

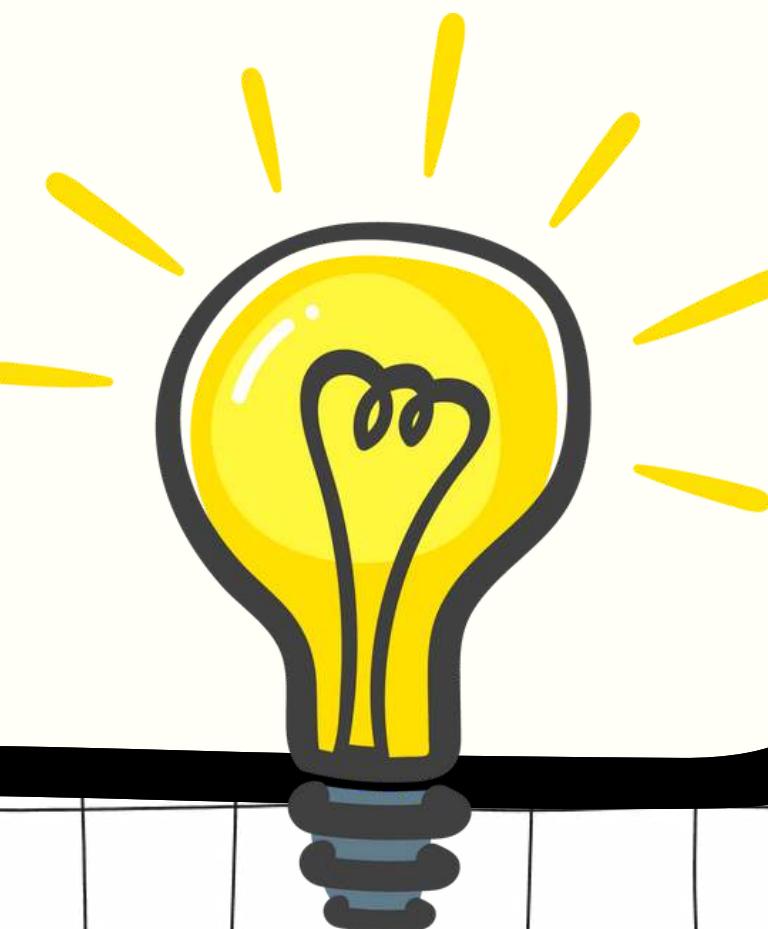


# Enhancing Open Source Security

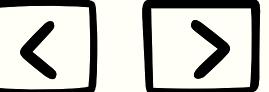
with Secure Supply Chain  
Consumption Framework (S2C2F)

# Enhancing Open Source Security

with Secure Supply Chain Consumption  
Framework (S2C2F)



# Contents



01

Problem Space

02

Solution

03

Introduction

04

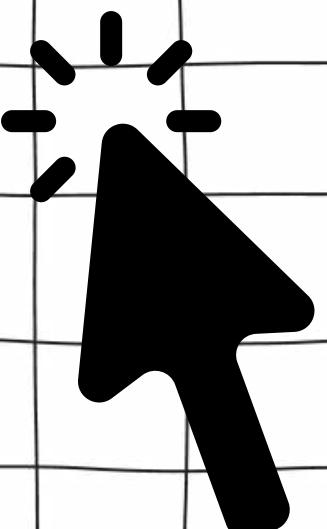
Overview

05

Key Practices

06

Examples



# Speakers



A speaker card template featuring a circular portrait placeholder at the top, three dots for a menu in the top right corner, and a white background area below for text. The card is enclosed in a rounded rectangle border.

**Kaif Ahsan**  
*Atlassian*



A speaker card template featuring a circular portrait placeholder at the top, three dots for a menu in the top right corner, and a white background area below for text. The card is enclosed in a rounded rectangle border.

**Kumar Soorya**  
*AWS*

# **Disclaimer #1**

The content is our personal  
research and does not represent  
the employers.

# **Disclaimer #2**

Operating under tight time  
conditions so certain elements  
will be high level.

They serve as references so  
attendees can dive deeper later.

**Let's understand  
the problem.**



# Rise of Supply Chain Threats

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

The Story of a Novel Supply Chain Attack



Alex Birsan · [Follow](#)

11 min read · Feb 10, 2021

19K

52

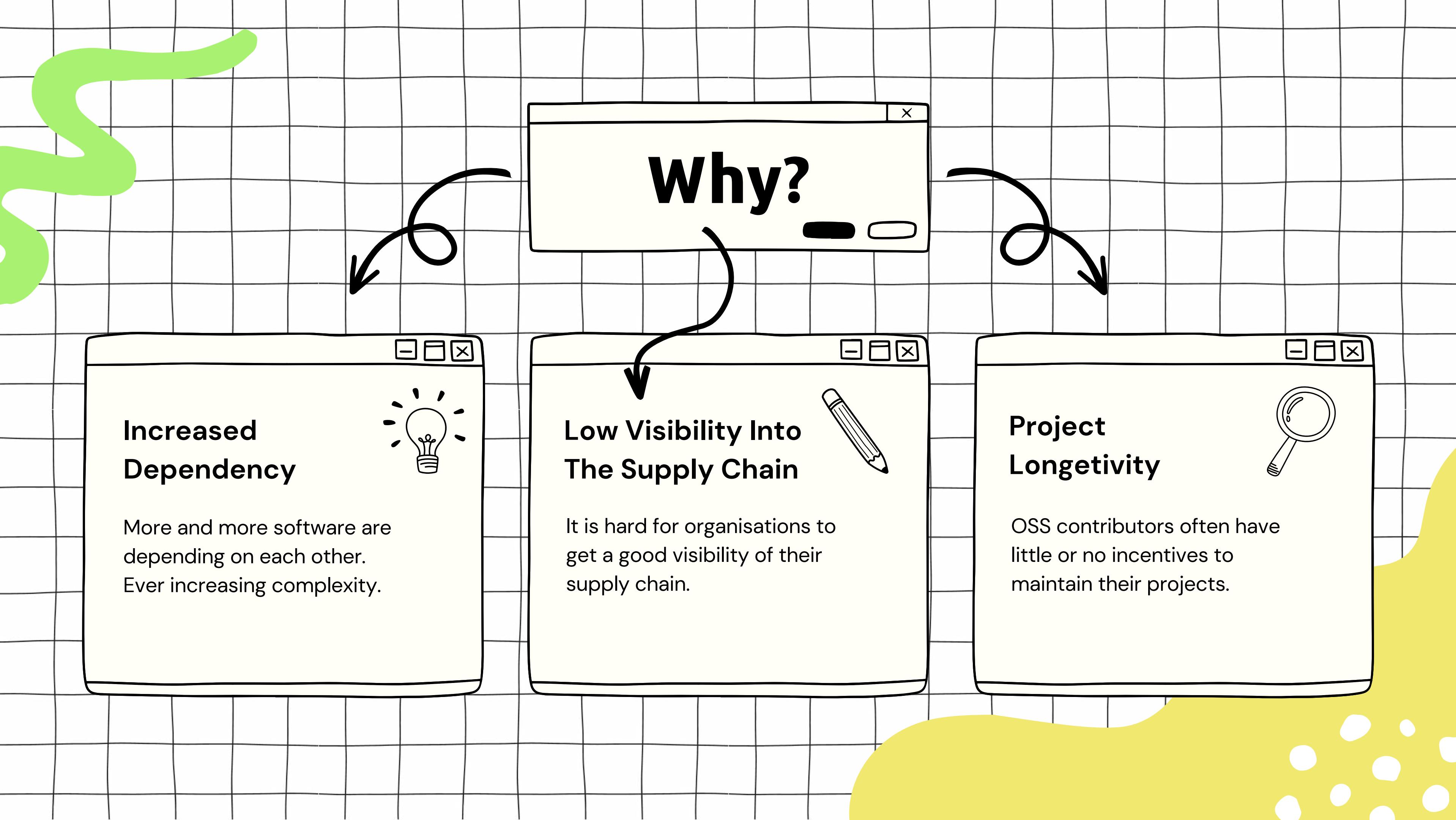


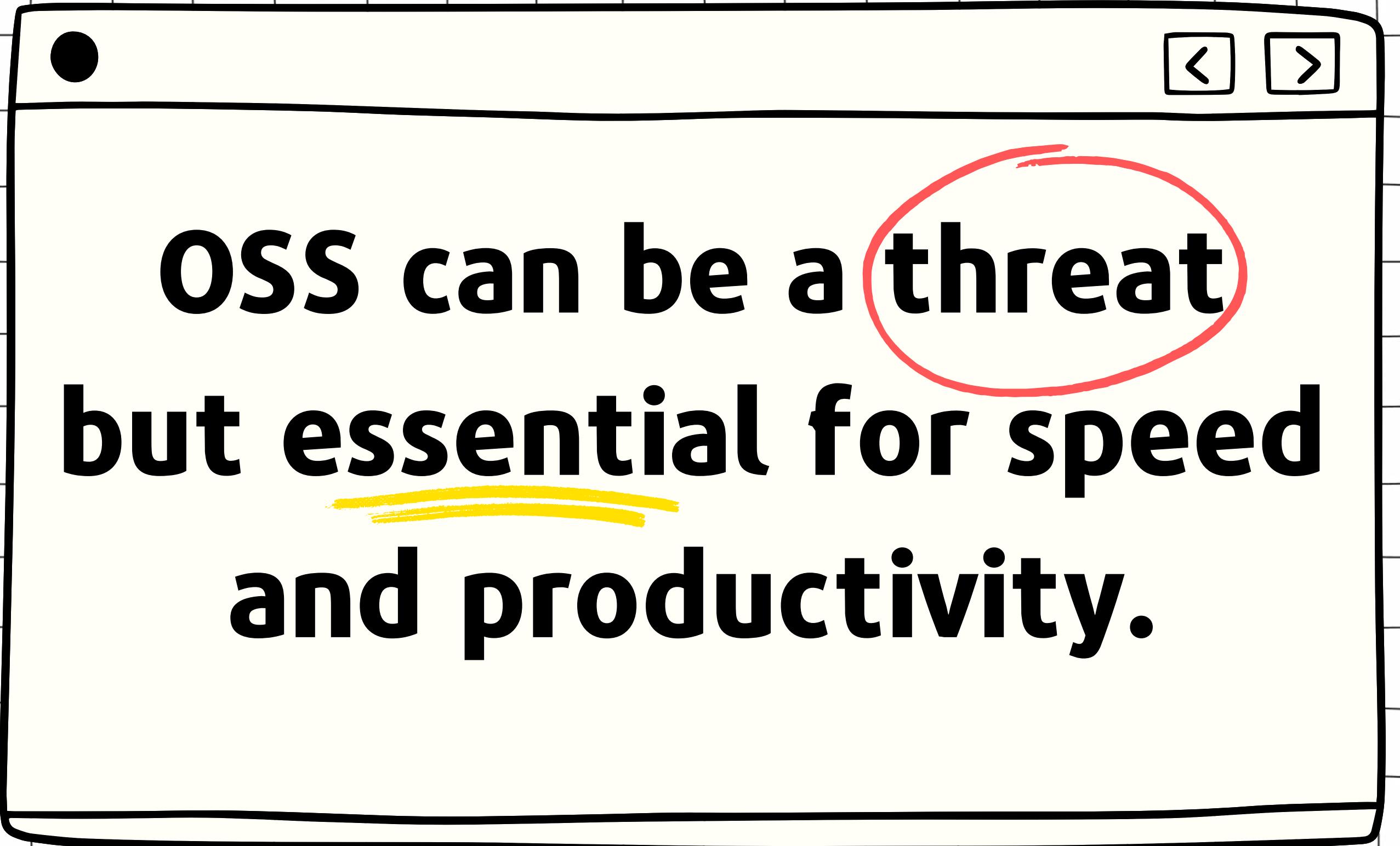
LILY HAY NEWMAN SECURITY DEC 10, 2021 2:54 PM

## 'The Internet Is on Fire'

A vulnerability in the Log4j logging framework has security teams scrambling to put in a fix.







OSS can be a threat  
but essential for speed  
and productivity.

# **Systematic Approach To Securely Consume OSS**

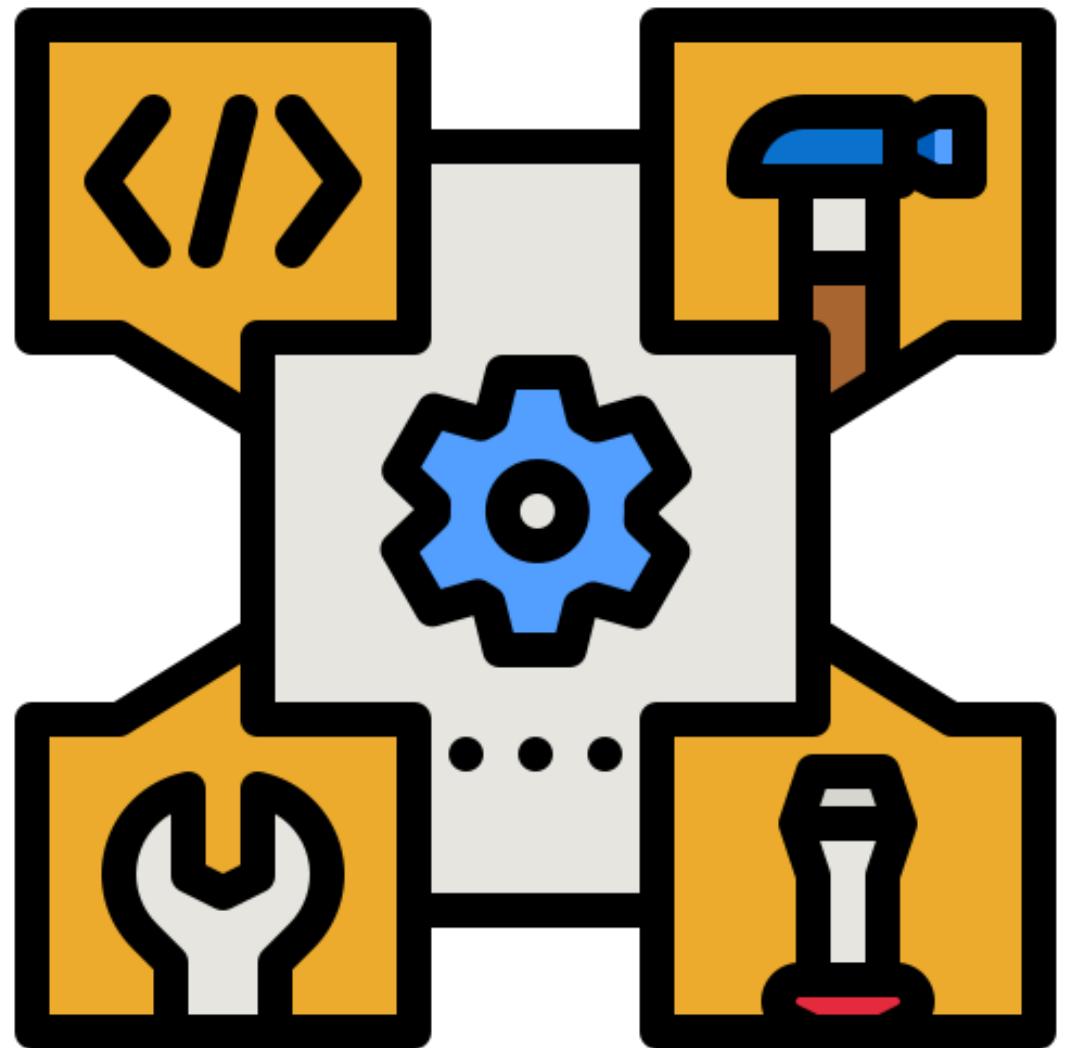


# **Secure Supply Chain Consumption Framework (S2C2F)**

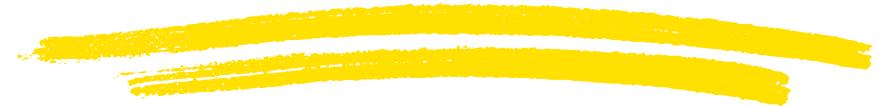
# **What Is S2C2F?**

---

- A framework built by Microsoft and adopted by OpenSSF.
- Guides how to consume OSS dependencies securely.
- Is a collection of practices and controls.
- Provides a roadmap for gradual maturity.

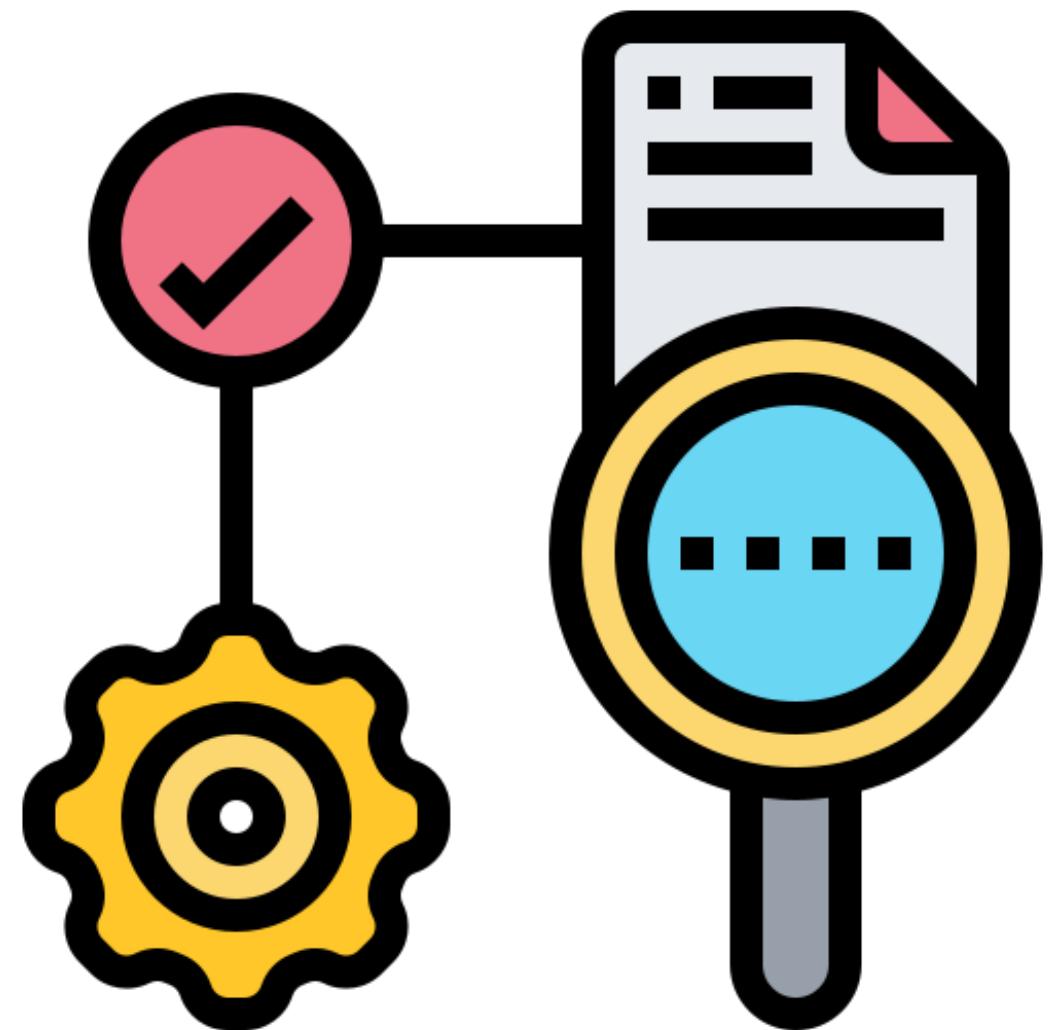


# why S2C2F?



# Why S2C2F?

- End-to-end process.
- Technology agnostic.
- Provides teams with a roadmap to follow.
- Lists a wide range of controls with implementation suggestions.



# 8 Key Practices



Ingest



Scan



Inventory



Update



Audit



Enforce



Rebuild



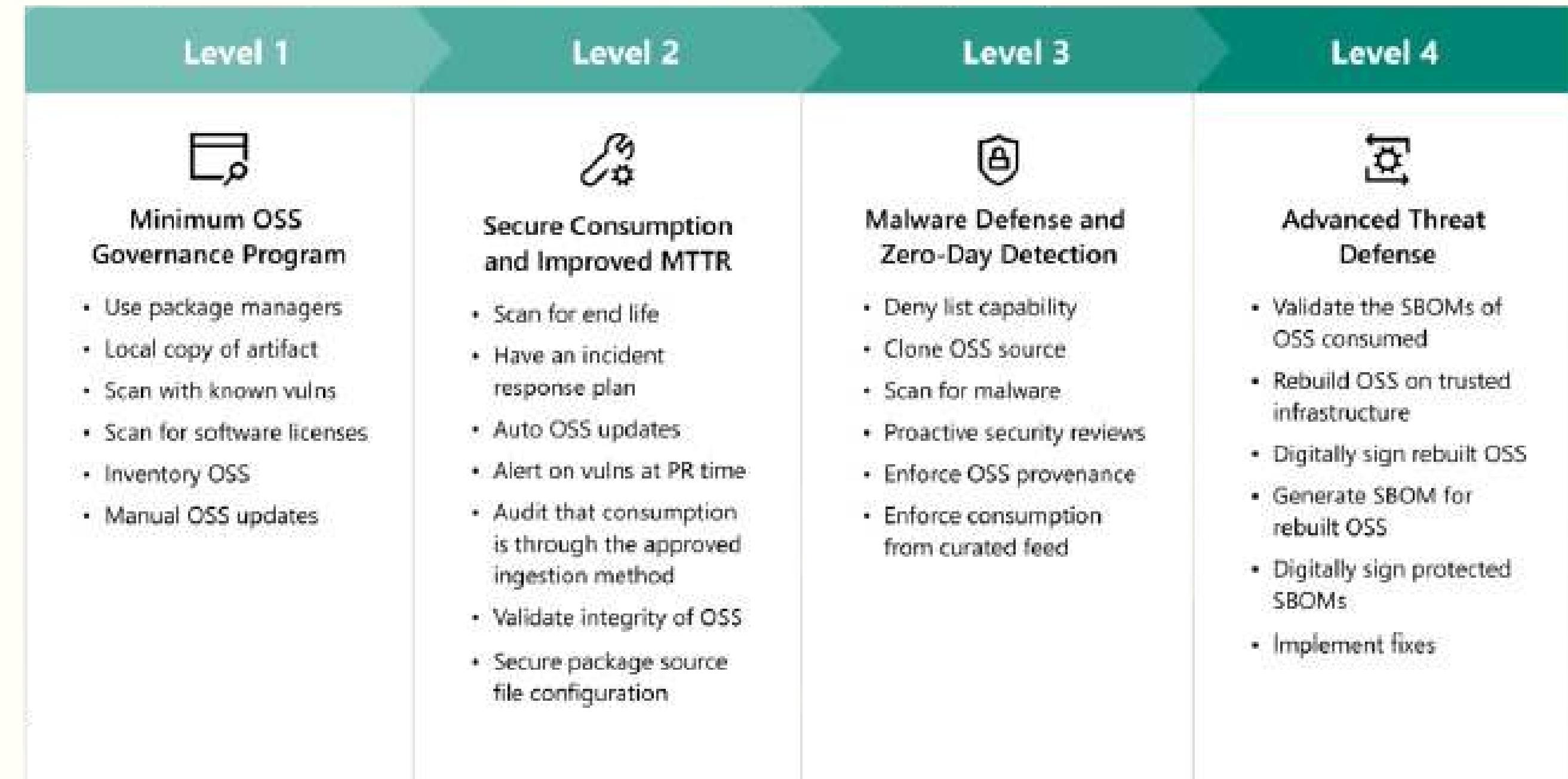
Fix

# Comprehensive List of Threats

OSS Supply Chain Threat	Real Example	Mitigation via S2C2F Requirement
Accidental vulnerabilities in OSS code or Containers that we inherit	<a href="#">SaltStack</a>	UPD-2 UPD-3
Intentional vulnerabilities/backdoors added to an OSS code base	<a href="#">colors v1.4.1</a>	SCA-5
A malicious actor compromises a distribution mirror of a package	<a href="#">phpMyAdmin</a>	AUD-3
A malicious actor compromises a known good OSS component and adds malicious code into the repo	<a href="#">ESLint incident</a>	ING-3 ENF-2 SCA-4
A malicious actor creates a malicious package that is similar in name to a popular OSS component to trick developers into downloading it	<a href="#">Typosquatting</a>	AUD-1 ENF-2 SCA-4

Example of list of threats and mitigation.

# Maturity Levels



Example of maturity roadmap provided.

# Detailed Controls

## Secure Supply Chain Consumption Framework Requirements

Below is a table of the requirements mapped to the 8 different practices. Two of the requirements have prerequisites identified that are outside the scope of this document to list as requirements.

Practice	Requirement ID	Maturity Level	Requirement Title	Benefit
Ingest it	ING-1	L1	Use package managers trusted by your organization	Your organization benefits from the inherent security provided by the package manager
	ING-2	L1	Use an OSS binary repository manager solution	Caches a local copy of the OSS artifact and protects against <a href="#">left-pad</a> incidents, enabling developers to continue to build even if upstream resources are unavailable
	ING-3	L3	Have a Deny List capability to block known malicious OSS from being consumed	Prevents ingestion of known malware by blocking ingestion as soon as a critically vulnerable OSS component is identified, such as <a href="#">colors v 1.4.1</a> , or if an OSS component is deemed malicious
	ING-4	L3	Mirror a copy of all OSS source code to an internal	Business Continuity and Disaster Recovery (BCDR)

Example of ingestion controls.

# Controls By Levels

Practice name	L1	L2	L3	L4
Ingest it – save a local copy of artifacts and source code	<p>[ING-1] Use package managers trusted by your organization</p> <p>[ING-2] Saving a local copy of the OSS artifact can be done by adopting an integrated package caching solution into your CI/CD infrastructure.</p> <p>All developers across your organization should standardize their consumption methods (using governed workflows) so that security policy can be enforced.</p> <p><b>Free Tools:</b> <a href="#">VCPKG for C/C++ OSS</a>, <a href="#">Pulp</a></p> <p><b>Paid Tools:</b> <a href="#">Artifacts</a>, <a href="#">GitHub Packages</a>, <a href="#">Azure</a></p>		<p>[ING-3] Having a Deny List capability to block ingestion of vulnerable and malicious OSS components is a required defensive tool in incident response situations.</p> <p>Having an incident response team that can rapidly respond and update the deny list is also critical.</p> <p><b>Paid Tool:</b> <a href="#">Nexus Firewall</a></p> <p>[ING-4] Saving a local copy of the OSS source code</p> <p><b>Free Tool:</b> <a href="#">Duplicating a repo</a></p>	

Example of control by level and resources provided.

# **Shortcomings of S2C2F**

- **Early** in the journey. Some rough edges need to be smoothed out.
- **Diminishing returns** of the higher levels.
- **Developer productivity decrease** caused by implementing the controls needs to be carefully considered.



# 8 Key Practices



Ingest



Scan



Inventory



Update



Audit



Enforce

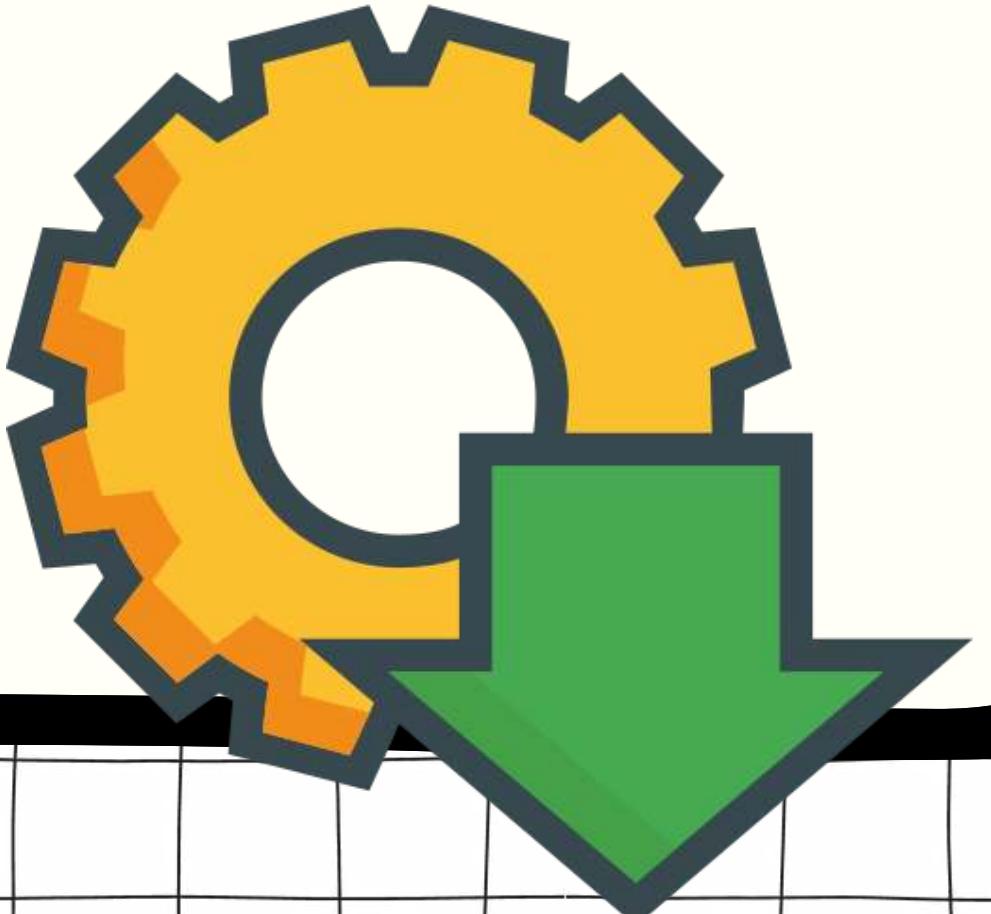


Rebuild



Fix

# Ingest

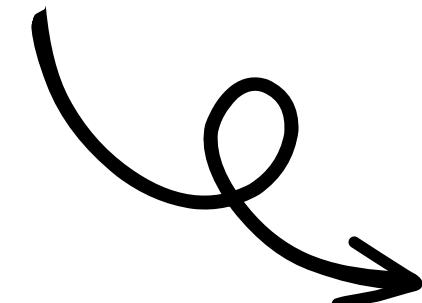


# Ingest

## Problem

Very little control over the source, resulting in very hard to prevent malicious dependencies from entering the supply chain.

...



## Approach

Establish more control over how developers consume OSS dependencies and prevent malicious packages from being used.

...

# Ingest - Controls

## **Level 1**

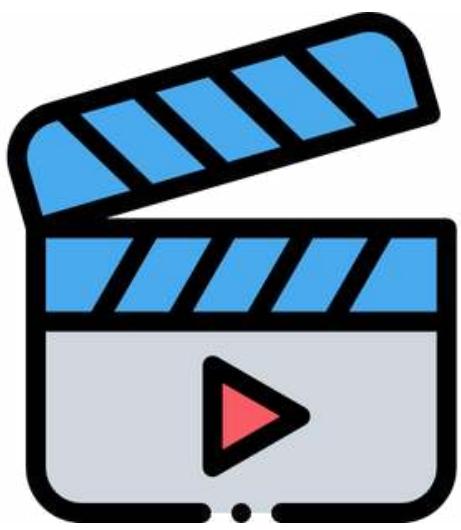
- Using package managers.
- Using OSS binary repository managers.

## **Level 3**

- Having a deny list capability.
- Mirroring a copy of all OSS dependencies internally.

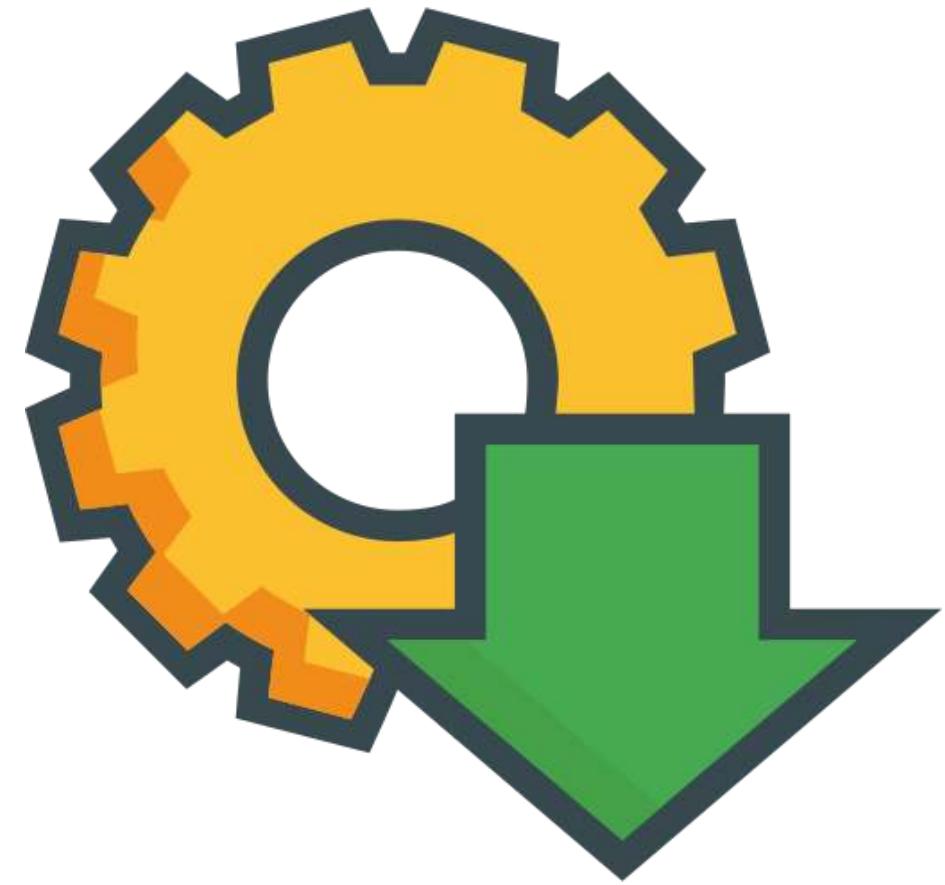


# Example



# Ingest - Example

- Preventing the build if malicious dependency exists in the software.
- Leverage existing free-to-use Synk GitHub Actions as sample tools.



# Ingest - Example

- Set up **GitHub Action** to run Snyk scan in every commit push.
- Fail build if critical or high-severity vulnerabilities exist in any of the packages.
- Notify of failure.



# Leveraging Pre-built GitHub Actions from Snyk

## GitHub Actions

*(i)* As of December 15, 2022, the GitHub Actions integration pages are being moved from the repository to the Snyk docs site. During this process the explanations will provide the same basic information but vary in presentation. If you need help contact [Snyk support](#).

### Overview of GitHub Actions Integration

Snyk offers a set of GitHub actions for using Snyk to check for vulnerabilities in your GitHub projects. These actions are based on the Snyk CLI and you can use all of its options and capabilities with the args in the properties of the action.

There is also a [Snyk Setup Action](#).

For additional information see the [GitHub Actions feature](#) page and the [GitHub custom actions documentation](#).

You must use a different action depending on the language or process you are using. This page provides detailed information that applies to [all GitHub Actions for Open Source languages and package managers](#). For [Open Source examples](#), see the pages listed in the next section, [GitHub Actions for Open Source languages and package managers](#).

For detailed information about the [Docker and IaC GitHub Actions and examples](#) see the pages listed in the subsequent section, [GitHub Actions for Snyk Container and Snyk Infrastructure as Code](#).

For detailed information about the [Setup Action and examples](#), see [Snyk Setup Action](#).

# Example Vulnerable Package

```
<!-- Dependency with a known critical vulnerability -->
<dependency>
    <groupId>commons-beanutils</groupId>
    <artifactId>commons-beanutils</artifactId>
    <version>1.9.2</version>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

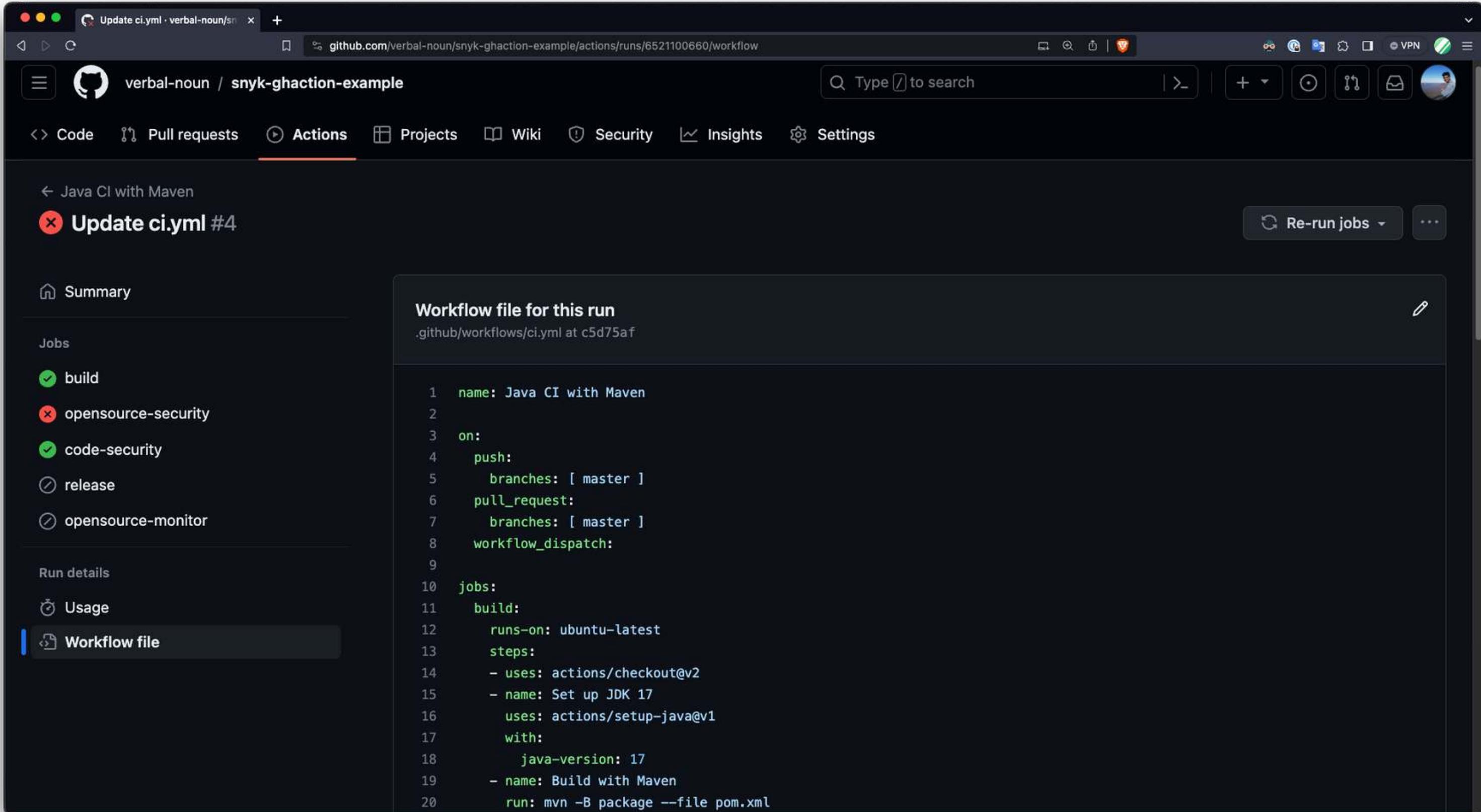
Example vulnerable package in pom.xml file

# Setting up Actions Workflow

```
22   opensource-security:
23     runs-on: ubuntu-latest
24     steps:
25       - uses: actions/checkout@master
26       - name: Run Snyk to check for vulnerabilities
27         uses: snyk/actions/maven@master
28         env:
29           SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
30         with:
31           args: --severity-threshold=high
```

Example step created in GitHub Actions

# Zooming out - Entire Workflow



The screenshot shows a GitHub Actions run page for a repository named "verbal-noun / snyk-ghaction-example". The page displays the workflow file for the current run, which is titled "Update ci.yml #4". The workflow file is defined in a .github/workflows/ci.yml file at commit c5d75af. The workflow has several jobs: "build" (green checkmark), "opensource-security" (red X), "code-security" (green checkmark), "release" (grey circle), and "opensource-monitor" (grey circle). The "Workflow file" tab is selected in the sidebar.

```
Workflow file for this run  
.github/workflows/ci.yml at c5d75af

1 name: Java CI with Maven
2
3 on:
4   push:
5     branches: [ master ]
6   pull_request:
7     branches: [ master ]
8   workflow_dispatch:
9
10 jobs:
11   build:
12     runs-on: ubuntu-latest
13     steps:
14       - uses: actions/checkout@v2
15       - name: Set up JDK 17
16         uses: actions/setup-java@v1
17         with:
18           java-version: 17
19       - name: Build with Maven
20         run: mvn -B package --file pom.xml
```

Visual purposes only - no need to read

# Let's see it in action



# Capturing issues in packages

opensource-security  
failed 3 minutes ago in 35s

Run Snyk to check for vulnerabilities 20s

```
Testing /github/workspace...
Tested 37 dependencies for known issues, found 7 issues, 7 vulnerable paths.

Issues to fix by upgrading:
Upgrade commons-beanutils:commons-beanutils@1.9.2 to commons-beanutils:commons-beanutils@1.9.4 to fix
  x Deserialization of Untrusted Data [High Severity]
  [https://security.snyk.io/vuln/SNYK-JAVA-COMMONSBEANUTILS-460111] in commons-beanutils:commons-beanutils@1.9.2
  introduced by commons-beanutils:commons-beanutils@1.9.2
  x Deserialization of Untrusted Data [Critical Severity]
  [https://security.snyk.io/vuln/SNYK-JAVA-COMMONSCOLLECTIONS-30078] in commons-collections:commons-collections@3.2.1
  introduced by commons-beanutils:commons-beanutils@1.9.2 > commons-collections:commons-collections@3.2.1
```



Scan

# Scan

## Problem

New vulnerabilities and issues are being discovered continuously in the OSS. It is hard to identify new risks without proactive scans.

...

## Approach

Establish scanning capability for a wide variety of issues depending on the company's priorities.

...

# Scan - Controls

## Level 1

- Scanning OSS for known vulnerabilities.
- Scanning OSS for licences.

## Level 2

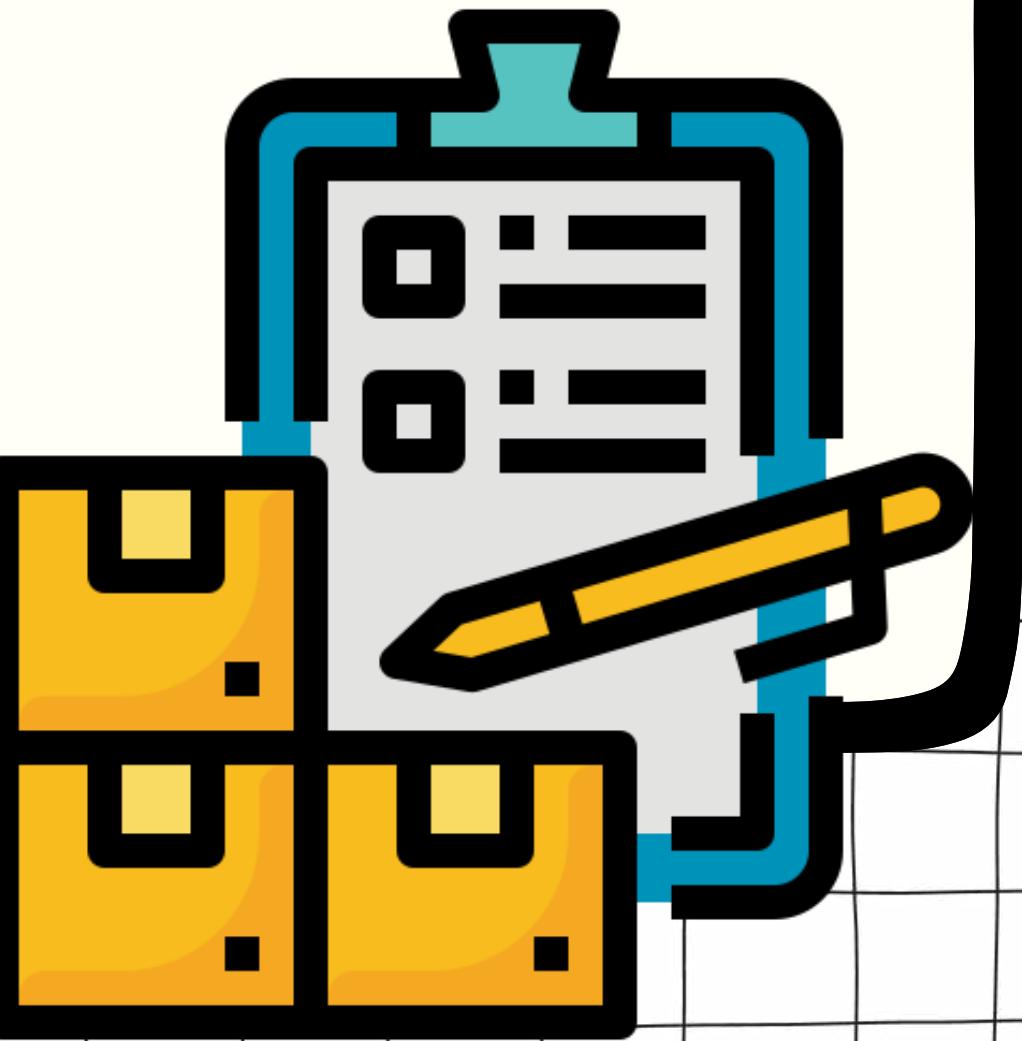
- Scanning for End-of-Life

## Level 3

- Scanning OSS for malware.
- Security review of OSS dependencies.



# Inventory



# Inventory

## Problem

It is hard to protect against threats if we do **not know** which dependencies we **use** directly and indirectly.

...

## Approach

Producing an **inventory** of dependencies and also **being prepared** to tackle issues when they arise based on inventory.

...

# Inventory - Controls

## **Level 1**

- Maintaining an automated inventory of all OSS used / Generating SBOM.

## **Level 2**

- Having an OSS incident response plan.



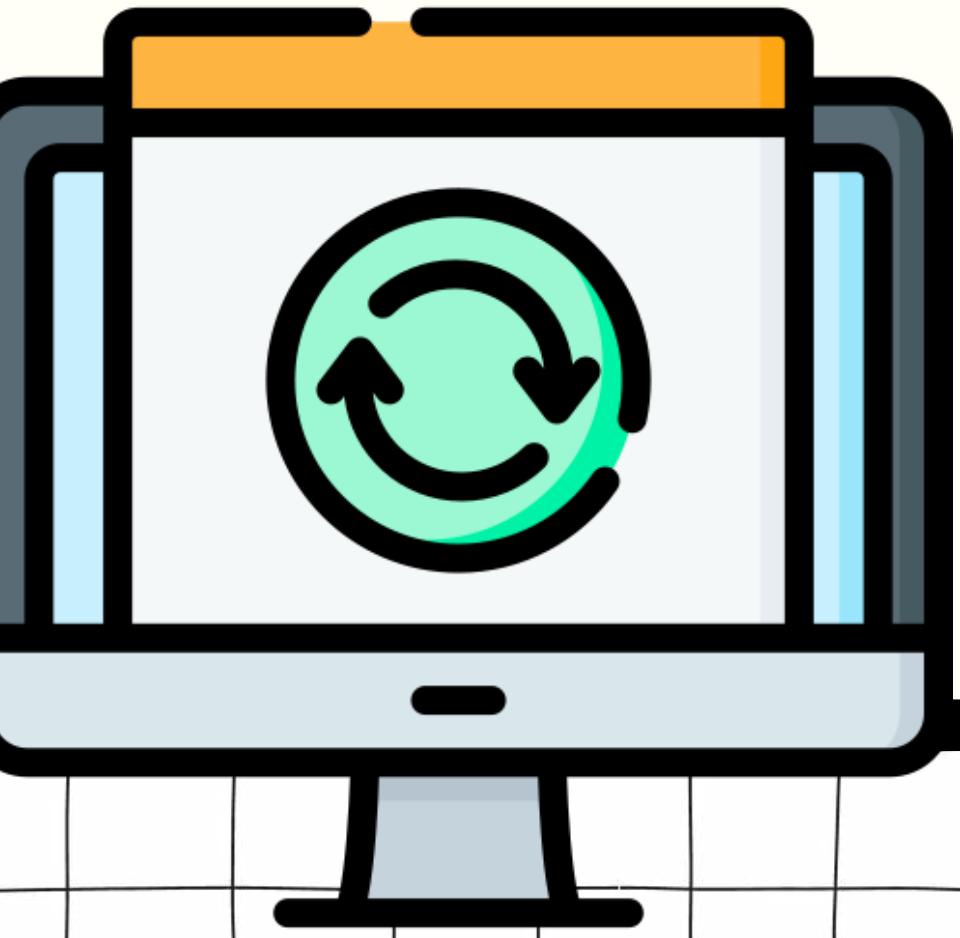
# Example





Just go to the vendor area, and you will get all the  
SBOM pitches you want. 😂

# Update



# Update

## Problem

When a new vulnerability is found, **attackers move very quickly**.

On the other hand, developers are often **not motivated** to prioritise fixing library issues.



## Approach

Reduce the **cognitive load** on developers and **time** to update through **automation** and **tooling**.



# Update - Controls

## Level 1

- Update vulnerable OSS manually.

## Level 2

- Enable automated OSS updates.
- Display OSS vulnerabilities as comments in Pull Requests (PRs)



# Update - Examples

The screenshot shows the GitHub organization page for Dependabot. At the top, there's a blue hexagonal icon with a white robot face and two blue hearts. Below it, the organization name "Dependabot" is displayed in bold black text, followed by the subtitle "Automated dependency updates built into GitHub". Key statistics are shown: 1.3k followers, United States of America location, a link to security features, an email address, and a green "Verified" badge. A "Pinned" section contains links to several repositories:

- dependabot-core** (Public) - Dependabot's core logic for creating update PR's. (Ruby, 3.6k stars, 863 forks)
- fetch-metadata** (Public) - Extract information about the dependencies being updated by a Dependabot-generated PR. (TypeScript, 115 stars, 60 forks)
- cli** (Public) - A tool for testing and debugging Dependabot update jobs. (Go, 107 stars, 21 forks)
- smoke-tests** (Public) - A collection of manifest files for various package managers and is used to perform end-to-end tests for Dependabot. (HCL, 12 stars, 13 forks)
- dependabot-script** (Public) - A simple script that demonstrates how to use Dependabot Core. (Ruby, 506 stars, 274 forks)
- demo** (Public template) - Fork me to try out Dependabot.

The screenshot shows the GitHub repository page for Renovate. The header "readme.md" is visible. The main content features a large teal circle with a white paint roller icon, followed by the word "Renovate" in a large, bold, sans-serif font. To the right is a decorative graphic of a green circuit board. Below this, the word "Renovate" is repeated in a smaller font, followed by a link icon. A sub-section titled "Automated dependency updates. Multi-platform and multi-language." is shown, along with a row of status badges: license AGPL-3.0-only, codecov 100%, renovate enabled, Build failing, docker pulls 1.3G, and openssf scorecard 8.1. A section titled "Why Use Renovate?" includes a bullet point: "Get automated Pull Requests to update your dependencies".

# Audit



# Audit

## Problem

Often, **protective controls are/can be bypassed** by attackers and well-meaning but misguided developers.

...

## Approach

Establishing the **ability to audit OSS consumption** to see if it's coming through the standardised consumption tools or not.

...

# **Audit - Controls**

## **Level 2**

- Detecting when OSS is consumed using a non-standard medium.
- Validating integrity.

## **Level 3**

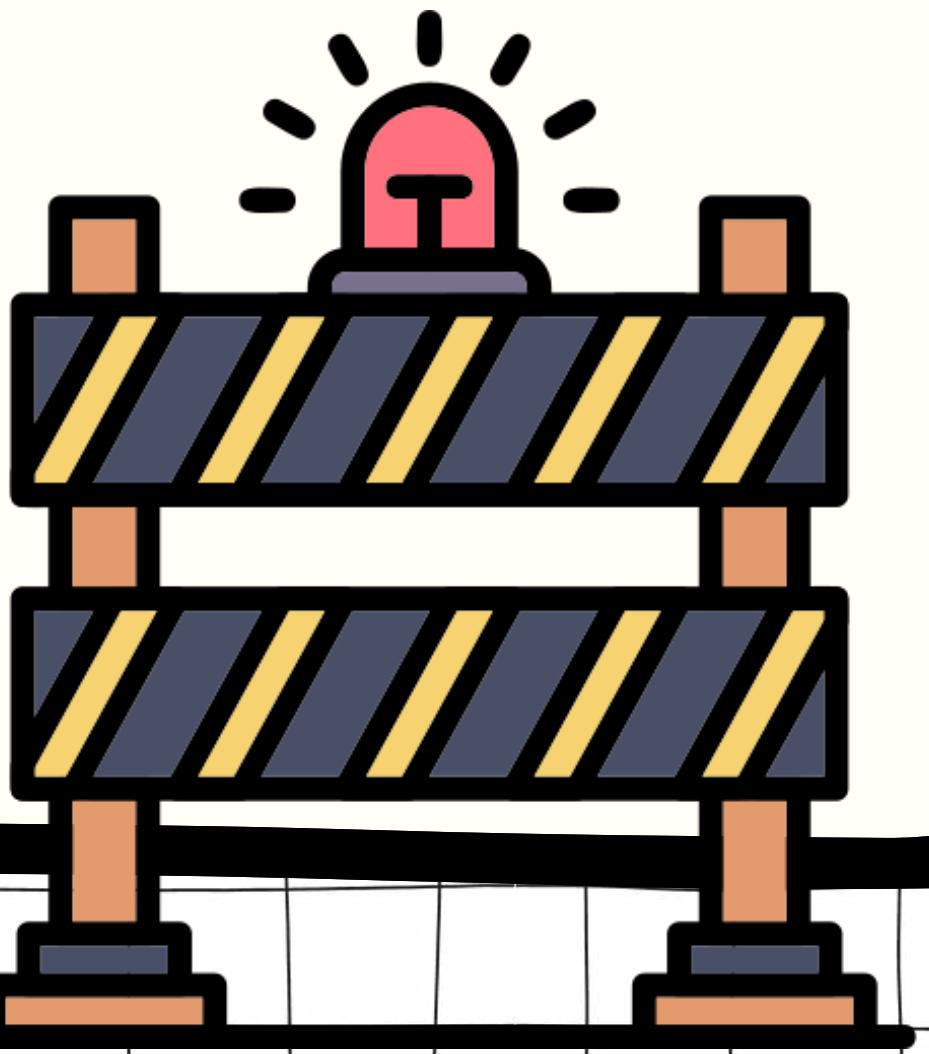
- Verifying the provenance of your OSS.

## **Level 4**

- Validating SBOM for provance data and its digital signature for integrity.



Enforce



# Enforce

## Problem

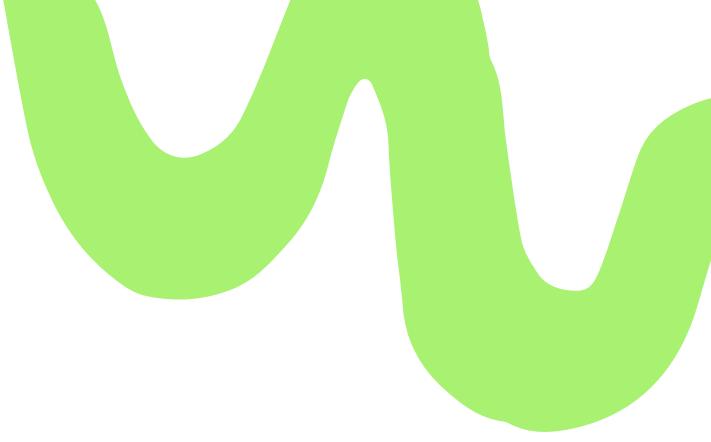
Often, **protective controls are/can be bypassed** by attackers and well-meaning but misguided developers.

...

## Approach

**Enforcement** of secure channels to be used only.

...



# **Enforce - Controls**

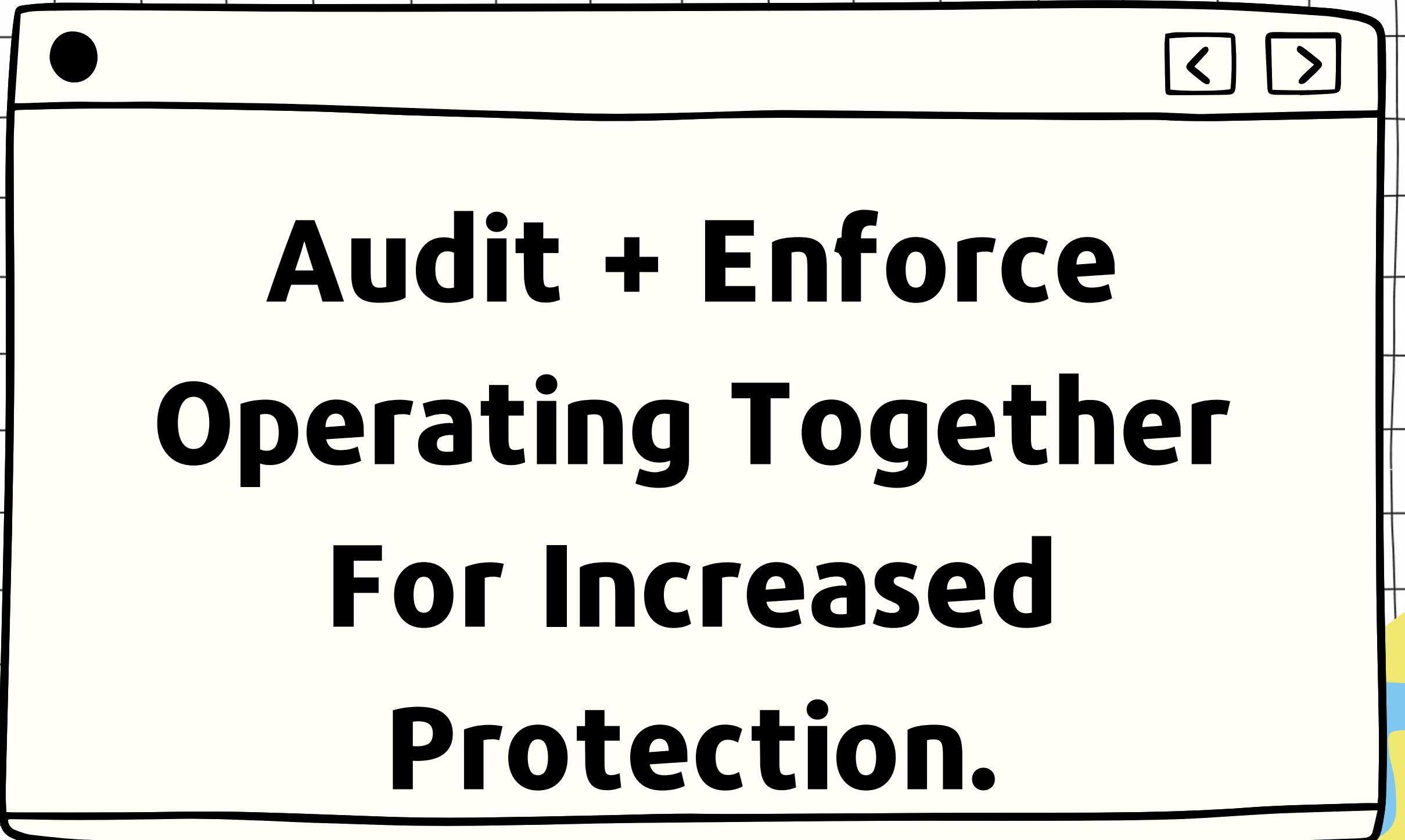
## **Level 2**

- Securely configuring package source files.
  - Package source mapping.
  - Single upstream feed.
  - Version pinning and lock files.

## **Level 3**

- Enforce usage of a curated OSS feed.





**Audit + Enforce**

**Operating Together**

**For Increased**

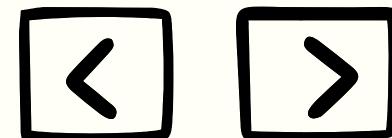
**Protection.**

# Parting Thoughts

- It's not one size fits all.
- We must identify our organisation's risks and prioritise controls accordingly.
- Contributing back to OSS and community.



# Everything Cyber



EVERYTHING CYBER

Cybersecurity  
and beyond

WEEKLY  
EPISODES



Everything Cyber

@Everything\_Cyber 244 subscribers 26 videos

Welcome to Everything Cyber. Our channel is dedicated to up-skilling ourse... >

[linkedin.com/company/everything-cyber](https://linkedin.com/company/everything-cyber)

Subscribed

HOME

VIDEOS

PLAYLISTS

COMMUNITY

CHANNELS

ABOUT



Videos ► Play all



Ep. 16 - Cybersecurity In  
Healthcare w/ Alana Tobgui

32 views • 8 days ago



Ep. 15 - Building Zero Trust  
Network Infrastructure w/...

80 views • 1 month ago



Ep. 14 - Data-driven security  
w/ Jeevan Singh

88 views • 1 month ago



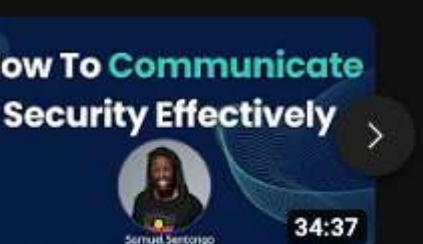
Ep. 13 - Leadership In  
Application Security w/...

330 views • 1 month ago



Ep. 12 - Level-up your Threat  
Intelligence game with the...

225 views • 4 months ago



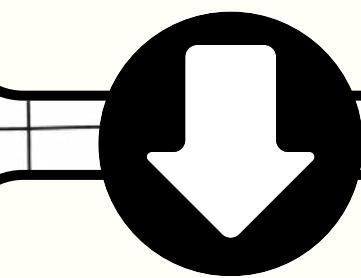
Ep. 11 - Leading with Impact:  
Strategies for Effective...

92 views • 5 months ago

CC

## Our Socials

-  Kaif Ahsan
-  Kumar Soorya



## Slides & More



# Thank you

Please feel free to reach out.





# THANK YOU

Slides and contact info:

