Analyzing the Criticality of NPM Packages Through a Time-Dependent Dependency Graph

Author
Anna Brands

Supervisors
Georgios Gousios, Diomidis Spinellis

1 Background

- Using other (open-source) packages as building blocks potentially leads to a large number of **transitive dependencies.**
- Packages that are massively (transitively) depended upon can be quite **vulnerable**. Some examples: *left-pad,* and *Log4Shell* [1],[2].
- Contemporary research has not yet taken into account the **time dimension.**

2 Research Questions

- **RQ1: "What should a graph data structure modeling package dependencies look like?"**
- **RQ2: "On average, does the introduction of the time dimension lead to a significant change in the number of dependent packages per package?"**
- **RQ3: "What are the most-critical packages on NPM?"**

(taking into account the time dimension)

3 Methodology

Designing time-based dependency graph → **Gathering & processing NPM package Metadata** → **Graph data structure implementation** → **Graph data structure verification (automated tests)** → **Querying the data structure for results**

4 Results

Package name	Accuracy score
@angular/animation	1.000
@angular/cli	0,014
@angular/pwa	0.047
@babel/core	0.039
@babel/cli	0.080
@babel/generator	0.188
@bazel/angular	0.150
@cocreate/cli	0.102
@codaco/eslint-plugin-spellcheck	0.005
@codaco/shared-consts	0

Accuracy scores for random sample (RQ1)

Let:

- A be the set of transitive dependencies resolved by NPM
- B be the set of transitive dependencies resolved using the implemented algorithm
- E be the number of dependencies that have a correct name but incorrect version

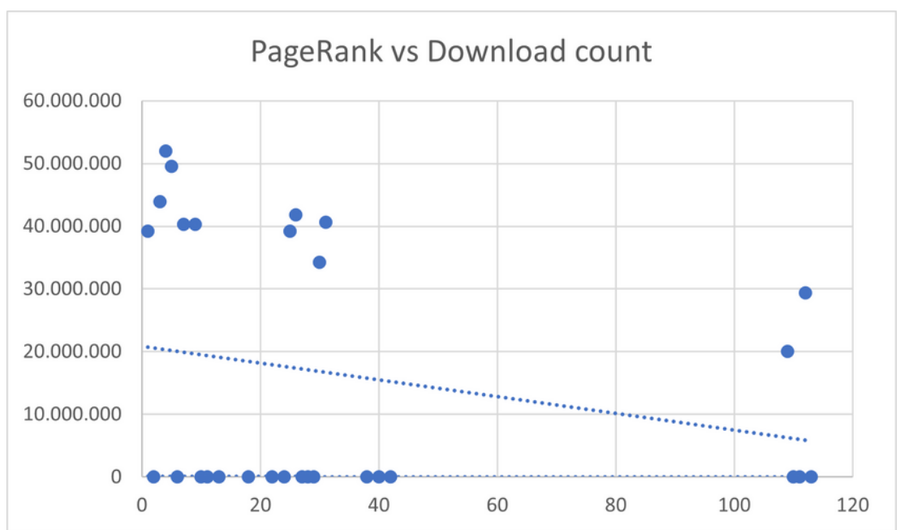
We calculate the accuracy of the algorithm by this formula:

$$Acc = \begin{cases} 1 - \frac{|A| - |(B \cap A)| + 0.5 * E}{|A|}, & \text{if } A \neq \emptyset \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

Accuracy score formula (RQ1)

Package	2019	2021	#trans. deps
@angular/animation	0	0	1
@angular/cli	0	0	383
@angular/pwa	0	0	60
@babel/core	0	0	129
@babel/generator	0	0	16
@bazel/angular	0	0	20
@cocreate/cli	0	0	891
@codaco/eslint-plugin-spellcheck	0	0	980
@codaco/shared-consts	0	0	870

2019, 2021 PageRank vs. number of transitive dependencies (RQ2)



Relation between PageRank and download count over time (RQ3)
(Pearson coefficient of -0.245)

5 Conclusion

- Time-based graph approach to exploring package dependency networks looks promising
- *babel* packages seem to be very important to the NPM dependency network
- Seemingly no positive correlation between package download count and criticality (measured by PageRank)

6 Limitations & Future Work

- Graph generation takes a lot of memory (100k packages take +/- 40GB) -> Make graph generation more memory-efficient before continuing analysis.
- NPM graph was too large to fit into memory leading to incomplete results -> First verify that the time-based graph works for all packages before proceeding.
- Time constraints caused only PageRank to be considered -> use other metrics in the future
- Analyse package managers other than Maven, NPM, PyPI and the debian package manager

References

[1] Schlueter, I. (2016, March 23). npm Blog Archive: kik, left-pad, and npm. NPM Blog. https://blog.npmjs.org/post/141577284765/kik-left-pad-and-npm

[2] Povolny, S., & McKee, D. (2021, December 10). Log4Shell Vulnerability is the Coal in our Stocking for 2021. McAfee Blog. https://www.mcafee.com/blogs/enterprise/mcafee-enterprise-atr/log4shell-vulnerability-is-the-coal-in-our-stocking-for-2021/