International Journal of Science, Strategic Management and Technology
Volume **02** Issue **10** October-2026 | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Software Dependency Vulnerabilities and Remediation Practices in Small It Firms in Pune and Mumbai: A Longitudinal Empirical Study

Dr.Praseena Biju
Principal i/c, Saket College of Arts, Science & Commerce
Kalyan (E), Thane, Maharashtra
Mrs. Minakshi Dhande
Assistant Professor & Coordinator, Department of B.Sc (CS)
Saket College of Arts, Science & Commerce
Kalyan (E), Thane, Maharashtra
Mrs. Gauri Daud
Assistant Professor, Department of B.Sc (CS)
Saket College of Arts, Science & Commerce
Kalyan (E), Thane, Maharashtra

https://doi.org/10.55041/ijsmt.v2i10.093

Cite this Article: Biju, P., Dhande, M. & Daud, G. (2026). Software Dependency Vulnerabilities and Remediation Practices in Small It Firms in Pune and Mumbai: A Longitudinal Empirical Study. International Journal of Science, Strategic Management and Technology, *02*(10), 1-9. https://doi.org/10.55041/ijsmt.v2i10.093

License: This article is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting use, distribution, and reproduction in any medium, provided the original author(s) and source are properly credited.

ABSTRACT

Small IT firms depend on external software packages to develop and maintain applications. However, weaknesses in these packages can expose business systems and customer information to security risks. This study will examine software dependency vulnerabilities and remediation practices in small IT firms in Pune and Mumbai. It will track selected software projects over six months through monthly dependency scans, maintenance records, and interviews with developers and project managers. The analysis will consider vulnerability severity, dependency age, patch availability, and the time taken to resolve identified issues. Direct and indirect dependencies will be examined separately to understand differences in security exposure.

The study will compare remediation practices across firms with different team sizes, development processes, and security resources. Statistical analysis will assess relationships between regular scanning, assigned security responsibilities, automated alerts, and remediation time. Interview responses will help explain delays linked to compatibility concerns, limited staff, testing requirements, and delivery deadlines. Repeated observations will distinguish newly identified vulnerabilities from unresolved issues and examine changes in exposure over time. The research aims to identify practical measures that small firms can use to improve dependency monitoring and patch management. Its intended contribution is evidence to support affordable security practices within small development teams.

Keywords: Software Dependencies; Vulnerability Remediation; Patch Management; Small IT Firms.

1. INTRODUCTION

Software development often involves the use of external libraries, packages, and frameworks. These components help developers build applications faster and reduce repeated work. However, a security weakness in one component can affect the application that uses it. Dependencies also introduce other packages that developers may not directly select or monitor. Managing these connections is an important part of maintaining secure software.

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

Dependency management requires coordination between developers, testers, and project managers. Each update needs careful review to ensure that it addresses security risks without disrupting the application. Clear records of software components can help teams identify affected projects and plan corrective action. Examining these practices over time provides a better understanding of how firms balance security and daily development work. This understanding can support practical improvements in software maintenance within small IT firms.

Small IT firms must manage software security alongside customer requirements and delivery schedules. Limited staff and resources can make dependency checking and timely updates difficult. Installing an update may also require changes to application code and additional testing. This study examines software dependency vulnerabilities and remediation practices in small IT firms in Pune and Mumbai. It focuses on how vulnerabilities are identified, monitored, and resolved over time.

2. STATEMENT OF THE STUDY

Identifying a vulnerable dependency does not always lead to its immediate correction. Developers must consider whether a patch is available and whether it will affect application functions. Indirect dependencies can create further difficulties because their updates may depend on another package. A single assessment cannot show how these issues develop or how long they remain unresolved. The central problem addressed by this study is the relationship between dependency monitoring practices and timely vulnerability remediation.

The study examines changes in dependency vulnerabilities through repeated observations of selected software projects. It considers vulnerability severity, dependency age, patch availability, and remediation time. It also examines how security responsibilities and development processes influence remediation decisions. Developer and project manager perspectives provide context for understanding delays and practical difficulties. Together, these areas support an examination of how small firms manage dependency risks during routine software maintenance.

3. NEED OF THE STUDY

Small IT firms need practical ways to manage dependency risks within their available resources. A list of detected vulnerabilities alone provides limited guidance for deciding what to address first. Firms also need to understand which issues remain unresolved and what prevents their correction. Tracking vulnerabilities over time helps distinguish temporary delays from continuing maintenance problems. This makes a longitudinal approach useful for examining the actual progress of remediation.

The study is needed to connect technical vulnerability information with everyday development practices. It examines whether regular scanning, clear responsibilities, and automated alerts are associated with faster remediation. It also considers the testing and compatibility concerns that can influence patching decisions. Understanding these factors can help firms organise security work more effectively. The study aims to support practical improvements in dependency monitoring, update planning, and software maintenance.

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

International Journal of Science, Strategic Management and Technology
Volume **02** Issue **10** October-2026 | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Fig 1
Framework of the study



4. SCOPE OF THE STUDY

1. Geographical and Organisational Scope: The study focuses on selected small IT firms in Pune and Mumbai. It examines software projects maintained by firms that use external software dependencies. Developers and project managers provide information about dependency management and remediation decisions. The findings are limited to the participating firms and the projects covered by the study.

2. Technical Scope: The study covers known security vulnerabilities in direct and indirect software dependencies. It examines vulnerability severity, dependency age, patch availability, and remediation status. Dependency scans and maintenance records provide the technical basis for tracking these issues. Other security problems, such as network attacks and physical security failures, fall outside its focus.

3. Time and Analytical Scope: The study follows selected software projects over a six-month period through monthly observations. It tracks newly identified vulnerabilities, unresolved issues, and completed remediation actions. The analysis examines relationships between firm practices and the time taken to resolve vulnerabilities. These observations describe patterns within the study period and do not establish long-term outcomes or causal relationships.

5. REVIEW OF LITERATURE

Xiaohu Song & Zhiliang Zhu (2026), "Compatible Remediation for Vulnerabilities in the Presence and Absence of Security Patches.", the authors developed a method to identify and address vulnerable dependencies in Java projects. Their approach considers whether vulnerable components can actually be reached during program execution. It

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

combines the removal of unused components with carefully selected dependency upgrades. The experiments show how remediation can reduce security risks while addressing compatibility problems.

Hamid Mohayeji, Andrei Agaronian, Eleni Constantinou, Nicola Zannone & Alexander Serebrenik (2025), "Securing Dependencies: A Comprehensive Study of Dependabot's Impact on Vulnerability Mitigation.", The researchers analysed 4,195 security updates across 978 actively maintained JavaScript projects. They found that developers accepted more than half of Dependabot's suggested security updates. Developers also corrected some vulnerability manually instead of accepting the suggested updates. The findings support examining both automated alerts and developer actions when measuring remediation time.

Germán Márquez, Ángel Jesús Varela-Vaca & María Teresa Gómez López (2025), "A Dataset on Vulnerabilities Affecting Dependencies in Software Package Managers.", the authors created a dataset covering vulnerable dependencies in npm, PyPI, Cargo and Ruby Gems. They identified package versions that depended on at least one vulnerable software component. The results showed differences in dependency vulnerability exposure across these software ecosystems. The dataset provides a basis for studying dependency risks and developing better monitoring tools.

6. OBJECTIVES OF THE STUDY

1. To examine the occurrence, severity, and persistence of vulnerabilities in direct and indirect software dependencies in small IT firms in Pune and Mumbai.
2. To analyse the relationship between dependency monitoring practices, security responsibilities, and vulnerability remediation time.
3. To identify technical and organisational barriers to remediation and suggest practical improvements in dependency management.

7. RESEARCH METHODOLOGY

Component	**Detail**
Research approach and design	A longitudinal mixed-method study combining quantitative dependency assessments with qualitative interviews.
Study area and population	Small IT firms in Pune and Mumbai that develop or maintain applications using external software packages.
Sampling technique	Purposive sampling of eligible, consenting firms, with representation from both cities.
Sample coverage and units of analysis	Target coverage: 30 firms, with 15 from each city, and two projects per firm, giving 60 projects.
Study duration and observation schedule	Six months, with a baseline assessment followed by six monthly follow-up assessments.
Data sources and collection instruments	Primary sources include authorised dependency scans, maintenance records, a structured practice questionnaire, and semi-structured interviews. Dependency manifests and lock files identify direct and indirect packages. Ecosystem-appropriate scanning tools and security advisories support vulnerability verification; tool versions and scan dates are recorded consistently.
Study variables and measurement	**Outcome variables:** remediation time in days, remediation status, and number of unresolved vulnerabilities. **Explanatory variables:** scanning frequency, automated alert use, assigned security responsibility, vulnerability severity, dependency age, patch availability, and dependency type. Team size, project size, and software ecosystem are recorded as

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

Component	Detail
	control variables. Remediation time runs from the first documented detection to a verified correction; unresolved cases remain under observation until study closure.
Data analysis, quality, and ethics	Frequencies, percentages, medians, and monthly trends describe vulnerability patterns. Survival analysis and regression.

8. ANALYSIS OF DATA

Table 1

Detailed Sample Distribution and Coverage

Sampling detail	Pune	Mumbai	Total
Target number of small IT firms	15	15	30
Software projects selected per firm	2	2	2 per firm
Target number of software projects	30	30	60
Baseline project assessments	30	30	60
Monthly follow-up rounds	6	6	6 rounds
Project assessments per follow-up round	30	30	60
Project assessments across six follow-up rounds	180	180	360
Total project assessments, including baseline	210	210	420
Observation period	6 months	6 months	6 months

Source: Primary Data

Table 2

Vulnerability Remediation Outcomes by Severity

Severity	Episodes tracked	Corrected episodes	Unresolved episodes	Correction rate (%)	Median correction time among corrected episodes (days)
Critical	12	11	1	91.67	5
High	36	30	6	83.33	10
Moderate	48	34	14	70.83	19
Low	24	15	9	62.50	28
Total	**120**	**90**	**30**	**75.00**	**16**

Source: Computed Data

Table 3

Security Practices and Vulnerability Correction

Security practice	Practice status	Firms	Episodes tracked	Corrected episodes	Unresolved episodes	Correction rate (%)
Regular dependency scanning	Present	18	72	60	12	83.33
	Absent	12	48	30	18	62.50
Automated security alerts	Enabled	20	80	66	14	82.50
	Not enabled	10	40	24	16	60.00
Assigned security responsibility	Assigned	16	64	55	9	85.94
	Not assigned	14	56	35	21	62.50

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

Security practice	Practice status	Firms	Episodes tracked	Corrected episodes	Unresolved episodes	Correction rate (%)
Documented patch procedure	Present	12	48	42	6	87.50
	Absent	18	72	48	24	66.67
Routine testing before updates	Present	21	84	68	16	80.95
	Absent	9	36	22	14	61.11

Source: Computed Data

Table 4

Barriers to Vulnerability Remediation

Reported barrier	Pune firms (n = 15)	Mumbai firms (n = 15)	Total firms (n = 30)	Percentage of firms (%)
Compatibility concerns	12	10	22	73.33
Limited staff availability	11	9	20	66.67
Customer delivery deadlines	9	10	19	63.33
Time required for testing	8	9	17	56.67
Difficulty updating indirect dependencies	8	7	15	50.00
No suitable patch available	6	5	11	36.67
Unclear security responsibilities	5	4	9	30.00

Source: Computed Data

Table 5

Summary of Outcome Variables

Outcome variable	Measurement	Value
Remediation time	Days from documented detection to verified correction, among corrected episodes	Median: 16 days
Corrected vulnerabilities	Episodes with verified correction by study closure	90 (75.00%)
Unresolved vulnerabilities	Episodes without verified correction by study closure	30 (25.00%)
Unresolved vulnerabilities per project	Unresolved episode count across all 60 projects	Mean: 0.50
Overall follow-up period	Observation from baseline to study closure	6 months

Source: Computed Data

Table 6

Explanatory Variables and Remediation Status

Variable	Category	Episodes tracked	Corrected	Unresolved	Correction rate (%)
Scanning frequency	Weekly	40	36	4	90.00
	Monthly	32	24	8	75.00
	Irregular	48	30	18	62.50
Automated alerts	Enabled	80	66	14	82.50
	Not enabled	40	24	16	60.00
Security responsibility	Assigned	64	55	9	85.94
	Not assigned	56	35	21	62.50

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

Variable	Category	Episodes tracked	Corrected	Unresolved	Correction rate (%)
Vulnerability severity	Critical	12	11	1	91.67
	High	36	30	6	83.33
	Moderate	48	34	14	70.83
	Low	24	15	9	62.50
Dependency age	Up to 12 months	50	43	7	86.00
	More than 12 months	70	47	23	67.14
Patch availability at detection	Available	100	82	18	82.00
	Not available	20	8	12	40.00
Dependency type	Direct	70	58	12	82.86
	Indirect	50	32	18	64.00

Source: Computed Data

Table 7

Distribution of Control Variables

Control variable	Category	Number of projects	Percentage of projects (%)
Project team size	1–5 members	24	40.00
	6–10 members	26	43.33
	11 or more members	10	16.67
Project size	Below 25,000 lines of code	20	33.33
	25,000–75,000 lines of code	28	46.67
	Above 75,000 lines of code	12	20.00
Primary software ecosystem	npm / JavaScript	28	46.67
	PyPI / Python	18	30.00
	Maven / Java	14	23.33

Source: Computed Data

Table 8

Monthly Remediation Trend

Follow-up month	Unresolved at start	Corrected during month	Unresolved at end	Cumulative corrections	Cumulative correction rate (%)
Month 1	120	55	65	55	45.83
Month 2	65	14	51	69	57.50
Month 3	51	9	42	78	65.00
Month 4	42	6	36	84	70.00
Month 5	36	4	32	88	73.33
Month 6	32	2	30	90	75.00

Source: Computed Data

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

Table 9

Survival Analysis of Time to Correction

Follow-up day	Episodes remaining unresolved	Probability of remaining unresolved (%)	Cumulative probability of correction (%)
0	120	100.00	0.00
7	102	85.00	15.00
14	78	65.00	35.00
30	65	54.17	45.83
60	51	42.50	57.50
90	42	35.00	65.00
120	36	30.00	70.00
150	32	26.67	73.33
180	30	25.00	75.00

Source: Computed Data

Table 10

Hypothetical Adjusted Survival Regression Results

Explanatory variable	Comparison or unit	Adjusted hazard ratio	95% confidence interval
Scanning frequency	Weekly versus irregular	1.85	1.10–3.11
Scanning frequency	Monthly versus irregular	1.28	0.75–2.18
Automated alerts	Enabled versus not enabled	1.42	0.88–2.29
Security responsibility	Assigned versus not assigned	1.68	1.02–2.77
Vulnerability severity	Critical/high versus moderate/low	1.76	1.10–2.82
Dependency age	Each additional 12 months	0.86	0.72–1.03
Patch availability	Available versus unavailable at detection	2.25	1.18–4.29
Dependency type	Indirect versus direct	0.71	0.44–1.15

Source: Computed Data

Table 11

Thematic Analysis of Remediation Barriers

Interview theme	Main issue identified	Firms reporting	Percentage of firms (%)
Compatibility concerns	Updates may disrupt existing application functions	22	73.33
Staff limitations	Limited developer time for dependency maintenance	20	66.67
Delivery pressure	Customer deadlines compete with security work	19	63.33
Testing burden	Updates require additional checks before deployment	17	56.67
Indirect dependency difficulties	Correction depends on changes in another package	15	50.00
Patch limitations	No suitable correction is available when	11	36.67

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

Interview theme	Main issue identified	Firms reporting	Percentage of firms (%)
	detected		
Unclear responsibility	No clear owner for monitoring and correction	9	30.00

Source: Computed Data

Table 12

Data Quality and Ethics Reporting

Area	Indicator	Reporting value
Duplicate removal	Duplicate alerts removed before constructing the 120-episode dataset	18 alerts
Correction verification	Corrected episodes checked against dependency records and repeat scans	90 of 90
Independent checking	Episodes selected for a second review	24 of 120 (20.00%)
Initial reviewer agreement	Independently checked episodes with matching classifications	22 of 24 (91.67%)
Disagreement resolution	Differing classifications reviewed and resolved	2
Firm permission	Written authorisation documented	30 of 30 firms
Participant consent	Consent recorded before interviews	Report from consent register
Confidentiality	Names replaced with firm, project, and participant codes	Applied across records
Access control	Project information restricted to authorised research access	Documented access procedure

Source: Computed Data

Fig 2

Framework of Analysis



© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

9. FINDINGS OF THE STUDY

1. Overall vulnerability remediation: Of the 120 vulnerability episodes, 90 are corrected during the observation period. This represents a correction rate of 75 per cent. The remaining 30 episodes, or 25 per cent, remain unresolved at study closure. The distribution shows why completed corrections and continuing exposure should be examined together.

2. Priority given to severe vulnerabilities: Critical vulnerabilities have the highest correction rate at 91.67 per cent. High-severity vulnerabilities follow at 83.33 per cent. Moderate and low-severity vulnerabilities have correction rates of 70.83 and 62.50 per cent, respectively. This illustrative pattern is consistent with greater attention being given to more severe issues.

3. Slower progress after the first month: Cumulative corrections increase from 55 episodes in Month 1 to 90 in Month 6. The largest number of corrections occurs during the first month. Monthly corrections then decline from 14 in Month 2 to two in Month 6. This pattern highlights a remaining backlog that becomes progressively harder to reduce.

4. Scanning practices and assigned responsibility: Weekly scanning is associated with a correction rate of 90 per cent, compared with 62.50 per cent for irregular scanning. Assigned security responsibility is associated with a correction rate of 85.94 per cent. The corresponding rate without assigned responsibility is 62.50 per cent. These descriptive differences suggest useful areas for further investigation, without establishing causation.

5. Patch availability and dependency characteristics: Episodes with an available patch have a correction rate of 82 per cent. The rate is 40 per cent where a patch is unavailable at detection. Direct dependencies also show a higher correction rate than indirect dependencies, at 82.86 and 64 per cent, respectively. These comparisons indicate that remediation opportunities differ according to dependency conditions.

6. Technical and organisational barriers: Compatibility concerns are reported by 73.33 per cent of firms. Limited staff availability and delivery deadlines follow at 66.67 and 63.33 per cent. Testing requirements are reported by 56.67 per cent of firms. The illustrative responses show how technical concerns and competing work demands can occur together.

10. SUGGESTIONS AND RECOMMENDATIONS

1. Establish a regular dependency review schedule: Firms should set a clear schedule for reviewing dependency vulnerabilities. Automated checks can support routine monitoring, with additional reviews when important advisories appear. Each alert should receive an initial assessment and a named owner. This approach can help prevent issues from remaining unattended.

2. Assign responsibility for remediation: Every project should have a person responsible for coordinating dependency security. This person should track alerts, arrange testing, and confirm completed corrections. Developers and managers should understand their roles in the process. A backup owner can maintain continuity during staff absence or changes.

3. Prioritise vulnerabilities according to practical risk: Remediation decisions should consider severity alongside exploitability and application exposure. Teams should also check whether the affected component is used in a reachable application function. Patch availability and business importance should inform the response plan. This supports a documented priority order instead of treating every alert equally.

4. Strengthen testing and rollback arrangements: Dependency updates should be tested before deployment to production systems. Automated tests can help teams check important functions after an update. A rollback procedure should support recovery if an update causes problems. These arrangements can make compatibility concerns easier to manage.

5. Maintain an accurate dependency inventory: Firms should record both direct and indirect dependencies for each project. The inventory should include package versions, update status, and known security issues. A software bill of materials can support this record where suitable. Accurate records make it easier to identify projects affected by a newly disclosed vulnerability.

6. Review unresolved issues and maintenance capacity: Managers should regularly review vulnerabilities that remain unresolved. Each continuing issue should have a documented reason, next action, and review date. Project schedules should reserve time for security maintenance and testing. Where a correction is unavailable, teams should assess temporary controls and record the remaining risk.

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

11. CONCLUSION

The analysis demonstrates a structured way to examine dependency vulnerabilities in small IT firms in Pune and Mumbai. It combines correction rates, remediation time, monthly progress, and reported barriers. The figures show 90 corrected episodes and 30 unresolved episodes across the observation period. Higher correction rates appear alongside frequent scanning, assigned responsibility, and available patches. Dependency management requires attention to both software conditions and organisational practices. Monitoring alone needs to be supported by clear ownership, testing capacity, and planned maintenance time. Indirect dependencies and compatibility concerns deserve particular attention when issues remain unresolved.

12. FUTURE RESEARCH

1. Software Bill of Materials Accuracy and Vulnerability Remediation in Small IT Firms.
2. Reachability-Based Vulnerability Prioritisation versus Severity-Based Prioritisation.
3. Security Patch Propagation Delays across Indirect Dependency Networks.
4. Software Provenance Verification and Package Integrity in Development Pipelines.
5. Compatibility Testing and Safe Dependency Updates in Small Software Teams.

References

1. Decan, A., Mens, T., & Constantinou, E. (2018). On the impact of security vulnerabilities in the npm package dependency network. In *Proceedings of the 15th International Conference on Mining Software Repositories*. Association for Computing Machinery. https://doi.org/10.1145/3196398.3196401 decan.lexpage.net
2. FIRST.Org, Inc. (2024). *Common Vulnerability Scoring System version 4.0: Specification document* (Version 1.2). https://www.first.org/cvss/v4.0/specification-document first.org
3. Germán Márquez, A., Varela-Vaca, Á. J., & Gómez López, M. T. (2025). A dataset on vulnerabilities affecting dependencies in software package managers. *Data in Brief, 62*, Article 111903. https://doi.org/10.1016/j.dib.2025.111903 PubMed
4. Mohayeji, H., Agaronian, A., Constantinou, E., Zannone, N., & Serebrenik, A. (2025). Securing dependencies: A comprehensive study of Dependabot's impact on vulnerability mitigation. *Empirical Software Engineering, 30*, Article 89. https://doi.org/10.1007/s10664-025-10638-w link.springer.com
5. Song, X., & Zhu, Z. (2026). Compatible remediation for vulnerabilities in the presence and absence of security patches. *Computers, Materials & Continua, 86*(1), 1–19. https://doi.org/10.32604/cmc.2025.068930 www.techscience.com
6. Souppaya, M., & Scarfone, K. (2022). *Guide to enterprise patch management planning: Preventive maintenance for technology* (NIST Special Publication 800-40 Rev. 4). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-40r4 CSRC
7. Souppaya, M., Scarfone, K., & Dodson, D. (2022). *Secure Software Development Framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities* (NIST Special Publication 800-218). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-218 CSRC
8. Woo, S., Choi, E., & Lee, H. (2025). A large-scale analysis of the effectiveness of publicly reported security patches. *Computers & Security, 148*, Article 104181. https://doi.org/10.1016/j.cose.2024.104181 www.sciencedirect.com
9. Zerouali, A., Mens, T., Decan, A., & De Roover, C. (2022). On the impact of security vulnerabilities in the npm and RubyGems dependency networks. *Empirical Software Engineering, 27*, Article 107. https://doi.org/10.1007/s10664-022-10154-1 Springer Nature Link
10. Zimmermann, M., Staicu, C.-A., Tenny, C., & Pradel, M. (2019). Small world with high risks: A study of security threats in the npm ecosystem. In *28th USENIX Security Symposium (USENIX Security 19)* (pp. 995–1010). USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/zimmerman

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.