KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

KubeCon CloudNativeCon Europe 2022

Securing Your Container Native Supply Chain with SLSA, Github and Tekton

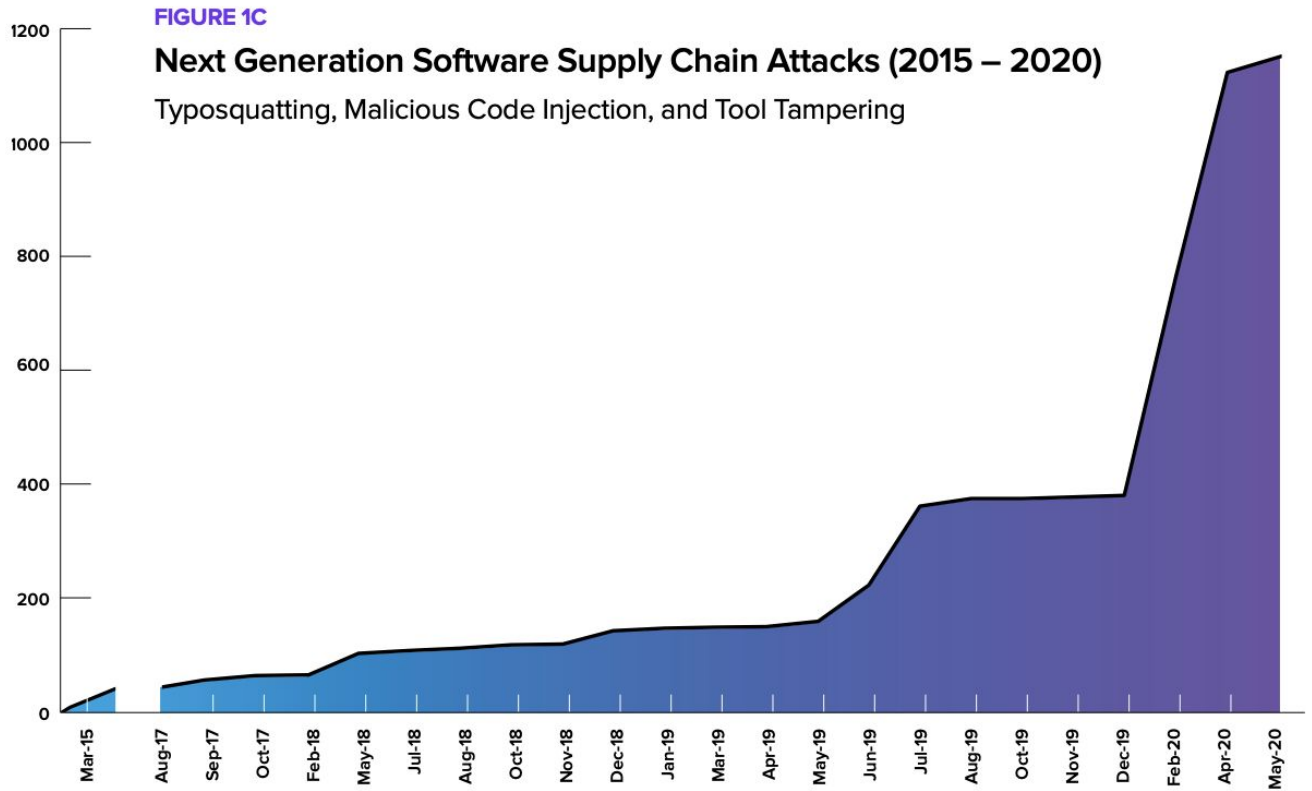
Laurent Simon, Google
Priya Wadhwa, Chainguard

About Us!

Laurent Simon
Security Engineer
Google

Priya Wadhwa
Software Engineer
Chainguard

Supply chain attacks on the rise



Source: https://www.sonatype.com/hubfs/Corporate/Software%20Supply%20Chain/2020/SON_SSSC-Report-2020_final_aug11.pdf

If you use Tekton or Github Workflows, you can secure your pipeline today!

Agenda

- SLSA Overview
- Sigstore
- Achieving SLSA 2 with Tekton
  - Demo!
- Achieving SLSA 3 with Github Workflows
  - Demo!

What is SLSA?

It's a **security framework** to help make
build artifacts tamper resistant.

Includes **metadata** ("provenance") about
who and **how** the build artifacts are created.

Limitations of signatures

- ✗ **Cannot trace** artifact/binary to its **source code**
- ✗ Git **repository**, **branch**, **version**, **tag**
- ✗ **Backdoors** inserted

Example of attack: recent npm color package attack

SLSA benefits

- ✔ **Trace** artifact/binary to its **source code**
- ✔ Identify git **repository**, **branch**, **version**, **tag**
- ✔ Assurance no **backdoors** inserted

SLSA Framework

Level	Description	Example
1	Documentation of the build process	Unsigned provenance
2	Tamper resistance of the build service	Hosted source/build, signed provenance
3	Extra resistance to specific threats	Security controls on host, non-falsifiable provenance
4	Highest levels of confidence and trust	Two-party review + hermetic builds

slsa.dev

SLSA Level 1

- Scripted build process
- Provenance is available

SLSA Level 2

- Source Code is version controlled
- Build service
- Provenance is authenticated
- Provenance is service generated

How will Tekton & Github Workflows apply the SLSA Framework and start signing?

Sigstore

Open source project making signing and verifying software easy and accessible

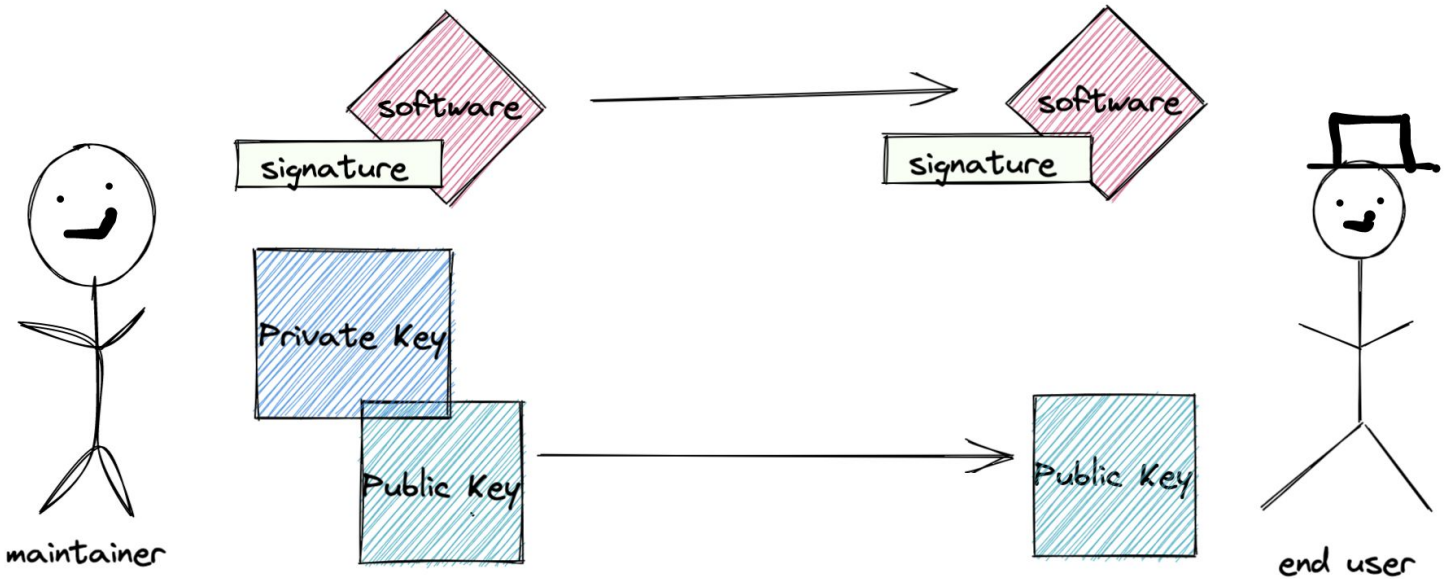
Cosign: CLI Tool and library for signing and verification

Fulcio: Certificate authority which issues short-lived code signing certificates

Rekor: Append-only transparency logs for storing signatures

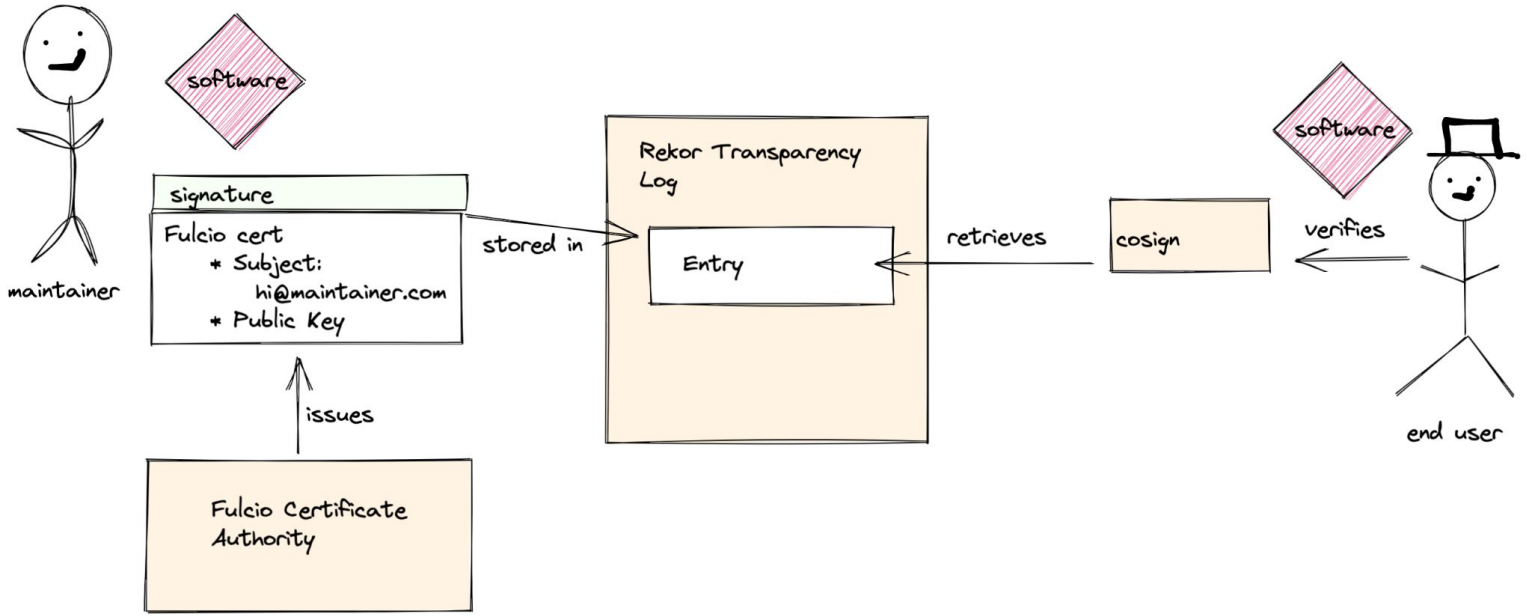
sigstore.dev

Sigstore eliminates the key management problem



sigstore.dev

Sigstore provides code-signing certs mapped to an identity



sigstore.dev

People and systems can request Fulcio certificates

- People can sign in with an email address
- Workloads can use the SPIFFE SVID specification
- Kubernetes Service Account
- Github Actions invocations

sigstore.dev

Tekton

What is Tekton?

- Continuous delivery system built on Kubernetes
- Leverages CRDs to run Tasks and Pipelines
- You can run Tekton Tasks and Pipelines on any cluster with Tekton installed

github.com/tektoncd/pipeline

Tekton Tasks

- The Tekton Task is the basic unit of configuration in Tekton
- You can deploy a Task to your cluster
- You can instantiate the Task with the `tkn` CLI into a TaskRun

```
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: hello
spec:
  steps:
    - name: hello
      image: ubuntu
      command:
        - echo
      args:
        - "Hello World!"
```

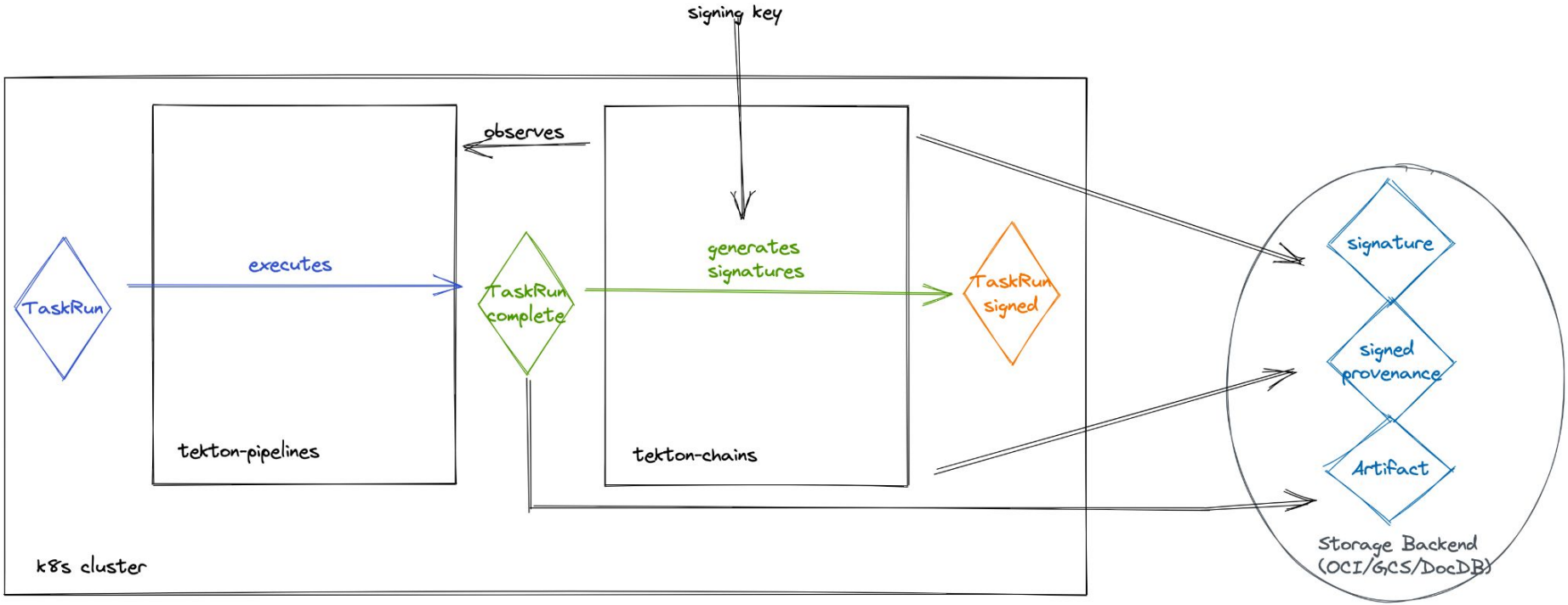
Outputs of a TaskRun

- Whether the TaskRun was successful
- What steps were run
- Any artifacts that may have been built

Tekton Chains: Supply Chain Security for Tekton

- Manages supply chain security for Tekton
- Leverages Sigstore under the hood
- Signs artifacts
- Generates signed provenance

github.com/tektoncd/chains


signing key
observes
TaskRun
executes
TaskRun complete
generates signatures
TaskRun signed
tekton-pipelines
tekton-chains
k8s cluster
signature
signed provenance
Artifact
Storage Backend (OCI/GCS/DocDB)

Demo!

SLSA 2 Requirements

Source - Version controlled	✓
Build - Scripted build	✓
Build - Build service	✓
Provenance - Available	✓
Provenance - Authenticated	✓
Provenance - Service generated	✓

slsa.dev/spec/v0.1/requirements

Tekton + Tekton Chains =

Github Actions and Workflows

What are 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-job:
    runs-on: ubuntu-latest
    steps:
      - name: my-step
        run: |
          echo "Hello World!"
```

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

GitHub Workflow example


```
name: hello-world
on: push
jobs:
  pr:
    steps:
      run: |
        ./run_pull_request.sh
  secret:
    steps:
      run: |
        ./access_secret.sh
```

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

SLSA 3 Requirements

Build - Ephemeral environment	✓ Fresh VM
Build - Isolated	✓
Provenance - Non forgeable	✓

slsa.dev/spec/v0.1/requirements

Builder isolation?

github.com/user/repo

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

Builder isolation?

github.com/user/repo

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

github.com/trusted/builder

```
name: trusted builder
jobs:
  build:
    steps:
      run: |
        ko publish

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

Builder isolation?

github.com/user/repo

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

github.com/trusted/builder

```
name: trusted builder
jobs:
  build:
    steps:
      run: |
        ko publish

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

Signature generation

- 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**

Web PKI Vs SLSA PKI

Web PKI

CA = Verisign,

Website certificate = www.google.com

SLSA PKI

CA = Sigstore

Builder certificate = github.com/trusted/builder

How to verify provenance


```
$ ./slsa-verifier
```

--artifact-oci container:tag

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

How to verify provenance


```
$ ./slsa-verifier
  --artifact-oci container:tag
  --source github.com/origin/repo
```

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

How to verify provenance


```
$ ./slsa-verifier
  --artifact-oci container:tag
  --source github.com/org/reponame
  --branch main
```

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

How to verify provenance


```
$ ./slsa-verifier
  --artifact-oci container:tag
  --source github.com/org/reponame
  --branch main
  --versioned-tag v1.2
```

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

Verification

- ✔ No public key key management. Built-in

Demo!

https://github.com/laurentsimon/kubecon-eu22

Github Actions and Workflows=

Thank you!

Laurent Simon
Twitter: lsim99@
Github: laurentsimon@

Priya Wadhwa
Twitter: priyawadhwa16@
Github: priyawadhwa@