# Time for Actions: A Longitudinal Study of the GitHub Actions Marketplace

Narong Chaiwut
*Department of Computer Science*
*Stony Brook University*
New York, USA
nchaiwut@cs.stonybrook.edu

Nick Nikiforakis
*Department of Computer Science*
*Stony Brook University*
New York, USA
nick@cs.stonybrook.edu

***Abstract*—GitHub Workflows have emerged as one of the most widely adopted CI/CD platforms, with millions of workflows integrated into modern software development. With the ability to encapsulate a subset of workflow functionality into reusable components known as GitHub Actions, any developer can publish their actions to the GitHub Actions Marketplace. To date, there has been no comprehensive analysis of the GitHub Actions Marketplace in terms of the number of actions, their growth over time, and their susceptibility to supply-chain attacks.**

**In this paper, we present a longitudinal study of GitHub actions, aiming to better understand this important, modern software ecosystem. Over a period of four months, we collected and analyzed over 23K GitHub Actions including their source code as well as metadata about their creators. Our analysis investigates marketplace trends, identifies prevalent security issues in GitHub Actions, and characterizes novel attack vectors, including actions typosquatting and dangling remote references.**

***Index Terms*—CI/CD, Github Actions, remote references.**

## I. Introduction

The software development process has shifted from linear models such as the Waterfall model to more iterative and automated approaches like DevOps. DevOps refers to the automation of processes across various tools and stakeholders—including development, deployment, and infrastructure—to enable rapid building, testing, deployment, and monitoring [1]. Many platforms now provide Continuous Integration and Continuous Deployment (CI/CD) as part of the DevOps toolchain, such as Jenkins [2], GitLab CI [3], and GitHub Actions [4]. Among the multitude of available CI/CD platforms, GitHub Actions is one of the most widely adopted ones [5], [6].

GitHub introduced GitHub Actions in 2019, and it quickly gained popularity due to its tight integration with GitHub's code hosting platform. This integration allows developers to define CI/CD pipelines directly within their repositories without needing to use an external platform. To reduce workflow duplication, GitHub started supporting reusable workflows and modular steps through third-party *actions*. These actions can be created and shared via a centralized ecosystem of reusable actions called GitHub Actions Marketplace [7].

However, the ability to integrate third-party actions into one's workflows introduces a new attack vector related to software supply-chain security. Malicious actors can publish harmful actions to the Marketplace or compromise existing ones to inject malicious code. In March 2025, industry researchers detected anomalous URL requests originating from the `tj-actions/changed-files` action, which was used by over 24,000 repositories [8]. Further investigations [9], [10] revealed that attackers were targeting sensitive repositories, including those belonging to Coinbase. This incident highlights the importance of analyzing the security posture of GitHub Actions, just as we analyze the posture of general third-party libraries and packages.

In this paper, we present a longitudinal study analyzing the GitHub Actions ecosystem. We perform week-by-week crawling of the GitHub Actions Marketplace to collect metadata on all available actions, including new and removed entries. This enables us to identify trends in the adoption of actions, observe new and deleted actions, and detect potential security issues.

Over a four-month period, we collected a total of 23,757 GitHub Actions. Our analysis shows that the overall number of actions exhibits a modest increase over time with more actions added every week than the ones deleted from the marketplace. Beyond ecosystem growth, we also analyzed security-related aspects of newly added and removed actions from the Marketplace, focusing on the way that developers integrate third-party actions in their code. The results reveal that the most frequently observed security issue was the absence of full-length commit SHA pinning (allowing attackers to compromise repos via their actions dependencies), with an average of 304 instances in newly added actions and 72 in removed actions across our observation period. Similarly, we also examined remote references within actions – including external URLs and public IP addresses – to identify unreachable hosts, dead links, and potential domain misuse, finding 732 affected actions.

Since developers are the ones who choose to integrate with third-party actions, we also study the levels of typosquatting abuse on the Actions marketplace. There, attackers register typo variants of popular actions with the hopes of capitalizing on a developer's typographic errors or confusion as to which one is the authoritative action. We identified 25 typosquatting repositories, many of which targeted popular actions such as `actions/checkout`.

Our contributions are summarized as follows:

- We developed custom crawlers and longitudinally tracked the GitHub actions marketplace over a four-month period, obtaining all code and data from that marketplace.
- We used our collected data to analyze trends in newly added, removed, and total actions, checking these actions against known software-supply-chain issues.
- We conducted the first systematic investigation of typosquatting and remote reference risks (e.g., unreachable URLs, parked domains, and dead IP hosts) in GitHub Actions, quantifying the potential takeover of the affected actions.

## II. BACKGROUND

### A. Github Actions

GitHub Actions refer to automated workflows designed to support DevOps processes. Through actions, developers can run specific workflows during commits, builds, and deployments of their source code [4]. To set up these workflows, developers need to create a `.github/workflows` directory and define workflow files using the `.yml` format. Fig. 1 provides a high-level overview of GitHub Actions, which consist of four primary components: workflows, jobs, steps, and actions.

**Workflow:** [12] A workflow is the top-level definition of the CI/CD pipeline. It allows developers to define and customize the automation process using the `on` keyword to specify triggers. Triggers can be based on time intervals (e.g., scheduled runs), code pushes, or pull requests.

**Job:** [13] A job represents a set of tasks that can be executed either sequentially or in parallel. Jobs are defined under the `jobs` section and are commonly used for tasks such as building code, running tests, or deploying to external services. GitHub provides isolated execution environments called runners (configured via the `runs-on` keyword), which can be GitHub-hosted or self-hosted.

**Step:** [14] Steps are the individual instructions within a job. Each step performs a specific command or action, such as executing a script or running a shell command.

**Action:** [15] An action is a unit of functionality within a step. While developers can define each step manually, they also have the option to reuse third-party actions, which help reduce repetition by encapsulating commonly used procedures. For instance, the widely used third-party action `actions/checkout` enables workflows to access the contents of a repository. GitHub allows developers to create three types of actions: i) **Composite Actions**, which consist of a sequence of shell commands or steps that are executed together to perform a task; ii) **JavaScript Actions**, which use Node.js and JavaScript code to perform more complex logic; and iii) **Docker Actions**, which run tasks inside a predefined Docker container environment. Third-party actions are referenced using the `uses` syntax and can be either private or published to the GitHub Actions Marketplace.

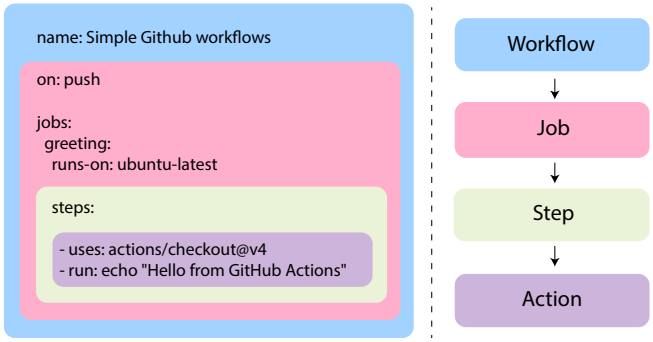



*Fig. 1: Github workflow components [11]*

### B. Github Marketplace

The GitHub Actions Marketplace is the central ecosystem that provides reusable GitHub Actions for developers. Actions are organized into various categories, and each listing includes useful metadata such as the creator's verification status, the associated GitHub repository, and a README file describing the action's functionality.

To publish a third-party action on the GitHub Marketplace, developers must include a metadata file named either `action.yml` or `action.yaml` at the root of the repository. This file must define three key components: *Inputs* - required arguments passed into the action, *Outputs* (optional) - values or artifacts returned from the action, and *Runs* - the execution strategy, specifying whether the action is a composite, JavaScript, or Docker-based.

Finally, developers must create a tag and a release in the GitHub repository, which will automatically publish the action to the GitHub Actions Marketplace. Listing. 1 illustrates an example of the `action.yml` metadata structure. In this example, the action defines `who-to-greet` as an input, `time` as an output, and uses the `node20` JavaScript environment, indicating that the action is implemented using JavaScript.

```yaml
name: 'Hello World'
description: 'Greet someone and record the time'

inputs:
  who-to-greet:
    description: 'who to greet'
    required: true
    default: 'World'

outputs:
  time:
    description: 'The time we greeted you'

runs:
  using: 'node20'
  main: 'index.js'
```

*Listing 1: Metadata for GitHub Action consists of `inputs`, `outputs`, and `runs` [16]*

## III. METHODOLOGY

To capture the longitudinal trends of the GitHub Actions Marketplace, we developed a crawling and analysis pipeline as illustrated in Fig. 2. The first component is a crawler that

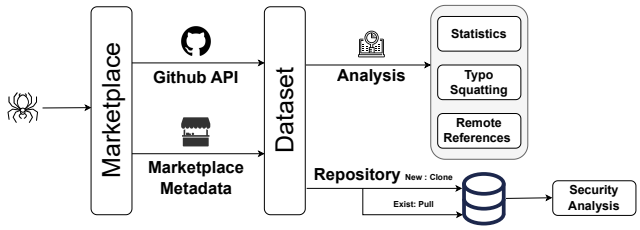



*Fig. 2: System diagram of our approach.*

continuously collects data from the GitHub Actions Marketplace. It scrapes all publicly listed GitHub Actions and clones the corresponding repositories to our local server. This ensures that we archive not only the metadata (e.g., name, creator, stars, and description) but also the actual implementation code for further analysis. The cloning is particularly useful for conducting security tests against the code, as well as to be able to investigate the actions that were later deleted by the platform or by their owner. The second component is an analysis system, which processes the collected data to extract insights from this evolving ecosystem. This includes analyzing marketplace trends (e.g., addition and removal rates), identifying security issues, and detecting potential attack vectors such as dangling remote references and typosquatting [5], [17], [18].

### A. Crawling System

The GitHub Actions Marketplace is dynamic, with new actions continuously published and existing ones potentially removed. Since actions are not available via GitHub APIs, their collection must be done via traditional web crawling. At the same time, crawling the Actions marketplace is not trivial given pagination drift (the actions listed on page $N$ are different at the beginning of the crawl vs. at the end of the crawl) and an upper limit of 500 pages per actions category, set by GitHub.

To address both the pagination drift and the page limit constraint, we sorted the actions in descending order based on their creation timestamps and launched three consecutive crawlers across all available pages. The results from each crawler were then merged, and duplicate entries were removed to construct a comprehensive dataset of unique GitHub Actions. To bypass the category-based limitation, we systematically crawled 24 categories listed in the GitHub Actions Marketplace (see Appendix A).

For each category and page, the crawler extracted hyperlinks to individual GitHub Actions listed in the Marketplace. From each action page, we collected metadata using the GitHub API, including: a repository's identifier, the repository's URL, creation date, last-update date, and number of stars. Pagination drift can cause our crawlers to either miss a new action or errenously conclude that a previously-listed action is no longer listed in the marketplace (e.g. is an action deleted or did it "move" to the page our crawler just finished analyzing?). To this end, we calculated the rate of false positives by double-checking the release dates and availability of actions in both newly detected and removal sets, then recalculated the number of actions across two consecutive crawls. We observed an average false positive rate of $0.71\,\%$ which is acceptable for our purposes.

Our pipeline automatically clones the repository of each discovered action upon first discovery and keeps it up-to-date by requesting updates upon subsequent crawls. We also timestamped each repository with the current crawl date, allowing us to later distinguish between newly added and unchanged actions.

### B. Github Actions Marketplace Analysis

To analyze the ecosystem of the GitHub Actions Marketplace, we began by determining the total number of actions, as well as identifying newly created and removed actions. We then computed several key statistics of the Marketplace, including the percentage of creator-verified actions and the number of repositories that utilize these actions.

For part of our security analysis, we leveraged GWChecker [5] to examine both action metadata and real-world workflow examples in each action repository. Specifically, we evaluated a full-length commit SHA pinning practices, focusing on whether actions were referenced using immutable commit hashes rather than floating tags. SHA pinning allows a repository to survive a supply-chain attack whereas floating tags can be abused by attackers to push arbitrary code to the actions that depend upon the compromised repo. We also investigated whether the actions were published by verified creators and whether they were officially listed on the GitHub Marketplace. Lastly, we analyzed the types of GitHub events that are used to trigger these action repositories, obtaining insights into how and when actions are executed in practice.

Moreover, in addition to direct attack vectors targeting GitHub Actions—such as compromising the action repositories themselves by script injection—we also explored typosquatting attacks targeting legitimate GitHub actions usernames. This is particularly relevant because GitHub usernames directly correspond to action username identifiers in the GitHub Actions Marketplace. For example, attackers could register the "aactions" GitHub account/organization name and offer the `aactions/checkout` action with the goal of hijacking some of the integration meant for `actions/checkout`. Following prior work [18], [19], we generated multiple categories of typos, including character-omission typos, character-permutation typos, character-substitution typos, and character-duplication typos. Other potential attack vectors, such as GitHub repository hijacking via username deletion and reuse, were excluded from our analysis due to GitHub's 90-day waiting period for re-registering deleted usernames [20], which mitigates the immediate risk of such attacks.

Lastly, due to the ability of GitHub Actions to include remote references, we conducted a security analysis focused on these external dependencies. We extracted all URLs and IP addresses from the collected action repositories, excluding

*TABLE I: Retention and Attribute Changes in the Same GitHub Actions Between First and Last Crawls*

| Status | Percentage |
|---|---|
| No Activity | 61.0% |
| Deleted | 1.2% |
| Updated | 37.8% |
| **Among Updated Actions:** | |
| • Gained Stars | 54.6% |
| • Had Updates | 83.9% |
| • Were Forked | 58.8% |
| • Became Verified Creators | 0.1% |
| • Were Transferred to New Owners | 1.5% |

those found within commented-out portions of code and text. For each URL, we performed DNS-based analysis to check whether the utilized domain names could be resolved. Additionally, we investigated whether any of the domains referenced in GitHub actions were parked [21]. In both cases, our goal is to identify actions that could be abused by adversaries by the mere purchasing of expired domain names and parked domains that are for sale.

## IV. Results

In this section, we report our findings on the GitHub Actions Marketplace. Our crawling and data-collection period lasted from December 18, 2024, to April 9, 2025. In total, we crawled 1,912 pages across all categories on our first crawl. The Actions Marketplace was crawled on a weekly basis and each crawl took approximately eleven hours to extract data, with an additional four hours to clone/update the identified actions-related repositories. As of the most recent crawl, we collected 23,757 GitHub Actions from the Marketplace.

### A. GitHub Marketplace Statistics

To analyze the longitudinal differences in GitHub Actions, we compared the set of retained actions between the first and last crawls. Specifically, we captured 22,379 actions on December 18, 2024, and re-examined the same set on April 9, 2025. As shown in Table I, 61.0% of the actions showed no activity during our monitoring period, indicating that developers created and published these actions without further maintenance or involvement, whereas 1.2% of them were deleted. 37.8% of the actions exhibited one or more updates (i.e. were under active development) during our observation period.

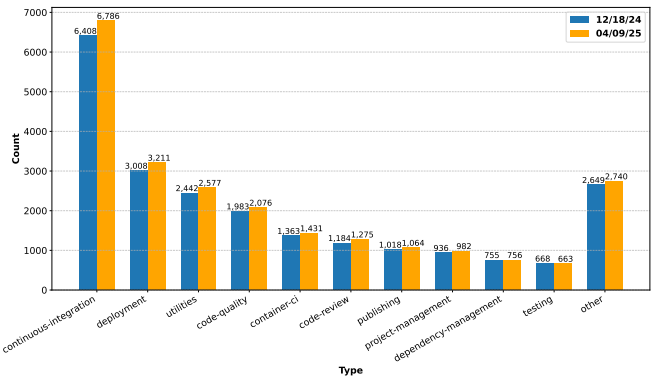



*Fig. 3: Github categories comparison between the first and last crawl*

Among the active actions, we observed consistent updates (e.g., repository commits), along with increases in community engagement metrics, such as stars and forks. In addition, 1.5% of the actions were transferred to different owners during this period.

GitHub Marketplace displays a verified creator badge on actions whose creators have successfully completed GitHub's verification process [22]. This badge is meant to enhance the trustworthiness of actions published by verified creators. However, only 8 actions (0.1%) changed their status from unverified to verified creator.

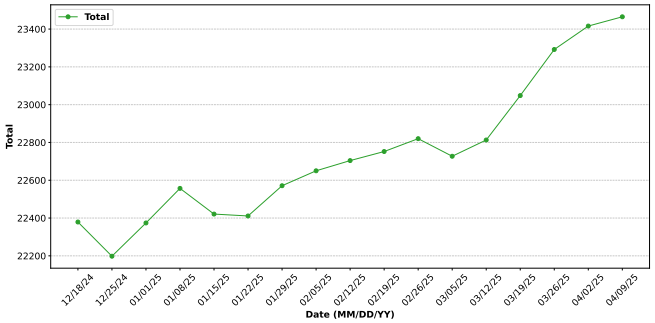
***Action categories:*** The GitHub Actions Marketplace classifies each action into categories to aid developers in identifying actions that are relevant to their use cases. At the beginning of our crawling period on December 18, 2024, there were 24 categories as represent in Appendix Table IV. When developers publish an action to the Marketplace, they are allowed to select up to two categories. Over time, the number of categories expanded to 32. Despite this expansion, we confirmed that our crawler continued to cover all actions, even those listed under the newly added categories.

Fig. 3 shows the distribution of actions across categories, comparing the first and last crawls (December 18, 2024 and April 9, 2025, respectively). The category with the largest number of actions in both crawls is Continuous Integration, which primarily relates to CI activities such as security checking, with 6,408 actions initially and 6,786 at the end (5.9% growth). Similarly, the second most common actions category was Deployment, associated with artifact deployment, showing 3,008 and 3,211 actions, respectively (6.7% growth). Most categories had significantly lower counts in both the first and last crawls.

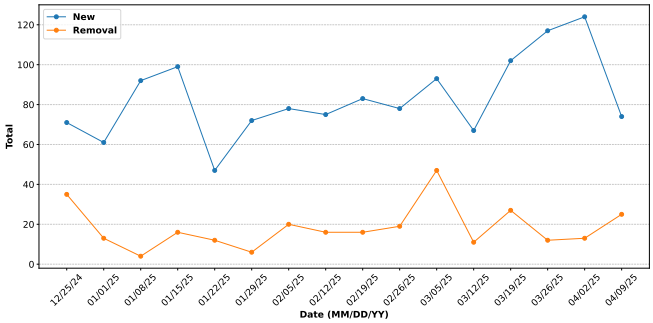
**In summary**, our data indicates steady growth in the GitHub Actions ecosystem throughout the collection period. The most dominant categories remain Continuous Integration and Deployment, which form the backbone of typical CI/CD pipelines, while the remaining categories receive comparatively less attention.

***New and Removed actions:*** Fig. 4 shows the trend of new, removed, and total actions over our data-collection period. The total number of actions has steadily increased since we began collecting data. Initially, there were 22,379 actions, which gradually increased to 23,465 by April 9, 2025.

The number of new actions fluctuated over time. The peak in new actions occurred on April 2, 2025, with 124 new actions added. In contrast, the number of removed actions was




*(a) Total number of GitHub Actions over time.*




*(b) Newly added and removed actions from the Marketplace.*

*Fig. 4: Week-by-week trends in the GitHub Actions Marketplace.*

generally lower than the number of additions, with the most removed actions (47) observed on our March 5, 2025 crawl.

***Creator Verification**:* To further understand the ecosystem of actions, we examined the ratio of verified to non-verified creators. GitHub's security hardening guidelines recommend using actions created by verified creators when integrating third-party GitHub Actions. Each GitHub Action page includes a creator verification status, which we extracted as part of the metadata, as described in the section III-A.

The trend in creator verification remained relatively stable over time. At the beginning of our data collection on December 18, 2024, 4.3 % of creators were verified, while 95.7 % were not. In the most recent crawl on April 9, 2025, the ratio was slightly different but showed no significant change: 4.2 % verified and 95.8 % non-verified. This aligns with findings reported in previous work [5].

The low verified-to-non-verified ratio may be attributed to the verification process, which requires creators to successfully pass GitHub's verification—a process currently reserved for organizations. When attempting to become verified creators, we received the following email response: *"The verified creator badge is currently reserved for organizations that are a GitHub Technology Partner. At this time, this designation is not available to individual developers. Only organizations that have completed the Technology Partner onboarding process and met our verification requirements are eligible to receive the badge."*

**In summary**, our longitudinal analysis of the GitHub Actions Marketplace revealed that more than 60% of actions became inactive over time, while only 8 actions changed their status to verified creator. Moreover, the proportion of verified creators has remained consistently low. Despite the overall growth of the Marketplace, verification has not gained substantial traction, with more than 95% of available actions being offered by non-verified accounts. We expect that this phenomenon is largely due to the verified creator badge being reserved for organizations, and not available to individual developers.

***GitHub Action Usage by Repositories**:* To analyze the number of repositories (dependents) that use each GitHub Action, we collected dependent counts from the dependency graph of each action's GitHub repository. From our analysis, only eight actions have been adopted by more than one million repositories, as presented in Table II.

*TABLE II: GitHub Marketplace Actions and Repository Usage*

| Action | GitHub Repository URL | Repo Usage |
|---|---|---|
| actions/checkout | https://github.com/actions/checkout | 12,931,705 |
| actions/setup-node-js-environment | https://github.com/actions/setup-node | 2,601,383 |
| Azure/azure-static-web-apps-deploy | https://github.com/Azure/static-web-apps-deploy | 1,974,551 |
| actions/upload-a-build-artifact | https://github.com/actions/upload-artifact | 1,606,872 |
| actions/setup-python | https://github.com/actions/setup-python | 1,363,807 |
| actions/cache | https://github.com/actions/cache | 1,187,697 |
| actions/deploy-github-pages-site | https://github.com/actions/deploy-pages | 1,023,483 |
| actions/upload-github-pages-artifact | https://github.com/actions/upload-pages-artifact | 1,001,552 |

The most widely used action is `actions/checkout`, which is commonly used to set the current workspace for a GitHub Actions runner used by almost 13M repositories. The second most popular action is `actions/setup-node`, which configures the Node.js version and environment, with 2.6M dependent repositories. Other frequently used actions include `Azure/azure-static-web-apps-deploy` which deploys artifacts to Azure Static Web Apps; `actions/upload-artifact`, which uploads artifacts from a runner to GitHub; and `actions/cache`, which is commonly used to cache dependencies during workflow execution.

Fig. 5 shows the distribution of GitHub Actions based on the number of repositories that use them. In the leftmost cluster, 20,477 actions are used by a total of 175,865 repositories. In contrast, the rightmost cluster contains only 8 actions, which are collectively used by 23,691,050 repositories.

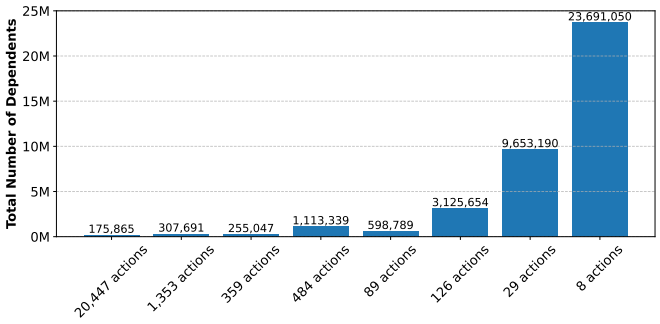



*Fig. 5: Actions grouped by their level of depedence.*

**In summary**, although the GitHub Actions ecosystem continues to grow, a small number of actions consistently dominate repository usage. Specifically, eight actions from just two accounts, `actions` and `Azure`, account for more than half of all usage. This observation aligns with the findings of Zimmermann et al. [23], who show that a small number of accounts are responsible for a disproportionately high number of widely used npm packages. As with all software monocultures, this is a double-edged sword. From the positive side, assuming that these accounts and actions contain no vulnerabilities and cannot be compromised, the majority of action-utilizing GitHub repos will not suffer from actions-related, supply-chain attacks. At the same time, these ultra-popular actions are prime targets for supply-chain attacks given their high ROI in terms of effort that an attacker would need to expend to compromise them vs. the yield of a successfull attack.

### B. Security Analysis

***Security properties:*** We examine the security properties defined in prior work [5] to understand i) how they manifest in GitHub Actions and ii) how these properties vary over time. Our analysis covers both the action metadata files (`action.yml`) and the workflow examples located in the `.github/workflows` directory of each repository. We classify the identified security properties into five categories, as shown in Fig. 6, based on both newly added and removed actions. The figure presents the number of security-related instances in each category, which we describe in more detail below.

1. *Execution Control* refers to insecure trigger configurations, particularly when actions are set to run on events such as `push` or `pull_request`. In such cases, an attacker could fork the repository and submit a malicious pull request, potentially leading to unauthorized code execution or unintended changes in the base branch. Over the course of our analysis, the number of actions exhibiting this configuration fluctuated. The highest observed instance occurred on March 26, 2025, with 136 newly published actions configured in this potentially insecure manner. In contrast, the number of removed actions exhibiting this configuration remained consistently lower throughout the observation period.

After investigating the `push` and `pull_request` workflow trigger configurations of March 26, we found that most of them are used for validating or debugging the action itself. For example, in the case of TypeScript/JavaScript actions, developers often clone an action template from [24] that includes vetting workflows such as `check_dist.yml` for checking the transpiled output in the `dist/` directory, `codeql-analysis.yml` for analyzing the action's codebase using `CodeQL`, and `linter.yml` for static analysis using `super-linter`. We consider these push/pull triggers reasonable, since the root repositories runs them to analyze the action's codebase.

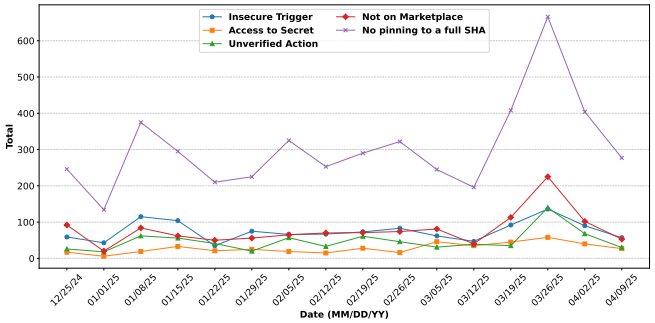
2. *Access to Secrets* captures instances where workflows or action metadata files contain hardcoded sensitive information, such as access tokens for third-party services. Among newly published actions, this issue was notably less frequent compared to other security properties. The number of affected actions remained consistently low across the entire data collection period, both for newly added and removed actions.

3. *Unverified Action Use* refers to whether workflows invoke actions authored by unverified creators. As reusable actions have become more prevalent, developers often include third-party actions within their testing workflows. These referenced actions may originate from unverified sources. Among new actions, we observed fluctuating patterns over time. However, the number of removed actions exhibiting this behavior was consistently lower.

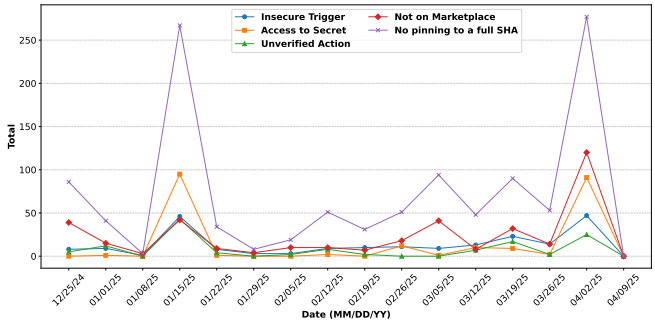
4. *Not on Marketplace* refers to workflows that depend on GitHub Actions not officially listed in the GitHub Marketplace. These off-marketplace actions may lack visibility, community vetting, or trust signals. While not inherently malicious, their use may introduce additional risks, especially when sourced from unverified or obscure accounts. On March 26, 2025, we observed a spike in new actions referencing off-marketplace dependencies, totaling 255 instances. This increase correlates with a rise in newly collected actions. In contrast, the number of removed actions referencing off-marketplace dependencies remained significantly lower due to the overall low volume of deletions during the same period.

5. *Not pinning to a full-length commit SHA* refers to the practice of referencing third-party actions without specifying their full-length commit SHA [25]. Pinning to a specific SHA is a recommended security best practice [26], as it ensures workflows depend on immutable versions of actions, thereby preventing unintended updates—even if the original repository is later modified or compromised.

Among all evaluated security properties, the absence of full-length commit SHA pinning was the most frequently observed issue when reusing third-party actions. Across all time frames, newly published actions consistently included over 304 instances that did not use commit-SHA references. The largest number of such cases occurred on March 26, 2025, with 666 new actions lacking SHA-based versioning. Similar to newly published actions, many removed actions also did not pin third-party references to a full-length commit hashes,




*(a) Security properties observed in newly added actions.*




*(b) Security properties observed in removed actions.*

*Fig. 6: Longitudinal trends of security properties in GitHub Actions: (a) new actions, and (b) removed actions.*

with notable peaks on January 15, 2025, and April 2, 2025, at 267 and 277 instances, respectively.

**In summary**, our longitudinal analysis of security properties in GitHub Actions reveals a strong correlation between the number of security-relevant configurations and the volume of newly added and removed actions at each time frame. Despite the continued growth of the GitHub Actions ecosystem, developers have yet to consistently adopt recommended security practices. Notably, the absence of version pinning to full-length commit SHA hashes remains a prevalent issue in GitHub Actions configurations.

***Typosquatting**:* Typosquatting refers to the practice of registering mistyped variations of legitimate names—to impersonate or redirect users to malicious resources. This type of attack is common in web environments, where attackers register domains similar to popular websites to trick users into visiting them [18], [27], [28].

In the GitHub ecosystem, this risk extends to repositories and GitHub Actions, as usernames are part of the action identifier. For example, if a user with the GitHub username `actionx` creates a repository named `my_action`, the corresponding URL would be: https://github.com/actionx/my_action. When the action is published to the GitHub Actions Marketplace, it receives a Marketplace URL like: https://github.com/marketplace/actions/action_name, where `action_name` is defined in the action metadata file `action.yml`. This Marketplace name must be globally unique.

To reuse a GitHub Action, a developer specifies it in the workflow file using the format: `github_username/repository@tag`, where `github_username` is the GitHub account name, `repository` is the GitHub Action repository, and `tag` refers to the specific version (typically a Git tag).

To analyze the potential for typosquatting in GitHub Actions, we generated typo variations of GitHub account names using established typo-generation techniques, as described in Section III. We focused on usernames collected from the Marketplace and used only legitimate repository names. Git tags were excluded from this analysis.

From our analysis, we generated 1,528,049 typo variants based on 15,527 unique GitHub usernames. Of these, 1,434,247 accounts returned `4xx` errors and were discarded, while 93,802 were valid GitHub account names.

We then mapped the valid typo-based GitHub accounts to legitimate action repositories to examine the existence of typosquatted GitHub Actions. We identified 25 GitHub Actions associated with typo-based accounts that were active at the time of analysis. Notably, three of these targeted some of the most widely used action accounts: `actions`, `aws-actions`, and `trufflesecurity`. The typo variants for `actions` included `actoins` and `actiions`, which hosted outdated clones of popular actions such as `checkout`, `cache`, and `upload-artifact`. The variant `aws-action` appeared to be a misspelling of `aws-actions`. That specific action included a warning telling developers not to integrate this specific action [29] so it could be someone's experiment or a form of defensive registration. Lastly, the typo `truflesecurity` was a fork of the original `trufflesecurity` repository. Most of the typo-squatting repositories were forked years ago from the authoritative actions and therefore lacked updates compared to their original actions repositories. While we could not establish malice through our manual analysis of these typosquatting actions, they still belonged to third-party accounts who could, at a moment's notice, update the actions to abuse the repositories that have accidentally linked to them. We contacted all affected stakeholders. A few responded, confirmed the typosquatting behavior, and subsequently deleted the accounts associated with the squatting `actions` and `actiions`.

**In summary**, typosquatting in the GitHub Actions Marketplace still occurs, although it is not widespread (0.06 % of actions were squatted). Most of the typo actions were either direct clones or forks of the original repositories. It is also worthwhile noting that GitHub metadata can be abused by attackers to obfuscate the true owner of a repo [30]. For example, when a GitHub repo is cloned, the original repo's history appears as the history of the cloned repo. This makes it difficult for someone to establish the true owner of the typosquatting action since it appears to be sharing contributors with the original one.

*TABLE III: GitHub Actions Referencing Expired Domains Eligible for Re-Registration*

| Action | Expired Domain | Usage | Verified |
|---|---|---|---|
| 1 | son****.com | 20,921 | Yes |
| 2 | fak****.com | 1,001 | Yes |
| 3 | sec****.com | 564 | Yes |
| 4 | eeL****.com | 301 | No |
| 5 | my****.com | 201 | No |

### C. Remote References

GitHub Actions allow the inclusion of remote references—such as URLs or IP addresses—across all action types, including composite, Docker, and JavaScript-based actions. This feature introduces an additional attack vector, as referenced URLs may expire and be re-registered by attackers, even when developers correctly pin third-party actions to full-length commit SHAs. Expired or abandoned URLs can also be monetized through services, such as, domain parking. Moreover, unreachable endpoints (via URLs or IP addresses) can lead to unexpected behavior or failures during workflow execution.

In this section, we analyze the security implications of remote references by performing DNS resolution checks, domain parking analysis for URLs, and dead host detection for IP addresses. To prevent abuse of the discovered issues, we anonymize the affected actions and domain names.

***URLs:*** We began our analysis by resolving DNS records for 9,799,201 URL instances extracted from 23,757 GitHub Action repositories. We found that 99.95 % of the instances had valid DNS resolutions, while 0.05 %—spanning 732 repositories—contained unresolved DNS entries. These unresolved URLs referenced a total of 571 unique domains, of which 312 (54.5 %) could be re-registered. For example, the domain son****.com appeared in the `env` section of a workflow within the repository of `Action1`, which has more than 20,000 dependent repositories. Table III presents examples of actions with expired domains that could be re-registered, along with their usage statistics (how many repos use the affected action) and creator-verification status.

Although parking domains pose less immediate harm than re-registered expired domains, they still represent a security concern. In addition to indicating potential abandonment or neglect, parked domains may be monetized through advertising and malicious redirections. We identified parked domains by checking the DNS records of referenced URLs against a known list of domain parking companies [21]. Out of the 29,657 unique URLs collected from DNS analysis, 194 (0.65 %) were detected as parked. An example parked domain is keyvalue.xyz, which currently hosts as a `baseUrl` JavaScript variable in *Redacted* action.

***IP Addresses:*** Using standard regular expressions, we extracted 7,013 hardcoded IP-address-like strings across all collected GitHub Action repositories. Unfortunately, given the similarity between software versions (e.g. 2.3.4.5) and IP addresses (e.g. 8.8.8.8), this set of IP addresses has to be further filtered to establish the true attack surface involving remote hosts addressed via their IP address.

To this end, we randomly sampled 100 IP addresses from the dataset for further manual analysis. Among these, 63 % were valid IP address values. Since ICMP may be blocked by firewalls, we established whether these hosts were reachable via port scanning. We conservatively marked a host as "alive" if it had at least one open port, regardless of whether that port matched ports referenced in GitHub actions. Of those, 25.4 % had at least one open port, while 74.6 % had no open ports. In many cases, the IP addresses without open ports were found in unit testing files (e.g., within `__test__` directories). If we extrapolate the results from our sample to the entire set of actions, more than 3,000 hardcoded IP addresses will be pointing to offline systems.

**In summary**, while the general threat of remote references and residual trust is known, our study is the first one that quantifies its presence in the context of GitHub actions. References (be it expired domain names or available IP addresses in public clouds) that can be controlled by attackers can be abused to perform supply-chain attacks to the affected action and any repos that rely upon that action. Importantly, these vulnerabilities are not resolved even if SHA pinning is used since domain names can expire and fall under the control of an attacker, without the source code of the including action ever changing.

### D. Removed actions and malware

Lastly, to understand whether attackers actively upload malicious actions to the GitHub Marketplace (as opposed to compromising existing ones), we analyzed actions that were removed from the Marketplace during our observation period. Our hypothesis was that if malicious actions were being uploaded, some of them would eventually be detected and removed by GitHub, making removed actions a potential signal for malicious activity.

We randomly sampled 146 actions that were removed from the Marketplace during our study period and manually analyzed their code and metadata for signs of malicious behavior. However, our analysis did not reveal any clear evidence of malice. Most removals appeared to be due to benign reasons such as developers deprecating their actions, consolidating duplicate functionality, or removing experimental/prototype actions. This negative result suggests that either attackers are not frequently attempting to upload overtly malicious actions to the GitHub Marketplace, or if they are, such actions remain undetected by GitHub's security measures. Given GitHub's security infrastructure and the visibility of the Marketplace, we opine that the former explanation is more likely – attackers appear to prefer compromising existing trusted actions rather than uploading new malicious ones.

## V. Related Work

To the best of our knowledge, we are the first ones to longitudinally analyze the GitHub Actions marketplace and

quantify vulnerabilities that could be abused in the context of supply-chain attacks. In this section, we discuss security-related research areas that are closely related with our work.

### A. Github actions analysis

With the growing popularity of GitHub Workflows as a core component of CI/CD pipelines, several studies have explored their security implications. Duncan et al. [31] conducted a study analyzing GitHub Workflows and categorized the key components, such as jobs, steps, and third-party actions. They also examined the outdatedness of GitHub Actions in workflows [32]. Moreover, Deliche et al. [33] conducted a preliminary study on the dependency relationships within GitHub Actions, including the evolution of action types and the characteristics of Docker-based actions. However, these studies do not address other important security-related issues, such as, the presence of typosquatting in the actions marketplace.

Benedetti et al. [34] proposed a pattern-matching approach to detect seven security smells within GitHub Workflows. Koishybayev et al. [5] introduced ten security properties to improve detection granularity, analyzing workflows across various CI/CD platforms. Specifically, they examined whether GitHub workflows and actions violated the proposed security conditions, which include: configuration of workflow permissions (e.g., whether default permissions are changed to read-only), workflow triggers (e.g., whether workflows can be triggered by `pull_request`), workflow secrets (e.g., storing secrets in plaintext), unverified action usage (e.g., using actions created by unverified creators), and references to third-party actions (e.g., whether a commit SHA is specified when referencing third-party actions). We adopt these security properties in our work to longitudinally analyze GitHub Actions available in the Marketplace.

GitHub later introduced `CodeQL` [35], a taint-tracking-based static analysis framework designed to identify vulnerabilities in Github workflows. Additionally, Muralee et al. [6] proposed a taint-tracking framework to trace the usage of third-party actions, enabling more comprehensive security coverage across CI/CD pipelines.

Beyond GitHub-specific analysis, Gu et al. [36] proposed a system to analyze the risks of token leakage across popular CI platforms. They demonstrated how compromised tokens can lead to privilege escalation, even under the least-privilege model. In subsequent work, Gu et al. [37] examined caching practices in CI platforms and their potential security risks. Li et al. [38] studied the attack surface of third-party plugins used in CI pipelines and introduced new attack vectors, such as plugin redirection hijacking.

Most of the aforementioned studies rely on GHArchive [39] to download repositories from GitHub and extract third-party actions using the `uses:` syntax from workflow files. In contrast, our work focuses on a longitudinal analysis of GitHub Actions directly collected from the GitHub Actions Marketplace. This approach enables us to observe trends over time and analyze additional security dimensions not previously explored.

### B. Security Threats from Remote References and Domains

Remote references refer to embedded URLs or IP addresses within development ecosystems, such as websites, email links, source code repositories, or configuration files. [17], [40]–[43]. These references can be abused when they expire or become inactive, allowing attackers to re-register and control them without the user's awareness—undermining the implicit trust placed in these external endpoints.

Typosquatting is an attack technique that exploits human errors in typing, where malicious actors register slight variations of legitimate names [44], [45]. Agten et al. [18] conducted a longitudinal study of typosquatting abuse across the 500 most popular websites over a seven-month period, demonstrating its widespread impact. Blog posts [46]–[48] have previously described case studies involving typosquatting in the world of GitHub actions but, to our knowledge, no systematic analysis has been conducted before.

Another related issue is domain parking, where unused or expired domains are monetized through advertisements or resale using domain parking services [21], [49], [50]. While this practice may appear benign, parked domains still represent a security concern since they opportunistically answer all requests sent to them and have been historically associated with malvertising.

Despite the importance of these issues, prior to our work, there had been no prior systematic study of remote reference abuse (including typosquatting and domain parking) in the context of GitHub Actions.

## VI. Conclusion

In this paper, we presented a longitudinal analysis of the GitHub Actions marketplace, a large ecosystem of reusable CI/CD workflows that developers can integrate in their GitHub repositories. Over a four-month longitudinal study, we collected more than 23,757 actions, observing the steady increase of actions that developers make available for others to use.

Despite this growth, security best practices, such as full-length commit SHA pinning and creator verification, remain underutilized, even though they are recommended in GitHub's security hardening guidelines. Next to quantifying the lack of best practices, we also investigated novel security risks including typosquatting actions and the threats of remote references in the context of GitHub actions. Namely, we identified 25 typosquatting action repositories that cloned or forked popular actions to capitalize on typos that developers make when attempting to use an action from the marketplace. Furthermore, remote references – such as URLs and IP addresses – are widely embedded in GitHub Actions. We identified almost 10 million URLs embedded in GitHub actions repositories, with 732 domains that could not be resolved, the majority of which could be immediately re-registered by attackers to serve malicious content and perform supply-chain attacks. These types of dangling remote references are particularly

problematic in the context of GitHub actions since attackers can exploit them without the need to commit new code on the affected repo, thereby bypassing the protections offered by SHA pinning.

Our results indicate that security practices are not consistently followed in the GitHub Actions ecosystem, increasing the potential for exploitation. Moreover, additional attack vectors and areas – such as third-party dependency validation and behavioral analysis of reused actions – warrant further research. We hope this work provides insights into the evolving landscape of GitHub Actions and raises awareness about its associated security challenges. GitHub actions *are* first-class citizens in the open-source software ecosystem and warrant the level of scrutiny applied to regular code repositories.

## Acknowledgments

We thank our shepherd and the reviewers for their valuable feedback. This work was supported by the Office of Naval Research (ONR) under grant N00014-24-1-2193, as well as by the National Science Foundation (NSF) under grants CNS-1941617 and CNS-2211575.

## References

[1] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *IEEE software*, vol. 33, no. 3, pp. 94–100, 2016.

[2] Jenkins. (2025, Jan.) Jenkins. [Online]. Available: https://www.jenkins.io/.

[3] GitLab. (2025, Jan.) Gitlab ci. [Online]. Available: https://docs.gitlab.com/ci/.

[4] GitHub. (2025, Jan.) Github actions. [Online]. Available: https://github.com/features/actions.

[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)*, 2022, pp. 2747–2763.

[6] 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. 6983–7000.

[7] GitHub. (2025, Jan.) Github marketplace. [Online]. Available: https://github.com/marketplace?type=actions.

[8] V. Sharma. (2025, Mar.) Harden-Runner detection: Tj-actions/changed-files action is compromised. [Online]. Available: https://www.stepsecurity.io/blog/harden-runner-detection-tj-actions-changed-files-action-is-compromised.

[9] R. McCarthy. (2025, Mar.) GitHub Action supply chain attack: Reviewdog/action-setup. [Online]. Available: https://www.wiz.io/blog/new-github-action-supply-chain-attack-reviewdog-action-setup.

[10] O. Gil, A. Hahami, A. Greenholts, and Y. Avital. (2025, Mar.) Github actions supply chain attack: A targeted attack on coinbase expanded to the widespread tj-actions/changed-files incident: Threat assessment (updated 3/21). [Online]. Available: https://unit42.paloaltonetworks.com/github-actions-supply-chain-attack/.

[11] Microsoft. (2025, July.) Identify the components of github actions. [Online]. Available: https://learn.microsoft.com/en-us/training/modules/github-actions-automate-tasks/2b-identify-components-workflow.

[12] GitHub. (2025, July.) About workflows. [Online]. Available: https://docs.github.com/en/actions/concepts/workflows-and-actions/about-workflows.

[13] GitHub . (2025, July.) Using jobs in a workflow. [Online]. Available: https://docs.github.com/en/actions/how-tos/writing-workflows/choosing-what-your-workflow-does/using-jobs-in-a-workflow.

[14] GitHub. (2025, July.) Workflow syntax for github actions. [Online]. Available: https://docs.github.com/en/actions/reference/workflow-syntax-for-github-actions#jobsjob_idsteps.

[15] GitHub . (2025, July.) Understanding github actions. [Online]. Available: https://docs.github.com/en/actions/get-started/understanding-github-actions#actions.

[16] GitHub. (2025, July.) Creating an action metadata file. [Online]. Available: https://docs.github.com/en/actions/tutorials/creating-a-javascript-action#creating-an-action-metadata-file.

[17] J. So, N. Miramirkhani, M. Ferdman, and N. Nikiforakis, "Domains do change their spots: Quantifying potential abuse of residual trust," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2130–2144.

[18] P. Agten, W. Joosen, F. Piessens, and N. Nikiforakis, "Seven months' worth of mistakes: A longitudinal study of typosquatting abuse," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society, 2015.

[19] Y.-M. Wang, D. Beck, J. Wang, C. Verbowski, and B. Daniels, "Strider typo-patrol: Discovery and analysis of systematic typo-squatting." *SRUTI*, vol. 6, no. 31-36, pp. 2–2, 2006.

[20] Github. (2025, Feb.) Deleting your personal account. [Online]. Available: https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-your-personal-account/deleting-your-personal-account.

[21] J. Zirngibl, S. Deusch, P. Sattler, J. Aulbach, G. Carle, and M. Jonker, "Domain parking: Largely present, rarely considered!" in *TMA*, 2022.

[22] GitHub. (2025, July.) About badges in github marketplace. [Online]. Available: https://docs.github.com/en/actions/how-tos/sharing-automations/creating-actions/publishing-actions-in-github-marketplace#about-badges-in-github-marketplace.

[23] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel, "Small world with high risks: A study of security threats in the npm ecosystem," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 995–1010.

[24] Actions. (2025, July.) Typescript-action. [Online]. Available: https://github.com/actions/typescript-action.

[25] Github. (2025, Feb.) Security hardening for github actions. [Online]. Available: https://docs.github.com/en/actions/security-for-github-actions/security-guides/security-hardening-for-github-actions.

[26] GitHub. (2025, July.) Using third-party actions. [Online]. Available: https://docs.github.com/en/actions/how-tos/security-for-github-actions/security-guides/security-hardening-for-github-actions#using-third-party-actions.

[27] N. Nikiforakis, S. Van Acker, W. Meert, L. Desmet, F. Piessens, and W. Joosen, "Bitsquatting: Exploiting bit-flips for fun, or profit?" in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 989–998.

[28] N. Nikiforakis, M. Balduzzi, L. Desmet, F. Piessens, and W. Joosen, "Soundsquatting: Uncovering the use of homophones in domain squatting," in *International Conference on Information Security*. Springer, 2014, pp. 291–308.

[29] AWS. (2025, Jan.) Aws for github actions - typosquat. [Online]. Available: https://github.com/aws-action.

[30] 0xdead8ead. (2025, Jan.) 0xdead8ead/gitfraud. [Online]. Available: https://github.com/0xdead8ead/gitfraud.

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

[32] A. Decan, T. Mens, and H. O. Delicheh, "On the outdatedness of workflows in the github actions ecosystem," *Journal of Systems and Software*, vol. 206, p. 111827, 2023.

[33] H. O. Delicheh, A. Decan, and T. Mens, "A preliminary study of github actions dependencies." in *SATToSE*, 2023, pp. 66–77.

[34] G. Benedetti, L. Verderame, and A. Merlo, "Automatic security assessment of github actions workflows," in *Proceedings of the ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*, 2022, pp. 37–45.

[35] GitHub. (2025, Feb.) Actions queries for codeql analysis. [Online]. Available: https://docs.github.com/en/code-security/code-scanning/managing-your-code-scanning-configuration/actions-built-in-queries.

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

[37] Y. Gu, L. Ying, H. Chai, Y. Pu, H. Duan, and X. Gao, "More haste, less speed: Cache related security threats in continuous integration services," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 1179–1197.
[38] X. Li, Y. Gu, C. Qiao, Z. Zhang, D. Liu, L. Ying, H. Duan, and X. Gao, "Toward understanding the security of plugins in continuous integration services," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 482–496.
[39] GH Archive. (2025, Jan.) Gh archive. [Online]. Available: https://www.gharchive.org/.
[40] C. Lever, R. Walls, Y. Nadji, D. Dagon, P. McDaniel, and M. Antonakakis, "Domain-z: 28 registrations later measuring the exploitation of residual trust in domains," in *IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 691–706.
[41] T. Moore and R. Clayton, "The ghosts of banking past: Empirical analysis of closed bank websites," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 33–48.
[42] D. Gruss, M. Schwarz, M. Wübbeling, S. Guggi, T. Malderle, S. More, and M. Lipp, "Use-after-freemail: Generalizing the use-after-free problem and applying it to email services," in *Proceedings of the Asia Conference on Computer and Communications Security*, 2018, pp. 297–311.
[43] E. Pauley, R. Sheatsley, B. Hoak, Q. Burke, Y. Beugin, and P. McDaniel, "Measuring and mitigating the risk of ip reuse on public clouds," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 558–575.
[44] S. E. Coull, A. M. White, T.-F. Yen, F. Monrose, and M. K. Reiter, "Understanding domain registration abuses," *Computers & security*, vol. 31, no. 7, pp. 806–815, 2012.
[45] J. Szurdi, B. Kocso, G. Cseh, J. Spring, M. Felegyhazi, and C. Kanich, "The long {"Taile"} of typosquatting domain names," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 191–206.
[46] Ofir Yakobi. (2025, July.) Watch the typo: Our poc exploit for typosquatting in github actions. [Online]. Available: https://orca.security/resources/blog/typosquatting-in-github-actions/.
[47] Ravie Lakshmanan. (2025, July.) Github actions vulnerable to typosquatting, exposing developers to hidden malicious code. [Online]. Available: https://thehackernews.com/2024/09/github-actions-vulnerable-to.html.
[48] Lucian Constantin. (2025, July.) Github actions typosquatting: A high-impact supply chain attack-in-waiting. [Online]. Available: https://www.csoonline.com/article/3506897/github-actions-typosquatting-a-high-impact-supply-chain-attack-in-waiting.html.
[49] S. Alrwais, K. Yuan, E. Alowaisheq, Z. Li, and X. Wang, "Understanding the dark side of domain parking," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 207–222.
[50] T. Vissers, W. Joosen, and N. Nikiforakis, "Parking sensors: Analyzing and detecting parked domains," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS 2015)*. Internet Society, 2015, pp. 53–53.

# APPENDIX A
# ADDITIONAL RESULTS

*TABLE IV: Distribution of GitHub Actions by Category*

| Category | Page |
|---|---|
| API management | 26 |
| Backup Utilities | 10 |
| Chat | 24 |
| Code quality | 105 |
| Code review | 97 |
| Container CI | 74 |
| Continuous integration | 450 |
| Dependency management | 60 |
| Deployment | 272 |
| Deployment Protection Rules | 2 |
| Game CI | 12 |
| GitHub Sponsors | 1 |
| IDEs | 3 |
| Learning | 12 |
| Localization | 7 |
| Mobile | 8 |
| Mobile CI | 15 |
| Monitoring | 26 |
| Project management | 70 |
| Publishing | 138 |
| Security | 72 |
| Support | 17 |
| Testing | 98 |
| Utilities | 401 |