

Vulnerable Components



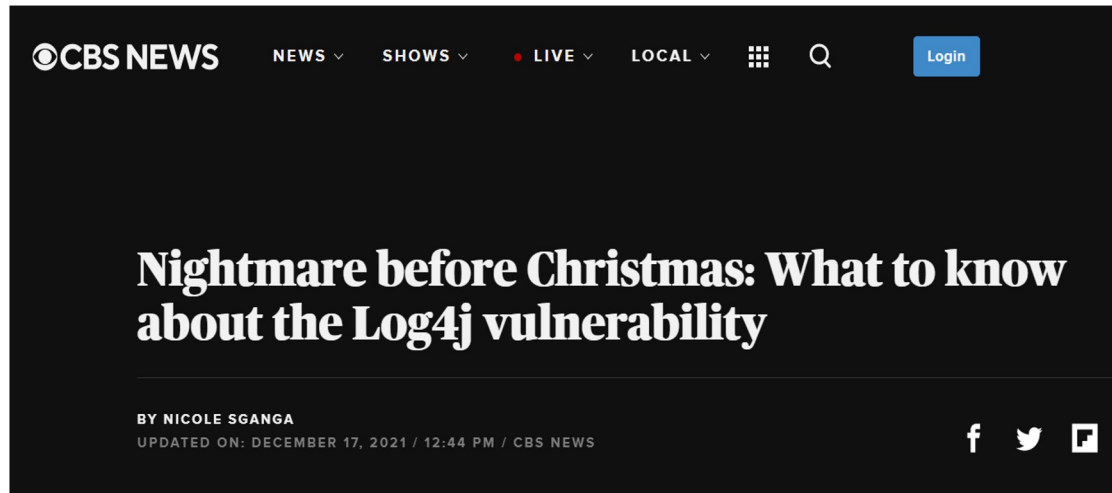
Laurie Williams

Laurie_williams@ncsu.edu

Agenda

- ▶ Overview
- ▶ Three supply chain-specific attacks
 - ▶ Typosquatting
 - ▶ Dependency confusion
 - ▶ New: manifest confusion
- ▶ Making good component choices
- ▶ Identifying vulnerabilities in components
 - ▶ Exercise
- ▶ Taxonomy of malicious commit attack vector
 - ▶ Exercise

Oops! Accidental dependency vulnerability







Code dependencies as an attack vector

Code dependencies as a weapon



FORTUNE RANKINGS ▾ MAGAZINE NEWSLETTERS PODCASTS MORE ▾ SEARCH SIGN IN [Subscribe Now](#)


Most Popular

-  Russia's ruble has almost totally recovered. Does that mean sanctions aren't working?
-  **PAID CONTENT**
Change your marketer experience and unleash growth for your business
FROM OPTIMIZEZY
-  Study finds ivermectin, the horse drug Joe Rogan championed as a COVID treatment, does nothing to cure the virus
-  Binance's founder, who accumulated as much wealth as Mark Zuckerberg in a quarter the time, explains how it feels to become unfathomably rich virtually overnight

INTERNATIONAL • UKRAINE INVASION

Russia's largest bank tells its clients to delay downloading software updates after 'protestware' attacks target Russian users

BY NICHOLAS GORDON
March 22, 2022 7:07 AM EDT

node-ipc 

11.1.0 • Public • Published 24 days ago





A6: Vulnerable and Outdated Components

- ▶ Components used in an application are outdated or have a vulnerability
- ▶ At the root of **software supply chain attacks** (think: Executive Order 14028).
- ▶ Notable CWEs:
 - ▶ 1104: Use of unmaintained third-party components
 - ▶ 1035: Using components with known vulnerabilities



Mindset shift required

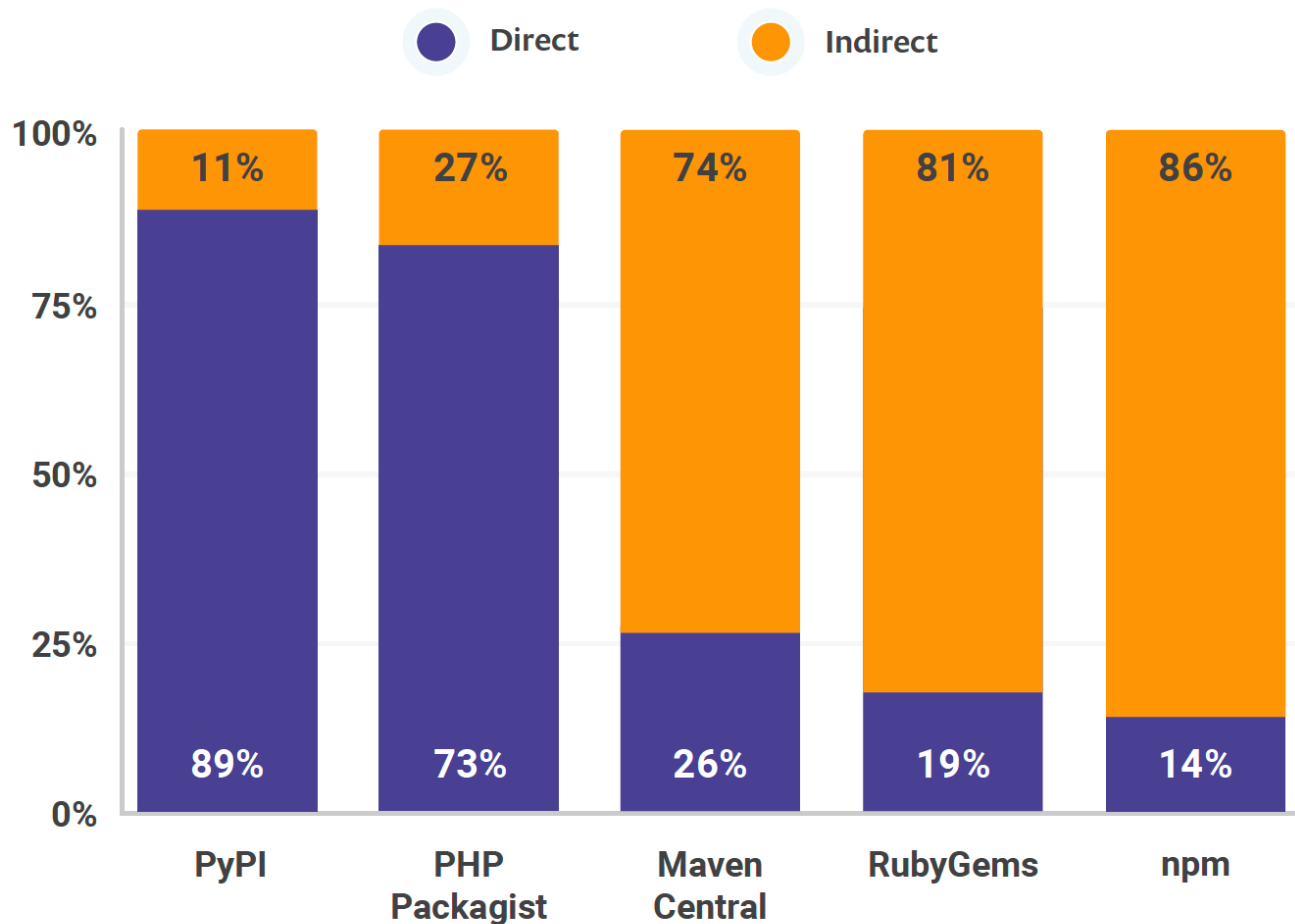
“Some might argue that it’s almost too easy to introduce a new dependency into your software systems. I’m definitely guilty of this in my previous life as an engineer. I remember pulling in random Python packages when building my own websites and not putting any thought into security. **It should be fine if so many other people are using the same package, right?”**

-- Kim Lewandowski, [Google Product Manager, founder Chainguard] and every other developer alive



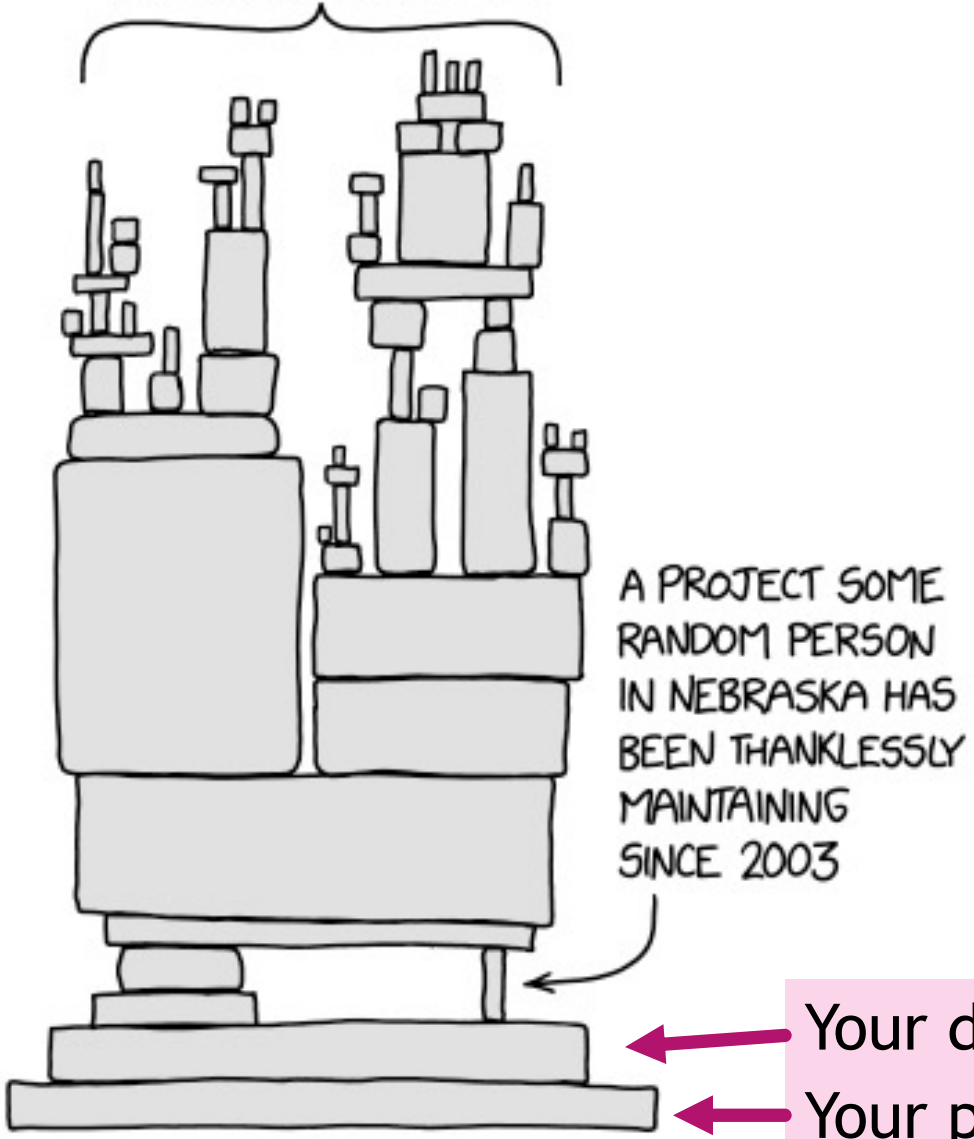
Transitive dependencies

Vulnerabilities from direct versus indirect dependencies



* Snyk: State of Open Source Dependencies 2020

ALL MODERN DIGITAL INFRASTRUCTURE



Which dependency has the potential to cause your project to crumble?

Ponder this ...



When you bring a third-party component into your project, it's like you are adding the developers of that component to your team.

Do you trust them?

How about the development teams for all the transitive dependencies?

Kind of mind blowing



Agenda

- ▶ Overview
- ▶ Three supply chain-specific attacks
 - ▶ Typosquatting
 - ▶ Dependency confusion
 - ▶ New: manifest confusion
- ▶ Making good component choices
- ▶ Identifying vulnerabilities in components
 - ▶ Exercise
- ▶ Taxonomy of malicious commit attack vector
 - ▶ Exercise

TYPOSQUATTING

AKA URL Hijacking — the practice of registering domains of known brands with the intent of tricking users into believing they are legitimate sites

COMMON TECHNIQUES

DROPPING THE DOT
AFTER 'WWW'

wwwaa.com

DROPPING ONE LETTER

apple.om

SWITCHING TWO
LETTERS

faecbook.com

DOUBLING CHARACTERS

twiitter.com

USING SIMILAR
LOOKING CHARACTERS

google.com (l vs l)

PRESSING A WRONG
KEY

costko.com

Source: <https://www.anomali.com/resources/infographics/typosquatting-more-than-just-a-typo>



“brandjacking”

often combined with a malicious payload that executes immediately using the built-in functionality of the developer’s build tool.

```
rust_decimle
rust_decimal
rsut_decimal
```

Recent Typosquatting
Attacks

Sonatype 2021 Supply Chain report

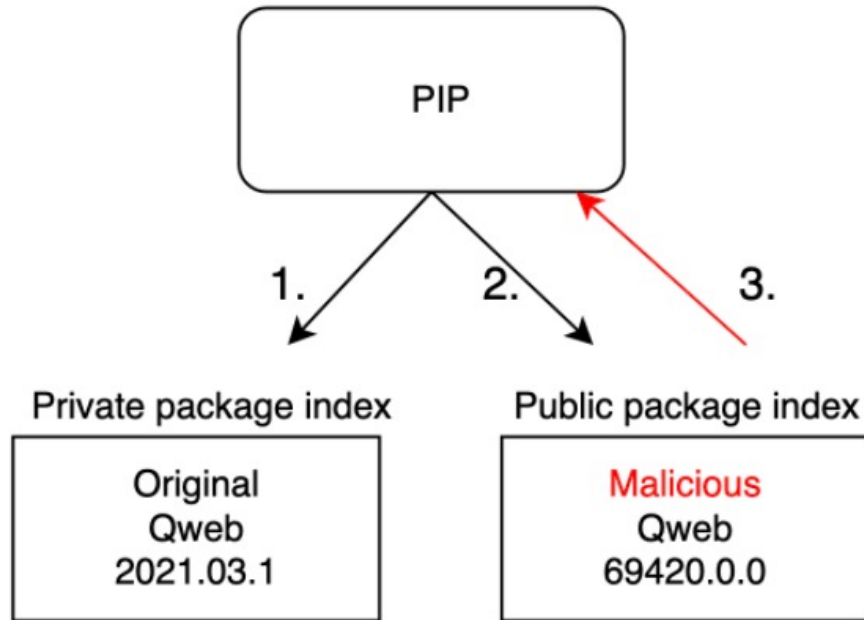
Dependency Confusion

Most common attack
2021
49% of organizations are
vulnerable

- ▶ Early build step - download source and dependencies from approved source and artifact repos
 - ▶ Anyone can freely upload code
- ▶ Install dependencies: Node has npm; Python's pip uses PyPi; RubyGems

```
pip install package_name
```
- ▶ **Typosquatting** - leverages typo'd versions of popular package names
- ▶ **Dependency confusion**: a software installer is tricked into pulling a malicious code file from a public repository instead of the intended file from an internal repository

Dependency Confusion - 2



- ▶ Public package contains higher version compared to private package
- ▶ If package indexing not done properly, it will automatically pull the higher version from the public registry


Dependency confusion - 3

- ▶ Finding private/internal packages (NPM)
 - ▶ Look at the package.json file

```
45ce538c8d flipper / desktop / eslint-plugin-flipper / package.json  
dependabot Bump @typescript-eslint/parser from 4.28.5 to 4.29.1 in /desktop (#2700) ...  
4 contributors  
41 lines (41 sloc) | 1017 Bytes  
1 {  
2   "name": "eslint-plugin-flipper",  
3   "version": "0.0.0",  
4   "private": true,  
5   "description": "Custom ESLint rules for Flipper",  
6   "repository": "facebook/flipper",  
7   "main": "lib/index.js",  
8   "flipperBundlerEntry": "src",  
9   "types": "lib/index.d.ts",  
10  "license": "MIT",  
11  "bugs": "https://github.com/facebook/flipper/issues",  
12  "dependencies": {  
13    "@typescript-eslint/experimental-utils": "^4.28.5",  
14    "fs-extra": "^10.0.0"  
15  },  
16  "devDependencies": {  
17    "@types/jest": "26.0.24"
```


Pinning dependencies

- ▶ Specify an exact version, under version control
- ▶ Example:
 - ▶ Npm lockfiles that list fix versions for all dependencies (direct and transitive)

 **snyk** Advisor

pin-dependencies-checker

v1.0.6

Package Health Score

48 / 100

POPULARITY	SMALL
MAINTENANCE	INACTIVE
SECURITY	NO KNOWN SECURITY ISSUES
COMMUNITY	LIMITED

Manifest confusion (npm)

- ▶ occurs when there is an inconsistency between a package's manifest information presented on the npm registry and the actual 'package.json' file in the tarball of the published npm package used when the package is installed.

The screenshot shows the npm registry page for the package `darcyclarke-manifest-pkg`. The package name, version `2.1.15`, and the `0 Dependencies` badge are circled in red. The `package.json` content is also circled, showing a version of `3.0.0` and a license of `ISC`, which differs from the registry information. The registry shows a license of `none`, which is also circled in red.

```
1 {
2   "name": "express",
3   "version": "3.0.0",
4   "main": "index.js",
5   "scripts": {
6     "install": "touch ./bad-pkg-write && echo \"bad pkg exec!\\n\""
7   },
8   "license": "ISC",
9   "dependencies": {
10    "sleepover": "*"
11  }
12 }
```

Install

```
> npm i darcyclarke-manifest-pkg
```

Version	License
2.1.15	none

Unpacked Size	Total Files
248 B	2

Last publish
an hour ago

Collaborators

RunKit

Risks of Manifest Confusion

- ▶ installation of unknown dependencies that won't show upon security tools
 - ▶ execution of unknown scripts, and
 - ▶ potentially also downgrade attacks
-
- ▶ Needed action:
 - ▶ Developers should manually read the `package.json` to determine version numbers, what dependencies will be installed, and what scripts will be executed
 - ▶ Tools emerging

Agenda

- ▶ Overview
- ▶ Three supply chain-specific attacks
 - ▶ Typosquatting
 - ▶ Dependency confusion
 - ▶ New: manifest confusion
- ▶ Making good component choices
- ▶ Identifying vulnerabilities in components
 - ▶ Exercise
- ▶ Taxonomy of malicious commit attack vector
 - ▶ Exercise

Potential weak links

... may increase risk of supply chain attack



People are often the weak link

(npm) Weak link signals

Expired Maintainer Domain

2,842 maintainers' email domains are expired.

Install Scripts

93.9% of malicious packages use install scripts.

Unmaintained Packages

58.6% packages & **44.3%** maintainers are inactive.

Too many Maintainers

421 popular packages have **14,566** maintainers.

Too many Contributors

45 maintainers supervise **2,780** contributors in **23** popular packages.

Overloaded Maintainers

4,743 maintainers own **52.4%** packages in npm.



OpenSSF Scorecard

OPEN SOURCE SECURITY FOUNDATION

```
Nusrat@MacBook-Pro-6 ~ % scorecard --repo=github.com/mochajs/mocha --checks Dangerous-Workflow --show-detail
Starting [Dangerous-Workflow]
Finished [Dangerous-Workflow]

RESULTS
-----
Aggregate score: 0.0 / 10

Check scores:
-----
```

SCORE	NAME	REASON	DETAILS	HTTP
0 / 10	Dangerous-Workflow	dangerous workflow patterns detected	Warn: untrusted code checkout '\${{ github.event.pull_request.head.sha }}': .github/workflows/browser-test.yml:18 Warn: secret accessible to pull requests '\${{ secrets.SAUCE_USERNAME }}': .github/workflows/browser-test.yml:33 Warn: secret accessible to pull requests '\${{ secrets.SAUCE_ACCESS_KEY }}': .github/workflows/browser-test.yml:34 Warn: secret accessible to pull requests '\${{ secrets.GITHUB_TOKEN }}': .github/workflows/browser-test.yml:39 Warn: secret accessible to pull requests '\${{ secrets.SAUCE_USERNAME }}': .github/workflows/mocha.yml:160 Warn: secret accessible to pull requests '\${{ secrets.SAUCE_ACCESS_KEY }}': .github/workflows/mocha.yml:161 Warn: secret accessible to pull requests '\${{ secrets.GITHUB_TOKEN }}': .github/workflows/mocha.yml:132	http

SCORE	NAME	REASON	HTTP
10 / 10	Binary-Artifacts	no binaries found in the repo	http
0 / 10	Branch-Protection	branch protection not enabled on development/release branches	http
10 / 10	CI-Tests	28 out of 28 merged PRs checked by a CI test -- score normalized to 10	http
0 / 10	CII-Best-Practices	no badge detected	http
9 / 10	Code-Review	GitHub code reviews found for 28 commits out of the last 30 -- score normalized to 9	http
10 / 10	Contributors	65 different companies found -- score normalized to 10	http
0 / 10	Dangerous-Workflow	dangerous workflow patterns detected	http
10 / 10	Dependency-Update-Tool	update tool detected	http
0 / 10	Fuzzing	project is not fuzzed	http
10 / 10	License	license file detected	http
10 / 10	Maintained	30 commit(s) out of 30 and 26 issue activity out of 30 found in the last 90 days -- score normalized to 10	http
?	Packaging	no published package detected	http
6 / 10	Pinned-Dependencies	dependency not pinned by hash detected -- score normalized to 6	http
0 / 10	SAST	SAST tool is not run on all commits -- score normalized to 0	http
10 / 10	Security-Policy	security policy file detected	http
?	Signed-Releases	no releases found	http
0 / 10	Token-Permissions	non read-only tokens detected in GitHub workflows	http
10 / 10	Vulnerabilities	no vulnerabilities detected	http

Dep.dev

io.fabric8:kubernetes-model-core 6.1.1

[Overview](#) [Dependencies](#) [Dependents](#) [Compare](#) [Versions](#)

Security Advisories

No advisories detected

Licenses

[Learn more about license information.](#)

5

LICENSES

Apache-2.0

DEPENDENCY LICENSES



Dependencies

39



[View all dependencies](#)

Dependents

181



[View dependents](#)

Published

September 1, 2022

Description

Java client for Kubernetes and OpenShift

Links

ORIGIN

<https://search.maven.org/artifact/io.fabric8/kubernetes-model-core/6.1.1/jar>

HOMEPAGE

<http://fabric8.io/>

REPO

<https://github.com/fabric8io/kubernetes-client>

Projects

[fabric8io/kubernetes-client](#)

GitHub

Java client for Kubernetes & OpenShift

[1k forks](#)

[3k stars](#)

OpenSSF scorecard

The [Open Source Security Foundation](#) is a cross-industry collaboration to improve the security of open source software (OSS). The [Scorecard](#) provides security health metrics for open source projects.

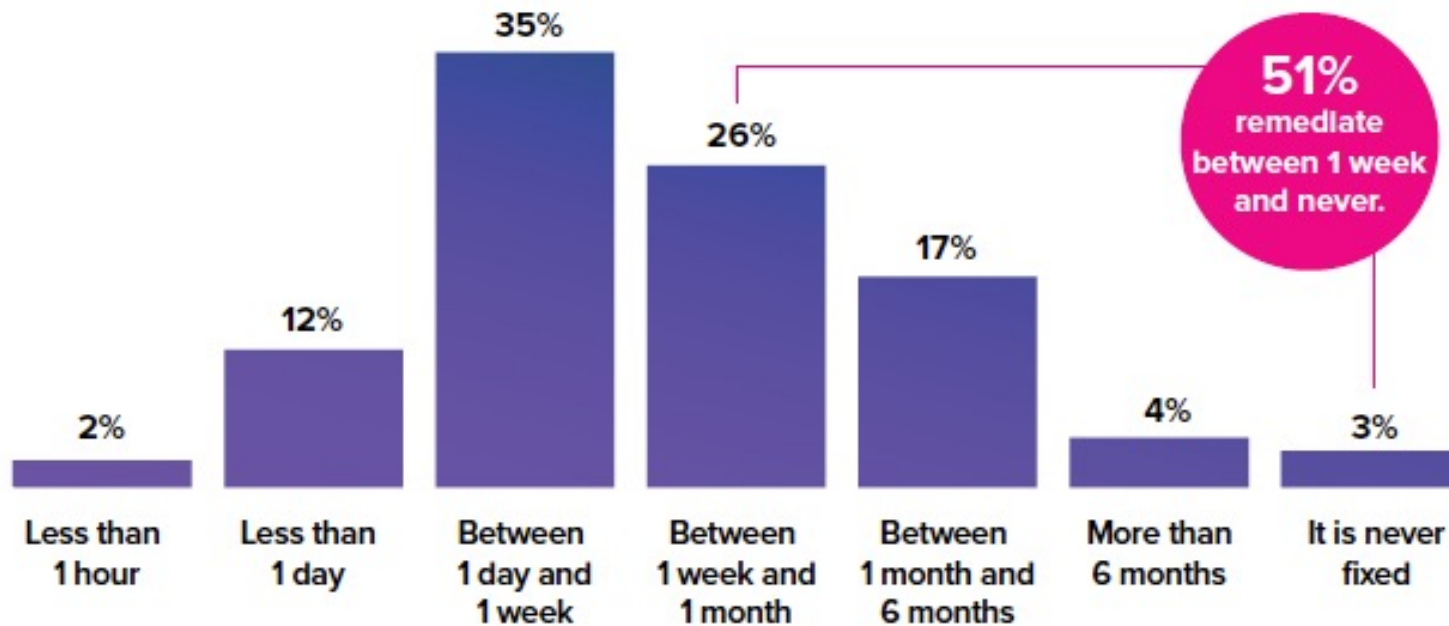
[View information about checks and how to fix failures.](#)

SCORE

7/10

Mean Time to Update (MTTU)

Time to Remediate Known OSS Vulnerabilities After Detection



What other weak links can you think of?

- ▶ If you want to make a good component choice, what should be consider?

Agenda

- ▶ Overview
- ▶ Three supply chain-specific attacks
 - ▶ Typosquatting
 - ▶ Dependency confusion
 - ▶ New: manifest confusion
- ▶ Making good component choices
- ▶ Identifying vulnerabilities in components
 - ▶ Exercise
- ▶ Taxonomy of malicious commit attack vector
 - ▶ Exercise

Software Component Analysis (SCA) Tools



username

We found a vulnerable dependency in a repository you have security alert access to.

reponame

Known **high severity** security vulnerability detected in `swagger-ui < 3.23.11` defined in [package-lock.json](#).
[package-lock.json](#) update suggested:
`swagger-ui ~> 3.23.11`.

Always verify the validity and compatibility of suggestions with your codebase.

Review vulnerable dependency

The screenshot shows a GitHub Security alert for the `waitress` dependency. The alert is titled "waitress" and is marked as "Open". It indicates that a fix has already been started but there is no bandwidth to fix it. The alert details a "Bump waitress from 1.3.0 to 1.4.3" and notes that 5 vulnerabilities were found in `requirements.txt`. The remediation section suggests upgrading to version 1.4.2 or later, with an example: `waitress>=1.4.2`. The details section shows the vulnerability ID "GHSA-968f-66r5-5v74" with a "high severity" label, and lists vulnerable versions as `< 1.4.2` and the patched version as `1.4.2`.



Software Component Analysis (SCA)

DependencyCheck Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
153	138	0

Summary

Total	High Priority	Normal Priority	Low Priority
153	24	111	18

Details

Files	Categories	Types	Warnings	Details	New	High	Normal	Low
-------	-------------------	-------	----------	---------	-----	------	--------	-----

Category	Total	Distribution
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer	5	
CWE-134 Uncontrolled Format String	1	
CWE-189 Numeric Errors	2	
CWE-20 Improper Input Validation	7	
CWE-200 Information Exposure	5	
CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	4	
CWE-264 Permissions, Privileges, and Access Controls	4	
CWE-287 Improper Authentication	2	
CWE-310 Cryptographic Issues	2	
CWE-399 Resource Management Errors	7	
CWE-59 Improper Link Resolution Before File Access ('Link Following')	4	
CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	14	
CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	2	
CWE-94 Improper Control of Generation of Code ('Code Injection')	10	
Total	153	

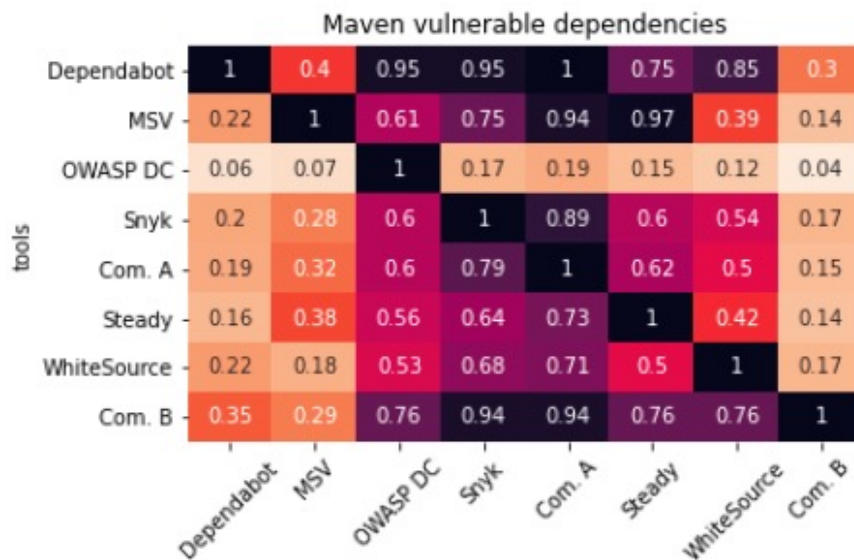
Table 2: Vulnerable Dependencies for Maven (Java) projects

Tool	Alert	Unique Dependency	Unique Package	Unique Vulnerability	CVE	Non-CVE	Scan Time (Minutes)
	Total (Median per project)						
OWASP DC	12,466 (254.0)	332 (38.0)	149 (36.0)	313 (117.0)	289	24	14.4
Snyk	4,902 (66.0)	96 (6.0)	46 (6.0)	189 (23.0)	178	11	15.1
Dependabot	136 (0.0)	20 (0.0)	11 (0.0)	61 (0.0)	61	0	NA
MSV	3,197 (58.0)	36 (12.0)	14 (12.0)	36 (22.0)	36	0	3.4
Steady	2,489 (51.0)	91 (20.0)	39 (19.0)	97 (41.0)	89	8	385.0
WhiteSource	434 (0.0)	76 (0.0)	44 (0.0)	146 (0.0)	127	19	NA
Commercial A	2,998 (70.0)	107 (24.0)	53 (24.0)	208 (70.0)	187	21	NA
Commercial B	205	35	35	127	127	0	NA

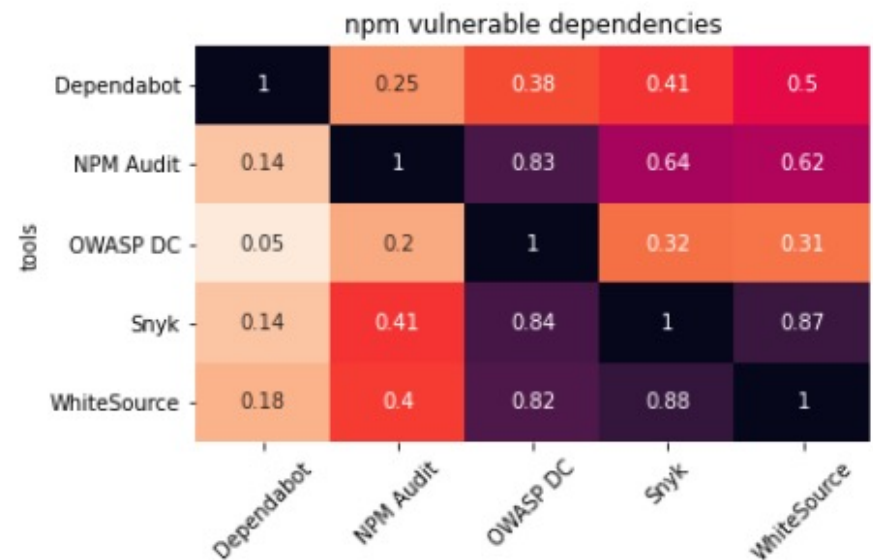
Table 3: Vulnerable Dependencies for npm (JavaScript) projects

Tool	Alert	Unique Dependency Path	Unique Dependency	Unique Package	Unique Vulnerability	CVE	Non- CVE	Scan Time (Minutes)
	Total (Median per project)							
OWASP DC	1,379 (208.0)	498 (72.0)	239 (71.0)	160 (57.0)	234 (71.0)	78	156	4.4
Snyk	2,210 (135.0)	1,004 (44.0)	90 (20.0)	54 (17.0)	121 (26.0)	79	42	1.0
Dependabot	97 (8.0)	NA	32 (1.0)	30 (1.0)	45 (4.0)	29	16	NA
npm audit	1,266 (37.0)	852 (28.0)	58 (12.0)	45 (12.0)	62 (16.0)	31	31	0.1
WhiteSource	205 (32.0)	205 (32.0)	89 (14.0)	55 (9.0)	96 (18.0)	58	38	NA

Overlap in finding same vulnerable components



(a) Overlap ratios for Maven vulnerable dependencies



(b) Overlap ratios for npm vulnerable dependencies

OWASP Juice Shop

- ▶ modern and sophisticated insecure web application

The screenshot displays the OWASP Juice Shop website interface. The header includes the site logo, name, search icon, account icon, and language selector (EN). The main content area is titled "All Products" and features a grid of product cards. Each card contains an illustration, the product name, volume, and price. The "Best Juice Shop Salesman Artwork" card is marked as "Sold Out". A cookie notice is visible at the bottom right.

Product Name	Volume	Price
Apple Juice	1000ml	1.99€
Apple Pomace		0.89€
Banana Juice	1000ml	1.99€
Best Juice Shop Salesman Artwork		5000€
Carrot Juice	1000ml	2.99€
Eggfruit Juice	500ml	8.99€
Fruit Press		89.99€
Green Smoothie		1.99€

This website uses fruit cookies to ensure you get the juiciest tracking experience. [But me wait!](#)

[Me want it!](#)

OWASP Dependency Check

- ▶ Look at JuiceShop report: <https://tinyurl.com/3yev9jt2>
- ▶ Pick a high severity/high confidence vulnerability. Go to the National Vulnerability Database (NVD) Common Vulnerability Enumeration (CVE) and summarize the vulnerability



Project: WolfpackShop

Scan information ([show all](#)):

- dependency-check version: 8.3.1
- Report Generated On: Sat, 8 Jul 2023 08:08:01 -0400
- Dependencies Scanned: 25057 (19009 unique)
- Vulnerable Dependencies: 42
- Vulnerabilities Found: 90
- Vulnerabilities Suppressed: 0
- ...

Analysis Exceptions

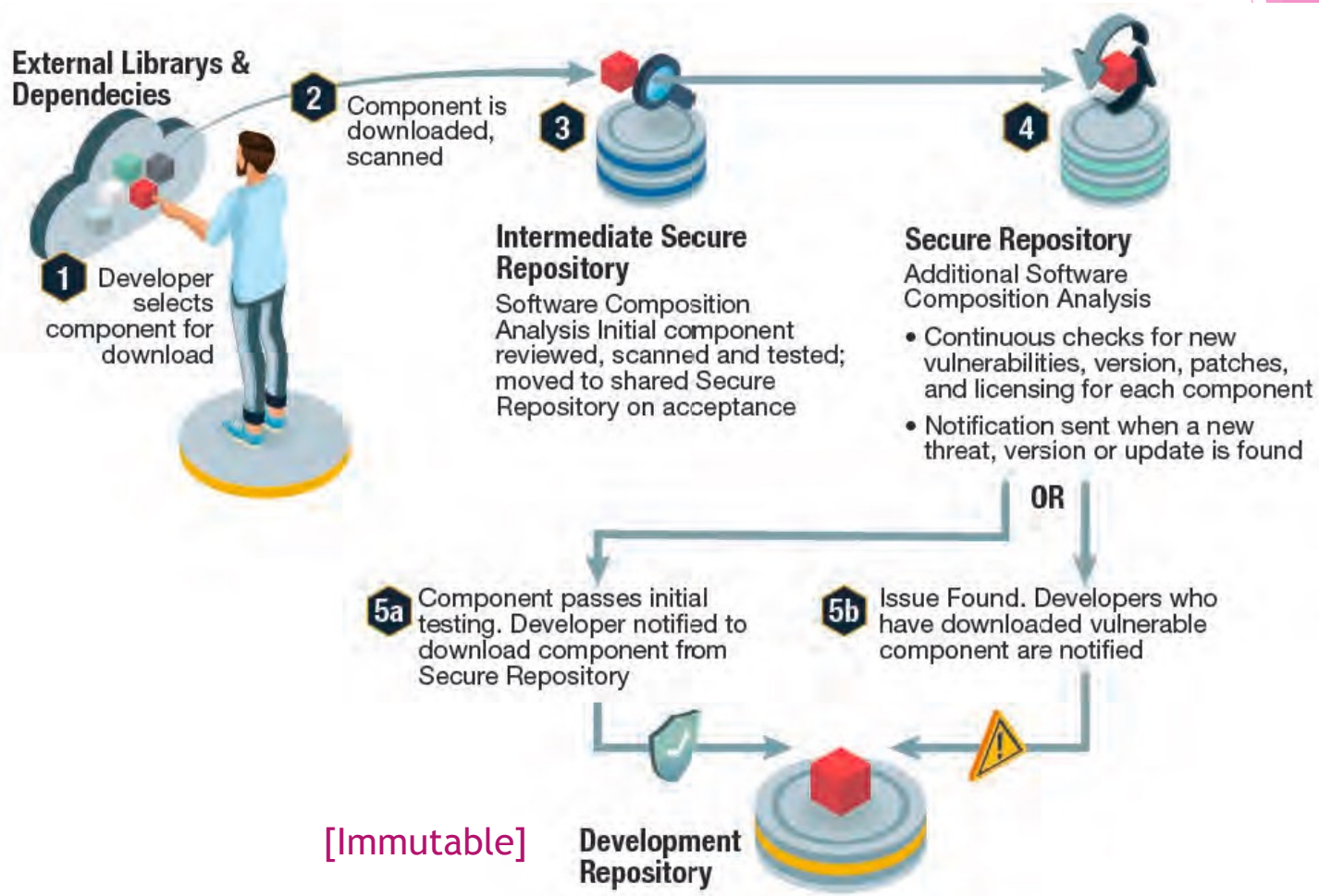
Unable to read yarn audit output.

Summary

Display: [Showing Vulnerable Dependencies \(click to show all\)](#)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
bench.js		pkg:javascript/underscore.js@1.7.0	HIGH	1		3
crypto-js:3.3.0	cpe:2.3:a:crypto-js_project:crypto-js:3.3.0:*:*:*:*:*	pkg:npm/crypto-js@3.3.0	MEDIUM	1	Highest	8
dottie:2.0.3	cpe:2.3:a:dottie_project:dottie:2.0.3:*:*:*:*:*	pkg:npm/dottie@2.0.3	HIGH	2	Highest	6
ecstatic:3.3.2	cpe:2.3:a:ecstatic_project:ecstatic:3.3.2:*:*:*:*:*	pkg:npm/ecstatic@3.3.2	HIGH	1	Highest	7
engine.io:4.1.2	cpe:2.3:a:socket:engine.io:4.1.2:*:*:*:*:*	pkg:npm/engine.io@4.1.2	MEDIUM	1	Highest	7
express-jwt:0.1.3	cpe:2.3:a:auth0:express-jwt:0.1.3:*:*:*:*:*	pkg:npm/express-jwt@0.1.3	CRITICAL	2	Highest	9
file-type:11.1.0	cpe:2.3:a:file-type_project:file-type:11.1.0:*:*:*:*:*	pkg:npm/file-type@11.1.0	MEDIUM	1	Highest	8

Secure Repository Process Flow



Updating vulnerable dependencies



Agenda

- ▶ Overview
- ▶ Three supply chain-specific attacks
 - ▶ Typosquatting
 - ▶ Dependency confusion
 - ▶ New: manifest confusion
- ▶ Making good component choices
- ▶ Identifying vulnerabilities in components
 - ▶ Exercise
- ▶ Taxonomy of malicious commit attack vector
 - ▶ Exercise

SoK: Taxonomy of Attacks on Open-Source Software Supply Chains

Piergiorgio Ladisa

SAP Security Research, Université de Rennes 1
piergiorgio.ladisa@sap.com,
piergiorgio.ladisa@irisa.fr

Henrik Plate

SAP Security Research*
henrik@endor.ai
*now at Endor Labs

Matias Martinez

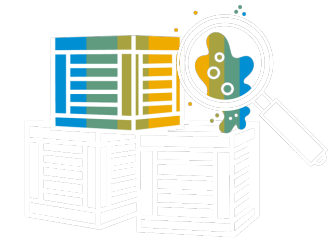
Université Polytechnique Hauts-de-France*
matias.martinez@upc.edu
*now at Universitat Politècnica de Catalunya-BarcelonaTech

Olivier Barais

Université de Rennes 1, INRIA/IRISA
olivier.barais@irisa.fr

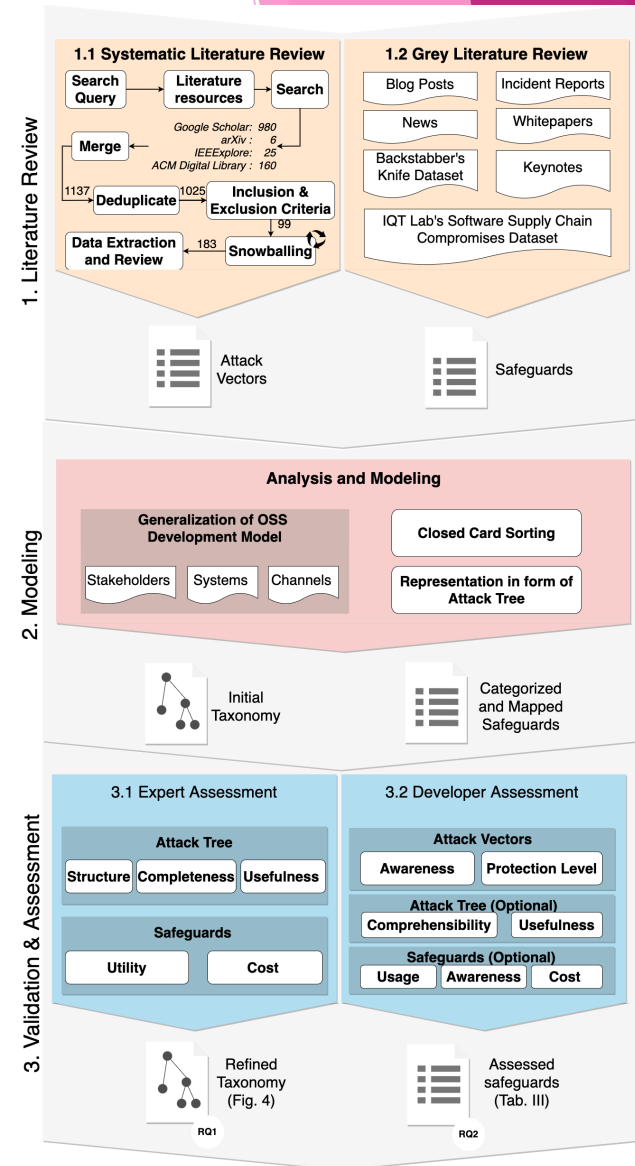
44th IEEE Symposium on Security and Privacy
May 22-25, 2023 - San Francisco, CA

Public



Methodology

- Systematic Literature Review and Grey Literature Review to collect
- All known attack vectors
- Associated safeguards
- Model the attack vectors in an attack tree and map the safeguards to each vector
- Conduct two user surveys to assess both the taxonomy and utility/cost of safeguards
- 17 experts
- 134 developers



Results: Taxonomy of Open-Source Software Supply Chain Attacks

The proposed taxonomy :

- Attacker's perspective
- Positively assessed by 17 experts
- The taxonomy, safeguards and references can be explored online using the [Risk Explorer for Software Supply Chain](#) [1]



117

Unique attack vectors



33

Unique high-level safeguards



370

Scientific and grey literature references

[1] <https://sap.github.io/risk-explorer-for-software-supply-chains/>

Risk Explorer for Software Supply Chain



Check it out!

SEARCHBARS **LEGEND**

Attack Vectors
Select...

Safeguards
Reproducible builds x

AV-300 Inject into Sources of Legitimate Package
This attack vector aims at injecting malicious code into the versioning control system of a legitimate project. Consequently, every user or system building the software from the sources will be affected by the attack, and - as far as this is the attackers intention - produce a binary package including the malicious code. Of course, this also comprises the project's standard build system producing the binary that will be distributed through the project's standard distribution mechanism, e.g., package repositories like Maven Central or npm. In other words, for the attacker, this technique has the

References

1. [Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks \(DIMVA\)](#) peer-reviewed
2. [Anomalous: Automated Detection of Anomalous and Potentially Malicious Commits on GitHub \(ICSE-SEIP\)](#) peer-reviewed
3. [Windows Malware Binaries in C/C++ GitHub Repositories: Prevalence and Lessons Learned](#) peer-reviewed
4. [In-toto: Practical Software Supply Chain Security](#)
5. [Vulnerabilities in Continuous Delivery Pipelines? & Pace Shur...](#) peer-reviewed

Mapped safeguards

- [SG-004] Manual Source Code Review

Safeguards inherited from [AV-001] Subvert Legitimate Package

- [SG-009] Remove un-used Dependencies
- [SG-029] Version Pinning

Safeguards inherited from [AV-000] Conduct Open-Source Supply Chain Attack

- [SG-001] Software Bill of Materials (SBOM)

Available online and open-source: <https://sap.github.io/risk-explorer-for-software-supply-chains/>

Exercise

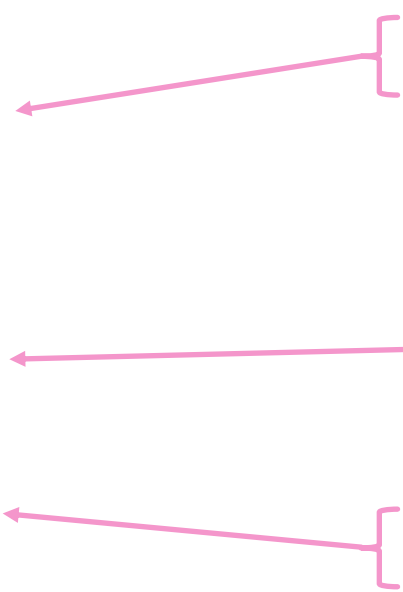


- ▶ Go to the Risk Explorer
 - ▶ <https://sap.github.io/risk-explorer-for-software-supply-chains>
 - ▶ <https://tinyurl.com/ymz63597>
- ▶ Go three levels deep in the tree
 - ▶ Summarize the attack and possible safeguards

Cost matters: Safeguards Utility & Cost Assessment

Safeguard	Experts					Developers		
	Utility		Cost		Mean U/C	Usage	Cost	
	Mean	Median	Mean	Median			Mean	Median
Protect production branch	4.2	4.0	2.0	2.0	2.10	Y=N	1.8	2.0
Remove un-used dependencies	4.3	5.0	2.1	2.0	2.05	Y=N	2.0	2.0
Version pinning [74] [72]	3.7	3.0	2.2	2.0	1.68	Y=N	2.1	2.0
Dependency resolution rules	4.1	4.0	2.6	3.0	1.58	Y=N	2.7	3.0
User account management	3.9	4.0	2.6	3.0	1.50	Y=N	2.3	2.5
Preventive squatting the released packages	3.1	3.0	2.9	3.0	1.07	Y=N	3.8	3.5
Runtime Application Self-Protection	3.7	4.0	4.2	4.0	0.88	Y=N	3.8	4.0
Manual source code review	4.1	4.0	4.8	5.0	0.85	Y=N	4.4	5.0
Build dependencies from sources	3.0	3.0	4.1	4.0	0.73	Y=N	3.8	4.0

Safeguard	Experts					Developers		
	Utility		Cost		Mean U/C	Usage	Cost	
	Mean	Median	Mean	Median			Mean	Median
Protect production branch	4.2	4.0	2.0	2.0	2.10	Y=N	1.8	2.0
Remove un-used dependencies	4.3	5.0	2.1	2.0	2.05	Y=N	2.0	2.0
Version pinning [74] [72]	3.7	3.0	2.2	2.0	1.68	Y=N	2.1	2.0
Dependency resolution rules	4.1	4.0	2.6	3.0	1.58	Y=N	2.7	3.0
User account management	3.9	4.0	2.6	3.0	1.50	Y=N	2.3	2.5
Secure authentication (e.g., MFA, password recycle, session timeout, token protection)	4.3	5.0	2.9	3.0	1.48	Y=N	2.5	3.0
Use of security, quality and health metrics	3.5	4.0	2.6	3.0	1.35	Y=N	2.7	3.0
Typo guard/Typo detection [15], [76]	3.9	4.0	2.9	4.0	1.34	Y=N	3.1	3.0
Use minimal set of trusted build dependencies in the release job [51]	4.1	4.0	3.1	3.0	1.32	Y=N	3.8	4.0
Integrity check of dependencies through cryptographic hashes [9]	3.3	3.0	2.5	2.0	1.32	Y=N	2.3	2.0
Maintain detailed SBOM [8] and perform SCA	4.2	5.0	3.4	4.0	1.24	Y=N	2.9	3.0
Ephemeral build environment [9]	3.6	3.0	2.9	3.0	1.24	Y=N	2.8	2.5
Prevent script execution	3.7	3.0	3.0	3.0	1.23	Y=N	2.4	2.0
Pull/Merge request review	4.6	5.0	3.8	4.0	1.21	Y=N	3.6	4.0
Restrict access to system resources of code executed during each build steps [51]	4.0	4.0	3.3	3.0	1.21	Y=N	3.8	3.5
Code signing	3.7	4.0	3.1	3.0	1.19	Y=N	3.1	3.0
Integrate Open-Source vulnerability scanner into CI/CD pipeline	3.8	4.0	3.3	3.0	1.15	Y=N	3.1	3.0
Use of dedicated build service [9]	3.6	4.0	3.3	3.0	1.09	Y=N	3.0	3.0
Preventive squatting the released packages	3.1	3.0	2.9	3.0	1.07	Y=N	3.8	3.5
Audit, security assessment, vulnerability assessment, penetration testing	4.3	4.0	4.1	4.0	1.05	Y=N	3.8	3.5
Reproducible builds	4.2	5.0	4.1	4.0	1.02	Y=N	3.5	4.0
Isolation of build steps [51]	3.1	3.0	3.1	3.0	1.00	Y=N	3.2	3.0
Scoped packages [72], [74]	2.9	3.0	2.9	3.0	1.00	Y=N	2.8	2.0
Establish internal repository mirrors and reference one private feed, not multiple [72]	3.6	3.0	3.7	4.0	0.97	Y=N	2.7	3.0
Application Security Testing	4.1	4.0	4.3	5.0	0.95	Y=N	3.7	3.0
Establish vetting process for Open-Source components hosted in internal/public repositories	4.1	4.0	4.3	5.0	0.95	Y=N	3.8	3.5
Code isolation and sandboxing	3.9	4.0	4.2	4.0	0.93	Y=N	3.2	3.0
Runtime Application Self-Protection	3.7	4.0	4.2	4.0	0.88	Y=N	3.8	4.0
Manual source code review	4.1	4.0	4.8	5.0	0.85	Y=N	4.4	5.0
Build dependencies from sources	3.0	3.0	4.1	4.0	0.73	Y=N	3.8	4.0



Summary

- ▶ Attackers are increasingly using vulnerabilities unintentionally injected into vulnerabilities or are maliciously injecting vulnerabilities into the supply chain
- ▶ We need to be smart about:
 - ▶ Detecting vulnerabilities
 - ▶ Updating components
 - ▶ Making good component choices
 - ▶ Implementing safeguards