
Making it easy to get to SLSA level 2
Theofilos Petsios (@theofilospe)

(or)

meet compliance requirements

hassle-free

while also getting value for your org

Software Supply Chain Attacks Gaining Popularity

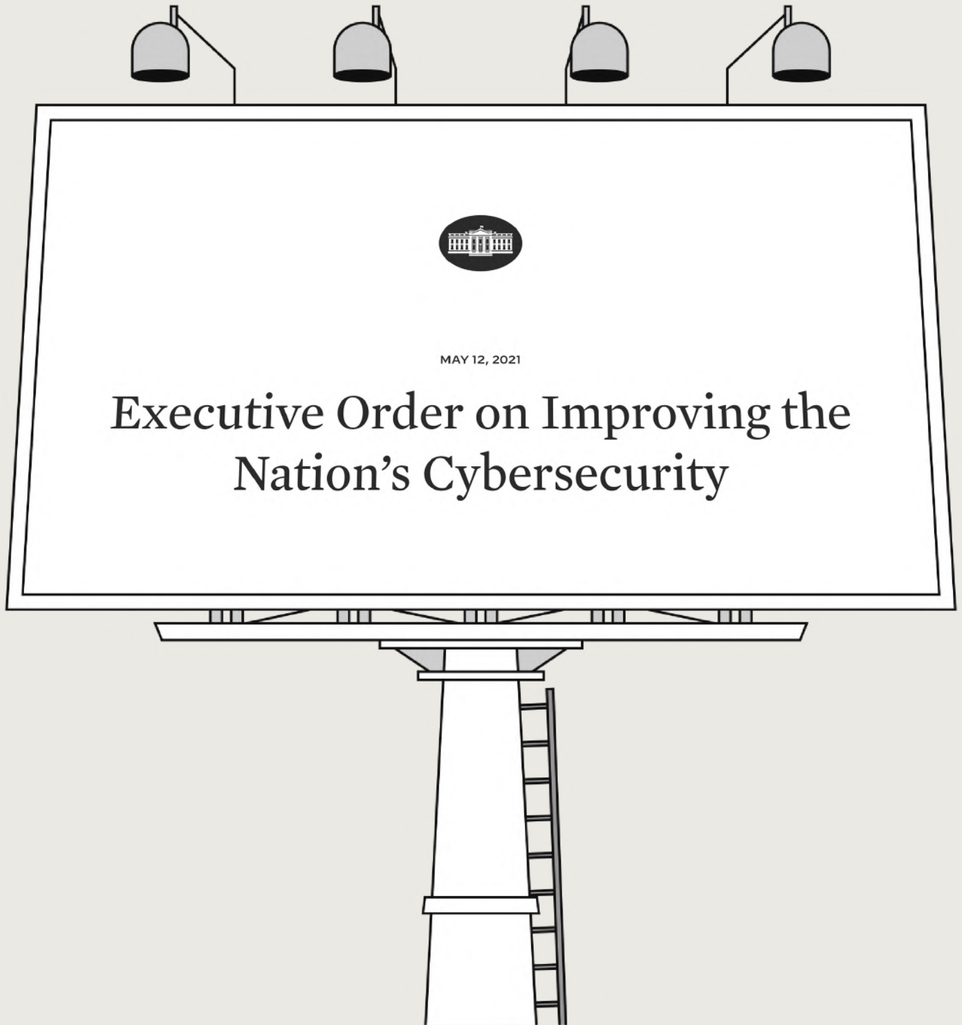
Sonatype → Average Yearly Increase Since 2019 was 742%

Help Net Security → Supply Chain Attacks Surpass Malware-Based Attacks by 40% (2022)

Synopsys & Capterra → 61% of US Businesses Impacted by a Supply Chain Attack (April 2022 - April 2023)

Popularity Drives Policy Changes

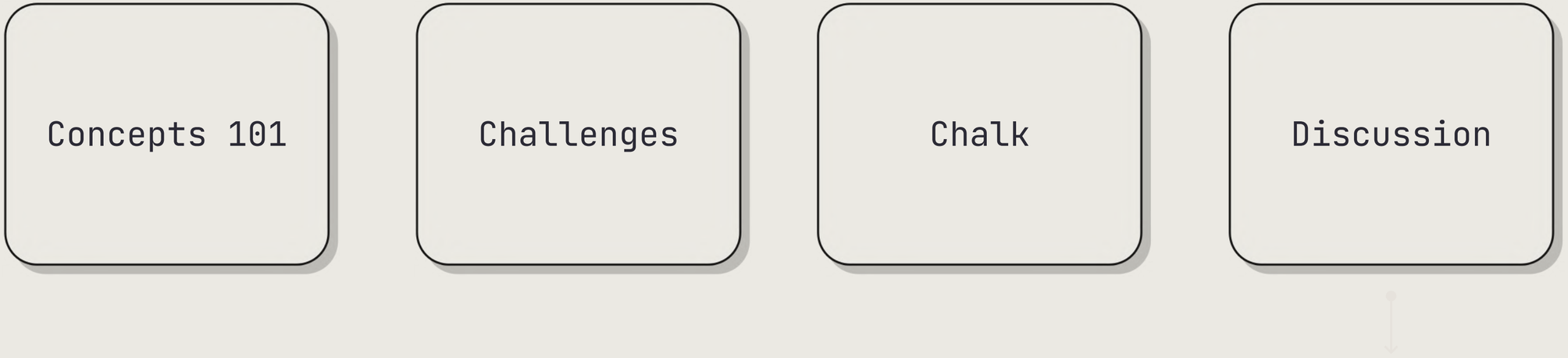
… guidance shall include standards, procedures, or criteria regarding:



> providing a purchaser a Software Bill of Materials (SBOM) for each product directly or by publishing it on a public website

> maintaining accurate and up-to-date data, provenance (i.e., origin) of software code or components

Overview



Software Bills of Materials (SBOMs)

"List of ingredients" for software


```
{
  "bomFormat": "CycloneDX",
  "specVersion": "1.2",
  "serialNumber": "urn:uuid:371ffb8c-c11e-42b5-b5b9-9280fc62783e",
  "version": 1,
  "metadata": {
    "timestamp": "2020-08-03T08:53:09.834Z",
    "tools": [
      {
        "vendor": "CycloneDX",
        "name": "Node.js module",
        "version": "2.0.0"
      }
    ],
    "component": {
      "type": "library",
      "bom-ref": "pkg:npm/protonmail-web@4.0.0-beta.20",
      "name": "protonmail-web",
      "version": "4.0.0-beta.20",
      "description": "Angular frontend for protonmail.com",
      "licenses": [
        {
          "license": {
            "id": "MIT"
          }
        }
      ],
      "purl": "pkg:npm/protonmail-web@4.0.0-beta.20",
      "externalReferences": [
        {
          "type": "website",
          "url": "https://github.com/ProtonMail/WebClient#readme"
        },
        {
          "type": "issue-tracker",
          "url": "https://github.com/ProtonMail/WebClient/issues"
        },
        {
          "type": "vcs",
          "url": "git+https://github.com/ProtonMail/WebClient.git"
        }
      ]
    }
  },
  "components": [
    {
      "type": "library",
      "bom-ref": "pkg:npm/%40babel/polyfill@7.10.4",
      "group": "@babel",
      "name": "polyfill",
      "version": "7.10.4",
      "description": "Provides polyfills necessary for a full ES2015+ environment",
      "hashes": [
        {
          "alg": "SHA-512",
          "content": "f0161c9d5a90e64303d875e81c89c11fb1f56ffb8fdca767c026173aa1675ea82e3b2baee38dd65eeb1b2146d611f6376744f19343
        }
      ],
      "licenses": [
        {
          "license": {
            "id": "MIT"
          }
        }
      ],
      "purl": "pkg:npm/%40babel/polyfill@7.10.4",
      "externalReferences": [
        {
```

Provenance

How did the artifact get here?

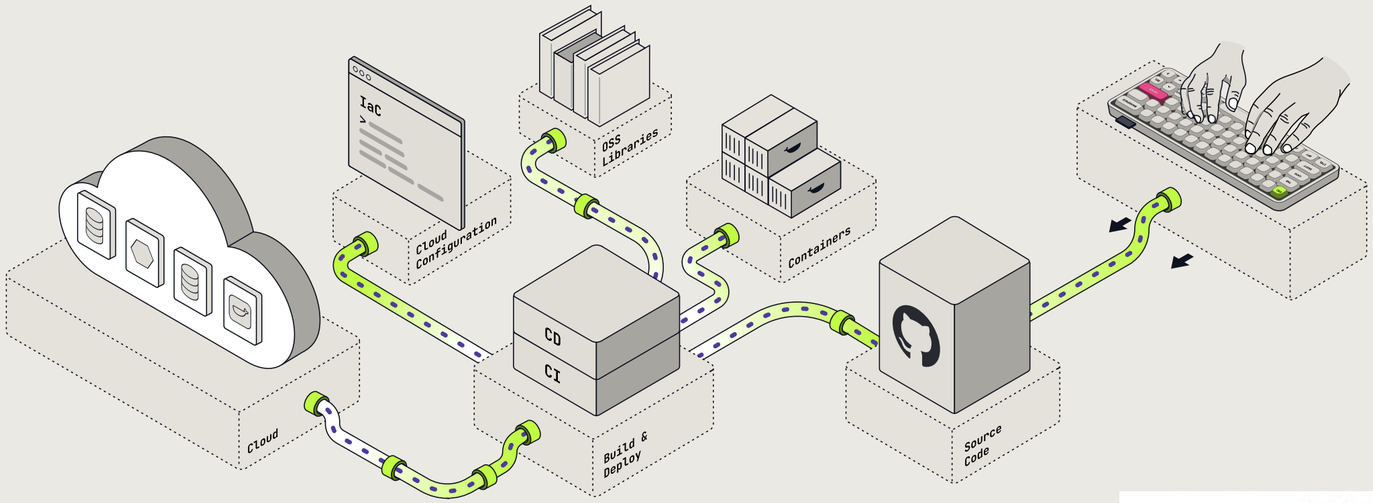
- Who created it?
- Who packaged it?
- Who transported it?

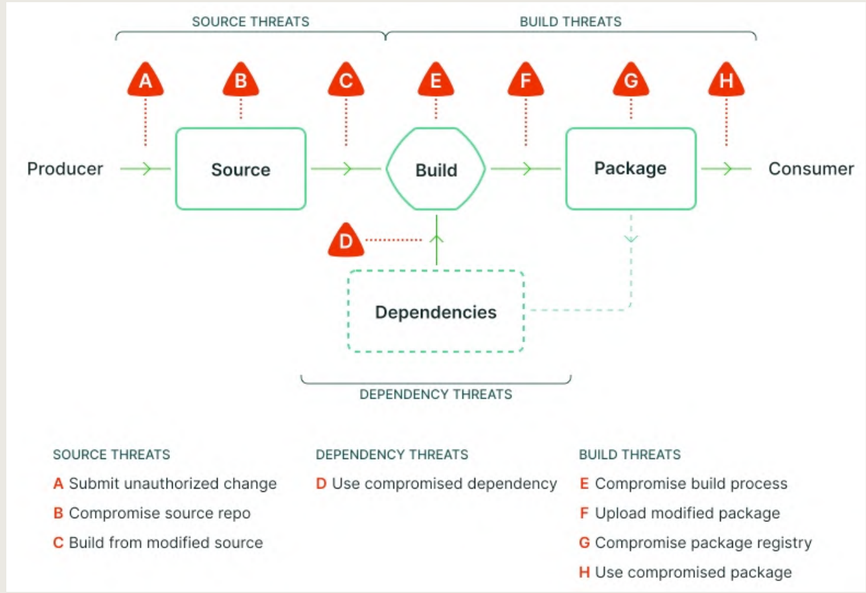
Reflections on Trusting Trust

Attestation

- Need to ensure data is not tampered with and comes from a trusted source
- Anyone should be able to verify the integrity
- Clear assumptions around the threat model

DevOps Pipelines In Practice





Source: slsa.dev

Supply chain Levels for Software Artifacts (SLSA)

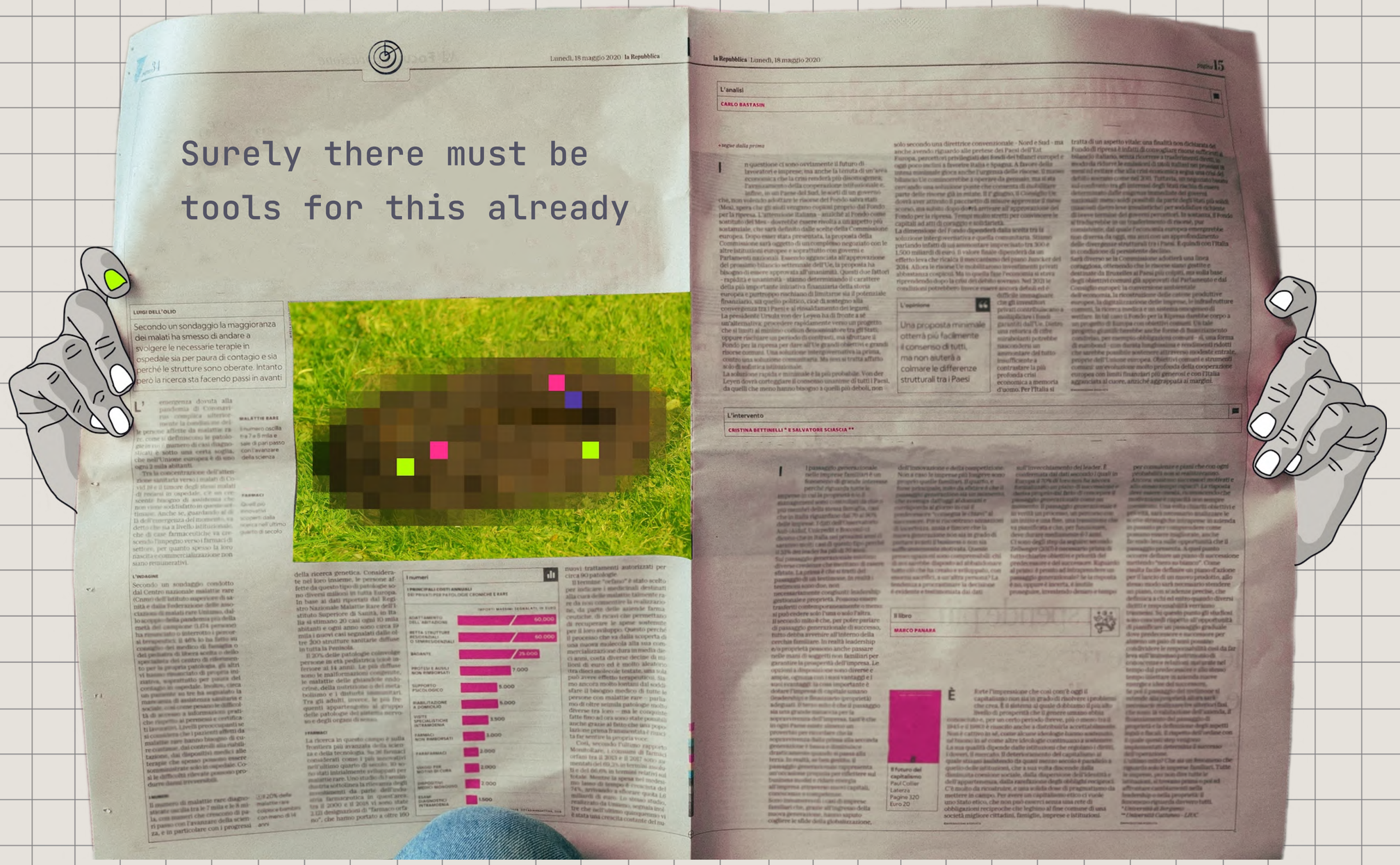
L1: Provenance shows how the package was built (documentation)
L2: Signed provenance, generated by hosted build platform (tampering post build)
L3: Hardened build platform (tampering during the build)

Implementer	Requirement	Degree	L1	L2	L3
Producer	Choose an appropriate build platform		✓	✓	✓
	Follow a consistent build process		✓	✓	✓
	Distribute provenance		✓	✓	✓
Build platform	Provenance generation	Exists	✓	✓	✓
		Authentic		✓	✓
		Unforgeable			✓
	Isolation strength	Hosted		✓	✓
		Isolated			✓

Source: slsa.dev

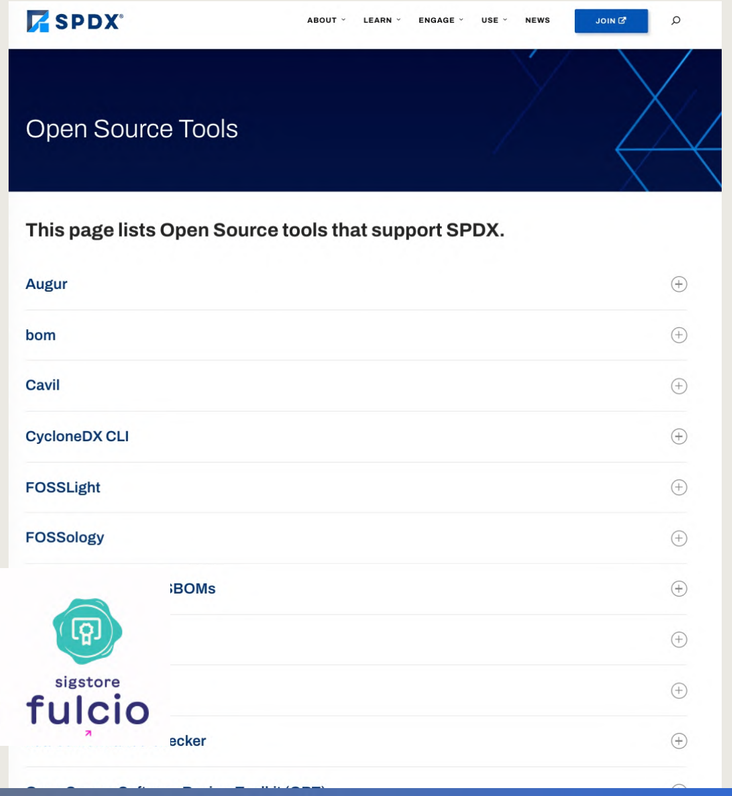
ok, how do I do this?

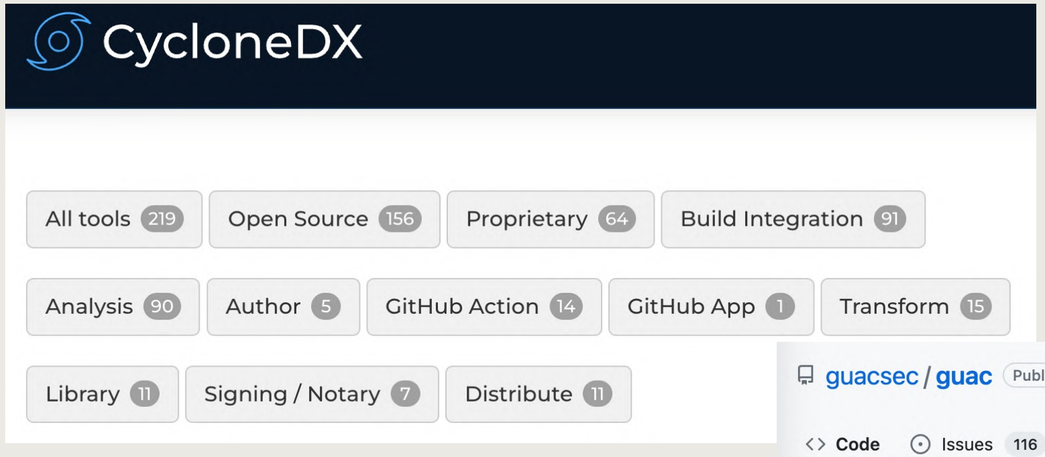
Surely there must be tools for this already

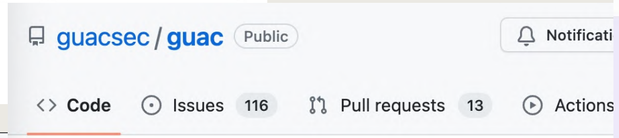


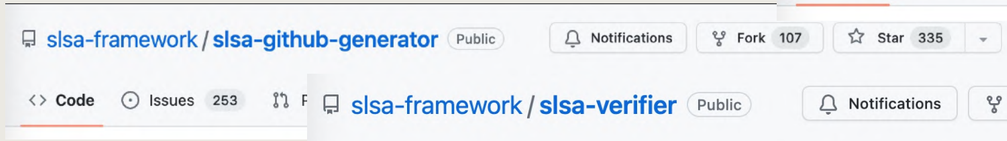
Where do I start?

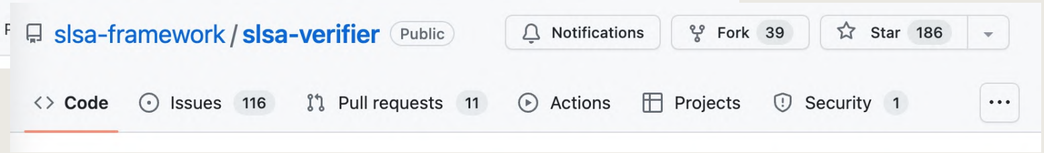
- Which standard / tool do I use?
- Outputs are largely inconsistent
- How do I deploy these / consume the data?

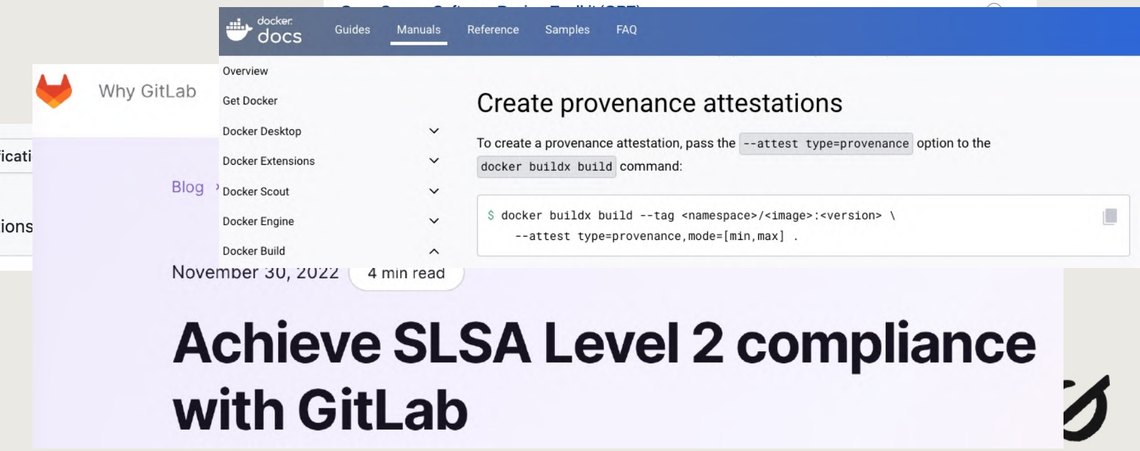




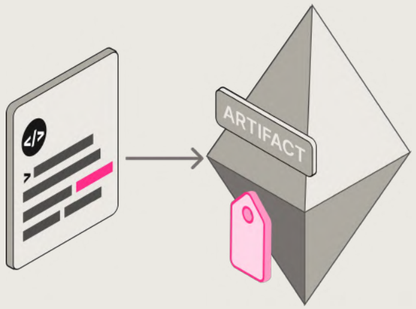




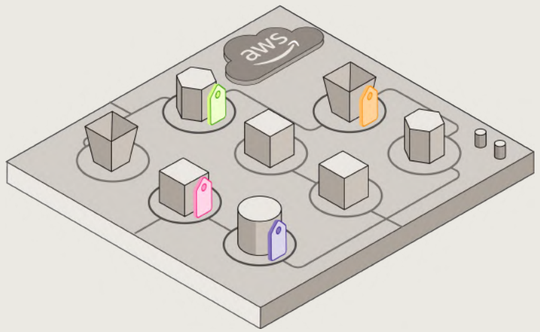




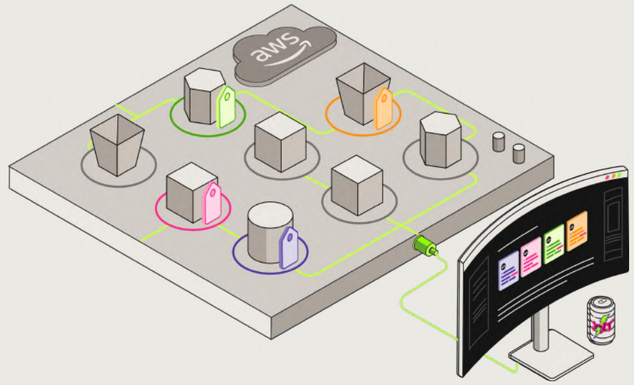
Introducing Chalk



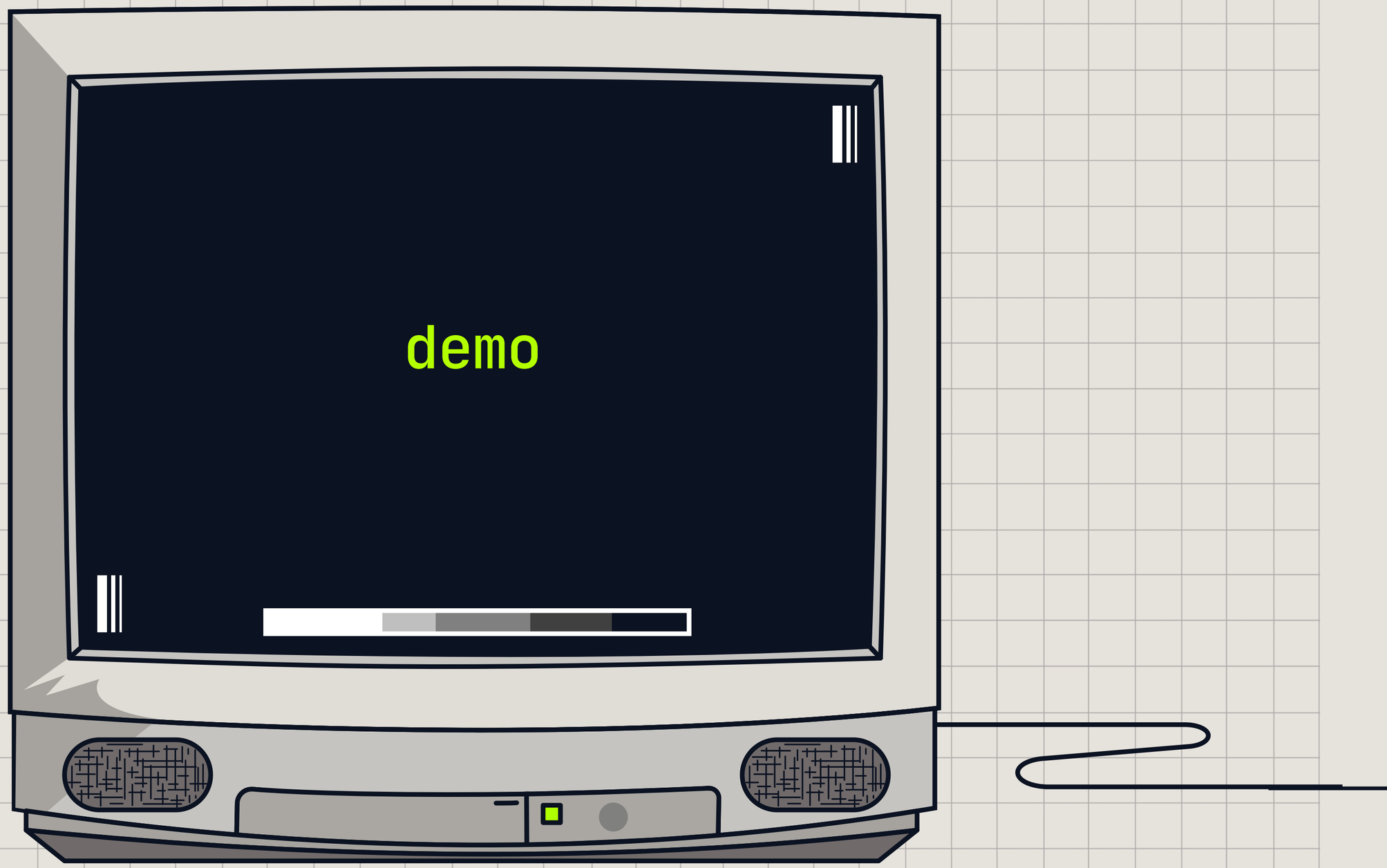
Embed metadata (chalk marks) into your artifacts (binaries/images etc.) during, or post build time



Deploy artifacts to production



Get metadata feeds (chalk reports) from artifacts running in production


demo

Deploying With Chalk

✅ Generate and distribute SBOMs

SBOM	{ "syft": { "$schema": "http://cyclonedx.org/sc

✅ Provenance

exec	crashappsec/chalk-demo
build	crashappsec/chalk-demo

✅ Artifact integrity

Details

Action ID: 36d5a4c99830d8bc

Operation: build

Datetime:

Signing data collected

Compliance:

✅SLSA L2

what more can you do?

Reduce Incident Response Times

Application Inventory & Change Management

Much, much more

- Not just containers!
- Deploy and run static and dynamic analysis tools
- Custom plugins for metadata surfacing!

Try it out!

- Open-Source @ https://github.com/crashappsec/chalk
- Written in Nim!
- Welcoming feature requests!

Backup Slides

A deeper look into chalk

- Rich set of out-of-the box available metadata options
- Metadata can be grouped and published independently:
  - Reports aggregate metadata of interest
  - Publish them to different destinations (sinks)
- Plugin architecture
- Configuration language
  (e.g., set rule to only set a report if within a docker container)
- Effortless config loading


```
./chalk load https://chalkdust.io/basic_compliance.c4m
```