Authenticating supply-chain metadata

Building remote code attestations on GitHub

Asra Ali, Laurent Simon (Google)

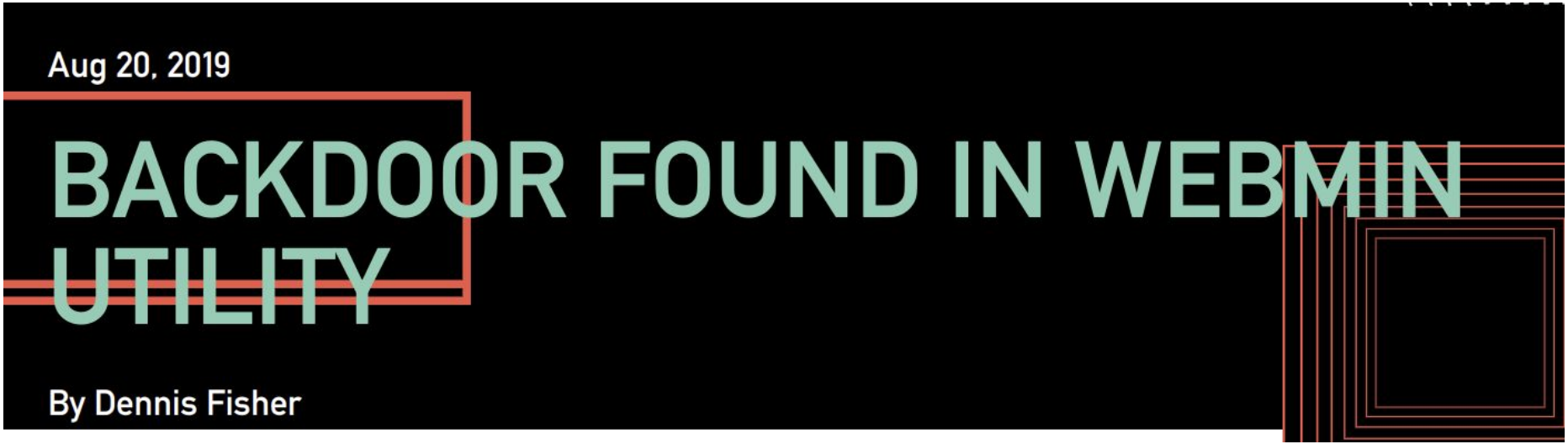
#ossummit @AsraEntr0py @lsim99

THE LINUX FOUNDATION

Agenda

- Motivating exploits
- Remote Code Attestations
- GitHub Reusable Workflows
  - Applications
  - Demo!
- GitHub Actions
  - Applications
- Concluding Thoughts

Supply Chain Compromises: Compromised Build Infra



See more here

Supply Chain Compromises: Malicious Dependency

Alex Birsan Follow

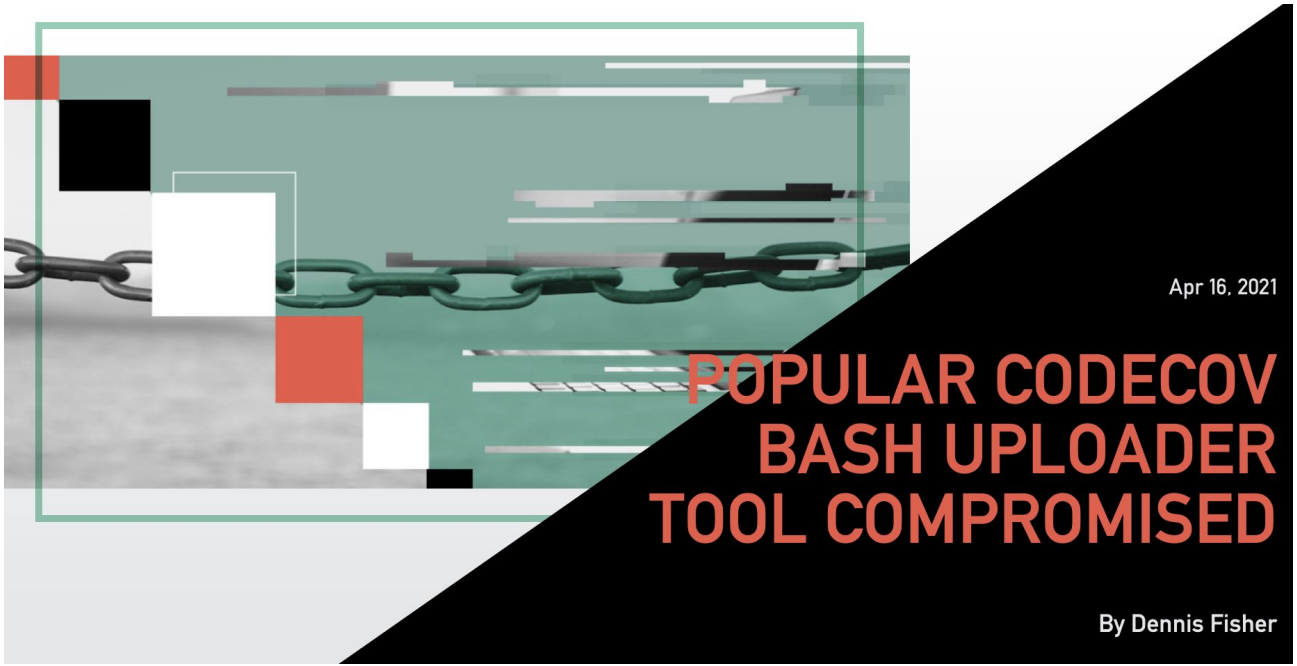
Feb 9, 2021 · 11 min read ★ · Listen

Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies

The Story of a Novel Supply Chain Attack

See more here

Supply Chain Compromises: Malicious Artifact Upload



See more here

Supply Chain Compromises

PyPI package 'keep' uses malicious 'request'

Some versions of PyPI packages, 'keep,' 'pyanxdns,' and 'api-res-py' were caught using a malicious dependency, 'request,'

Large-scale npm attack targets Azure developers with malicious packages

The JFrog Security Research team identified hundreds of malicious packages designed to steal PII in a large scale typosquatting attack

By Andrey Polkovnychenko and Shachar Menashe

March 23, 2022

8 min read

SH AR E:

*We need **trustworthy information** on our software artifacts to secure our pipeline!*

Remote Code Attestation

Code Attestations and Trust

Attestation is the "*issue of a statement, based on a decision, that fulfillment of specified requirements has been demonstrated*." [ISO/IEC 17000]

Code Attestations and Trust

An **attestation** is the a piece of data representing a **proof** of an event.

Code Attestations and Trust

An **attestation** is a piece of data representing a **proof** of an event.

- *Environment*: Where? When?

Code Attestations

An **attestation** is a piece of data representing a **proof** of an event.

- *Environment*: Where? When?
- *Materials*: What is being used?

Code Attestations

An **attestation** is a piece of data representing a **proof** of an event.

- *Environment*: Where? When?
- *Materials*: What is being used?
- *Recipe*: What steps? What configuration?

Code Attestations

An **attestation** is a piece of data representing a **proof** of an event.

- *Environment*: Where? When?
- *Materials*: What is being used?
- *Recipe*: What steps? What configuration?
- ***Subject*: What is the output? Can it be traceable to the materials with the process in the environment?**

Code Attestations

In **software**, we can attest to events like:

- Builds
- Code Scanning
- Code Commits
- Releases
- Vulnerability disclosure

Code Attestations: How would they have helped?

- ✔ **Trace** artifact/binary to its **source code**
- ✔ Assurance no **backdoors** inserted
- ✔ Identify if the artifact uses any malicious **dependencies**

Trust

Code Attestations: Integrity & Trust

How can one **trust** an attestation?

Code Attestations: Integrity & Trust

How can one **trust** an attestation?

- Do I trust the **producer?**

Code Attestations: Integrity & Trust

How can one **trust** an attestation?

- Do I trust the **producer?**
- Do I trust the attestation was not **interfered** with when it was produced?

Code Attestations: Integrity & Trust

How can one **trust** an attestation?

- Do I trust the **producer?**
- Do I trust the attestation was not **interfered** with when it was produced?
- Do I trust the attestation was not **altered**?

GitHub Reusable Workflows

GitHub Workflows

- The standard way to run CI on GitHub, including releases
- Defined in your repository under .github/workflows
- You can run arbitrary commands
- You can define "trigger events": push, pull_request, etc

docs.github.com/en/actions/using-workflows

GitHub Workflow Example

```
name: hello-world
on: push
jobs:
  my-event:
    runs-on: ubuntu-latest
    steps:
      - name: my-step
        run: |
          echo "Hello World!"
```

docs.github.com/en/actions/using-workflows

THE LINUX FOUNDATION

GitHub Workflow Trust

```
name: hello-world
on: push
jobs:
  my-event:
    runs-on: ubuntu-latest
    steps:
      - name: my-step
        run: |
          echo "Hello World!"
```

docs.github.com/en/actions/using-workflows

A Naive Attempt

```
name: hello-world
on: push
jobs:
  my-event:
    runs-on: ubuntu-latest
    steps:
      - name: my-step
        run: |
            echo "Hello World!"
  record-my-event:
    runs-on: ubuntu-latest
    steps:
      - name: generate attestation
        run: |
            …
```

docs.github.com/en/actions/using-workflows

Problems

Interference:
Can my-job, other workflow steps, or maintainers interact with record-my-job to produce false info?

Integrity:
Can we tamper-proof the attestation?

Authenticity:
Can we prove that the attestation was produced in this workflow?

docs.github.com/en/actions/using-workflows

Isolation/Interference

github.com/user/repo

```
name: release CI
on: push
jobs:
  release:
    uses: trusted/builder@v1.2.3
```

Isolation/Interference

github.com/user/repo

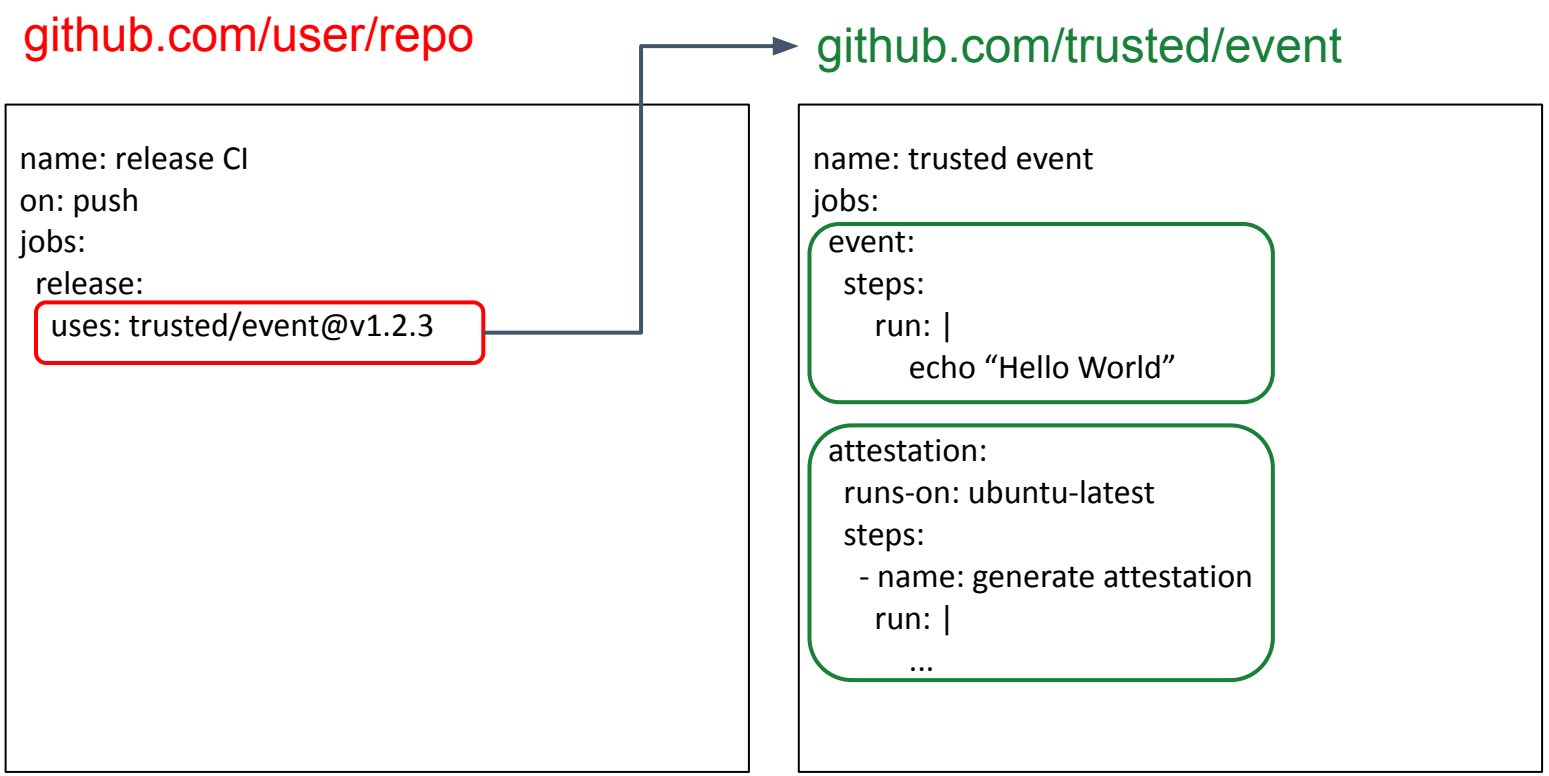
```
name: release CI
on: push
jobs:
  release:
    uses: trusted/event@v1.2.3
```

github.com/trusted/event

```
name: trusted event
jobs:
  event:
    steps:
      run: |
        echo “Hello World”

  attestation:
    runs-on: ubuntu-latest
    steps:
      - name: generate attestation
        run: |
          ...
```

Isolation/Interference



Problems

Interference:
Can my-job, other workflow steps, or maintainers interact with record-my-job to produce false info?

docs.github.com/en/actions/using-workflows

Key Management: Authenticity & Build Identity

Interference:
Can my-job, other workflow steps, or maintainers interact with record-my-job to produce false info?

Integrity:
Can we tamper-proof the attestation?

Authenticity:
Can we prove that the attestation was produced in this workflow?

docs.github.com/en/actions/using-workflows

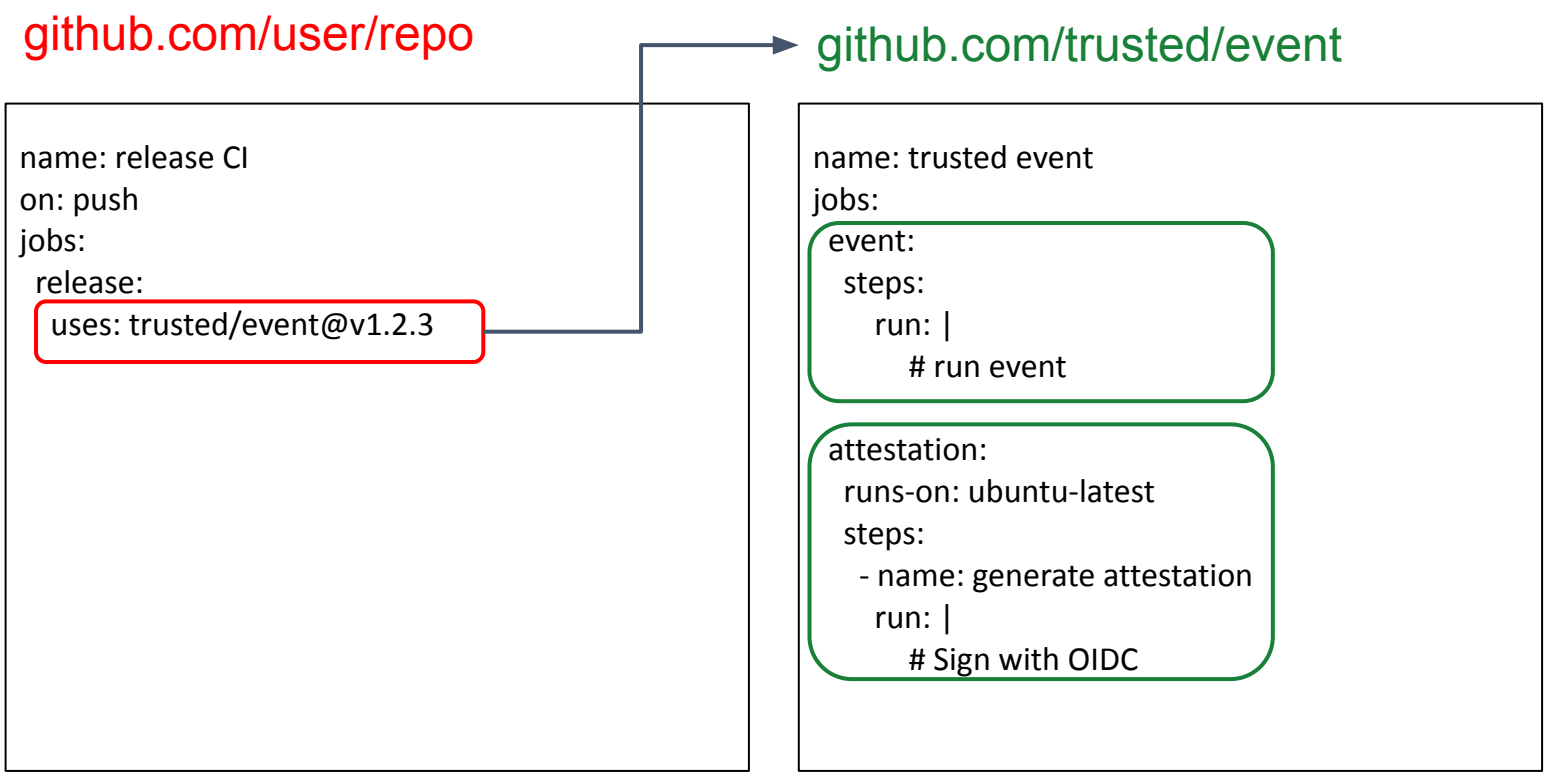
Signatures with Authenticity

- Uses "workload identity" (similar to SPIFFE)
- Using OpenID Connect (OIDC), trusted builder is provisioned with a singing certificate
- Certificate that signs the provenance:
    - X509v3 SubjectAlt: **github.com/trusted/builder@v1.2.3**

Sigstore Keyless on GitHub

```
            X509v3 Subject Alternative Name: critical
                URI:https://github.com/slsa-framework/slsa-github-generator/.github/workfl
ows/builder_go_slsa3.yml@refs/tags/v1.1.1
            1.3.6.1.4.1.57264.1.1:
                https://token.actions.githubusercontent.com
            1.3.6.1.4.1.57264.1.6:
                refs/tags/v1.0.0
            1.3.6.1.4.1.57264.1.3:
                ce8ec300588fac82a8dadb90958afa0aaff7a5a1
            1.3.6.1.4.1.57264.1.2:
                push
            1.3.6.1.4.1.57264.1.5:
                asraa/slsa-example
            1.3.6.1.4.1.57264.1.4:
                SLSA go releaser
```

Putting it together: Attestations on GitHub

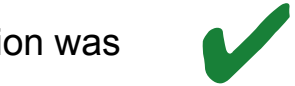


A setup for verifiable attestations on GitHub

Interference:
Can my-job, other workflow steps, or maintainers interact with record-my-job to produce false info?

Integrity:
Can we tamper-proof the attestation?

Authenticity:
Can we prove that the attestation was produced in this workflow?



docs.github.com/en/actions/using-workflows

Verifying Attestations

Given an attestation:

1. Verify the signature on the attestation: **integrity**
2. Verify the prover identity: **authenticity & isolation**
3. *Now you have trust; so verify the statement content!*

Applications

Artifact Attestation - SLSA

Artifact/Release Attestation

https://github.com/slsa-framework/slsa-github-generator

Artifact Attestation - SLSA

Establish a source-to-artifact provenance that allows verifying claims about a binary.

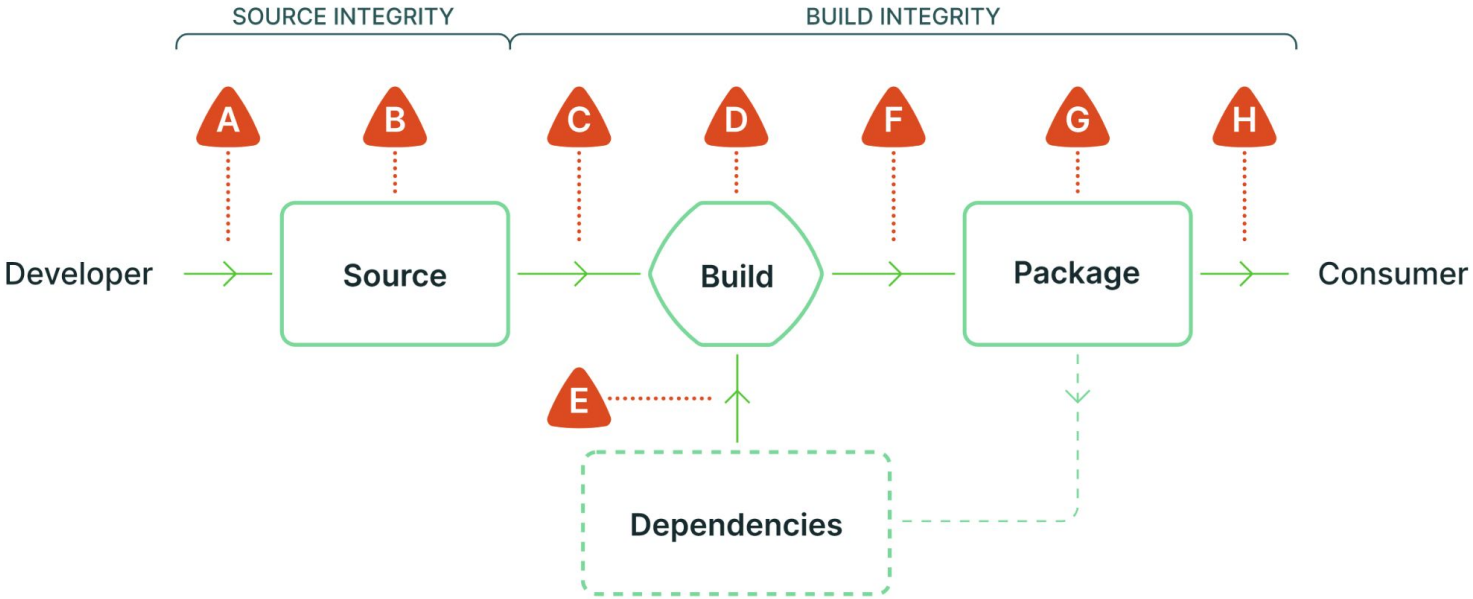
What were the sources?

Where was it built?

How was it built?

https://github.com/slsa-framework/slsa

Threats in the software pipeline



A Submit unauthorized change

B Compromise source repo

C Build from modified source

D Compromise build process

E Use compromised dependency

F Upload modified package

G Compromise package repo

H Use compromised package

Use case: GitHub Dependency API (GET)


```
{
  "change_type": "added",
  "manifest": "package.json",
  "ecosystem": "npm",
  "name": "helmet",
  "version": "4.6.0",
  "package_url": "pkg:npm/helmet@4.6.0",
  "source_repository_url": "https://github.com/helmetjs/helmet",
},
```


https://docs.github.com/en/rest/dependency-graph/dependency-review#get-a-diff-of-the-dependencies-between-commits

Use case 1: GitHub Dependency API (GET)


```
{
  "change_type": "added",
  "manifest": "package.json",
  "ecosystem": "npm",
  "name": "helmet",
  "version": "4.6.0",
  "package_url": "pkg:npm/helmet@4.6.0",
  "source_repository_url": "https://github.com/helmetjs/helmet",
},
```


https://docs.github.com/en/rest/dependency-graph/dependency-review#get-a-diff-of-the-dependencies-between-commits

Use case 2: Policies

Policy for control plane, e.g. k8's policy engine

- OPA gatekeeper https://github.com/open-policy-agent/gatekeeper
- Kyverno https://kyverno.io/

Build time

- Verification of a container's base image

Installation time

- Library, package (npm, pip, apt-get, etc.)

https://github.com/slsa-framework/slsa

Use case 3: GitHub Dependency API (POST)

- Package managers, *like Gradle and sbt, dependencies cannot reliably be parsed statically. Dependencies are determined at build time*

- POST API - SBOM

https://github.blog/2022-06-17-creating-comprehensive-dependency-graph-build-time-detection/

Use case 4: other metadata

- Coverage results for CodeCov
- Static analyzers

Release of SLSA3 Go builders

Blog:

https://slsa.dev/blog/2022/06/slsa-github-workflows

Code:

https://github.com/slsa-framework/slsa-github-generator

(Generic SLSA generator coming in July!)

Demo!

SLSA Golang builders

Try out the SLSA3 builders, let us know on Twitter #ossf #ossummit @AsraEntr0py @lsim99

OpenSSF's sos.dev rewards developers for improving their supply chain

https://github.com/slsa-framework/slsa-github-generator

GitHub Actions Workflows

Crowdsourcing Trusted Computations

Can we achieve the same without reusable workflows?

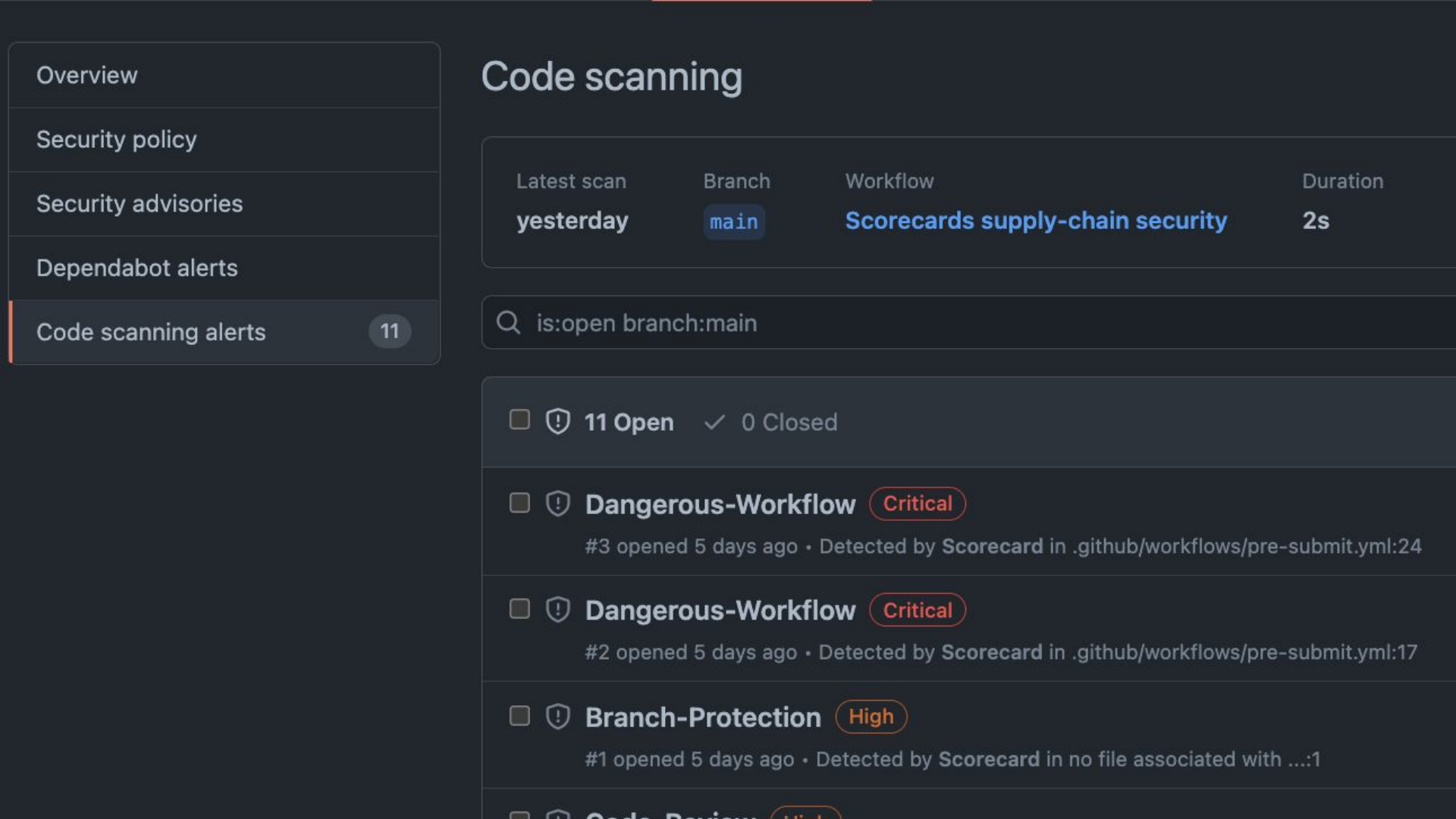
https://github.com/ossf/scorecard-action

Crowdsourcing Trusted Computations

Motivating example: crowdsource scorecard results

- GitHub Action
- Weekly scans for > 1M projects

https://github.com/ossf/scorecard-action


Overview
Security policy
Security advisories
Dependabot alerts
Code scanning alerts 11
Code scanning
Latest scan
yesterday
Branch
main
Workflow
Scorecards supply-chain security
Duration
2s
is:open branch:main
11 Open
0 Closed
Dangerous-Workflow
Critical
#3 opened 5 days ago • Detected by Scorecard in .github/workflows/pre-submit.yml:24
Dangerous-Workflow
Critical
#2 opened 5 days ago • Detected by Scorecard in .github/workflows/pre-submit.yml:17
Branch-Protection
High
#1 opened 5 days ago • Detected by Scorecard in no file associated with ...:1

Crowdsourcing Trusted Computations

Attestation:

*How can a GitHub Action user **attest** that their result was produced by the scorecard-action?*

https://github.com/ossf/scorecard-action

Crowdsourcing Trusted Computations

Problem: no isolation guarantees for an Action

https://github.com/ossf/scorecard-action

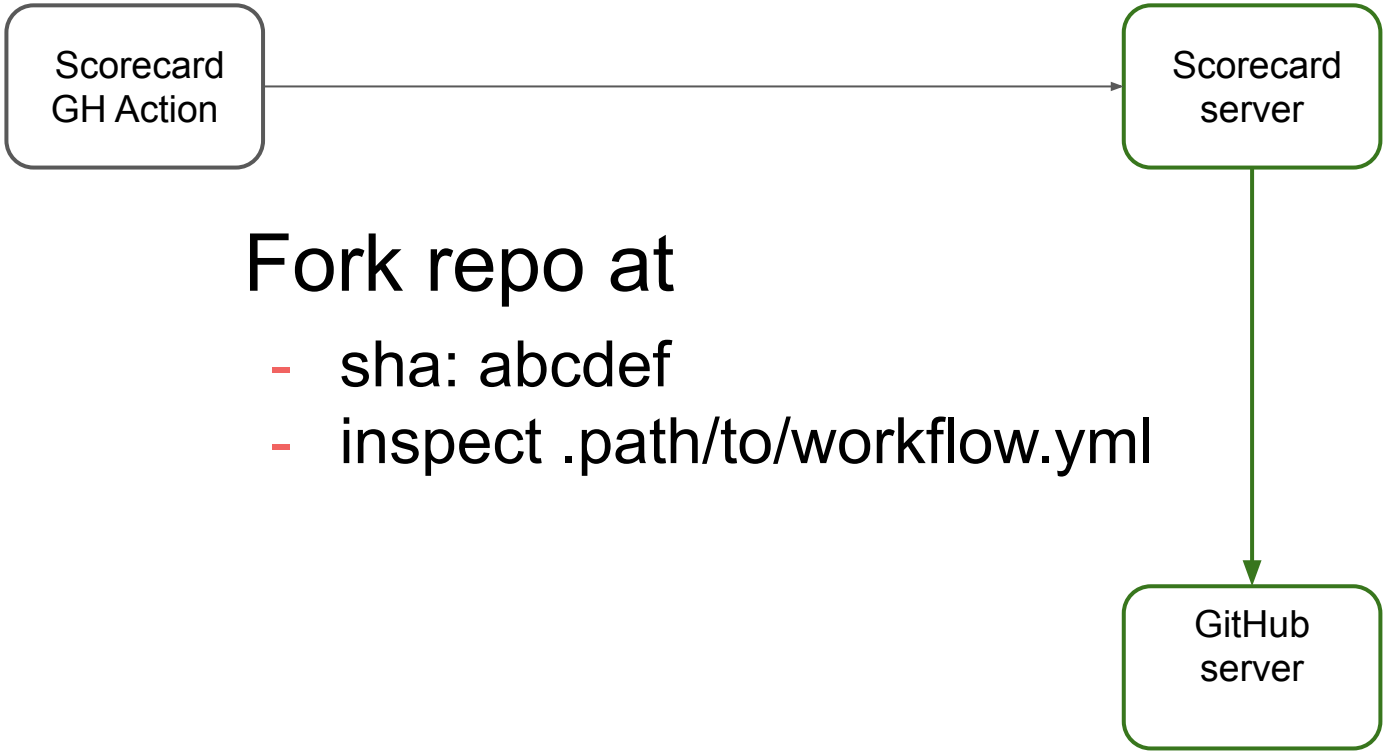
Crowdsourcing Trusted Computations

Signing certificate with OIDC token

- sha: abcdef
- repository_name
- workflow_path: .path/to/workflow.yml

https://github.com/ossf/scorecard-action

Crowdsourcing Trusted Computations



https://github.com/ossf/scorecard-action

Crowdsourcing Trusted Computations

Validation of workflow

- ✔ No GitHub-hosted runner
- ✔ Correct permissions
- ✔ No additional script, jobs, etc
- ✔ No user-provided containers or services
- ✔ ...

https://github.com/ossf/scorecard-action

Conclusion

Conclusion

- Signing with OIDC gives integrity and authenticity
- Proving isolation with GitHub actions is tricky, but can still be achieved
- *Use GitHub Reusable workflows for isolation for free!*

Conclusion

- Try out our SLSA building workflows today
- Easter egg: There are easter eggs in the github.com/asraa/slsa-example. Try verifying the provenance :)
- Let us know on Twitter #ossf #ossummit @AsraEntr0py @lsim99

Thank you!

https://slsa.dev/
https://www.sigstore.dev/

Sigstore Slack
SLSA Get Involved
SLSA vs. Software Supply Chain Attacks
Achieving SLSA 3 Compliance with GitHub Actions and Sigstore for Go modules
Builder v1 release

THE LINUX FOUNDATION


SUPPLYCHAIN
SECURITYCON
@
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
THE LINUX FOUNDATION

Binary Attestation

Transparent Release