



Insecure GitHub Action Workflows

Finding and exploiting them for fun and profit

Simon Gerst | 26. October 2023



GitHub Actions



What?

- CI/CD
- Automate workflows
- Run on GitHub servers
- Free for public repos

GitHub Actions



What?

- CI/CD
- Automate workflows
- Run on GitHub servers
- Free for public repos

Why?

- Easy to use
- Easy to integrate
- Easy to extend
- Free for public repos

GitHub Actions



What?

- CI/CD
- Automate workflows
- Run on GitHub servers
- Free for public repos

Why?

- Easy to use
- Easy to integrate
- Easy to extend
- Free for public repos

Why attack?

- Can modify source code → supply chain attacks ≫
- Give access to secrets (e.g. API keys) \rightarrow escalate privileges



- YAML files (\(\mathbb{H}^a\))
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

ahttps://ruudvanasseldonk.com/2023/01/
11/the-yaml-document-from-hell



- YAML files (\(\mathbb{H}^a\))
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
Example workflow

name: hello-world
on: [push]
jobs:
   hello_world_job:
   runs-on: ubuntu-latest
   steps:
        - name: Hello World
        run: echo "Hello_World_u${{_ugithub.actor_u}}"
name: Name of the workflow
```

ahttps://ruudvanasseldonk.com/2023/01/
11/the-yaml-document-from-hell



- YAML files (##a)
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
Example workflow
name: hello-world
   [push]
iobs:
  hello_world_job:
  runs-on: ubuntu-latest
    steps:
      - name: Hello World
        run: echo "Hello World $ { { ugithub.actor } } "
```

Trigger for the workflow

^ahttps://ruudvanasseldonk.com/2023/01/ 11/the-yaml-document-from-hell



- YAML files (\(\mathbb{H}^a\))
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
name: hello-world
on: [push]
jobs:
hello_world_job:
runs-on: ubuntu-latest
steps:
    - name: Hello World
    run: echo "Hello_World_u${{_ugithub.actor_u}}"
```

ahttps://ruudvanasseldonk.com/2023/01/
11/the-yaml-document-from-hell



- YAML files (\(\mathbb{H}^a\))
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
Example workflow

name: hello-world
on: [push]
jobs:
    hello_world_job:
    runs-on: ubuntu-latest
    steps:
    - name: Hello World
        run: echo "Hello_World_u${{_ugithub.actor_u}}"

hello_world_job Name of the job
```

ahttps://ruudvanasseldonk.com/2023/01/ 11/the-yaml-document-from-hell



- YAML files (\(\mathbb{H}^a\))
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
Example workflow

name: hello-world
on: [push]
jobs:
   hello_world_job:
   runs-on: ubuntu-latest
   steps:
        - name: Hello World
        run: echo "Hello_World_u${{\undergotengthingstyle="text-align: center;"}"

runs-on) OS to run the job on
```

ahttps://ruudvanasseldonk.com/2023/01/ 11/the-yaml-document-from-hell



- YAML files (\(\mathbb{H}^a\))
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
name: hello-world
on: [push]
jobs:
   hello_world_job:
   runs-on: ubuntu-latest
    steps:
    - name: Hello World
        run: echo "Hello_World_u${{_ugithub.actor_u}}"
steps
Steps Steps to run
```

ahttps://ruudvanasseldonk.com/2023/01/ 11/the-yaml-document-from-hell



- YAML files (\bigcolor{black}{\bigcolor{black}{a}})
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
Example workflow
name: hello-world
on: [push]
iobs:
  hello_world_job:
  runs-on: ubuntu-latest
    steps:
      - name: Hello World
        run: echo "Hello World ${{ github.actor }}"
name: Hello World Name of the step
```

^ahttps://ruudvanasseldonk.com/2023/01/ 11/the-yaml-document-from-hell



- YAML files (\(\mathbb{H}^a\))
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
Example workflow

name: hello-world
on: [push]
jobs:
   hello_world_job:
   runs-on: ubuntu-latest
   steps:
        - name: Hello World
        run: echo "Hello_World_${{ugithub.actor_u}}"

run: ... A shell command step
```

ahttps://ruudvanasseldonk.com/2023/01/ 11/the-yaml-document-from-hell



- YAML files (##a)
- Must be stored in .github/workflows
- Can be triggered by events
- Can be triggered manually
- Can be triggered by other workflows

```
Example workflow
name: hello-world
on: [push]
iobs:
  hello_world_job:
  runs-on: ubuntu-latest
    steps:
      - name: Hello World
        run: echo "Hello World ${{ github.actor }}"
        A shell command step
                                   Command to run
echo "Hello_World_${{_github.actor_}}}"
```

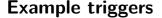
^ahttps://ruudvanasseldonk.com/2023/01/ 11/the-yaml-document-from-hell

Example triggers



Push to a branch: on: push

Pull request: on: pull_request





- Push to a branch: on: push
- Pull request: on: pull_request

```
name: Build and Test
on:
  [push, pull_request]:
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Install dependencies
        run: npm install
      - name: Build project
        run: npm run build
      - name: Run tests
        run: npm test
```





Push to a branch: on: push

Pull request: on: pull_request

Issue comment: on: issue_comment

Example triggers



- Push to a branch: on: push
- Pull request: on: pull_request
- Issue comment: on: issue_comment

```
name: Comment Notification
on:
  issue comment:
    types: [created]
iobs:
  notify:
    runs-on: ubuntu-latest
    steps:
      - name: Send notification
        uses: peter-evans/slack-action@v3
        with:
          slack_secret: ${{ secrets.
              SLACK_WEBHOOK_URL }}
          text: "New_comment_by_${{_github.event.
              comment.user.login_\}:\\${{\ugithub.
              event.comment.bodyu}}"
```





Push to a branch: on: push

Pull request: on: pull_request

Issue comment: on: issue_comment

Cron: on: schedule

Example triggers



Push to a branch: on: push

Pull request: on: pull_request

Issue comment: on: issue_comment

Cron: on: schedule

```
name: Stale Issues
on:
  schedule:
  - cron: '20_16_*_*_*' # Run every day at 16:20 UTC
iobs:
  stale:
    runs-on: ubuntu-latest
    steps:
      - name: Close Stale Issues
        uses: actions/stale@v3.0.5
        with:
          repo-token: ${{ secrets.GITHUB_TOKEN }}
          stale-issue-message: 'This issue is stale i
               because, it, has, been, open, 30, days, with,
              nowactivity.'
          days-before-stale: 30
```





- Push to a branch: on: push
- Pull request: on: pull_request
- Issue comment: on: issue_comment
- Cron: on: schedule
- Manual (Workflow dispatch):

on: workflow_dispatch





```
Push to a branch: on: push
```

Pull request: on: pull_request

Issue comment: on: issue_comment

Cron: on: schedule

Manual (Workflow dispatch):

on: workflow_dispatch

```
on:
  workflow_dispatch:
    inputs:
      V8 VERSION:
        description: "V8_VERSION"
        required: false
        type: string
        default: ""
      V8 BUILD MODE:
        description: "V8_BUILD_MODE"
        default: "release-fuzz"
        required: true
        type: choice
        options:
          - release-fuzz
          - debug-no-fuzz
```



■ Every workflow gets an API token: GITHUB_TOKEN



- Every workflow gets an API token: GITHUB_TOKEN
- Base permissions for the token: can be set to (permissive) or (restricted)





- Every workflow gets an API token: GITHUB_TOKEN
- Base permissions for the token: can be set to (permissive) or (restricted)
- (permissive): practically read/write access to everything
- (restricted): only read access or no access at all



- Every workflow gets an API token: GITHUB_ТОКЕN
- Base permissions for the token: can be set to (permissive) or (restricted)
- (permissive): practically read/write access to everything
- (restricted): only read access or no access at all
- If pull request is from a fork: reduce all permissions to a maximum of read access
- No secrets are available



- Every workflow gets an API token: GITHUB_TOKEN
- Base permissions for the token: can be set to (permissive) or (restricted)
- (permissive): practically read/write access to everything
- (restricted): only read access or no access at all
- If pull request is from a fork: reduce all permissions to a maximum of read access
- No secrets are available
- E.g. issue_comment, issues, push, workflow_run give base permissions and secret access.



- Every workflow gets an API token: GITHUB_TOKEN
- Base permissions for the token: can be set to (permissive) or (restricted)
- (permissive): practically read/write access to everything
- (restricted): only read access or no access at all
- If pull request is from a fork: reduce all permissions to a maximum of read access
- No secrets are available
- E.g. issue_comment, issues, push, workflow_run give base permissions and secret access.
- pull request target gives base permissions and secret access even for forks.
- Only pull request is restricted for forks

User-controlled input



User-controlled input — the root of all evil

Templating in workflows



- **\$**{{ ... }}
- Inserted as is into workflow file before execution
- Predefined variables: github, env, secrets, ...
- Predefined functions: fromJson, startsWith, endsWith, contains, ...



```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "Theucommentuisu${{ugithub.event.comment.bodyu}}"
```



```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_${{\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.body_\undergotengthetaggithub.event.comment.bo
```

Comment body is This is a test



```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_This_is_ua_test"
```

Comment body is This is a test



```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_This_is_a_test"
```

- Comment body is This is a test
- Experienced developers CTF players will notice something:



```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_This_is_a_test"
```

- Comment body is This is a test
- Experienced developers CTF players will notice something:
- We are mixing code and data



Example workflow — unsafe templating

```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_${{\undergotenture} github.event.comment.body_\undergoten}}"
```

- Comment body is This is a test
- Experienced developers CTF players will notice something:
- We are mixing code and data
- Comment body is \$(touch /tmp/pwned)



Example workflow — unsafe templating

```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_(touch_/tmp/pwned)"
```

- Comment body is This is a test
- Experienced developers CTF players will notice something:
- We are mixing code and data
- Comment body is \$(touch /tmp/pwned)



Example workflow — unsafe templating

```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_$(touch_l/tmp/pwned)"
```

- Comment body is This is a test
- Experienced developers CTF players will notice something:
- We are mixing code and data
- Comment body is \$(touch /tmp/pwned)
- Oops, we just let someone execute arbitrary code



- Don't mix code and data
- Instead: Store data in environment variables



- Don't mix code and data
- Instead: Store data in environment variables

```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_$COMMENT"
        env:
        COMMENT: ${{ github.event.comment.body }}}
```



- Don't mix code and data
- Instead: Store data in environment variables

```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_is_$COMMENT"
        env:
        COMMENT: ${{ github.event.comment.body }}}
```

■ The shell will *not* interpret the environment variable as code



- Don't mix code and data
- Instead: Store data in environment variables

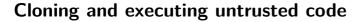
```
name: Example Workflow
on:
    issue_comment:
        types: [created]
jobs:
    example_job:
    runs-on: ubuntu-latest
    steps:
    - name : Print comment
        run: echo "The_comment_uis_u$COMMENT"
        env:
        COMMENT: ${{ github.event.comment.body }}}
```

- The shell will *not* interpret the environment variable as code
- → No code injection



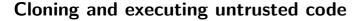


on: pull_request_target (and some other triggers!) gives access to secrets and a token with write access





- on: pull_request_target (and some other triggers!) gives access to secrets and a token with write access
- Attacker opens PR against repository
- Code is cloned and executed





- on: pull_request_target (and some other triggers!) gives access to secrets and a token with write
 access
- Attacker opens PR against repository
- Code is cloned and executed
- → Oops, we just let someone execute arbitrary code



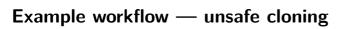


- on: pull_request_target (and some other triggers!) gives access to secrets and a token with write access
- Attacker opens PR against repository
- Code is cloned and executed
- → Oops, we just let someone execute arbitrary code
- on: pull_request is safe





- on: pull_request_target (and some other triggers!) gives access to secrets and a token with write access
- Attacker opens PR against repository
- Code is cloned and executed
- → Oops, we just let someone execute arbitrary code
- on: pull_request is safe
- → Doesn't give access to secrets and write access





```
on:
  pull_request_target
jobs:
 build:
    name: Build and test
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
      with:
        ref: (${{ github.event.pull_request.head.sha }})
    - uses: actions/setup-node@v1
    - run:
        nom install
        npm build
    - run: gh pr comment ${{ github.event.pull_request.number }} -b "Build_successful"
```

- \${{ github.event.pull_request.head.sha }}) is user-controlled and cloned
- npm install and npm build can execute user-controlled code!



Don't clone and execute untrusted code



- Don't clone and execute untrusted code
- Just as with binaries: code should be executable XOR writable
- → clone and execute untrusted code in *unprivileged* environment XOR clone and don't execute untrusted code in *privileged* environment



- Don't clone and execute untrusted code
- Just as with binaries: code should be executable XOR writable
- → clone and execute untrusted code in *unprivileged* environment XOR clone and don't execute untrusted code in *privileged* environment
- Sometimes this is not possible (e.g. execute tests and post results as comment)



- Don't clone and execute untrusted code
- Just as with binaries: code should be executable XOR writable
- → clone and execute untrusted code in *unprivileged* environment XOR clone and don't execute untrusted code in *privileged* environment
- Sometimes this is not possible (e.g. execute tests and post results as comment)
- → *Split* the workflow into two parts:
- Unprivileged workflow: clone and execute untrusted code
- Save results as artifacts



- Don't clone and execute untrusted code
- Just as with binaries: code should be executable XOR writable
- → clone and execute untrusted code in *unprivileged* environment XOR clone and don't execute untrusted code in *privileged* environment
- Sometimes this is not possible (e.g. execute tests and post results as comment)
- → *Split* the workflow into two parts:
- Unprivileged workflow: clone and execute untrusted code
- Save results as artifacts
- Privileged workflow: fetch artifacts and work with them



How to fix this? — Workflow splitting

```
on:
   pull_request_target
jobs:
   build:
      name: Build and test
      runs-on: ubuntu-latest
      steps:
      - uses: actions/checkout@v2
      with:
           ref: ${{ github.event.pull_request.head.sha }}
      - uses: actions/setup-node@v1
      - run: |
            npm install
            npm build
```





pull_request unprivileged, executes untrusted code

```
on:
  pull_request
iobs:
  build:
    name: Build and test
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
      with:
        ref: ${{ github.event.pull_request.
            head.sha }}
    - uses: actions/setup-node@v1
    - run:
        npm install
        npm build
    # upload artifact if needed
```

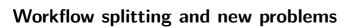




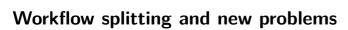
pull_request unprivileged, executes untrusted code

```
on:
  pull_request
iobs:
  build:
    name: Build and test
    runs - on: ubuntu - latest
    steps:
    - uses: actions/checkout@v2
      with:
        ref: ${{ github.event.pull_request.
            head.sha }}
    - uses: actions/setup-node@v1
    - run:
        npm install
        npm build
    # upload artifact if needed
```

workflow_run: privileged, doesn't execute untrusted code on: workflow_run: workflows: - name-of-previous-workflow types: - completed status: - Success iobs: comment: name: Build and test runs-on: ubuntu-latest steps: - run: gh pr comment \${{ github.event .workflow_run.pull_requests[0]. number }} -b "Buildusuccessful"



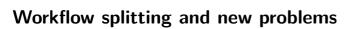






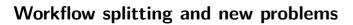
```
on:
   pull_request
jobs:
build:
   # ...
   # upload PR number as artifact
   - run: echo ${{ github.event.}
        pull_request.number }} > pr_number
   - uses: actions/upload-artifact@v2
   with:
        name: pr_number
        path: pr_number
```

```
on:
    workflow_run:
    # ...
jobs:
    comment:
    # ...
    - uses: actions/download-artifact@v2
    with:
        name: pr_number
    - run: echo "PR_NUMBER=$(cat_\( \)
        pr_number)" >> $GITHUB_ENV
    - run: gh pr comment $PR_NUMBER -b "
        Build_successful"
```





```
on:
   pull_request
jobs:
   build:
    # ...
    # upload PR number as artifact
   - run: echo ${{ github.event.}
        pull_request.number }} > pr_number
   - uses: actions/upload-artifact@v2
    with:
        name: pr_number
        path: pr_number
```

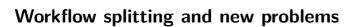




```
on:
    pull_request
jobs:
    build:
    # ...
    # upload PR number as artifact
    - run: echo ${{ github.event.}
        pull_request.number }} > pr_number
    - uses: actions/upload-artifact@v2
    with:
        name: pr_number
        path: pr_number
```

We completely control the value of pr_number!

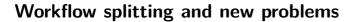
```
workflow_run:
    # ...
jobs:
comment:
    # ...
    - uses: actions/download-artifact@v2
    with:
        name: pr_number
    - run:
        echo "PR_NUMBER=$(cat_pr_number)"
        >> $GITHUB_ENV
        - run: gh pr comment $PR_NUMBER -b "
        Build_usuccessful"
```





```
on:
                                                 on:
                                                   workflow run:
  pull request
jobs:
                                                     # . . .
  build:
                                                 jobs:
                                                   comment:
    # upload PR number as artifact
    - run: echo ${{ github.event.
                                                       - uses: actions/download-artifact@v2
        pull_request.number }} > pr_number
                                                         with:
    - uses: actions/upload-artifact@v2
                                                           name: pr_number
      with:
                                                       - run:
                                                           echo "PR_NUMBER=$(catupr_number)"
        name: pr_number
                                                           >> ($GITHUB ENV)
        path: pr number
                                                       - run: gh pr comment $PR_NUMBER -b "
                                                           Buildusuccessful"
```

- We completely control the value of pr_number!
- \$GITHUB_ENV is a special file used for setting environment variables





```
on:
                                                 on:
  pull_request
                                                   workflow run:
jobs:
                                                     # . . .
  build:
                                                 jobs:
                                                   comment:
    # upload PR number as artifact
    - run: echo ${{ github.event.
                                                       - uses: actions/download-artifact@v2
        pull_request.number }} > pr_number
                                                         with:
    - uses: actions/upload-artifact@v2
                                                           name: pr_number
      with:
                                                       - run:
                                                           echo "PR_NUMBER=$(catupr_number)"
        name: pr number
                                                           >> ($GITHUB ENV)
        path: pr number
                                                       - run: gh pr comment $PR NUMBER -b "
                                                           Buildusuccessful"
```

- We completely control the value of pr_number!
- (\$GITHUB_ENV) is a special file used for setting environment variables
- We can inject a newline in pr_number which allows us to set arbitrary environment variables



Only really one solution:





Only really one solution:
Be *very* careful when working with untrusted data.



Only really one solution: Be *very* careful when working with untrusted data. Verify that the types are correct, that the data is sanitized, that the data is escaped, ...

Bonus problems with pull_request_target

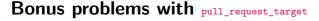


pull_request_target uses the workflow file of the PR target branch





- pull_request_target uses the workflow file of the PR target branch
- All other triggers use the workflow file of the default branch





- pull_request_target uses the workflow file of the PR target branch
- All other triggers use the workflow file of the default branch
- → We can have different workflows for different branches





- pull_request_target uses the workflow file of the PR target branch
- All other triggers use the workflow file of the default branch
- → We can have different workflows for different branches
- Bug might only be fixed on the default branch!





- pull_request_target uses the workflow file of the PR target branch
- All other triggers use the workflow file of the default branch
- → We can have different workflows for different branches
- Bug might only be fixed on the default branch!
- → Just exploit another, still vulnerable branch 🤒



Scenario: pull_request_target workflow, only permissions for comments



- Scenario: pull_request_target workflow, only permissions for comments
- This is safe, right?



- Scenario: pull_request_target workflow, only permissions for comments
- This is safe, right?
- → GitHub Actions provide *caches*



- Scenario: pull_request_target workflow, only permissions for comments
- This is safe, right?
- → GitHub Actions provide *caches*
- Normally they are not shared between pull requests, but for pull_request_target they are!



- Scenario: pull_request_target workflow, only permissions for comments
- This is safe, right?
- → GitHub Actions provide *caches*
- Normally they are not shared between pull requests, but for pull_request_target they are!
- → Poison the cache, hope that a more privileged workflow uses it



- Scenario: pull_request_target workflow, only permissions for comments
- This is safe, right?
- → GitHub Actions provide *caches*
- Normally they are not shared between pull requests, but for pull_request_target they are!
- → Poison the cache, hope that a more privileged workflow uses it
- Profit

Resources



- https://securitylab.github.com/research/github-actions-preventing-pwn-requests/
- https://securitylab.github.com/research/github-actions-untrusted-input/
- https://securitylab.github.com/research/github-actions-building-blocks/
- https://github.blog/ 2023-08-09-four-tips-to-keep-your-github-actions-workflows-secure/
- https://docs.github.com/de/actions/security-guides/ security-hardening-for-github-actions

Questions?

Ping intrigus on Slack

OI

DM @intrigus_ on twitter (note the underscore)

or

shoot me a mail at insecure-gh-actions-23@intrigus.org