLMs meet cybersecurity: a systematic literature review," *Cybersecurity*, vol. 8, 2025.

[38] Y. Guo, C. Patsakis, Q. Hu, Q. Tang, and F. Casino, "Outside the Comfort Zone: Analysing LLM Capabilities in Software Vulnerability Detection," in *Proceedings of the 29th European Symposium on Research in Computer Security (ESORICS '24)*, vol. 14982. Springer, 2024, pp. 271–289.

[39] X. Gao, J. Ruan, Z. Zhang *et al.*, "MMReview: A Multidisciplinary and Multimodal Benchmark for LLM-Based Peer Review Automation," arXiv:2508.14146, 2025.

[40] A. Bhaskar and V. Stodden, "Reproscreener: Leveraging LLMs for Assessing Computational Reproducibility of Machine Learning Pipelines," in *Proceedings of the 2nd ACM Conference on Reproducibility and Replicability (REP '24)*. ACM, 2024, pp. 101–109.

[41] Y. Yang, H. Duan, J. Liu, and K. Y. Tam, "LLM-Measure: Generating Valid, Consistent, and Reproducible Text-Based Measures for Social Science Research," arXiv:2409.12722, 2024.

[42] Y. Qin, Y. Ye, J. Fang *et al.*, "UI-TARS: Pioneering Automated GUI Interaction with Native Agents," arXiv:2501.12326, 2025.