NPM Ekosisteminde Yönlü Karmaşık Ağ Analizi

Yusuf Talha ARABACI

21 Ekim 2025

Özet

Bu çalışma, NPM ekosistemindeki popüler Top N paketi, bağımlılık ilişkilerine göre yönlü bir karmaşık ağ (Dependent → Dependency) olarak modellemekte ve yapısal riskleri merkeziyet metrikleriyle incelemektedir. Veri, her çalıştırmada API'lerden (öncelikle ecosyste.ms; yedek olarak npm registry ve npms.io) çekilmektedir. Ağ, NetworkX ile kurulmakta; in-degree, out-degree ve betweenness merkeziyet metrikleri hesaplanmaktadır. Büyük graflarda betweenness hesabı örnekleme (k) ile hızlandırılmaktadır. Çalışma ayrıca bileşik bir risk skoru (normalize edilmiş in/out/between ağırlıklı toplamı) ve risk tabanlı sağlamlık (robustluk) analizi (kritik düğümlerin çıkarılması) önermektedir. Üretilen tüm çıktılar (CSV/JSON ve PNG/SVG görseller) `results/` dizininde saklanır.

1 Giriş

Yazılım tedarik zinciri saldırılarında (SSCA), tek bir bağımlılığın ele geçirilmesi geniş çapta zincirleme etkilere yol açabilir. NPM ekosistemi, yoğun bağımlılık ilişkilerine sahip olup, paketlerin yapısal konumuna göre sistemik risk taşıyabilmektedir. Bu çalışma, Top N paket üzerinden inşa edilen yönlü bağımlılık ağı ile aşağıdaki sorulara odaklanır:

- Hangi düğümler (paketler) yapısal olarak kritik (yüksek in-degree, yüksek betweenness)?
- Hangi düğümler geniş bağımlılık yüzeyine sahip (yüksek out-degree)?
- Merkeziyetlere dayalı bileşik bir risk skoru ile risk liderleri nasıl sıralanır?
- Kritik düğümler çıkarıldığında ağın bağlanırlılığı nasıl değişir (robustluk)?

2 Amaç

Amaç, yazılım tedarik zinciri güvenliğini paket-içi zafiyetlerin ötesine taşıyarak, bağımlılık ağı topolojisini de hesaba katan yapısal bir ölçüt geliştirmektir. NPM paketleri yönlü bir karmaşık ağ olarak modellenerek, her paketin ağ içindeki yapısal önemi, ele geçirilmesi durumunda yaratabileceği basamaklanma (cascading) etkisi ve bunun sistemik risk üzerindeki nicel etkileri ölçülebilir.

3 Veri ve Yöntem

Top N paket listesi, her çalıştırmada API'lerden çekilir (öncelik ecosyste.ms; yedek olarak npm registry ve npms.io). Bağımlılıklar, npm registry'de paketlerin en güncel sürümlerinin `dependencies` alanından alınır; isteğe bağlı `peerDependencies` de eklenebilir. Yönlü ağ, Dependent → Dependency yönüyle kurulur. Büyük graflarda betweenness örneklemeli (k) hesaplanır.

4 Ağ Özeti

Tablo 1: Graf İstatistikleri (özet)

Ölçüt	Değer
Düğüm sayısı	1139
Kenar sayısı	2164
Bileşen sayısı (zayıf)	160
En büyük bileşen boyutu	853
Ortalama in-degree	1.8999
Ortalama out-degree	1.8999

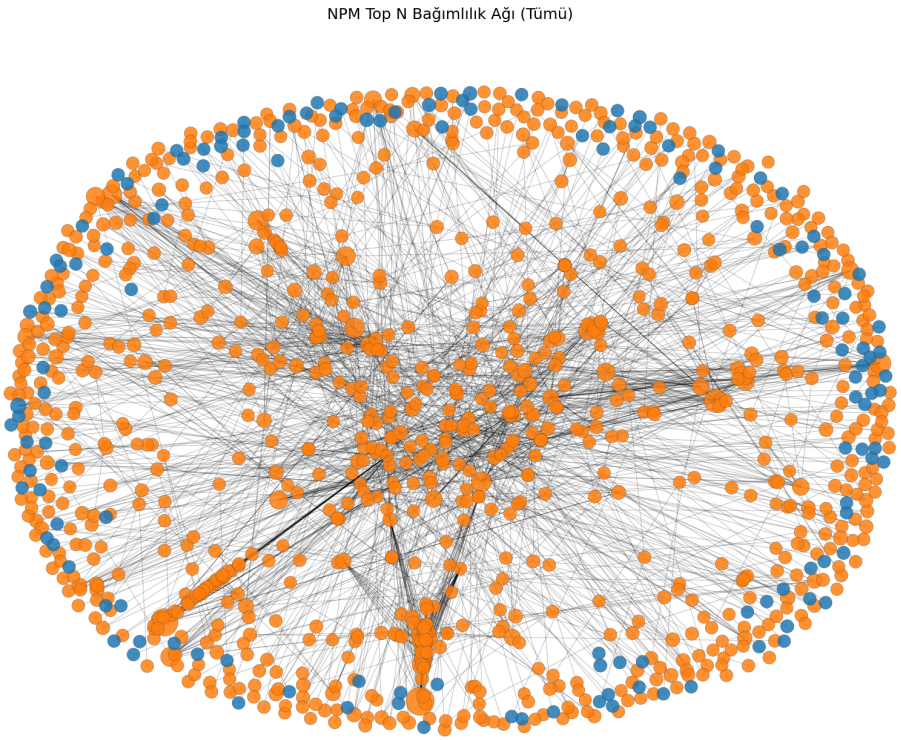
5 Ağ Modeli ve Metrikler

Model, NetworkX ile kurulmuş bir `DiGraph` yapısıdır. Temel metrikler: (i) *In-degree:* düğüme gelen kenar sayısı (pakete dayanan paket sayısı); (ii) *Out-degree:* düğümün dış bağımlılık sayısı; (iii) *Betweenness:* en kısa yollardaki aracılık (köprü) rolü.

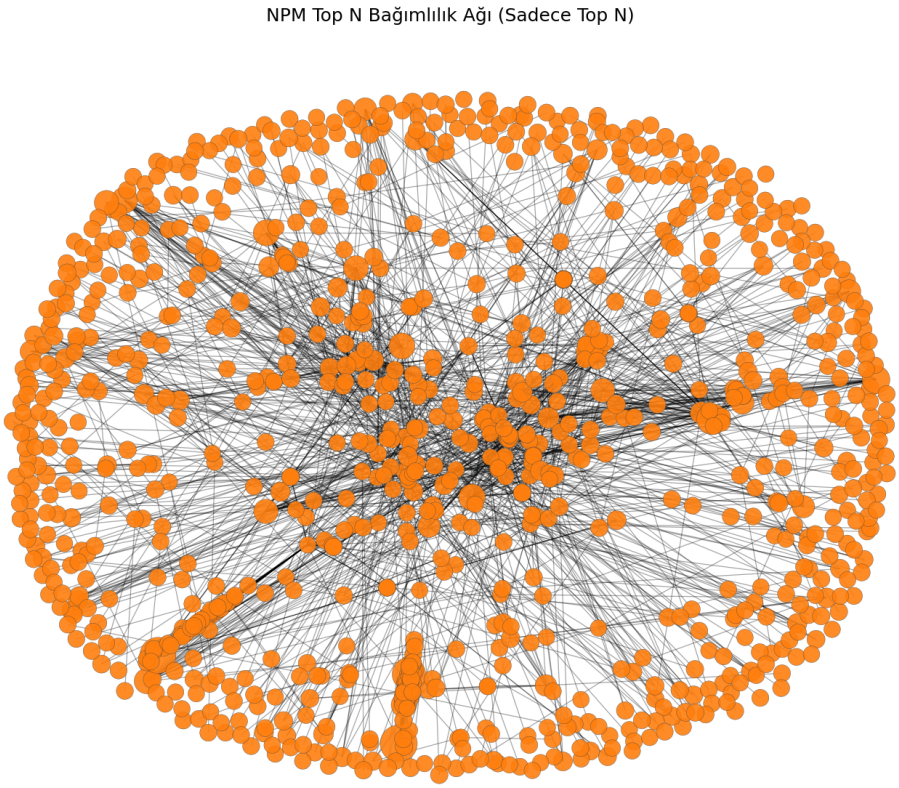
6 Bulgular

Bu bölümde, `results/` dizinindeki çıktılar kullanılarak görsel ve tablolu özetler sunulmaktadır.

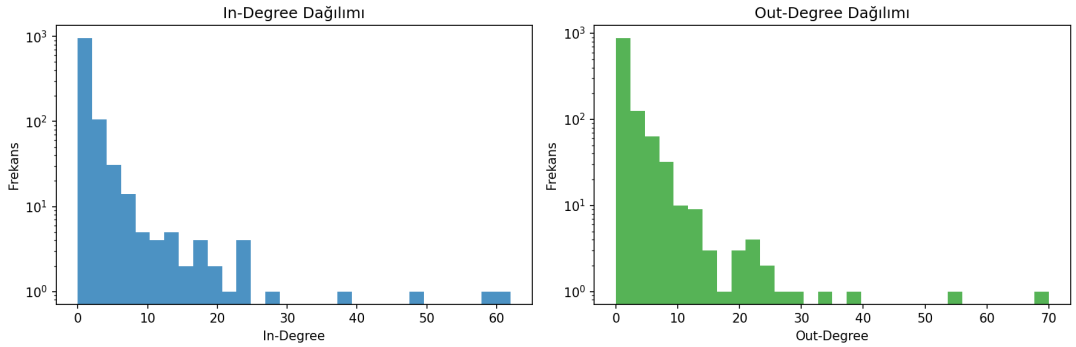
6.1 Ağ Görselleştirmeleri



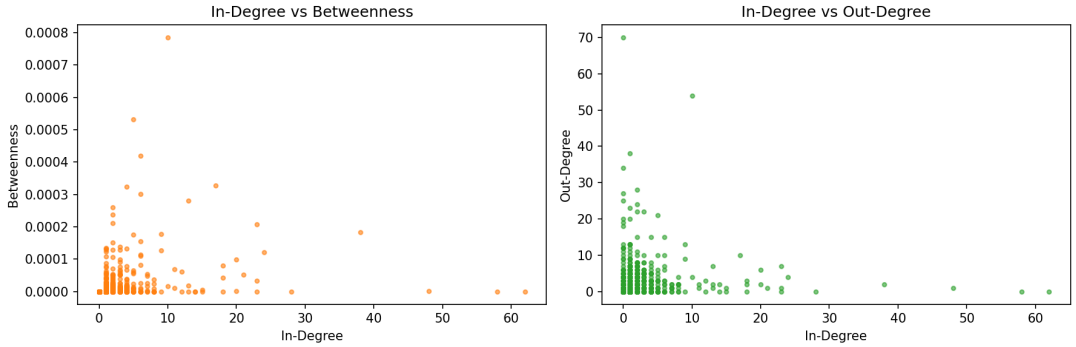
Şekil 1: Top N + bağımlılıkların oluşturduğu yönlü ağ (düğüm boyutu: in-degree; renk: Top N turuncu / diğerleri mavi).



Şekil 2: Sadece Top N düğümlerin indüklenmiş alt-ağı.



Şekil 3: In-degree ve out-degree histogramları (log ölçek).

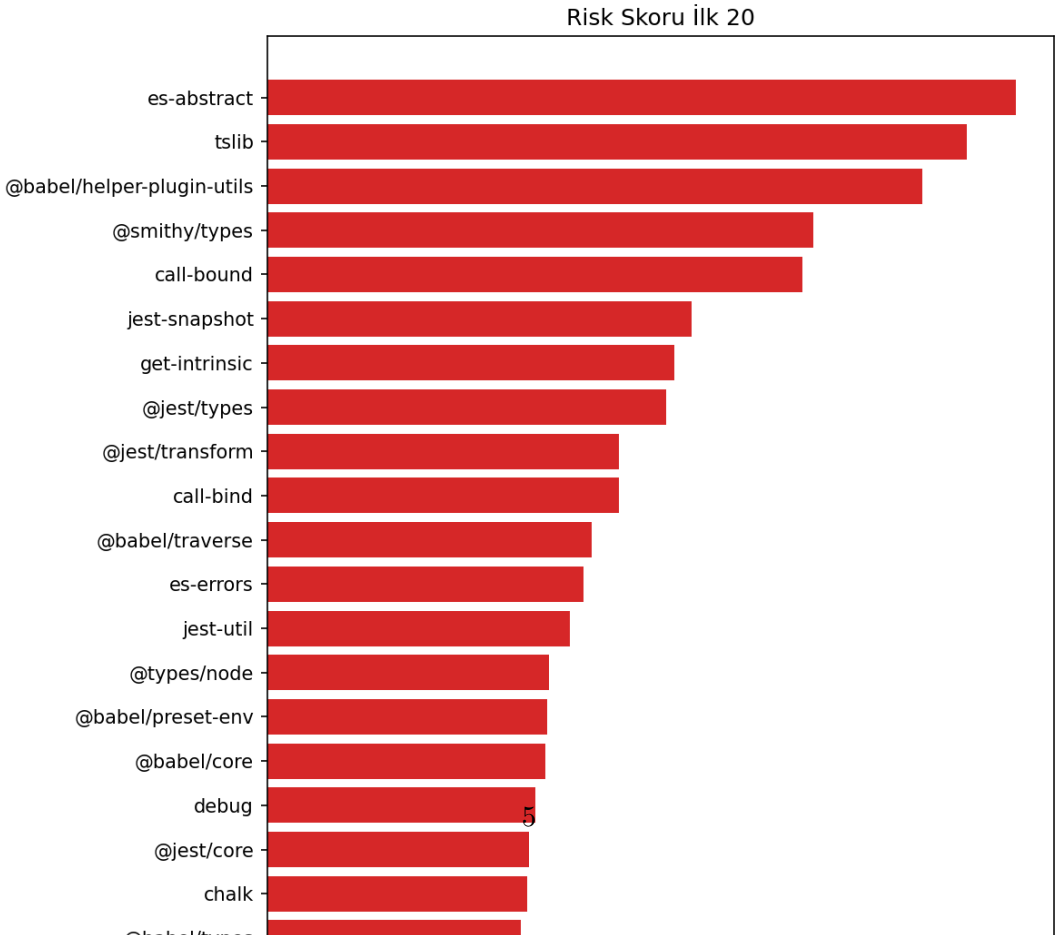


Şekil 4: Korelasyonlar: In-degree vs Betweenness (solda), In-degree vs Out-degree (sağda).

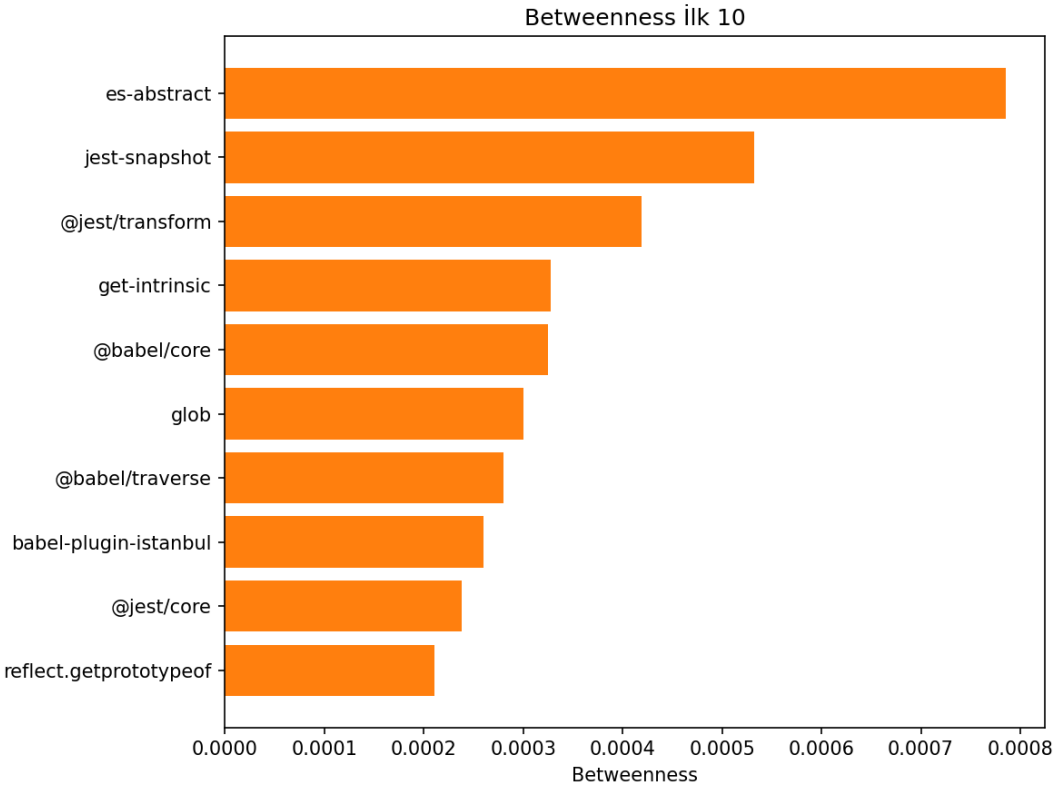
6.2 Derece Dağılımları ve Korelasyonlar

6.3 Merkeziyet Liderleri

6.4 Risk Skoru ve Robustluk



Şekil 5: İlk 10 In-degree (sol) ve Out-degree (sağ).



Şekil 6: İlk 10 Betweenness.

Kenar Betweenness İlk 10. Aşağıdaki tablo, en yüksek *edge betweenness* değerine sahip 10 kenarı gösterir.

Tablo 2: Edge Betweenness Ilk 10 (Yuksek kopru kenarlar)

U	V	Edge Betweenness
@jest/transform	babel-plugin-istanbul	0.000222
@jest/expect	jest-snapshot	0.000212
jest-snapshot	@jest/transform	0.000206
jest	@jest/core	0.000150
call-bound	get-intrinsic	0.000150
glob	jackspeak	0.000147
reflect.getprototypeof	which-builtin-type	0.000146
jackspeak	@isaacs/cliui	0.000140
babel-plugin-istanbul	test-exclude	0.000139
@babel/core	@babel/helper-compilation-targets	0.000138

Risk Skoru İlk 20. Bileşik risk skoruna göre ilk 20 paket.

Tablo 3: Top 20 Risk Skoru

Paket	Risk	In-Degree	Out-Degree	Betweenness	TopN?
es-abstract	0.534931	10	54	0.000785	True
tslib	0.500000	62	0	0.000000	True
@babel/helper-plugin-utils	0.467742	58	0	0.000000	True
@smithy/types	0.390249	48	1	0.000001	True
call-bound	0.382129	38	2	0.000183	True
jest-snapshot	0.303511	5	21	0.000532	True
get-intrinsic	0.290993	17	10	0.000328	True
@jest/types	0.284735	23	7	0.000207	True
@jest/transform	0.251456	6	15	0.000419	True
call-bind	0.251171	24	4	0.000121	True
@babel/traverse	0.231984	13	7	0.000280	True
es-errors	0.225806	28	0	0.000000	True
jest-util	0.216164	20	6	0.000099	True
@types/node	0.201331	23	1	0.000034	True
@babel/preset-env	0.200000	0	70	0.000000	True
@babel/core	0.199075	4	15	0.000324	True
debug	0.191845	21	1	0.000051	True
@jest/core	0.187008	2	28	0.000238	True
chalk	0.185484	23	0	0.000000	True
@babel/types	0.181088	18	2	0.000079	True

Dereceye Göre İlk 20. In/Out/Betweenness sıralamaları.

Tablo 4: Top 20 In-Degree (Toplam Dugumler)

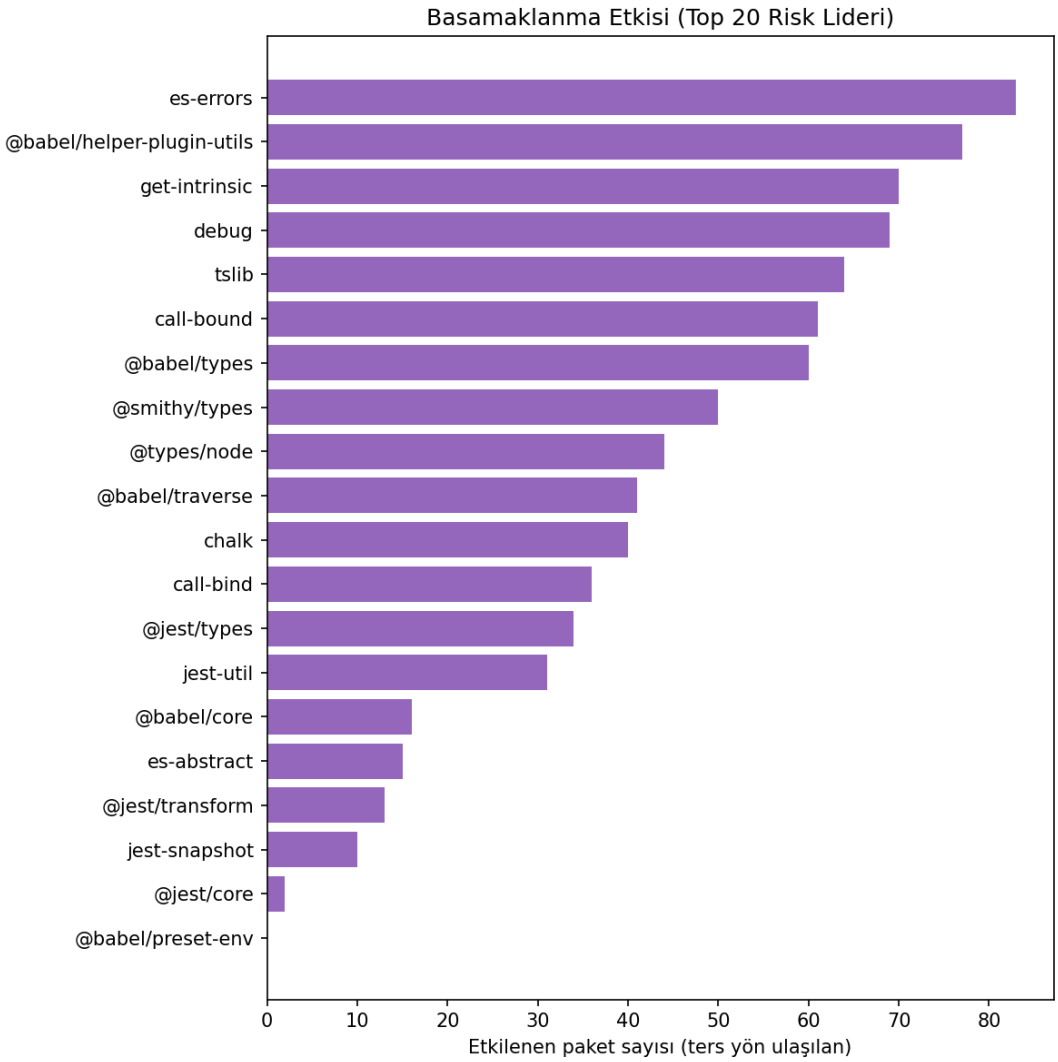
Paket	In-Degree	Out-Degree	Betweenness	TopN?
tslib	62	0	0.000000	True
@babel/helper-plugin-utils	58	0	0.000000	True
@smithy/types	48	1	0.000001	True
call-bound	38	2	0.000183	True
es-errors	28	0	0.000000	True
call-bind	24	4	0.000121	True
@jest/types	23	7	0.000207	True
@types/node	23	1	0.000034	True
chalk	23	0	0.000000	True
debug	21	1	0.000051	True
@aws-sdk/types	20	2	0.000002	True
jest-util	20	6	0.000099	True
@babel/types	18	2	0.000079	True
define-properties	18	3	0.000043	True
graceful-fs	18	0	0.000000	True
get-intrinsic	17	10	0.000328	True
es-object-atoms	15	1	0.000006	True
gopd	15	0	0.000000	True
@smithy/protocol-http	14	2	0.000000	True
semver	14	0	0.000000	True

Tablo 5: Top 20 Out-Degree (Toplam Dugumler)

Paket	Out-Degree	In-Degree	Betweenness	TopN?
@babel/preset-env	70	0	0.000000	True
es-abstract	54	10	0.000785	True
@aws-sdk/client-sso	38	1	0.000075	True
eslint	34	0	0.000000	True
@jest/core	28	2	0.000238	True
express	27	0	0.000000	True
webpack	25	0	0.000000	True
jest-config	24	2	0.000128	True
@jest/reporters	23	1	0.000039	True
jest-runner	22	2	0.000042	True
jest-runtime	22	3	0.000129	True
jest-snapshot	21	5	0.000532	True
jest-circus	20	1	0.000040	True
jsdom	20	0	0.000000	True
eslint-plugin-import	19	0	0.000000	True
eslint-plugin-react	18	0	0.000000	True
@babel/core	15	4	0.000324	True
@jest/transform	15	6	0.000419	True
babel-preset-current-node-syntax	15	2	0.000151	True
@aws-sdk/core	13	9	0.000127	True

Tablo 6: Top 20 Betweenness (Toplam Dugumler)

Paket	Betweenness	In-Degree	Out-Degree	TopN?
es-abstract	0.000785	10	54	True
jest-snapshot	0.000532	5	21	True
@jest/transform	0.000419	6	15	True
get-intrinsic	0.000328	17	10	True
@babel/core	0.000324	4	15	True
glob	0.000301	6	6	True
@babel/traverse	0.000280	13	7	True
babel-plugin-istanbul	0.000260	2	5	True
@jest/core	0.000238	2	28	True
reflect.getprototypeof	0.000211	2	8	True
@jest/types	0.000207	23	7	True
call-bound	0.000183	38	2	True
jest-message-util	0.000178	9	9	True
@babel/helper-compilation-targets	0.000176	5	5	True
jest-haste-map	0.000156	6	10	True
babel-preset-current-node-syntax	0.000151	2	15	True
@babel/generator	0.000139	3	5	True
which-builtin-type	0.000134	1	13	True
browserslist	0.000133	4	5	True
jackspeak	0.000132	1	1	True



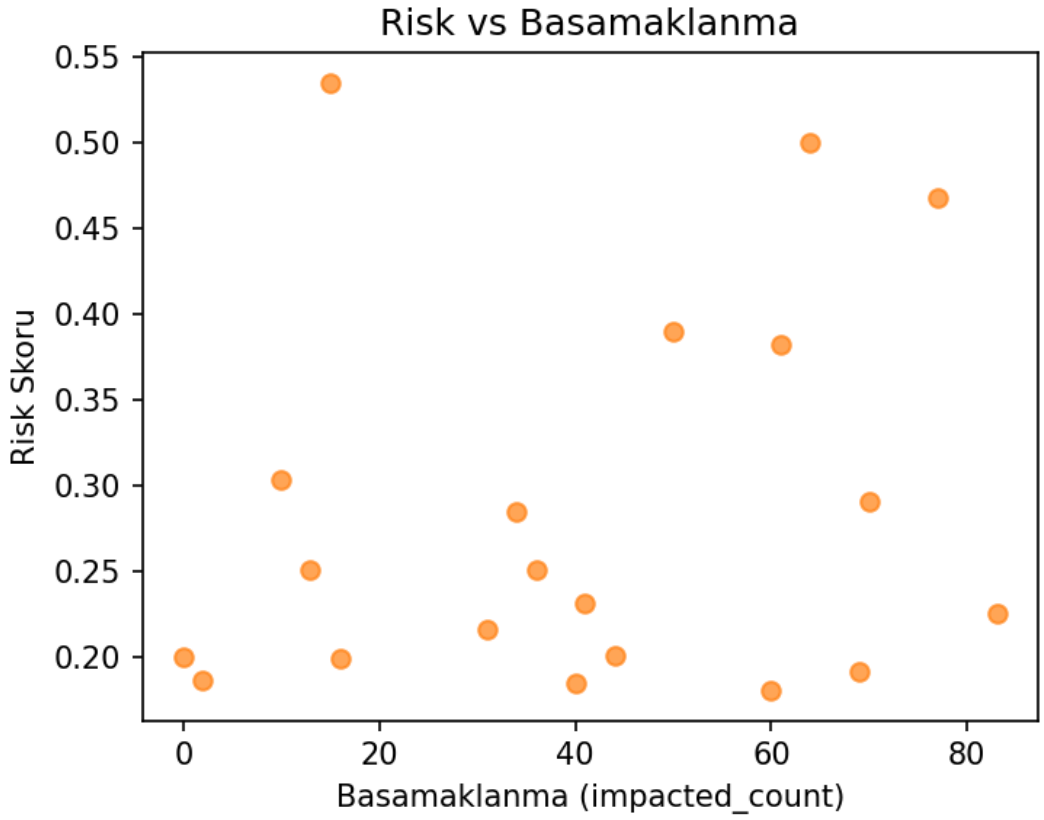
Şekil 8: Risk skoruna göre en riskli 20 paket için basamaklanma (cascading impact) büyüklüğü.

Tablo 7: Basamaklanma Etkisi: Top 20 (Ters yonde etkilenebilecek paket sayisi)

Paket	Etkilenen Paket Sayisi
es-errors	83
@babel/helper-plugin-utils	77
get-intrinsic	70
debug	69
tslib	64
call-bound	61
@babel/types	60
@smithy/types	50
@types/node	44
@babel/traverse	41
chalk	40
call-bind	36
@jest/types	34
jest-util	31

Paket	Etkilenen Paket Sayisi
@babel/core	16
es-abstract	15
@jest/transform	13
jest-snapshot	10
@jest/core	2
@babel/preset-env	0

Risk–Basamaklanma İlişkisi. Ek olarak, risk skoru ile basamaklanma etkisi arasındaki ilişki incelenmiştir.



Şekil 9: Risk skoru ile basamaklanma etkisi arasındaki ilişki (scatter).

7 Öne Çıkan Paketler (Kısa Açıklamalar)

Aşağıda, ağda yapısal olarak öne çıkan ve sık karşılaşılan 20 paket için kısa açıklamalar verilmiştir. Liste, geniş kullanım alanları nedeniyle tedarik zinciri riskine işaret eden altyapı kütüphanelerini vurgular.

1. `es-abstract`: ECMAScript spesifikasyonundaki soyut işlemleri uygular; çok sayıda paketin altyapısında kullanılır.
2. `tslib`: TypeScript derleyicisinin çalışma zamanı yardımcılarını içerir; yaygın kullanımı nedeniyle kritik olabilir.

3. `@babel/helper-plugin-utils`: Babel eklentileri geliştirmek için yardımcı fonksiyonlar sağlar; ekosistemde geniş bağımlılık ağına sahiptir.
4. `@smithy/types`: AWS SDK (v3) içindeki tip altyapısı; büyük ölçekli projelerde zincir etkisine adaydır.
5. `call-bound`: JS *intrinsic* metotları güvenli biçimde çağırmak/bağlamak için yardımcı; çok sayıda modül tarafından kullanılır.
6. `jest-snapshot`: Jest için *snapshot* testleri; test zincirinde kritik, dolaylı etkisi olabilir.
7. `get-intrinsic`: ECMAScript *intrinsic* nesnelerini güvenli elde etmek için yardımcı; alt katmanda yaygın kullanılır.
8. `@jest/types`: Jest ekosisteminde ortak tip altyapısı; modüller arası etkileşimi artırır.
9. `@jest/transform`: Jest'te kaynak dönüşüm (transpile) katmanı; test-üretim zincirinde önemli.
10. `call-bind`: Fonksiyon bağlaması için yardımcı; ağda düğüm yoğunluğu yüksek küçük kütüphanelerden.
11. `@babel/traverse`: AST üzerinde gezinme; kod dönüşüm süreçlerinde temel modüllerden.
12. `es-errors`: ECMAScript hata tipleri ve soyut işlemler için yardımcı; altyapı rolünde.
13. `jest-util`: Jest ekosisteminde çeşitli yardımcılar; test altyapısında kritik rol.
14. `@types/node`: Node.js için tip deklarasyonları; TypeScript ile yaygın standart bağımlılık.
15. `@babel/preset-env`: Hedef ortama göre derleme yapan Babel preset'i; yüksek yayılım riski.
16. `@babel/core`: Babel'in çekirdeği; derleme zincirinin üst katmanı, sistemik risk yüksektir.
17. `debug`: *Namespace*'li loglama; geçmişte tedarik zinciri saldırısı vakaları olmuştur.
18. `chalk`: Terminal stil kütüphanesi; popülerlik nedeniyle hedef olabilen modüllerden.
19. `@jest/core`: Jest'in çekirdeği; test sürecinin kalbinde.
20. `@babel/types`: AST düğümlerini oluşturma/denetleme; kod dönüşümünde temel işlevler.

8 Risk Skoru Tanımı

Normalize edilmiş metrikler (min–max) üzerinden ağırlıklandırılmış bir risk skoru tanımlanmıştır: $\text{risk}(n) = w_{in}\,\tilde{d}_{in}(n) + w_{out}\,\tilde{d}_{out}(n) + w_b\,\tilde{b}(n)$. Ağırlıklar (örn. $w_{in} = 0.5,\ w_{out} = 0.2,\ w_b = 0.3$) senaryoya göre ayarlanabilir.

9 Robustluk Analizi

Risk skoruna göre en kritik $k \in \{1, 3, 5\}$ düğüm çıkarıldığında zayıf bağlanırlık bileşen sayısı, en büyük bileşen boyutu ve (mümkünse) çap raporlanır (bkz. `robustness_risk.json`).

10 Sınırlamalar

Sınırlamalar: (i) Varsayılan olarak yalnızca `dependencies` kullanılır; `peerDependencies` isteğe bağlıdır. (ii) Betweenness hesabı büyük graflarda örneklemelidir; kesin değerlerin yakınsaması ağ büyüklüğüne ve k seçimine bağlıdır. (iii) Global *dependent* sayıları doğrudan dahil edilmemiştir.

11 Sonuç

Bu çalışma, NPM ekosistemindeki popüler paketler için yönlü bağımlılık ağını kullanarak yapısal riskleri nicel olarak analiz etmektedir. Merkeziyet metrikleri, bileşik risk skoru ve robustluk analizleri, kritik düğümlerin ve köprülerin pratikte nasıl belirleneceğine dair bir çerçeve sunar.

Çoğaltılabilirlik. Tüm kod ve çıktı üretimi depo içindedir. `analysis.ipynb` defteri çalıştırılarak `results/` dizini yeniden üretilebilir ve bu makale LaTeX dosyası ile raporlanabilir.

Kaynaklar

[1] Newman, M. (2010). *Networks: An Introduction*. Oxford University Press.

[2] Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*.

[3] Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.