A ONE-SIDED TALE:

Investigating a few *Code Dependency* Risks

Funmilayo (Funmi) Olaiya & Anuradha Kulkarni
Cheriton School of Computer Science
University of Waterloo

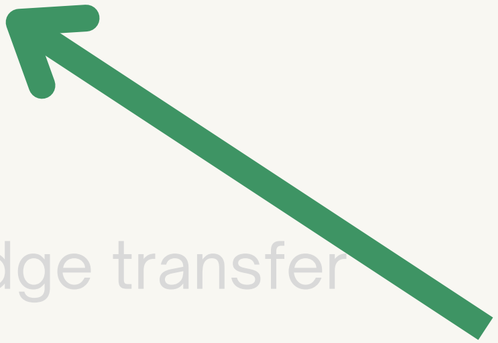
INTRODUCTION

Software Supply Chain

- code dependencies
- code copying
- author-code knowledge transfer

Software Supply Chain

- code dependencies
- code copying
- author-code knowledge transfer



Software Supply Chain

- code dependencies
- code copying
- author-code knowledge transfer

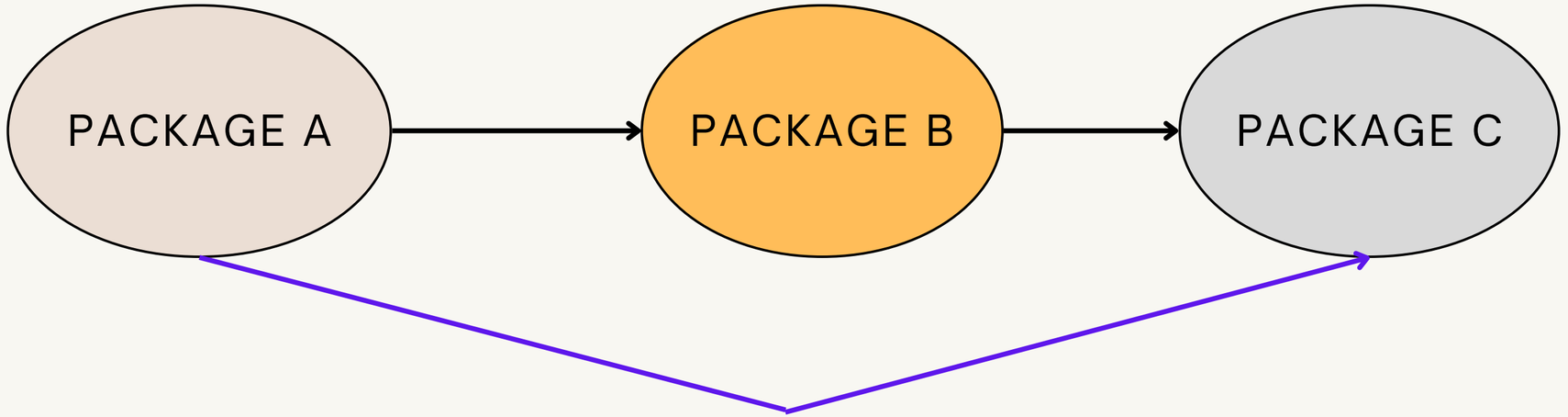
NIST

- Software Supply Chain Security Guidance

Software Supply Chain

- code dependencies

Direct & Indirect Dependencies



MOTIVATION

Code Dependency Risks

- risky dependencies with certain vulnerabilities & license compatibility issues
- *Surviving Software Dependencies* by Russ Cox
- our goal - to promote a strong need for dependency & license inspection

  ~why do these risks exist and why should we care? ~ how do we stay on top of them?

BACKGROUND

Dependency Issues

poorly written code

lack of code maintenance

poor documentation

Security Vulnerabilities

outdated/malicious dependencies

supply chain attacks

configuration issues

License Incompatibility

DATA COLLECTION AND TOOLS

JavaScript Projects

- widely used projects with a commercial perspective ~ WOC
- *package.json* files!
- Snyk tool

Into Data Collection

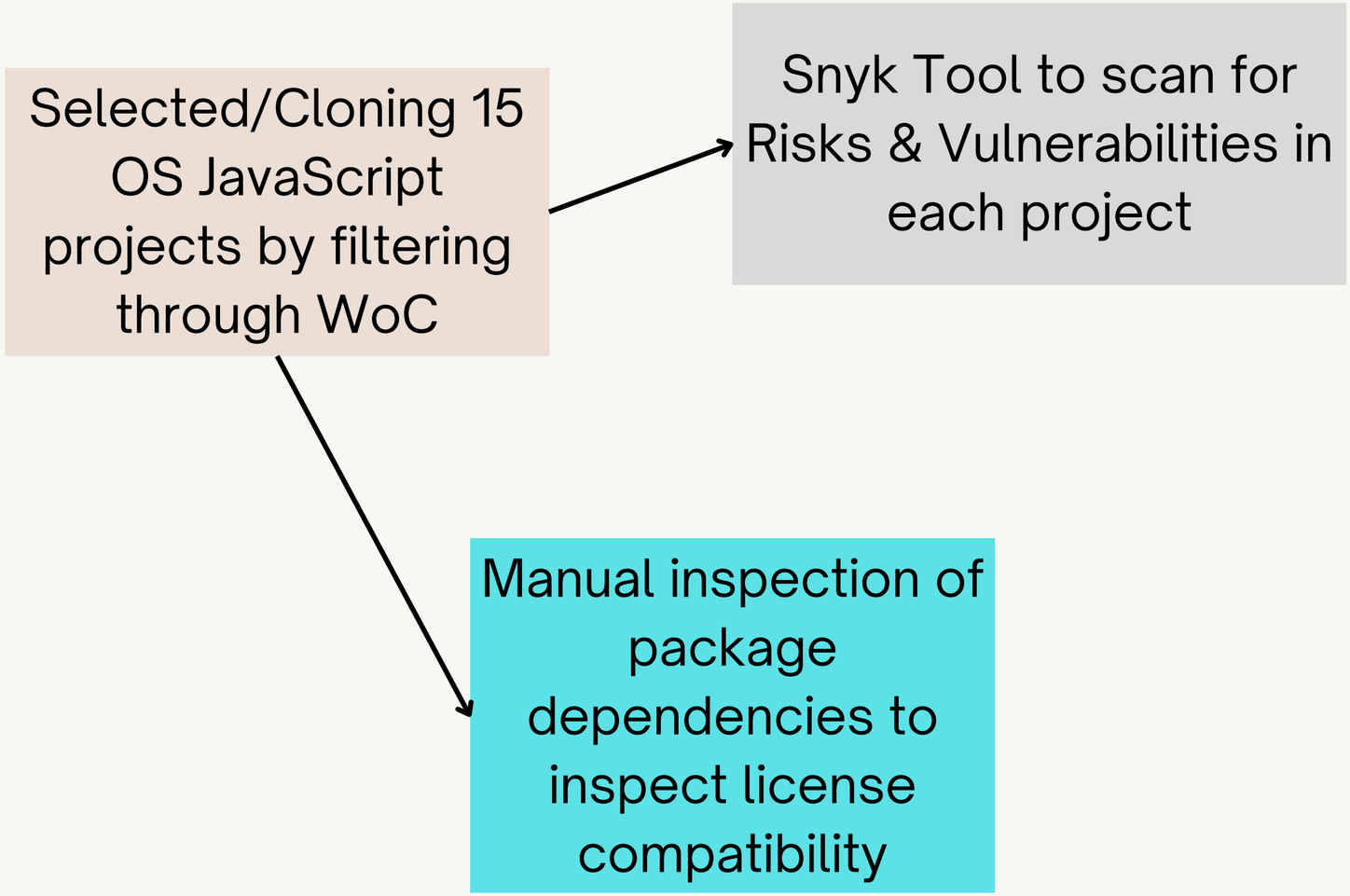
- risky dependencies with their vulnerabilities ~ 75
- the category of the severity of the risks/vulnerabilities through ~ SVD
- licenses of these risky dependencies and others that aren't risky ~ 770 / 75
- Inspection of dependencies based on Maintenance and Usage ~ *Surviving Software Dependencies* by Russ Cox
- are these dependencies direct or indirect ~ *present in package.json*?

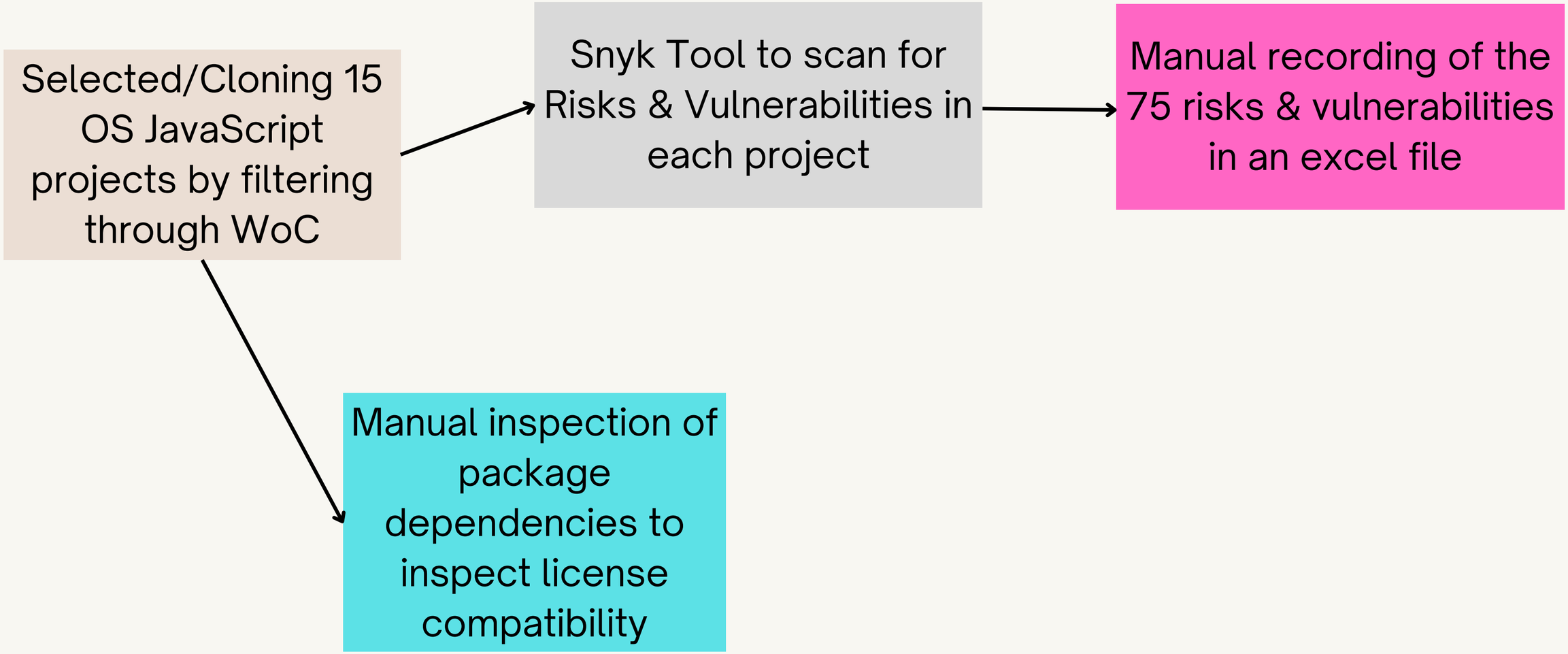
METHODOLOGY

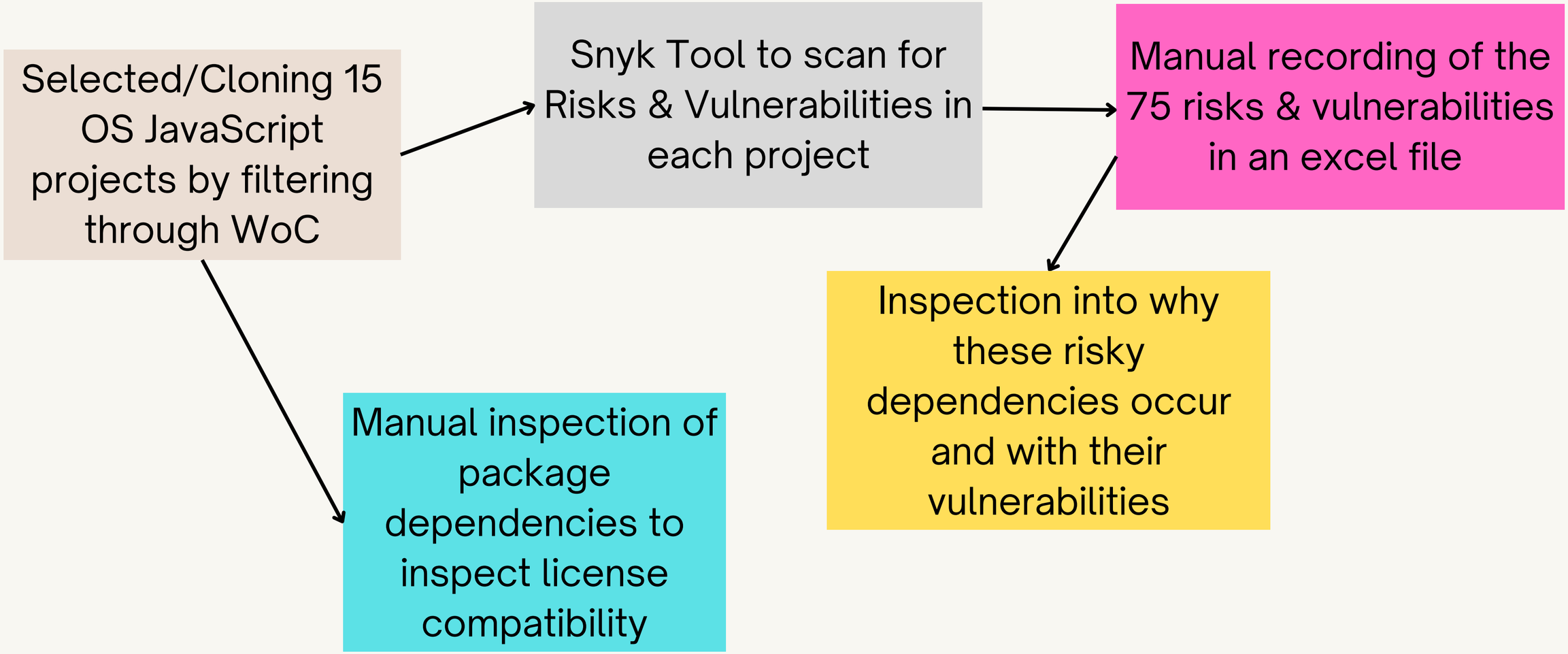
Selected/Cloning 15 OS JavaScript projects by filtering through WoC

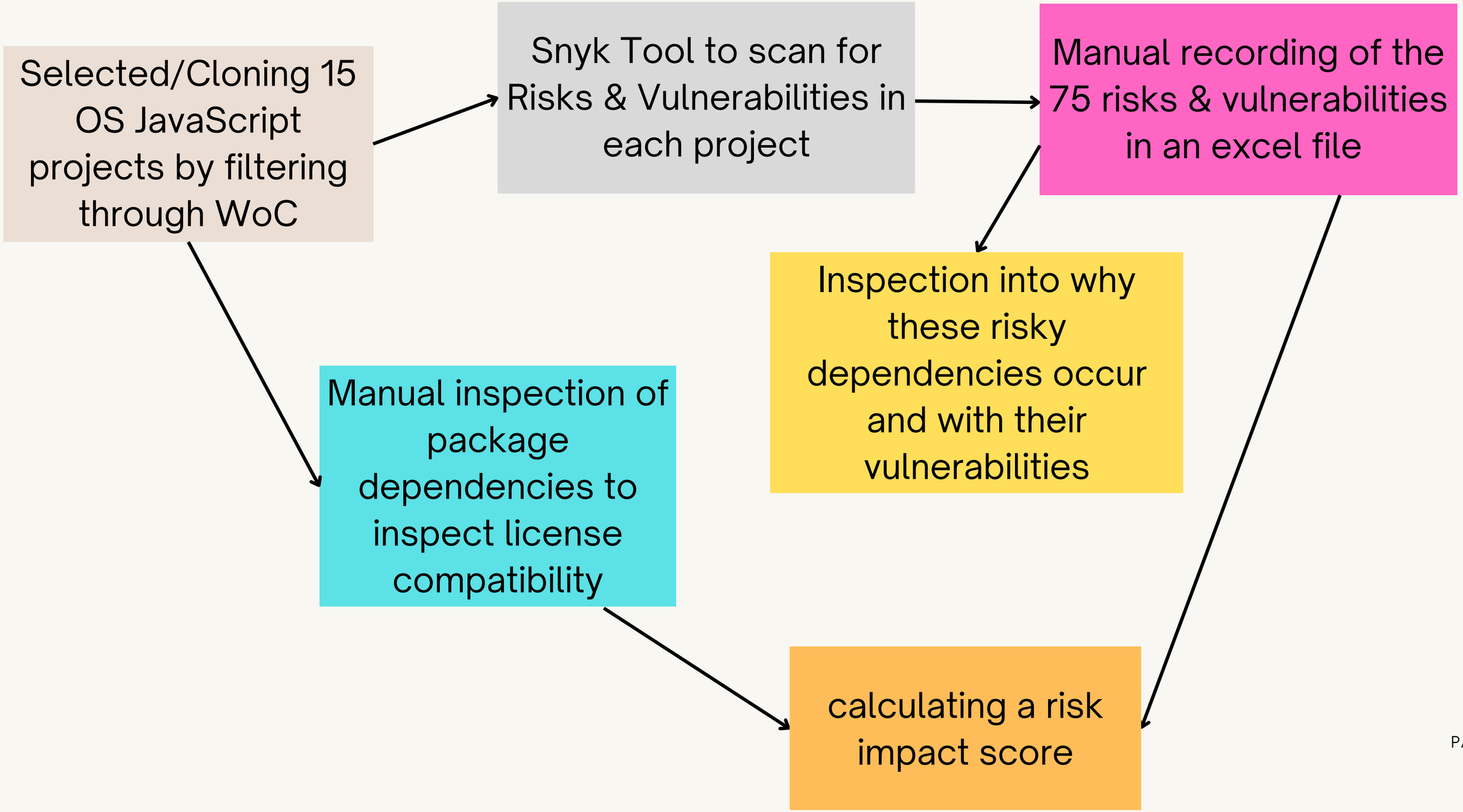
Selected/Cloning 15 OS JavaScript projects by filtering through WoC

Snyk Tool to scan for Risks & Vulnerabilities in each project


Selected/Cloning 15 OS JavaScript projects by filtering through WoC
Snyk Tool to scan for Risks & Vulnerabilities in each project
Manual inspection of package dependencies to inspect license compatibility


Selected/Cloning 15 OS JavaScript projects by filtering through WoC
Snyk Tool to scan for Risks & Vulnerabilities in each project
Manual recording of the 75 risks & vulnerabilities in an excel file
Manual inspection of package dependencies to inspect license compatibility


Selected/Cloning 15 OS JavaScript projects by filtering through WoC
Snyk Tool to scan for Risks & Vulnerabilities in each project
Manual recording of the 75 risks & vulnerabilities in an excel file
Inspection into why these risky dependencies occur and with their vulnerabilities
Manual inspection of package dependencies to inspect license compatibility


Selected/Cloning 15 OS JavaScript projects by filtering through WoC
Snyk Tool to scan for Risks & Vulnerabilities in each project
Manual recording of the 75 risks & vulnerabilities in an excel file
Inspection into why these risky dependencies occur and with their vulnerabilities
Manual inspection of package dependencies to inspect license compatibility
calculating a risk impact score

RESULTS

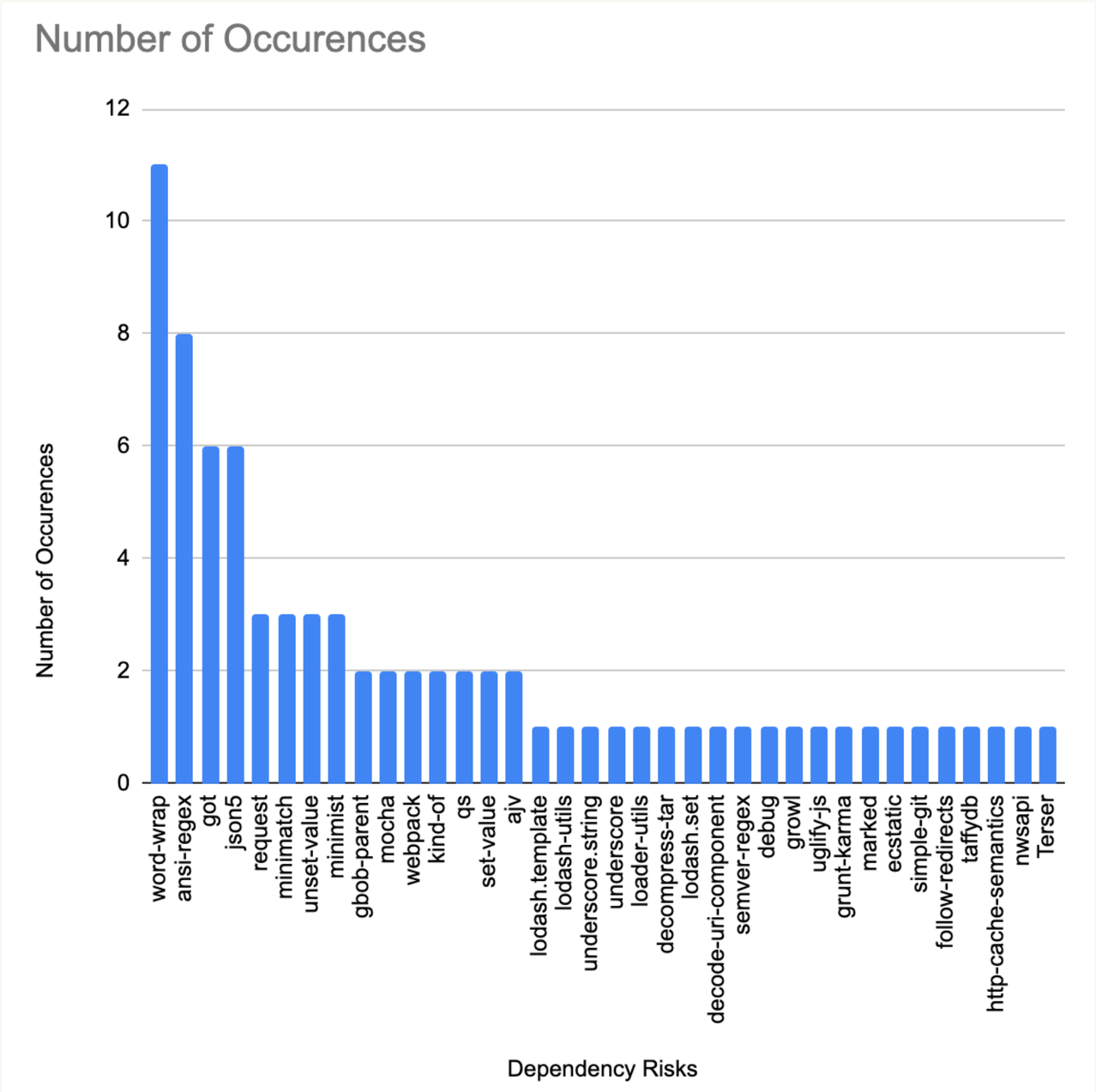
RQ1: WHAT RISKS ARE INVOLVED WITH CODE DEPENDENCIES AND WHY DO THEY EXIST?

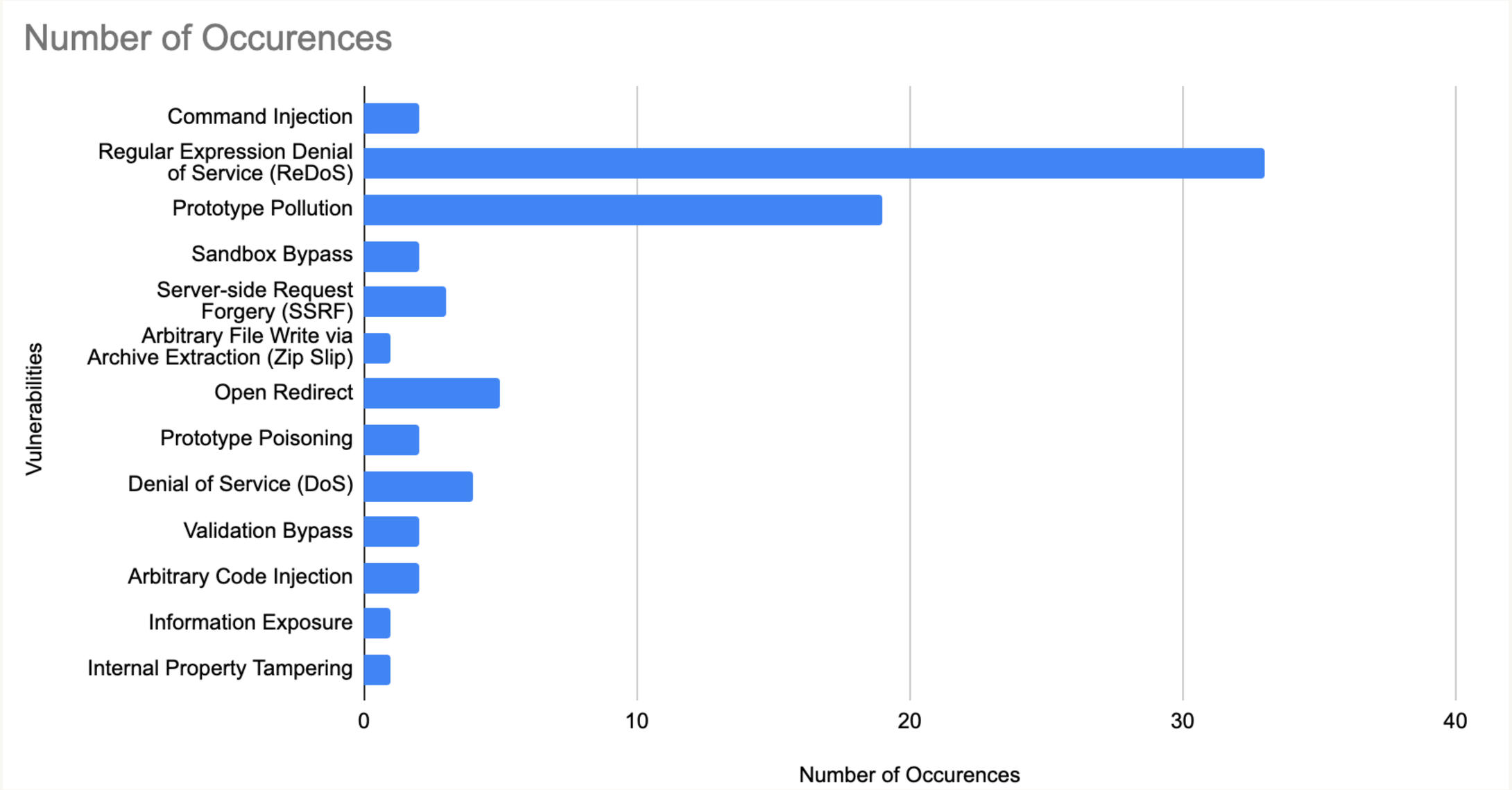
NUMBER OF RISKS IN EACH PROJECT

Projects	No. of Found Risks
ember.js	5
tone.js	5
npm/cli	2
prettier	2
bootstrap	2
adonisjs/core	8
superstruct	3
googlechromelabs/jvsu	1
whatsapp-web-reveng	10
jquery	12
apollographql/apollo-client	1
freecodecamp	4
trekhleb/javascript-algorithms	1
Glider.js	9
pencil.js	10

JQUERY RISKS

Projects	Risks	Vulnerabilities
jquery	semver-regex	Regular Expression Denial of Service (ReDoS)
	debug	ReDoS
	request	Server-side Request Forgery (SSRF)
	growl	Arbitrary Code Injection
	ansi-regex	ReDos
	minimatch	ReDos
	uglify-js	ReDos
	word-wrap	ReDos
	ajv	Prototype Pollution
	underscore	Arbitrary Code Injection
	grunt-karma	Prototype Pollution
	mocha	ReDos


Number of Occurences
12
10
8
6
4
2
0
Number of Occurences
word-wrap
ansi-regex
got
json5
request
minimatch
unset-value
minimist
gbob-parent
mocha
webpack
kind-of
qs
set-value
ajv
lodash.template
lodash-utils
underscore.string
underscore
loader-utils
decompress-tar
lodash.set
decode-uri-component
semver-regex
debug
growl
uglify-js
grunt-karma
marked
ecstatic
simple-git
follow-redirects
taffydb
http-cache-semantics
nwsapi
Terser
Dependency Risks


Number of Occurences
Command Injection
Regular Expression Denial of Service (ReDoS)
Prototype Pollution
Sandbox Bypass
Server-side Request Forgery (SSRF)
Arbitrary File Write via Archive Extraction (Zip Slip)
Open Redirect
Prototype Poisoning
Denial of Service (DoS)
Validation Bypass
Arbitrary Code Injection
Information Exposure
Internal Property Tampering
Vulnerabilities
0
10
20
30
40
Number of Occurences

RISKY DIRECT DEPENDENCIES

Projects	Risks	File
tone.js	webpack	package.json
	mocha	package.json
GoogleChromeLabs/jvsu	got	package.json
prettier	json5	package.json
jquery	uglify-js	package.json
	grunt-karma	package.json
pencil.js	webpack	package.json

RISKY DEPENDENCIES MAINTENANCE AND USAGE

Risky Dependencies	Last Commit Date	No. of Dependents
word-wrap	May 4, 2018	1365
ansi-regex	October 28, 2021	1530
got	March 12, 2023	7,780
json5	January 5, 2023	3,625
request	February 11, 2020	54,988
minimatch	March 22, 2023	7,011
unset-value	February 11, 2022	703
minimist	February, 25, 2023	21,800
glob-parent	September 29, 2021	1,415
mocha	March 30, 2023	9,624
webpack	March 29, 2023	27,164
kind-of	January 16, 2020	1,312
qs	March 6, 2023	13,889
set-value	September 12, 2021	1, 016
ajv	January 8, 2023	9,799
lodash.template	August 18, 2015	965
lodash-utils	Not Found	Not Found
underscore.string	January 23, 2022	3,059
underscore	November 29, 2022	22,736
loader-utils	January 13, 2023	6,565
decompress-tar	July 27, 2017	53
lodash.set	April 23, 2021	1,702
decode-uri-component	December 19, 2022	808
semver-regex	July 8, 2022	243
debug	March 31, 2022	46, 478
growl	November 25, 2020	485
uglify-js	January 16, 2023	4,347
grunt-karma	May 19, 2021	69
marked	March 27, 2023	7,294
ecstatic	April 1, 2020	321
simple-git	March 27, 2023	3,500
follow-redirects	December 8, 2022	1,278
taffydb	January 14, 2021	134
http-cache-semantics	January 26, 2023	377
nwsapi	March 13, 2023	383
Terser	March 26, 2023	1,712

- the two most common risky dependencies have not had a commit to the main/master branch for at least 2 years and more
- not found ~ possibly deleted dependency

RQ 2: WHAT ARE THE LIKELY IMPACTS OF THESE RISKS?

Customization to the OWASP Risk Rating Methodology.

- Each component has a number of options ~ **likelihood** rating from 0 to 9
- Four factors:
  - **Ease of Discovery:** *Easy (0) and Difficult (9)*
  - **Severity of Risk:** *Low (2.25), Medium (4.5), High (6.75), and Critical (9)*
  - **Popularity Index:** *Less Popular (0), Mildly Popular (4.5), and Most Popular (9)*
  - ***Business impact due to license incompatibility:*** *No Violation (0), License Information not Available (4.5), and Violation (9)*

Likelihood	Impact Intensity
0 to <3	LOW
3 to <6	MEDIUM
6 to 9	HIGH

Fig. 1. Likelihood and Impact Intensity Metric

- 30 had a medium impact and 45 had the lowest impact

ADDRESSING SOME DATA FACTS

- Why is **ReDos** the most common vulnerability? *ansi-regex? word-wrap?*
- ***Freezing the Web: A Study of ReDoS Vulnerabilities in JavaScript-based Web Servers*** *by* Staicu et al.

  - *developers are trained to think about the correctness of regular expressions. What of the security aspect?*
  - *Effects of ReDos ~ bypass checks*

- For license compatibility, *we used the FLOSS license graph by David A. Wheeler.*
- So why no license compatibility risks? *Why does almost everything have the MIT license?*
- License Compatibility X Dependency Risks X Vulnerabilities

THREATS TO VALIDITY (EXTERNAL VALIDITY)

- Diversity and Representativeness
- Time Constraints
- Data's Validity in Relation to Time

DISCUSSION, CONCLUSION & FUTURE WORK

Discussion

- How do developers perceive dependency versions? **tone.js** had a webpack risk with a Sandbox Bypass vulnerability. Introduced in **webpack@5.74.0** as webpack has updated to **webpack@5.77.0**
- **Sole Authors** of package dependencies
- **Inspection of dependencies** ~ Paper (Surviving Software Dependencies) by Russ Cox
- Dependency hell ~ an undending loop

Concluding Remarks:

- More than 60 of the risks we found were not direct dependencies
- No license incompatibilities ~ MIT
- Exploring the actual benefits of code dependencies & why these risky dependencies still have dependents?

THANK YOU !

APPENDIX

Furthermore...

- ReDos are incredibly powerful, but they aren't very easy to read or write, and most developers know just enough to be dangerous
- Prototype pollution is an injection attack that targets JavaScript runtimes. With prototype pollution, an attacker might control the default values of an object's properties
- Command injection exploits a programming flaw to execute system commands without proper input validation, escaping, or sanitization, which may lead to arbitrary commands executed by a malicious attacker