sigstore

Sigstore Policy Controller

(and creating your own policies)

John Osborne
josborne@chainguard.dev
@CloudLvlMidnite

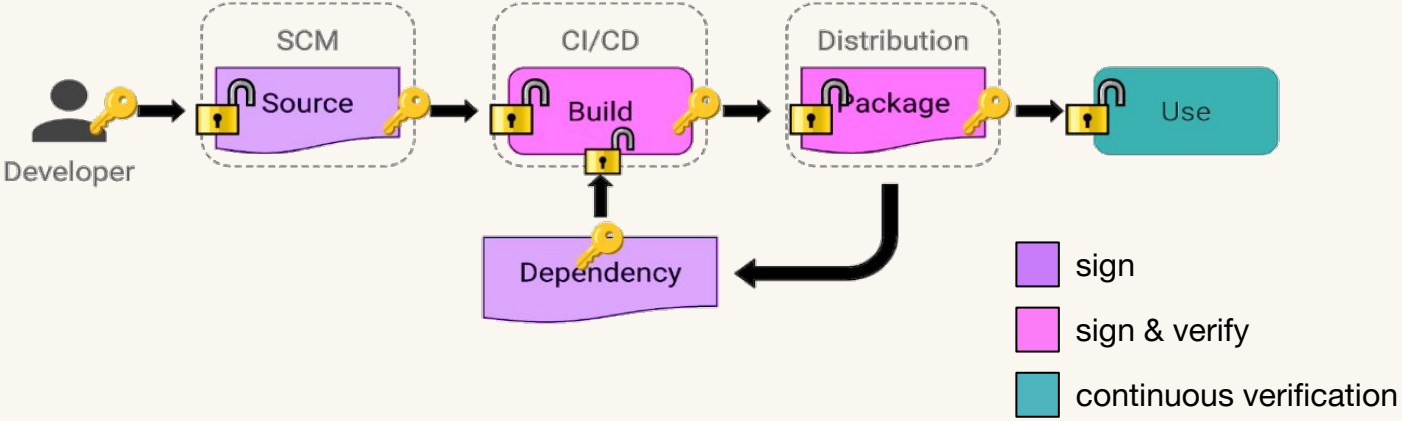
A Quick Primer on Sigstore

Automate Signing and Verification of Handoffs

SLSA

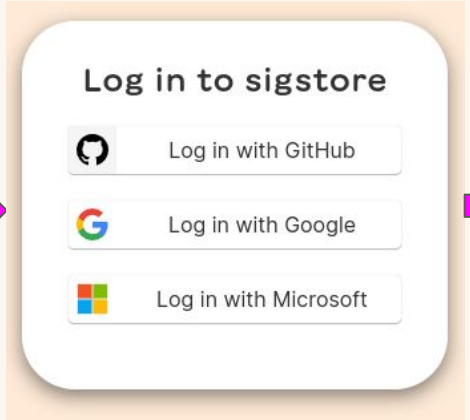
NIST SSDF

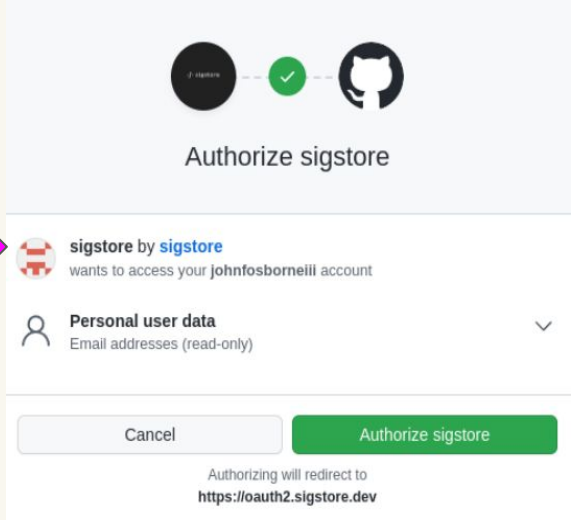
CIS Benchmarks



Easy and Automated Signing

```
josborne@fedora-system76:~
❯ cosign sign-blob frontend-deployment.yaml
Using payload from: frontend-deployment.yaml
Generating ephemeral keys...
Retrieving signed certificate...
```





Git commit signing

+ sigstore = sigstore gitsign

Sign a commit
git commit -S

Verify a commit
git verify-commit <revision>

```
❯ git log --show-signature -1
commit 2c86a54dc2269b22198d0c2235e2dba9cc7fad27 (HEAD ->
tlog index: 5841082
gitsign: Signature made using certificate ID 0x067f00b53a
gitsign: Good signature from [josborne@chainguard.dev]
Validated Git signature: true
Validated Rekor entry: true
Author: John Osborne <josborne@chainguard.dev>
Date:   Tue Oct 25 10:34:38 2022 -0400

    adding date to date.out
```

- Signature stored in the git commit
- Certificate valid for 10 min

sigstore cosign

Easy and Automated Signing

sigstore gitsign

Images $ cosign sign josborne/myimage

Blobs $ cosign sign-blob ./myblob

Attestations $ cosign attest josborne/myimage \
--predicate ./att.json

gitsign must be installed
git commits $ git commit -S

Images $ cosign verify josborne/myimage

Blobs $ cosign verify-blob ./myblob

Attestations $ cosign verify-attestation josborne/myimage

signature and rekor log index available in git commit
git commits $ git log --show-signature

Sigstore is GA

ACTIVE CONTRIBUTORS

666 ↑111.43%

4000+ GitHub Stars

6.4M+ Log Entries

Chainguard, Google, Red Hat, Sonatype, Amazon, Philips, SUSE, VMWare, Citi, Anchore, etc

Validate 3rd Party Dependencies with Sigstore

Maven™

RubyGems

TEKTON

crates.io
Browse All Crates
The Rust community's crate registry

npm

KUBEWARDEN

python™

Artifact HUB

GitHub

goreleaser
GoReleaser is a release automation tool for Go projects

urllib3

flux

PyPi

Kyverno

KEDA

Cython

Sigstore Policy Controller

(Kubernetes Admission Go/No-Go)

Common Supply Chain Policies

```
cosign attest --type slsaprovenance
```

Sign


```yaml
_type: https://in-toto.io/Statement/v0.1
predicateType: https://slsa.dev/provenance/v0.2
predicate:
  builder:
    id: https://github.com/laurentsimon/slsa-github-generator-ko/.github/workflows/slsa3-builder.yml@
  buildType: https://github.com/slsa-framework/slsa-github-generator-go@v1
  invocation:
    configSource:
      uri: git+https://github.comlaurentsimon/slsa-on-github-test@refs/heads/main.git
      digest:
        sha1: ad1ada158145ccfa006aac936061d030468542f
      entryPoint: Ko Caller
  buildConfig:
    steps:
      - command:
          - ~/go/bin/ko
          - publish
          - '-B'
          - '--platform=linux/amd64,linux/arm64,linux/386,linux/arm'
          - '--tags=tag5,tag6'
          - laurentsimon/helloworld
        env:
          - KO_DOCKER_REPO=laurentsimon/helloworld
  materials:
    - uri: git+laurentsimon/slsa-on-github-test.git
      digest:
        sha1: ad1ada158145ccfa006aac936061d030468542f
```


Verify

```
cosign verify-attestation
```

Authenticated & Non-Falsifiable
Build Service

Common Supply Chain Policies

```
cosign attest --type slsaprovenance
```

Dan Signs

```
cosign attest --type slsaprovenance
```

Kim Signs

```
_type: https://in-toto.io/Statement/v0.1
subject:
- digest:
    sha1: 859b387b985ea0f414e4e8099c9f874acb217b94
predicateType: https://chainguard.dev/CodeReview/v1
predicate:
  repo:
    uri: https://github.com/example/my-project
    branch: main
  author: mailto:engineering@chainguard.dev
  reviewers:
  - mailto:dan@chainguard.dev
  - mailto:kim@chainguard.dev
```

Code Review

Verify Dan

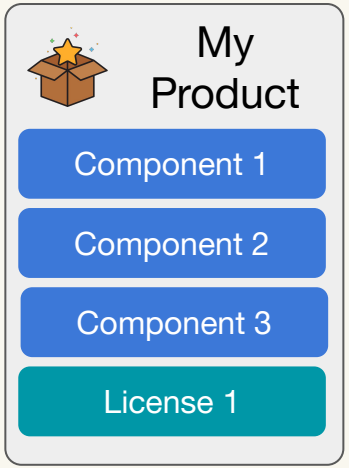
```
cosign verify-attestation
```

Verify Kim

```
cosign verify-attestation
```

Common Supply Chain Policies

Signed SBOMs ✅




```
kind: ClusterImagePolicy
metadata:
  name: log4shell
spec:
...
  attestations:
    - predicateType: cyclonedx
      name: log4shellcyclonedx
      policy:
...
          let log4shell_names = ["log4j-api", "log4j-core"]
          let log4shell_versions = ["2.0-beta9", "2.0-rc1", "2.0-rc2", "2.0", "2.0.1",
              "2.0.2", "2.1", "2.2", "2.3", "2.4", "2.4.1", "2.5"]
          predicate: {
...
            list.Contains(log4shell_versions, version) {
            err: strings.Join(["Error: CycloneDX SBOM contains package",
                name, "version", version, "which is",
                "vulnerable to Log4Shell (CVE-2021-44228)"
            ], " ")
            name: err
...
    - predicateType: spdxjson
```

Verify

```
cosign verify-attestation
```

Validate SBOM Content ✅

Admission - How To Pass

- ClusterImagePolicy key sections
  - glob: what images
  - authorities: who/what signed them
  - attestations (optional): signed evidence
- A single CIP is validated when
  - Any authority signs/attest image ✅
- An image is admitted when
  - All CIP pass ✅

```yaml
apiVersion: policy.sigstore.dev/v1beta1
kind: ClusterImagePolicy
metadata:
  name: chainguard-sbom-spdx-keyless-attestation
spec:
  images:
  - glob: "distroless.dev/*"
  authorities:
  - name: keyless
    keyless:
      url: "https://fulcio.sigstore.dev"
      identities:
        - issuer: "https://accounts.google.com"
          subjectRegExp: ".+@chainguard.dev$"
    attestations:
    - name: must-have-spdx
      predicateType: spdx
      policy:
        type: cue
        data: |
          predicateType: "https://spdx.dev/Document"
```

Signature SPEC

Reference Table

Sigstore Policy Controller

- Opt-in per namespace to enable
- Remove namespace label to disable
- Granular label selector

```
kubectl label ns default \
 policy.sigstore.dev/include=true --overwrite
```

```yaml
apiVersion: cosigned.sigstore.dev/v1beta1
kind: ClusterImagePolicy
metadata:
  name: image-policy
spec:
  match:
  - resource: jobs
    group: batch
    version: v1
  - resource: pods
    version: v1
  images:
```

(Optional)

Note: Chainguard migrating all base policies upstream

CIP Modes - What To Do There Is A Match

```yaml
apiVersion: policy.sigstore.dev/v1beta1
kind: ClusterImagePolicy
metadata:
  name: vuln-cve-2022-42889-text4shell
spec:
  mode: warn
  images:
  - glob: "ghcr.io/chainguard-dev/*"
  authorities:
```

Per Policy

```
❯ kubectl get configmap config-policy-controller
   --namespace=cosign-system -o yaml | pygmentize
apiVersion: v1
data:
  no-match-policy: warn
kind: ConfigMap
metadata:
  creationTimestamp: "2022-10-20T16:03:23Z"
  name: config-policy-controller
  namespace: cosign-system
```

Per Cluster

warn | deny | admit

Building ClusterImagePolicy

Artifact signing

- Only images and authorities required
- Optionally static pass/fail (catch-all scenario)

Attestations

- Built-in support for in-toto
- Built-in schemas (predicate types)
- ***For custom data validations we leverage the power of CUE, here's a primer…***

```yaml
apiVersion: policy.sigstore.dev/v1beta1
kind: ClusterImagePolicy
metadata:
  name: chainguard-sbom-spdx-keyless-attestation
spec:
  images:
  - glob: "distroless.dev/*"
  authorities:
  - name: keyless
    keyless:
      url: "https://fulcio.sigstore.dev"
      identities:
        - issuer: "https://accounts.google.com"
          subjectRegExp: ".+@chainguard.dev$"
    attestations:
    - name: must-have-spdx
      predicateType: spdx
      policy:
        type: cue
        data: |
          predicateType: "https://spdx.dev/Document"
```

CUE for Data Validation

Key Things to Know about CUE	Applicability to ClusterImagePolicy
CUE is a JSON superset	Validate based on JSON/JSON Schema
CUE treats types/values the same	Validate based on types or values
Order is irrelevant	Validate consistently (trim inferred rules)
Define how flexible you want to be	Validate with flexibility for future changes

Looking for specific syntax? Check the SPEC

CUE for Data Validation

CUE is a JSON superset

- Opens large ecosystem
- Most tools output JSON
- Use CUE or raw JSON for data validation
- Nice tooling
  - **$ cue import** # JSON schema ⇒ CUE
  - **$ cue export** # CUE ⇒ JSON
  - **$ cue eval** # Validate JSON
- Unmarshal embedded JSON (double encoded attestations)

```
❯ cat brussels.json
{
    "brussels": {
        "name": "Brussels",
        "population": 1200000,
        "capital": true
    }
}
```

```
❯ cue eval brussels.json
brussels: {
    name:       "Brussels"
    population: 1200000
    capital:    true
}
```

CUE for Data Validation

CUE ignores the order of the rules

- Reduces complexity
- Less brittle and error prone
- Reduce boilerplate
- Allows for CUE trim…

```
predicateType: "cosign.sigstore.dev/attestation/vuln/v1"
predicate: {
  scanner: {
    result: {
      runs: [...{
        tool: {
          driver: {
            rules: [...{
              id: id
                properties: {
                  "security-severity": string
                  severityFloat: strconv.ParseFloat(properties."security-severity"
                  if severityFloat > 9.0 {
                    expectedError: "no error",
                    err: strings.Join(["Error: contains high severity vulnerabilit

                    expectedError: err
                }
              }
            }]
          }
        }
      }]
    }
  }
}
```

CUE trim

Order independence allows CUE to remove explicit constraints that are already inferred

```
❯ cue trim cve.cue -s --outfile=cve-trim.cue
```

```
predicateType: "cosign.sigstore.dev/attestation/vuln/v1"
predicate: scanner: result: runs: [...{
        tool: driver: rules: [...{
                id: id
                properties: {
                        "security-severity": string
                        severityFloat:       strconv.ParseFloat(properties."security-severity"
                        if severityFloat > 9.0 {
                                expectedError: "no error"
                                err:           strings.Join(["Error: contains high severity vu
                                expectedError: err
                        }
                }
        }]
}]
```

CUE Data Validation In Action

Let's incrementally validate this attestation!

- **Validate repo** 1
- **Validate author's email domain** 2
- **Validate the review is independent** 3

```
{
  "repo": {
    "type": "git",
    "uri": "https://github.com/example/my-project",
    "branch": "main"
  },
  "author": "mailto:alice@example.com",
  "reviewer": "mailto:bob@example.com"
}
```

codereview.json

You may notice…

- There's only two fields to validate
- When the predicateType is set to custom the data gets cast into one long JSON encoded string

Attestation


```
_type: https://in-toto.io/Statement/v0.1
predicateType: cosign.sigstore.dev/attestation/v1
subject:
  - name: gcr.io/image-scans/cncf-webinar
    digest:
      sha256: f6827ec9bd51ad519e6e3aabb2e18487a852b34f57178166f8f70b81bff89b26
predicate:
  Data: "{\n\t\"repo\": {\n\t\t\"type\": \"git\",\n\t\t\"uri\": \"https://githu
  Timestamp: '2022-11-07T18:12:41Z'
```


https://rekor.tlog.dev/?logIndex=6691180

Use strings or JSON

```
policy:
    type: cue
    data: |
      predicate: {
          Data: !~ "(.*)(bad){3}(.*)"
      }
```

Parse the string
Fail if the attestation has the phrase badbadbad

```
policy:
  type: cue
  data: |
    import (
        "encoding/json"
    )
    predicate: {
        Data: string
        jsonData: {...} & json.Unmarshal(Data)
    }
```

Parse JSON
Built-in CUE packages to Unmarshal and validate

Validate Repo Data

```
"repo": {
  "type": "git",
  "uri": "https://github.com/example/my-project",
  "branch": "main"
},
```

Validates if the following is true: 1

- **branch can be set to main or origin/main**
- **uri must**
  - **Be a string type**
  - **Start with https://github.com/example/**
- **type is an optional string**
- **open struct (allow for other fields later)**

```
#Repo: {
    branch: "main" | "origin/main"
    uri: string
    uri: =~ "https:\/\/github.com\/example\/.*$"
    type?: string
    ...     // open struct
}

predicate: #Predicate & {
    Data: string
    jsonData: {...} & json.Unmarshal(Data) & {
      repo: #Repo
    }
}
```

```
cosign verify-attestation \
    --policy=repo-check.cue \
    --type=custom \
    gcr.io/image-scans/cncf-webinar@sha256:f6
```

Validate Email Domain of Author

```
"author": "mailto:alice@example.com",
```

Validates if the following is true: 2

- **author must**
  - **Be a string type**
  - **end with @example.com**

```
cosign verify-attestation \
    --policy=author-email.cue \
    --type=custom \
    gcr.io/image-scans/cncf-webinar@sha256:f6
```

```
#AuthorEmail: {
    author: string
    author: =~ "^.*@example.com$"
}
predicate: #Predicate & {
  Data: string
  jsonData: {...} & json.Unmarshal(Data) & {
    author: #AuthorEmail.author
  }
}
```

Validate Independent Review

```
"author": "mailto:alice@example.com",
"reviewer": "mailto:bob@example.com"
```

Validates if the following is true: 3

- **author and reviewer must**
  - **both be strings**
  - **not be equal**

```
cosign verify-attestation \
    --policy=independent-review.cue \
    --type=custom \
    gcr.io/image-scans/cncf-webinar@sha256:f6
```

```
#IndependentReview: {
author:   string
reviewer: string & !=author
...
}
predicate: #Predicate & {
  Data: string
  jsonData: {...} & json.Unmarshal(Data) & {
    #IndependentReview
  }
}
```

Putting It All Together

Let's put this all together!

- Aggregate CUE Definitions ✅
- CUE trim (optional) ✅
- cosign verify-attestation ✅
- Copy/Paste into CIP ✅

```
policy:
  type: cue
  data: |
    ...
    predicate: #Predicate & {
        Data: string
        jsonData: {...} & json.Unmarshal(Data) & {
          repo: #Repo
          author: #AuthorEmail.author
          #IndependentReview
        }
    }
```

Note: See repo for full example

Thank You

All code from this presentation can be found at:
https://github.com/chainguard-dev/sigstore-custom-policies

Check the CUE SPEC for a syntax guide

sigstore.dev
Slack channel