NPM Tedarik Zincirinde Topolojik Risk Analizi ve Kritiklik Haritalaması
Topological Risk Analysis and Criticality Mapping in the NPM Supply Chain

Yusuf Talha ARABACI
Yazılım Mühendisliği Anabilim Dalı
Karabük Üniversitesi
Karabük, Türkiye
yusuftalhaarabaci@hotmail.com

***Özetçe*—Modern yazılım geliştirmenin omurgasını oluşturan NPM ekosistemi, tekil bir bağımlılıktaki güvenlik zafiyetinin veya kötü niyetli kod değişikliğinin, dolaylı bağlantılar üzerinden hızla yayılarak sistemik bir krize dönüşme potansiyelini taşır. Bu araştırma, paketlerin kod içeriğine odaklanan geleneksel yaklaşımların ötesine geçerek, paketler arası ilişkilerin *topolojik* izlerini sürer ve ekosistemin en kritik düğümlerini bu yapısal analizle saptar. Çalışmada, en fazla bağımlıya sahip 1000 paket üzerinden 7 kademe derinliğinde yönlü bir bağımlılık ağı inşa edilmiştir. Ağ üzerinde hesaplanan in-degree, out-degree ve betweenness merkeziyet ölçümleri, min-max normalizasyonu ile işlenerek *Bileşik Risk Skoru* (BRS) altında bütünleşik bir metriğe dönüştürülmüştür. Kritik düğümlerin ağdan çıkarılmasının en büyük bağlı bileşen (LCC) ve ağ erişilebilirliği üzerindeki yıkıcı etkisini ölçmek adına sağlamlık ve kaskad analizleri uygulanmıştır. Elde edilen bulgular, ağın iskeletini oluşturan az sayıda paketin sistemik riskin aslan payını yüklendiğini ve bu düğümlerin güvenliğinin tüm ekosistem için hayati bir zorunluluk olduğunu gözler önüne sermektedir.**

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

***Abstract*—In the NPM ecosystem, which lies at the heart of modern software development, a vulnerability or malicious alteration in a single dependency can propagate rapidly through the transitive dependency chain, impacting thousands of projects within a short timeframe. Unlike traditional approaches focusing on code content, this study aims to identify high-impact nodes by analyzing the *topological* patterns of inter-package relationships. A directed dependency graph was constructed using the Top 1000 packages selected based on the highest number of dependents, extending to a depth of 7 levels. In-degree, out-degree, and betweenness centralities calculated on this network were processed via min–max normalization and combined into a unified metric termed the *Composite Risk Score* (BRS). Robustness and cascade analyses were conducted to measure the impact of removing critical nodes on the largest connected component and reachability. The results indicate that a small number of backbone packages carry a significant portion of the systemic risk, highlighting that the security of these nodes is vital for the entire ecosystem.**

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

I. GİRİŞ

Yazılım geliştirme pratikleri, verimliliği maksimize etmek ve tekerleği yeniden icat etmekten kaçınmak adına NPM, PyPI ve RubyGems gibi açık kaynak paket yöneticilerine giderek daha fazla yaslanmaktadır. Bu modüler mimari, geliştirme süreçlerine ivme kazandırsa da, tedarik zincirindeki en ufak bir çatlağın sistemik bir krize evrilmesine zemin hazırlayan kırılgan bir yapı ortaya çıkarmıştır [1]. Milyonlarca paketi ve bunlar arasındaki girift bağımlılık ağını barındıran NPM ekosistemi, sunduğu geniş saldırı yüzeyiyle tehdit aktörleri için cazip bir hedef konumundadır. Literatürdeki bulgular, bu ağın "küçük dünya" (small-world) karakteristiği sergilediğini, dolayısıyla az sayıda bakımcının veya paketin ekosistem üzerinde orantısız bir etki gücüne sahip olduğunu doğrulamaktadır [16], [20], [25].

Tedarik zinciri saldırıları, güvenilir paketlerin ele geçirilmesinden isim benzerliği (typosquatting) tuzaklarına kadar geniş bir yelpazede tezahür eder. "Backstabber's Knife Collection" [4] ve "The Hitchhiker's Guide" [24] gibi kapsamlı derlemeler, saldırıların anatomisini kurulum ve çalışma zamanı (runtime) evreleri üzerinden irdelerken; Duan ve ark. [28], kayıt defteri (registry) düzeyindeki istismarlara odaklanarak yüzlerce kötü niyetli paketin varlığını belgelemiştir. Bu tehditlere karşı Amalfi [19], Cerebro ve OSCAR [29] gibi makine öğrenmesi ve dinamik analiz tabanlı savunma mekanizmaları geliştirilmiş olsa da, ekosistemin devasa ölçeği, her bir paketin derinlemesine taranmasını maliyet ve zaman açısından sürdürülemez kılmaktadır.

Mevcut literatür kritik düğümlerin varlığını teslim etmekle birlikte, bu düğümlerin sistematik tespiti ve *in-toto* [13] gibi güvenlik politikalarına entegrasyonu konusunda metodolojik bir boşluk barındırmaktadır. Analizler ekseriyetle popülerlik (indirme sayısı) veya statik kod analizi gibi metriklere hapsolmakta, ağ topolojisinden neşet eden yapısal riskleri bütüncül bir perspektifle değerlendirmekte yetersiz kalmaktadır. Bu çalışma, paket içeriklerinden soyutlanarak, yalnızca paketler arası bağımlılıkların topolojik mimarisine odaklanmakta ve yapısal riskin haritalandırılması için özgün bir metodoloji sunarak söz konusu boşluğu doldurmayı amaçlamaktadır.

Araştırmanın literatüre sunduğu temel katkılar şunlardır:

1) Resmî çözümleme kuralları çerçevesinde inşa edilen yönlü graf üzerinde in-degree, out-degree ve betweenness ölçümleri min–max yöntemiyle ölçeklenmiş; $w_{in} = 0.5$, $w_{out} = 0.2$, $w_{btw} = 0.3$ ağırlıklarıyla sentezlenerek **Bileşik Risk Skoru (BRS)** tanımlanmıştır.
2) BRS sıralaması, kaskad etki ve en büyük bağlı bileşen (LCC) analizleriyle ilişkilendirilerek, topolojik riskin sistemik yansımaları nicel verilerle ortaya konulmuştur.
3) Elde edilen bulgular, güvenlik analistleri ve paket yöneticileri için önceliklendirilmiş bir izleme listesi sunarak, kısıtlı güvenlik kaynaklarının nokta atışı kullanılabilmesine olanak tanımaktadır.

II. Veri ve Yöntem

A. Veri Setinin Oluşturulması ve Kapsam

NPM ekosisteminin risk haritasını çıkarmak amacıyla yürütülen ön çalışmalarda, "Most Downloaded" ve "Trending" gibi farklı örneklem stratejileri masaya yatırılmıştır. Ancak yapılan fizibilite analizleri, tedarik zinciri riskini en gerçekçi şekilde modelleyen yaklaşımın, en fazla sayıda projeye altyapı sağlayan paketlere odaklanmak olduğunu göstermiştir. Dolayısıyla analizin başlangıç noktası olarak, ekosistemin kılcal damarlarına kadar nüfuz etmiş (örneğin `lodash`, `chalk`, `express`) ilk 1000 paket seçilmiştir. Bu paketler, hiyerarşinin uç noktalarındaki son kullanıcı uygulamalarından ziyade, tüm sistemin üzerine inşa edildiği temel yapı taşlarını temsil eder.

NPM ekosistemi, en üstte uygulamaların yer aldığı ve aşağıya doğru kütüphanelere dallanan ters bir ağaç yapısını andırır; bu çalışmada seçilen örneklem ise sistemin yükünü sırtlayan merkezi katmana tekabül etmektedir. Analiz sürecinde, bu temel paketlerin `dependencies` listeleri izlenerek 7. dereceye kadar inilmiş ve tedarik zincirinin derinliği haritalandırılmıştır. Paketlerin, kendilerini kullanan üst katman projelerine dair doğrudan bir referans içermemesi, analizi yukarı yönlü (upstream) takip etmeyi güçleştirmektedir. Bu teknik kısıtlılığı aşmak adına, NPM API aracılığıyla her paketin toplam "bağımlı sayısı" (dependents) sorgulanmış ve bu metrik, örneklem seçiminde ana kriter olarak belirlenerek analizin ekosistemin en kritik katmanına odaklanması sağlanmıştır.

Yalnızca üretim bağımlılıkları (`dependencies`) filtreye tabi tutulmuş ve döngüsel referanslar ayıklanarak 1.506 düğüm ve 3.058 kenardan müteşekkil yönlü bir graf inşa edilmiştir. Ortaya çıkan bu graf, ekosistemdeki kritik düğümleri ve yapısal kırılganlıkları kristalize etmektedir. Grafik inşası ve analizi Python tabanlı NetworkX kütüphanesiyle gerçekleştirilmiş; veri setleri ve Gephi uyumlu çıktılar (`gephi_nodes.csv`, `gephi_edges.csv`) `results/` dizininde arşivlenmiştir.

B. Merkeziyet Ölçüleri ve Normalizasyon

Ağdaki her bir paketin önemini ve sistemik risk potansiyelini nicel olarak ifade edebilmek adına üç temel merkeziyet metriğine başvurulmuştur:

- **In-Degree:** Bir paketin popülaritesini ve doğrudan etki alanını tanımlayan bu metrik, o pakete doğrudan bağımlı olan diğer paketlerin sayısını ifade eder.
- **Out-Degree:** Bir paketin işlevselliğini sürdürebilmek için ihtiyaç duyduğu dış bağımlılıkların sayısıdır. Yüksek bir out-degree değeri, paketin saldırı yüzeyinin genişliğine ve dış kaynaklı risklere karşı kırılganlığına işaret eder.
- **Betweenness (Arasılık):** Bir paketin, ağdaki diğer düğümler arasındaki en kısa yollar üzerinde ne sıklıkla yer aldığını ölçer. Bu metrik, paketin ağ içindeki "köprü" rolünü ve bilgi/risk akışını denetleme kapasitesini yansıtır. Hesaplama maliyetini optimize etmek amacıyla $k = 20$ örneklemli bir yaklaşım benimsenmiştir.

Farklı ölçeklerdeki bu üç metriği ortak bir paydada buluşturmak için her biri min-max normalizasyonu ile [0,1] aralığına indirgenmiştir:

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

C. Bileşik Risk Skoru (BRS)

Normalizasyon sürecinden geçen değerler, her bir metriğin risk potansiyelini farklı boyutlarıyla yansıtması gözetilerek, ağırlıklı bir formülle bütünleşik bir skora tahvil edilmiştir:

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

Bu formülasyonda in-degree'ye en yüksek ağırlığın ($w_{in} = 0.5$) atfedilmesi, bir paketin tehlikeye girmesinin doğrudan etkileyeceği proje sayısının sistemik risk açısından taşıdığı kritik öneme dayanır. Ağdaki dolaylı yayılım riskini ve stratejik konumu temsil eden betweenness ise ikinci en yüksek ağırlıkla ($w_{btw} = 0.3$) modeldeki yerini almıştır.

D. Sağlamlık ve Kaskad Analizleri

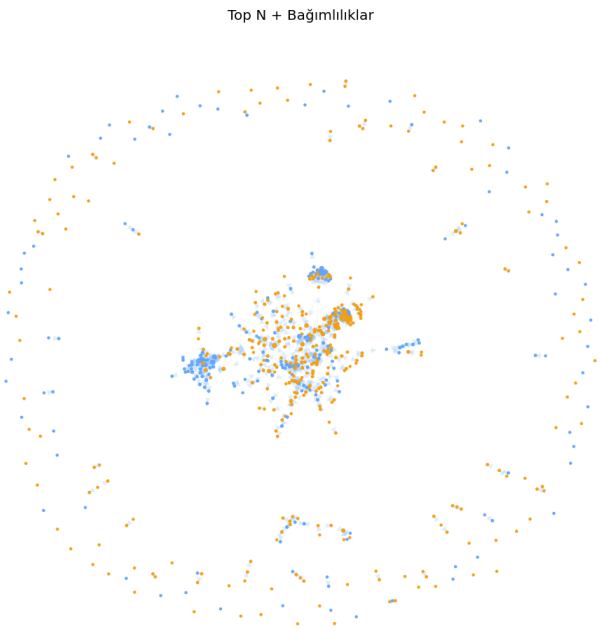
BRS metriğinin sistemik riski öngörme kapasitesini sınamak amacıyla "hedefli saldırı" simülasyonları kurgulanmıştır. Bu senaryolarda, en yüksek BRS skoruna sahip paketler ağdan kademeli olarak izole edilmiş; bu müdahalenin En Büyük Bağlı Bileşen (LCC) boyutuna, ortalama yol uzunluğuna ve ağın genel erişilebilirliğine (reachability) olan etkisi mercek altına alınmıştır. Buna ek olarak, her bir düğümün potansiyel etki alanını (transitif kapanış) ölçen bir kaskad analizi yürütülerek, BRS'nin öngörü gücü çapraz doğrulamaya tabi tutulmuştur.

III. Bulgular ve Değerlendirme

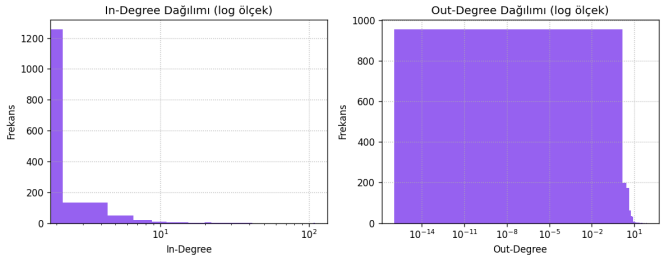
A. Ağ Topolojisi ve Derece Dağılımı

TABLO I. Ağ İstatistikleri

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



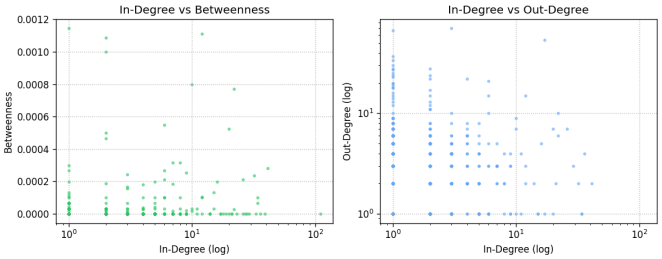
Şekil 1. Top 1000 paket ağının görselleştirmesi. Yoğun bölgeler, ekosistemin omurgasını oluşturan alt kümeleri işaret etmektedir.



Şekil 2. In-degree ve out-degree histogramları. Dağılımın ağır kuyruklu (heavy-tailed) yapısı, merkeziyetin az sayıda pakette toplandığını doğrulamaktadır.

Şekil 1'te sunulan ağ görselleştirmesi, yapının homojen bir dağılım yerine belirli odak noktalarında yoğunlaşan belirgin bir kümelenme (clustering) eğilimi taşıdığını göstermektedir. Derece dağılımlarının irdelendiği Şekil 2, bu yapısal karakteristiği daha somut verilerle destekler: Ağ, düğümlerin kahir ekseriyetinin sınırlı sayıda bağlantıya sahip olduğu, buna karşın "hub" niteliğindeki elit bir grubun yüzlerce bağlantıyı yönettiği, "ölçekten bağımsız" (scale-free) bir topoloji arz etmektedir. Dağılımın ağır kuyruklu (heavy-tailed) yapısı, merkeziyetin ve dolayısıyla riskin, az sayıda paketin tekelinde toplandığının açık bir göstergesidir.

B. Merkeziyet İlişkileri



Şekil 3. Merkeziyet ölçüleri arasındaki korelasyonlar.

Merkeziyet ölçümleri arasındaki korelasyonlar (Şekil 3) mercek altına alındığında, in-degree ile betweenness arasında pozitif yönlü ancak asimetrik bir ilişki göze çarpar. Bu bulgu, yüksek popülerliğe (in-degree) sahip olmayan bazı paketlerin, ağın izole kümelerini birbirine bağlayan kritik köprüler (yüksek betweenness) olarak stratejik bir misyon üstlenebildiğine işaret eder. Dolayısıyla, odağını yalnızca popülerliğe kilitleyen analizler, bu tür "gizli" kritiklikleri ıskalama riskiyle karşı karşıyadır.

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

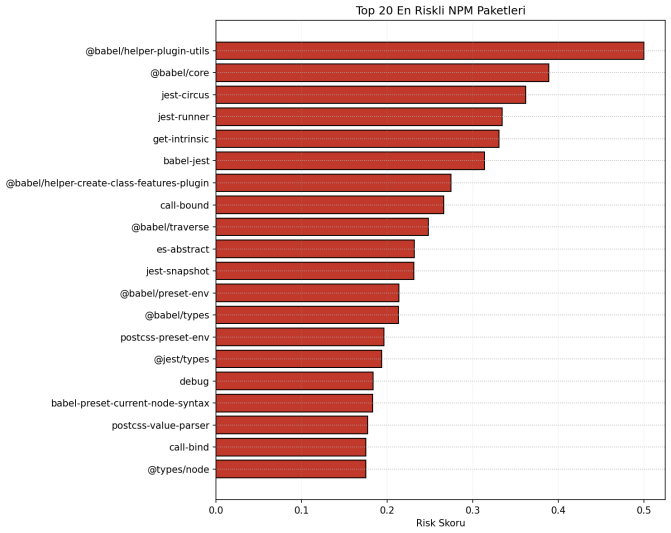
Paket	Out-Degree	In-Degree	Betweenness	TopN?
@babel/preset-env	70	3	0.000000	True
postcss-preset-env	67	1	0.000000	True
es-abstract	54	17	0.000000	False
react-scripts	48	0	0.000000	True
workbox-build	37	1	0.000000	True
eslint	34	1	0.000000	True
cssnano-preset-default	30	1	0.000000	True
webpack-dev-server	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
babel-plugin-istanbul	0.000466	2	5	True

Merkeziyet metriklerine dayalı sıralamalar, ekosistemdeki farklı kritiklik profillerini ayrıştırmamıza olanak tanır. Tablo II'de listelenen ve in-degree zirvesini tutan paketler, ekosistemin en popüler ve güven duyulan yapı taşlarını temsil eder. Öte yandan, Tablo III'de öne çıkan `@babel/preset-env` gibi paketler, sahip oldukları yoğun dış bağımlılıklar nedeniyle geniş bir saldırı yüzeyi sunmakta ve tedarik zincirinin alt katmanlarından sızabilecek tehditlere karşı kırılgan bir profil çizmektedir. Tablo IV ise, popülerlikten bağımsız olarak, ağdaki bilgi ve risk trafiğini yöneten `jest-circus` gibi stratejik darboğazları ifşa eder. Bu üç farklı perspektif, tek boyutlu analizlerin riskin bütüncül resmini çizmekte yetersiz kalacağını kanıtlar niteliktedir.

D. Bileşik Risk Skoru (BRS) Sıralaması



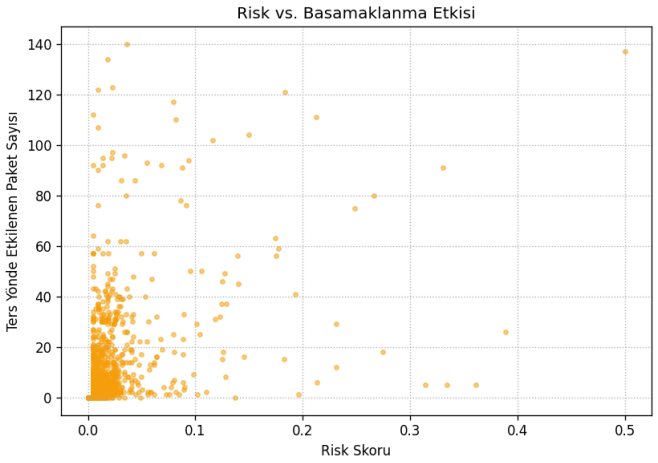
Şekil 4. En yüksek Bileşik Risk Skoruna (BRS) sahip 20 paket.

TABLO V. TOP 20 BILEŞIK RISK SKORU (BRS)

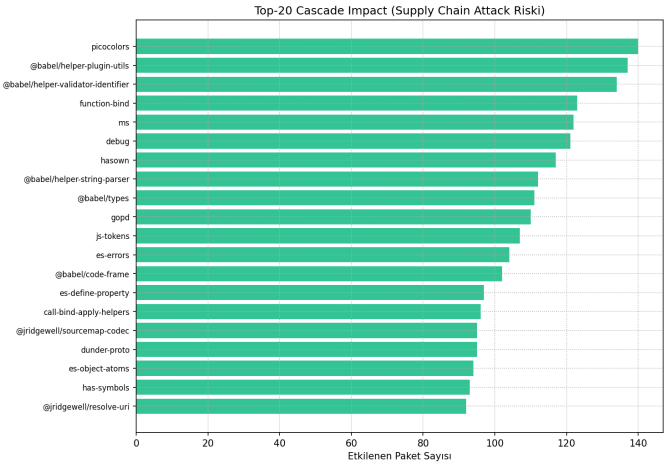
Paket	Risk	In-Degree	Out-Degree	Betweenness	TopN?
@babel/helper-plugin-utils	0.500000	110	0	0.000000	True
@babel/core	0.388827	12	15	0.001112	True
jest-circus	0.361688	1	20	0.001144	False
jest-runner	0.334157	2	22	0.001000	True
get-intrinsic	0.330606	22	10	0.000771	True
babel-jest	0.313975	2	7	0.001087	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

Geliştirilen BRS modeli, üç farklı risk boyutunu (popülerlik, bağımlılık yüzeyi, köprü rolü) tek bir potada eriterek, tekil metriklerin yakalayamadığı hibrit risk profillerini görünür kılar. Şekil 4 ve Tablo V'te izlendiği üzere, `@babel/core` ve `jest-runner` gibi paketler, hem yüksek popülariteleri hem de ağdaki stratejik konumları sebebiyle risk sıralamasının zirvesine yerleşmektedir. Bu hiyerarşi, güvenlik denetimleri ve kaynak tahsisinde önceliklendirilmesi elzem olan "yüksek değerli hedefleri" (high-value targets) tartışmaya yer bırakmayacak netlikte tanımlar.

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 yıkıcı etkisi.

BRS'nin sistemik riski öngörme kabiliyetini sınayan hedefli saldırı simülasyonları, modelin geçerliliğini çarpıcı bir biçimde ortaya koymaktadır. Şekil 6'te görüldüğü üzere, BRS zirvesindeki paketlerin ağdan koparılması, rastgele düğüm seçimine kıyasla En Büyük Bağlı Bileşen (LCC) boyutunda çok daha yıkıcı bir çöküşe neden olmakta ve ağın bütünlüğünü hızla parçalamaktadır. BRS skoru ile kaskad etki (bir düğümün kaybından etkilenen toplam paket sayısı) arasındaki güçlü pozitif korelasyon (Şekil 5) da bu tespiti perçinler. Bu veriler, BRS'nin yalnızca statik bir kritiklik ölçütü olmadığını, aynı zamanda bir düğümün ele geçirilmesinin tüm ekosistemde tetikleyebileceği zincirleme reaksiyonun da güvenilir bir habercisi olduğunu teyit etmektedir.

IV. TARTIŞMA VE SONUÇ

Bu araştırma, NPM ekosistemindeki sistemik riski, paket içeriğinden bağımsız olarak salt topolojik parametreler üze-

rinden modelleyen BRS (Bileşik Risk Skoru) metodolojisini literatüre kazandırmış ve geçerliliğini ampirik olarak kanıtlamıştır. Yürütülen analizler, ekosistem güvenliğinin, tehlikeye girmeleri halinde domino etkisiyle ağın bütünlüğünü çökertme potansiyeli taşıyan az sayıda "omurga" paketin sırtında yükseldiğini nicel verilerle doğrulamıştır.

Elde edilen bulgular ve önerilen model, teorik bir egzersiz olmanın ötesine geçerek, yazılım tedarik zinciri güvenliğini operasyonel düzeyde tahkim edecek somut stratejiler sunmaktadır:

- **Tespit Hatlarında Önceliklendirme:** Amalfi ve Cerebro gibi savunma mekanizmalarında tarama kaynaklarının, BRS skoru yüksek paketlere kanalize edilerek verimliliğin artırılması.
- **Politika Geliştirme:** in-toto gibi katı güvenlik protokollerinin ve dijital imza zorunluluğunun, öncelikle bu çalışmada saptanan kritik düğümlere uygulanarak riskin kaynağında boğulması.
- **Bakımcı Farkındalığı:** Kritik paket sahiplerine risk skorlarının bildirilmesi yoluyla, iki faktörlü kimlik doğrulama (2FA) ve daha titiz kod inceleme pratiklerinin benimsenmesinin teşvik edilmesi.

Gelecek çalışmaların rotası, geliştirilen modeli paketler arasındaki teknik bağların ötesine taşıyarak, geliştirici ağları (maintainer networks) ve sosyal bağımlılıklar gibi insan faktörlerini de kapsayacak şekilde genişletmektir. Ayrıca, ekosistemin zamana bağlı dinamiklerini mercek altına alan boylamsal (temporal) bir yaklaşım, riskin evrimini ve dönüşümünü anlamlandırmak adına paha biçilmez bir perspektif sunacaktır.

V. Yeniden Üretilebirlik

Çalışmanın şeffaflığını ve tekrarlanabilirliğini sağlamak adına tüm kaynak kodlar ve veri setleri erişime açıktır:

- **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] P. Jaisri, B. Reid, and R. G. Kula, "A preliminary study on self-contained libraries in the NPM ecosystem," 2024.

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

[4] M. Ohm et al., "Backstabber's knife collection: A review of open source software supply chain attacks," in *DIMVA*, 2020.

[5] I. Rahman et al., "Characterizing dependency update practice of NPM, PyPI and Cargo packages," 2024.

[6] T. G. Hastings, "Combating source poisoning and next-generation software supply chain attacks," 2024.

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

[8] C. Liu et al., "Demystifying vulnerability propagation via dependency trees in npm," in *ICSE*, 2022.

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

[10] A. J. Jafari et al., "Dependency practices for vulnerability mitigation," 2023.

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

[12] D. Y. K. Yip, "Empirical study on dependency-based attacks in Node.js," 2022.

[13] S. Torres-Arias, "In-toto: Practical software supply chain security," in *USENIX Sec.*, 2020.

[14] J. Zhang et al., "Malicious package detection in NPM and PyPI using a single model of malicious behavior sequence," 2023.

[15] S. Halder et al., "Malicious package detection using metadata information," 2024.

[16] A. Hafner, A. Mur, and J. Bernard, "Node package manager's dependency network robustness," 2021.

[17] P. Ladisa et al., "On the feasibility of cross-language detection of malicious packages in npm and PyPI," 2023.

[18] A. Zerouali et al., "On the impact of security vulnerabilities in the npm and RubyGems dependency networks," 2022.

[19] A. Sejfia and M. Schafer, "Practical automated detection of malicious npm packages (Amalfi)," in *ICSE*, 2022.

[20] M. Zimmermann et al., "Small world with high risks: Security threats in npm," in *USENIX Sec.*, 2019.

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

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

[23] M. Ohm et al., "Supporting detection via unsupervised signature generation (ACME)," 2021.

[24] P. Ladisa et al., "The hitchhiker's guide to malicious third-party dependencies," in *IEEE S&P*, 2023.

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

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

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

[28] R. Duan et al., "Towards measuring supply chain attacks on package managers," in *NDSS*, 2020.

[29] X. Zheng et al., "Towards robust detection of OSS supply chain poisoning (OSCAR)," 2024.

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