Yazılım Tedarik Zincirinde Kritiklik Haritalaması
NPM Ekosisteminde Topolojik Risk Analizi

Yusuf Talha ARABACI

Ekim 2025

Özet

NPM ekosisteminde tek bir bağımlılıktaki kusur veya kötü niyetli değişiklik, transitif bağımlılıklar üzerinden çok kısa sürede binlerce projeye sirayet edebilir. Bu çalışma, paket içeriklerinden ziyade paketler arası ilişkilerin *topolojik* örüntülerine odaklanarak yüksek etki alanlı düğümlerin belirlenmesini amaçlar. Son 12 aylık indirme verisiyle seçilen Top 1000 paket üzerinden yönlü bağımlılık grafiği kurulmuş; in-degree, out-degree ve betweenness merkeziyetleri min–max normalizasyonu ile *Bileşik Risk Skoru*na (BRS) dönüştürülmüştür. Sağlamlık ve kaskad analizleriyle kritik düğümlerin çıkarımının en büyük bağlı bileşen ve erişilebilirlik üzerindeki etkisi ölçülmüş, sonuçlar uzun tablolar ve görsellerle raporlanmıştır.

1 Giriş ve Motivasyon

Açık kaynak paket yöneticileri modern yazılım geliştirmenin vazgeçilmez bir bileşeni hâline gelirken, bağımlılık zincirlerindeki tekil kırılmalar sistemik risk üretmektedir. NPM ekosistemi, milyonlarca paket ve yoğun ters-bağımlı ağ yapısı nedeniyle *kırılgan omurga düğümlerine* sahiptir. Mevcut araştırmalar kritik düğümlerin varlığını ortaya koysa da, bu düğümlerin tespit hatlarına nasıl öncelik kuyruğu olarak aktarılacağı ve politika/bütünlük süreçlerine nasıl bağlanacağı yeterince detaylandırılmamıştır.

Çalışmanın Katkıları

1. Resmî çözümleme kurallarına sadık kalarak kurulan yönlü graf üzerinde, in-/out-degree ve betweenness ölçülerini min–max ölçekleyen ve $w_{in} = 0.5$, $w_{out} = 0.2$, $w_{btw} = 0.3$ ağırlıklarıyla birleştiren **Bileşik Risk Skoru (BRS)** tanımlanmıştır.
2. BRS çıktıları; kaskad etki, en büyük bağlı bileşen (LCC) ve köprü kenar analizleriyle ilişkilendirilerek topolojik riskin sistemik yansımaları nicelleştirilmiştir.
3. Tüm veriler `results/` klasöründe izlenebilir CSV/JSON çıktılarına, tablolar ise `analysis/make_tables.py` betiğiyle yeniden üretilebilir LaTeX dosyalarına bağlanarak *uçtan uca şeffaflık* sağlanmıştır.

2 Literatür Özeti

Literatür taraması dört eksende toplanmaktadır: (i) tehdit taksonomileri, (ii) topolojik analizler, (iii) tespit hatları ve (iv) politika/bütünlük çalışmaları.

2.1 Tehdit Taksonomileri ve Vaka Derlemeleri

Backstabber's Knife Collection ve The Hitchhiker's Guide gibi derlemeler, kayıt saldırılarının kurulum/anlık yürütme evrelerine dair kapsamlı taksonomiler sunar. Duan ve arkadaşları kayıt istismarı fenomenini hem istatistiksel hem de niteliksel yöntemlerle inceleyerek 339 yeni kötü paket bildirmektedir. Bu çalışmalar, olay farkındalığını artırsa da *önceliklendirilmiş müdahale listeleri* üretmemektedir.

2.2 Topoloji ve Merkeziyet Çalışmaları

Zimmermann, Hafner ve Oldnall tarafından yürütülen çalışmalar NPM ağının küçük-dünya özellikleri taşıdığını ve az sayıdaki bakımcının orantısız etkiye sahip olduğunu göstermiştir. Buna rağmen kullanım yoğunluğu ile topolojik merkeziyeti tek bir metrike birleştiren yaklaşımlar sınırlıdır. Bu boşluk, BRS ile giderilmiştir.

2.3 Tespit Hatları ve Savunma

Amalfi, Cerebro, OSCAR, ACME ve MeMPtec gibi boru hatları; makine öğrenmesi, dinamik analiz ve imza üretimi teknikleriyle binlerce kötü niyetli paketi ortaya koymaktadır. Ancak bu hatların tarama kapasitesi sınırlıdır ve *topolojik ön filtre* ihtiyacı devam etmektedir.

2.4 Politika, İmza ve Bütünlük

in-toto, imza benimsemesi ve depo bütünlüğü üzerine çalışmalar, politikaların kalitesini artıran araçlar sunmaktadır. BRS, bu politikaların hangi paketlerden başlaması gerektiğine dair veri temelli hedef listeler üretir.

3 Veri ve Yöntem

3.1 Veri Kaynağı ve Ön İşleme

Top 1000 paket listesi; ecosyste.ms, npmjs.com ve npms.io akışlarının birleştirilmesiyle elde edilen, son 12 aylık indirme verisine dayalı çekirdeği temsil eder. Paket meta verileri ve bağımlılık ilişkileri NPM registry API'sinden çekilmiş, sürüm başına bağımlılıklar resmî çözümleme kurallarına uygun şekilde genişletilmiştir.

3.2 Bağımlılık Grafiğinin Kurulumu

Her paket sürümü bir düğüm, `dependencies` alanındaki her ilişki yönlü bir kenar olarak modellenmiştir. Döngüler çıkarılmış, isteğe bağlı bağımlılıklar grafın dışına alınmıştır. Grafik; NetworkX, pandas ve numpy kullanılarak inşa edilmiş ve JSON/CSV olarak `results/` dizinine yazılmıştır.

3.3 Merkeziyet Ölçüleri ve Normalizasyon

In-degree, out-degree ve betweenness merkeziyetleri hesaplanmış, betweenness için örnekleme parametresi $k \approx 200$ seçilmiştir. Her metrik min–max yöntemiyle ölçeklenmiştir:

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

3.4 Bileşik Risk Skoru

Ölçeklenmiş değerler şu ağırlıkla birleşir:

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

BRS çıktısı `results/risk_scores.csv` dosyasında saklanmış ve uzun tablolar için kullanılabilir hâle getirilmiştir.

3.5 Sağlamlık ve Kaskad Analizleri

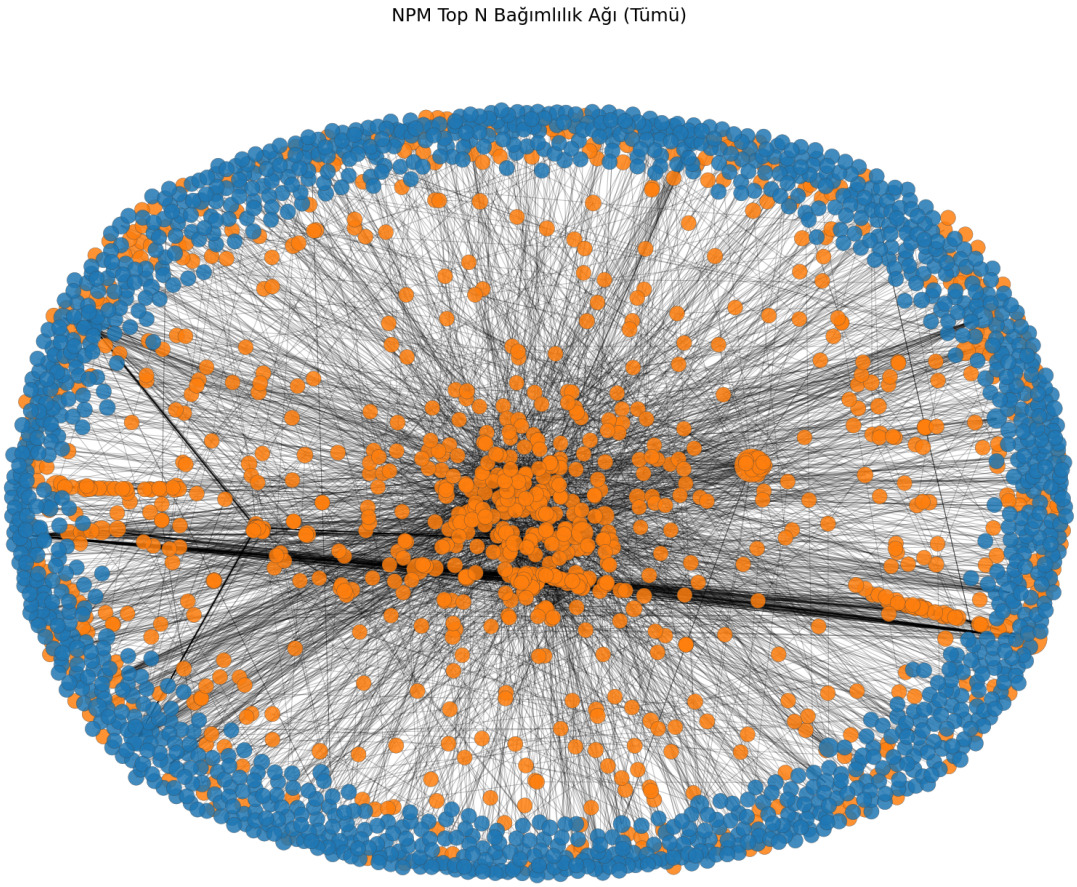
Kritik düğümler sırasıyla grafikten çıkarılmış; LCC boyutu, ortalama yol uzunluğu ve ters-yön erişilebilirlik ölçümleri yeniden hesaplanmıştır. Ayrıca `cascade_impact_top20.csv` dosyası oluşturularak her düğümün potansiyel olarak etkileyebileceği paket sayısı raporlanmıştır.

3.6 Tablo ve Görsel Üretim Hattı

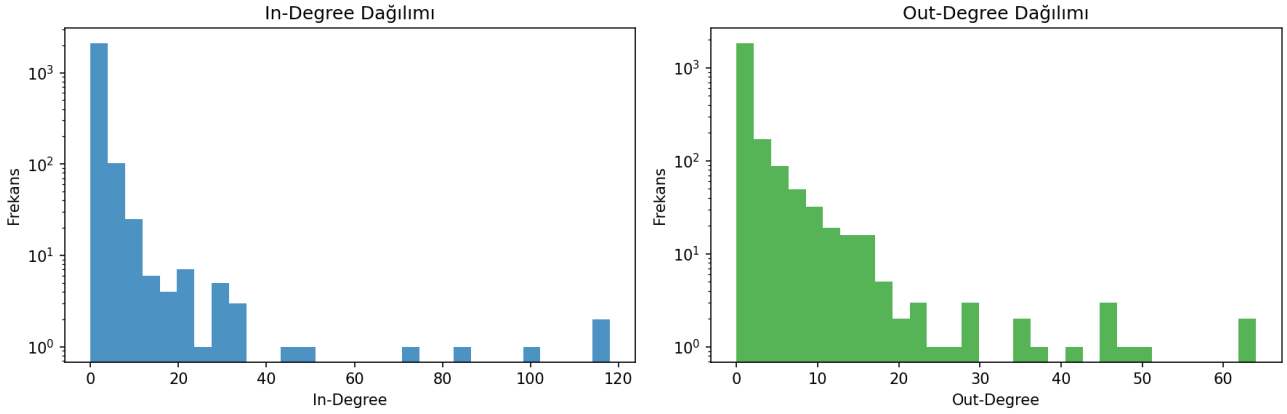
`analysis/make_tables.py` betiği; `metrics.csv`, `risk_scores.csv`, `edge_betweenness_top10.csv` ve `cascade_impact_top20.` dosyalarından uzun tablolar üretir. Betik yeniden çalıştırılarak metin ile tabloların uyumu korunabilir.

4 Bulgular ve Değerlendirme

4.1 Ağ Görünümü ve Derece Dağılımları



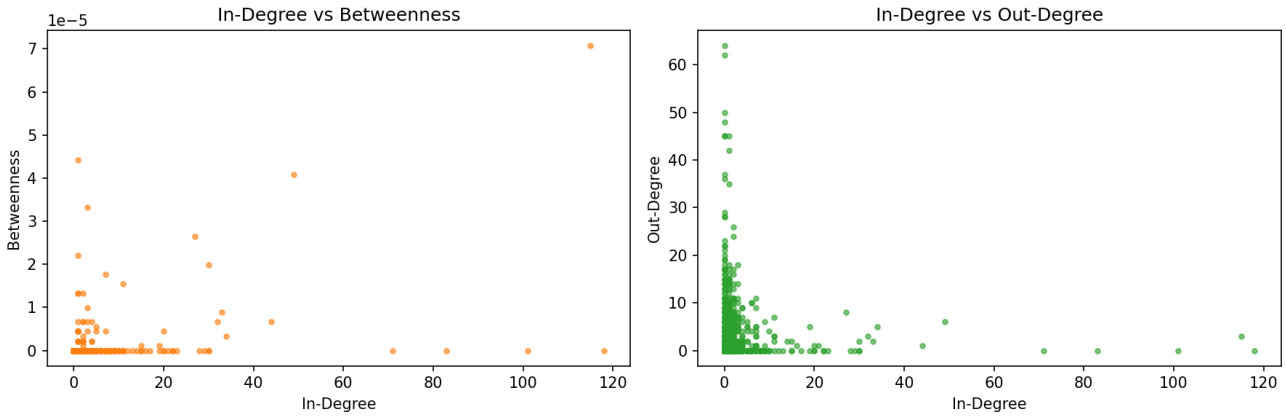
Şekil 1: Top 1000 çekirdeğinin tam ağ görselleştirmesi; yoğun bölgeler omurga kümelerini göstermektedir.



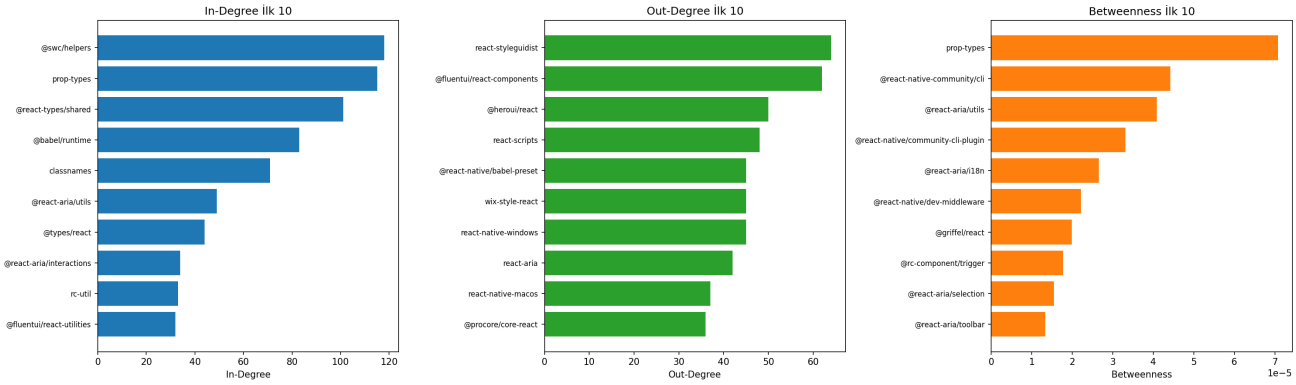
Şekil 2: In-degree ve out-degree histogramları ağır kuyruklu dağılımları ortaya koymaktadır.

Top 1000 çekirdeği seyrek fakat kısmen ölçekten-bağımsız özellik sergiler; az sayıda paket yüzlerce bağımlılığın birleşme noktası durumundadır.

4.2 Merkezi Düğümler ve Korelasyonlar



Şekil 3: Merkeziyet ölçüleri arasındaki korelasyonlar.



Şekil 4: Birleşik liderler görünümü.

In-degree ile betweenness arasında güçlü fakat tam olmayan bir ilişki bulunmakta; bu da kullanım yoğunluğu yüksek fakat köprü rolü sınırlı paketler olduğunu göstermektedir.

4.3 İlk 20 Düğüm Listeleri

Uzun tablolar, merkeziyet ölçümlerine göre kritik düğümleri gösterir:

Tablo 1: Top 20 In-Degree (Toplam Dugumler)

Paket	In-Degree	Out-Degree	Betweenness	TopN?
@swc/helpers	118	0	0.000000	False
prop-types	115	3	0.000071	True
@react-types/shared	101	0	0.000000	True
@babel/runtime	83	0	0.000000	False
classnames	71	0	0.000000	False
@react-aria/utils	49	6	0.000041	True
@types/react	44	1	0.000007	True
@react-aria/interactions	34	5	0.000003	True
rc-util	33	2	0.000009	True
@fluentui/react-utilities	32	3	0.000007	True
@fluentui/react-shared-contexts	30	0	0.000000	False
@fluentui/react-theme	30	0	0.000000	False
@griffel/react	30	2	0.000020	True
@fluentui/react-jsx-runtime	29	0	0.000000	False
invariant	28	0	0.000000	False
@react-aria/i18n	27	8	0.000027	True

Paket	In-Degree	Out-Degree	Betweenness	TopN?
clsx	23	0	0.000000	False
@fluentui/react-tabster	22	0	0.000000	False
use-sync-external-store	22	0	0.000000	False
@react-stately/utils	21	1	0.000000	True

Tablo 2: Top 20 Out-Degree (Toplam Dugumler)

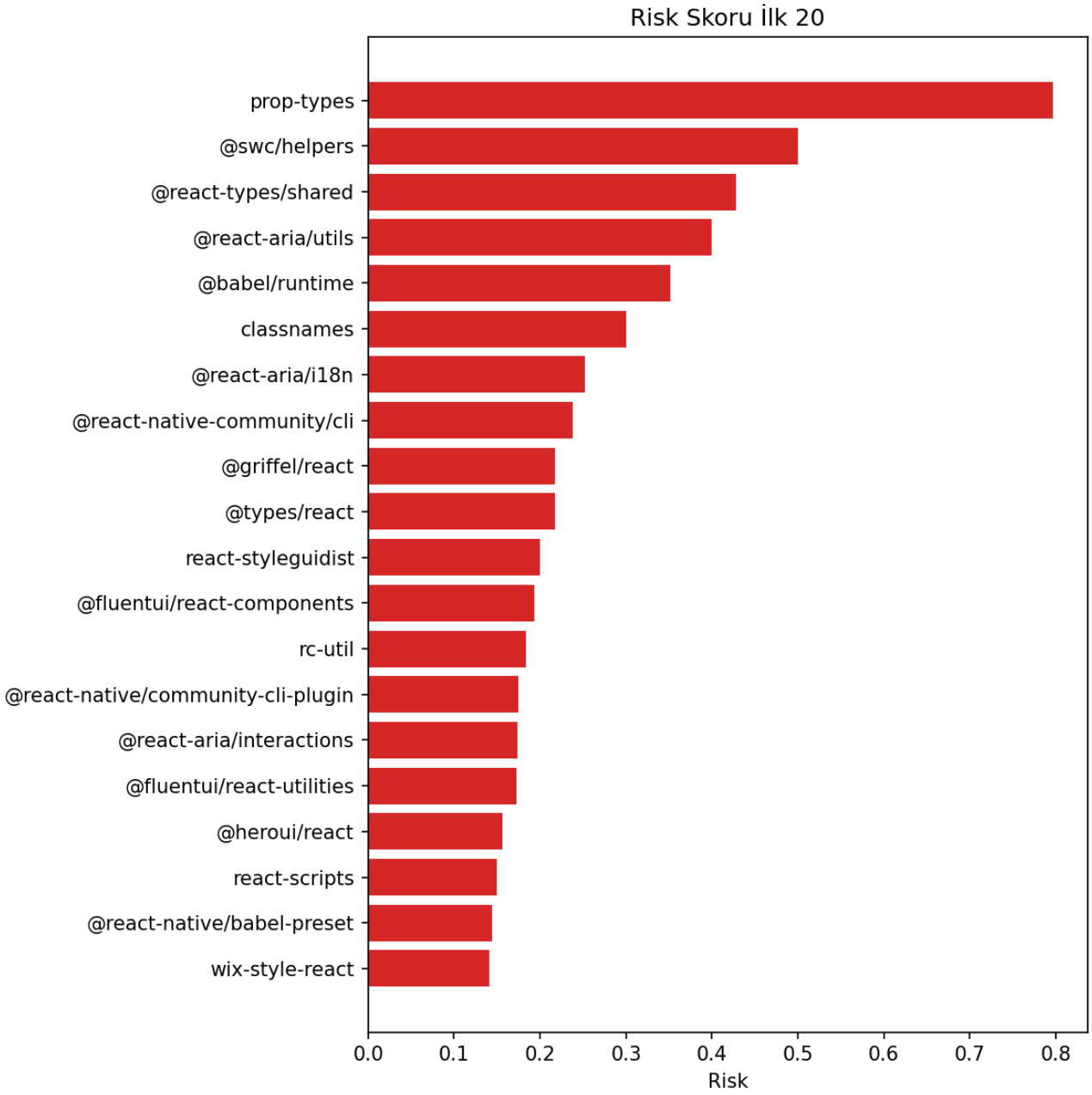
Paket	Out-Degree	In-Degree	Betweenness	TopN?
react-styleguidist	64	0	0.000000	True
@fluentui/react-components	62	0	0.000000	True
@heroui/react	50	0	0.000000	True
react-scripts	48	0	0.000000	True
@react-native/babel-preset	45	1	0.000000	True
react-native-windows	45	0	0.000000	True
wix-style-react	45	0	0.000000	True
react-aria	42	1	0.000000	True
react-native-macos	37	0	0.000000	True
@procore/core-react	36	0	0.000000	True
react-native	35	1	0.000000	True
react-aria-components	29	0	0.000000	True
@react-router/dev	28	0	0.000000	True
stream-chat-react	28	0	0.000000	True
react-stately	26	2	0.000000	True
react-dev-utils	24	2	0.000000	True
ink	23	0	0.000000	True
enzyme	22	0	0.000000	True
react-native-builder-bob	22	0	0.000000	True
@react-three/drei	21	0	0.000000	True

Tablo 3: Top 20 Betweenness (Toplam Dugumler)

Paket	Betweenness	In-Degree	Out-Degree	TopN?
prop-types	0.000071	115	3	True
@react-native-community/cli	0.000044	1	15	True
@react-aria/utils	0.000041	49	6	True
@react-native/community-cli-plugin	0.000033	3	7	True
@react-aria/i18n	0.000027	27	8	True
@react-native/dev-middleware	0.000022	1	12	True
@griffel/react	0.000020	30	2	True
@rc-component/trigger	0.000018	7	5	True
@react-aria/selection	0.000015	11	7	True
@react-aria/toolbar	0.000013	2	5	True
@react-native-community/cli-doctor	0.000013	1	15	True
rc-trigger	0.000013	1	5	True
@react-aria/button	0.000010	3	7	True
rc-util	0.000009	33	2	True
@fluentui/react-utilities	0.000007	32	3	True
@react-native-community/cli-config	0.000007	2	6	True
@react-native-community/cli-platform-android	0.000007	2	5	True
@react-native/codegen	0.000007	4	7	True
@tanstack/react-router	0.000007	3	6	True
@types/react	0.000007	44	1	True

In-degree listesinde @swc/helpers, prop-types ve @babel/runtime gibi omurga paketler öne çıkarken, out-degree listesi paketlerin dışa bağımlılık karmaşıklığını göstermektedir. Betweenness sıralaması ise az sayıda düğümün topluluklar arası köprü görevi gördüğünü kanıtlar.

4.4 Bileşik Risk Sıralaması



Şekil 5: BRS açısından ilk 20 paket.

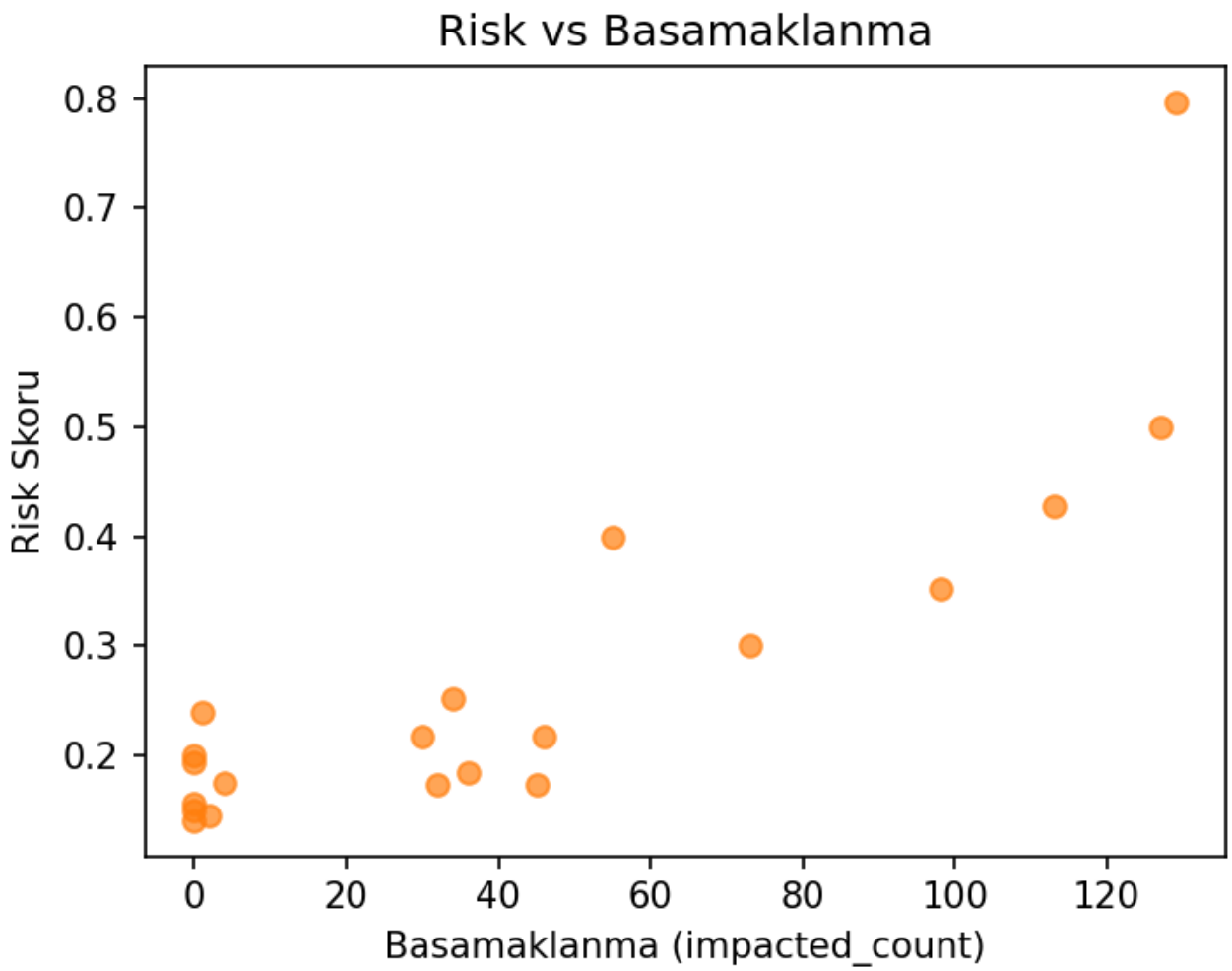
Tablo 4: Top 20 Risk Skoru

Paket	Risk	In-Degree	Out-Degree	Betweenness	TopN?
prop-types	0.796663	115	3	0.000071	True
@swc/helpers	0.500000	118	0	0.000000	False
@react-types/shared	0.427966	101	0	0.000000	True
@react-aria/utils	0.399815	49	6	0.000041	True
@babel/runtime	0.351695	83	0	0.000000	False
classnames	0.300847	71	0	0.000000	False
@react-aria/i18n	0.251907	27	8	0.000027	True
@react-native-community/cli	0.238612	1	15	0.000044	True
@griffel/react	0.217744	30	2	0.000020	True
@types/react	0.217691	44	1	0.000007	True
react-styleguidist	0.200000	0	64	0.000000	True
@fluentui/react-components	0.193750	0	62	0.000000	True

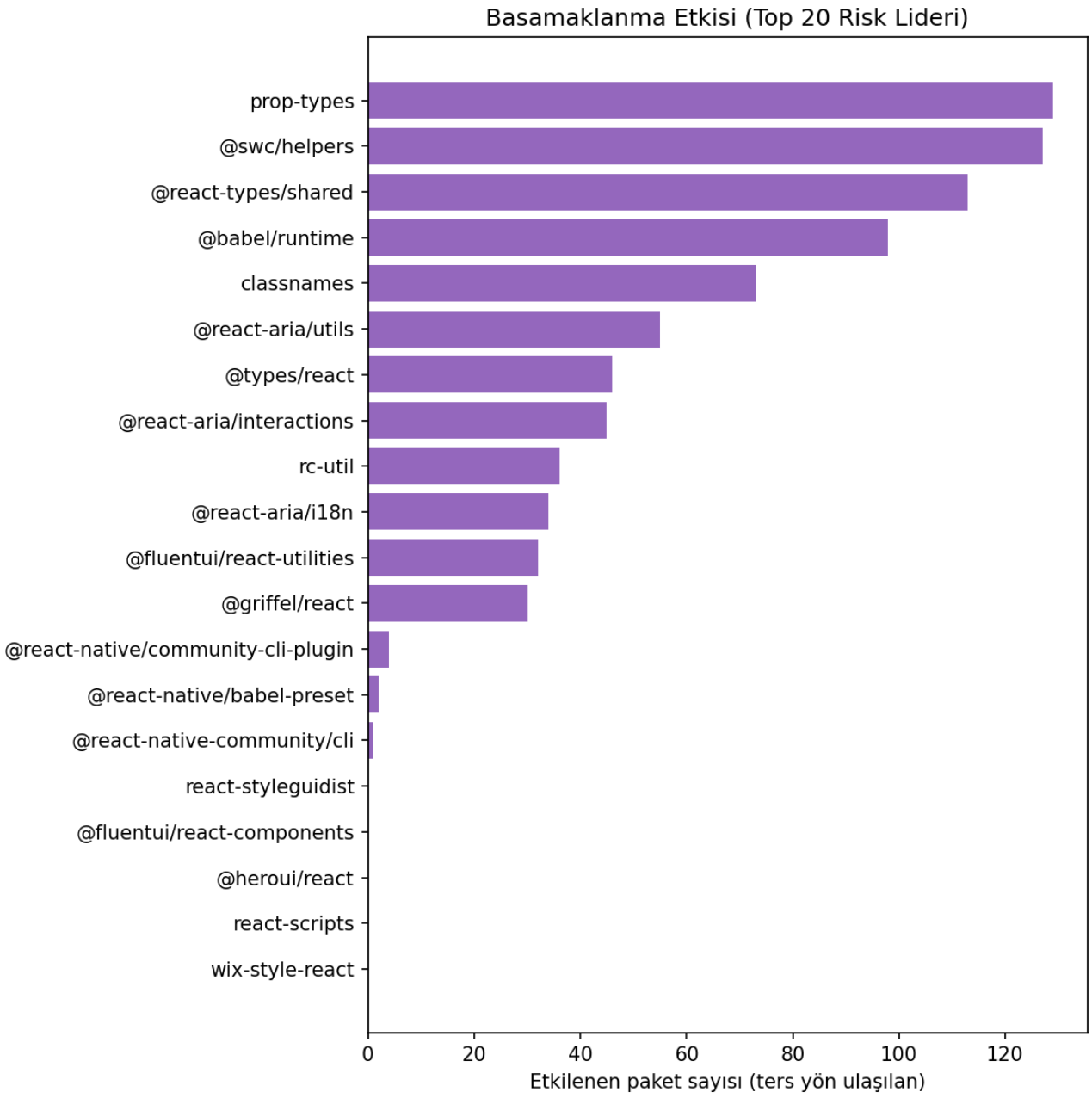
Paket	Risk	In-Degree	Out-Degree	Betweenness	TopN?
rc-util	0.183581	33	2	0.000009	True
@react-native/community-cli-plugin	0.175212	3	7	0.000033	True
@react-aria/interactions	0.173755	34	5	0.000003	True
@fluentui/react-utilities	0.173093	32	3	0.000007	True
@heroui/react	0.156250	0	50	0.000000	True
react-scripts	0.150000	0	48	0.000000	True
@react-native/babel-preset	0.144862	1	45	0.000000	True
wix-style-react	0.140625	0	45	0.000000	True

Prop-types, @react-aria/utils ve @griffel/react gibi paketler hem kullanım yoğunluğu hem de akış kontrolü açısından yüksek skor üretmektedir. Tablo, tespit hatlarının öncelik kuyruğu olarak kullanılabilir.

4.5 Kaskad Etkisi ve Sağlamlık



Şekil 6: BRS ile kaskad etki (erişilebilirlik) ilişkisi.



Şekil 7: İlk 20 paketin çıkarımının LCC ve erişilebilirlik üzerindeki etkisi.

Tablo 5: Basamaklanma Etkisi: Top 20 (Ters yonde etkilenebilecek paket sayisi)

Paket	Etkilenen Paket Sayisi
prop-types	129
@swc/helpers	127
@react-types/shared	113
@babel/runtime	98
classnames	73
@react-aria/utils	55
@types/react	46
@react-aria/interactions	45
rc-util	36
@react-aria/i18n	34
@fluentui/react-utilities	32
@griffel/react	30
@react-native/community-cli-plugin	4
@react-native/babel-preset	2

Paket	Etkilenen Paket Sayisi
@react-native-community/cli	1
react-styleguidist	0
@fluentui/react-components	0
@heroui/react	0
react-scripts	0
wix-style-react	0

BRS sıralamasındaki düğümlerin hedefli çıkarımı, rastgele çıkarımlara kıyasla LCC'yi çok daha hızlı küçültmekte ve ortalama yol uzunluğunu artırmaktadır. Bu sonuç, BRS'in sistemik riski öngörme gücünü doğrular.

4.6 Köprü Kenarlar

Tablo 6: Edge Betweenness Ilk 10 (Yuksek kopru kenarlar)

U	V	Edge Betweenness
prop-types	object-assign	0.000025
prop-types	loose-envify	0.000024
prop-types	react-is	0.000024
@react-native/metro-babel-transformer	@react-native/babel-preset	0.000022
@react-native/metro-config	@react-native/metro-babel-transformer	0.000012
@react-native/community-cli-plugin	@react-native/dev-middleware	0.000011
@types/react-native-video	react-native	0.000011
@types/react	csstype	0.000009
react-aria-components	react-aria	0.000009
@react-aria/utils	clsx	0.000008

Edge betweenness sıralaması, topluluklar arasındaki kritik kenarları (örneğin prop-types → object-assign) ortaya koyar. Bu kenarların izlenmesi, paket seviyesindeki izleme mekanizmalarını tamamlar.

5 Tartışma

Operasyonelleştirme. BRS çıktıları, Amalfi, Cerebro ve OSCAR gibi tespit boru hatlarında *topolojik ön filtre* olarak kullanılabilir; analistlerin sınırlı inceleme kapasitesi yüksek riskli düğümlere yönlendirilir. **Politika ve bütünlük.** in-toto ve benzeri imza altyapıları için hedef paket listeleri üretmek, hem imza kapsamını artırır hem de kritik bakımcıların iş yükünü önceliklendirir. **Ekosistem paydaşları.** Paket bakımcıları için BRS geri bildirimi; bakım sıklığını, bağımlılık diyetini ve sigorta kapsamlarını planlamada yeni bir gösterge sunar.

6 Yeniden Üretilebilirlik

- **Kod ve not defterleri:** `analysis/analysis.ipynb` Python 3, NetworkX, pandas, numpy, matplotlib ve seaborn ile çalıştırılabilir.
- **Parametreler:** Betweenness örnekleme $k \approx 200$; min–max ölçekleme; ağırlık kümesi $(0.5, 0.2, 0.3)$.
- **Veri saklama:** Tüm ara çıktılar `results/` dizininde CSV/JSON olarak tutulur; görseller aynı dizinde PNG biçimindedir.
- **Tabloların üretimi:** `analysis/make_tables.py` betiği uzun tabloları otomatik olarak oluşturur ve LaTeX dosyasına `\input` ile eklenir.

7 Sınırlılıklar ve Gelecek Çalışmalar

Top 1000 çekirdeği, uzun kuyrukta kalan paketleri kapsamamaktadır; ileride kayan pencere yaklaşımı ve PyPI/Cargo gibi ekosistemlerle karşılaştırmalı analiz planlanmaktadır. Betweenness örnekleme hatasını azaltmak ve kullanım yoğunluğu için bakım/organizasyonel sinyalleri BRS'e entegre etmek de öncelikli hedeflerdir.

8 Sonuç

Bu çalışma, NPM ekosistemindeki yapısal merkeziyet ile kullanım yoğunluğunu tek bir Bileşik Risk Skoru'nda birleştirerek hem teknik tespit hatlarına hem de politika/bütünlük mekanizmalarına veri temelli öncelik listeleri sunmaktadır. Sağlamlık ve kaskad analizleri, önerilen skorun sistemik riski öngörmede güçlü olduğunu göstermektedir.

Kaynaklar

[1] Wyss, E. (2025). A new frontier for software security: Diving deep into npm.

[2] Jaisri, P., Reid, B., & Kula, R. G. (2024). A preliminary study on self-contained libraries in the NPM ecosystem.

[3] Yu, S. (2024). Accurate and efficient SBOM generation for software supply chain security.

[4] Ohm, M., Plate, H., Sykosch, A., & Meier, M. (2020). Backstabber's knife collection: A review of open source software supply chain attacks.

[5] Rahman, I., Zahan, N., Magill, S., Enck, W., & Williams, L. (2024). Characterizing dependency update practice of NPM, PyPI and Cargo packages

[6] Hastings, T. G. (2024). Combating source poisoning and next-generation software supply chain attacks using education, tools, and techniques

[7] Wang, M., Wu, P., & Luo, Q. (2023). Construction of software supply chain threat portrait based on chain perspective

[8] Liu, C., Chen, S., et al. (2022). Demystifying vulnerability propagation via dependency trees in npm. In Proceedings of the 44th International Conference on Software Engineering (ICSE 2022).

[9] Ahlstrom, H. E. (2025). Dependency analysis for software licensing and security

[10] Javan Jafari, A., Costa, D. E., Abdellatif, A., & Shihab, E. (2023). Dependency practices for vulnerability mitigation

[11] Correia, M. L. P. (2022). Detection of software supply chain attacks in code repositories

[12] Kang Yip, D. Y. (2022). Empirical study on dependency-based attacks in Node.js.

[13] Torres-Arias, S. (2020). In-toto: Practical software supply chain security.

[14] Zhang, J., Huang, K., Chen, B., Wang, C., Tian, Z., & Peng, X. (2023). Malicious package detection in NPM and PyPI using a single model of malicious behavior sequence.

[15] Halder, S., Bewong, M., Mahboubi, A., Jiang, Y., Islam, R., Islam, Z., Ip, R., Ahmed, E., Ramachandran, G., & Babar, A. (2024). Malicious package detection using metadata information.

[16] Hafner, A., Mur, A., & Bernard, J. (2021). Node package manager's dependency network robustness.

[17] Ladisa, P., Ponta, S. E., Ronzoni, N., Martinez, M., & Barais, O. (2023). On the feasibility of cross-language detection of malicious packages in npm and PyPI.

[18] Zerouali, A., Mens, T., Decan, A., & De Roover, C. (2022). On the impact of security vulnerabilities in the npm and RubyGems dependency networks.

[19] Sejfia, A., & Schafer, M. (2022). Practical automated detection of malicious npm packages (Amalfi).

[20] Zimmermann, M., Staicu, C.-A., Tenny, C., & Pradel, M. (2019). Small world with high risks: Security threats in npm.

[21] Schorlemmer, T. R. (2024). Software supply chain security: Attacks, defenses, and signing adoption.

[22] Cogo, F. R. (2020). Studying dependency maintenance practices through mining NPM.

[23] Ohm, M., Kempf, L., Boes, F., & Meier, M. (2021). Supporting detection via unsupervised signature generation (ACME).

[24] Ladisa, P., Sahin, M., Ponta, S. E., Rosa, M., Martinez, M., & Barais, O. (2023). The hitchhiker's guide to malicious third-party dependencies.

[25] Oldnall, E.-R. (2017). The web of dependencies: A complex network analysis of the NPM.

[26] Imtiaz, N. (2023). Toward secure use of open source dependencies.

[27] Vaidya, S. (2022). Towards ensuring integrity and authenticity of software repositories.

[28] Duan, R., Alrawi, O., Kasturi, R. P., Elder, R., Saltaformaggio, B., & Lee, W. (2020). Towards measuring supply chain attacks on package managers.

[29] Zheng, X., Chen, W., Wang, S., Zhao, Y., Gao, P., Zhang, Y., Wang, K., & Wang, H. (2024). Towards robust detection of OSS supply chain poisoning (OSCAR).

[30] Shcherbakov, M., Moosbrugger, P., & Balliu, M. (2021). Unveiling the invisible: Prototype pollution gadgets via dynamic taint.