Action Required: A Mixed-Methods Study of Security Practices in GitHub Actions

Yusuke Kubo*‡, Fumihiro Kanei*, Mitsuaki Akiyama†, Takuro Wakai‡ and Tatsuya Mori‡§¶
*NTT DOCOMO BUSINESS, Inc.
Email: yuusuke.kubo@ntt.com, fumihiro.kanei@ntt.com
†NTT, Inc.
Email: mitsuaki.akiyama@ntt.com
‡Waseda University
Email: ykubo@nsl.cs.waseda.ac.jp, wakataku@nsl.cs.waseda.ac.jp, mori@nsl.cs.waseda.ac.jp
§ NICT ¶ RIKEN AIP

***Abstract*—GitHub Actions has become a dominant Continuous Integration/Continuous Delivery (CI/CD) platform, yet recent supply chain attacks like SolarWinds and tj-actions/changed-files highlight critical security vulnerabilities in such systems. While GitHub provides official security practices to mitigate these risks, the extent of their real-world implementation remains unknown. We present a mixed-methods study analyzing 338,812 public repositories and surveying over 100 developers to understand security practice implementation in GitHub Actions. Our findings reveal alarmingly low implementation rates across five key security practices, ranging from 0.6% to 52.9%. We identify three primary barriers: lack of awareness (up to 71.6% of non-adopters were unaware of practices), misconceptions about applicability, and concerns about operational costs. Repository characteristics such as organization ownership and recent development activity significantly correlate with better security practice implementation. Based on these empirical insights, we derive actionable recommendations that align intervention strategies with appropriate levels of automation, improve notification design to support awareness, strengthen platform- and IDE-level assistance, and clarify documentation on risks and applicability.**

I. Introduction

Continuous Integration and Continuous Delivery (CI/CD) has become fundamental pillars of modern software development. By automating code integration, testing, and deployment processes, these methodologies enable development teams to deliver software updates more frequently and reliably, with notable improvements in development efficiency and faster product releases. As software systems grow increasingly complex and development cycles shorten, CI/CD has evolved from an optional enhancement to an essential practice across the industry.

However, the widespread adoption of CI/CD has introduced new attack vectors that adversaries actively exploit. Supply chain attacks targeting CI/CD pipelines have emerged as a critical security concern, as demonstrated by high-profile incidents such as the 2020 SolarWinds attack [1], where attackers compromised the software build process to distribute malware through legitimate software updates, affecting approximately 18,000 organizations including US government agencies and major corporations. Among various CI/CD platforms, GitHub Actions has emerged as a dominant solution due to its seamless integration with GitHub repositories and ease of configuration [2], [3], yet this widespread adoption has exposed platform-specific vulnerabilities. The 2025 tj-actions/changed-files incident [4] exemplified these risks, where a compromised personal access token was used to retroactively modify release tags, injecting malicious code that extracted CI/CD secrets from memory and printed them in workflow logs, impacting numerous public repositories and prompting GitHub's emergency intervention.

These incidents highlight the urgent need to understand and address security challenges specific to GitHub Actions. Recent studies have systematically identified several vulnerabilities in the platform, including excessive permission grants, inadequate secret management, and risks associated with third-party action execution [5], [6]. In response to these concerns, GitHub has published official security practices and recommendations [7]. However, despite the availability of these guidelines, there remains a significant gap in our understanding: *to what extent are these security practices actually adopted in real-world development, and what factors influence their implementation?* Without empirical evidence of implementation patterns and barriers, security guidelines remain theoretical constructs that may fail to protect the millions of projects relying on GitHub Actions, leaving the software supply chain vulnerable to preventable attacks.

To address this gap, we present a comprehensive investigation of security practice implementation in GitHub Actions. Our study aims to systematically evaluate the current state of security practice implementation, identify the factors that influence implementation, and understand the barriers that prevent developers from implementing recommended practices. Specifically, our research addresses three key questions:

- **RQ1:** To what extent are GitHub Actions security practices

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240483
www.ndss-symposium.org

implemented in real-world repositories?
- **RQ2:** What repository characteristics are associated with the implementation or non-implementation of security practices?
- **RQ3:** What factors prevent developers from implementing security practices?

We employ a mixed-methods approach, combining quantitative analysis of repository data with qualitative insights from developer surveys, to answer these questions. For the measurement study, we analyzed 338,812 public GitHub repositories that use GitHub Actions, collected from the SEART-GHS [8] dataset. We developed automated detection methods to assess the implementation of five key security practices and employed statistical modeling techniques to examine the relationships between repository characteristics and practice implementation patterns. For the user study, we surveyed over 100 developers working with GitHub Actions recruited from our repository dataset through an online questionnaire, capturing diverse perspectives across different experience levels, organizational contexts, and geographical regions. This combination of quantitative repository analysis and qualitative developer insights provides both empirical evidence of implementation patterns and detailed insights into the human factors influencing security practice implementation.

This paper makes the following contributions:

- **Methodological framework:** We develop and validate automated detection methods to identify whether GitHub Actions security practices are implemented in repositories, achieving 100% accuracy on manual validation. Our detection techniques provide a reusable framework for continuous monitoring and are publicly available as the artifacts [9].
- **Large-scale empirical analysis:** Through analysis of 338.8K public repositories, we provide the first quantitative assessment of security practice implementation rates in GitHub Actions, revealing critically low implementation (0.6%–52.9%). We identify statistically significant relationships between repository characteristics (e.g., recent development activity, number of workflows, organization ownership) and practice implementation across five key security practices.
- **Developer perspectives and barriers:** Based on surveys with 102 developers working with GitHub Actions, we uncover that the primary barriers to implementation are lack of awareness (21.3%–71.6% were unaware of practices), lack of understanding and misconception (four participants did not implement practices due to misconception), and concerns about operational costs (25.3% cited maintenance burden). These findings suggest that improving security requires not only technical solutions but also better developer education and tool support.
- **Actionable recommendations:** Drawing from our findings, we propose concrete improvements including: aligning intervention strategies with appropriate levels of automation, introducing well-designed notifications to raise awareness, strengthening platform- and IDE-level assistance, and clarifying documentation on risks and applicability We have shared these insights with GitHub to inform platform improvements (see Appendix A for details).

II. Background

A. GitHub Actions

1) Overview: GitHub Actions [10] is a CI/CD platform provided by GitHub, characterized by its native integration with GitHub repositories. Other widely used CI/CD platforms include GitLab CI/CD [11], CircleCI [12], and Travis CI [13]. Since its general availability at the end of 2019, GitHub Actions has rapidly dominated the CI/CD market. Golzadeh et al. conducted a quantitative analysis of approximately 90K OSS projects on GitHub from 2012 to 2021, reporting that GitHub Actions achieved 51.7% market share by 2022 and reached usage levels comparable to the previously leading Travis CI within just 18 months of its release [2]. This success is attributed to its bundling strategy with GitHub services and its ecosystem of reusable actions [3], establishing GitHub Actions as a central infrastructure in modern software development.

2) Workflow Syntax: The core functionality of GitHub Actions is based on workflows, which are defined in the repository and used to automate various parts of the development process [14]. Figure 1 shows an example workflow. Workflows are written in YAML format and stored in the repository's `.github/workflows` directory. They are triggered by events such as pushes or pull requests. Each workflow consists of one or more jobs, with each job representing a distinct unit of execution. Each job consists of multiple steps, which are the smallest units of execution that describe operations such as running shell commands (via the `run` field) or invoking actions (via the `uses` field).

In automation scenarios where runtime information is required, developers can refer to the *github context* [15], which provides a set of built-in variables. The github context provides access to metadata during workflow execution, such as the event type or repository name, via the syntax `${{ github.xxx }}`. This mechanism enables conditional branching and dynamic command execution based on runtime context. The github context has a hierarchical structure that includes `github.event` and `github.env`, allowing various properties to be controlled in great detail.

3) Actions: In GitHub Actions, an action [16] is a reusable component that encapsulates a specific process or task. Developers can invoke an action within a workflow step using the `uses` field. There are three types of action implementations: JavaScript action, Docker action, and composite action, which combine multiple steps. Actions can be defined locally within a repository, retrieved from public GitHub repositories, or executed by referencing a container image directly. When referencing an action from a public repository, use the format `owner/repo@ref`, where ref can be a tag, branch name, or commit SHA. In this study, we refer to publicly available actions as public actions. GitHub provides a platform called GitHub Marketplace [17] for discovering and sharing public

```yaml
name: Example Workflow # Workflow Name
on:
  pull_request: # Trigger Event
    branches:
      - main # Target Branch
jobs:
  build: # Job Name
    runs-on: ubuntu-latest  # Running Environment
    steps:
      - name: Check out code
        uses: actions/checkout@v4 # Using action
      - name: Show Pull Request Title
        env:
          PR_TITLE: ${{
          ↪  github.event.pull_request.title }} #
          ↪  Access github context
        run: echo "$PR_TITLE"
```

Fig. 1. Example of GitHub Actions Workflow.

actions. This platform includes both officially maintained actions (e.g., `actions/checkout`) and actions published by external developers or organizations. In this study, actions developed by neither GitHub nor end users themselves are referred to as third-party actions.

4) Events Log: GitHub provides logging features, such as Security Log [18] and Audit Log [19], which record activity history. These logs also include events related to GitHub Actions. For example, when a workflow is enabled, the `workflows.enable_workflow` event is logged. These logs can be searched and reviewed through the Web UI, allowing developers to track configuration changes and use them for auditing and security purposes.

B. Related Work

1) Security Risks in CI/CD Platforms: Several measurement studies have systematically analyzed security vulnerabilities in CI/CD platforms, particularly GitHub Actions. Koishybayev et al. conducted a comprehensive analysis revealing that 99.8% of workflows operate with excessive privileges and 23.7% are vulnerable to malicious pull request attacks, highlighting fundamental security gaps in current CI implementations [5]. Gu et al. examined token management across major CI platforms, identifying four novel attack vectors and demonstrating that token leakage and excessive privilege configurations affect repositories even in large-scale organizations including GitHub, Google, and Microsoft [6]. Benedetti et al. developed GHAST (GitHub Actions Security Tool) and identified 24,905 security issues across 50 open-source projects [20], while Muralee et al. introduced ARGUS, the first static taint analysis framework specifically designed for GitHub Actions, analyzing over 2.7 million workflows from 1 million repositories [21]. While these studies effectively identify and categorize security risks in CI/CD environments, they primarily focus on vulnerability detection rather than examining the preventive security practices that development teams actually implement to mitigate these risks.

2) GitHub Security Features and User Perspectives: Research on GitHub's security features from a user perspective remains limited. Ayala et al. conducted multiple user studies with OSS maintainers, including 80 survey responses and 22 interviews. Their study examined perceptions and utilization of GitHub's security features, identifying supply chain trust issues and lack of automation as primary challenges, while also highlighting insufficient awareness and feature complexity as barriers to adoption [22]. However, this user study focuses on general repository security rather than CI/CD-specific security practices, leaving unaddressed the unique challenges of continuous integration workflows such as secret management in automated environments, dependency security in build processes, and access control for automated deployments.

3) Research Gap and Contribution: As discussed above, existing research primarily focuses on either identifying CI/CD vulnerabilities [5], [6], [20], [21] or examining general GitHub security features [22], but none specifically investigates security practices within GitHub Actions. In contrast, this study presents the first empirical investigation that focuses not on detecting vulnerabilities, but on the preventive measures officially recommended by GitHub, examining both their implementation in practice and the factors that hinder broader implementation. Furthermore, no study combines measurement and user study methodologies to provide a holistic understanding of CI/CD security. Our research addresses these gaps by systematically examining the implementation of security practices in GitHub Actions through a dual approach that quantifies practice implementation while understanding practitioner challenges and decision-making processes, bridging the gap between risk identification and practice implementation.

III. Mixed-method Study Design

A. Overview of Mixed-method Approach

Our goal is to understand how security practices for GitHub Actions are implemented by software developers and what barriers hinder their implementation. To this end, we adopted a mixed-methods approach that combines a measurement study and a user study. The measurement study quantitatively analyzes a large-scale dataset of GitHub repositories using GitHub Actions. We examine how widely security practices are implemented (**RQ1**) and how repository characteristics associate to their implementation (**RQ2**). The user study complements this by investigating the understanding and attitudes of developers working with GitHub Actions toward security practices. Through this analysis, we aim to identify perceived barriers to implementation (**RQ3**), and to better understand the factors that influence developers' security-related behaviors.

B. Security Practices

As part of its security measures for GitHub Actions, GitHub publishes a set of security practices [7]. In this study, we refer to these practices as security practices in GitHub Actions. These practices are updated periodically. For this study, we used publicly available information as of June 12, 2025[1].

[1]A minor revision to the security practices was made after July 11, 2025. However, we confirmed that the change does not affect this study. See Appendix B for details.

1) Practice Selection: GitHub's security practices cover topics such as workflow configuration, secret management, and environment protection. In total, 16 practices are recommended. They span the entire GitHub Actions life cycle, from workflow design to operation and monitoring. However, covering all 16 practices is not feasible from our study design perspective. In our measurement study, settings that are only visible to repository owners cannot be analyzed. In our user study, accounting for differences in developers' knowledge and access privileges would complicate the design and burden participants. Given these constraints, we narrowed the scope of our investigation to focus on practices applicable to both studies. This approach allows for an integrated analysis of the results from our measurement and user studies. We decided not to select practices based on their perceived security importance, because no objective criteria exist for comparing the relative importance of individual practices. In the absence of such criteria, any prioritization would rely on subjective judgment and could introduce arbitrary biases.

The selection of target practices was based on the following four criteria: (1) The practice involves the concrete configuration, implementation, or use of a feature or tool, rather than merely supporting developers' understanding or awareness; (2) Its implementation status (i.e., whether it is implemented) can be determined using publicly available data; (3) Whether the practice is applicable to a given repository can be identified based on publicly available data; (4) It does not require high-level privileges, meaning it can be implemented by users other than repository owners or admins.

Based on the above criteria, we selected five practices (IDs 2, 4, 6, 8, and 11 in Table VI in Appendix C) as the common targets of both the measurement and user study. The following practices were excluded from the scope: those consisting primarily of knowledge or informational items rather than actionable practices (IDs 3, 12, and 14); those that cannot be analyzed using open data (IDs 1, 5, 9, 13, 15, and 16); and those for which only individuals with administrative privileges can provide responses (ID 10). In addition, one practice (ID 7) was excluded because it overlaps with the target practice (ID 6). The rationale for these selections is further discussed in Appendix C. Note that we included supplementary questions about two additional practices (IDs 9 and 16) in our user study, which was not constrained by public data availability. These practices were selected and added based on feedback from our pilot interviews conducted during the survey design phase.

2) Selected Practices: We provide an overview of the five practices selected for analysis. In this study, each practice is assigned a unique identifier and name.

P1: CODEOWNERS. CODEOWNERS [23] feature enables the assignment of responsibility for specific files or directories to individuals or teams. This mechanism enforces proper review procedures for code changes, helping to prevent malicious modifications or accidental errors. To configure CODEOWNERS, create a CODEOWNERS file in the appropriate repository location and list the file paths along with their corresponding reviewers. Consequently, the designated reviewers are automatically requested to review any changes made to the specified files or directories. The 2025 GhostAction incident [24], in which malicious workflow additions led to attacks, highlights the important role that CODEOWNERS plays in discouraging such modifications.

P2: Mitigating Script Injection. A script injection attack can occur when an attacker adds malicious commands and scripts to github context, and the workflow interprets those strings as code which is then executed on the runner. To mitigate this type of attack, the following two approaches to implementing workflows are recommended: (1) Use a JavaScript action that processes the github context value, and pass the context value to the action as an argument; (2) Assign the github context value to an environment variable, and use the environment variable in the script. Both approaches aim to avoid handling the github context directly within scripts. The 2025 S1ngularity attack [25], in which a script directly processed the PR title, demonstrates the importance of safe context handling to eliminate this injection vector.

P3: OpenSSF Scorecard. Scorecard [26] is a tool that analyzes repositories and generates a score representing a repository's security risk. It evaluates repositories from various perspectives, such as risky workflow configurations, the use of dependency management tools, and branch protection settings. This enables developers to assess the security posture of a repository and identify areas for improvement. Scorecard is available as an action [27] and a workflow template [28], both of which can be executed within the user's own repository. The results can be viewed in the repository's security tab or in the logs of the corresponding workflow run. Past security incidents suggest that weak configurations often serve as entry points for attacks. Continuous assessment with Scorecard is therefore essential for identifying such risks at an early stage.

P4: Pinning Third-party Actions. Executing third-party actions created or compromised by malicious actors may result in the leakage of sensitive information or the unauthorized manipulation of the repository. To mitigate these risks, two recommended practices focus on how actions are referenced, specifically, the `ref` in the `owner/repo@ref` format[2].

- **P4-1: Pinning to SHA.** Pin actions to a full commit hash (SHA). This ensures that the exact same code is always executed, protecting against future changes or tampering. Although this is the most reliable method, it requires additional effort to verify and specify the hash value.
- **P4-2: Pinning to TAG.** Pin actions to a tag if the creator is trusted. This method offers better readability and ease of use. However, because tags can be reassigned to different commits, there is a risk that a compromised action could maliciously alter the tag to point to a harmful commit.

The 2025 tj-actions/changed-files incident [4] highlights the importance of appropriate reference methods against supply-chain risks. In this case, P4-1 would have been effective.

[2]In our user study, practice P4 is divided into P4-1 and P4-2 for analysis purposes; therefore, sub-identifiers are assigned accordingly.

P5: Dependabot. Continuously managing and updating the action versions in workflows is essential for maintaining security. In GitHub Actions, using Dependabot [29] is recommended as a way to automate this process. When `github-actions` is specified as the package ecosystem in the configuration file, Dependabot automatically creates pull requests when action updates are available, prompting developers to upgrade to newer versions. Thus, Dependabot enables the automatic management of action dependencies in GitHub Actions. This practice is recommended for all repositories using public actions, regardless of the programming language used. To avoid using outdated action versions that contain vulnerabilities reported in the GitHub Advisory Database [30], managing updates with Dependabot is effective.

IV. Measurement Study Methodology

This section presents the methodology designed to address **RQ1** and **RQ2** through open data analysis.

A. Repository Dataset

To minimize the load on the GitHub platform and facilitate reproducibility through precise data specification, this study leverages existing datasets rather than collecting repositories directly. Among the major GitHub datasets, we use SEART-GHS [8], which has been widely used in software engineering research [31], [32]. SEART-GHS is a structured dataset that includes repository-level metadata, such as the number of contributors and commits. It is provided in a format suitable for large-scale analysis, enabling multifaceted investigations based on repository attributes. However, it is limited to repositories with at least 10 stars, introducing a bias toward relatively popular projects. This bias is acceptable for our study because repositories with more stars are more likely to use GitHub Actions [33], making this dataset particularly relevant for analyzing GitHub Actions security practices. As SEART-GHS does not directly indicate whether repositories use GitHub Actions, we extended the dataset by querying the GitHub API and cloning repositories to identify and analyze GitHub Actions usage, as described in the following section.

Dataset Enhancement Procedure. To supplement SEART-GHS with GitHub Actions usage information, we enhanced the dataset through the following procedure:

1) Generate a list of GitHub repositories using the SEART-GHS. Forked repositories are excluded to avoid duplicate analysis, as they often contain content similar to their upstream sources. Archived repositories are also excluded, as they are read-only and pose no active security risk.
2) Use the GitHub API[3] to check whether each repository contains any workflow files.
3) If workflows are detected, classify the repository as using GitHub Actions and clone it to a local server.
4) Examine the contents of each cloned repository and exclude those that do not contain workflow files. This step is necessary because some GitHub features, such as GitHub Pages [34], may trigger a positive response from the GitHub API even if the repository contains no actual workflow files.

We collected 1,675,884 repository records from SEART-GHS on May 12, 2025. The dataset contains repositories created before December 31, 2024. From this dataset, we identified those that use GitHub Actions, and between June 6 and 7, 2025, we cloned a total of 338,812 repositories, which included 861,680 workflows and 3,424,855 actions. This corresponds to approximately 20.2% of the entire SEART-GHS.

B. Framework for Automated Detection of Security Practices

To address **RQ1**, we constructed an automated framework to detect the implementation status of GitHub Actions security practices. This framework is based on patterns defined according to GitHub's official documentation and the specifications of related features and tools. Each detection method specifies both the applicability criteria (which repositories are subject to the practice) and the detection procedure (how to identify actual implementation).

1) Security Practice Detection Schemes:

P1: CODEOWNERS Detection.

Applicability: P1 applies to all repositories.

Detection: We detect P1 implementation by verifying three conditions: (1) a CODEOWNERS file exists; (2) the file is in a valid location (`.github/`, repository root, or `docs/`); and (3) the file contains at least one rule. All three conditions must be met.

P2: Mitigating Script Injection Detection.

Applicability: P2 applies to repositories containing workflow steps that use the github.event context[4]. Specifically, repositories with at least one step using the github.event context in certain patterns shown below are applicable for P2.

Detection: We parse all workflow files and classify each step into four categories: (1) use of `github.event` in the `run`; (2) use of an environment variable set from `github.event` in the `run`; (3) use of `github.event` or an environment variable set from `github.event` in the `with`; (4) all other cases. Repositories with steps in categories (1), (2), or (3) are applicable for P2. Among these applicable repositories, those that do not include any steps in category (1) are detected as implementing P2.

P3: Scorecard Detection.

Applicability: P3 applies to public repositories only, as this practice is designed for open-source projects.

Detection: We detect P3 implementation through three steps: enumerate all workflow files, parse the `uses` fields, and check for the `ossf/scorecard` action. Any repository using this action implements P3.

P4: Pinning Detection.

Applicability: P4 applies to repositories using third-party actions—those created by entities other than GitHub or the repository owner. We determine whether an action is third-party by examining the owner component (i.e., the `owner`

[3]We used the Enterprise plan, which has a higher rate limit.

[4]The github.event context within the github context, as this is most relevant to injection risk

in `owner/repo@ref`), which requires identifying official GitHub accounts. For this, we use GitHub's verified domain mechanism [35], recognizing accounts registered under *github.com* as official. Repositories using at least one third-party action are applicable for P4.

Detection: We detect P4 implementation in three steps. First, we extract all third-party actions from workflow files. Second, we verify each action meets one of two criteria: (1) referenced by a full 40-character commit hash, or (2) has a verified badge and is referenced by a tag. We determine reference types and verification status using GitHub API data and GitHub Marketplace information (see Appendix D). Third, we confirm P4 implementation only when all third-party actions meet these criteria.

P5: Dependabot Detection.

Applicability: P5 applies to repositories using public actions (`owner/repo@ref`). Due to Dependabot's specifications, only public actions are subject to dependency management. Repositories using only local actions (`./path/to/dir`) or container images (`docker://image:tag`) are not applicable for P5.

Detection: We detect P5 implementation by checking three conditions: (1) a `dependabot.yml` or `dependabot.yaml` file exists; (2) the file is in the `.github` directory and has valid YAML syntax; and (3) the file specifies `github-actions` as a managed ecosystem. All conditions must be satisfied.

2) Validation of the Framework Implementation: The framework developed in this study employs the two-aspect approach described above: determining applicability and detecting implementation for each security practice. Since these methods are based on clearly defined patterns from official documentation, they should theoretically produce accurate results without false positives or false negatives. However, coding errors could potentially introduce inaccuracies in the analysis. To ensure reliability, we manually validated the framework outputs by inspecting the code, workflows, and configuration files of randomly selected repositories. Specifically, we examined 100 repositories containing 250 workflows and 981 actions. We verified both applicability decisions and implementation detection for each practice. All validation cases yielded the expected results, confirming the reliability of our framework for subsequent analyses.

C. Statistical Modeling

To address **RQ2**, we conducted logistic regression analysis to identify repository characteristics associated with the implementation of security practices. For this analysis, we converted the implementation status of each relevant security practice identified in Section IV-B into a binary variable (1: implemented, 0: not implemented), which we use as the dependent variable. In addition, we use repository-level features as independent variables as shown in Table I. These include project popularity and activity indicators (e.g., stars, contributors, commits, and recent update), code characteristics (e.g., codebase size and repository age), workflow attributes (e.g., number of workflow files and workflow developers), and owner type (e.g., whether the repository owner is a user account or an organization account). These variables were not predetermined, but rather, were selected through an iterative process during the analysis. As in previous studies [36], [37], [38], we assessed multicollinearity using the variance inflation factor (VIF), and excluded variables with high correlations (see Appendix E). Additionally, statistical significance and practical interpretability were considered during the selection process.

TABLE I
INDEPENDENT VARIABLES USED IN LOGISTIC REGRESSION ANALYSIS.

Independent Variables	Source	Type
Number of Stars	SEART-GHS	Continuous
Number of Contributors	SEART-GHS	Continuous
Number of Commits	SEART-GHS	Continuous
Codebase Size	SEART-GHS	Continuous
Repository Age	SEART-GHS+	Continuous
Recent Activity	SEART-GHS+	Binary
Number of Workflow Files	Cloned Repository	Continuous
Number of Workflow Developers	Cloned Repository	Continuous
Owner Type	GitHub API	Categorical

SEART-GHS+ indicates that the SEART-GHS metadata has been processed. Repository Age: Days between repository creation and repository collection. Recent Activity: Whether the repository had commits after 2025-01-01, which is approximately five months before repository collection. Owner Type: Whether the repository owner account is User or Organization.

We examined how these variables relate to the implementation of security practices using odds ratios, which indicate the direction and strength of association. The interpretation of the odds ratio depends on the type of independent variable. In this analysis, an odds ratio greater than 1 for a continuous variable indicates a positive association between the variable and the likelihood of implementing the security practice. An odds ratio greater than 1 for categorical or binary variables indicates that the security practice is more likely to be implemented than in the reference category. Importantly, the magnitude of an odds ratio does not necessarily correspond to statistical significance. In particular, when the sample size is large, even minor effects may yield p-values below the significance threshold. Therefore, it is inappropriate to rely solely on p-values when evaluating statistical validity. In this study, both p-values and 95% confidence intervals were used to assess statistical significance. A result is considered statistically significant if the p-value is below 0.05 and the 95% confidence interval does not include 1.

V. MEASUREMENT STUDY RESULT

This section presents the results related to **RQ1** and **RQ2**.

A. Implementation Rates of Security Practices

We define the implementation rate of a security practice as the proportion of repositories that actually implement the practice, after excluding those for which the practice is inapplicable. For example, a repository is considered applicable to the security practice **P5 (Dependabot)** when it uses public

TABLE II
IMPLEMENTATION RATES OF EACH SECURITY PRACTICE.

Security practice	Target repo.	Implementation rate
P1 CODEOWNERS	338,812	7.1% (23,890)
P2 Mitigating Script Injection	43,516	52.9% (23,005)
P3 OpenSSF Scorecard	338,812	0.6% (1,965)
P4 Pinning Third-party Actions	233,124	16.2% (37,693)
P5 Dependabot	332,928	10.7% (35,588)

actions; therefore, repositories that do not use public actions are excluded from the P5 implementation rate calculation. Using our framework described in Section IV-B, we first filter out inapplicable repositories, then determine which of the remaining repositories have implemented the practice. Based on this, we calculate the implementation rate.

Table II presents the implementation rates for each security practice. None of the security practices were widely implemented, despite a wide variation in implementation rates, ranging from 0.6% to 52.9%. In particular, practices involving the use of security features and tools, such as P1 (CODEOWNERS), P3 (OpenSSF Scorecard), and P5 (Dependabot), showed notably low implementation rates, each below 11%.

We further analyzed incomplete or incorrect configurations found in non-implementation cases where configuration files were present. This analysis focused on P1 and P5, for which specification compliance can be evaluated based on the presence, location, and content of the configuration file. For P1, among 314,922 non-implementation repositories, 1,287 had a configuration file. These included invalid file placement (873, 0.28% of all non-implementation cases), and file lacking ownership rules (414, 0.13%). For P5, among 297,340 non-implementation repositories, 21,061 had a configuration file. These included invalid file placement (757, 0.25%), syntax errors (134, 0.05%), and configurations that did not include the `github-actions` ecosystem (20,170, 6.78%). These cases do not meet the required specifications and thus cannot be regarded as implementing the practices.

Answer to RQ1: The implementation rates of all practices ranged from 0.6% to 52.9%, indicating that none are widely implemented. In particular, practices involving the use of security tools and features (P1: CODEOWNERS, P3: Scorecard, P5: Dependabot) showed low implementation rates between 0.6% and 10.7%, suggesting significant gaps in implementation.

B. Repository Characteristics Associated with Security Practice Implementation

We applied logistic regression to assess associations between repository characteristics and the implementation of security practices. The results of logistic regression analysis are shown in Table III. The listed values represent the odds ratios for each independent variable. We consider an effect statistically significant when the p-value is below 0.05 and the 95% CI does not include 1 as described in Section IV-C.

Number of Stars, Commits, Codebase Size, and Repository Age. Number of stars and commits, codebase size, and repository age rarely had a statistically significant impact on the implementation of each security practice. Even when a significant impact was observed, the effect size was minimal. In other words, these characteristics did not substantially influence whether security practices were implemented. It was shown that high repository popularity (Stars), scale (Code base size, Commits), or maturity (Repository Age) do not necessarily imply the implementation of security practices.

Number of Contributors. Number of contributors had a statistically significant impact on P1, P2, P3, and P5. Repositories with more contributors were more likely to implement P1 and P3, but less likely to implement P2 and P5. However, since the odds ratios were close to 1, representing the effect of a single additional contributor, the overall impact was limited. This suggests that an increased number of contributors does not necessarily mean a greater proportion of them are directly involved in implementing security practices.

Recent Activity. Recent activity had a statistically significant impact on P1, P3, P4, and P5. Repositories with recent activity were more likely to implement these practices compared to those without. As expected, repositories with recent activity reflecting continued maintenance were more likely to implement security practices.

Number of Workflow Files. Number of workflow files had a statistically significant impact on all security practices. Specifically, repositories with more workflow files were more likely to implement P1, P3, and P5, but less likely to implement P2 and P4. P2 and P4 are practices related to the writing of workflow files (see Section III-B2). These results suggest that, as the number of workflows increases, it may become more difficult to consistently apply these practices.

Number of Workflow Developers. Number of workflow developers had a statistically significant impact on all security practices. Specifically, repositories with more workflow authors were more likely to implement P1, P3, P4, and P5, while less likely to implement P2. Unlike the number of contributors, which showed no clear trend, the number of workflow developers exhibited consistent patterns. This suggests that developers involved in implementing and maintaining workflows may play a central role in the implementation of security practices.

Owner Type. Owner type had a statistically significant impact on all security practices. Specifically, repositories owned by user accounts were more likely to implement P2 and P4, whereas those owned by organization accounts were more likely to implement P1, P3, and P5. This suggests that repositories under organizational ownership tend to adopt security-related features and tools such as P1, P3, and P5 (see Section III-B2). In other words, these results imply that organizational governance may encourage the implementation of security practices.

Answer to RQ2: Repositories with recent activity reflecting ongoing maintenance were more likely to im-

TABLE III
LOGISTIC REGRESSION RESULTS FOR REPOSITORY CHARACTERISTICS AND SECURITY PRACTICES.

Independent Variables	P1	P2	P3	P4	P5
Number of Stars	.99998*** [0.99997, 0.99998]	1.00000+ [0.99999, 1.00000]	1.00000+ [1.00000, 1.00001]	1.00001*** [1.00000, 1.00001]	.99999*** [0.99999, 1.00000]
Number of Contributors	**1.00167*** [1.00135, 1.00198]**	**.99895*** [0.99856, 0.99934]**	**1.00232*** [1.00164, 1.00300]**	.99956* [0.99921, 0.99991]	**.99825*** [0.99792, 0.99858]**
Number of Commits	1.00000*** [0.99999, 1.00000]	1.00000+ [1.00000, 1.00000]	1.00000** [1.00000, 1.00000]	1.00000+ [1.00000, 1.00000]	.99999*** [0.99999, 0.99999]
Codebase Size	1.00000* [1.00000, 1.00000]	1.00000** [1.00000, 1.00000]	1.00000** [1.00000, 1.00000]	1.00000*** [1.00000, 1.00000]	1.00000*** [1.00000, 1.00000]
Repository Age	.99989*** [0.99987, 0.99990]	1.00007*** [1.00006, 1.00009]	1.00002+ [0.99999, 1.00006]	.99999** [0.99998, 1.00000]	1.00011*** [1.00010, 1.00012]
Recent Activity	**1.41553*** [1.37468, 1.45758]**	.98246+ [0.94361, 1.02291]	**2.69133*** [2.41585, 2.99821]**	**1.07503*** [1.05091, 1.09971]**	**2.33900*** [2.28188, 2.39755]**
Number of Workflow Files	**1.05185*** [1.04799, 1.05572]**	**.97287*** [0.96831, 0.97745]**	**1.03052*** [1.02526, 1.03581]**	**.97246*** [0.96803, 0.97691]**	**1.05856*** [1.05475, 1.06238]**
Number of Workflow Developers	**1.10088*** [1.09676, 1.10502]**	**.98085*** [0.97661, 0.98511]**	**1.04417*** [1.03814, 1.05024]**	**1.01408*** [1.01041, 1.01776]**	**1.12425*** [1.12001, 1.12849]**
Owner Type	**.24995*** [0.24078, 0.25947]**	**1.05489* [1.00966, 1.10216]**	**.35556*** [0.31570, 0.40044]**	**1.06167*** [1.03721, 1.08672]**	**.89690*** [0.87462, 0.91974]**

The listed values indicate the odds ratio. Brackets show the 95% confidence interval. Significance levels are + $p > .05$; * $p < .05$; ** $p < .01$; *** $p < .001$. Values are shown in bold when the odds ratio is statistically significant and differs from 1.000 by at least 0.001. For binary variables such as Recent Activity, the odds ratio compares cases with value 1 against those with value 0. For categorical variables such as Owner Type, User is treated as the reference category, and the odds ratio for Organization is interpreted relative to it. For all other (continuous) variables, the odds ratio represents the change in odds associated with a one-unit increase in the variable.

plement security practices. In contrast, high repository popularity (number of stars), scale (codebase size and number of commits), maturity (repository age), or available developer resources (number of contributors) did not necessarily lead to greater implementation of security practices. As the number of workflows increases, it may become more difficult to consistently apply practices related to workflow configuration, specifically P2 (Mitigating Script Injection) and P4 (Pinning). The developers who implement and maintain workflows (workflow authors) may play a more central role in implementing these practices than the overall number of contributors. Finally, the type of repository ownership showed a notable correlation with the implementation of security practices. Repositories owned by organization accounts were more likely to implement P1 (CODEOWNERS), P3 (Scorecard), and P5 (Dependabot), which involve using security-related features and tools. This suggests that organizational governance may encourage the implementation of security practices.

VI. USER STUDY METHODOLOGY

This section presents the methodology designed to address **RQ3** through an online survey. The full text of the questionnaire used in our survey is available in the supplementary materials included in the artifacts [9].

A. Survey Design

Our survey included questions about GitHub repositories contributed to, their use of GitHub Actions, and their attitudes toward security practices. For GitHub repositories, we asked about basic statistics (e.g., stars, contributors), repository ownership, participants' roles, and configuration. For GitHub Actions, we asked about their usage and operation, including the types of tasks they automate and the sources of information they refer to when configuring GitHub Actions. Questions on security practices included whether participants implemented each practice, the reasons for not implementing practices, and factors considered important when implementing practices. Implementation status and its reasons were asked for each practice (P1–P5) described in Section III-B. For P4, we further distinguished between two variants: P4-1 (Pinning to SHA) and P4-2 (Pinning to TAG). While both address the same security risks (i.e., securely referencing third-party actions), their operational details differ, and we expected these differences to influence developers' perceptions. We recruited developers with experience using GitHub Actions and collected their responses. Note that we intentionally avoided linking user study responses to identifiable public repositories to preserve participant privacy and anonymity. The details of the questionnaire are described in Section VI-B, the recruitment procedure in Section VI-C, and the analysis methods in Section VI-D.

Designing Questionnaire and Piloting. To design the questionnaire, we conducted pilot interviews with five developers who had experience using GitHub Actions. These developers, with 3 to 12 years of development experience, were recruited through snowball sampling via the authors' professional contacts. Although all participants worked at the same company, they were involved in different development projects. In the pilot interviews, we asked participants about how they use GitHub Actions in practice and how they perceive the security practices recommended by GitHub. Although we did not give monetary compensation to the participants, they were offered a summary of the study findings as an incentive. Based on the interview results and existing research [22], [39], [40],

[41], [42], [43], we developed an initial questionnaire. We then conducted multiple rounds of pilot surveys and revisions. Revisions included clarifying ambiguous wording, deleting duplicate questions/options, and ensuring that the questionnaire was of an appropriate volume. In total, three rounds of pilot surveys were conducted with eight professional developers before finalizing the instrument. The questionnaire was initially written in the authors' native language and translated into English. To ensure linguistic clarity and accuracy, the English version was reviewed by two pilot participants, one a native English speaker and the other bilingual.

Summarizing Documentation of Security Practices. Before asking questions about security practices (e.g., implementation status), we provided a brief explanation of the corresponding practice. Instead of directly quoting GitHub's official documentation on GitHub Actions security practices [7], we prepared summarized descriptions for each practice. This approach was based on feedback from the pilot interviews, where participants noted that the official documentation was lengthy and difficult to read (i.e., it contained unfamiliar technical terms and assumed prior knowledge of security concepts, which some developers lacked). To minimize participants' cognitive loads and avoid misinterpretation of the practices, we developed concise and accessible explanations that preserve the key points of the original content. These explanations were repeatedly reviewed and revised by two security experts to ensure consistency with the official documentation. Details of the development process are available in Appendix F.

B. Questionnaire

The questionnaire included four sections: questions about GitHub repositories, GitHub Actions, security practices, and participant demographics. The demographic section covered general demographic information, years of experience in software development and GitHub Actions, and questions on vulnerability identification and mitigation from the Secure Software Development Self-Efficacy Scale (SSD-SES) [40].

Questions about GitHub Repository. We asked participants to provide basic information about the repositories they were involved in, including the primary programming language, the number of stars, contributors, and commits, as well as the date of the most recent commit. Additionally, we asked about the type of owner account for the repository (i.e., user or organization) and the participant's role in the repository (e.g., admin/owner, collaborator, contributor). For participants with admin or owner roles, we asked whether they had configured GitHub Actions to allow the creation/approval of pull requests, which corresponds to practice ID 9 in Table VI in Appendix C. Because these settings are only visible to users with administrative privileges, we asked these questions only to participants with the appropriate level of access.

Questions about GitHub Actions. We asked participants about their use of GitHub Actions, including the types of tasks they automate (e.g., build, test, deployment), the information sources they refer to when configuring workflows (e.g., official documentation, web articles, developer forums), the perceived severity of the negative impact on development if GitHub Actions were to become temporarily unavailable (measured on a 4-point Likert scale), whether their workflow files contain sensitive information, and whether they use third-party actions. These questions were designed based on existing studies [41], [39] that examined how GitHub Actions are used in practice.

Questions about Security Practices. For each security practice (P1–P5), we asked participants whether they currently implement the practice. Before each question, we presented a brief explanation of the practice described in Section VI-A. Participants selected from options indicating whether they implement the practice, do not implement it (either never or no longer), use an alternative approach to achieve the same goal, or are uncertain about its use or meaning. Those selecting an alternative approach were asked to describe it briefly in an open-ended format. Participants who did not implement the practice were asked to select the reason for not implementing it from a list of options, including an "Other" option with an open-ended answer form. These options were developed based on the results of our pilot interviews and existing studies [42], [22], [41]. We also asked participants who indicated they were not aware of the practice whether they would be willing to implement it. Additionally, participants were asked to select up to three factors they considered important for implementing security practices from a list of options. Unlike earlier questions, this question targeted general perceptions across practices, rather than individual ones.

We also asked about log auditing in GitHub Actions, corresponding to practice ID 16 in Table VI in Appendix C. GitHub provides event logs related to GitHub Actions (see Section II-A4). Participants were asked whether they regularly audit these logs, and those who did not were asked to select the reasons for not auditing these logs in a multiple-choice format. We also asked participants to indicate the extent to which the platform (i.e., GitHub) should autonomously support or take actions for log auditing. This was measured using a 6-level scale based on the automation levels described in existing research [43]. We designed these questions based on feedback from pilot interviews, which highlighted the challenges of log auditing (e.g., implementation cost, the need for automation).

C. Recruitment

Our survey was conducted in July 2025 using the Qualtrics [44]. We recruited participants by sending emails containing a link to the questionnaire. The contact information of developers was collected from GitHub. Because mass emailing a large number of developers could result in reaching non-eligible participants (e.g., developers with no experience using GitHub Actions) and cause survey fatigue, we applied criteria to narrow down the target population. First, we retrieved the top 100 contributors (ranked by number of contributions) from each repository in our dataset (Section IV-A), resulting in a total of 1,439,072 unique GitHub accounts. We further selected 3,405 accounts that met the following criteria: (1) publicly listed an email address on their GitHub profile page; (2) were involved in configuring GitHub Actions; and (3)

were not bots. From these accounts, we randomly sampled 1,800 accounts, with half primarily contributing to repositories owned by user accounts and the other half to repositories owned by organizations. As a result, recruitment emails were sent to 1,800 developers whose email addresses were publicly available. While no monetary compensation was provided, we offered participants a summary of the survey results as an incentive. We describe the details of the filtering criteria for participants in Appendix G and ethical considerations related to recruitment, informed consent, privacy, and anonymity, as well as incentives, in Appendix A2.

D. Data Analysis

For the close-ended questions, we calculated the number and proportion of responses for each option. Blank responses were excluded from the denominator when calculating proportions. Due to the limited number of participants, we did not perform statistical hypothesis testing in our quantitative analysis; instead, we focused on reporting and comparing raw counts and proportions. This approach was taken to avoid the issue of low statistical power in hypothesis testing [45].

Open-ended responses in the questionnaire, specifically those describing alternative approaches to security practices and those provided under "Others" when explaining reasons for not implementing practices, were coded by two coders. Each coder independently created a codebook by assigning codes to the responses. After this initial coding, the coders compared their respective codebooks and resolved any coding conflicts through discussion. This interactive process resulted in a finalized codebook that reflected consensus between the coders. Because the coding was conducted through consensus rather than statistical agreement, reporting inter-coder reliability was not required, following established guidelines in qualitative research [46]. Some responses initially describing alternative approaches were actually reasons for non-implementation, and vice versa. Therefore, coding was conducted in two stages. First, coders classified each response as either describing an alternative approach or explaining a reason for not implementing the practice. Second, they applied thematic codes to each classified response.

VII. User Study Result

This section presents the results related to **RQ3**. The complete set of responses to each question in our questionnaire is available in the supplementary materials included in the artifacts [9].

A. Participant Statistics

1) Demographics: Of the 151 survey accesses, 102 participants completed the questionnaire. The remaining 49 incomplete responses were excluded from all analyses in this paper. Participants were overwhelmingly male (Male: N=92, Female: N=4, Non-binary: N=1, Prefer not to say: N=5). This distribution is consistent with existing studies targeting developers such as OSS maintainers [22], [47], [48]. Their countries of residence covered 28 nations. The top four were the United States (N=22), Japan (N=11), Germany (N=8), and France (N=8). Participants were nearly evenly divided between company employees (N=47) and freelancers or independent developers (N=43). The average years of experience in software development was 12.3 (Md=8.5), and the average SSD-SES score was 26.3. This score is comparable to that reported in existing research [49] on recruiting developers through a freelancing platform (i.e., Freelancer.com [50]). On average, participants took 32.6 minutes (Md=20.1) to complete the questionnaire. We analyzed responses from 102 completed questionnaires.

2) Characteristics of Repository: The repositories that participants were involved in developing were almost evenly owned by user and organization accounts (User: N=53, 52.0%, Organization: N=49, 48.0%). A majority of participants (N=88, 86.3%) reported that the latest commit in their repository was within the past month, suggesting that many were involved in actively maintained repositories. Most participants (N=90, 88.2%) held an admin or owner role.

3) Usage of GitHub Actions: The most common tasks automated with GitHub Actions were build (N=92, 90.2%), testing (N=83, 81.4%), and linting (N=60, 58.8%). Most participants (N=97, 95%) reported referring to GitHub's official documentation when configuring GitHub Actions. Nearly all participants (N=99, 97.1%) reported using third-party actions.

TABLE IV
Implementation Rates of Security Practices and the Top 3 Reasons not Implementing Security Practices.

SPs	Implementation rates	Reasons not implementing SPs	% of participants (by reason)
P1	17.6% (18/102)	Lack of awareness	41.4% (29/70)
		Unnecessary/overly strict	40.0% (28/70)
		Maintenance/operational costs	4.3% (3/70)
P2	46.1% (47/102)	Lack of awareness	50.0% (19/38)
		Unnecessary/overly strict	31.6% (12/38)
		Maintenance/operational costs	5.3% (2/38)
		Unclear risks or benefits	5.3% (2/38)
P3	5.9% (6/101)	Lack of awareness	71.6% (63/88)
		Unnecessary/overly strict	20.5% (18/88)
		Maintenance/operational costs	1.1% (1/88)
		Requires learning effort	1.1% (1/88)
		Cannot get approval from team	1.1% (1/88)
P4-1	19.4% (19/98)	Unnecessary/overly strict	30.7% (23/75)
		Maintenance/operational costs	25.3% (19/75)
		Lack of awareness	21.3% (16/75)
P4-2	61.2% (60/98)	Lack of awareness	38.7% (12/31)
		Unnecessary/overly strict	22.6% (7/31)
		Maintenance/operational costs	12.9% (4/31)
P5	43.4% (43/99)	Lack of awareness	36.2% (17/47)
		Unnecessary/overly strict	25.5% (12/47)
		Maintenance/operational costs	8.5% (4/47)

SPs: Security practices. Implementation rates represent the proportion of respondents who reported currently implementing each practice. Percentages of reasons for not implementing a practice are calculated only among respondents who indicated that they do not currently implement the practice. Parentheses show raw counts: for implementation rates, (number of implementers/total respondents); for reasons, (number selecting the reason/number of non-implementers). When multiple reasons are tied within the top 3, all tied reasons are presented.

B. Quantitative Analysis

1) Implementation Rate of Security Practices: Table IV presents the implementation rates of each security practice,

along with the top three reasons for not implementing them. Implementation rates varied by practice, ranging from 5.9% (N=6) for P3 (Scorecard) to 61.2% (N=60) for P4-2 (Pinning to TAG). Compared to the measurement study results (Table II), the user study participants reported higher implementation rates for P1 (CODEOWNERS), P3, P4-2, and P5 (Dependabot). This may be due to the recruitment criteria of the user study, which targeted developers with many contributions related to GitHub Actions and relied on voluntary participation, potentially resulting in higher implementation rates.

2) Reasons Not Implementing Security Practices: As shown in Table IV, the most common reason for not implementing each security practice, except P4-1 (Pinning to SHA), was lack of awareness (21.3%–71.6%). This suggests that overall awareness of security practices in GitHub Actions remains low. In particular, for P3, 71.6% (N=63) of participants who did not implement the practice reported being unaware of it, highlighting the particularly low awareness of the OpenSSF Scorecard. Among participants who had been unaware of a practice, an average of 46.8% (27.3%–76.5% for each practice) indicated they would implement it after learning about it through this survey. This positive reaction suggests that increasing developers' awareness could substantially promote their implementation. Additionally, for P4-1, concern about increased maintenance or operational costs was the second most selected reason (N=19, 25.3%). Participants in the pilot interviews mentioned that pinning third-party actions using SHA (P4-1) increases maintenance costs, as manual updates are required when an action is updated. These findings suggest that developers tend to view P4-1 as burdensome in terms of maintenance effort.

3) Important Factors when Implementing Security Practices: The factor most commonly perceived as important when participants implement security practices was "Low maintenance and operational overhead" (N=81, 79.4%), selected at a notably higher rate than the other factors. This aligns with earlier results showing that, for certain security practices, maintenance or operational costs were a major barrier to implementation (see Section VII-B2). The second most selected factor was "Easy to set up and quick to adopt" (N=47, 46.1%), followed by "Clearly effective in mitigating relevant risks" (N=38, 37.3%). Regarding "Clearly effective in mitigating relevant risks", participants in the pilot interviews mentioned that it is unclear which specific risks each security practice addresses. These results suggest that clarifying the types of risks addressed by each security practice is important for encouraging their implementation.

4) Log Auditing in GitHub Actions: A majority of participants (N=80, 81.6%) reported that they do not audit the event log in GitHub Actions. The top three reasons for not conducting log audits were: "I do not see the need for auditing" (N=36, 45%), "I was not aware of security logs or audit logs" (N=35, 43.8%), and "Our team lacks sufficient personnel or time resources" (N=22, 27.5%). These reasons reflect trends similar to those observed for not implementing security practices (Section VII-B2). The most preferred level of automation in log auditing was "Human Approval", selected by 58 participants (59.2%). In contrast, very few participants supported "Fully Automated" (N=1, 1.0%) or "Execute and Inform" (N=2, 2.0%). These results indicate that while developers are receptive to automated support for log auditing in GitHub Actions, they generally prefer to retain final control over decision-making.

C. Qualitative Analysis

1) Alternative Approaches to Security Practices: We analyzed open-ended responses regarding alternative approaches to security practices with the procedure described in Section VI-D. A total of 28 valid responses were analyzed, and two responses were excluded as invalid (i.e., responses unrelated to the question).

For P3, P4-1, and P5, many participants referred to the use of alternative tools in place of the practices (14 responses). For example, some participants noted using static analysis tools instead of the OpenSSF Scorecard (P3), or using Renovate [51] to manage dependency updates instead of Dependabot (P5). In the case of P2 (Mitigating Script Injection) and P4-2, several participants (7 responses) indicated that they used alternative security features to mitigate related risks. For example, instead of implementing P4-2, some participants specified action versions with SHA, meaning that they were intentionally implementing P4-1 as an alternative to P4-2. Others mentioned restricting the use of third-party actions using an allowlist as an alternative to P4-2. Note that this allowlist feature [52] is available only to organization accounts on GitHub, not to individual user accounts. For P1, some participants (7 responses) reported not using CODEOWNERS but ensuring that all workflow file updates are reviewed as part of their development process.

2) Reasons Not Implementing Security Practices: We analyzed the open-ended responses regarding reasons for not implementing security practices, using the same approach in Section VII-C1. A total of 35 valid responses were analyzed, and two invalid responses were excluded.

The most frequent reason was that the practice was "Not Applicable" to the participants' repository (22 responses). Many noted that the practice was not relevant to their context (i.e., for P1, a participant answered *"I am the sole maintainer."*) On the other hand, we observed some responses that included misconceptions about the applicability of the practices. For example, three participants indicated that they do not implement P5 because Dependabot does not support the programming language they use. However, this practice involves updating dependencies in GitHub Actions workflows and is applicable to any repository that uses third-party actions, regardless of the programming language used.

Three responses expressed concerns about the potential negative impact on software development. For instance, one participant mentioned that P1 could act as a barrier to entry for new contributors, while another indicated a preference

for updating dependencies manually through community discussion rather than relying on automation (P5). Additionally, similar to the misconception mentioned earlier, a participant mistakenly interpreted P5 as language-specific tooling, rather than its intended focus on GitHub Actions.

Three responses raised concerns about the potential negative impact on security. Notably, one participant commented on P4-1 (Pin actions to a full commit hash (SHA)) as follows: *"This could make us more vulnerable by not automatically using the latest version of an action that might have important security fixes."* While P4-1 ensures that actions are immutable and not tampered with, it requires explicit updates to receive patches. This concern can be addressed by combining P4-1 with P5 (Dependabot), which detects and notifies about updates. In this case, the participant had not implemented P5 and was unaware of its existence, suggesting that the benefits of using P4-1 and P5 together may not be well understood among developers. We note that the official GitHub documentation on security practices does not currently explain such complementary use.

Two participants indicated a lack of resources as the reason they did not implement P4-1, which is consistent with the concern about maintenance and operational costs reported in Section VII-B2. Other responses (5 total) included concerns such as high learning cost (P4-1), the difficulty in configuration (P3), and a lack of a specific reason for not implementing (P4-1).

Answer to RQ3: Three key barriers to implementing security practices in GitHub Actions were identified.
Lack of Awareness. Many developers were unaware of the practices. Between 21.3% and 71.6% of participants who had not implemented specific practices reported not knowing about them. Additionally, 43.8% of those not performing log auditing were unaware that GitHub Actions generates audit logs (Sections VII-B2 and VII-B4).
Lack of Understanding and Misconceptions. Some participants misunderstood the applicability or benefits of practices. In open-ended responses, participants stated that certain practices were not relevant to their projects or might negatively affect development processes. In four cases, non-implementation was clearly due to misconceptions about the practice's purpose or applicability (Section VII-C2). Although few, these examples suggest that misunderstanding can hinder implementation.
Cost and Resource Concerns. Concerns about operational overhead and limited resources were common. For example, 25.3% of participants not implementing P4-1 (Pinning to SHA) cited maintenance burden. Open-ended responses echoed resource constraints as a barrier to P4-1 implementation (Sections VII-B2 and VII-C2). Similarly, 27.5% of participants not performing log auditing cited insufficient resources (Section VII-B4). These findings align with prior research [53], [54], confirming that security often receives lower priority under resource constraints, even in GitHub Actions environments.

VIII. Recommendations and Implications

Action is required to address the challenges developers face in implementing security practices in GitHub Actions. Based on the findings from our measurement and user studies, we first discuss appropriate levels of automation in developer interventions and recommend suitable intervention strategies. We then propose improvements to the documentation of security practices. Finally, we discuss the limitations of our study.

A. Levels of Automation in Developer Interventions

Interventions targeting developers can vary in their level of automation, ranging from informational prompts to fully automated enforcement. While some security practices are difficult or risky to apply through complete automation due to possible side effects, higher levels of automation can still be desirable when aiming to reduce developers' workload and cognitive burden in implementing security practices.

Developers' Preference for Human-Approved Automation. A large majority of participants expressed a desire for automated platform support when performing remediation actions, such as configuration changes to repositories or workflow code (Section VII-B4). However, regarding the preferred level of such automation, the most common choice was "Human Approval", where developers retain the final decision. In contrast, only a small fraction favored fully automated approaches (i.e., "Fully Automated" and "Execute and Inform"). This tendency likely reflects developers' concerns about potential malfunctions or unintended side effects of automated remediation, as well as uncertainties regarding the operational impacts of configuration changes. In particular, participants emphasized the importance of maintaining human oversight to ensure accountability for security and operational decisions.

Potential Drawbacks of Fully Automated Interventions. Fully automated interventions for practice implementation have several potential drawbacks. For P1 (CODEOWNERS), each repository has a different development structure (e.g., responsibility for configuring and reviewing GitHub Actions), which makes it difficult to automatically determine and apply an appropriate `CODEOWNERS` configuration. For P2 (Mitigating Script Injection) and P4 (Pinning), automatically modifying code in workflow files could cause workflow malfunctions or introduce latent bugs. For P3 (Scorecard), applying this practice would require running the OpenSSF Scorecard as a workflow, thereby consuming the repository's limited free GitHub Actions runtime minutes and storage [55] and potentially leading to unintended usage of paid resources. For P5 (Dependabot), since this configuration automatically generates Pull Requests (PRs) for updating the versions of Actions, fully automating the configuration without considering developers' awareness or understanding of security practices may cause annoyance or frustration among them. This type of annoyance is often referred to as *notification fatigue*, which we further discuss in the next subsection.

B. Recommended Developer Interventions

Technical support from development platforms and tools can reduce the development and maintenance burden on projects, as well as the cognitive load on individual developers. However, as shown in Section VIII-A, developers have different preferences regarding automation, and there are potential drawbacks to excessive automation. Therefore, we recommend that relevant stakeholders adopt suitable intervention strategies that balance automation with developer autonomy.

Notification of Security Practices. Our measurement study revealed that security practices are generally underutilized across GitHub projects, regardless of contributor count. In addition, our user study found that the most common reason for not implementing security practices was simply a lack of awareness and that on average, about 46.8% of participants who became aware of the practices indicated willingness to implement them (Sections VII-B2 and VII-B4). These findings suggest that increasing developers' awareness is key to improving implementation. One promising approach is to automatically detect whether security practices are implemented in a given project and notify developers accordingly. This kind of feedback could help even less experienced developers become aware of relevant practices. The security practices we analyzed in our measurement study can be detected using open data such as code artifacts and repository metadata. Therefore, the detection methods we developed can be readily applied to build such automated support systems. We plan to release our detection tools as publicly available artifacts.

However, excessive notifications can lead to *notification fatigue*, where developers become overwhelmed and start ignoring alerts, reducing their overall effectiveness. Existing studies [56], [57], [58] have pointed out that both the design of notification messages (e.g., use of examples, indication of importance, clear language) and their alignment with development workflows (e.g., context-aware notifications, appropriate timing) significantly influence developers' acceptance and understanding of notifications. Rather than issuing notifications indiscriminately, notification systems (i.e., platform) should be carefully designed with these factors in mind. As a basic design principle, limiting notifications to repositories where the relevant security practices are applicable (as described in Section IV-B1) could enhance notification effectiveness and help mitigate fatigue.

Platform-level Support. Existing work [22] has identified the large number of manual steps required to configure such features as a barrier to adoption. Similarly, in the context of GitHub Actions, simplifying configuration steps or introducing partial automation can facilitate the practical implementation of recommended security practices. The configuration required to implement each practice varies according to its characteristics. For example, P1 requires assigning appropriate reviewers according to each repository's development structure. In contrast, P3 and P5 can be easily applied to most repositories by using configuration templates that are already available through the Web UI. Based on these observations, for P1, we recommend partially automating the currently manual process of configuring the `CODEOWNERS` file (e.g., enabling configuration file generation via the Web UI and providing reviewer-assignment suggestions based on repository structure). For P3 and P5, we further recommend enhancing the existing template-based support by introducing a "one-click activation" mechanism that enables developers to easily implement practices such as running the OpenSSF Scorecard or enabling Dependabot for managing GitHub Actions dependencies.

IDE-level Support. While platform-based support, such as automated checks and notifications on GitHub, can help developers implement security practices, it typically occurs after code is committed. This post-hoc support can be inefficient for mitigating script injection (P2) and pinning third-party actions (P4), especially in large repositories with many actions. Our measurement study revealed that repositories with more workflows were less likely to implement P2 and P4, suggesting that maintenance burdens increase with scale (Section V-B). Similarly, for P4, our user study indicated concerns about the effort required to maintain pinned SHAs (Section VII-C2). This is because pinning requires developers to specify a commit ID (SHA) for each third-party action used, and maintaining these pins can become time-consuming and error-prone, especially when actions are frequently updated. Real-time support within an integrated development environment (IDE) is essential to reduce this burden and avoid costly rework. Integrating assistance for script injection prevention and pinning third-party actions into the development workflow would enable developers to adopt the practice naturally as they code. For P2, tools that automatically suggest code that prevents script injection risks can support developers in writing secure workflow configurations. For P4, tools that automatically suggest available references (i.e., the candidate `ref` components in `owner/repo@ref`) could streamline pinning. Such support can be implemented as an extension in Visual Studio Code (an IDE tightly integrated with GitHub) and enhanced with AI tools, which approximately one-third of our user study participants reported using.

C. Improvements for Documentation

Regardless of the level of interventions or automation provided to developers, documentation of security practices remains essential, as it explains the role and significance of each practice. However, our study revealed that such documentation contains several issues that hinder developers' understanding and implementation of these practices. Our user study revealed that awareness of GitHub Actions security practices was generally low. Participants also indicated uncertainty about which risks each practice addresses, difficulty understanding which practices apply to their own projects, and misconceptions that discouraged implementation. To address these issues, we propose improvements to documentation.

Mapping Security Practices to Risks. In our user study, approximately one-third of participants indicated that a key factor in implementing a security practice is its clear effectiveness in mitigating relevant risks (Section VII-B3). Additionally, in

TABLE V
RECOMMENDED SCOPE FOR EACH SECURITY PRACTICE.

Security practice	Target
P1 CODEOWNERS	All Repositories
P2 Mitigating Script Injection	Workflows using github context
P3 OpenSSF Scorecard	All public repositories
P4 Pinning Third-party Actions	All repositories using third-party actions
P5 Dependabot	All repositories using public actions (for dependency management)

our pilot study, some participants noted that it was unclear which risks each practice addressed. These findings highlight the importance of explicitly mapping each security practice to the specific risks it mitigates. Such clarity helps developers understand the value of each practice and make informed decisions about implementation. As shown in Section III-B2, referencing actual security incidents helps make these risks more concrete and clarifies the importance of each practice.

Guidance Feature Indicating Who Should Use Each Practice. Our findings suggest that misconceptions can hinder the implementation of security practices. In our user study, several participants mentioned their project's programming language as a reason for not using Dependabot (Section VII-C2). However, in the context of GitHub Actions, Dependabot automatically updates public actions regardless of the project's language. Therefore, any repository using public actions should benefit from enabling this feature. GitHub's documentation for the platform's built-in features (e.g., branch protection rules [59]) includes a *"Who can use this feature?"* section that clearly specifies applicable users and conditions. While this guidance applies to built-in features, a similar format could help clarify when each security practice is relevant, reducing misconceptions and aiding implementation decisions. For reference, Table V outlines the recommended scope for each security practice examined in this study.

D. Limitations

Generalizability and Applicability. This study analyzed 5 of the 16 security practices, with two additional practices examined in the user study, selected based on their observability and the participant burden and feasibility of obtaining valid responses. Because each practice has distinct use cases and implementation procedures, the findings may not fully generalize to the excluded ones. Additionally, while our mixed-methods approach provides a comprehensive understanding of the current state of security practice, it necessarily constrained the set of practices that could be analyzed. Conducting the measurement or user study independently would allow the scope of practices to be expanded.

Sample Size of Participants. Recruiting expert participants, particularly software developers, remains a well-known challenge in empirical software engineering research. Generally, studies involving software developers tend to have smaller sample sizes than those involving end-users. However, our sample size (N=102) exceeds that of prior comparable studies involving software developers on GitHub (e.g., N=80–90 [22], [60]). Accordingly, our analysis is based on descriptive statistics and qualitative coding of open-ended responses, which are standard practices in studies with limited samples.

Analysis of the Developers' Roles and Responsibilities. Our user study targeted developers with experience configuring GitHub Actions, but it did not distinguish participants' specific roles or responsibilities related to GitHub Actions. Although most participants (88.2%) were repository owners or admins, suggesting that they had particular responsibility for configuration and operation, their actual responsibilities in practice may vary. Accounting for more fine-grained roles and responsibilities (e.g., security decision-making authority and role division within the team) may yield more profound insights into factors that hinder the implementation of security practices and represent an important direction for future work.

IX. CONCLUSION

Modern CI/CD pipelines have become both critical infrastructure and attractive targets for supply chain attacks. Our mixed-methods study combining quantitative analysis of GitHub repositories with qualitative developer surveys reveals critically low security practice implementation in GitHub Actions. The quantitative analysis identified repository characteristics associated with implementation, while developer surveys uncovered three primary barriers: lack of awareness, misconceptions about applicability, and maintenance concerns. These complementary findings form the basis for our conclusion that improving security requires intervention strategies aligned with appropriate levels of automation, the introduction of well-designed notifications to raise awareness, strengthened platform- and IDE-level assistance to support practical implementation, and clearer documentation that maps practices to risks and applicability. Future work should leverage mixed-methods approaches to track adoption longitudinally, expand to other CI/CD platforms, and evaluate proposed interventions.

REFERENCES

[1] SolarWinds, "Solarwinds security advisory," (2025-08-02 accessed). [Online]. Available: https://www.solarwinds.com/sa-overview/securityadvisory

[2] M. Golzadeh, A. Decan, and T. Mens, "On the rise and fall of CI services in GitHub," in *Proceedings of the 29th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2022, pp. 679–690.

[3] H. Sheth, "How GitHub Actions won CI," May 2024, blog post. [Online]. Available: https://harshal.sheth.io/2024/05/13/github-actions.html

[4] GitHub Advisory Database, "CVE-2025-30066: tj-actions/changed-files Compromised by Malicious Code Injection," March 2025, accessed: 2025-08-06. [Online]. Available: https://github.com/advisories/GHSA-mrrh-fwg8-r2c3

[5] 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.

[6] Y. Gu, L. Ying, H. Chai, C. Qiao, H. Duan, and X. Gao, "Continuous intrusion: Characterizing the security of continuous integration services," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 1561–1577.

[7] GitHub, "Secure use reference," (2025-08-03 accessed). [Online]. Available: https://docs.github.com/en/actions/reference/security/secure-use

[8] 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.

[9] Y. Kubo, F. Kanei, M. Akiyama, T. Wakai, and T. Mori, "Artifact - action required: A mixed-methods study of security practices in github actions," Dec. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.17916232

[10] GitHub, "About github actions," (2025-08-02 accessed). [Online]. Available: https://docs.github.com/en/actions/about-github-actions

[11] GitLab, (2025-08-02 accessed). [Online]. Available: https://about.gitlab.com/

[12] CircleCI, (2025-08-02 accessed). [Online]. Available: https://circleci.com/

[13] TravisCI, (2025-08-02 accessed). [Online]. Available: https://www.travis-ci.com/

[14] GitHub, "About workflows," (2025-08-02 accessed). [Online]. Available: https://docs.github.com/en/actions/writing-workflows/about-workflows

[15] ——, "Context reference," (2025-08-05 accessed). [Online]. Available: https://docs.github.com/en/actions/reference/workflows-and-actions/contexts

[16] ——, "About custom actions," (2025-08-05 accessed). [Online]. Available: https://docs.github.com/en/actions/concepts/workflows-and-actions/custom-actions

[17] ——, "Marketplace," (2025-08-02 accessed). [Online]. Available: https://github.com/marketplace

[18] ——, "Security log events," (2025-08-06 accessed). [Online]. Available: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/security-log-events

[19] ——, "Reviewing the audit log for your organization," (2025-08-06 accessed). [Online]. Available: https://docs.github.com/en/organizations/keeping-your-organization-secure/managing-security-settings-for-your-organization/reviewing-the-audit-log-for-your-organization

[20] 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*, 2022, pp. 25–34.

[21] 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)*, 2023, pp. 2391–2408.

[22] J. Ayala, Y.-J. Tung, and J. Garcia, "A mixed-methods study of open-source software maintainers on vulnerability management and platform security features," in *34th USENIX Security Symposium (USENIX Security 25)*, 2025.

[23] GitHub, "About code owners," (2025-08-02 accessed). [Online]. Available: https://docs.github.com/en/repositories/managing-your-repositorys-\\settings-and-features/customizing-your-repository/about-code-owners

[24] GitGuardian, "The ghostaction campaign: 3,325 secrets stolen through compromised github workflows," (2025-11-18 accessed). [Online]. Available: https://blog.gitguardian.com/ghostaction-campaign-3-325-secrets-stolen/

[25] GitHub, "Malicious versions of nx and some supporting plugins were published," (2025-11-18 accessed). [Online]. Available: https://github.com/nrwl/nx/security/advisories/GHSA-cxm3-wv7p-598c

[26] OpenSSF, "Openssf scorecard," (2025-08-02 accessed). [Online]. Available: https://openssf.org/projects/scorecard/

[27] GitHub, "Scorecards' github action," (2025-08-02 accessed). [Online]. Available: https://github.com/marketplace/actions/ossf-scorecard-action

[28] ——, "Starter workflows," (2025-08-05 accessed). [Online]. Available: https://github.com/actions/starter-workflows

[29] ——, "Keeping your supply chain secure with dependabot," (2025-08-02 accessed). [Online]. Available: https://docs.github.com/en/code-security/dependabot

[30] ——, "Github advisory database," (2025-11-18 accessed). [Online]. Available: https://github.com/advisories

[31] P. Valenzuela-Toledo, A. Bergel, T. Kehrer, and O. Nierstrasz, " The Hidden Costs of Automation: An Empirical Study on GitHub Actions Workflow Maintenance ," in *2024 IEEE International Conference on Source Code Analysis and Manipulation (SCAM)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2024, pp. 213–223.

[32] A. Decan, T. Mens, P. R. Mazrae, and M. Golzadeh, "On the use of github actions in software development repositories," in *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2022, pp. 235–245.

[33] T. Chen, Y. Zhang, S. Chen, T. Wang, and Y. Wu, "Let's supercharge the workflows: An empirical study of GitHub actions," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, Dec. 2021, pp. 01–10.

[34] GitHub, "Getting started with github pages," (2025-08-02 accessed). [Online]. Available: https://docs.github.com/en/pages/getting-started-with-github-pages

[35] GitHub, "Verifying or approving a domain for your organization," (2025-08-05 accessed). [Online]. Available: https://docs.github.com/en/organizations/managing-organization-settings/verifying-or-approving-a-domain-for-your-organization

[36] S. Katcher, L. Wang, C. Yang, C. Messdaghi, M. L. Mazurek, M. Chetty, K. R. Fulton, and D. Votipka, "A survey of cybersecurity professionals' perceptions and experiences of safety and belonging in the community," in *Proceedings of the Twentieth USENIX Conference on Usable Privacy and Security*, ser. SOUPS '24. USA: USENIX Association, 2024.

[37] C. Guo, B. Campbell, A. Kapadia, M. K. Reiter, and K. Caine, "Effect of mood, location, trust, and presence of others on Video-Based social authentication," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1–18.

[38] W. Cao, C. Xia, S. T. Peddinti, D. Lie, N. Taft, and L. M. Austin, "A large scale study of user behavior, expectations and engagement with android permissions," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 803–820.

[39] T. Kinsman, M. Wessel, M. A. Gerosa, and C. Treude, "How do software developers use github actions to automate their workflows?" in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 2021, pp. 420–431.

[40] D. Votipka, D. Abrokwa, and M. L. Mazurek, "Building and validating a scale for secure software development self-efficacy," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–20.

[41] S. G. Saroar and M. Nayebi, "Developers' perception of github actions: A survey analysis," in *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 121–130.

[42] M. Wessel, I. Wiese, I. Steinmacher, and M. A. Gerosa, "Don't disturb me: Challenges of interacting with software bots on open source software projects," *Proc. ACM Hum.-Comput. Interact.*, vol. 5, no. CSCW2, Oct. 2021.

[43] N. Roch, H. Sievers, L. Schöni, and V. Zimmermann, "Navigating autonomy: unveiling security experts' perspectives on augmented intelligence in cybersecurity," in *Proceedings of the Twentieth USENIX Conference on Usable Privacy and Security*, ser. SOUPS '24. USA: USENIX Association, 2024.

[44] Qualtrics XM, (2025-08-03 accessed). [Online]. Available: https://www.qualtrics.com/

[45] A.-M. Ortloff, C. Tiefenau, and M. Smith, "Sok: i have the (developer) power! sample size estimation for fisher's exact, chi-squared, mcnemar's, wilcoxon rank-sum, wilcoxon signed-rank and t-tests in developer-centered usable security," in *Proceedings of the Nineteenth USENIX Conference on Usable Privacy and Security*, ser. SOUPS '23. USA: USENIX Association, 2023.

[46] N. McDonald, S. Schoenebeck, and A. Forte, "Reliability and inter-rater reliability in qualitative research: Norms and guidelines for cscw and hci practice," *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. CSCW, Nov. 2019.

[47] J. T. Liang, T. Zimmermann, and D. Ford, "Understanding skills for oss communities on github," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 170–182.

[48] Y. Huang, D. Ford, and T. Zimmermann, "Leaving my fingerprints: Motivations and challenges of contributing to oss for social good," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 1020–1032.

[49] H. Kaur, S. Klivan, D. Votipka, Y. Acar, and S. Fahl, "Where to recruit for security development studies: Comparing six software developer

samples," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 4041–4058.
[50] "Freelancer.com," (2025-08-06 accessed). [Online]. Available: https://www.freelancer.com/
[51] "Renovatebot / renovate," (2025-08-05 accessed). [Online]. Available: https://github.com/renovatebot/renovate
[52] GitHub, "Disabling or limiting github actions for your organization," (2025-08-05 accessed). [Online]. Available: https://docs.github.com/en/organizations/managing-organization-settings/disabling-or-limiting-github-actions-for-your-organization
[53] M. Tahaei and K. Vaniea, "A survey on developer-centred security," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, 2019, pp. 129–138.
[54] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, and K. Kinder-Kurlanda, "Can security become a routine? a study of organizational change in an agile software development group," in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, ser. CSCW '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 2489–2503.
[55] GitHub, "Github actions billing," (2025-10-24 accessed). [Online]. Available: https://docs.github.com/en/billing/concepts/product-billing/github-actions
[56] M. Tahaei, K. Vaniea, K. K. Beznosov, and M. K. Wolters, "Security notifications in static analysis tools: Developers' attitudes, comprehension, and ability to act on them," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021.
[57] A. Danilova, A. Naiakshina, and M. Smith, "One size does not fit all: a grounded theory and online survey study of developer preferences for security warning types," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 136–148.
[58] M. Wessel, I. Wiese, I. Steinmacher, and M. A. Gerosa, "Don't disturb me: Challenges of interacting with software bots on open source software projects," *Proc. ACM Hum.-Comput. Interact.*, vol. 5, no. CSCW2, Oct. 2021.
[59] GitHub, "Managing a branch protection rule," (2025-08-05 accessed). [Online]. Available: https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/managing-a-branch-protection-rule
[60] S. G. Saroar and M. Nayebi, "Developers' perception of github actions: A survey analysis," in *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 121–130.
[61] C. Utz, S. Amft, M. Degeling, T. Holz, S. Fahl, and F. Schaub, "Privacy rarely considered: Exploring considerations in the adoption of third-party services by websites," *Proc. Priv. Enhancing Technol.*, vol. 2023, no. 1, pp. 5–28, 2023.
[62] GitHub, "Towards a secure by default github actions," (2025-11-20 accessed). [Online]. Available: https://github.com/orgs/community/discussions/179107
[63] ——, "Configuring default setup for code scanning," (2025-08-06 accessed). [Online]. Available: https://docs.github.com/en/code-security/code-scanning/enabling-code-scanning/configuring-default-setup-for-code-scanning
[64] L. K. Organization, "gitrevisions(7) manual page," (2025-08-06 accessed). [Online]. Available: https://www.kernel.org/pub/software/scm/git/docs/gitrevisions.html

APPENDIX

A. Ethical Considerations

We designed and conducted both the measurement study and user study in accordance with the ethical guidelines outlined in the Menlo Report and the research ethics policies of the authors' affiliated institutions. We confirmed that both studies are exempt from Institutional Review Board (IRB) review.

1) Measurement Study: The measurement study was conducted with minimal ethical risk by employing non-intrusive methods and reducing load on target servers.

Minimizing Server Load. To avoid exceeding GitHub API rate limits, our study controlled the number of API requests sent to target repositories. Specifically, we monitored the rate limit information included in API responses and paused requests once fewer than 100 requests remained until the rate limit reset (which occurs every hour). Moreover, we limited the collected data to the minimum necessary for analysis. We utilized a public dataset (SEART-GHS) for regression analysis metadata to minimize additional data collection.

Non-intrusive Testing. We cloned target repositories locally for code and file analysis without making any direct modifications or writes to the repositories on GitHub.

2) User Study: In the user study, we carefully considered reducing participant burden, obtaining informed consent, and protecting participant privacy and anonymity.

Recruiting. Mass emailing by mechanically harvesting developer emails from platforms raises ethical concerns [61] and can cause survey fatigue. To mitigate this, we filtered potential participants based on specific criteria to target developers appropriate for our study. Specifically, we selected developers involved in repositories using GitHub Actions who publicly disclosed their email addresses (identified via GitHub REST API). Consequently, recruitment emails were sent to 1,800 developers, yielding 102 valid responses (5.7%). Although direct comparison with prior work is difficult due to inconsistent reporting of recruitment volumes, our targeted approach helped avoid unnecessary contact with developers unlikely to participate, reducing survey fatigue.

Informed Consent. Participants were provided with information about the study's purpose, target audience, data handling, incentives (summary of results shared upon request), and the option to withdraw at any time. We obtained consent via a checkbox before proceeding with the questionnaire.

Privacy and Anonymity. We did not collect personally identifiable or sensitive information. Participants were encouraged to skip any questions they preferred not to answer due to their roles or affiliations. We securely stored collected data with access restricted to the authors.

Incentive. We provided no monetary compensation. Participants were offered a summary of the survey results as an incentive. This non-monetary approach aligns with common practice in professional studies and relies on voluntary participation. Positive feedback from participants indicated that the study itself served as a learning opportunity about security practices, acting as an intrinsic incentive.

3) Disclosure and Feedback from GitHub: We shared our findings and recommendations with GitHub in October 2025. GitHub provided high-level feedback in November 2025 that our results, particularly RQ3 on barriers to implementing security practices, align with their ongoing shift toward strengthening default security protections in GitHub Actions rather than relying solely on individual developer effort. They noted that several areas identified in our study correspond with features already planned or under development, such as automated security warnings and improved workflow linting. These topics are also being actively discussed in their community forum,

particularly in a community discussion on making GitHub Actions more secure by default, which began in November 2025 [62]. GitHub further expressed appreciation for research that raises awareness of Actions security and indicated that our insights may help inform future platform-level improvements.

Conversely, notifying individual repositories about missing security practices remains impractical due to the absence of immediate critical vulnerabilities in most cases and the large number of repositories involved. Therefore, our disclosure engagement focuses on contributing to ecosystem-wide, platform-level improvements through GitHub rather than targeting individual repositories.

B. Updates in GitHub's Security Practice Documentation

The contents of the security practices published by GitHub were updated after July 11, 2025. However, the practices investigated in this study remain unchanged, and therefore, the update does not affect our study. The main changes are: (1) modification of the web page's file path, (2) categorization of practices into two types (workflow writing and security features), (3) addition of practices related to dependency management, and (4) removal of several items (ID 3, 12, and 13 in Table VI). The version of the documentation used in this study can be accessed from the following Web Archive: https://web.archive.org/web/20250701141038/https://docs.github.com/en/actions/how-tos/security-for-github-actions/security-guides/security-hardening-for-github-actions

C. Rationale for Security Practice Selection

Table VI presents the evaluation results for each practice based on the criteria in Section III-B1. The excluded practices and the reasons for their exclusion are as follows:

IDs 3, 12, and 14: These practices correspond to background explanations or developer warnings (e.g., the mechanism of script injection or the behavior of GitHub-hosted runners). Since their purpose is to provide knowledge rather than concrete actions, they do not satisfy criteria 1.

IDs 9 and 16: These practices are configurable from the repository's settings page and are not visible in the codebase. Therefore, their implementation cannot be observed from open data, and they do not satisfy criteria 2.

ID 15: While the use of self-hosted runners can be analyzed, it is not possible to assess the use of JIT runners or runner management strategies from open data. Hence, this practice does not meet criteria 2.

IDs 1 and 5: The implementation of practices related to Secrets and OpenID Connect can be observed using open data. However, it is not possible to identify repositories that are expected to implement these practices but do not, meaning that we cannot know the target population required to analyze implementation rates. Therefore, these practices do not satisfy Criterion 3.

ID 10: Code scanning can be enabled by adding a workflow file, even without elevated privileges. However, GitHub's official documentation [63] notes that high privileges are required and assumes privileged users for setup. Thus, it does not satisfy criteria 4. In addition, a review of the practice history using Web Archive shows that this item was introduced between December 17 and 22, 2024. Because our dataset consists of repositories created before December 31, 2024, and our studies were conducted in May–July 2025, the practice had been available for too short a period for meaningful adoption or developer experience, making it unsuitable for analysis.

D. P4: Action-Level Detection Method

The method for determining action reference strings in the analysis is shown in Table VII, and the method for determining practice implementation is shown in Table VIII.

E. Variance Inflation Factor (VIF)

We assessed multicollinearity among the independent variables used in our regression analysis by calculating their Variance Inflation Factor (VIF) values. Table IX presents the VIF values for each independent variable used in the logistic regression models. Each column corresponds to a model built for a different security practice, and each row represents a specific independent variable. Although there is no universally accepted threshold for VIF values, prior studies [36], [37], [38] have often regarded a range below 3 or 5 as acceptable, considering values exceeding that range as indicative of excessive correlation among independent variables. In this study, all independent variables had VIF values below 2, suggesting that multicollinearity is not a concern.

F. Summarizing Documentation of Security Practices

We developed the descriptions of each security practice (Section III-B) through the following procedure, and used them in the questionnaire for the user study.

First, one of the authors drafted the descriptions by summarizing the official documentation on GitHub Actions security practices [7], based on the following policies: (1) quoting original text from the official documentation as much as possible; (2) supplementing the explanations by including information from documents linked within one hop of the original page (e.g., definitions of technical terms or explanations of features that the practice depends on); (3) incorporating any descriptions of the practice's effectiveness or the threats/risks it is intended to mitigate, if such information was provided in the source. The drafted descriptions were then reviewed and revised iteratively by two security experts with experience using GitHub Actions. Their review focused on ensuring that the descriptions did not contain overstatements (i.e., information not present in the original sources) or understatements (i.e., omission of important information), and that a wide range of participants would be able to answer subsequent questions about security practices after reading the text. Each expert independently conducted the review. When one expert proposed a revision, the final version was confirmed only after the other expert agreed with the proposed change. This process of review and revision was repeated a total of four times until both experts fully agreed on the final set of descriptions.

TABLE VI
CRITERIA-BASED SELECTION OF GITHUB-DEFINED SECURITY PRACTICE.

ID	Security Practice	Select	Type	Detectability	Eligibility	High Privilege
1	Using Secrets		Setting	True	False	True
2	Using CODEOWNERS to monitor changes	✓	Feature	True	True	False
3	Understanding the risk of script injections		Knowledge	-	-	-
4	Good practices for mitigating script injection attacks	✓	Coding	True	True	False
5	Using OpenID Connect to access cloud resources		Setting	True	False	False
6	Using third-party actions	✓	Coding	True	True	False
7	Reusing third-party workflows		Coding	True	True	False
8	Using Dependabot version updates to keep actions up to date	✓	Tool	True	True	False
9	Preventing GitHub Actions from creating or approving pull requests		Setting	False	-	True
10	Using code scanning to secure workflows		Tool	True	True	True
11	Using OpenSSF Scorecards to secure workflow dependencies	✓	Tool	True	True	False
12	Potential impact of a compromised runner		Knowledge	-	-	-
13	Considering cross-repository access		Setting	False	-	True
14	Hardening for GitHub-hosted runners		Knowledge	-	-	-
15	Hardening for self-hosted runners		Setting	False	-	True
16	Auditing GitHub Actions events		Feature	False	-	True

TABLE VII
CRITERIA FOR DETERMINING HOW ACTIONS ARE REFERENCED.

Reference	Criteria
SHA	40-character hash.
Tag	Matches a tag in the action repository.
Branch	Matches a branch in the action repository.
Default	Not specified; defaults to the repository's default branch.
ShortSHA	Prefix of a valid commit hash, verified via GitHub API.
NotFound	No match above, or the action repository not found.

To ensure accuracy, the classification for Tag and Branch is based on the actual list of releases (tags) and branches retrieved from the action repository. If a reference string matches both a tag and a branch, it is classified as a Tag in accordance with Git's specification [64].

TABLE VIII
P4: ACTION-LEVEL IMPLEMENTATION CRITERIA.

Verification Status	Reference Type	Implemented
Verified	SHA	True
	Tag	True
	Others	False
Not Verified	SHA	True
	Tag	False
	Others	False

Others includes reference patterns such as Branch, Default, ShortSHA, and NotFound (references to non-existent actions).

G. Recruitment Criteria

During the recruitment process for the user study, we collected contact information for eligible developers from GitHub. To avoid potential negative effects (e.g., recruiting developers who are not relevant to the study objectives or causing survey fatigue by sending invitations to a large number of developers), we employed the following filtering procedure to narrow down the target population.

First, we retrieved the top 100 contributors (ranked by number of contributions) from each repository in our dataset (Section IV-A). This process yielded a total of 1,439,072 GitHub accounts. Next, we applied the following three criteria to filter the extracted accounts:

TABLE IX
VIF VALUES ACROSS FIVE LOGISTIC REGRESSION MODELS.

Independent Variables	P1	P2	P3	P4	P5
Number of Stars	1.23	1.27	1.23	1.26	1.23
Number of Contributors	1.75	1.91	1.75	1.83	1.76
Number of Commits	1.13	1.17	1.13	1.14	1.13
Codebase Size	1.04	1.08	1.04	1.03	1.04
Repository Age	1.07	1.10	1.07	1.08	1.07
Recent Activity	1.06	1.03	1.06	1.05	1.06
Number of Workflow Files	1.16	1.13	1.16	1.16	1.17
Number of Workflow Developers	1.58	1.65	1.58	1.64	1.60
Owner Type	1.10	1.11	1.10	1.11	1.10

(1) Accounts that explicitly publish a contact email address on their profile: We retrieved the profile information of each candidate account and included only those that explicitly displayed a contact email address. Prior research has raised ethical concerns regarding the use of email addresses collected automatically from Git commit logs for recruitment purposes [61]. In consideration of this, we limited our recruitment targets to developers who had intentionally made their contact information publicly available.

(2) Accounts involved in configuring GitHub Actions: We identified accounts that were actively involved in configuring GitHub Actions based on their contribution history. Specifically, we included accounts that had made 50 or more contributions related to GitHub Actions (e.g., commits to workflow files) and had contributed to GitHub Actions at least once after January 1, 2025.

(3) Accounts that are not bots: Since some GitHub accounts are operated automatically via the API (i.e., bots), such accounts were excluded as inappropriate for participation in the study. Specifically, we excluded accounts that either contained the string "bot" in the username or were labeled as type "Bot" in metadata retrieved via the GitHub REST API.

From the 3,405 eligible accounts, we randomly sampled 1,800 accounts, with half primarily contributing to individual-owned repositories and the other half to organization-owned repositories, and sent recruitment emails to their publicly listed email addresses.

APPENDIX A
ARTIFACT APPENDIX

This study conducted a mixed-method study to investigate the implementation status of security practices in GitHub Actions and identified the challenges associated with their use. Specifically, we combined a measurement study analyzing public repositories using open data with a user study based on a developer survey.

This artifact provides the scripts used in the measurement study. These include the scripts used for dataset creation (Section IV-A) and for analyzing security practices (Sections IV-B and V-A). We also describe how to use these scripts. These scripts can be used not only to reproduce the measurement study presented in this research but also as a foundation for future studies on GitHub Actions. Furthermore, developers can use them to evaluate their own repositories and to assess and improve the implementation of security practices.

A. Description & Requirements

Our artifact consists of a set of Python and Shell scripts that run on a local machine to collect and store data from GitHub and perform subsequent analyses on the collected data.

1) How to access: Our artifact is available at https://github.com/yksec14/gha_security_research and citable via the DOI https://doi.org/10.5281/zenodo.17916232.

2) Hardware dependencies: The artifact has no specific hardware requirements. It can be run on commodity hardware.

3) Software dependencies: The following software requirements are based on the verified environment.

- **OS:** Ubuntu 22.04. The scripts are expected to work on other Unix-based systems such as macOS as well.
- **Python:** Version 3.13.9. All required Python packages are listed in `requirements.txt`.
- **API Key:** Fine-grained Personal Access Token (GitHub). If you do not have a GitHub account, please create one first.

4) Benchmarks: We provide benchmark repositories that reproduce typical implementation patterns of the security practices analyzed in experiment E2. A small example dataset, including five repositories and their corresponding analysis results, is available in the `example_data` directory in the artifact. The analysis workflow of experiment E2 can be tested by running the provided script `test_example.sh`. For detailed descriptions and usage instructions, please refer to the `README.md`.

B. Artifact Installation & Configuration

Clone the artifact from the GitHub repository (https://github.com/yksec14/gha_security_research) or Download zip file from DOI link (https://doi.org/10.5281/zenodo.17916232). The repository contains a `README.md` file with step-by-step instructions for installation and dependency setup.

C. Experiment Workflow

The artifact consists of two experiments: (E1) repository dataset creation and (E2) analysis of security practices. In E1, repositories using GitHub Actions are collected through the GitHub API, and the resulting dataset is stored locally. E2 analyzes the collected repositories to examine the adoption of five security practices. Both experiments are implemented as Python scripts that can be executed sequentially.

The experiments in this artifact are designed as minimal examples to satisfy the requirement that experiments should complete within one day and run on a commodity desktop machine, while still allowing verification of the main results. Specifically, the target repositories are limited to those created in 2024, and up to 50 repositories created in each month are analyzed, resulting in at most 600 repositories included in the analysis.

D. Major Claims

This artifact supports the following claims made in our paper:

- (C1): Based on SEART-GHS (SEART GitHub Search Engine), we constructed a dataset of repositories that use GitHub Actions (Section IV-A).
- (C2): We developed an analysis framework that determines, for five security practices observable from public data, whether a given repository is subject to each practice and whether it is actually implemented (Section IV-B).
- (C3): The analysis shows that the implementation rates of the examined security practices are low among public repositories (Section V-A, Table II).

E. Evaluation

All evaluation procedures are performed via the terminal application, with the working directory set to the root directory of the artifact. Please refer to the `README.md` file for detailed instructions on each step.

1) Experiment (E1): [repository dataset creation] [5 human-minutes + 50 compute-minutes]: This experiment collects repositories created in 2024 and identifies those that use GitHub Actions by querying the GitHub API. The identified repositories are cloned to the local environment, producing a dataset that includes both a list of repositories using GitHub Actions and their cloned copies, which are used for subsequent analysis in Experiment 2.

[Preparation] Access SEART-GHS[5] and specify the "Created Between" period as January 1, 2024 to December 31, 2024 to search for repositories. This fixed period defines the scope of repositories analyzed in this evaluation. Once the search results are displayed, click "Download CSV" to download the results in CSV format. The file will be downloaded as a tar.gz archive; please extract it and save the resulting file as `./data/raw/results.csv`.

[Execution] Run the following command.

```
$ ./create_dataset.sh 2024-01 2024-12
```

[5]https://seart-ghs.si.usi.ch/

[Results] When executed, the script outputs debug messages during the process, and a brief summary is displayed at the end.

```
=== GitHub Actions Usage Summary ===
Total Repositories: 596
Repositories Using GitHub Actions: 192
GitHub Actions Usage Percentage: 32.21%
```

The collected data are stored locally. The list of repositories identified as using GitHub Actions is saved in JSON format under `./data/dataset/gha_check/`, organized by month. Each repository is cloned following the structure `./data/dataset/cloned_repos/{owner}/{repository}`.

2) Experiment (E2): [analysis of security practices] [1 human-minutes + 15 compute-minutes]: This experiment analyzes the repositories collected in experiment E1 to evaluate the adoption of five security practices. For each repository, the analysis framework inspects relevant configuration and workflow files to determine whether each practice is applicable and whether it is actually implemented. The resulting data represent the implementation rates of the five practices, which serve as the basis for Table II in the paper.

[Preparation] Run the following command in the terminal from the project's root directory.

```
./pre_analysis.sh 2024-01 2024-12
```

[Execution] Run the following command.

```
./analyze_security_practices.sh 2024-01
    2024-12
```

[Results] When executed, the script outputs debug messages during the process, and a brief summary is displayed at the end. The resulting data represent the implementation rates of each security practices.

```
=== Security Practice Implementation Results
    ===
practice1:
  Target Repositories: 192
  Implemented Repositories: 17
  Implementation Rate: 8.85%

practice2:
  Target Repositories: 39
  Implemented Repositories: 18
  Implementation Rate: 46.15%

...
```

The results are stored locally under `./data/analyzed_data/`. Each security practice has its own directory (e.g., `./data/analyzed_data/practice1`) where the results are organized by month.

F. Customization

The experiment is designed with a minimal configuration that satisfies the requirements of the Artifact Evaluation. By modifying several options, it is possible to run large-scale experiments comparable to those conducted in the main study. (1) Setting the value of the `DEBUG` variable in `settings.py` to `False` removes the limit of 50 repositories per month. (2) The parameter specifying the target period can be changed from the fixed year 2024 to any arbitrary time range.

Note that these modifications increase both hardware and API requirements. In our main study, analyzing 338,812 repositories required approximately 30 TB of storage. Additional attention is needed regarding the GitHub API. The standard rate limit for general users is 5,000 requests per hour, which results in extremely long execution times. For large-scale experiments, we recommend using the enterprise plan, which increases the rate limit to 15,000 requests per hour.

G. Notes

This artifact package additionally contains supplementary materials for our user study, such as the recruitment email, questionnaire, and the complete set of study results. These materials are available under the `supplementary_materials` directory within the artifact package.