Lorena Ronquillo – KEA Københavns Erhvervsakademi

Security of Software Supply Chain

Who am I?

Lorena Ronquillo

- *Education*
  - MSc, PhD in Computer Science from Autonomous University of Barcelona (UAB), Spain.
  - Postdoc in cryptography, IT University of Copenhagen (ITU), Denmark.

- *Work experience*
  - Software developer at various international companies.
  - Former cryptography and security researcher in e-voting.
  - IT Security consultant.
  - Working at Copenhagen School of Design and Technology (KEA) since 2016.

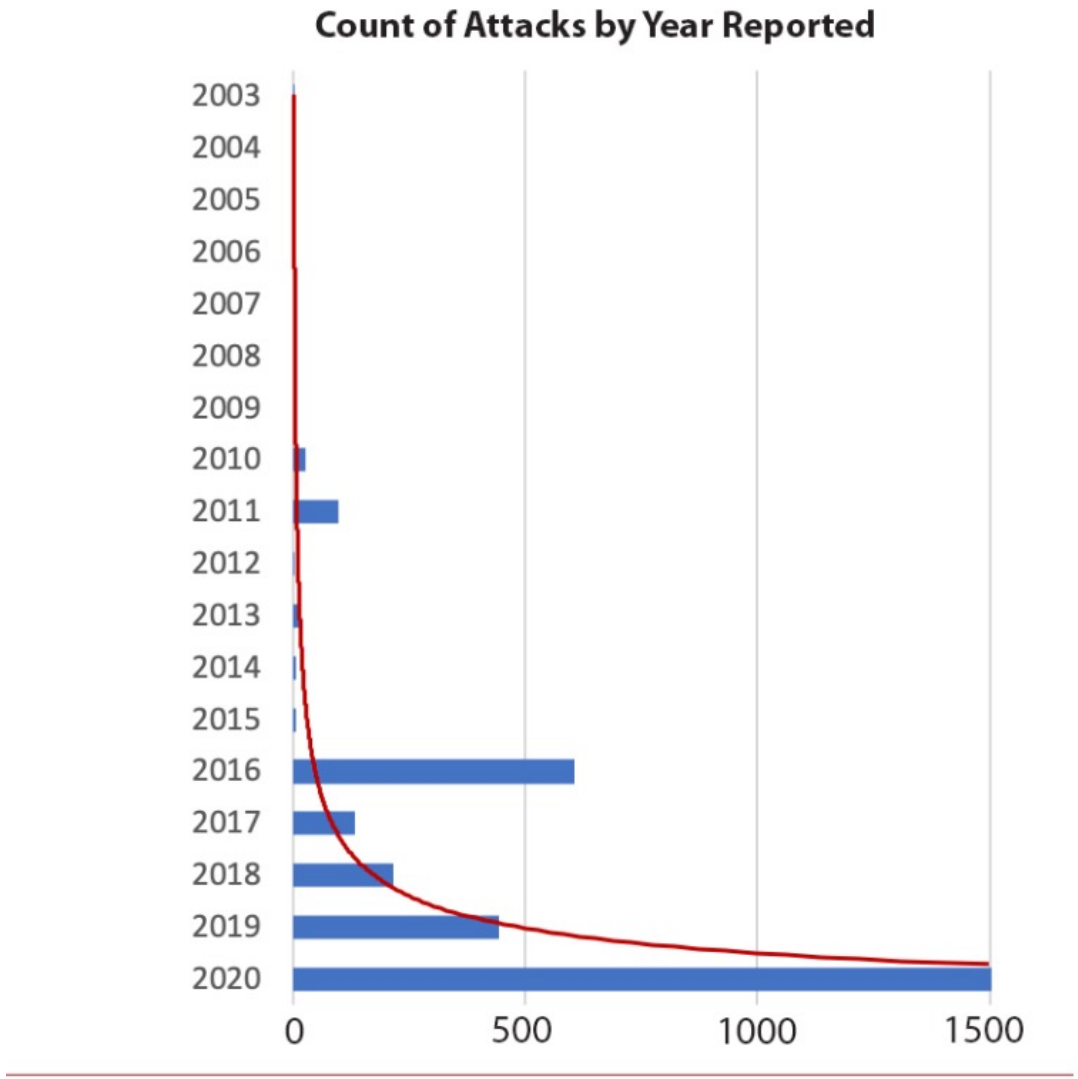
- *Interests*
  - Application security, DevSecOps, cryptography.

Agenda

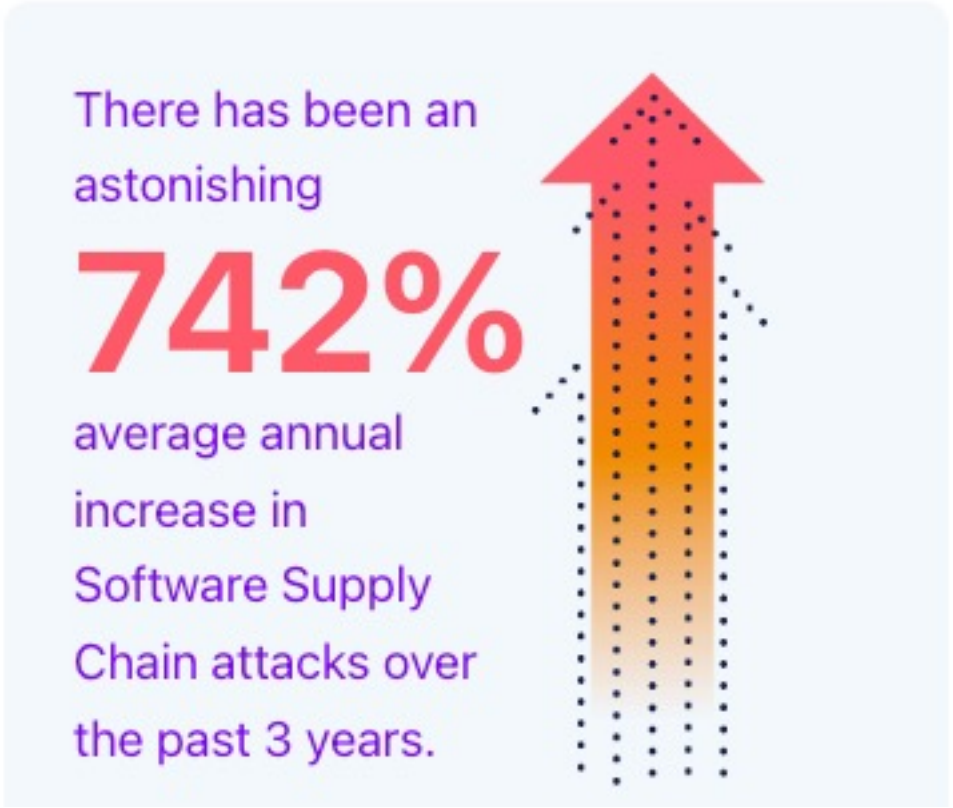
1. What is Software Supply Chain?

2. Software Supply Chain Attacks

3. Preventing Software Supply Chain Attacks



Source: D. Geer, B. Tozer, J.S. Meyers. For Good Measure: counting broken links, a quant's view of software supply chain security. ;login: Winter 2020, Vol. 45, No. 4.



Source: 2022 State of Software Supply Chain report by Sonatype

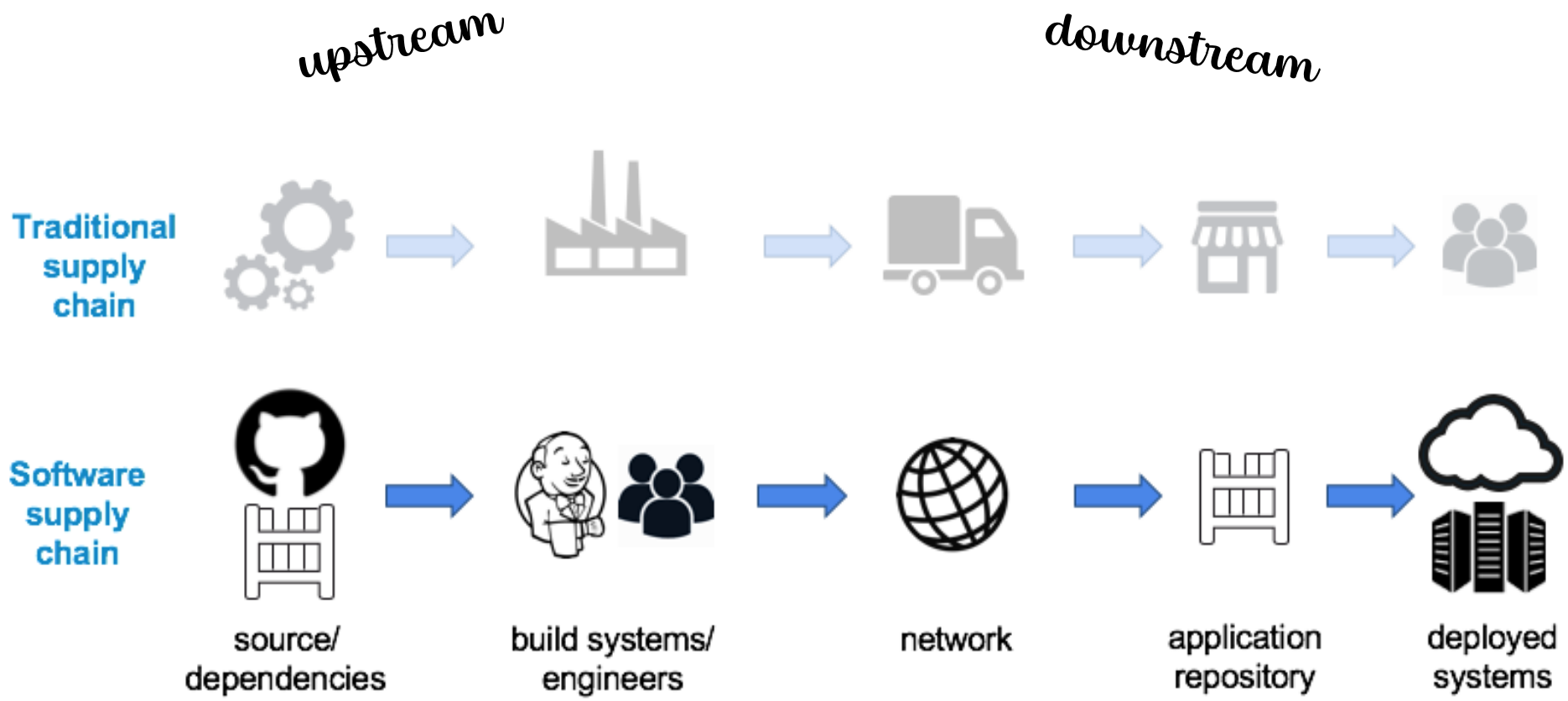
4x increase in the number of organized software supply chain attacks since early 2020.

Source:
Threat Landscape for Supply Chain Attacks
report by the European Union Agency for Cybersecurity (ENISA)

45% of all organizations will experience attacks on their software supply chains by 2025.

Source:
How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks
report by Gartner

Supply chain



Source: https://blog.convisoappsec.com/en/is-your-software-supply-chain-secure/

Agenda

1. What is Software Supply Chain?

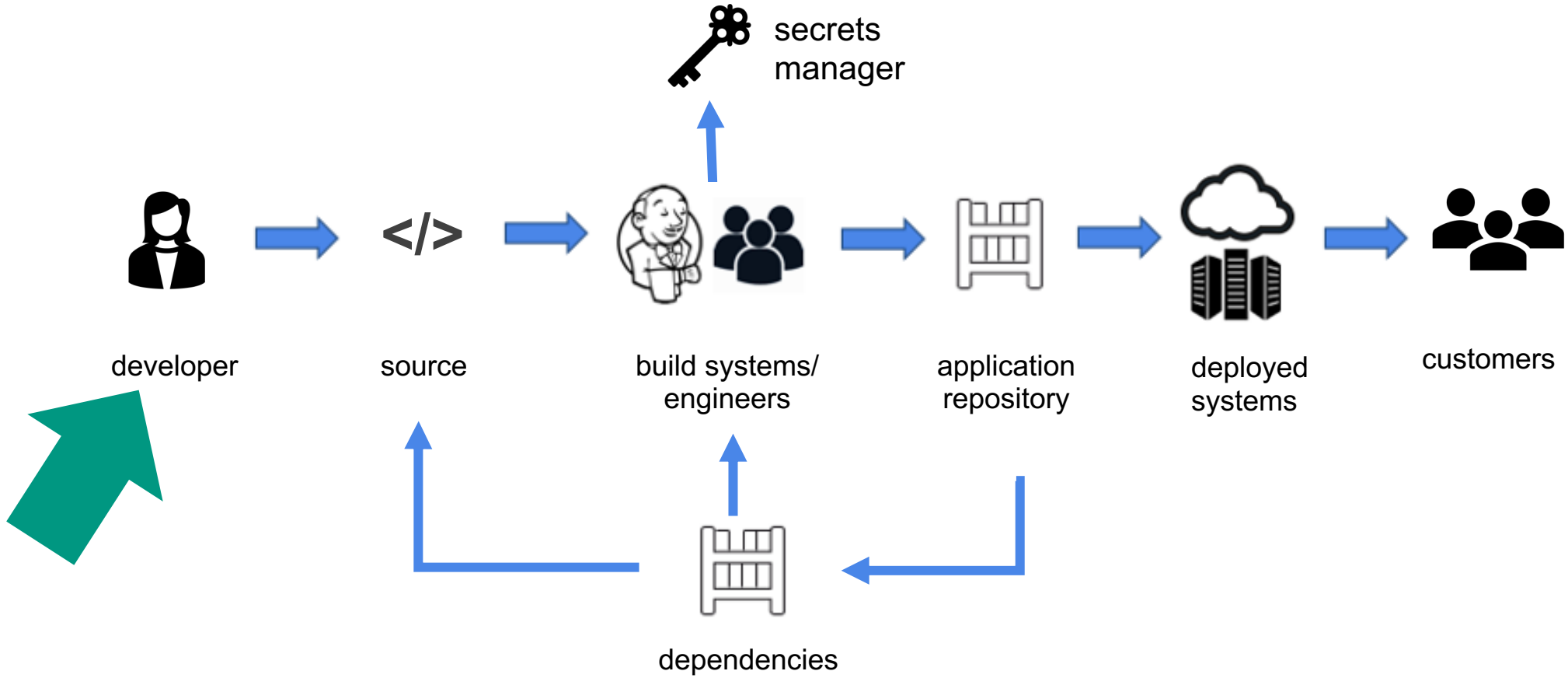
2. Software Supply Chain Attacks

3. Preventing Software Supply Chain Attacks

> (…) if you can’t break into your target, you find the technology that they use and break in there instead.

Mikko Hypponen,
chief research at F-Secure

Software supply chain



Octopus scanner malware, May 2020

Security Lab

Bounties Research Advisories Get Involved Events

May 28, 2020

The Octopus Scanner Malware: Attacking the open source supply chain

Alvaro Munoz

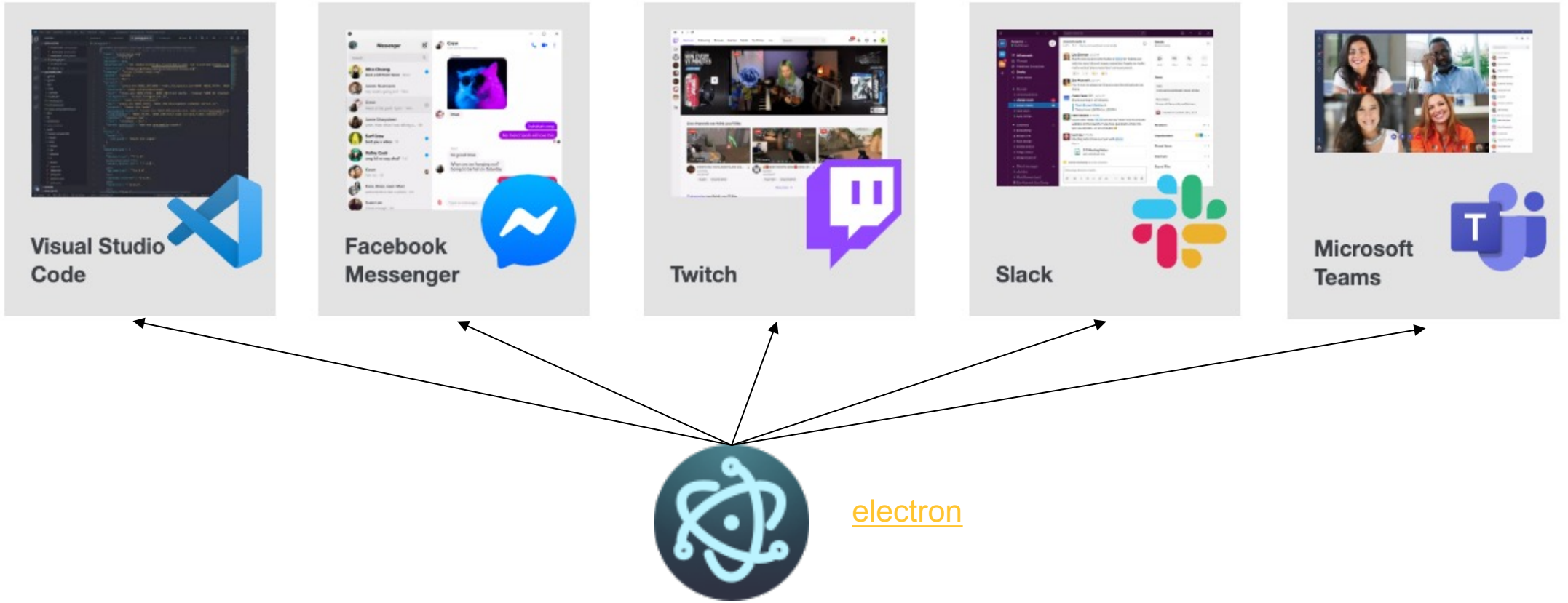
Securing the open source supply chain is an enormous task. It goes far beyond a security assessment or just patching for the latest CVEs. Supply chain security is about the integrity of the entire software development and delivery ecosystem. From the code commits themselves, to how they flow through the CI/CD pipeline, to the actual delivery of releases, there's the potential for loss of integrity and security concerns, throughout the entire lifecycle.

In the past few years the open source supply chain experienced a variety of attacks. From developer credential hijacks aimed at introducing backdoors, like in the event stream incident, to a seemingly nonstop stream of typosquatting attacks against popular package managers such as npm and pypi.

Sometimes something as innocent as a misinterpreted warning can make a developer comment out a single line to dramatic effect. The line between backdoor and typo can often be hard to differentiate, and often the circumstances of the commit, not the commit itself, are the only clear indication of intent

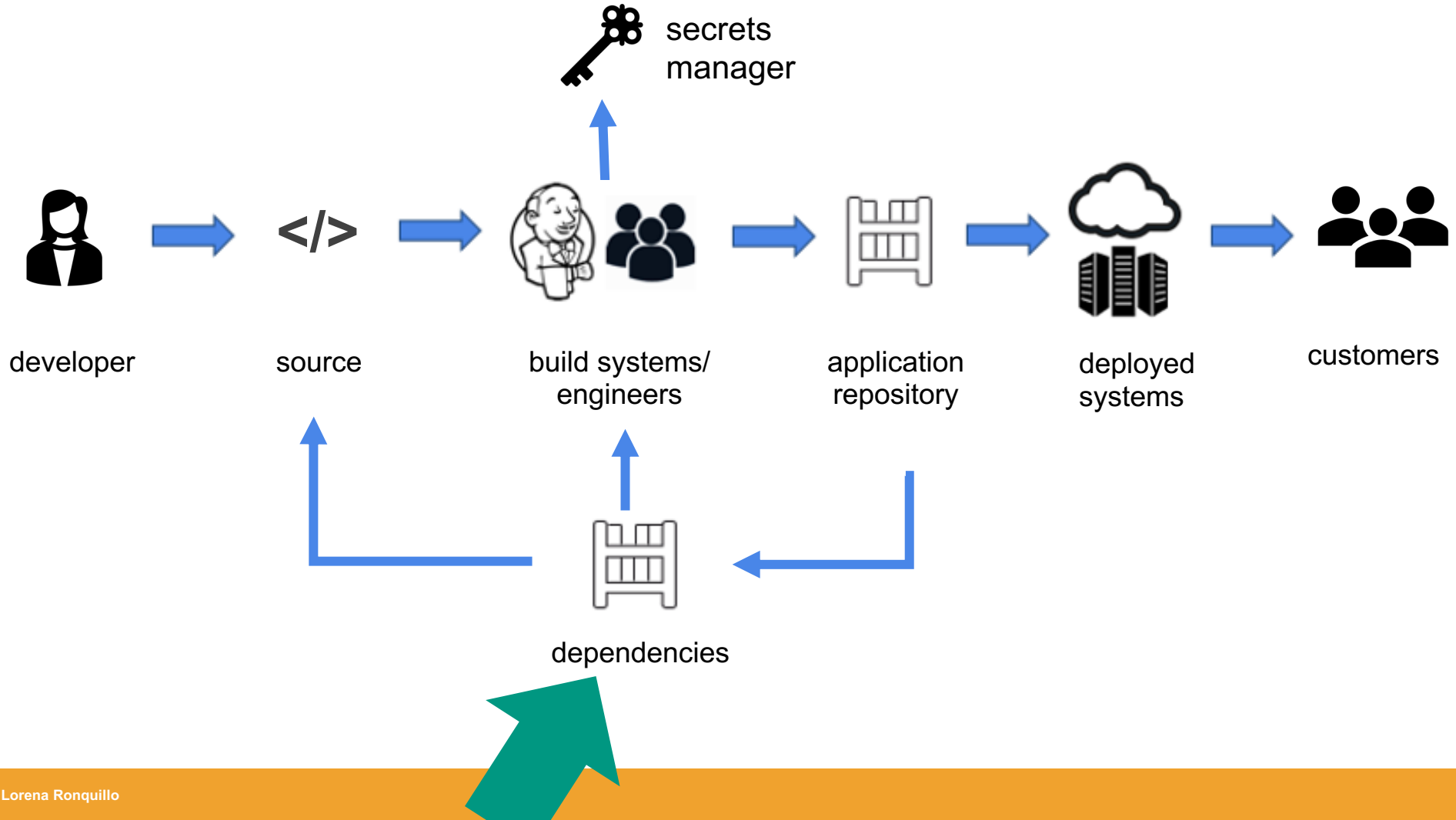
Apache NetBeans

Source: https://securitylab.github.com/research/octopus-scanner-malware-open-source-supply-chain/



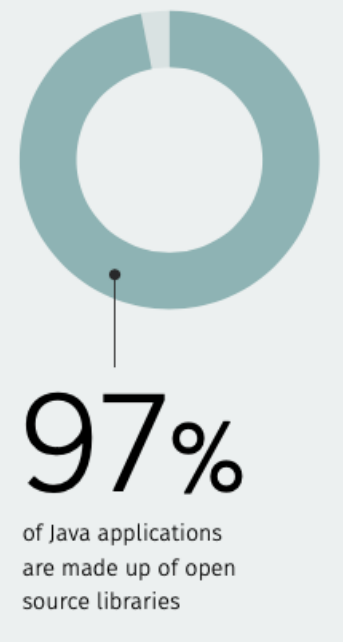
- 71 dependencies
- 1173 contributors
- It only takes one to have malicious intents, and it could have a lot of impact.

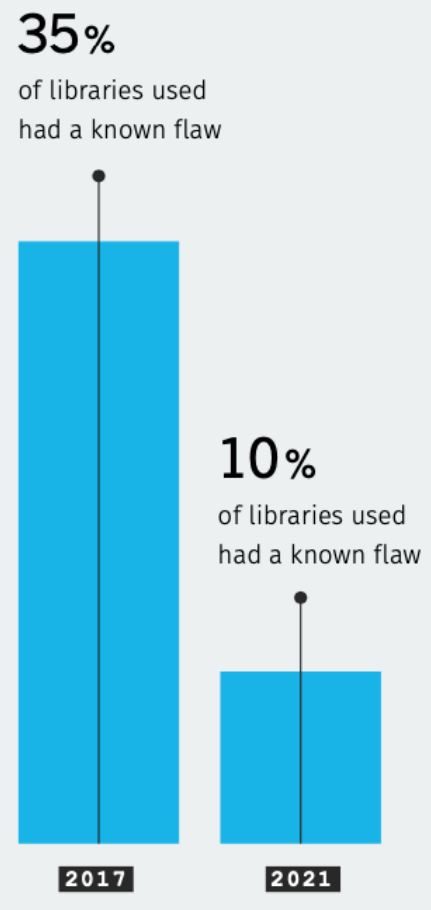
Software supply chain

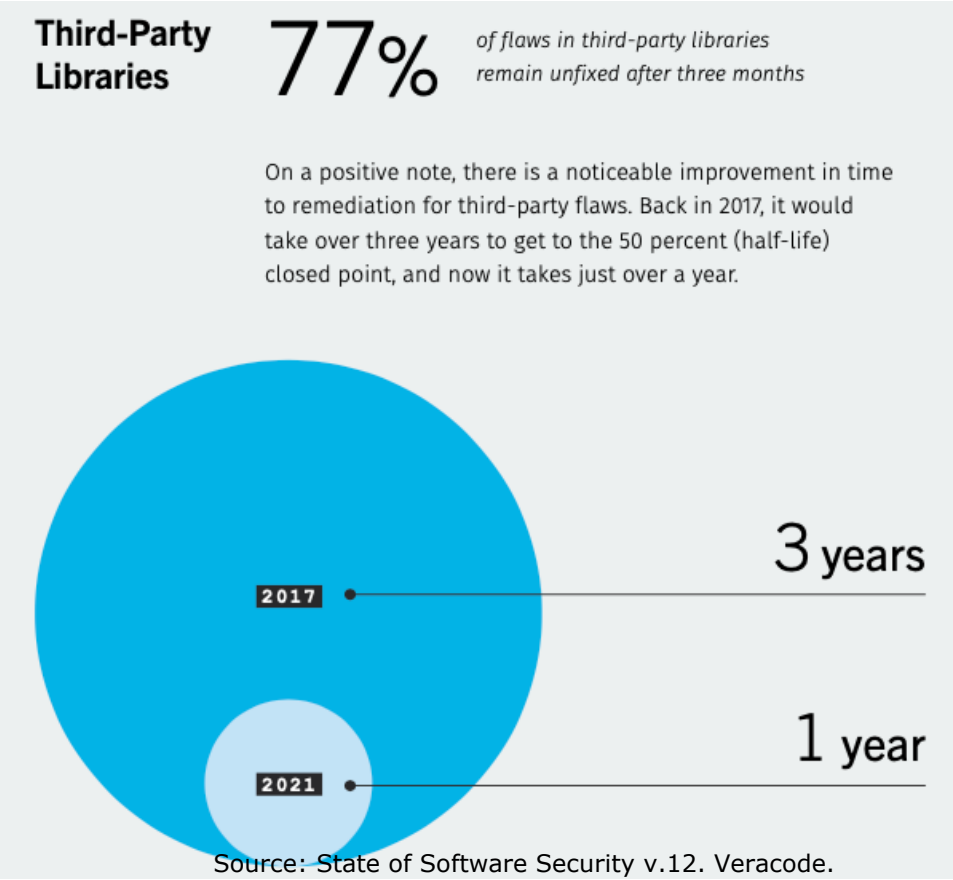


Open-source libraries

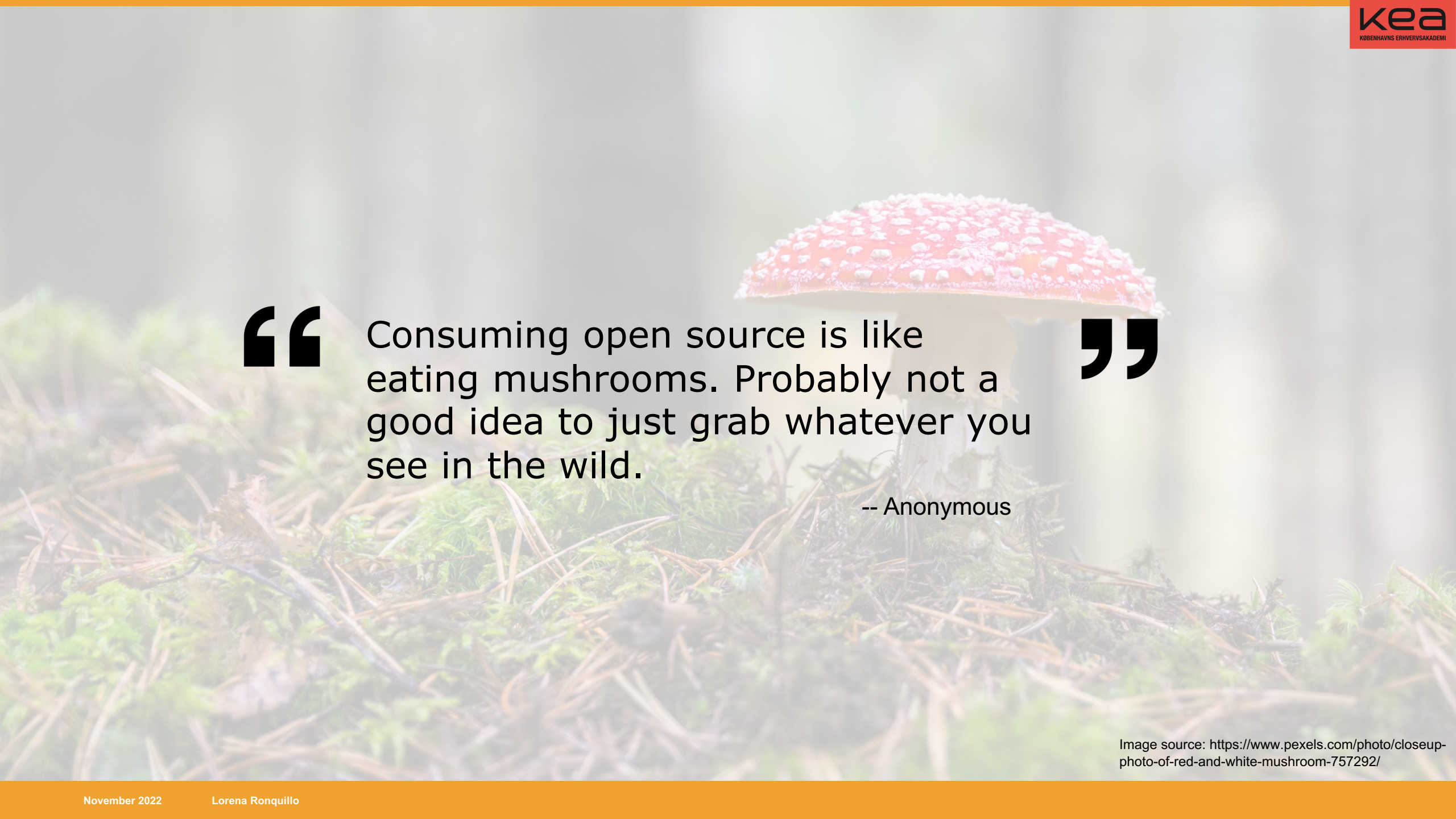
According to the Linux Foundation, between 70-90% of modern software is composed of open-source software.







Source: State of Software Security v.12. Veracode.
https://linuxfoundation.org/wp-content/uploads/LFResearch_Harvard_Census_II.pdf



> Consuming open source is like eating mushrooms. Probably not a good idea to just grab whatever you see in the wild.
>
> -- Anonymous

Image source: https://www.pexels.com/photo/closeup-photo-of-red-and-white-mushroom-757292/

Log4Shell attack, December 2021

Vulnerability in Apache's Log4j logging framework

Critical with a CVSS score of 10 (highest score possible)

Allows attackers to execute code remotely in any vulnerable environment

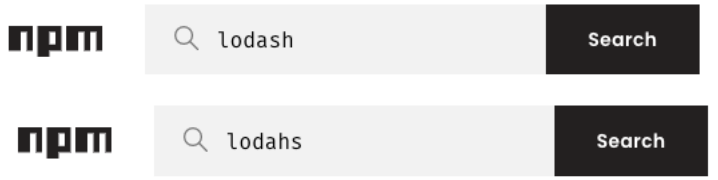
- Dangerous because of the ubiquity of the library: present in major platforms from Amazon Web Services to VMware.
- The ease of exploiting the vulnerability compounds its impact.
  - Attackers could remotely take over any internet-connected service that uses certain versions of the Log4j library anywhere in the software stack.
- Cyber Safety Review Board report on Log4j:
  - Organizations that responded most effectively to the Log4j event were also the ones that understood better their use of Log4j and had technical resources and processes to:
    - Manage assets
    - Assess risk
    - Mobilize their organization and partners to action

Source: https://www.zdnet.com/article/security-warning-new-zero-day-in-the-log4j-java-library-is-already-being-exploited/

https://www.cisa.gov/sites/default/files/publications/CSRB-Report-on-Log4-July-11-2022_508.pdf

Typosquatting attack, November 2019

- electorn, loadyaml, lodashs packages from npm



lodash DT

4.17.21 • Public • Published a year ago

Readme | Explore BETA | 0 Dependencies | 159,366 Dependents | 114 Versions

lodash v4.17.21

The Lodash library exported as Node.js modules.

Installation

Using npm:

```
$ npm i -g npm
$ npm i --save lodash
```

In Node.js:

```
// Load the full build.
var _ = require('lodash');
// Load the core build.
var _ = require('lodash/core');
// Load the FP build for immutable auto-curried iteratee-first data-last me
```

Install

```
> npm i lodash
```

Repository
github.com/lodash/lodash

Homepage
lodash.com/

Weekly Downloads
46,538,360

Version
4.17.21

License
MIT

Unpacked Size
1.41 MB

Total Files
1054

lodahs

0.0.1-security • Public • Published 3 years ago

Readme | Explore BETA | 0 Dependencies | 0 Dependents | 1 Versions

Security holding package

This package name is not currently in use, but was formerly occupied by another package. To avoid malicious use, npm is hanging on to the package name, but loosely, and we'll probably give it to you if you want it.

You may adopt this package by contacting support@npmjs.com and requesting the name.

Keywords

none

Install

```
> npm i lodahs
```

Repository
github.com/npm/security-holder

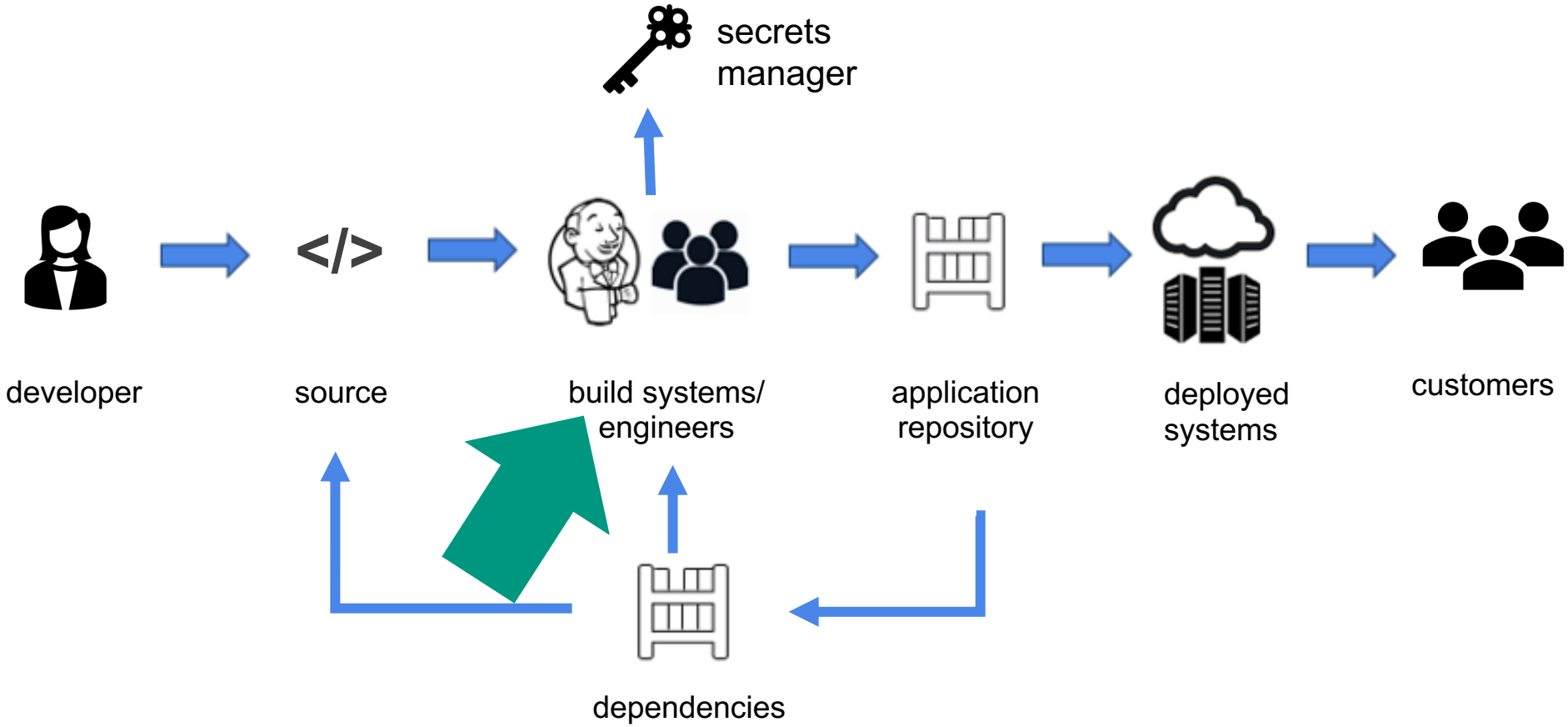
Homepage
github.com/npm/security-holder#readme

Weekly Downloads
10

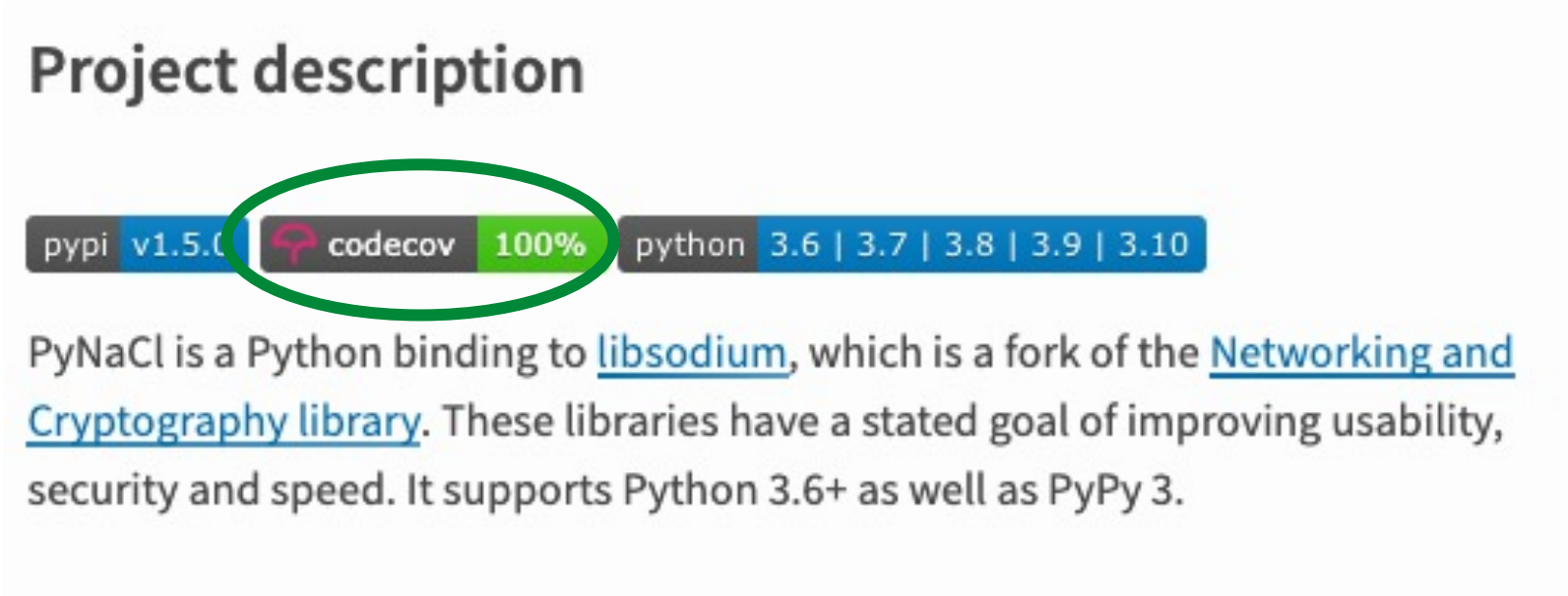
Version
0.0.1-security

License
none

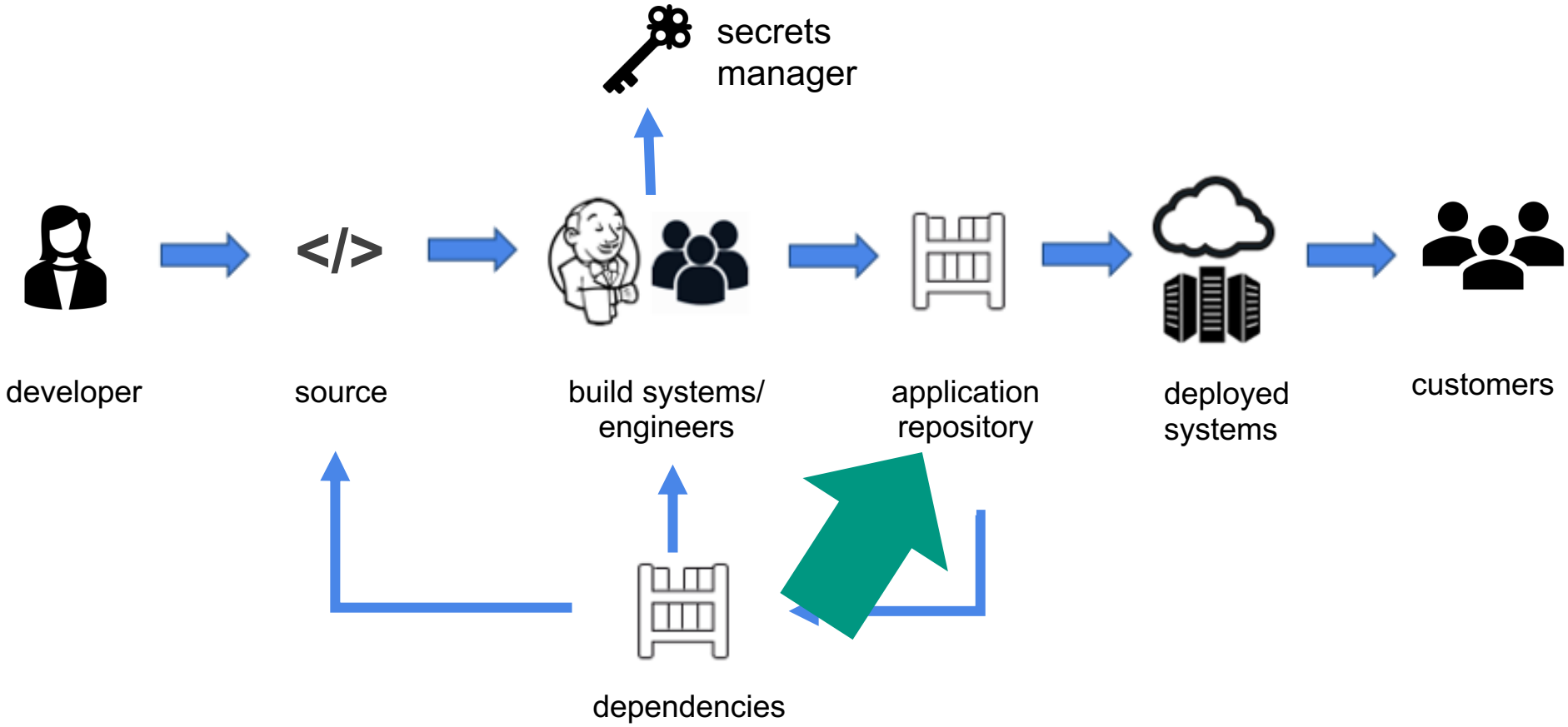
Software supply chain



Codecov attack, April 2021



Software supply chain



SolarWinds, December 2020

SolarWinds Orion was used by tens of thousands of downstream systems and run with privileged access to their networks

- Users included high-profile clients such as: US military, state departments, and some of the biggest corporations.

Threat actors gained access to SolarWinds' development infrastructure and injected malicious code into Orion update binaries.

18.000 customers automatically pulled these updates, planting backdoors into their systems, which hackers used to install even more malware that helped them spy on companies and organizations.

- Bad actors could exploit private networks at will.

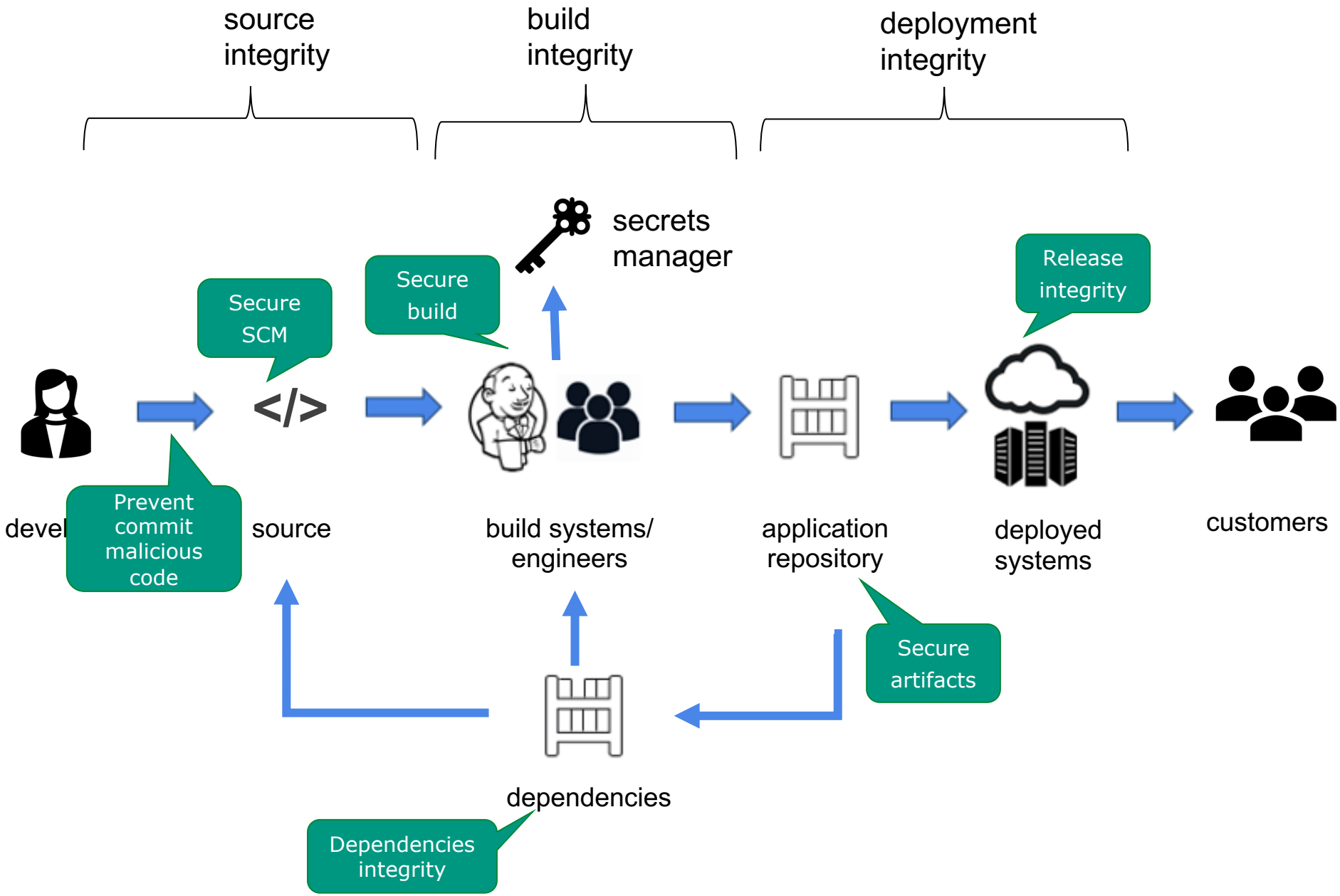
Source: 2021 State of Software Supply Chain report – by Sonatype

Agenda

1. What is Software Supply Chain?

2. Software Supply Chain Attacks

3. Preventing Software Supply Chain Attacks


source integrity
build integrity
deployment integrity
secrets manager
Secure SCM
Secure build
Release integrity
</>
Prevent commit malicious code
devel
source
build systems/ engineers
application repository
deployed systems
customers
Secure artifacts
dependencies
Dependencies integrity

Software supply chain security guides

A lot of literature on secure development, from secure coding to securing the global lifecycle.

- supply chain security is mostly a blind spot for many existing frameworks.

Best practice guides:

- NIST's Software Supply Chain Security Guidance, May 2021
- The Open Source Software Security Mobilization Plan from The Linux Foundation
- Center for Internet Security (CIS)'s Software Supply Chain Security Guide

July 2022

CIS' Software Supply Chain Security guide

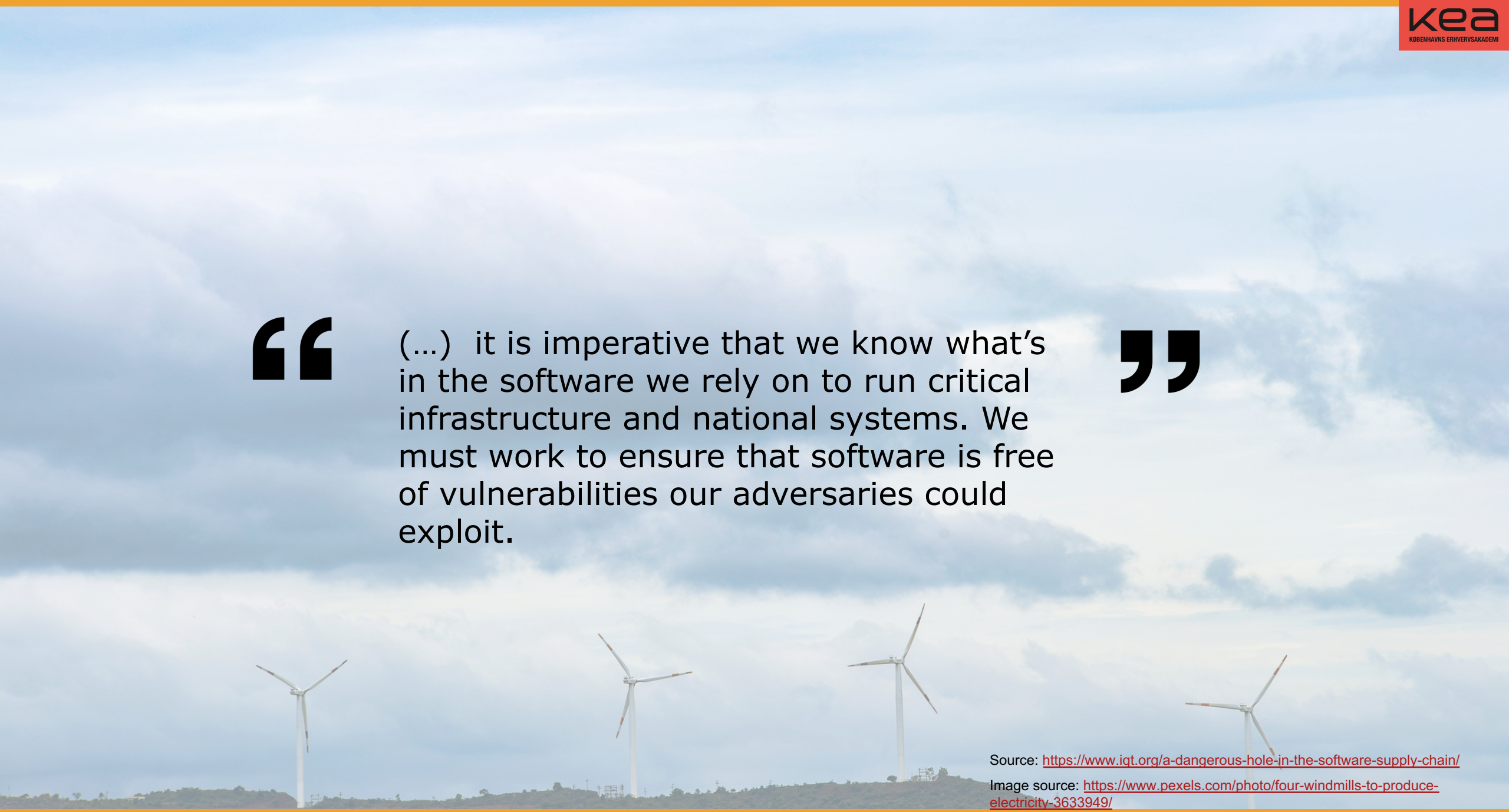
More than 100 recommendations across five core areas:

- Source code
- Build pipelines
- Dependencies
- Artifacts
- Deployment

Some terms from the guide:

- Container hardening
- Transparency
- SCA
- SBOM
- Cryptographic verification of artifacts

…



> (…) it is imperative that we know what's in the software we rely on to run critical infrastructure and national systems. We must work to ensure that software is free of vulnerabilities our adversaries could exploit.

Source: https://www.iqt.org/a-dangerous-hole-in-the-software-supply-chain/

Image source: https://www.pexels.com/photo/four-windmills-to-produce-electricity-3633949/

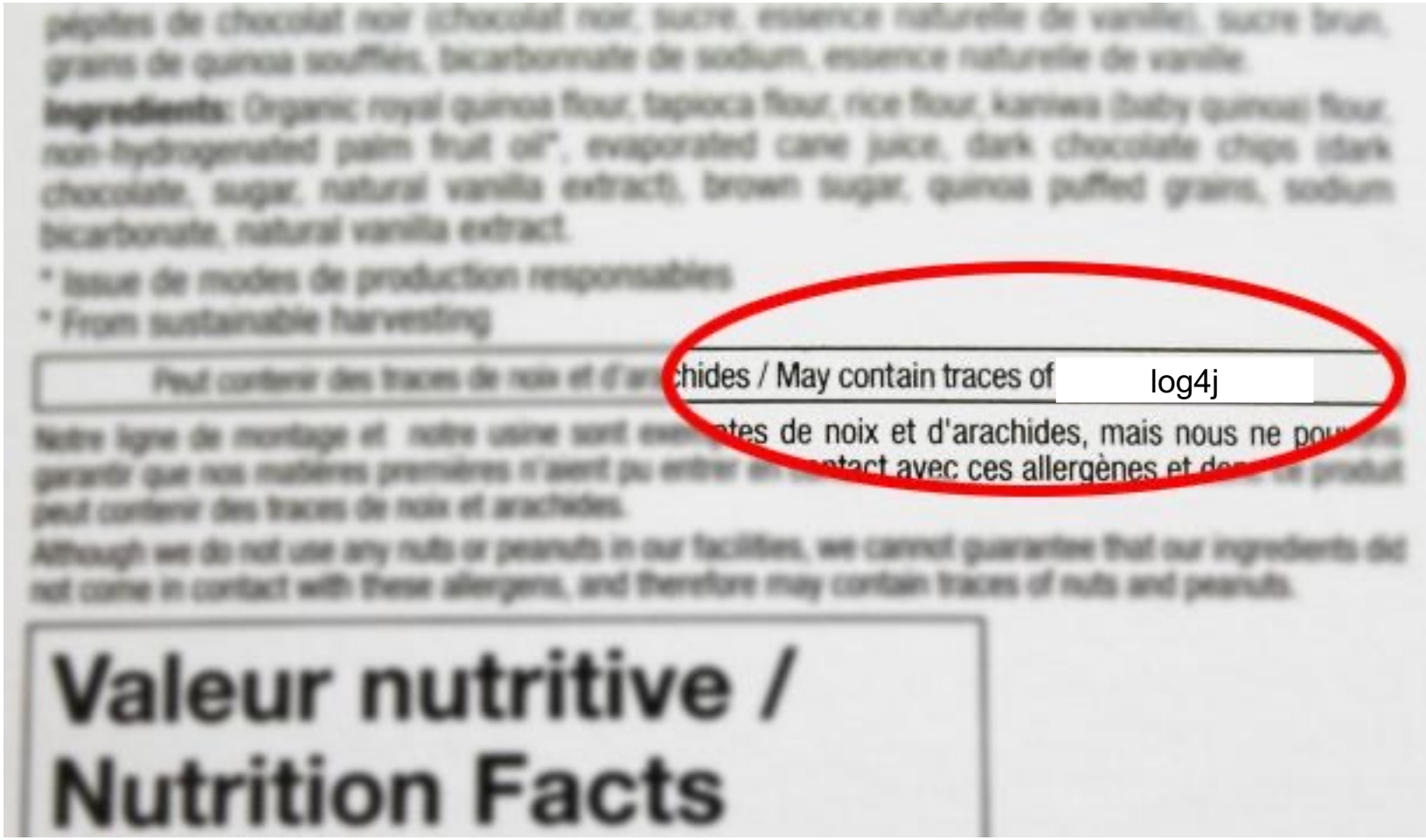
Software Bill-of-Materials (SBOM)

Like food labels.

A list that includes but is not limited to commercial, open source, and off-the-shelf software and hardware components that are or could become susceptible to vulnerabilities

Only 45% of companies currently use an SBOM as part of their development and application security practices.

Federal mandate in the US since July 2021.



SBOM

An SBOM should include a set of claims including but not limited to:

- Who built it?
- Who provided it?
- What is in a given software package?
- When was it created?
- Where was it created?
- What are its dependencies and its dependencies' dependencies?

It often doesn't include:

- How?

Source: https://www.mend.io/sbom/

SBOM formats

Leading SBOM standards (none of them is de facto industry standard yet):

- **Software Package Data Exchange** (SPDX) led by The Linux Foundation. Has met all requirements for standardization and quality assurance as defined by ISO. **Most adopted one by major corporations such as Intel, Microsoft, Siemens, and Sony.**
- Software Identification Tagging (SWID), not for security use cases. Focused on licensing.
- **CycloneDX** led by OWASP.

SPDX and CycloneDX are likely to stay for some time and both software producers and consumers would be best positioned if they can support both formats.

Source:
https://www.csoonline.com/article/3668530/sbom-formats-spdx-and-cyclonedx-compared.html

Tools to create SBOMs

Anchore: built upon two open-source projects, **Syft**, a command-line interface tool and library for generating SBOMs from container images and filesystems, and **Grype**, a vulnerability scanning tool for container images and filesystems.

FOSSA: scans projects and automatically identifies both the direct and deep dependencies.

Mend: SBOM tool + SCA tool. Can be used to track components, including direct and transitive dependencies, identify vulnerabilities, provide a remediation path, and update SBOM records automatically as components change.

Rezilion: uses dynamic runtime analysis to track your software attack surface as your code changes. It constantly looks up weaknesses for your code's components.

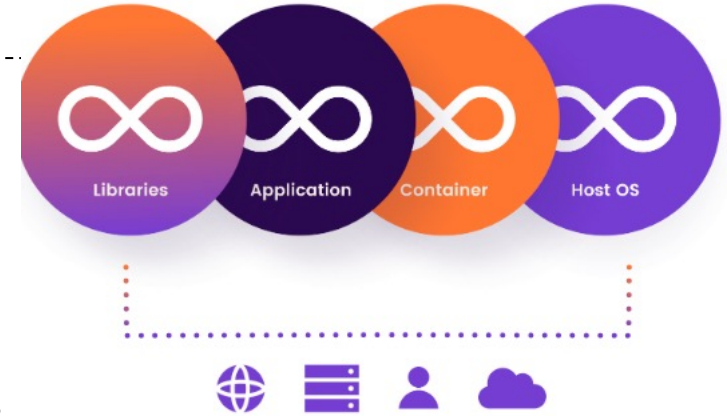
SPDX SBOM Generator: standalone open-source tool from The Linux Foundation. It creates SBOMs from your current package managers or build systems.

Tern: open-source SBOM project. Instead of working with package managers or build systems, it generates an SBOM for container images and Dockerfiles.

TauruSeer: offered as a software-as-a-service (SaaS).

Vigilant Ops: its SaaS platform generates, maintains and authenticates the sharing of certified SBOMs.

Microsofts' SBOM tool: open-source, fairly new one. It can reference to other SBOM documents for capturing a full dependency tree.



Source:
https://www.rezilion.com/platform/dynamic-sbom

https://www.csoonline.com/article/3667483/8-top-sbom-tools-to-consider.html

DEMO: Syft SBOM generator from container images

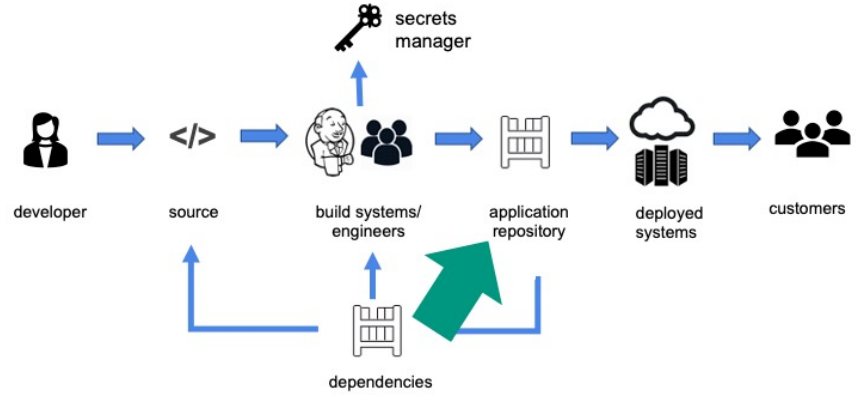
https://github.com/anchore/syft/

But before that…

What are containers?

- Allow to run software in an isolated environment
- A container has everything that the application needs for running and be deployed

- Advantages:
  - Makes configuration and deployment easier
  - Consistency of environments (development, production, testing..)
  - Scalability (by firing up more containers). Needs orchestration (Kubernetes, Docker Swarm..)



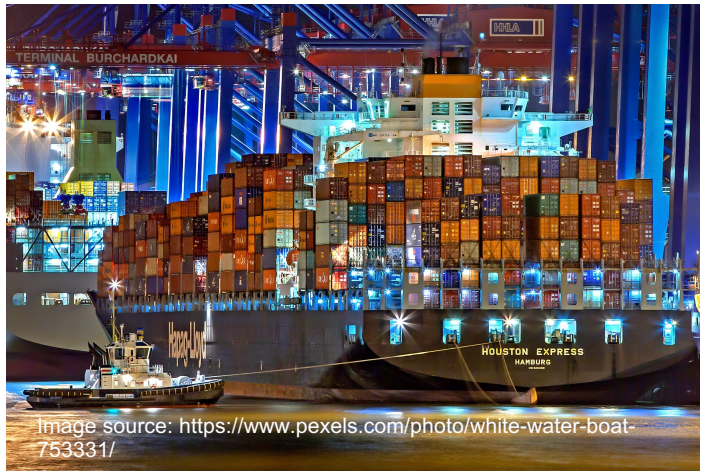


Image source: https://www.pexels.com/photo/white-water-boat-753331/

DEMO: Syft SBOM generator from container images

https://github.com/anchore/syft/

DEMO: Grype vulnerability scanning tool for container images

https://github.com/anchore/grype/

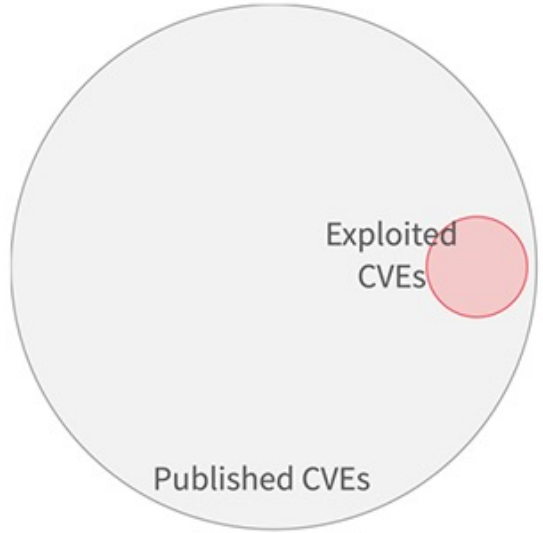
SBOM challenges

- Lack of clarity about SBOM generation, discovery and distribution
  - Open source project: security.txt
  - Real-time SBOM publication

- CVE noise in large environments limits the effectiveness.
  - Too many vulnerabilities!
    - Organisations are able to fix between 5% and 20% of known vulnerabilities per month.

- Vulnerability vs exploitability
  - Only 2%-7% of vulnerabilities are exploited in the wild.

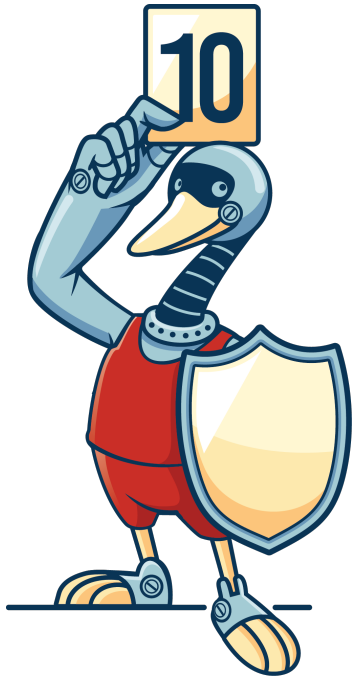
- VEX (Vulnerability Exploitability Exchange) : am I affected?
  - An opportunity to reduce false positives.
  - No tools yet generating VEX documents



OpenSSF's Scorecards project

Evaluates security health metrics for OSS projects

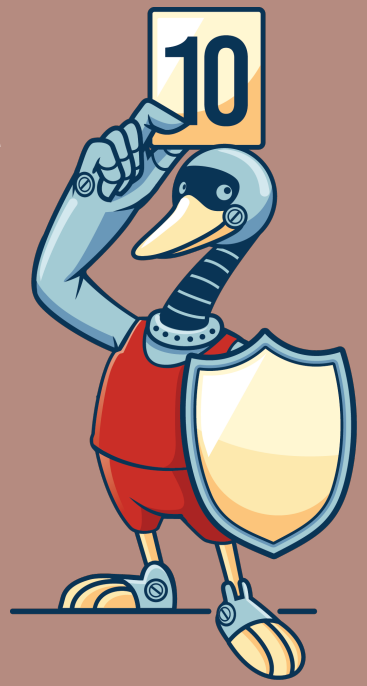
Uses a scoring scale of critical, high, medium and low.



Source: https://github.com/ossf/scorecard

Name	Description	Risk Level
Binary-Artifacts	Is the project free of checked-in binaries?	High
Branch-Protection	Does the project use Branch Protection ?	High
CI-Tests	Does the project run tests in CI, e.g. GitHub Actions, Prow?	Low
CII-Best-Practices	Does the project have a CII Best Practices Badge?	Low
Code-Review	Does the project require code review before code is merged?	High
Contributors	Does the project have contributors from at least two different organizations?	Low
Dangerous-Workflow	Does the project avoid dangerous coding patterns in GitHub Action workflows?	Critical
Dependency-Update-Tool	Does the project use tools to help update its dependencies?	High
Fuzzing	Does the project use fuzzing tools, e.g. OSS-Fuzz?	Medium
License	Does the project declare a license?	Low
Maintained	Is the project maintained?	High
Pinned-Dependencies	Does the project declare and pin dependencies?	Medium
Packaging	Does the project build and publish official packages from CI/CD, e.g. GitHub Publishing ?	Medium
SAST	Does the project use static code analysis tools, e.g. CodeQL, LGTM, SonarCloud?	Medium
Security-Policy	Does the project contain a security policy?	Medium
Signed-Releases	Does the project cryptographically sign releases?	High
Token-Permissions	Does the project declare GitHub workflow tokens as read only?	High
Vulnerabilities	Does the project have unfixed vulnerabilities? Uses the OSV service.	High

DEMO: Scorecards project



https://github.com/ossf/scorecard

Supply Chain Levels of Software Artifacts (SLSA)

Launched by the OpenSSF

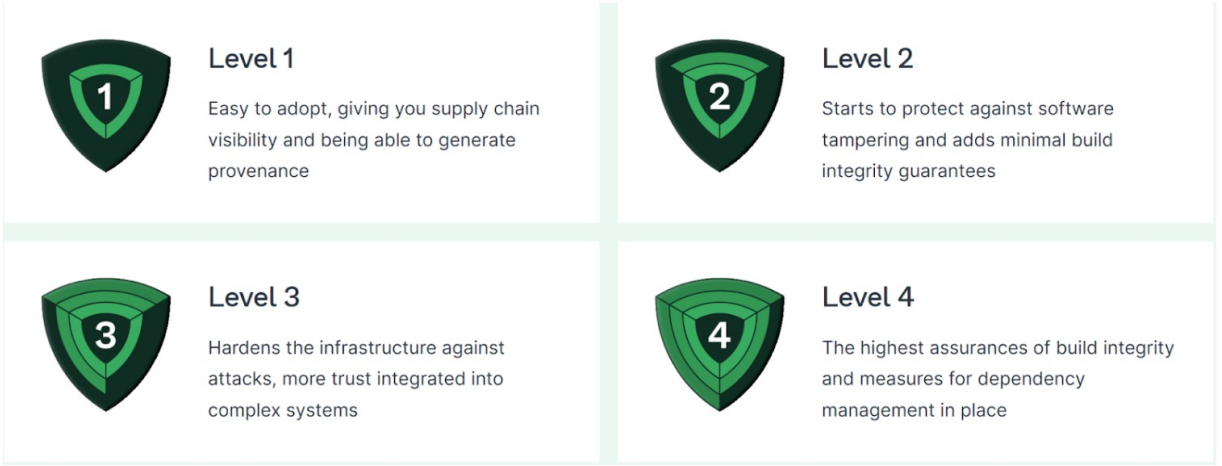
Check-list of standards and controls to prevent tampering, improve integrity, and secure packages and infrastructure in your projects.

- Not a list of best practices to follow.

Provides a common language for increasing levels of software security and supply chain integrity.

Can be thought of **supply chain security capability model**.

- To see how a larger project achieved SLSA 1, check out the path the Kubernetes project followed here.



Source: https://slsa.dev/

Requirement	SLSA 1	SLSA 2	SLSA 3	SLSA 4
Source - Version controlled		✓	✓	✓
Source - Verified history			✓	✓
Source - Retained indefinitely			18 mo.	✓
Source - Two-person reviewed				✓
Build - Scripted build	✓	✓	✓	✓
Build - Build service		✓	✓	✓
Build - Build as code			✓	✓
Build - Ephemeral environment			✓	✓
Build - Isolated			✓	✓
Build - Parameterless				✓
Build - Hermetic				✓
Build - Reproducible				○
Provenance - Available	✓	✓	✓	✓
Provenance - Authenticated		✓	✓	✓

Source: https://slsa.dev/spec/v0.1/requirements

Summarizing

- Software supply chain is complex and fragile
  - Presents risk to countless organizations, consumers, suppliers and industries.
- Securing it is a challenge
  - Attacks to software supply chain have drastically increased in the last few years.
  - Mainly a consumer problem.
- Different ways of fighting this:
  - Best practices (CIS' guide and others)
  - Emerging frameworks and corresponding tools (SBOM, VEX, Sigstore, SLSA)
  - Open-source project initiatives (OpenSSF's Scorecards)
- Many other aspects left, worth considering: container security, choice of secrets manager, and other ways of hardening the software manufacturing process.

To learn more about software supply chain security

- A History of Software Supply Chain Attacks by Sonatype
- Catalog of supply chain compromises by the Cloud Native Computing Foundation (CNCF)
- ATT&CK-like matrix on CI/CD pipeline-specific risks
- CI/CD Goat: Deliberately vulnerable CI/CD environment to get familiar with the characteristics of this new attack surface.
- Threat Landscape for Supply Chain Attacks report by the European Union Agency for Cybersecurity (ENISA)
- How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks report by Gartner


Thanks!
Questions?