

DWX

# DEVELOPER WEEK '23

## Protect yourself against supply chain attacks

Rob Bos

DevOps Consultant – Xpirit

The Netherlands

<https://devopsjournal.io>



<https://myoctocat.com>

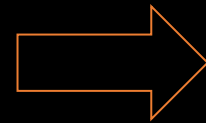
# Topics

- **Why:**
  - Attack examples
- **Protection / Maturity: frameworks**
  - OWASP Software Component Verification Standard (SCVS)
  - Supply chain Levels for Software Artifacts (SLSA)

# Supply Chain – Dependencies

- Libraries used by your application:

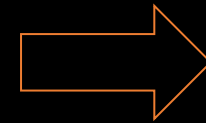
- Authentication
- Encryption
- Database connections



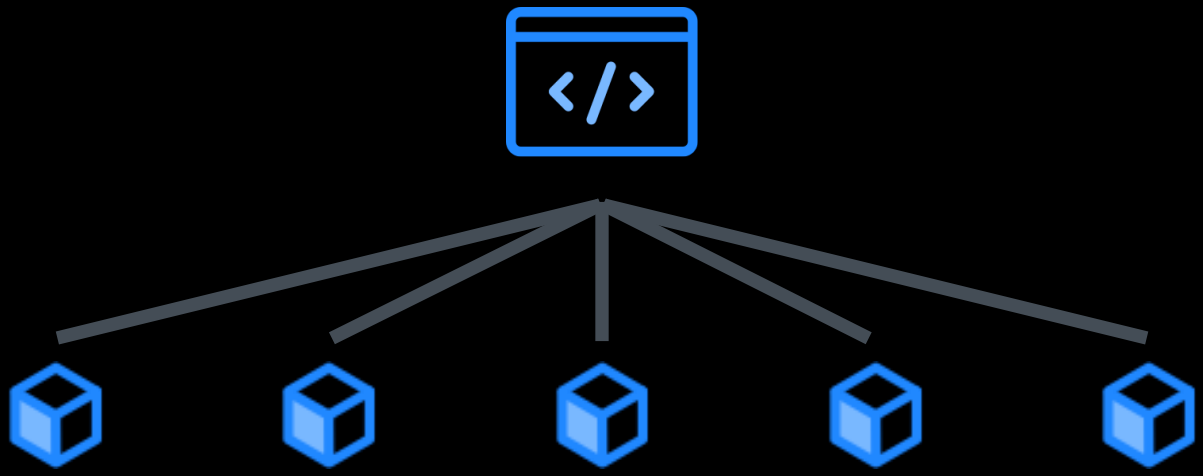
Package managers

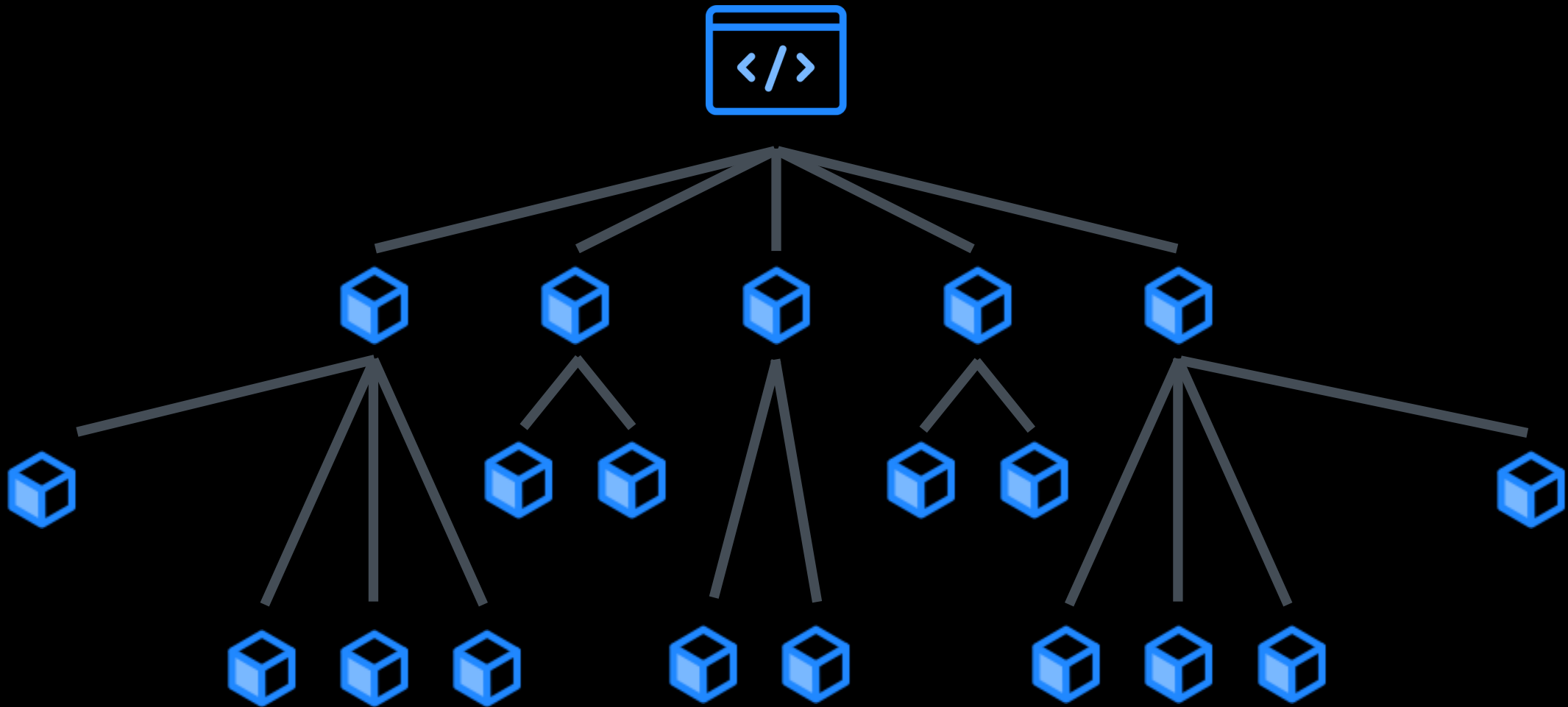
- Tooling used for building your application:

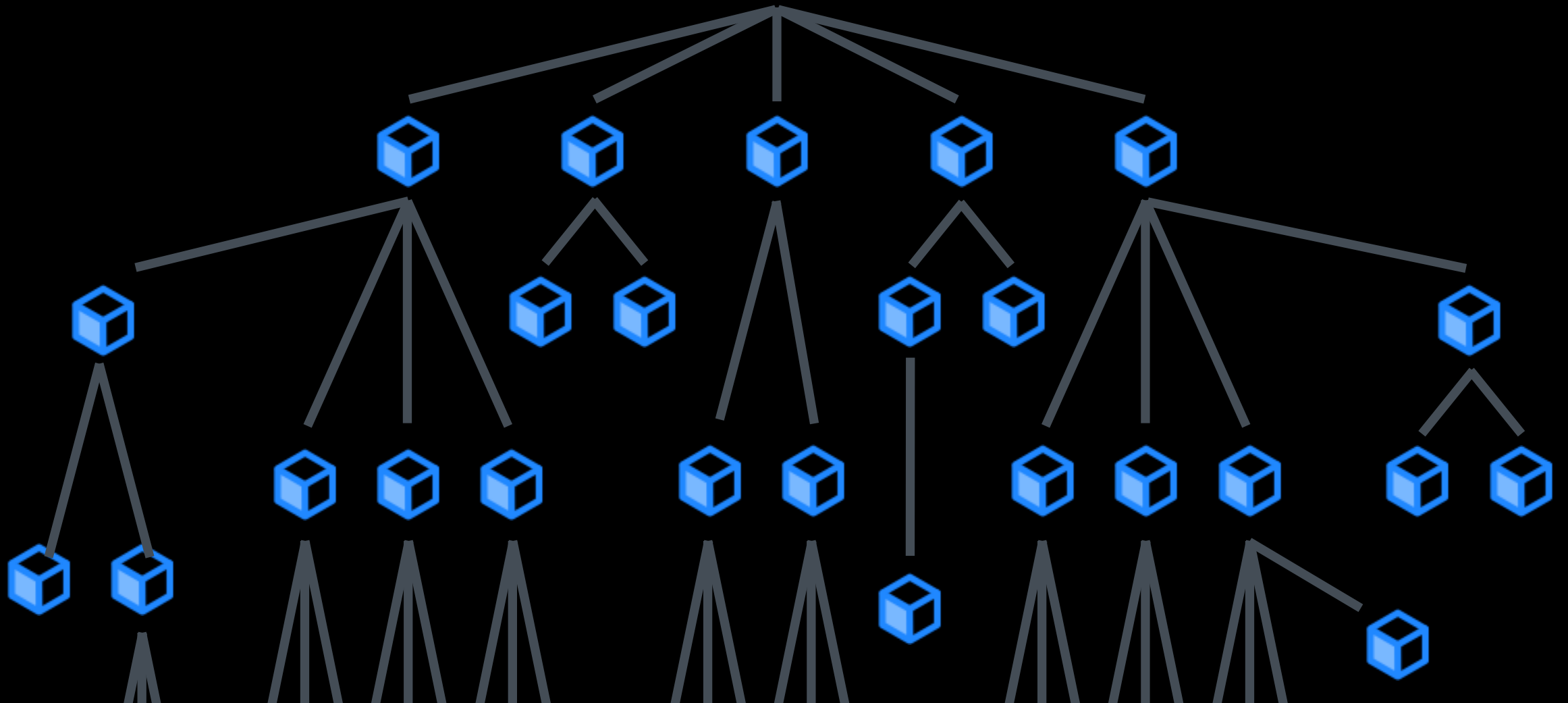
- npm ci
- dotnet build
- pipelines

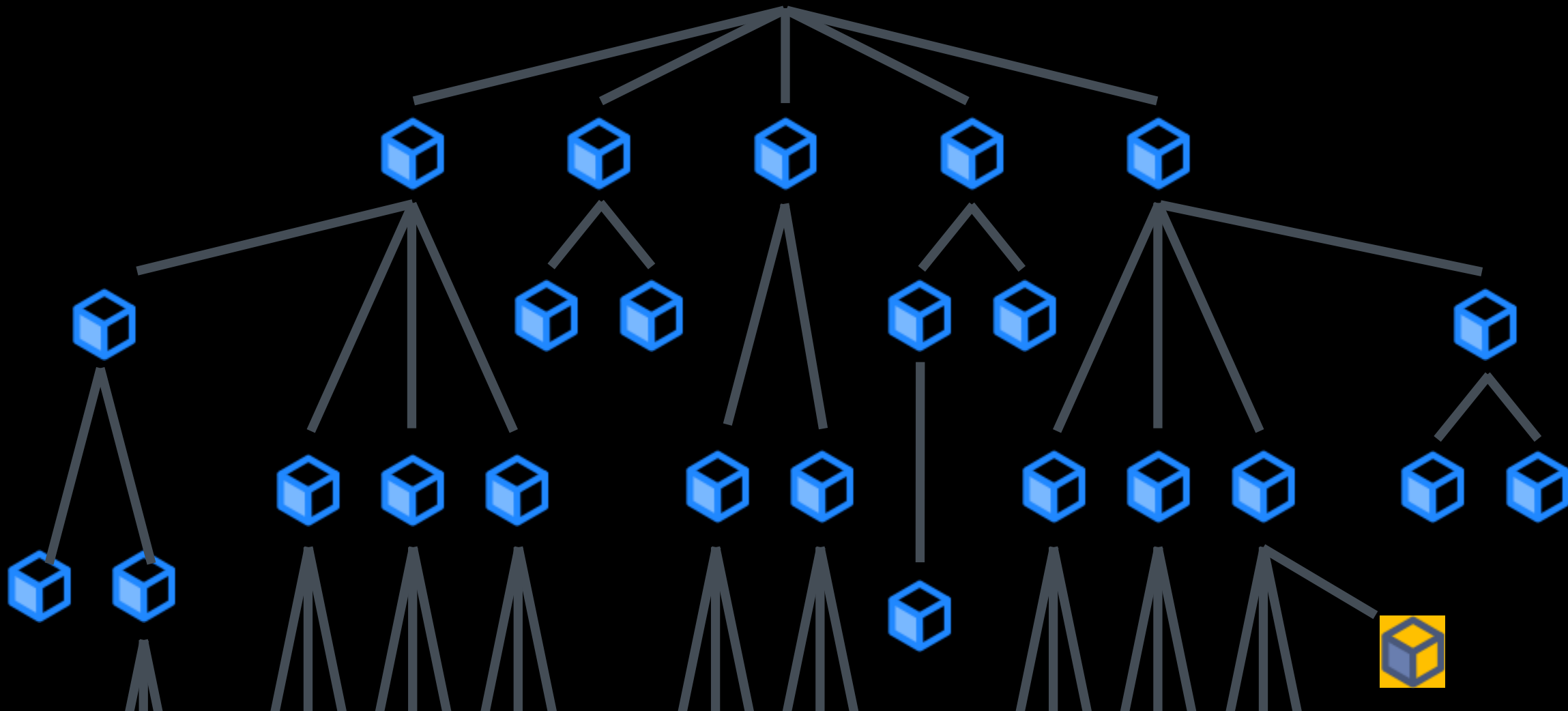


Build tools









~80-90% of all code in distributed applications is Open Source





**4 years**

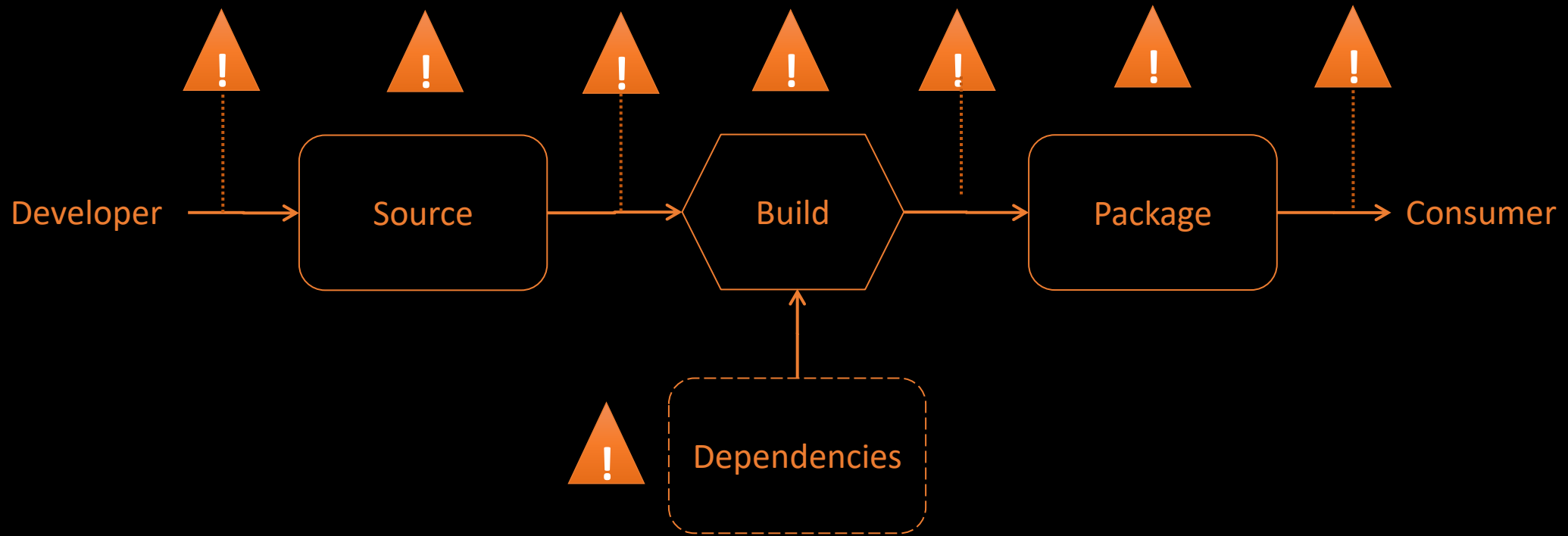
On average, vulnerabilities go undetected for four years before being identified.

Sometimes, even longer than that:  
Log4j was vulnerable for ~7 years

# 180+ days

Mean time to remediate (MTTR)  
Industry norm

# Attack entries

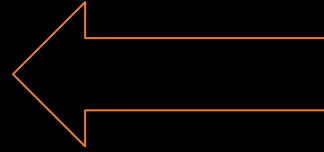


# Supply Chain Confusion

- Typo squatting
- Namespace shadowing
- Configuration files
  
- Pipeline attacks
- Pipeline artifact attacks

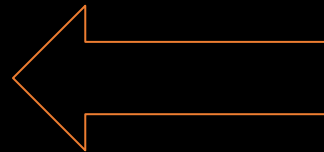
# Typo squatting 101

`npm install crossenv`



Steals all your  
environment variables

`npm install cross-env`



Normal package

# Add Namespaces

```
npm install @babel/helper-regex
```

More specific, less change of squatting

Not all publishers use a namespace

# March 2022 – Namespace confusion


```
npm install @azure/core-tracing
```

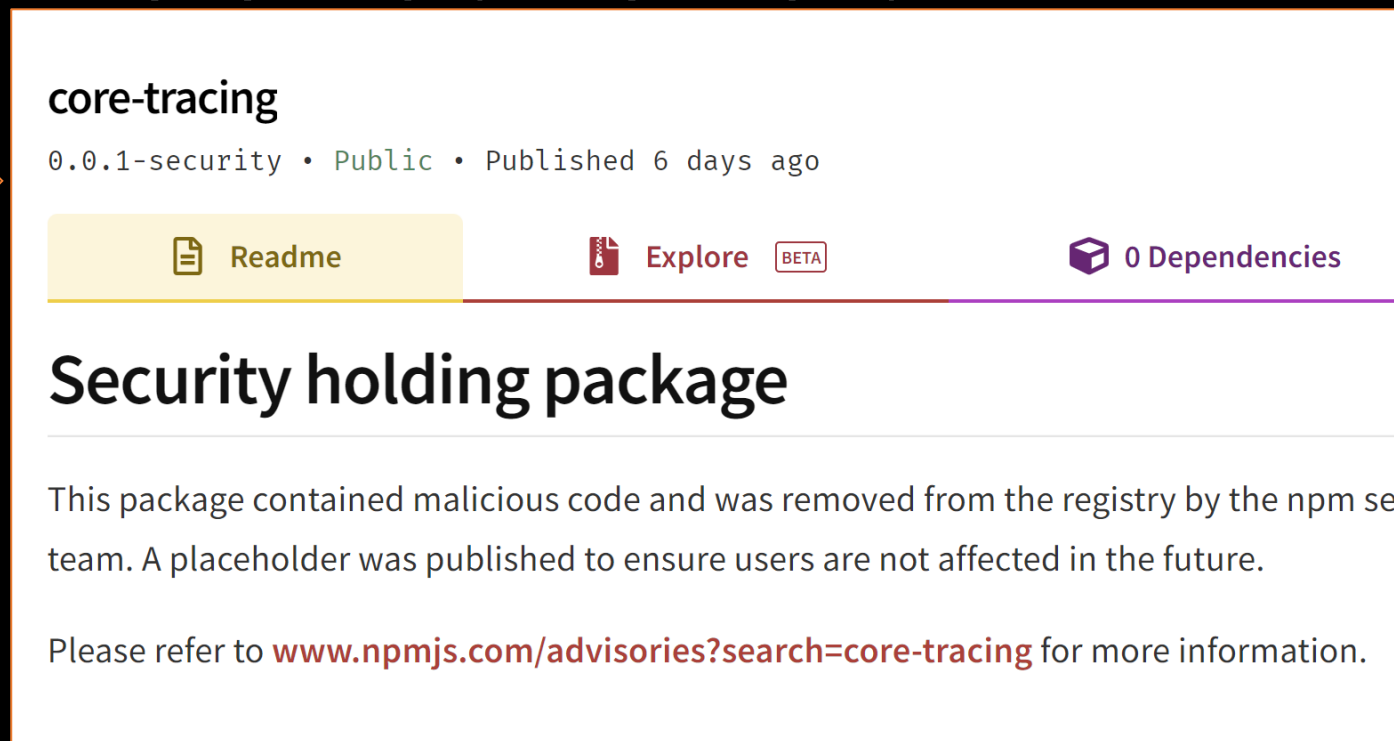
vs

```
npm install core-tracing
```

→ 218 packages squatted

# Timelines

- Attacks spread fast, lots of targets
- @azure namespace attack: 50 downloads per package (x 218 packages!)
- Found within 1-2 days
- **Blocked by npm** after noticing 
- Damage is already done by then



The screenshot shows the NPM package page for 'core-tracing'. The package version is 0.0.1-security, it is public, and was published 6 days ago. There are buttons for 'Readme', 'Explore', and 'BETA'. It shows 0 dependencies. Below the package information, there is a section titled 'Security holding package' with a warning message: 'This package contained malicious code and was removed from the registry by the npm security team. A placeholder was published to ensure users are not affected in the future.' It also includes a link to [www.npmjs.com/advisories?search=core-tracing](http://www.npmjs.com/advisories?search=core-tracing) for more information.



# Typo squatting 101

And this is only getting worse!

## The danger of .zip domains

This downloads postgres v15

<https://github.com/postgres/postgres/tags/v15.zip>

hostname

This resolves to v15.zip domain

<https://github.com/postgres/postgres/tags/@v15.zip>

userinfo

hostname

# Protect yourself

- Software Composition Analysis is the starting point
- Use SAST or DAST (shift left or you are too late!)
- Package manager scanners:
  - WhiteSource
  - BlackDuck
  - JFrog Artifactory + Xray
  - Snyk.io
  - GitHub Dependabot + Vulnerability alerts



Checks package + version against CVE databases

**AFTER THE FACT**

# Protect yourself

`npm audit`

- Gets a list of the dependencies and posts that to:

<https://registry.npmjs.org/-/npm/v1/security/advisories/bulk>

- Might send out private data!
- Any packages without a version field will be ignored!

**AFTER THE FACT**

# npm audit - results

```
node_modules/url-parse
```

```
ws 5.0.0 - 5.2.2 || 6.0.0 - 6.2.1
```

```
Severity: moderate
```

```
ReDoS in Sec-WebSocket-Protocol header - https://github.com/advisories/GHSA-6fc8-4gx4-v693
```

```
ReDoS in Sec-WebSocket-Protocol header - https://github.com/advisories/GHSA-6fc8-4gx4-v693
```

```
fix available via `npm audit fix`
```

```
node_modules/jest-environment-jsdom-fourteen/node_modules/ws
```

```
node_modules/webpack-dev-server/node_modules/ws
```

```
node_modules/ws
```

```
57 vulnerabilities (25 moderate, 31 high, 1 critical)
```

```
To address issues that do not require attention, run:
```

```
npm audit fix
```

```
To address all issues (including breaking changes), run:
```

```
npm audit fix --force
```

**AFTER THE FACT**

# npm install -g npq

- Wrapper from snyk.io
- Alias it to overwrite npm commands

```
→ npm install amp-html
✓ Checking package maturity
✗ Identifying package author...
✗ Checking package download popularity
✓ Checking availability of a LICENSE
✓ Checking availability of a README
✓ Identifying package repository...
✓ Checking package for pre/post install scripts
✗ Checking for known vulnerabilities
Detected possible issues with the following packages:
[amp-html]
- the package description has no e-mail associated with author(s). Proceed with care.
- detected a low download-count package (downloads last month < 20)
- 1 vulnerabilitie(s) found: https://snyk.io/vuln/npm:amp-html

? Would you like to continue installing package(s)? (y/N) █
```

# .npmrc – misconfiguration

```
registry=https://registry.npmjs.org/
```

```
@myscope:registry=https://mycustomregistry.example.org
```

All your private packages will now get pulled from npmjs.org!

# Topics

- Why:
  - Attack examples
- Protection / Maturity: frameworks
  - OWASP Software Component Verification Standard (SCVS)
  - Supply chain Levels for Software Artifacts (SLSA)

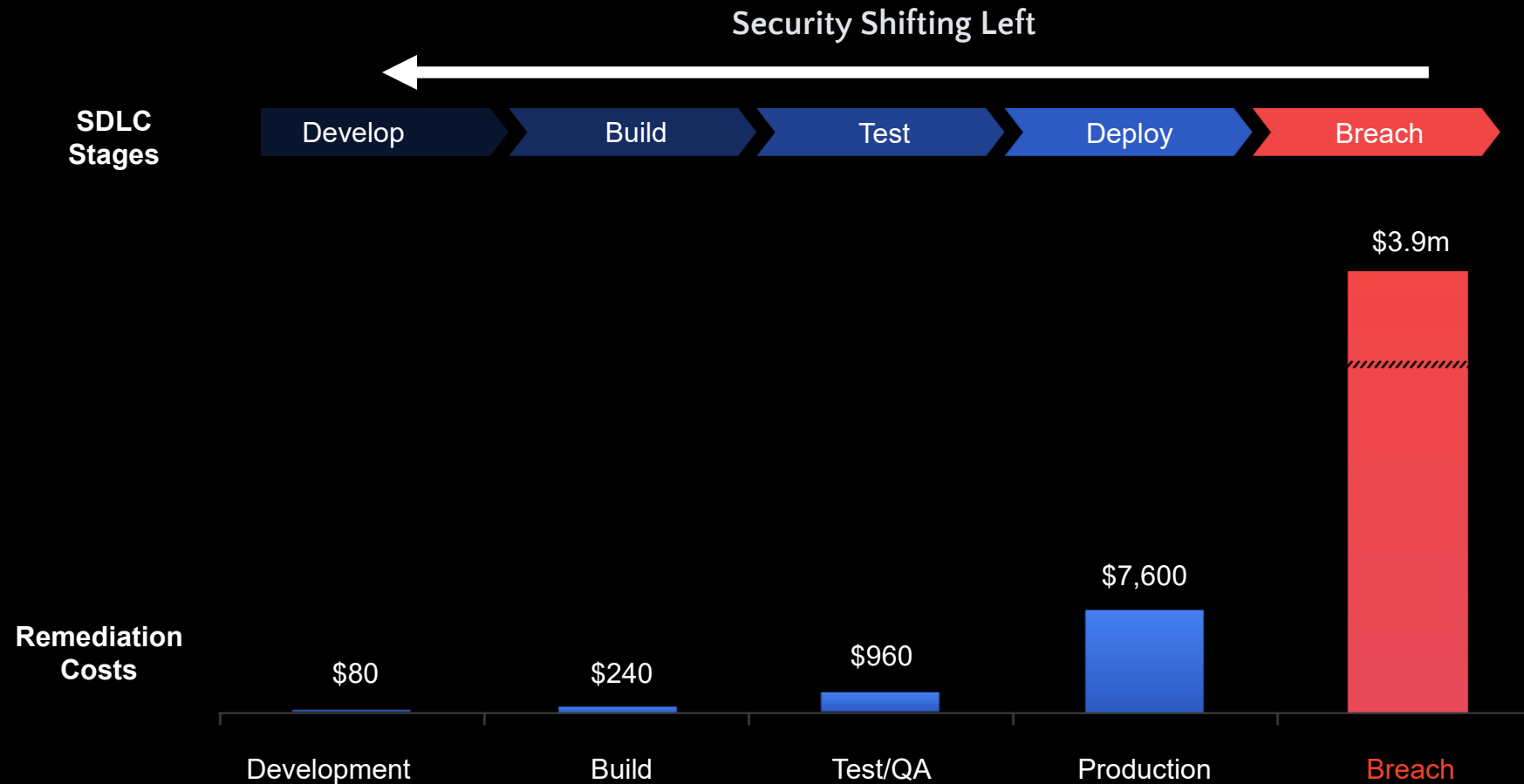


# Mistakes happen

Some of those become a security risk!

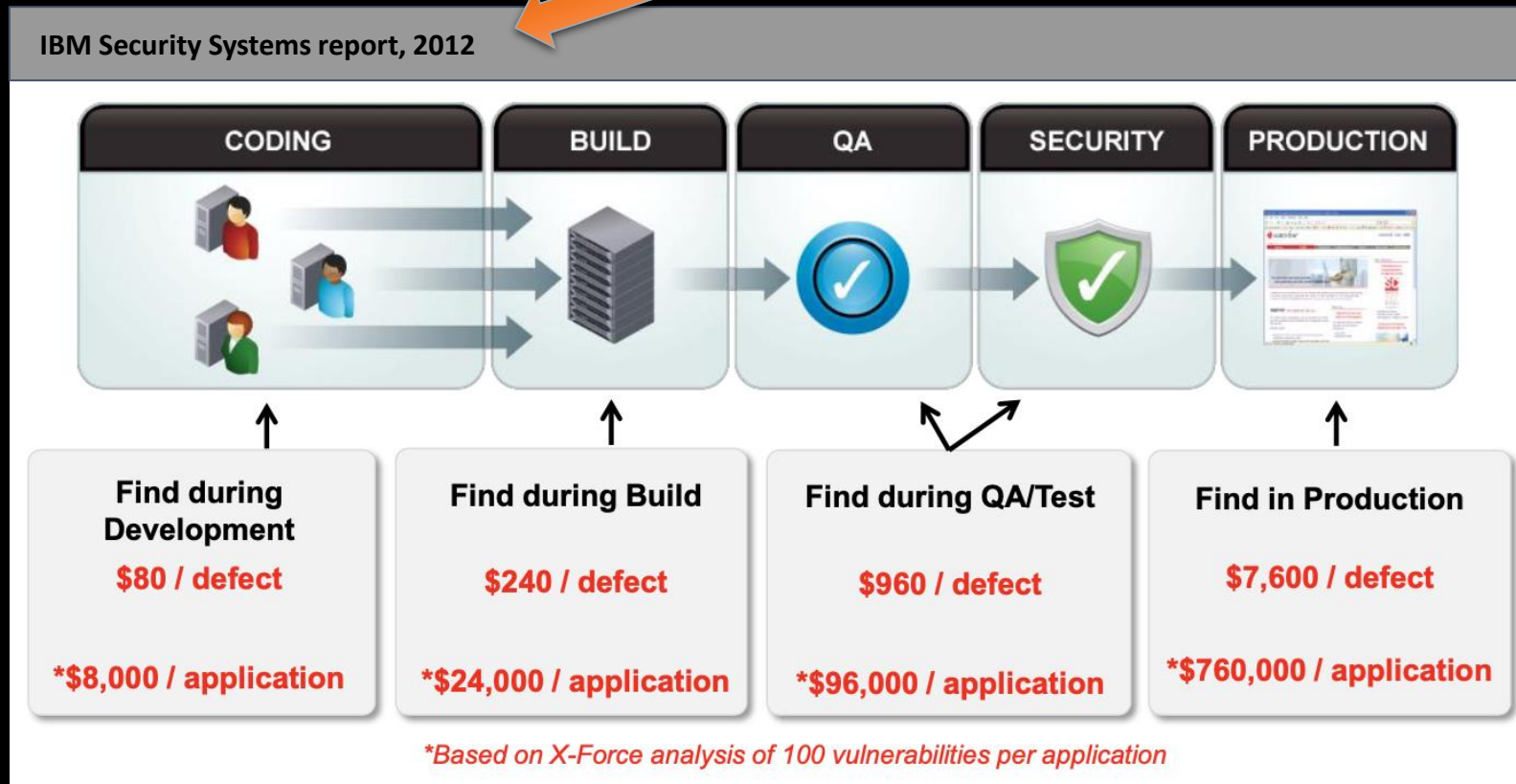


# Everyone wants to shift security left...



Source: Ponemon Institute Cost of a Data Breach 2020

# ... but the industry has been trying to shift left for at least a decade



# Frameworks



**OWASP**

**Software Component Verification Standard**

v1 since 2020: <https://xpir.it/SCVS>



**SLSA**

**The Linux Foundation**

**Supply chain Levels for Software Artifacts**

v1.0: April 2023: <https://slsa.dev/>

# OWASP SCVS: Software Component Verification Standard

- Assessment of your software components and how they came to be



Packages/Source



Pipelines

# OWASP SCVS

	L1	L2	L3	L4
V1 – Inventory				
V2 – Software Bill of Materials (SBOM)				
V3 – Build Environment				
V4 – Package Management				
V5 – Component Analysis				
V6 – Pedigree and Provenance				

# OWASP SCVS – V1 Inventory

All direct and transitive components and their versions are known at completion of a build

Package managers are used to manage all third-party binary components

Software bill of materials continuously maintained and current for all systems

Software bill of materials are required for new procurements

The component type is known throughout inventory

The component function is known throughout inventory

# V1 - Inventory

## Software Composition Analysis

- GitHub **Dependabot**
- Black Duck
- Mend (form. WhiteSource) Bolt
- Snyk
- Jfrog Xray

Dependabot Example: <https://github.com/devops-actions/issue-comment-tag>

# OWASP SCVS – V2 Software Bill of Materials

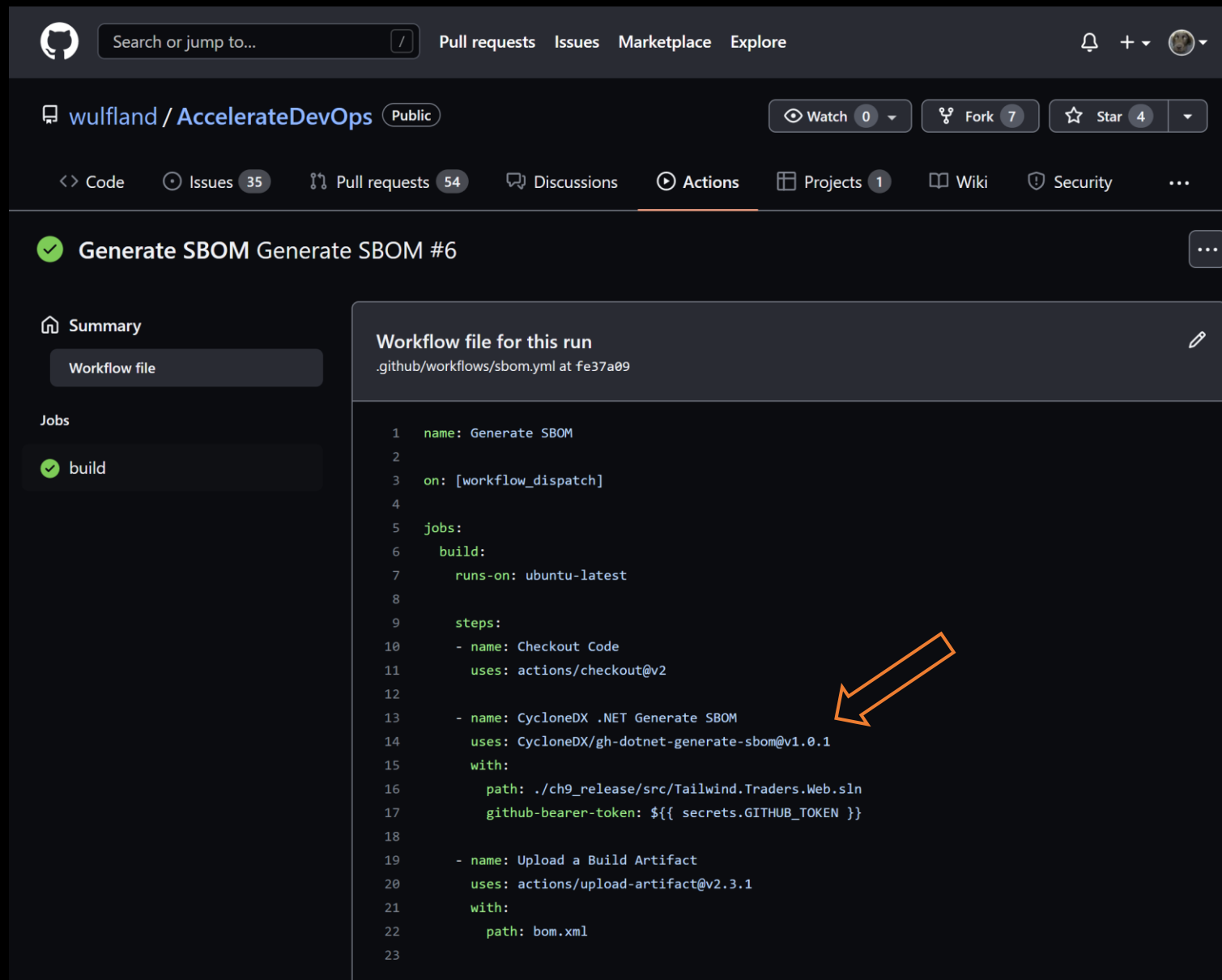
- SBOM creation is automated and reproducible
- SBOM has been signed by publisher, supplier, or certifying authority
- SBOM signature verification exists and is **performed**
- SBOM is analyzed for risk



# V2 Software Bill of Materials

- Multiple standards for SBOM formats:
  - SPDX (Software Package Data Exchange) – Linux Foundation
    - Focusses on license information
    - ISO Standard
  - CycloneDX – OWASP
    - Focusses on vulnerabilities and security
- Example SBOM creation with GitHub Actions & CycloneDX:  
[AccelerateDevOps/sbom.yml at main · wulfland/AccelerateDevOps · GitHub](https://github.com/wulfland/AccelerateDevOps/blob/main/accelerate-devops/sbom.yml)

# V2 Software Bill of Materials – demo



The screenshot shows a GitHub repository page for 'wulfland / AccelerateDevOps'. The main content is a workflow run titled 'Generate SBOM Generate SBOM #6'. The workflow file is displayed in a code editor, showing the following configuration:

```
1 name: Generate SBOM
2
3 on: [workflow_dispatch]
4
5 jobs:
6   build:
7     runs-on: ubuntu-latest
8
9     steps:
10      - name: Checkout Code
11        uses: actions/checkout@v2
12
13      - name: CycloneDX .NET Generate SBOM
14        uses: CycloneDX/gh-dotnet-generate-sbom@v1.0.1
15        with:
16          path: ./ch9_release/src/Tailwind.Traders.Web.sln
17          github-bearer-token: ${ secrets.GITHUB_TOKEN }
18
19      - name: Upload a Build Artifact
20        uses: actions/upload-artifact@v2.3.1
21        with:
22          path: bom.xml
23
```

An orange arrow points to the 'CycloneDX .NET Generate SBOM' step in the workflow file.

# V2 Software Bill of Materials – demo

```
bom.xml x
C: > Users > RobBos > AppData > Local > Temp > Temp1_artifact.zip > bom.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <bom xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" serialNumber=
3  <metadata>
4  <tools>
5  <tool>
6  <vendor>CycloneDX</vendor>
7  <name>CycloneDX module for .NET</name>
8  <version>2.3.0.0</version>
9  </tool>
10 </tools>
11 <component type="application" bom-ref="Tailwind.Traders.Web@0.0.0">
12 <name>Tailwind.Traders.Web</name>
13 <version>0.0.0</version>
14 </component>
15 </metadata>
16 <components>
17 <component type="library" bom-ref="pkg:nuget/Azure.Core@1.8.1">
18 <publisher>Microsoft</publisher>
19 <name>Azure.Core</name>
20 <version>1.8.1</version>
21 <description>This is the implementation of the Azure Client Pipeline</description>
22 <scope>required</scope>
23 <hashes>
24 <hash alg="SHA-512">87135CD530138F27E7C52BA23FB91CE37DE7AE0A016E08D4A5ABF33BD80B0D168996E3A437378B8BDC16667E2
25 </hashes>
26 <licenses>
27 <license>
28 <id>MIT</id>
29 </license>
30 </licenses>
31 <copyright>© Microsoft Corporation. All rights reserved.</copyright>
32 <purl>pkg:nuget/Azure.Core@1.8.1</purl>
33 <externalReferences>
34 <reference type="website">
35 <url>https://github.com/Azure/azure-sdk-for-net/blob/Azure.Core_1.8.1/sdk/core/Azure.Core/README.md</url>
36 </reference>
```

# OWASP SCVS V3 – Build Environment

- Application build pipeline may only perform builds of source code maintained in version control systems
- Application build pipeline prohibits alteration of DNS and network settings during build
- Application build pipeline prohibits alteration of certificate trust stores
- Application build pipeline enforces authentication and defaults to deny
- Application build pipeline enforces authorization and defaults to deny
- Application build pipeline requires separation of concerns for the modification of system settings
- Application build pipeline maintains a verifiable audit log of all system changes
- Application build pipeline maintains a verifiable audit log of all build job changes
- Application build pipeline has required maintenance cadence where the entire stack is updated, patched, and re-certified for use
- Compilers, version control clients, development utilities, and software development kits are analyzed and monitored for tampering, trojans, or malicious code
- All build-time manipulations to source or binaries are known and well defined
- Checksums of all first-party and third-party components are documented for every build
- Checksums of all components are accessible and delivered out-of-band whenever those components are packaged or distributed
- Unused direct and transitive components have been identified
- Unused direct and transitive components have been removed from the application

# OWASP SCVS V3 – Build Environment

- Application build pipeline maintains a verifiable audit log of:
  - build job changes
  - system changes
- The entire build stack is updated, patched, and re-certified for use
- Everything is analyzed and monitored for tampering, trojans, or malicious code
  - Compilers
  - Version control clients
  - Development utilities
  - Software development kits

# V3 – Build Environment

- Application build pipeline maintains a verifiable audit log of
  - build job changes
  - system changes
- Example in GitHub Actions: end-to-end traceability:
  - Workflow changes in commits
  - Execution environment in the logs
    - <https://github.com/devops-actions/issue-comment-tag/blob/main/.github/workflows/testing.yml>

# V3 – Build Environment – demo

The screenshot shows the GitHub Actions interface for the repository `devops-actions / issue-comment-tag`. The workflow `Bump node-fetch from 2.6.6 to 2.6.7 (#14) Build the action #57` is shown as successful. The `build` job is selected, displaying its logs. The logs indicate the runner version is `2.289.1` and the environment is `ubuntu-20.04` with `LTS` version `20220227.1`. A red box highlights the `Included Software` line, which points to a specific virtual environment image release.

```
1 Current runner version: '2.289.1'
2 ▼Operating System
3   Ubuntu
4   20.04.4
5   LTS
6 ▼Virtual Environment
7   Environment: ubuntu-20.04
8   Version: 20220227.1
9   Included Software: https://github.com/actions/virtual-environments/blob/ubuntu20/20220227.1/images/linux/Ubuntu2004-Readme.md
10  Image Release: https://github.com/actions/virtual-environments/releases/tag/ubuntu20%2F20220227.1
11 ▼Virtual Environment Provisioner
12  1.0.0.0-main-20220307-1
13 ▶GITHUB_TOKEN Permissions
14 Secret source: Actions
15 Prepare workflow directory
16 Prepare all required actions
17 Getting action download info
18 Download action repository 'actions/checkout@v2' (SHA:ec3a7ce113134d7a93b817d10a8272cb61118579)
```

# V3 – Build Environment – demo

<https://github.com/actions/virtual-environments/blob/ubuntu20/20220227.1/images/linux/Ubuntu2004-Readme.md>

actions / virtual-environments Public

Watch 199 Fork 2k Star 5.4k

Code Issues 36 Pull requests 5 Discussions Actions Projects Wiki Security

ubuntu20/20220... virtual-environments / images / linux / Ubuntu2004-Readme.md

459680 Updating readme file for ubuntu20 version 20220227.1 Latest commit 47f728c on Feb 28 History

4 contributors

390 lines (355 sloc) | 17.4 KB

Raw Blame

**Announcements**

[Ubuntu] Issue with libstdc++ cannot allocate memory in static TLS block

**Ubuntu 20.04.4 LTS**

- Linux kernel version: 5.11.0-1028-azure
- Image Version: 20220227.1

**Installed Software**

**Language and Runtime**

- Bash 5.0.17(1)-release
- Clang 10.0.0, 11.0.0, 12.0.0



# What happens during build?

- GitHub Actions: runner will download the actions
- Actions do their thing(s)?
  - <https://xpir.it/universe-2021>
- Harden runner: [link](#)

# OWASP SCVS V4 – Package management

- Package repository components have been published with multi-factor authentication
- Package repository notifies **publishers & users** of security issues
- Package repository requires code signing to publish packages to production repositories
- Package manager verifies the integrity of packages when they are retrieved:
  - From remote repository
  - From file system

# OWASP SCVS V4 – Package management

- Package repository components have been published with multi-factor authentication – **npm improvements with 2FA**
- Package repository notifies publishers & users of security issues
  - **Dependabot does this**
- Package repository requires code signing to publish packages to production repositories
  - Most package managers do not have support
  - **npm focusses on 2FA**
  - NuGet, Maven / Gradle / Ant (uploads through Maven central), RubyGems

# OWASP SCVS V5 – Component Analysis

- Component is analyzed using linters and/or static analysis tools
- Linting and/or static analysis is performed with every upgrade
- An automated process for:
  - Identifying all publicly disclosed vulnerabilities is used
  - Identifying confirmed dataflow exploitability is used
  - For identifying end-of-life / end-of-support components is used

# V5 – Component Analysis

- Component is analyzed using linters and/or static analysis tools
  - Trigger workflow on push
- Linting and/or static analysis is performed with every upgrade
  - Branch protection rules
- An automated process for:
  - Identifying all publicly disclosed vulnerabilities is used: Dependabot
  - Identifying confirmed dataflow exploitability is used: SAST / DAST
  - For identifying end-of-life / end-of-support components is used: ?

# OWASP SCVS V6 – Pedigree and Provenance

- Point of origin is verifiable for components
- Chain of custody is auditable for components

# V6 – Pedigree and Provenance

- Point of origin is verifiable for components
  - Dependabot dependency graph
- Chain of custody is auditable for components
  - Where did the component came from:
    - Pipeline link
    - Commit history

Example: <https://github.com/devops-actions/issue-comment-tag/releases>

# V6 – Pedigree and Provenance

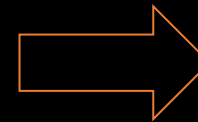
- Point of origin is verifiable for components
  - Cosign + sigstore = verification options for containers
- Chain of custody is auditable for components
  - SLSA is doing this for other binaries



# Supply Chain – Dependencies

- Libraries used by your application:

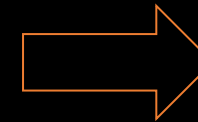
- Authentication
- Encryption
- Database connections



Package managers

- Tooling used for building your application:

- npm ci
- dotnet build
- pipelines



Build tools

# Topics

- Why:
  - Attack examples
- Protection / Maturity: frameworks
  - OWASP Software Component Verification Standard (SCVS)
  - Supply chain Levels for Software Artifacts (SLSA)

DWX

# DEVELOPER WEEK '23

## Protect yourself against supply chain attacks

Rob Bos

DevOps Consultant – Xpirit

The Netherlands

<https://devopsjournal.io>



<https://myoctocat.com>