

# NPM Tedarik Zincirinde Topolojik Risk Analizi ve Kritiklik Haritalaması

*Topological Risk Analysis and Criticality Mapping in the NPM Supply Chain*

**Özetçe**—NPM gibi merkezi paket yöneticileri, kullanımına hazırlanan paketler halinde sundukları kod kütüphaneleriyle uygulama geliştirme süreçlerine ivme kazandırsa da, bu bileşenler arasındaki iç içe geçmiş bağımlılık ilişkileri, ekosistemi karmaşık ve kırılgan bir yapıya dönüştürmüştür. Mevcut güvenlik yaklaşımı, ağıın yapısal mimarisinden kaynaklanan ve zincirleme etki potansiyeli taşıyan bu sistemik riskleri tespit etmekte yetersiz kalmaktadır. Bu çalışmada, NPM ekosistemindeki sistemik risklerin, paket içeriğinden bağımsız topolojik analiz yöntemleriyle haritalandırılması amaçlanmıştır. Analiz kapsamında, NPM ekosisteminde en çok bağımlıya sahip 1000 paket ve bunların 7. kademe derinliğine kadar uzanan bağımlılıkları modellenerek yönlü bir graf oluşturulmuştur. Ağ üzerinde in-degree, out-degree ve betweenness metrikleri hesaplanarak, bu değerlerin ağırlıklı kombinasyonuyla "Bileşik Risk Skoru" (BRS) geliştirilmiştir. Hesaplanan metriclerin istatistiksel dağılımı, ağıın ölçekten bağımsız (scale-free) bir topoloji sergilediğini; buna bağlı olarak riskin, ekosistemin omurgasını oluşturan as sayıda kritik düğüm üzerinde yoğunlaştığı ortaya koymaktadır. Sağlıklık simülasyonları, yüksek BRS skoruna sahip paketlerin ağıdan koparılmasının, ağıın bütünlüğünde yıkıcı bir çöküse yol açtığını doğrulamıştır. Araştırma, kısıtlı güvenlik kaynaklarının rastgele taramalar yerine ekosistemin omurgasını oluşturan kritik düğümlere yönlendirilmesi gerektiğini vurgulayarak, literatüre topoloji tabanlı özgün bir perspektif kazandırmaktadır.

**Anahtar Kelimeler**—*Yazılım tedarik zinciri güvenliği, NPM, bağımlılık ağı analizi, topolojik risk, kaskad etkisi.*

**Abstract**—While centralized package managers like NPM accelerate software development by providing ready-to-use code libraries, the intricate web of nested dependencies among these components has transformed the ecosystem into a complex and fragile structure. Current security approaches remain insufficient in detecting these systemic risks, which stem from the network's structural architecture and possess the potential for cascading effects. This study aims to map systemic risks within the NPM ecosystem using topological analysis methods that are independent of package content. Within the scope of the analysis, a directed graph was constructed by modeling the top 1,000 packages with the most dependents in the NPM ecosystem, along with their dependencies extending up to a depth of 7. In-degree, out-degree, and betweenness metrics were calculated on the network, and a novel 'Composite Risk Score' (BRS) was developed through a weighted combination of these values. The statistical distribution of the calculated metrics reveals that the network exhibits a scale-free topology; consequently, risk is concentrated on a small number of critical nodes that form the backbone of the ecosystem. Robustness simulations confirmed that the removal of packages with high BRS scores leads to a destructive collapse in network integrity. This research contributes a novel topology-based perspective to the literature by emphasizing that limited security resources should be directed toward the critical nodes forming the ecosystem's backbone, rather than relying on random scans.

**Keywords**—*Software supply chain security, NPM, dependency network analysis, topological risk, cascade effect.*

## I. GİRİŞ

Modern yazılım mühendisliğinin temel dinamiklerinden biri olan verimlilik arayışı, geliştirme süreçlerini NPM, PyPI ve RubyGems gibi merkezi paket yöneticilerinin sunduğu kullanıma hazır kod kütüphanelerine bağımlı kılmıştır [1], [2]. Bu modüler mimari, yazılım geliştirme süreçlerini hızlandırmakla birlikte, tedarik zincirinin herhangi bir noktasındaki zayıflıktan tüm ekosisteme yayılabilen kırılgan bir zemin oluşturmuştur [1]. Milyonlarca paketi ve bunlar arasındaki girdi bağımlılık ilişkilerini barındıran NPM ekosistemi, sunduğu geniş saldırısı yüzeyiyle tehdit aktörleri için cazip bir hedef konumundadır [3]. Zayıflıların bağımlılık ağları üzerinden kontrolsüz yayılmış [4], [5] ve paketlerin bakım süreçlerindeki aksaklılıklar [6]–[8], bu riski artırmaktadır. Literatür, bu ağıın "küçük dünya" (small-world) özellikleri taşıdığını, dolayısıyla sınırlı sayıda paketin veya bakımının ekosistem üzerinde orantısız bir etkiye sahip olduğunu doğrulamaktadır [9]–[12].

Tedarik zinciri saldırıları, güvenilir paketlerin ele geçirilmesinden, popüler paket isimlerinin taklit edildiği "typosquatting" tuzaklarına kadar geniş bir yelpazede kendini göstermektedir [13]. Kaynak zehirlenmesi (source poisoning) [14], prototip kirliliği (prototype pollution) [15] ve Node.js mimarisine özgü bağımlılık tabanlı saldırılar [16], tehditlerin çeşitliliğini ortaya koymaktadır. "Backstabber's Knife Collection" [13] ve "The Hitchhiker's Guide" [17] gibi kapsamlı çalışmalar, saldırıların anatomisini kurulum ve çalışma zamanı evreleri üzerinden incelerken; Duan ve ark. [2], kayıt defteri (registry) düzeyindeki istismarlara odaklanarak yüzlerce kötü niyetli paketin varlığını belgelemektedir.

Bu tehditlere karşı Amalfi [18], Cerebro ve OSCAR [19] gibi makine öğrenmesi ve dinamik analiz tabanlı savunma mekanizmalarının yanı sıra; meta veri analizi [20], kötü niyetli davranış sekansları [21], diller arası tespit [22], güvenli kullanım pratikleri [23] ve imza tabanlı yaklaşımlar [24]–[26] geliştirilmiştir. Ancak ekosistemin devasa ölçü, her bir paketin derinlemesine taranmasını maliyet ve zaman açısından sürdürülemez kılmaktadır.

Literatürde, paketlerin bakım durumu [6] veya bilinen zayıflıların yayılma dinamikleri [4] üzerinden risk analizi yapan değerli çalışmalar mevcuttur. Ancak, ağıın topolojik yapısından kaynaklanan sistemik çöküş risklerini (kaskad etkisini) ölçen ve bunları operasyonel bir risk skoruna dönüştüren yaklaşımlar kısıtlıdır. Mevcut çalışmalar genellikle ağıın genel karakteristiklerini betimsel düzeye ortaya koymakta; güvenlik kaynaklarının optimizasyonu için somut bir önceliklendirme mekanizması sunma konusunda eksik kalmaktadır. Bu çalışmada, paket içeriklerinden bağımsız olarak, yalnızca paketler arası bağımlılıkların topolojik mimarisine odaklanılmış ve yapısal riskin haritalandırılması hedeflenmiştir.

Çalışma kapsamında, NPM'in standart bağımlılık çözümleme algoritmaları temel alınarak inşa edilen yönlü graf üze-

rinde in-degree, out-degree ve betweenness ölçümüleri ağırlıklandırılarak **Bileşik Risk Skoru (BRS)** geliştirilmiştir. Bu model, kaskad etki ve en büyük bağlı bileşen (LCC) analizle-riyle desteklenerek, topolojik riskin sistemik yansımaları nice verilerle ortaya konulmuştur. Böylece, güvenlik analistleri ve paket yöneticileri için önceliklendirilmiş bir izleme listesi oluşturularak, kısıtlı güvenlik kaynaklarının en kritik noktalara yönlendirilmesi amaçlanmıştır.

## II. VERİ VE YÖNTEM

### A. Veri Setinin Oluşturulması ve Kapsam

Çalışmada, salt indirme sayıları yerine sistemik etkiye merkeze alan bir örneklem stratejisi benimsenmiştir. Bu doğrultuda, `ecosyste.ms` verilerine göre en çok bağımlıya (*dependents*) sahip ilk 1.000 paket başlangıç kümesi (seed) olarak belirlenmiş ve bu kümenin 7. derinlik seviyesine kadar olan tüm bağımlılıkları (*dependencies*) analiz kapsamına alınmıştır. Döngüsel referansların giderildiği veri ön işleme sürecinin ardından, 1.506 düğüm ve 3.058 kenardan oluşan yönlü bir graf elde edilmiştir. Ağın modellenmesi ve analiz süreçlerinde Python NetworkX kütüphanesi kullanılmıştır.

### B. Merkeziyet Ölçüleri ve Normalizasyon

Paketlerin ağ içindeki önemini belirlemek için üç merkeziyet metriği kullanılmıştır:

- **In-Degree:** Doğrudan bağımlı paket sayısını ifade eder. Popülerite ve doğrudan etki alanının göstergesi sidir.
- **Out-Degree:** Dış bağımlılık sayısını gösterir. Yüksek out-degree, saldırı yüzeyinin genişliğini ifade eder.
- **Betweenness:** En kısa yollar üzerinde bulunma sıklığıdır. Paketin ağıdaki köprü rolünü ve stratejik konumunu yansıtır.

### C. Matematiksel Model

NPM ekosistemi, paketlerin düğümleri ( $V$ ) ve bağımlılıkların kenarları ( $E$ ) temsil ettiği yönlü bir graf  $G = (V, E)$  olarak modellenmiştir. Her bir düğüm  $v \in V$  için risk skoru hesaplanmadan önce, metrikler Min-Max yöntemiyle  $[0, 1]$  aralığına normalize edilmiştir:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (1)$$

Burada  $x$ , ilgili metriğin ham değerini;  $x'$  ise normalize edilmiş değerini ifade eder.

### D. Bileşik Risk Skoru (BRS)

Normalizasyon sonrası değerler ağırlıklandırılarak Bileşik Risk Skoru (BRS) hesaplanmıştır:

$$\text{BRS} = 0.5 \cdot \text{in}' + 0.2 \cdot \text{out}' + 0.3 \cdot \text{btw}'$$

Formülde in-degree ( $w_{in} = 0.5$ ) en yüksek ağırlığa sahiptir. Bu durum, paketin etkileyeceği proje sayısının sistemik risk göstergesi olmasına ilişkilidir. Betweenness ( $w_{btw} = 0.3$ ) ikinci faktör olarak modele dahil edilmiştir. Ağırlık katsayıları, tedarik zinciri saldırılarda "etki alanı"nın (impact radius), "saldırı yüzeyi"nden daha kritik bir risk faktörü olduğu varsayımına dayanılarak empirik olarak belirlenmiştir.

### E. Sağlamlık ve Kaskad Analizleri

Model geçerliliğinin testi için hedefli saldırı simülasyonları uygulanmıştır. Yüksek BRS skorlu paketler ağdan çıkarılarak En Büyük Bağlı Bileşen (LCC) boyutu ve ağ erişilebilirliği üzerindeki etkiler analiz edilmiştir. Ayrıca kaskad analizi ile modelin öngörü gücü doğrulanmıştır.

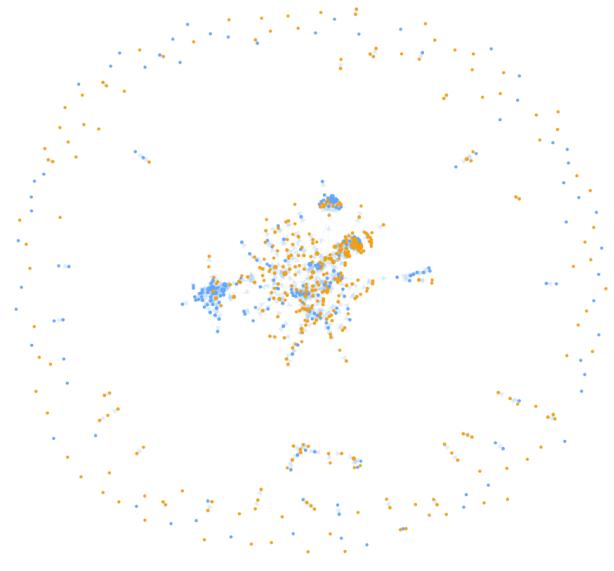
## III. BULGULAR VE DEĞERLENDİRME

### A. Ağ Topolojisi ve Yapısal Karakteristikler

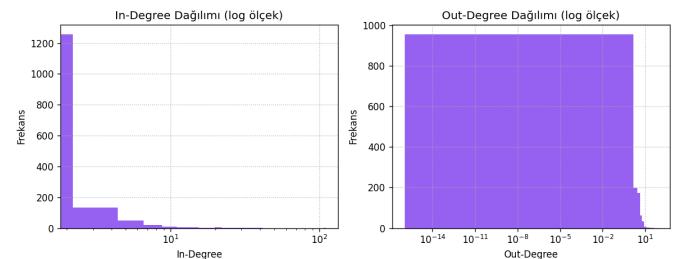
TABLO I. AĞ İSTATİSTİKLERİ

Metrik	Değer
Düğüm Sayısı (Nodes)	1506
Kenar Sayısı (Edges)	3058
Ortalama Derece (Avg Degree)	2.03
Yoğunluk (Density)	0.0013
Ortalama Arasılık (Avg Betweenness)	$1.05 \times 10^{-5}$

Top N + Bağımlılıklar



Şekil 1. Top 1000 paket ağının görselleştirmesi. Yoğun bölgeler alt kümeleri işaret etmektedir.

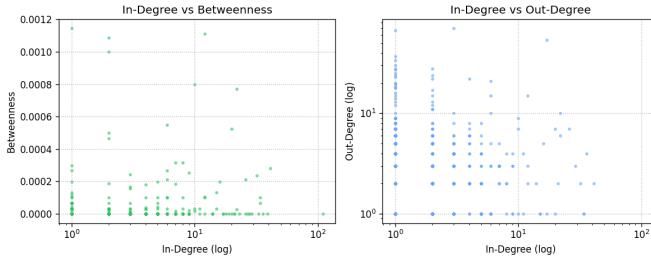


Şekil 2. In-degree ve out-degree histogramları. Dağılımın ağır kuyruklu yapısı görülmektedir.

Şekil 1'de ağ topolojisinin kümelenme yapısı görülmektedir. Şekil 2'deki derece dağılımları, ağın ölçekte bağımsız

(scale-free) yapısını göstermektedir. Düğümlerin çoğunluğu az sayıda bağlantıya sahipken, az sayıda düğüm yüksek bağlantı sayısına sahiptir. Bu bulgu, riskin belirli bir bölümde yoğunlaştığını işaret etmektedir.

### B. Merkeziyet İlişkileri



Şekil 3. Merkeziyet ölçüleri arasındaki korelasyonlar.

Korelasyon matrisi (Şekil 3), in-degree ile betweenness arasında asimetrik bir ilişki olduğunu göstermektedir. Yüksek popüleriteye sahip olmayan bazı paketlerin köprü görevi gördüğü belirlenmiştir. Sadece indirme sayılarına odaklanan analizlerin yapısal riskleri tespit etmede yetersiz kalabileceği değerlendirilmektedir.

### C. Kritik Düğümlerin Analizi

TABLO II. TOP 10 IN-DEGREE

Paket	In-Degree	Out-Degree	Betweenness	TopN?
@babel/helper-plugin-utils	110	0	0.000000	True
call-bound	41	2	0.000283	False
postcss-value-parser	39	0	0.000000	True
call-bind	36	4	0.000000	False
@types/node	34	1	0.000067	False
debug	34	1	0.000100	True
es-errors	33	0	0.000000	False
@babel/types	32	2	0.000236	True
define-properties	29	3	0.000000	False
chalk	28	0	0.000000	False

TABLO III. TOP 10 OUT-DEGREE

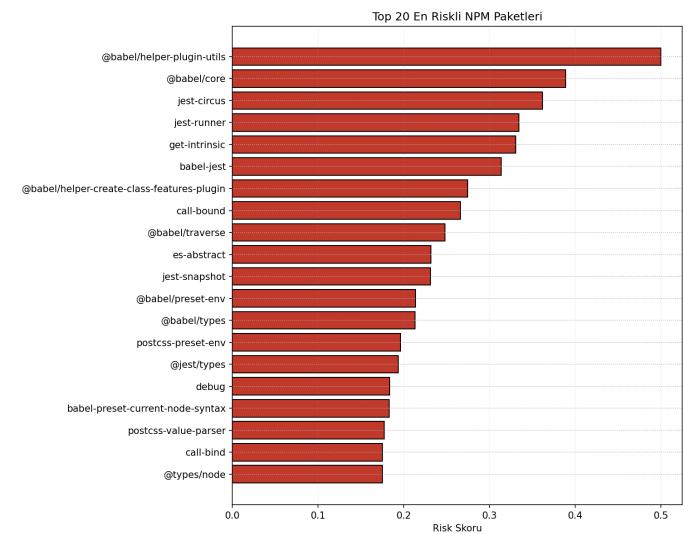
Paket	Out-Degree	In-Degree	Betweenness	TopN?
@babel/preset-env	70	3	0.000000	True
postcss-preset-env	67	1	0.000000	True
es-abstract	54	17	0.000000	False
react-scripts	48	0	0.000000	True
workbox-build	37	1	0.000000	True
eslint	34	1	0.000000	True
cssnano-preset-default	30	1	0.000000	True
webpack-dev-server	28	1	0.000000	True
@jest/core	28	2	0.000000	True
express	27	1	0.000000	True

TABLO IV. TOP 10 BETWEENNESS

Paket	Betweenness	In-Degree	Out-Degree	TopN?
jest-circus	0.001144	1	20	False
@babel/core	0.001112	12	15	True
babel-jest	0.001087	2	7	True
jest-runner	0.001000	2	22	True
@babel/helper-create-class-features-plugin	0.000798	10	7	True
get-intrinsic	0.000771	22	10	True
jest-snapshot	0.000549	6	21	True
@babel/traverse	0.000523	20	7	True
babel-preset-current-node-syntax	0.000499	2	15	False
postcss-value-parser	0.000466	2	5	True

Tablo II'deki yüksek in-degree değerine sahip paketler, sık kullanılan bileşenlerdir. Tablo III'deki yüksek out-degree'li paketler geniş bir saldırıcı yüzeyi oluşturmaktadır. Tablo IV, ağ trafığını yöneten paketleri göstermektedir. Örneğin, jest-circus paketi düşük in-degree (1) değerine sahip olmasına rağmen, yüksek betweenness değeriyle ağda kritik bir köprü rolü üstlenmektedir. Benzer şekilde @babel/core, hem popülerliği hem de stratejik konumuya dikkat çekmektedir. Bu durum, risk analizinde tek bir metriğin yeterli olmadığını ve yapısal köprülerin önemini kanıtlamaktadır.

### D. Bileşik Risk Skoru (BRS) Sıralaması



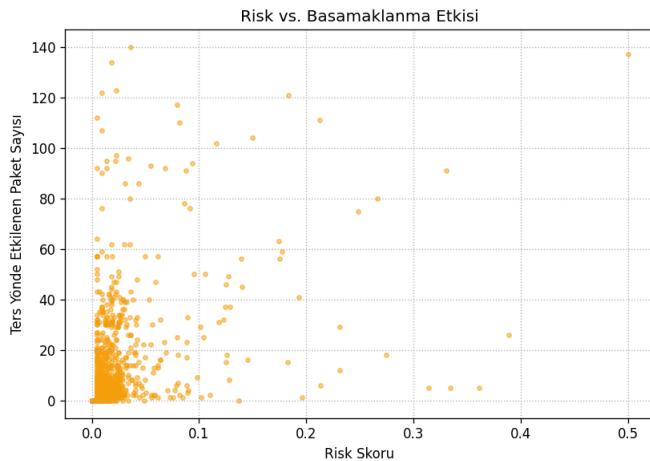
Şekil 4. En yüksek Bileşik Risk Skoru (BRS) sahip 20 paket.

TABLO V. TOP 20 BİLEŞİK RISK SKORU (BRS)

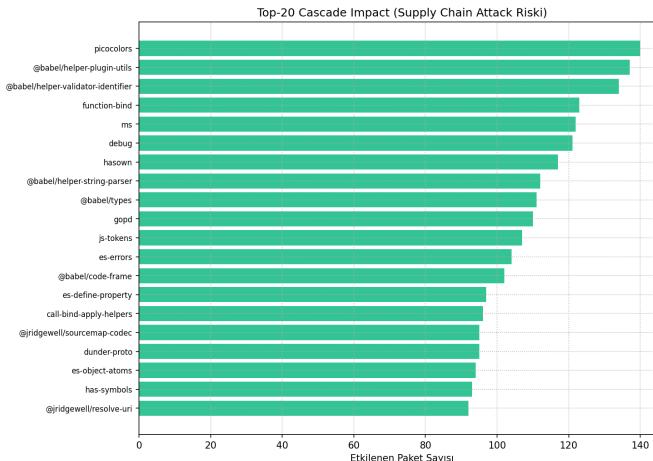
Paket	Risk	In-Degree	Out-Degree	Betweenness	TopN?
@babel/helper-plugin-utils	0.500000	110	0	0.000000	True
@babel/core	0.388827	12	15	0.001112	True
jest-circus	0.361688	1	20	0.001144	False
jest-runner	0.334157	2	22	0.001000	True
get-intrinsic	0.330606	22	10	0.000771	True
babel-jest	0.313975	2	7	0.001087	True
@babel/helper-create-class-features-plugin	0.274757	10	7	0.000798	True
call-bound	0.266206	41	2	0.000283	False
@babel/traverse	0.248118	20	7	0.000523	True
es-abstract	0.231558	17	54	0.000000	False
jest-snapshot	0.231168	6	21	0.000549	True
@babel/preset-env	0.213636	3	70	0.000000	True
@babel/types	0.212942	32	2	0.000236	True
postcss-preset-env	0.195974	1	67	0.000000	True
@jest/types	0.193414	26	7	0.000211	True
debug	0.183565	34	1	0.000100	True
babel-preset-current-node-syntax	0.182762	2	15	0.000499	False
postcss-value-parser	0.177273	39	0	0.000000	True
call-bind	0.175065	36	4	0.000000	False
@types/node	0.174844	34	1	0.000067	False

BRS modeli; popülerlik, saldırıcı yüzeyi ve stratejik konumu birleştirmektedir. Şekil 4 ve Tablo V'te görüldüğü üzere, hem popülerliği hem de ağ konumu nedeniyle belirli paketler üst sıralarda yer almaktadır. Sıralama, güvenlik denetimlerinde önceliklendirilecek paketleri göstermektedir.

## E. Sistemik Etki ve Kaskad Analizi



Şekil 5. BRS ile kaskad etki (erişilebilirlik) arasındaki ilişki.



Şekil 6. İlk 20 paketin çıkarılmasının LCC ve erişilebilirlik üzerindeki etkisi.

Simülasyon sonuçları BRS modelini desteklemektedir. Şekil 6'da, yüksek BRS skorlu paketlerin çıkarılmasının LCC boyutunda düşüse neden olduğu görülmektedir. BRS skoru ile kaskad etki arasındaki korelasyon (Şekil 5), metriğin risk öngörüsündeki başarısını göstermektedir.

## IV. TARTIŞMA VE SONUÇ

Bu çalışmada, literatürdeki betimsel analizlerin ötesine geçilerek, NPM ekosistemindeki yapısal riskleri ölçülebilir bir değere dönüştüren **Bileşik Risk Skoru (BRS)** modeli sunulmuştur. Giriş bölümünde belirtilen "operasyonel önceklendirmeye" eksikliği, elde edilen BRS sıralaması ile giderilmiştir. Analizler, ağıın güvenliğinin sadece popüler paketlere değil, jest-circus veya @babel/core gibi altyapısal köprü paketlere de bağlı olduğunu göstermiştir. Elde edilen bulgular, Zimmermann ve arkadaşlarının [9] belirttikleri "küçük dünya" ağ yapısını doğrulamakta; ancak riskin sadece popüler paketlerde değil, köprü (bridge) niteliğindeki ara katman paketlerinde de yoğunlaştığını ortaya koymak literatürü genişletmektedir.

## A. Çalışmanın Kısıtları

Analiz, `package.json` dosyalarında tanımlı statik bağımlılıklar ile sınırlanmıştır. Çalışma zamanında (runtime) dinamik olarak yüklenen paketler veya geliştirici hesaplarının güvenilirliği (reputation) bu aşamada kapsam dışı bırakılmıştır. Ayrıca analiz, anlık bir kesit (snapshot) üzerinden yapılmış olup, ağıın zamansal evrimini içermemektedir. Benzer şekilde, lisans uyumluluğu gibi yasal riskler [27] de bu çalışmanın kapsamı dışındadır.

## B. Öneriler

Bulgular doğrultusunda aşağıdaki öneriler geliştirilmiştir:

- Önceliklendirme:** Güvenlik taramalarında kaynakların BRS skoru yüksek paketlere yönlendirilmesi.
- Politika Geliştirme:** Güvenlik protokollerinin [28] ve SBOM uygulamalarının [29] kritik düğümlere uygulanması.
- Farkındalık:** Paket sahiplerinin risk skorları hakkında bilgilendirilmesi.

Gelecek çalışmalarında modelin geliştirici ağlarını kapsayan şekilde genişletilmesi önerilmektedir.

## V. YENİDEN ÜRETİLEBİRLİK

Çalışmanın tekrarlanabilirliği için kaynak kodlar ve veri setleri paylaşılmıştır:

- Analiz Kodları:** `analysis/analysis.ipynb` (Python 3, NetworkX, pandas).
- Veri Çıktıları:** Tüm ara sonuçlar ve metrikler `results/` dizininde CSV/JSON formatında sunulmuştur.

## KAYNAKLAR

- [1] E. Wyss, "A new frontier for software security: Diving deep into npm," 2025.
- [2] R. Duan et al., "Towards measuring supply chain attacks on package managers," in *NDSS*, 2020.
- [3] M. Wang, P. Wu, and Q. Luo, "Construction of software supply chain threat portrait based on chain perspective," 2023.
- [4] C. Liu et al., "Demystifying vulnerability propagation via dependency trees in npm," in *ICSE*, 2022.
- [5] A. Zerouali et al., "On the impact of security vulnerabilities in the npm and RubyGems dependency networks," 2022.
- [6] I. Rahman et al., "Characterizing dependency update practice of NPM, PyPI and Cargo packages," 2024.
- [7] F. R. Cogo, "Studying dependency maintenance practices through mining NPM," 2020.
- [8] A. J. Jafari et al., "Dependency practices for vulnerability mitigation," 2023.
- [9] M. Zimmermann et al., "Small world with high risks: Security threats in npm," in *USENIX Sec.*, 2019.
- [10] A. Hafner, A. Mur, and J. Bernard, "Node package manager's dependency network robustness," 2021.
- [11] E.-R. Oldnall, "The web of dependencies: A complex network analysis of the NPM," 2017.
- [12] P. Jaisri, B. Reid, and R. G. Kula, "A preliminary study on self-contained libraries in the NPM ecosystem," 2024.
- [13] M. Ohm et al., "Backstabber's knife collection: A review of open source software supply chain attacks," in *DIMVA*, 2020.

- [14] T. G. Hastings, “Combating source poisoning and next-generation software supply chain attacks,” 2024.
- [15] M. Shcherbakov, P. Moosbrugger, and M. Balliu, “Unveiling the invisible: Prototype pollution gadgets via dynamic taint,” 2021.
- [16] D. Y. K. Yip, “Empirical study on dependency-based attacks in Node.js,” 2022.
- [17] P. Ladisa et al., “The hitchhiker’s guide to malicious third-party dependencies,” in *IEEE S&P*, 2023.
- [18] A. Sejia and M. Schafer, “Practical automated detection of malicious npm packages (Amalfi),” in *ICSE*, 2022.
- [19] X. Zheng et al., “Towards robust detection of OSS supply chain poisoning (OSCAR),” 2024.
- [20] S. Halder et al., “Malicious package detection using metadata information,” 2024.
- [21] J. Zhang et al., “Malicious package detection in NPM and PyPI using a single model of malicious behavior sequence,” 2023.
- [22] P. Ladisa et al., “On the feasibility of cross-language detection of malicious packages in npm and PyPI,” 2023.
- [23] N. Imtiaz, “Toward secure use of open source dependencies,” 2023.
- [24] M. L. P. Correia, “Detection of software supply chain attacks in code repositories,” 2022.
- [25] M. Ohm et al., “Supporting detection via unsupervised signature generation (ACME),” 2021.
- [26] T. R. Schorlemmer, “Software supply chain security: Attacks, defenses, and signing adoption,” 2024.
- [27] H. E. Ahlstrom, “Dependency analysis for software licensing and security,” 2025.
- [28] S. Torres-Arias, “In-toto: Practical software supply chain security,” in *USENIX Sec.*, 2020.
- [29] S. Yu, “Accurate and efficient SBOM generation for software supply chain security,” 2024.