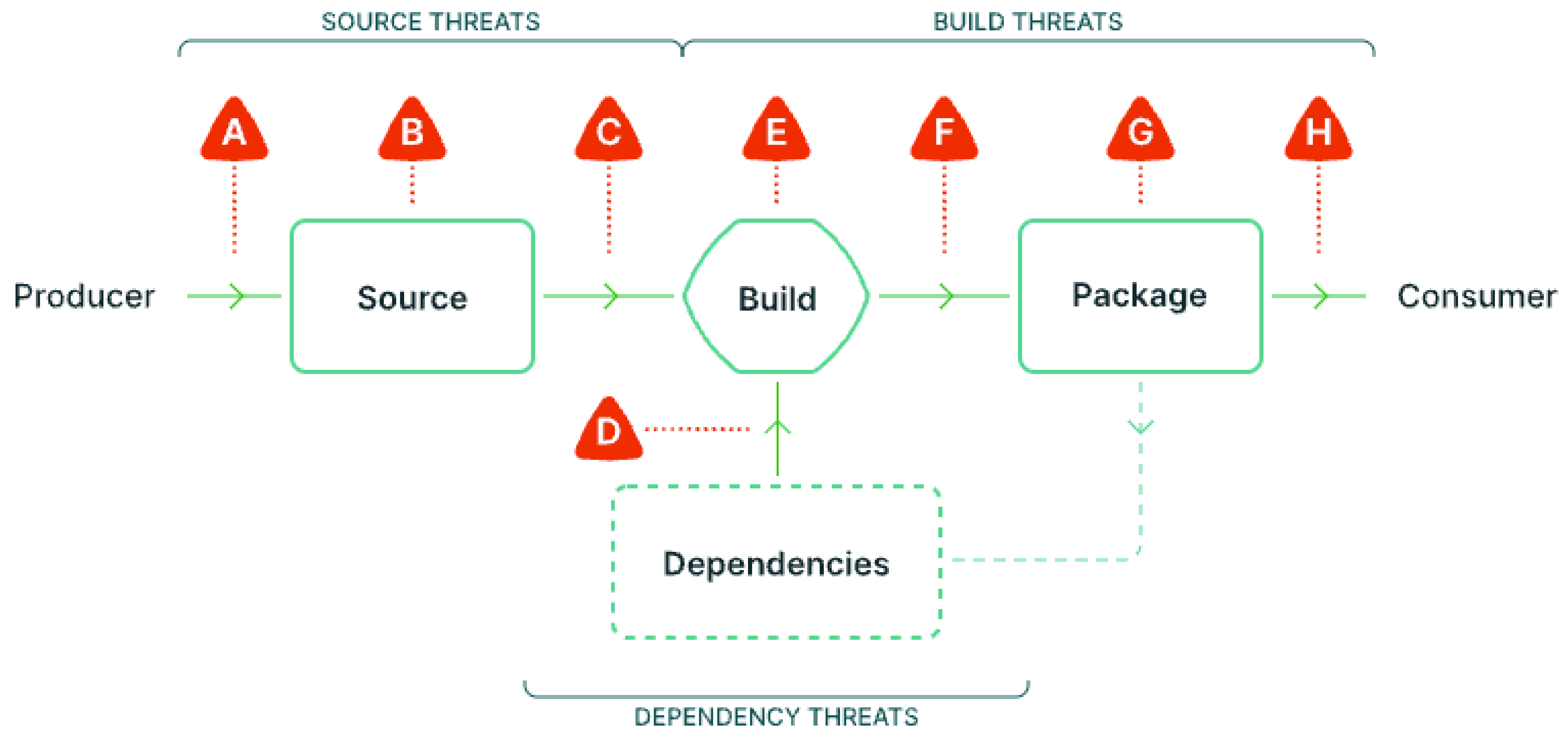
Dirty-Waters: Detecting Software Supply Chain Smells

Raphina Liu

25th, April, 2025

Software Supply Chain Threats



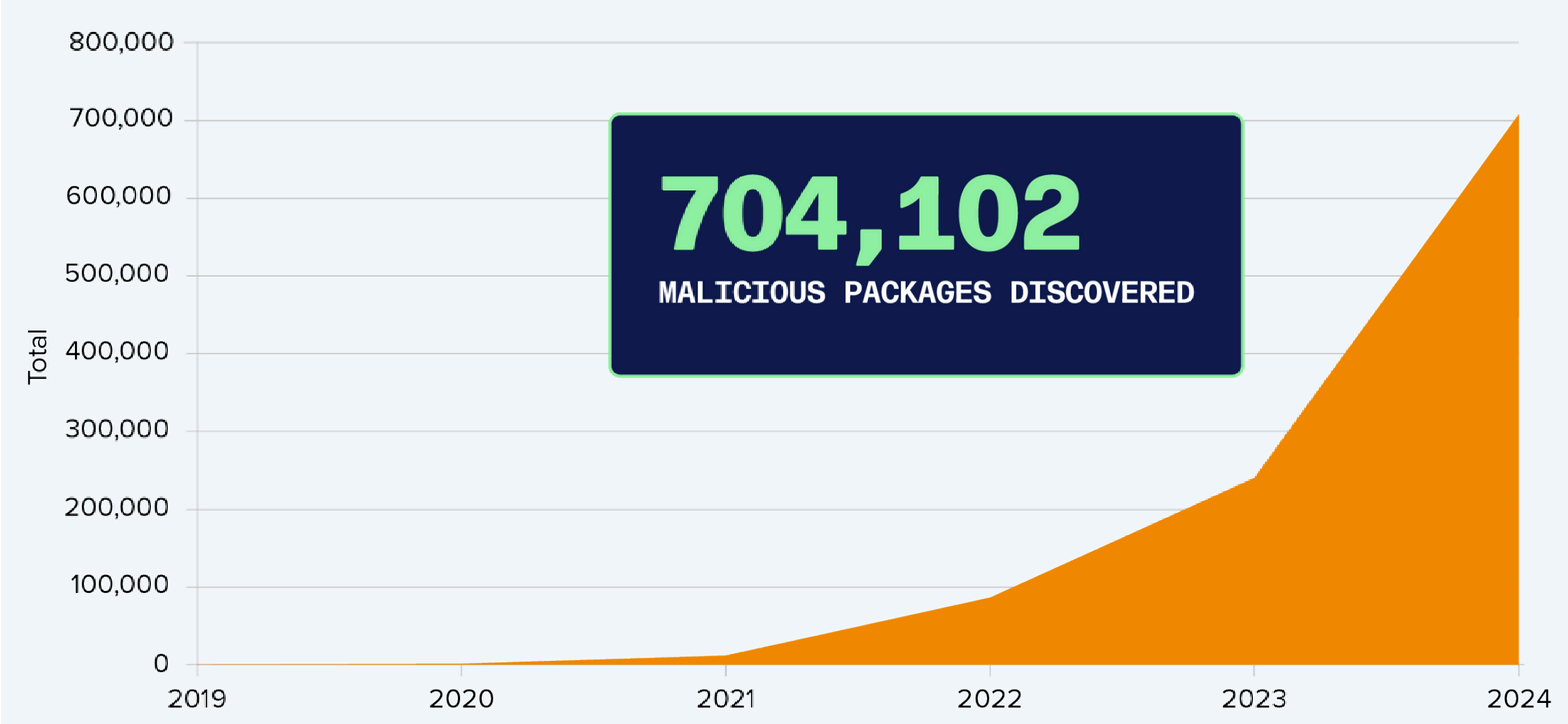
SOURCE THREATS	DEPENDENCY THREATS	BUILD THREATS
A Submit unauthorized change	D Use compromised dependency	E Compromise build process
B Compromise source repo		F Upload modified package
C Build from modified source		G Compromise package registry
		H Use compromised package

https://slsa.dev/spec/v1.0/terminology

Why is Important - Software Supply Chain Attacks

In 2023 alone, the number of malicious packages identified **doubled the total found in all previous years combined.** The number of attacks detected in the software supply chain **doubled again** in 2024.

Next Generation Software Supply Chain Attacks (2019–2024)



Malicious OSS packages discovered (2019-2024).

Motivation

Current Scenario

Using open-source dependencies is essential in modern software development.

Problem

This practice put significant trust in third-party code, while there is little support for developers to assess this trust.

Software Supply Chain Smells

Software Supply Chain Smells

Definition: **A software supply chain smell is a package which match specific patterns that indicate potential security issues, today or to come in the future.**

Software Supply Chain Smells Types

- S1. Dependencies with no/invalid link to source code repositories (High)
- S2. Dependencies with no tag/commit SHA for release (Medium)
- S3. Deprecated Dependencies (Medium)
- S4. Depends on a fork (Low)
- S5. Dependencies with no build attestation(Medium)
- S6. Dependencies without/with invalid code signature(Low)
- S7. Dependencies with alias(Low)

Software Supply Chain Smells Types

- **S1. No/invalid link to source code repositories** (Untraceable source code change)
- **S2. No tag/commit SHA for release** (Impossible to have reproducible builds)
- **S3. Deprecated Dependencies** (Potential exposure to vulnerabilities)
- **S4. Depends on a fork** (Potential repo confusion attack)
- **S5. Dependencies with no build attestation** (No source-to-artifact guarantee)
- **S6. Dependencies without/with invalid code signature** (No authenticity guarantee)
- **S7. Dependencies with alias** (Potential dependency alias confusion attacks)

Dirty-Waters

A novel tool for checking software supply chain smells that delivers human-readable reports about supply chain.

Dirty-Waters

Combines information from:

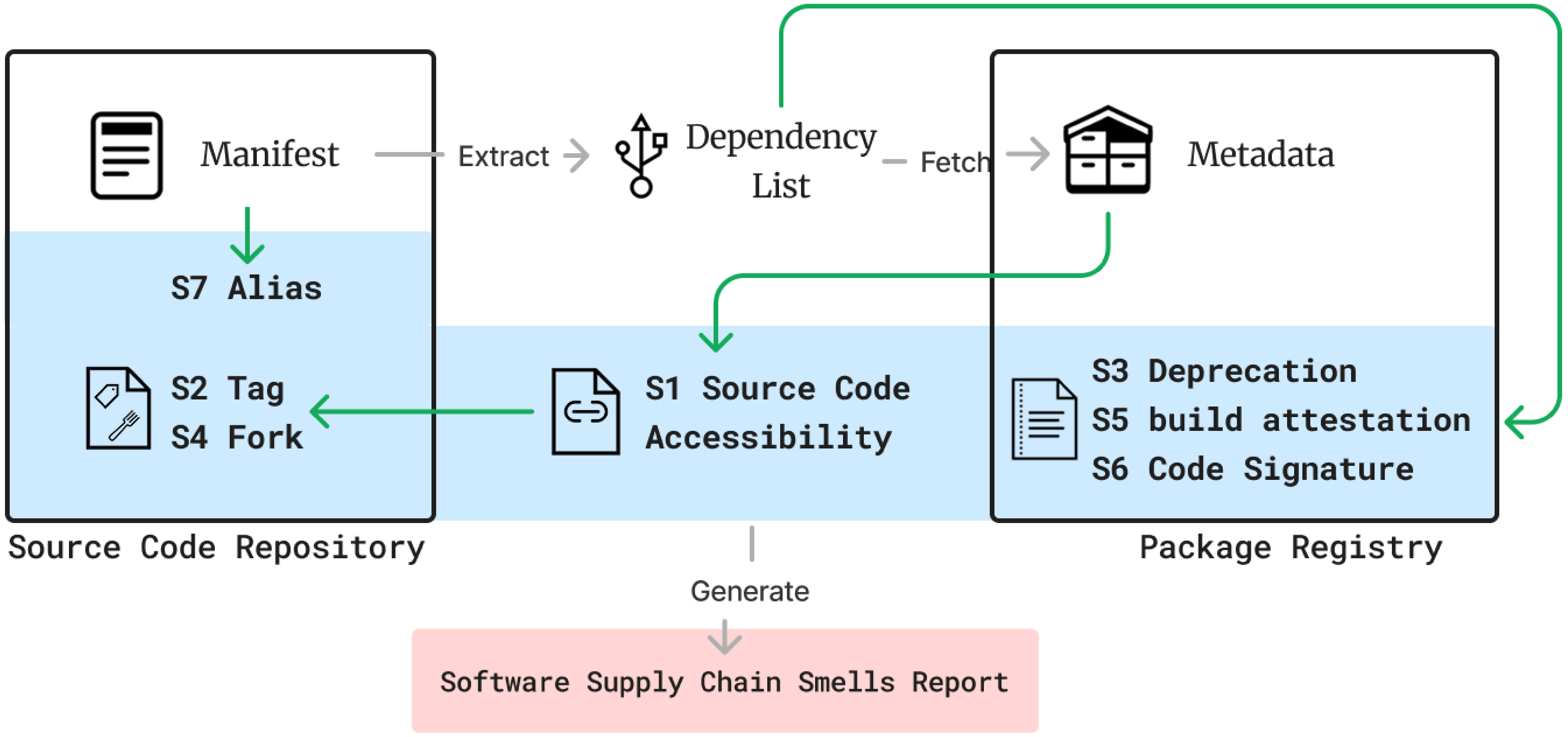
1) the local dependency file in the repository

2) the remote package registry

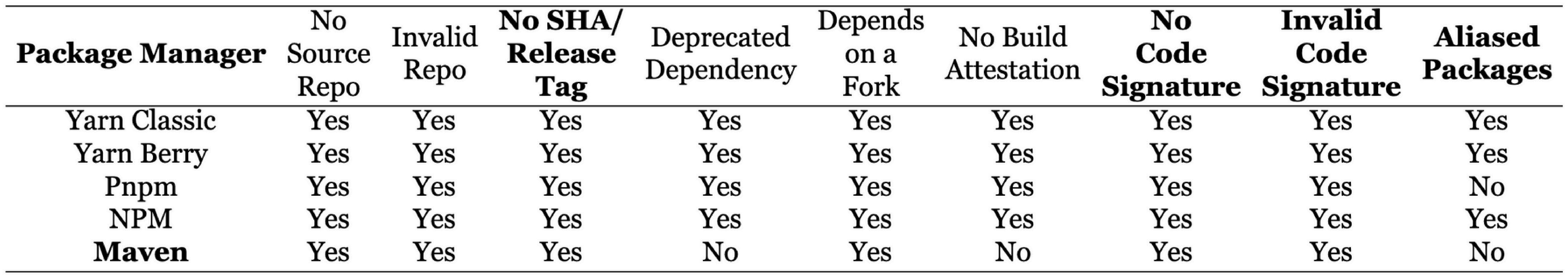
3) the source code repository

Dirty-Waters

Support Java and JavaScript (So far)



Dirty-Waters



Package Manager	No Source Repo	Invalid Repo	**No SHA/ Release Tag**	Deprecated Dependency	Depends on a Fork	No Build Attestation	**No Code Signature**	**Invalid Code Signature**	**Aliased Packages**
Yarn Classic	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Yarn Berry	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Pnpm	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
NPM	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Maven	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No

Table 3.1.1: SSC Smell Checks Currently Supported by Dirty-Waters.

Demo

Dirty-Waters

Dirty-Waters-Actions

Automation is a key for people to use the tools.
As such, we are creating a CI github action integration.

👍 to @Diogo

Dirty-Waters-Action

The Action Provides Some Options

gradual_report: 'Enable gradual report functionality'

fail_on_high_severity: 'Break CI if high severity issues are found'

x_to_fail: 'Percentage threshold for the number of high or medium severity issues to fail the CI'

allow_pr_comment: 'Post analysis results as a PR comment if CI breaks'

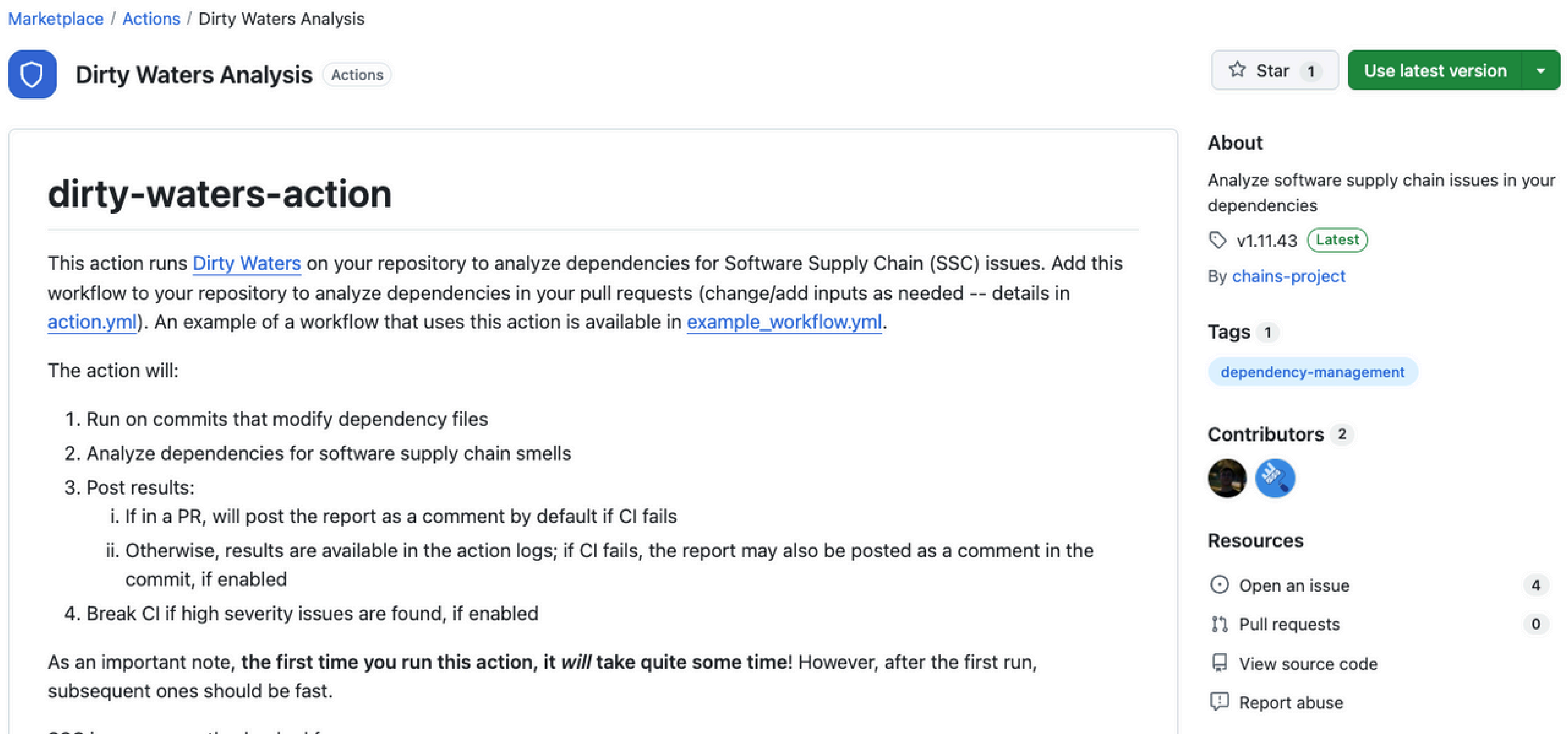
comment_on_commit: 'Post analysis results as a commit comment if CI breaks'

github_event_before: 'GitHub event before SHA, to retrieve the previous cache key'

ignore_cache: 'Ignore the repository cache for this run'

Demo

Dirty-Waters-Action



dirty-waters-action

This action runs Dirty Waters on your repository to analyze dependencies for Software Supply Chain (SSC) issues. Add this workflow to your repository to analyze dependencies in your pull requests (change/add inputs as needed -- details in action.yml). An example of a workflow that uses this action is available in example_workflow.yml.

The action will:

1. Run on commits that modify dependency files
2. Analyze dependencies for software supply chain smells
3. Post results:
   1. If in a PR, will post the report as a comment by default if CI fails
   2. Otherwise, results are available in the action logs; if CI fails, the report may also be posted as a comment in the commit, if enabled
4. Break CI if high severity issues are found, if enabled

As an important note, **the first time you run this action, it *will* take quite some time**! However, after the first run, subsequent ones should be fast.

About

Analyze software supply chain issues in your dependencies

v1.11.43 Latest

By chains-project

Tags 1

dependency-management

Contributors 2

Resources

Open an issue	4
Pull requests	0
View source code	
Report abuse	

Ongoing & Future Work

- Apply the tool to top Java&JS projects

- Add other software supply chain smells
- Add support to other languages
- Add support to other source code repositories

The Severity!

- S1. Dependencies with no/invalid link to source code repositories (High)
- S2. Dependencies with no tag/commit SHA for release (Medium)
- S3. Deprecated Dependencies (Medium)
- S4. Depends on a fork (Low)
- S5. Dependencies with no build attestation(Medium)
- S6. Dependencies without/with invalid code signature(Low)
- S7. Dependencies with alias(Low)

Help us define the severity categories we should give to each issue!

25, April, 2025

Dirty-Waters: Detecting Software Supply Chain Smells

https://github.com/chains-project/dirty-waters

The survey

Please help us define the importance categories we should give to each issue!

Contact

raphina@kth.se dgaspar@kth.se