Revisiting Security Practices for GitHub Actions Workflows

Jiangnan Huang
Radboud University
Nijmegen, The Netherlands

Bin Lin[†]
Hangzhou Dianzi University
Hangzhou, China

***Abstract*—GitHub Actions, a built-in CI/CD service of GitHub released in 2019, has become one of the most widely adopted tools among developers for automating software development workflows. This popularity, however, brings security challenges, as vulnerable workflows can expose repositories and software supply chains to significant risks. Existing studies have highlighted several types of potential security issues. Over the past few years, GitHub has been constantly promoting better security practices, and developers have gained experience in using GitHub Actions. Investigating how developers' practices for handling GitHub Actions security have changed over time could offer valuable insights for further strengthening the security of these workflows. In this study, we analyzed non-optimal security practices in 18,938 workflows from 5,246 active GitHub repositories. By comparing the prevalence of issues spotted in two different years (2022 and 2024), we find that the instances of *No Permissions Specified* have significantly reduced as more developers now explicitly define permissions in their workflows. However, other issues, such as *Confidential Data Disclosure*, remain prevalent, underscoring the need for continued vigilance and further research in this domain.**

***Index Terms*—GitHub Actions, CI/CD, Software Repositories, Security, Software Vulnerability**

I. Introduction

GitHub [1], one of the largest social coding platforms, hosts more than 420 million repositories and 100 million developers by September 2024 [2]. Over the past decade, developers on GitHub have embraced third-party CI/CD tools (e.g., Travis CI [3], Jenkins [4]) to streamline workflows such as building, testing, and deployment [5]. In response, GitHub introduced GitHub Actions (GHA) [6] in November 2019, a native CI/CD solution that quickly surpassed other CI/CD tools in popularity and dominated the market within only 18 months [7]. Similar to other CI/CD tools, developers can specify automated tasks in GHA workflows through configuration files. Like traditional source code, GHA workflows are also developed and adjusted over the lifespan of a project to meet evolving needs. However, their capability to modify and control repository content makes them attractive targets for potential attacks.

Prior studies have been conducted to address security concerns within GHA workflows. [8]–[10] Koishybayev *et al.* [8] first conducted a systematic security analysis of GHA ecosystem and found security properties that impact workflows in GHA. Meanwhile, Benedetti *et al.* [9] identified four types of security issues in GHA workflows and developed GHAST, a security assessment methodology, which was applied to 50 open-source projects, uncovering over 20,000 issues. More recently, Muralee *et al.* [10] introduced ARGUS, a framework using taint analysis to detect command injection issues. This is the latest large-scale security analysis and used GHA workflows collected in 2022. Together, these research efforts highlight the critical security risks associated with GHA workflows.

Due to widespread security concerns, GitHub has been constantly promoting good security practices for GHA workflows [11]. While previous research has introduced security tools and developers have gained experience with GHA, it remains unclear whether security practices have improved and which issues are still prevalent. In this study, we revisit the security practices for GHA workflows integrating the dataset by Muralee *et al.* [10]. While they only focused on potential command injection issues, we examine multiple types of security issues. More specifically, we compiled a new dataset of 18,938 GHA workflows from 5,246 active repositories, adapted and expanded Benedetti *et al.*'s GHAST [9] to assess security issues in both the old (2022) and latest (2024) versions of these workflows. Understanding the prevalence of security issues over the years and how developers' security practices change can help developers adapt better strategies to safeguard GHA workflows. Our findings reveal that while certain types of issues have been improved, overall security practices remain concerning, with 33,362 issues identified in the current workflows.

II. Study Design

This study aims to investigate the following two research questions (RQs):

- **RQ$_1$ (prevalence)**: *How prevalent are different types of security issues in GHA workflows?* This RQ aims to provide an overview of the prevalence of various security issues in GHA, focusing on identifying and understanding the non-optimal security practices in current workflows.
- **RQ$_2$ (mitigation)**: *To what extent are various types of security issues being mitigated?* After two years since the last major study on security issues in GHA workflows, this RQ aims to assess whether the security practice has been improved. By comparing current security issues with those presented two years ago, we offer insights into how developers dealt with GHA security and which issues received the most attention for mitigation.

[†] Corresponding Author.

TABLE I: Types of Security Vulnerabilities in GHA workflows

Vulnerabilities Issue	**Description**
Confidential Data Disclosure	Confidential data (*e.g.,* API keys, passwords) in `secrets` context are not adequately secured in environment variables, which can lead to exposure in job logs or unintentional transmission to external hosts.
Command Injection	Part of inputs from the actor (*i.e.,* pull request's title) is used directly in the job with `github` context, potentially introducing command injection vulnerabilities if the input includes malicious code.
Misconfigurations Issue	**Description**
Unverified Action Version Unpinned	Third-party actions from unverified creators are used in the workflow and *a)* or *b)*: *a)* The versions are not specified which can lead to run failures or unexpected behavior with updates. *b)* Tags (e.g. "v2") are used to specified the version which allow attackers to exploit tag-reuse, redirecting workflows to malicious actions.
No Permissions Specified	The `permissions` key is not specified in the workflow and/or job(s) to adjust the default permissions of the `GITHUB_TOKEN`. Specifying `permissions` helps restrict the access to the minimum necessary level.

TABLE II: Security levels of the trigger-events

Security level	**E.S.**	**Description**
Restricted	1	The attacker must rely on assistance from a repository owner to trigger the workflow. *e.g.,* Triggering `push` requires the attacker to have maintainer status granted by an owner.
Supervised	2	The attacker must meet specific conditions to trigger the event. *e.g.,* Triggering `pull_request` requires the maintainer to accept it, thereby starting the workflow.
Unsupervised	3	The attacker needs no permissions from repository owners and can trigger the workflow at any time through external actions. *e.g.,* Any GitHub user with repository access can trigger `issues`.

* **E.S.:** Exploitability Score.

A. Data Preparation

To answer our RQs, we collected a large dataset of GitHub repositories via the SEART GitHub Search Engine [12]. To mitigate common threats in mining software repositories [13], the following constraints were applied:

- **Development activity.** We selected projects created before 2023 that were still active (last commit within 1 month) at data collection to ensure sufficient development history for studying GHA workflows, excluding forks to avoid redundant data.
- **Popularity.** We only selected projects with at least 100 stars and 100 commits, with number of forks over 100, to avoid including potentially irrelevant toy projects.

We also excluded projects not using GHA workflows. The final list comprises 19,988 repositories. As we would also like to understand how security practices change, we integrated the data from Muralee et al. [10] (ARGUS dataset), which was collected between November and December 2022. This dataset covers over 1 million unique repositories and 2.7 million GHA workflows and is available upon request for research [1]. We adopted the intersection between our data and theirs as this ensures the repositories containing GHA workflows two years ago without the need for cloning all the repositories and checking out previous versions. After merging by workflow path, we obtained 5,813 unique repositories, encompassing a total of 24,634 GHA workflows. We retrieve all their current workflows through the GitHub Rest API [14]. To ensure a fair comparison of practices over two years, we further excluded newly generated workflows and those no longer available in the current repositories. In the end, we obtained 18,938 GHA workflows from 5,246 repositories.

[1]https://secureci.org/dataset.html

B. GHA Workflow Security Evaluation

We applied enhanced GitHub Actions Security Tester (GHAST) to analyze the security vulnerabilities of the collected GHA workflows for both 2022 and 2024 versions. GHAST is a publicly available tool by Benedetti *et al.* [9]. To align the tool with our study's objectives, several modifications were made, including few logic improvements, information preservation, edge case handling, and bug fixes. The enhanced version of GHAST detects four types of security issues within GHA workflows, including two security *vulnerabilities* and two *misconfigurations*. Table I details the categories of these security issues. Note that the original GHAST detects outdated actions, identifying whether a new release is available for third-party action in use. We, instead, detect *Unverified Action Version Unpinned*, emphasizing the risks posed by unverified (detected through cross-check with GitHub Marketplace profiles as done by Koishybayev *et al.* [8]), unpinned (*i.e.,* action not pinned to a full-length commit SHA) third-party actions. We made this change to the tool as it aligns more closely with the recommended practice by GitHub [15]. Like GHAST, we also classify trigger-events into three security levels (listed in Table II), indicating the complexity an attacker must navigate to activate specific events and initiate workflows.

C. Replication Package

To support replication studies and future extensions, our enhanced GHAST tool is available online at https://github.com/jiangnanpro/Security-of-GHA-workflow. Due to potential security concerns, we do not openly publish the dataset used in the study.

III. Prevalence of Security Issues

To answer RQ1, we analyzed the current version of the 18,938 collected real-world GHA workflows, identifying 33,362 security issues in total, including 2,731 vulnerabilities and 30,631 misconfigurations. Table III (column "# issues 2024") shows the distribution of all four types of identified security issues according to the security level of each affected workflow.

A. Security Vulnerabilities

1) Confidential Data Disclosure: We found 2,617 cases where the confidential data used with `secret` context are not set as an environment variable, enabling jobs to expose those secrets *e.g.,* by printing in a log output or sending them to an external host. Among the discovered issues, 52 of them sit in workflows which can be triggered using unsupervised events, raising the risk of confidential data leakage.

2) Command Injection: We identified 114 issues that can expose workflows to command injection risks, of which only one does not need any granted permission by repository owners (exploitability score of 3) and 35 are under workflow with supervised events (exploitability score of 2). These vulnerabilities enable attackers to inject malicious inputs, triggering direct execution within the GHA Runner and compromising the repository and/or the execution environment.

B. Workflow Misconfigurations

1) Unverified Action Version Unpinned: We identified 16,439 issues attributed to such *actions*, with 677 of these in unsupervised workflows. In general, *actions* from unverified authors carry greater security risks than those from verified sources. When using unverified *actions*, developers are expected to thoroughly audit the source code to ensure there are no security threats. However, if *actions* used are not pinned to specific SHAs, they remain vulnerable even if tagged as latest version or left untagged. This vulnerability stems from the possibility of a malicious update by the unverified creator, which not only compromises the workflow and its repository, but also introduces stability concerns.

2) No Permissions Specified: Our findings indicate that substantial workflows (74.9%, 14,192) omit explicit permission declarations altogether, relying only on the default permissions of `GITHUB_TOKEN`, which could inadvertently grant broader access than intended. Fig. 1 b) shows that 15.5% (2,944) of workflows specify permissions only at the workflow level, rather than assigning them separately for each job. By contrast, only 9.5% (1,802) of workflows follow security guidelines and adhere to the principle of least privilege by declaring permissions at the job level to restrict access effectively [16].

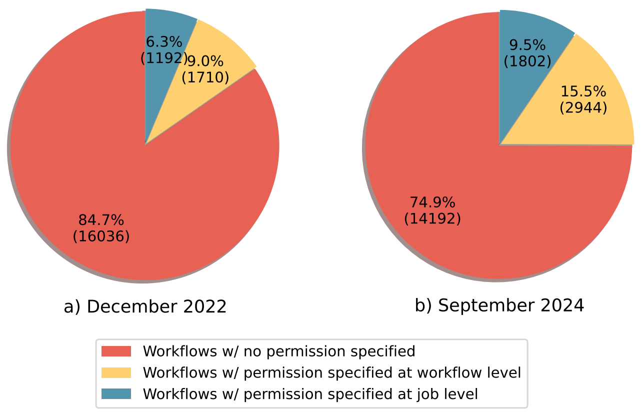

Fig. 1: Application of permission in GHA workflows.

IV. Exploring the Mitigation of Security Issues

To address RQ2, we compared the security issues of the 18,938 GHA workflows across the 2022 and 2024 versions, which is detailed in Table III. The numbers in the 2024 versions that are lower that the ones in 2022 versions are marked with underlines. Overall, the comparison reveals positive progress, with the percentage of issue-free workflows increasing from 10.2% (1,938) to 17.3% (3,283). Additionally, the total number of identified security issues dropped a little, from 34,629 to 33,362. Despite these improvements, security issues continue to be widespread in GHA workflows, highlighting the need for ongoing attention and mitigation efforts.

A. Trigger-events

Between the 2022 and 2024 versions of the GHA workflows, we observe an increase of 93 potentially risky events (Exploitability score > 1). However, with the total number of applied events rising from 32,127 to 33,426, the percentage of risky events has slightly decreased by 1.1%. Overall, no significant change is observed on the usage of risky trigger-events. This result is expected, given that the core functionality of the workflows has largely remained unchanged. As a result, the trigger-events themselves are likely to remain consistent over time, reflecting the established practices and preferences of developers when configuring their workflows.

B. Security Vulnerabilities

As shown in Table III, *Confidential Data Disclosure* vulnerabilities have increased by 272 (from 2,345 to 2,617) in the 2024 version compared to 2022. This increase is observed in workflows with all security levels, indicating a broader exposure to this vulnerability in current workflows. The number of workflows affected has also increased by 17. As for *Command Injection*, the number of occurrences has increased by 12, with the number of affected workflows rising by 15.

In general, the analysis does not reveal statistically significant differences (p-values $\geq$ 0.05) in the number of security vulnerabilities between the two workflow versions, indicating limited progress in mitigating these issues in GHA workflows. Given the potential to compromise repository integrity and expose sensitive data, these issues should be prioritized to capture developers' attention and encourage proactive remediation efforts. In our follow-up research, we aim to explore developers' perspectives on these vulnerabilities to better understand barriers to risk mitigation and identify opportunities for enhancing security practices in GHA workflows.

TABLE III: Distribution of Security Issues in GHA workflows

Security Issues	**# issues 2022**				**# issues 2024**				**# wfs 2022**	**# wfs 2024**	**P-value**
Exploitability score	1	2	3	*All*	1	2	3	*All*	-	-	-
Confidential Data Disclosure	1584	707	54	2345	1786	779	52	2617	1121	1138	0.355
Command Injection	74	28	0	102	78	35	1	114	86	101	0.523
All Vulnerabilities	1658	735	54	2447	1864	814	53	2731	1159	1185	0.320
Unverified Action Version Unpinned	6236	9121	789	16146	5947	9815	677	16439	7929	7500	0.074
No Permissions Specified	6385	8958	693	16036	5522	8081	589	14192	16036	14192	0.0
All Misconfigurations	12621	18079	1482	32182	11469	17896	1266	30631	16920	15526	0.0
All Issues	14279	18814	1536	34629	13333	18710	1319	33362	17000	15655	0.0

* **# wfs:** # GHA workflows affected by this issue; **P-Value** obtained using Wilcoxon's Signed Ranks Test with the pairs of # issues per workflow (2022 vs. 2024).

C. Workflow Misconfigurations

In terms of misconfigurations, some progress has been observed. As shown in Table III, although the number of *Unverified Action Version Unpinned* has increased by 293 (from 16,146 to 16,439), 429 fewer workflows contain this issue, suggesting increased awareness and efforts from the developers to address the misconfiguration. The same trend applies to permission-related misconfigurations: *No permission specified* issues decreased by 1,844 (from 16,036 to 14,192).

Fig. 1 illustrates the use of the `permissions` key across workflows, revealing a broader adoption at both workflow and job level in 2024 compared to 2022. This result indicates the increased attention to configuration security. However, the best practices recommend setting permissions at the job level to follow the least privilege principle [16], indicating the room for improvement despite the positive progress.

Overall, the comparison of the numbers of misconfigurations per workflow across two versions results in a statistically significant difference (p-value < 0.05), which is in line with our observation.

V. Threats to Validity

Threats to internal validity: A key threat to our study is GHAST's reliability in detecting security issues in GHA workflows. While tools like ARGUS specialize in specific issues, we chose GHAST for its broader coverage. To mitigate this threat, GHAST has been improved to align with our research objectives. A manual review of 40 reports generated by the enhanced GHAST confirmed a 100% true positive rate. Our future work will incorporate tools like CodeQL [17] and ARGUS for a more comprehensive security analysis.

Threats to external validity: Another threat is the generalizability of our results due to the selection of studied GHA workflows. To mitigate this, we selected repositories from diverse domains with varying popularity and activity levels. In this study, we only studied open-source projects, and proprietary software might show different patterns. In the future, we plan to conduct a systematic and regular analysis of a broader range of repositories to monitor the evolution of security practices in GHA workflows.

VI. Related Work

Koishybayev *et al.* [8] conducted a systematic security analysis of GHA ecosystem, identifying critical security properties that impact workflows in GHA and other CI platforms. Benedetti *et al.* [9] identified four types of security issues in GHA workflows and developed a security assessment tool GHAST. We improved their tool and conducted an more in-depth analysis of different types of security issues, aligned to the security practices recommended by GitHub. Moreover, we studied the security practice change over last two years. Muralee *et al.* [10] introduced ARGUS, a framework leveraging taint analysis to detect command injection issues in GHA.

Delicheh *et al.* [18] analyzed the JavaScript *actions* used in GHA and revealed that over 54% *actions* containing security weaknesses. Khatami *et al.* [19] manually analyzed frequent change patterns in workflows to identify workflow smells. Seven types are confirmed with developers through pull requests. Furthermore, Ayala *et al.* [20] conducted an empirical study to measure the usage of GHA workflows and security policies in 173k popular repositories, highlighting the strong need for broader adoption of security policies.

VII. Conclusions and Future Works

Our study provides a comprehensive evaluation of security practices in GitHub Actions workflows over a two-year period, highlighting both advancements and persistent security issues in the CI/CD landscape.

Building on top of this study, our future work aims to engage with repository owners by sharing the identified security issues with our advices for improvement, together with a developer survey. This outreach seeks to capture the diverse awareness levels and perspectives developers hold on GHA security, helping to reveal the full landscape of priorities, challenges, and potential solutions in this area. To date, we have sent around 40 security reports and received positive feedback from developers, some of them updating their workflows based on our recommendations. Furthermore, we will continuously monitor GHA security practices on a regular basis using the pipeline developed in this study. This ongoing effort aims to provide a more comprehensive and dynamic overview of the evolution of developers' security practices within GHA.

References

[1] GitHub, "Github." [Online]. Available: https://github.com/

[2] ——, "Github about." [Online]. Available: https://github.com/about

[3] T. CI, "Travis ci." [Online]. Available: https://www.travis-ci.com/

[4] Jenkins, "Jenkins." [Online]. Available: https://www.jenkins.io/

[5] P. Rostami Mazrae, T. Mens, M. Golzadeh, and A. Decan, "On the usage, co-usage and migration of ci/cd tools: A qualitative analysis," *Empirical Software Engineering*, vol. 28, no. 2, p. 52, 2023.

[6] GitHub, "Github actions," https://github.com/features/actions.

[7] M. Golzadeh, A. Decan, and T. Mens, "On the rise and fall of ci services in github," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2022, pp. 662–672.

[8] I. Koishybayev, A. Nahapetyan, R. Zachariah, S. Muralee, B. Reaves, A. Kapravelos, and A. Machiry, "Characterizing the security of github CI workflows," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 2747–2763. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/koishybayev

[9] G. Benedetti, L. Verderame, and A. Merlo, "Automatic security assessment of github actions workflows," in *Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*, ser. SCORED'22. New York, NY, USA: Association for Computing Machinery, 2022, p. 37–45. [Online]. Available: https://doi.org/10.1145/3560835.3564554

[10] S. Muralee, I. Koishybayev, A. Nahapetyan, G. Tystahl, B. Reaves, A. Bianchi, W. Enck, A. Kapravelos, and A. Machiry, "ARGUS: A framework for staged static taint analysis of GitHub workflows and actions," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 6983–7000. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/muralee

[11] GitHub, "Security hardening for github actions," https://docs.github.com/en/actions/security-for-github-actions/security-guides/security-hardening-for-github-actions.

[12] O. Dabic, E. Aghajani, and G. Bavota, "Sampling projects in github for MSR studies," in *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021*. IEEE, 2021, pp. 560–564.

[13] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining github," *Empirical Softw. Engg.*, vol. 21, no. 5, p. 2035–2071, Oct. 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9393-5

[14] GitHub, "Rest api endpoints for workflows," https://docs.github.com/en/rest/actions/workflows?apiVersion=2022-11-28.

[15] ——, "Using third-party actions," https://docs.github.com/en/actions/security-for-github-actions/security-guides/security-hardening-for-github-actions#using-third-party-actions.

[16] ——, "Restricting permissions for tokens," https://docs.github.com/en/actions/security-for-github-actions/security-guides/automatic-token-authentication#modifying-the-permissions-for-the-github_token.

[17] ——, "Codeql," https://codeql.github.com/.

[18] H. O. Delicheh, A. Decan, and T. Mens, "Quantifying security issues in reusable javascript actions in github workflows," in *21st International Conference on Mining Software Repositories (MSR '24)*, F.R.S.-FNRS - Fonds de la Recherche Scientifique [BE]. ACM, 15 April 2024.

[19] A. Khatami, C. Willekens, and A. Zaidman, "Catching smells in the act: A github actions workflow investigation," in *24th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM). IEEE*, 2024.

[20] J. Ayala and J. Garcia, "An empirical study on workflows and security policies in popular github repositories," in *2023 IEEE/ACM 1st International Workshop on Software Vulnerability (SVM)*, 2023, pp. 6–9.