# GITHUB ACTIONS VULNERABILITIES

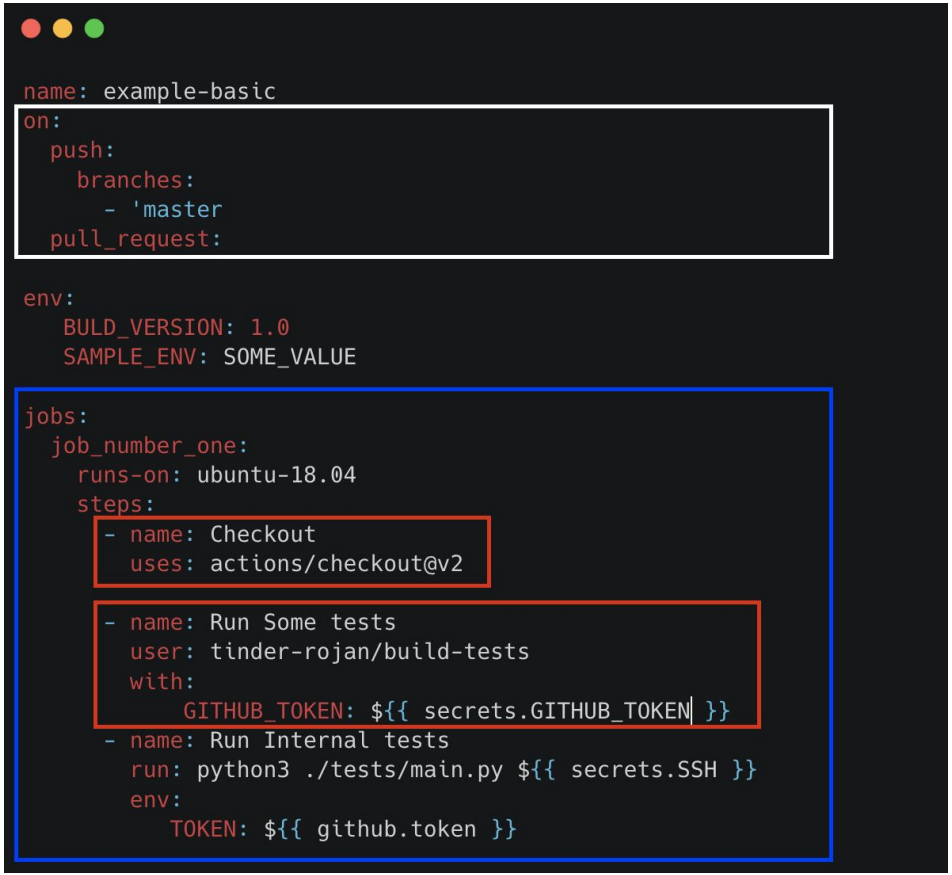
TINDER SECURITY LABS

fwd:cloudsec

# What is GitHub Actions (GHA) tl;dr

- Run automations directly through GitHub via event-driven triggers.
- Actions are defined as workflows written in YAML.
- Common use cases:
  - PR validations
  - Issue triage, stale checks etc
  - Automated deployments to infrastructure stacks

# Sample workflow



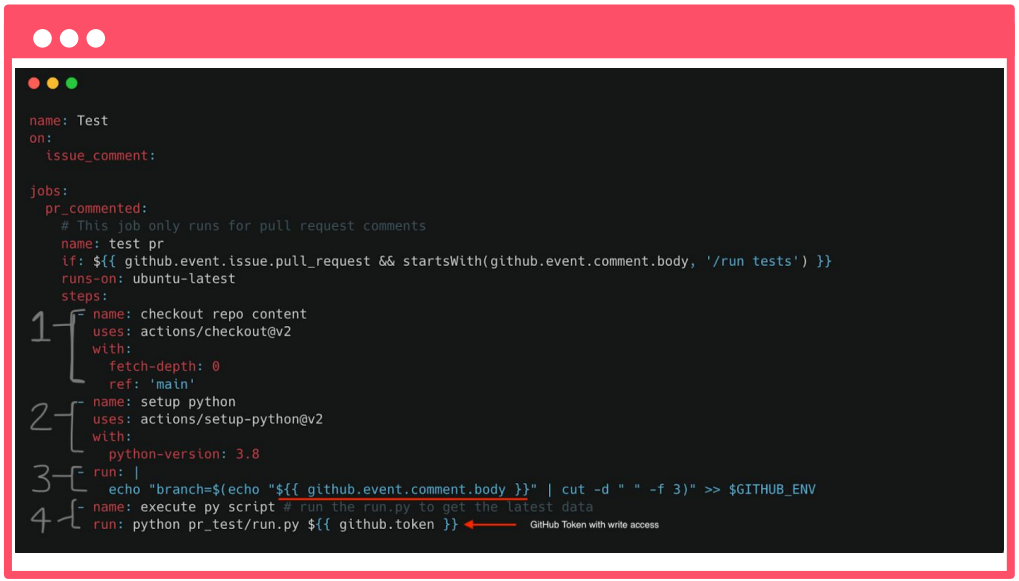
```yaml
name: example-basic
on:
  push:
    branches:
      - 'master
  pull_request:

env:
   BULD_VERSION: 1.0
   SAMPLE_ENV: SOME_VALUE

jobs:
  job_number_one:
    runs-on: ubuntu-18.04
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Run Some tests
        user: tinder-rojan/build-tests
        with:
            GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      - name: Run Internal tests
        run: python3 ./tests/main.py ${{ secrets.SSH }}
        env:
           TOKEN: ${{ github.token }}
```

# User Controlled event triggers

- Some triggers are automated while some are user/guest controlled
  - issue
  - issue_comment
  - pull_request
  - pull_request_target
- issue & issue_comment
  - Triggered when user comments/creates an issue
  - Also triggered in comments made on pull requests
- pull_request & pull_request_target
  - When a PR is created/modified/labeled/etc.

# Vulnerability #1

# User Controlled Input




Event triggers like issue and issue_comment contain user input

- Issue title
- Comment body

Inputs are treated as executable command when used in run with GitHub Context (**${{ github.*** }})versus set as env variable.

# User Controlled Runtime files


It is common to test user PR before merging them.

- pull_request: Limited access to base repository
  - Base repository's secrets are not accessible
  - GITHUB_TOKEN secret is scoped to read access only
- **pull_request_target**
  - Run on base repository and has access to secrets
  - GITHUB_TOKEN is default scoped

# Code Execution Impact

- Retrieve the repository's GITHUB_TOKEN
  - *actions/checkout* (frequently used in workflows) stores an auth token in the git config file
  - ~~Default permissions give write access to the repository and other maintainer level permissions.~~ Default format is read only and now requires specific permissions to be set.
- Retrieve secrets used by other steps (not jobs)
  - Use the initial command execution to overwrite runner files for future steps and retrieve secrets associated with those steps.

# Vulnerability #2

# Supply Chain Exploit

- **Workflows common use third party extensions via *uses***
- **Edge case exploit: *What happens if the maintainer changes their username?***
  - Cloning/using the action should auto-redirect to the new username
  - tinder-rojan/sample-action@v1 ➡️ tinder-rojannew/sample-action@v1
  - **Problem**
    - Can someone claim tinder-rojan and hijack the namespace? (tinder-rojan/sample-action@v1)

# Supply Chain Exploit - Example

- **In 2022, we scanned workflows to find all takeoverable usernames**
- **Example:**
  - *papeloto/action-zip* **used by 316+ workflows including major organizations**
  - **Redacted use case**
    - When PR is merged to main branch run an integration test
    - Run integration test and create a zip file with the integration test result

# Supply Chain Exploit - Example

- **Exploit case:**
  - Successful takeover of papeloto/action-zip namespace allowed RCE on most workflows.

```
1  Warning: Unexpected input(s) 'slack_hook', valid inputs are ['entryPoint', 'args']
2  ▶Run papeloto/action-zip@v1
4  /usr/bin/docker run --name cd98ffeeb6269ecf9420e9ab15cef67c33a35_77653f --label 4cd98f --workdir /github/workspa
   HOME -e GITHUB_JOB -e GITHUB_REF -e GITHUB_SHA -e GITHUB_REPOSITORY -e GITHUB_REPOSITORY_OWNER -e GITHUB_RUN_ID
   GITHUB_RETENTION_DAYS -e GITHUB_RUN_ATTEMPT -e GITHUB_ACTOR -e GITHUB_WORKFLOW -e GITHUB_HEAD_REF -e GITHUB_BASE
   GITHUB_SERVER_URL -e GITHUB_API_URL -e GITHUB_GRAPHQL_URL -e GITHUB_REF_NAME -e GITHUB_REF_PROTECTED -e GITHUB_R
   GITHUB_ACTION -e GITHUB_EVENT_PATH -e GITHUB_ACTION_REPOSITORY -e GITHUB_ACTION_REF -e GITHUB_PATH -e GITHUB_ENV
   RUNNER_OS -e RUNNER_ARCH -e RUNNER_NAME -e RUNNER_TOOL_CACHE -e RUNNER_TEMP -e RUNNER_WORKSPACE -e ACTIONS_RUNTI
   e ACTIONS_CACHE_URL -e GITHUB_ACTIONS=true -e CI=true -v "/var/run/docker.sock":"/var/run/docker.sock" -v
   "/home/runner/work/_temp/_github_home":"/github/home" -v "/home/runner/work/_temp/_github_workflow":"/github/wor
   "/home/runner/work/_temp/_runner_file_commands":"/github/file_commands" -v "/home/runner/work/repo-steal/repo-st
   4cd98f:feeb6269ecf9420e9ab15cef67c33a35
5  Hello world. This user changed their username to vimtor. Please fix your workflow. Code: AL3901C
```

# Supply Chain Exploit - Example

- **Exploit case:**
  - Successful takeover of papeloto/action-zip namespace allowed RCE on most workflows.
  - RCE allowed to write into *main* branch of repositories and steal various secrets
    - AWS Access Key
    - Aws Secret Access Key
    - Private key and private key passphrase
  - RCE on customers of vulnerable company by compromising nightly builds

# Supply Chain Exploit - Securing

- **Use commit hash instead of release tags or branch names when calling third-party GitHub Actions.**
  - *-uses: papeloto/action-zip@26a249fb00d43ca98dad77a4b3838025fc226aa1*
- **Only use actions from trusted/long term maintainers.**
- **Watch your build logs for any potential suspicious activity when running third party actions.**

# Vulnerability #3

# AWS OIDC + GitHub Actions

**Past**

- Required using AWS Access Key and Secret Key to use AWS CLI through GHA.
- Risks of long term APIs being disclosed if a GHA job was exploited

**Now**

- Use OIDC to request a short-term session for a given role in AWS
- Session expires after the job is finished
- Reduces risk of API key disclosures.

# OIDC Setup in AWS

**Securing OIDC for GHA requires creating a Trust Policy to validate:**

- ***aud***
  - **ClientID request sent by GitHub**
  - **Usually it is a repository name: https://github.com/org/repo_name**
  - **Can be customized & static: *sts.amazonaws.com***
- ***sub***
  - **Needed to validate proper access control**
  - **Cannot be customized *fully***
  - **Default header:** *repo:ORG_NAME/REPO_NAME:ref:BRANCH_NAME*

# OIDC - GHA Setup

**Using and setting up an OIDC in GHA is super simple**

- **AWS provided GHA - configure-aws-credentials**

```yaml
- name: Assume the AWS role
  continue-on-error: true
  id: configure-aws-credentials
  if: github.event_name != 'pull_request'
  uses: aws-actions/configure-aws-credentials@v1
    with:
      role-to-assume: arn:aws:iam::223121549624:role/hhvm-github-actions
      aws-region: us-west-2
```

# OIDC & GHA Theory

**There will be some companies/organizations who will incorrectly set up their AWS Trust Policy allowing attackers to get access to their AWS accounts.**

- Misconfiguration most likely to happen if the *sub* header is not validated.

# OIDC & GHA Vulnerability - Example

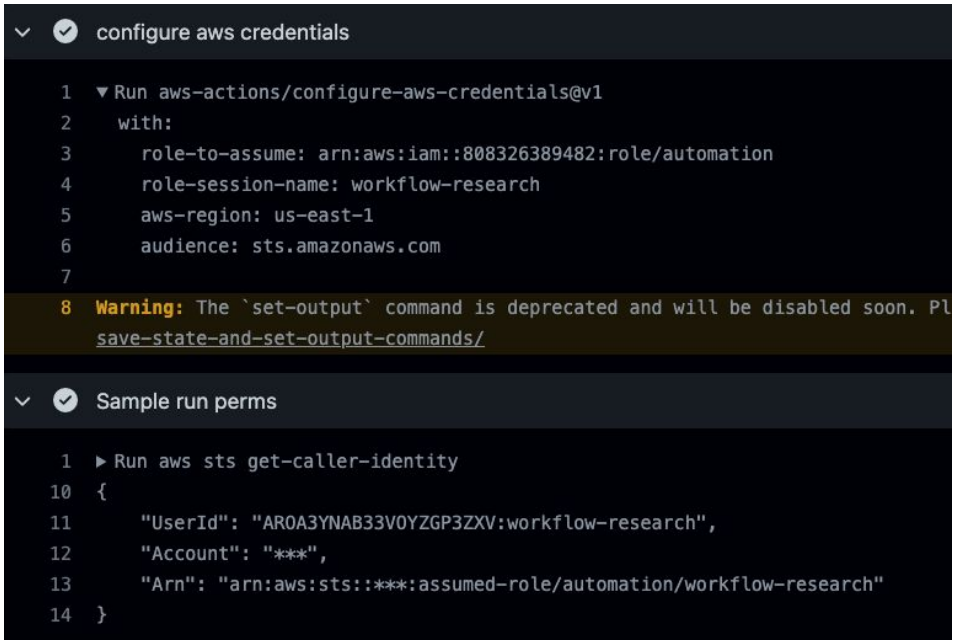
**Affected Organization: AWS**

- **awsdocs/aws-doc-sdk-examples**
  - **Repo with AWS SDK test cases**
  - **Coming soon: Docker images to run sample AWS codes to test our different AWS features.**
  - **Workflow built docker images and pushed to AWS' ECR**

```
33          - name: Configure AWS credentials
34            uses: aws-actions/configure-aws-credentials@master # More informat
35            with:
36              role-to-assume: arn:aws:iam::808326389482:role/automation
37              aws-region: us-east-1
38
```

# OIDC & GHA Vulnerability - Example

## awsdocs/aws-doc-sdk-examples



# OIDC & GHA Vulnerability - Example

**awsdocs/aws-doc-sdk-examples - Investigating further**

- **Looking through past commit, we found additional roles**

```
uses: aws-actions/configure-aws-credentials@master # More info
with:
  role-to-assume: arn:aws:iam::260778392212:role/admin
  role-to-assume: arn:aws:iam::808326389482:role/automation
  aws-region: us-east-1
```

# OIDC & GHA Vulnerability - Example

## awsdocs/aws-doc-sdk-examples - Investigating further

- Looking through past commit, we found additional roles

# OIDC & GHA Vulnerability - Highlights

- We scanned a large dataset of GitHub Action workflows through by extracting IAM roles from matching workflows through Sourcegraph
- Vulnerability highlights
  - 3 vulnerable IAM roles in AWS' repositories/infrastructure
  - Multiple Web3 companies vulnerable
  - Access to a government body's internal infrastructure
  - Multiple educational institutions vulnerable
- ***All vulnerable organizations were notified and vulnerabilities were swiftly patched.***

# Securing your OIDC & GHA

- Check trust policy settings to confirm that all OIDC for GitHub Actions have proper validation in place for *sub* header.
  - Steampipe: https://hub.steampipe.io/plugins/turbot/aws/tables/aws_iam_role#verify-the-trust-policy-of-role-has-validation-conditions-when-used-with-github-actions

**Verify the Trust policy of Role has validation conditions when used with GitHub Actions #**

```
select
  iam.arn as resource,
  iam.description,
  iam.assume_role_policy_std,
  case
    when pstatement -> 'Condition' -> 'StringLike' -> 'token.actions.githubusercontent.com:sub' is not null
    or pstatement -> 'Condition' -> 'StringEquals' -> 'token.actions.githubusercontent.com:sub' is not null then 'ok'
    else 'alarm'
  end as status,
  case
    when pstatement -> 'Condition' -> 'StringLike' -> 'token.actions.githubusercontent.com:sub' is not null
    or pstatement -> 'Condition' -> 'StringEquals' -> 'token.actions.githubusercontent.com:sub' is not null then
iam.arn || ' Condition Check Exists'
    else iam.arn || ' Missing Condition Check'
  end as reason
from
  aws_iam_role as iam,
  jsonb_array_elements(iam.assume_role_policy_std -> 'Statement') as pstatement
where
  pstatement -> 'Action' ? & array [ 'sts:assumerolewithwebidentity' ]
  and (pstatement -> 'Principal' -> 'Federated') :: text like '%token.actions.githubusercontent.com%'
order by
  status asc
```

# Securing your OIDC & GHA

- Open Source Tool
  - https://github.com/TinderSec/oidc-scanner-aws - GitHub Action to test and flag any vulnerable IAM roles used by GitHub workflows

```yaml
name: GHA Scanner Action - OIDC
on:
 workflow_dispatch:

permissions:
  id-token: write
  contents: read

jobs:
  ScanVulns:
    env:
      AWS_REGION: us-east-1
    runs-on: ubuntu-latest
    steps:
      - name: Git clone the repository
        uses: actions/checkout@v1
      - name: Action IP
        run: curl https://ifconfig.me
      - uses: actions/setup-node@v2
      - run: npm install @actions/core@1.6.0-beta.0
      - run: pip install boto3
      - uses: TinderSec/oidc-scanner-aws@main
        env:
          PAT: ${{ github.token }}
        with:
          organization: ORG TO SCAN
```

# Conclusion - Securing your GHA

- Sanitize user inputs before passing them into arguments
- Do not build or run from untrusted / user-controlled files
- Watch for potential supply chain exploits through username takeovers
- Secure your OIDC configuration

# Thank You

- github.com/TinderSec
- medium.com/Tinder
- twitter.com/TinderEng