Proceedings of the 3rd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/2025.30641

JPSM: A Privilege Score Model for Libraries

Changzheng Guan[1*], Hanlin Song[2]

[1]Hongwen School Qingdao Campus, Qingdao, China
[2]Faculty of Science and Technology, Beijing Normal University - Hong Kong Baptist University United International College, Zhuhai, China
**Corresponding Author. Email: Jayden.Guanchangzheng@hongwenfeh-qd.com*

Abstract. More and more attention is paid to the security of software applications, particularly threatened by the use of third-party libraries. From the node package Manager (NPM), it can be imported the leader loophole. This paper introduces a model (JPSM). The security level of the database can be roughly estimated. pass Assigning weights to different permissions, we support Providing quantifiable risk assessment. Our approach includes A recursive algorithm that evaluates cumulative permissions by introducing a tool downloaded called Mir-sa to improve the accuracy of safety assessments. In this paper, the implementation of this method in JavaScript is discussed. Python and Node.js, showing it in im- Demonstrate security awareness and encourage secure coding Practice.

Keywords: JPSM, Score Model, NPM.

1. Introduction

In the fast-paced world of software development, using third-party libraries has become essential for speeding up innovation. The Node Package Manager (NPM) exemplifies this trend with its vast collection of libraries that developers use to enhance their work. However, relying on these external libraries brings potential security risks that can be exploited, making it crucial to have a strong method for evaluating the security threats associated with these components [1].

The paper at hand introduces a pioneering approach to evaluating the security risks associated with NPM libraries through the calculation of a privilege score. This score is the culmination of a meticulous static analysis process, facilitated by the Mir tool, which enables the extraction and assessment of library permissions without execution. By assigning different weights to various types of permissions, our methodology offers a nuanced perspective on the potential security implications, providing a more granular understanding of risk.

Our approach is methodically structured, commencing with a running example that vividly illustrates the practical application of our method. This is followed by an exhaustive background section that underscores the significance of security in software development and addresses the unique challenges that NPM libraries present. We elucidate the current vulnerabilities and highlight the dearth of effective security assessment tools tailored for NPM libraries.

Delving deeper, the design and implementation section unravels the technical intricacies of our privilege score calculation [2]. We present a detailed exposition of the algorithm, emphasizing its

DOI: 10.54254/2755-2721/2025.30641

recursive nature, which accounts for the cumulative effect of permissions across library dependencies.

This innovative aspect of our approach allows for a more accurate and comprehensive security assessment.

The evaluation section chronicles the application of our method to a curated selection of NPM libraries. The results offer profound insights into the security posture of these components, revealing the prevalence of overlooked vulnerabilities and the efficacy of our privilege score in identifying high-risk libraries.

Next, we delve into these findings, considering their wider impact on developers, development teams, and the software development community at large. We explore how our privilege score can guide development strategies, improve security measures, and promote a culture of heightened security awareness.

To place our work in the context of existing research, we include a section on related work that offers a thorough review of current methods for evaluating library security. This section compares our approach with prevailing practices, emphasizing the novel aspects of our methodology and its potential to reshape the field of library security assessment.

To sum up, this paper presents a new concept for evaluating the security of NPM libraries and advocates for making proactive security assessment a standard part of the development process. We consider how our privilege score contributes to heightened security awareness and enhances the overall security framework within software development.

Our ultimate goal is to empower developers with the tools and knowledge to navigate the complex terrain of library dependencies, fostering a community that prioritizes security and upholds the highest standards of software integrity.

2. Related work

The paper Changing Permissions With Numbers [3] provided a strategy that presents permissions in a numerical form. We gained the idea that express permissions and safety index of files in the form of values.

Additionally, the model in this paper may be more systematic and comprehensive, considering the risks of

different permission combinations, providing a detailed risk analysis method, and providing a way of thinking for the algorithm optimization of script code.

A risk-scoring feedback model for webpages and web users based on browsing behavior:

The JPSM and the model by Ben Neria et al. [4] both aim to bolster security through automated analysis but differ in their focal points and methodologies. The JPSM focuses on evaluating NPM libraries with a hybrid of static and dynamic analysis, assigning "privilege scores" based on requested permissions, thus providing a rule-based evaluation approach. In contrast, Ben Neria et al.'s model uses static analysis and machine learning, specifically spectral clustering, to identify and score risky webpages and user behaviors based on browsing patterns [4].

Despite their differences, both models emphasize proactive security by automating risk identification and integrating user behavior into their assessments. The JPSM's recursive evaluation and the article's model's direct analysis of user webpage interactions both leverage feedback mechanisms to refine risk assessments continuously. Collectively, they underscore the importance of preemptive strategies in identifying and mitigating security vulnerabilities.

DOI: 10.54254/2755-2721/2025.30641

3. Running example

```
/*!
 * array-first
 * <https://github.com/jonschlinkert/array-first>
 *
 * Copyright (c) 2014-2016 Jon Schlinkert.
 * Licensed under the MIT License
 */
'use strict';

var should = require('should');
var first = require('./');

describe('first', function() {
it('should throw an error if the value passed is not an array:', function() {
    (function() {
    first();
    }).should.throw('array-first expects an array as the first argument.');
    });

  it('should return the first element in the array:', function() {
    first(['a', 'b', 'c', 'd', 'e', 'f']).should.eql('a');
    first(['a', 'b', 'c', 'd', 'e', 'f'], 1).should.eql('a'); });

  it('should the first n elements of the array:', function() {
    first(['a', 'b', 'c', 'd', 'e', 'f'], 3).should.eql(['a', 'b', 'c']);
  });
  });
```

An Example array-first library takes the beginning part of an array and returns the entire extracted part or only the first element, depending on the value of the num parameter. We calculate the privilege score of it by the model. Following is the result.

```
// extract Json file with open(json_file_path, 'r') as file: permData = json.load(file)

// find all paths of files all_file_paths = list(permData.keys()) //

find the path of common file floder common_folder =

get_common_folder(all_file_paths)

// initialize total score total_score = 0

// iterate over each file and its corresponding permissions for file_path,

filePerm in permData.items(): // calculate the score of single file file_score

= sum(calculate_score(perm_string) for perm_string

in filePerm.values()) // add all scores

total_score += file_score
```

Table 1. Privilege score. The table below outlines the score of the array-first library

Library Name	Privilege Score	JS File Number
array-first	55	2

4. Background

Third-party libraries simplify the development of extensive software systems. However, these libraries often run with more privileges than necessary to perform their functions [5]. Such excessive

DOI: 10.54254/2755-2721/2025.30641

privilege can be exploited at runtime through inputs provided to a library, even if the library itself is not malicious [6]. Mir is introduced, a system that addresses dynamic compromise by implementing a detailed read-write-execute (RWX) permission model at the interfaces of libraries: every field of every free variable in the context of an imported library is controlled by a set of permissions. With help from Mir tools, it is fast to extract basic permission information of files

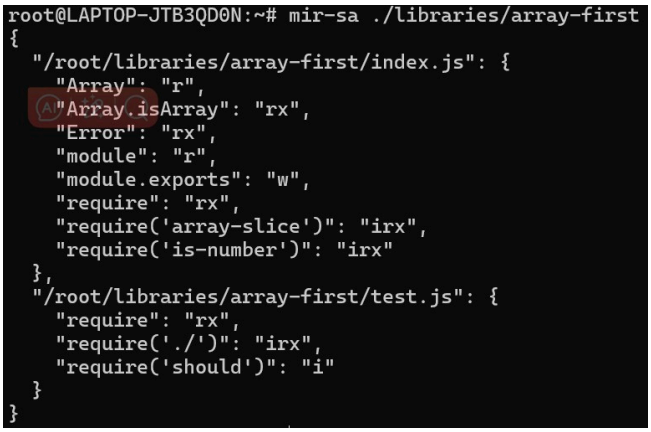

Figure 1. Extract info of array-first

5. Design and implementation

In the research, we have developed a script to evaluate and quantify file permissions within a given directory structure, leveraging a JSON file as input. The script's primary goal is to compute a total permission score based on predefined criteria, which can be useful for security assessments and policy compliance checks. The following sections provide a detailed description of the design choices and implementation details.

5.1. Input handling and rule-making

The initial step requires importing relevant libraries that facilitate file handling and data processing. For instance, libraries such as os and json are commonly employed for interacting with the file system and managing JSON data, respectively. Subsequently, a set of rules or a scoring schema is defined to quantify different file permissions. This involves creating a structured mapping of permission strings to numerical scores, which will later be used to assess the security or accessibility of individual files.

5.2. Determining the common directory

To identify the common directory among all file paths, the script uses the os.path.commonprefix method. This function finds the longest common prefix of the file paths, which is then processed to exclude filenames and retain only the directory path. Finding the common directory helps in focusing the permission analysis on a specific location. This approach provides a clearer context for assessing permissions across a shared directory.

```
def get_common_folder(all_paths):
    common_folder = os.path.commonprefix(all_paths)
    return os.path.dirname(common_folder)
```

Figure 2. Function to achieve 4.2

DOI: 10.54254/2755-2721/2025.30641

5.3. Aggregating results

The script calculates the total permission score by summing the scores for all files and outputs the result. This aggregation step involves iterating through each file's permissions, calculating individual scores, and summing them up.

```
total_score = 0
for file_path, filePerm in permData.items():
    file_score = sum(calculate_score(perm_string) for perm_string in filePerm.values())
    total_score += file_score
```

Figure 3. Iterate over files and their permissions

```
def calculate_score(perm_string):
    total_score = 0
    for perm in perm_string:
        total_score += permission_scores.get(perm, 0)
    return total_score
```

Figure 4. Result a score for single permission

6. Evaluation

People may have doubts about how we carried out the test if the result was correct and how we could optimize the performance of the model when handling more scale libraries. Here are the answers.

6.1. Correctness

To check whether the results obtained by our execution of the script are accurate and correct, the permissions of the script tested were extracted by using mir-sa tool and scored manually according to the weight calculation rules which have formulated. Eventually, the results obtained by two strategies were exactly equal.

Table 2. Score by script. The table below outlines the score of libraries

Library Name	Privilege Score	Score by Hand	JS file number
array-first	55	55	2
array-last	132	132	12
array-range	45	45	2
arr-flatten	265	265	21
dedupe	328	328	40
is-sorted	42	42	2

6.2. Optimization

6.2.1. Parallelization opportunities

The current script processes file permissions serially, which could be inefficient when dealing with a large number of files or extensive permission data. To improve efficiency, consider parallelizing the computation tasks.

DOI: 10.54254/2755-2721/2025.30641

6.2.2. Improve authenticity with complex weighting algorithms

The script currently uses a straightaway static scoring mechanism for file permissions. To make the model more in line with the real online world and provide a more nuanced security assessment, consider implementing more advanced weighting and scoring functions and algorithms related to mathematics and statistics.

6.2.3. Update of software

The current use of language is Python 3.12.4. In the future, new categories or updating of software is possible

7. Conclusion

In the increasingly serious field of network security and user information protection, quantifying file permissions may be a new path or another form of firewall corresponding to ever-changing network attacks, with the maturity of the application algorithm, JPSM will be able to carry out a more comprehensive and more systematic calculation of file permissions weights JPSM's implementation are all open source and available for download: https://github.com/Jayden-Guan/JPSM/tree/main

References

[1] Nikos Vasilakis, Cristian-Alexandru Staicu, Grigoris Ntousakis, Konstantinos Kallas, Ben Karel, André DeHon, and Michael Pradel. 2021. Preventing dynamic library compromise on node. js via rwx-based privilege reduction. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 1821–1838.
[2] Simon Parkinson, Saad Khan. 2022. A Survey on Empirical Security Analysis of Access-control Systems: A Real-world Perspective
[3] Paul A. Kline, Walter A. J. Schiller, 1995 Changing Permissions With Numbers
[4] Michal Ben Neria, Nancy-Sarah Yacovzada, and Irad Ben-Gal. 2017. A risk-scoring feedback model for webpages and web users based on browsing behavior. ACM Transactions on Intelligent Systems and Technology (TIST) 8, 4 (2017), 1–21.
[5] Ui Hyun Park, Jeong-hyeop Hon, Auk Kim, Kyung Ho Son. Department of Convergence Security, Kangwon National University, 2023. Endpoint Device Risk-Scoring Algorithm Proposal for Zero Trust.
[6] Simon Parkinson, Saad Khan, James Bray, Daiyaan Shreef. 2019. Creeper: a tool for detecting permission creep in file system access controls